Intelligent Systems Reference Library 224

Angel D. Sappa *Editor*

ICT Applications for Smart Cities

Springer

Intelligent Systems Reference Library

Volume 224

The aim of this series is to publish a Reference Library, including novel advances and developments in all aspects of Intelligent Systems in an easily accessible and well structured form. The series includes reference works, handbooks, compendia, textbooks, well-structured monographs, dictionaries, and encyclopedias. It contains well integrated knowledge and current information in the field of Intelligent Systems. The series covers the theory, applications, and design methods of Intelligent Systems. Virtually all disciplines such as engineering, computer science, avionics, business, e-commerce, environment, healthcare, physics and life science are included. The list of topics spans all the areas of modern intelligent systems such as: Ambient intelligence, Computational intelligence, Social intelligence, Computational neuroscience, Artificial life, Virtual society, Cognitive systems, DNA and immunity-based systems, e-Learning and teaching, Human-centred computing and Machine ethics, Intelligent control, Intelligent data analysis, Knowledge-based paradigms, Knowledge management, Intelligent agents, Intelligent decision making, Intelligent network security, Interactive entertainment, Learning paradigms, Recommender systems, Robotics and Mechatronics including human-machine teaming, Self-organizing and adaptive systems, Soft computing including Neural systems, Fuzzy systems, Evolutionary computing and the Fusion of these paradigms, Perception and Vision, Web intelligence and Multimedia.

Indexed by SCOPUS, DBLP, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

Angel D. Sappa
Editor

ICT Applications for Smart Cities

Springer

Editor
Angel D. Sappa
Escuela Superior Politécnica del Litoral
Guayaquil, Ecuador

Computer Vision Center
Barcelona, Spain

ISSN 1868-4394 ISSN 1868-4408 (electronic)
Intelligent Systems Reference Library
ISBN 978-3-031-06306-0 ISBN 978-3-031-06307-7 (eBook)
https://doi.org/10.1007/978-3-031-06307-7

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

By 2050, it is predicted that 85% of the world's population would make their homes in cities. In the coming decades, urban centers are likely to face an increasing number of problems, some of which are linked to issues such as (a) monitoring and maintenance of urban heritage, (b) traffic planning in urban environments, (c) efficient use of energy and the use of renewable resources, (d) early warning systems, and (e) optimal management of resources in emergency situations, among others.

In recent years, different solutions to these problems have been suggested and raised under the umbrella of what has been called "Smart Cities". A Smart City can be described as a city that applies Information and Communication Technologies (ICTs) to provide it with an infrastructure that guarantees the following among others: (i) sustainable development, (ii) an increase in the quality of life of citizens, (iii) greater efficiency of available resources, and (iv) active citizen participation. Therefore, a Smart City is a socially, economically, and environmentally sustainable city, maintaining a balance between these aspects and always having the individual (the citizen) as the main beneficiary.

This book presents different successful case studies of ICTs in the application of Smart Cities. These prototypes were developed and evaluated in the framework of the Ibero-American Research Network TICs4CI funded by the CYTED program. This network started in 2018 and last for 5 years involving more than 50 researchers from eight research institutions hailing from seven Ibero-American countries. More specifically, the book describes underlying technologies and practical implementations of several applications developed in the following areas:

- Urban environment monitoring.
- Intelligent mobility.
- Waste recycling processes.
- Computer-aided diagnosis in healthcare systems.
- Computer vision-based approaches for efficiency in production processes.

Guayaquil, Ecuador — Angel D. Sappa
March 2022

Acknowledgements

This book would not have been possible without the collaboration and support of several people. First and foremost, I would like to thank all members of the TICs4CI Ibero-American Network and their institutions, who developed the different contributions and prototypes under the umbrella of Smart City applications. Research on these topics helped to find fruitful collaborations among the partners of the network and to contribute to the state of the art. Special thanks to Prof. Ángel Sanchez, from Universidad Rey Juan Carlos (Madrid, Spain), who motivated the setup of this network a couple of years before the collaboration among our partners started; after all these years, different applications, research collaborations, and mobility actions become a reality allowing the validation of models and prototypes relevant for Smart Cities. Special thanks as well to all the researchers and engineers from the different institutions who contributed with their work during all these years. Finally, I wish to thank the CYTED program for the financial support through the "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559). Without a doubt, this project is just the bridge built to drive further and stronger collaborations among the different partners of the TICs4CI network.


espol


universidade
de aveiro


UNIVERSIDAD DEL BÍO-BÍO


uff


TICs4CI

Contents

Chapter 1
Art Graffiti Detection in Urban Images Using Deep Learning

Tacio Souza Bomfim, Éldman de Oliveira Nunes, and Ángel Sánchez

Abstract Art graffiti can be considered as a type of urban street art which is actually present in most cities worldwide. Many artists who began as street artists have successfully moved to mainstream art, including art galleries. In consequence, the artistic graffiti produced by these authors became valuable works which are part of the cultural heritage in cities. When understanding the economic value of these public art initiatives within the smart cities context, the preservation of artistic graffiti (mainly, against vandalism) becomes essential. This fact will make it possible for municipal governments and urban planners to implement such graffiti maintenance initiatives in the future. In this context, this paper describes a deep learning-based methodology to accurately detect urban graffiti in complex images. The different graffiti varieties (i.e., 3D, stencil or wildstyle, among others) and the multiple variabilities present in these artistic elements on street scenes (such as partial occlusions or their reduced size) make this object detection problem challenging. Our experimental results using different datasets endorse the effectiveness of this proposal.

1.1 Introduction

The term *graffiti* refers to some types of writings or drawings made on a wall or other surfaces, usually without permission and within public view [5]. Graffiti ranges from simple written words to elaborate wall paintings. Nowadays, countless acts of vandalism graffiti are committed daily against public and private properties around the world. The costs caused by such damage are huge, and correspond to direct costs

T. S. Bomfim · É. de O. Nunes
UNIFACS, Salvador-BA 41720-200, Brazil
e-mail: tacio.tsb@gmail.com

É. de O. Nunes
e-mail: eldman.nunes@unifacs.br

Á. Sánchez (✉)
ETSII - URJC, 28933 Móstoles (Madrid), Spain
e-mail: angel.sanchez@urjc.es

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_1

(a) (b)

Fig. 1.1 Differences between graffiti types: **a** vandalism and **b** artistic

of cleaning surfaces (buildings, public transport vehicles, among others) and loss of value of properties repeatedly damaged by graffiti. For example, annual costs of graffiti removal in USA were estimated to be more than USD 12 billion in 2015.

In contrast to those graffiti that is considered as vandalism, other type of graffiti is becoming more widely recognized as a type of artwork [3]. Art graffiti is a form of visual communication created in public places that is legally produced and usually involves making usage of public spaces. Nowadays, graffiti artists such as Banksy have exhibited their graffiti-style paintings commercially in gallery and museum spaces. Figure 1.1 shows respective sample images of vandalism and art graffiti.

Since art graffiti is considered a form of outdoor cultural heritage to be preserved, municipal investments need to be dedicated to its conservation since this kind of art manifestation also produces an economic return [4]. The protection and dissemination of this kind of heritage in cities through the use of Information and Communication Technologies (ICTs) is currently an important aspect in the development of Smart Cities [9]. On the one hand, cultural tourism demands the development of applications that allow people with mobile phones to receive real-time information by taking a photograph. On the other hand, it is necessary to detect and guarantee the degree of conservation of this type of graffiti from captured images in order to be able to carry out actions for the recovery of these urban elements that could have suffered some kind of vandalism or deterioration [3, 10].

Art graffiti usually features color and technique variations, and it is aesthetically elegant, following an idea of the fine arts, different from vandalism graffiti. The artistic graffiti has a wide variety of categories [4], which use different techniques such as 3D, corresponding to those drawings made with a perspective and providing an effect of depth. They present an illusion very close to reality, making this technique much appreciated; the stencil that has become very popular, mainly by the artist Bansky. This art mode uses shapes made of cardboard, paper, metal, plastic, or other materials and sprays to create the design or text; the Piece, which is made by hand and contains at least three color and wildstyle that is based on letters, but in

a distorted and interconnected way, which makes it very difficult to understand this type of graffiti. Usually this technique also presents points, arrows and other types of elements. Wildstyle also has a strong connection with hip hop.

This paper describes a deep learning-based methodology to accurately detect urban art graffiti in complex images. As far as we know, the problem of artistic graffiti detection in images has not been researched previously using deep networks. The works found in the literature investigate only on vandalism graffiti problems, such as detection [16], image retrieval [17], quantification [14], and graffiti classification [11]. Most recent work on vandalism graffiti processing have applied some types of convolutional neural networks (CNN), such as VGG-16 or ResNet (see, for example, references [11] or [6]).

Our paper is organized as follows. Section 1.2 describes the object detection problem in general, and art graffiti detection as a particular case. Section 1.3 summarizes the different YOLO detection models considered in this study. Our dataset of images for the experiments is explained in Sect. 1.4. In Sect. 1.5, we describe the metrics, the experiments performed and analyze the corresponding results. Finally, Sect. 1.6 concludes this study.

1.2 Detection of Art Graffiti

Object detection is a challenging task in Computer Vision that has received large attention in last years, especially with the development of Deep Learning [15, 18]. It presents many applications related with video surveillance, automated vehicle system robot vision or machine inspection, among many others. The problem consists of recognizing and localizing some classes of objects present in a static image or in a video. Recognizing (or classifying) means determining the categories (from a given set of classes) of all object instances present in the scene together with their respective network confidence values on these detections. Localizing consists of returning the coordinates of each bounding box containing any considered object instance in the scene. The detection problem is different from (semantic) instance segmentation where the goal is identifying for each pixel of the image the object instance (for every considered type of object) to which the pixel belongs. Some difficulties in the object detection problem [18] include aspects such as geometrical variations like scale changes (e.g., small size ratio between the object and the image containing it) and rotations of the objects (e.g., due to scene perspective the objects may not appear as frontal); partial occlusion of objects by other elements in the scene; illumination conditions (i.e., changes due to weather conditions, natural or artificial light); among others but not limited to these ones. Note that some images may contain several combined variabilities (e.g., small, rotated and partially occluded objects). In addition to detection accuracy, another important aspect to consider is how to speed up the detection task.

Detecting art graffiti in images can be considered as an object detection problem with the multiple variabilities above indicated. Art graffiti images are, in general,

much more diverse and elaborated as those including vandalism graffiti (as shown by Fig. 1.1), and also contain much more colors and textures. Another interesting aspect that can also difficult the detection of graffiti (both vandalism and art ones) in images is the varying nature of covered materials, i.e., the surfaces where graffiti are painted [1], for example stone walls, wooden fences, metal boxes or glass windows, among others.

1.3 Object Detection Deep Architectures

This section outlines object detection deep architectures, in special the YOLO model and its variants that were applied in this work.

1.3.1 YOLO Models

Redmon and collaborators proposed in 2015 [13] the new object detector model called YOLO (acronym of "You Only Look Once"), which handles the object detection as a one-stage regression problem by taking an input image and learning simultaneously the class probabilities and the bounding box object coordinates. This first version of YOLO was also called YOLOv1, and since them the successive improved versions of this architecture (YOLOv2, YOLOv3, YOLOv4, and YOLOv5, respectively) have gained much popularity within the Computer Vision community.

Different from previous two-stage detection networks, like R-CNN and faster R-CNN, the YOLO model used only one-stage detection. That is, it can make predictions with only one "pass" in the network. This feature made the YOLO architecture extremely fast, at least 1000 times faster than R-CNN and 100 times faster than Fast R-CNN.

The architecture of all YOLO models have some similar components which are summarized next:

- *Backbone*: A convolutional neural network that accumulates and produces visual features with different shapes and sizes. Classification models like ResNet, VGG, and EfficientNet are used as feature extractors.
- *Neck*: This component consists of a set of layers that receive the output features extracted by the Backbone (at different resolutions), and integrate and blend these characteristics before passing them on to the prediction layer. For example, models like Feature Pyramid Networks (FPN) or Path Aggregation networks (PAN) have been used for such purpose.
- *Head*: This component takes in features from the Neck along with the bounding box predictions. It performs the classification along with regression on the features and produces the bounding box coordinates to complete the detection

process. Generally, it produces four output values per detection: the x and y center coordinates, and width and height of detected object, respectively.

In the next subsections, we summarize the main specific features of the three YOLO architectures used in our experiments: YOLOv4, YOLOv5 and YOLOv4-tiny, respectively.

1.3.2 YOLOv4

YOLOV4 was released by Alexey Bochkovskiy et al. in their 2020 paper "YOLOv4: Optimal Speed and Accuracy of Object Detection" [2]. This model is ahead in performance on other convolutional detection models like EfficientNet and ResNext50. Like YOLOv3, it has the Darknet53 model as Backbone component. It has a speed of 62 frames per second with an mAP (mean Average Precision) of 43.5 percent on the Microsoft COCO dataset.

As technical improvements with respect to YOLOv3, YOLOv4 introduces as new elements the bag of freebies and the bag of specials.

Bag of freebies (BOF) are a set of techniques enabling an improvement of the model in performance without increasing the inference cost. In particular:

- *Data augmentation techniques*: CutMix, MixUp, CutOut, …
- *Bounding box regression loss types*: MSE, IoU, CIoU, DIoU, …
- *Regularization techniques*: Dropout, DropPath, DropBlock, …
- *Normalization techniques*: Mini-batch, Iteration-batch, GPU normalization, …

Bag of specials (BOS) consist in techniques that increase accuracy while increasing the inference computation cost by only a small amount. In particular:

- *Spatial attention modules (SAM)*: Spatial Attention (SA), Channel-wise Attention (CA), …
- *Non-max suppression modules (NMS)*
- *Non-linear activation functions*: ReLU, SELU, Leaky, Mish, …
- *Skip-Connections*: Weighted Residual Connections (WRC), Cross-Stage Partial connections (CSP), …

Figure 1.2 illustrates the layer structure of YOLOv4 network used in our experiments.

1.3.3 YOLOv5

One month after the release of YOLOv4, the version 5 of this model, created by Glenn Jocher, was published. This novelty caused a series of discussions in the scientific

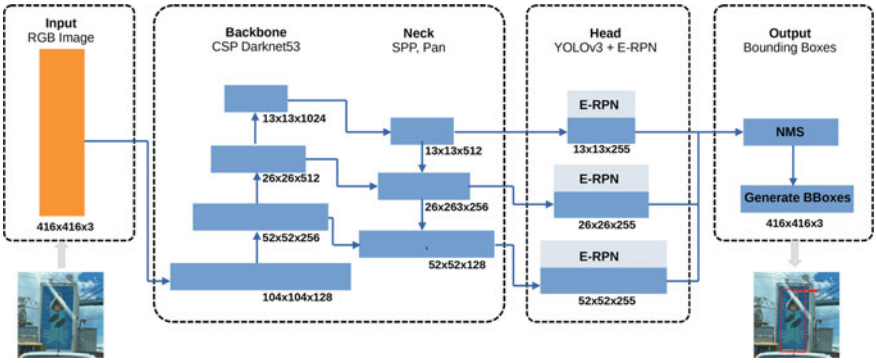

Fig. 1.2 Schematic representation of YOLOv4 architecture

community, first for not having been developed by the original author of the YOLO network, then for not having published a release paper [8].

YOLOv5 uses the PyTorch framework. This version now uses CSPDarknet53 as the Backbone, only the PANet as the Neck and the YOLO layer as the Head, the same as previous versions. An innovation of the YOLOv5 network is the self-learning of bounding box anchors. YOLOv5 achieves the same if not better accuracy (mAP of 55.6) than YOLOv4 model while taking less computation power.

Some of the technical improvements of YOLOv5 over the previous version of this architecture are the following ones: an easier framework to train and test (PyTorch); a better data augmentation and loss calculations (using PyTorch framework); auto-learning of anchor boxes (do not need to be added manually now); use of cross-stage partial connections (CSP) in the backbone; use of path aggregation network (PAN) in the neck of the model; support of YAML files, which greatly enhances the layout and readability of model configuration files; among other advantages.

1.3.4 YOLOv4-tiny

YOLOv4-tiny is the compressed version of YOLOv4 designed to train on machines that have less computing power [7]. The model weights are around 16 megabytes large, allowing it to train on 350 images in 1 h when using a Tesla P100 GPU. YOLOv4-tiny has an inference speed of 3 ms on the Tesla P100, making it one of the fastest object detection models to exist.

YOLOv4-tiny utilizes a couple of different changes from the original YOLOv4 network to help it achieve these fast speeds. First and foremost, the number of convolutional layers in the CSP backbone are compressed with a total of 29 pretrained convolutional layers. Additionally, the number of YOLO layers has been reduced to two instead of three, and there are fewer anchor boxes for prediction.

YOLOv4-Tiny has comparatively competitive results with YOLOv4 given the size reduction. It achieves 40% mAP @.5 on the MS COCO dataset.

1.4 Dataset

There is a lack of available public labelled datasets of art graffiti images. Therefore, to perform the experiments of this study, we created our own dataset of annotated graffiti images.

The database used in this project was created with images of own authorship and also from sites with free copyright, such as unsplash (https://unsplash.com/), pixebay (https://pixabay.com/pt/), pexels (https://www.pexels.com/pt-br/) and Google maps. Table 1.1 shows the distribution of the set of images in terms of their source. A total of 522 images were collected, containing 603 annotations of artistic graffiti.

An analysis carried out in the created image dataset concerns to the size ratios of the annotated graffiti in relation to the corresponding image sizes. This information is relevant to be considered in the separation of training, validation and testing bases in a uniform way. In addition, graffiti with small sizes are, in most cases, more difficult to detect and, therefore, when evaluating the results, this information is quite useful. Table 1.2 describes the amount of graffiti according to their size ratios in relation to the image that contains them.

The number of art graffiti with a proportion smaller than 20%, is approximately a half of the total. These detections tend to be more challenging for neural networks, because the localization of small images needs a much higher precision [15]. Figure 1.3 shows two images with respective graffiti sizes less than 10% and more than 80% of the image size.

Table 1.1 Distribution of images in our dataset

Source	No. images	No. annotations
Own authorship	127	164
Other sources	395	439

Table 1.2 Distribution of graffiti Sizes with respect to image sizes

Ratio size	Quantity
Smaller than 10%	177
Between 10% and 20%	127
Between 20% and 30%	91
Between 30% and 40%	68
Between 40% and 50%	48
Larger than 50%	92

Fig. 1.3 Two examples of graffiti sizes: **a** less than 10% and **b** more than 80% of the image size

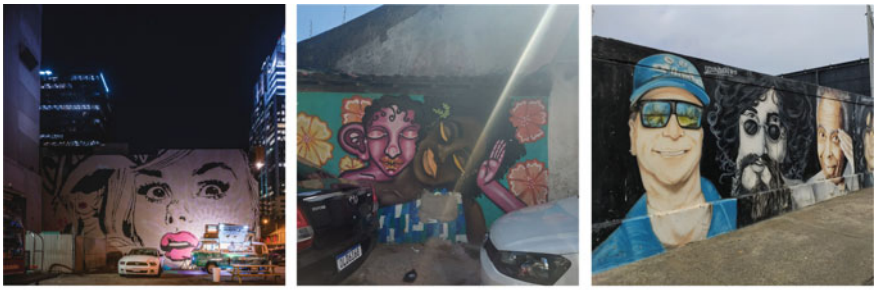

Fig. 1.4 Examples of variabilities present in the graffiti dataset: **a** low-contrast, **b** partial occlusion, and **c** perspective

Additionally, our graffiti dataset present other variabilities as shown in Fig. 1.4. These include large brightness differences (Fig. 1.4a), since in a real situation, the system must be able to detect graffiti at night or in low light, cloudy weather, rain, shadow, among other climatic variations; partial occlusions (Fig. 1.4b); and image perspectives or inclinations (Fig. 1.4c).

The image dataset was distributed into three groups: training (with 76% of images), testing (16%) and validation (8%), respectively. Table 1.3 shows the distribution of images, according to considered variabilities, in these groups.

Table 1.3 Distribution of the image dataset in training, validation and test sets

Dataset	Images	Skewed	Occluded	Highly-contrasted
Train	395	88	66	43
Validation	42	9	8	2
Test	85	25	29	6

1.5 Experiments

This section describes the evaluation metrics employed, presents the different tests performed using different YOLO architectures, and shows the respective quantitative and qualitative results.

1.5.1 Evaluation Metrics and Computing Resources

To measure and compare the results of each experiment, the following evaluation metrics were calculated: Precision (P), Recall (R) and F1 score (F1), as follows:

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad F1 = 2\frac{P \times R}{P + R} \tag{1.1}$$

For all experiments performed, we also computed the Average Precision for the detected class of "Art Graffiti" objects, using 11-point interpolation [12], and represented it using Precision-Recall curves. In each of the tests, the performances were analyzed under different aspects, such as, for example, the size of the graffiti in relation with the image size, its inclination, variation of luminosity and occlusion, respectively.

It is important to point out that all the training was carried out using the Google Colab virtual environment, which provides for free a cloud GPU processing service. All detections in images were performed on a specific GPU, with lower capacity, to standardize detection times. The GPU to perform the detection has the following configuration:

- Manufacturer: Nvidia
- Model: GeForce MX110
- Capacity: 2 GB
- Processor: Intel(R) Core(TM) i5-10210U

Table 1.4 Confusion matrix corresponding to Test 1

Actual	Predicted	
	Graffiti	No Graffiti
Graffiti	79	17
No Graffiti	4	4

1.5.2 Experiments

Network training was carried out using three different input sizes (i.e. image resolutions): 416 × 416 (which is the standard one for this network), 512 × 512 and 608 × 608, respectively. Additional tests were also executed by first increasing the sharpness (i.e., the contrast) of images a 50% previous to training the YOLOv4 detector. We also have performed some tests with YOLOv5 and with the YOLOv4-tiny network (suitable for limited computing resources), both with the smaller 416 × 416 image resolution. The number of test graffiti objects in images was 96 for all the experiments.

For the four first experiments (i.e., using the YOLOv4 network), the following network hyperparameters values were used:

- Batch size: 64
- Subdivision: 64
- Momentum: 0.9
- Decay: 0.0005
- Learning rate: 0.001
- Max Batches: 3000

1.5.2.1 Test 1: Image Resolution 416 × 416 (YOLOv4)

This test was performed on the basis of the test images being resized to a spatial resolution of 416 × 416 pixels and keeping their aspect ratios. The main goal of this test was to train the network with the default settings in order to verify the detection accuracy for test images and compare the results with those from the remaining experiments. Table 1.4 shows the confusion matrix of the results. The average detection time per test image was 527 ms.

Respective values of Precision, Recall and F1-Score for this experiment were: 95.18, 82.29 and 88.27%. Finally, the Precision-Recall curve corresponding to this experiment is shown by Fig. 1.5 where the mAP (mean Average Precision) is calculated in all experiments using the 11-point interpolation technique.

Fig. 1.5 Precision-Recall curve of Test1 1

Table 1.5 Confusion matrix corresponding to Test 2

Actual	Predicted	
	Graffiti	No Graffiti
Graffiti	66	30
No Graffiti	8	7

1.5.2.2 Test 2: Image Resolution 512 × 512 (YOLOv4)

This test was performed on the basis of the test images being resized to a spatial resolution of 512 × 512 pixels and keeping their aspect ratios. The main goal of this test and Test 3 was to train the network with images of higher spatial resolutions than in the initial experiment, and to perform comparisons using the considered detection metrics. Table 1.5 shows the confusion matrix with the results of this test. The average detection time per test image was 772 ms.

Respective values of Precision, Recall and F1-Score for this experiment were: 89.18, 68.75 and 77.64%. Finally, the Precision-Recall curve corresponding to this experiment is shown by Fig. 1.6.

1.5.2.3 Test 3: Image Resolution 608 × 608 (YOLOv4)

This test was performed on the basis of the test images being resized to a higher spatial resolution than in previous test (608 × 608 pixels) and keeping their aspect

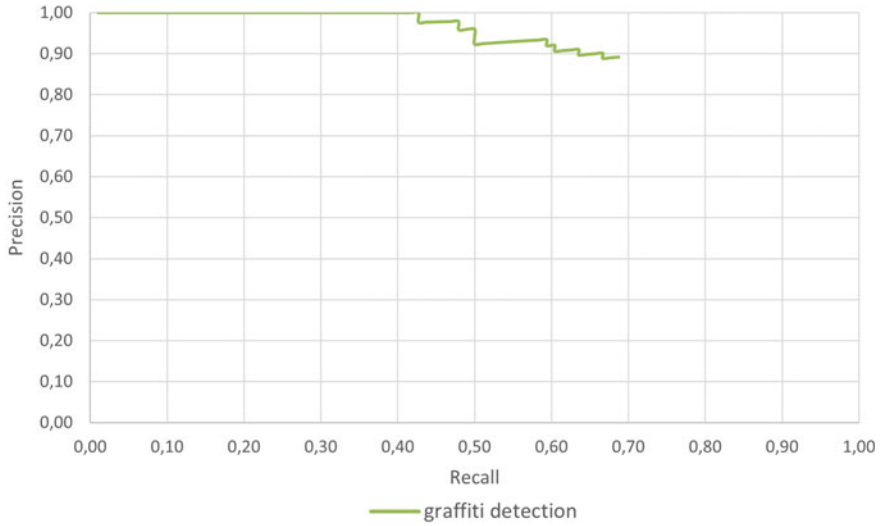

Fig. 1.6 Precision-Recall curve of Test 2

Table 1.6 Confusion matrix corresponding to Test 3

Actual	Predicted	
	Graffiti	No Graffiti
Graffiti	69	27
No Graffiti	12	8

ratios. Table 1.6 shows the confusion matrix with the results of this test. The average detection time per test image was 1.553 ms.

Respective values of Precision, Recall and F1-Score for this experiment were: 85.18, 71.88 and 77.97%. Finally, the Precision-Recall curve corresponding to this experiment is shown by Fig. 1.7.

1.5.2.4 Test 4: Image Resolution 416 × 416 with Sharpening (YOLOv4)

In this experiment, an enhancement pre-processing was performed to images in order to increase their image contrast by a factor of 1.5 (i.e, increasing of a 50%). This sharpening transformation was applied to the training and validation images after these were resized to 416 × 416 by keeping their aspect ratio. The goal of this test is to verify whether or not increasing image sharpness produces more accurate detection results, since in some images, the illumination conditions or the graffiti drawings themselves can worsen the results.

Fig. 1.7 Precision-Recall curve of Test 3

Table 1.7 Confusion matrix corresponding to Test 4

Actual	Predicted	
	Graffiti	No Graffiti
Graffiti	83	13
No Graffiti	5	2

Table 1.7 shows the confusion matrix with the results of this test. The average detection time per test image was 386 ms.

Respective values of Precision, Recall and F1-Score for this experiment were: 94.32, 86.46 and 90.22%. Finally, the Precision-Recall curve corresponding to this experiment is shown by Fig. 1.8.

1.5.2.5 Test 5: Image Resolution 416 × 416 with Sharpening (YOLOv5)

This test was performed using the images properly resized to the original resolution of 416 × 416 pixels but using the YOLOv5 network with the following configuration hyperparameters:

- Batch size: 64
- Momentum: 0.937
- Decay: 0.0005
- Learning rate: 0.001
- Max Batches: 3000

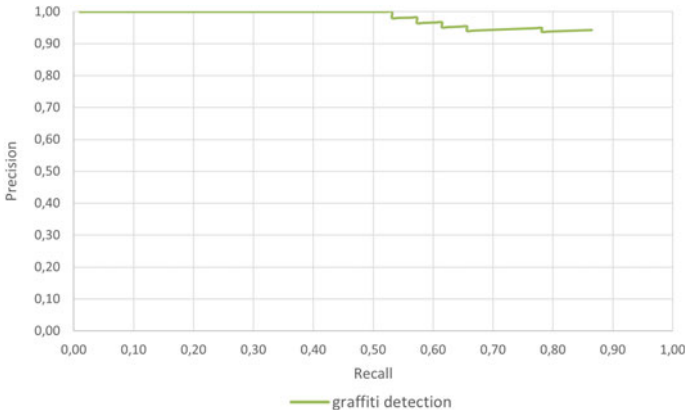

Fig. 1.8 Precision-Recall curve of Test 4

Table 1.8 Confusion matrix corresponding to Test 5

Actual	Predicted	
	Graffiti	No Graffiti
Graffiti	78	18
No Graffiti	15	39

The purpose of this experiment is compare the results produced by YOLOv4 and YOLOv5 models on the same test images.

Table 1.8 shows the confusion matrix with the results of this test. The average detection time per test image was 386 ms.

Respective values of Precision, Recall and F1-Score for this experiment were: 83.87, 81.25 and 82.54%. Finally, the Precision-Recall curve corresponding to this experiment is shown by Fig. 1.9.

1.5.2.6 Test 6: Image Resolution 416 × 416 (YOLOv4-tiny)

The purpose of this is to analyze the performance of this model with more limited resources in comparison with the other considered architectures. The test was performed on the basis of the test images being resized to a spatial resolution of 416×416 pixels and keeping their aspect ratios. The values of training hyperparameters for this model were the following ones:

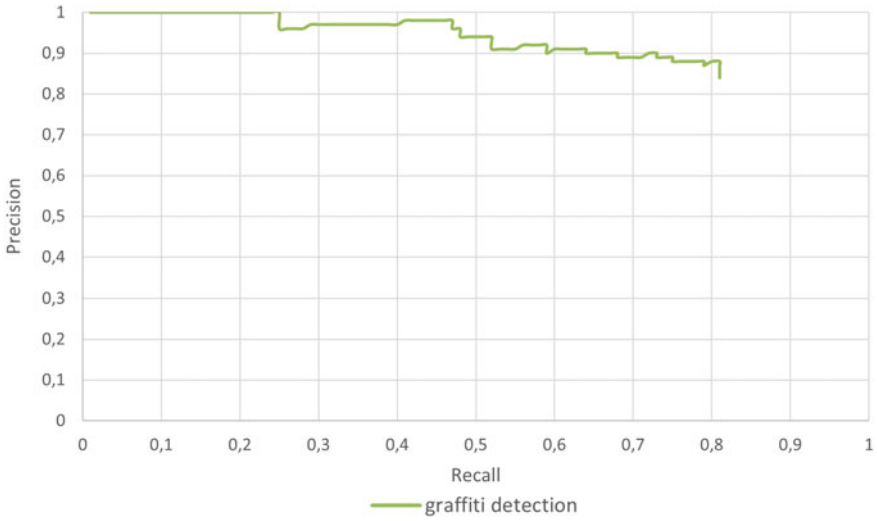

Fig. 1.9 Precision-Recall curve of Test 5

Table 1.9 Confusion matrix corresponding to Test 6

Actual	Predicted	
	Graffiti	No Graffiti
Graffiti	50	46
No Graffiti	10	3

- Batch size: 64
- Subdivision: 64
- Momentum: 0.9
- Decay: 0.0005
- Learning rate: 0.00261
- Max Batches: 3000

Table 1.9 shows the confusion matrix of the results. The average detection time per test image was 51 ms.

Respective values of Precision, Recall and F1-Score for this experiment were: 83.33, 52.08 and 64.10%. Finally, the Precision-Recall curve corresponding to this experiment is shown by Fig. 1.10 where the mAP (mean Average Precision) is calculated in all experiments using the 11-point interpolation technique.

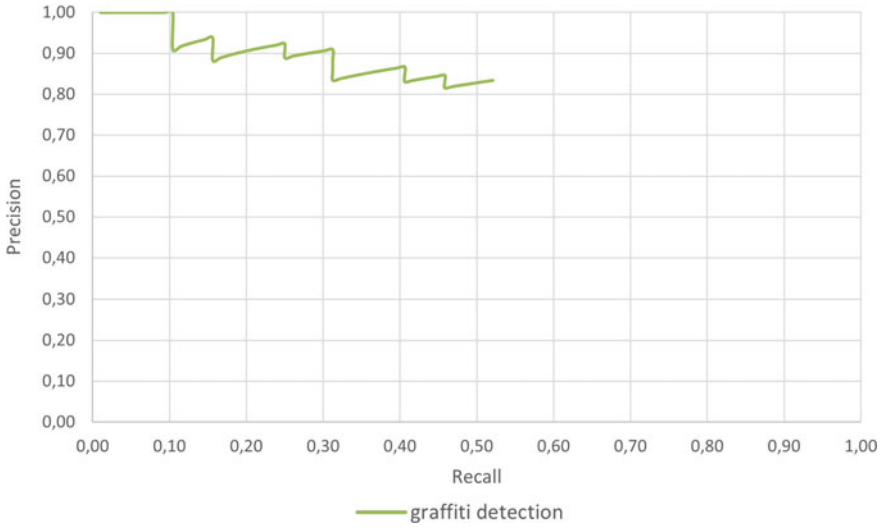

Fig. 1.10 Precision-Recall curve of Test 6

1.5.2.7 Analysis of Results

Among all the tests performed, the one that obtained the highest global score (i.e., F1-Score) was Test 4, that is, input size 416×416 in YOLOv4, applying sharpening to the training and validation images, according Fig. 1.11 and Table 1.10. If the evolution measures are evaluated separately, Test 1 (input dimension 416) obtained the highest Precision, while Test 4 (sharpening application) obtained the highest Recall. It was possible to verify the application of sharpness improved the detection mainly of cases in which the artistic graffiti was more faded or with few colors.

Regarding the test using YOLOv5, it is possible to verify that it obtained a worse result than YOLOv4, when compared to Test 1, which had the same network input pattern (416 × 416). It is important to point out that to have a conclusion about which network would have the best performance, it will be necessary to carry out more tests.

Test 6, with YOLOv4-tiny for mobile applications, had a low recall when compared to the other tests; however it obtained an accuracy very close to all tests. As previously mentioned, this model is an option for real-time object detection in videos on mobile and embedded devices.

Finally, we also present some qualitative results corresponding to a randomly chosen test image. Figure 1.12 shows for this same image the visualisation results of Intersection over Union (IoU) corresponding to the respective six experiments carried out. In order to clarify these visual results, we also show in Table 1.11 the respective confidence values returned by the network and also the corresponding IoU results. It can be noticed that best confidence value corresponds to Test 1 working with a lower image resolution (416 × 416), and best IoU result corresponds to Test 5 using YOLOv5 also with the lower resolution.

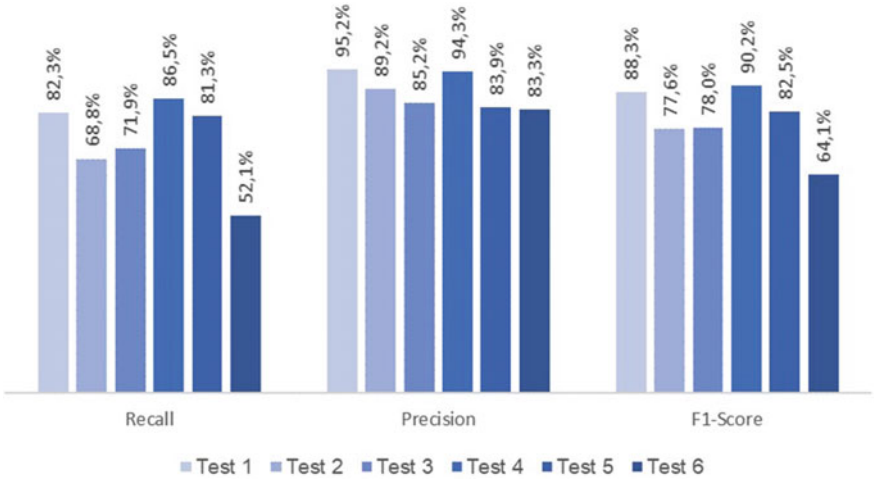

Fig. 1.11 Comparison of metrics values between tests

Table 1.10 Comparative test results

Test	Architecture	Resolution	Test time (ms)	Recall	Precision	F1-Score
Test 1	YOLOv4	416	527	82.3	95.2	88.3
Test 2	YOLOv4	512	772	68.8	89.2	77.6
Test 3	YOLOv4	608	1,553	71.9	85.2	78.0
Test 4	YOLOv4	416	386	86.5	94.3	90.2
Test 5	YOLOv5	416	386	81.3	83.9	82.5
Test 6	YOLOv4-tiny	416	51	52.1	83.3	64.1

The reported poor results for the YOLOv4-tiny model (Test 6) in Table 1.11 correspond to its execution with the same computer resources as previous tests. Note that YOLOv4-tiny can also be executed in a smartphone and also work for videos at real-time. We achieved much better results for the Confidence and IoU metrics (respectively, 0.93 and 0.84) using a smartphone Xiaomi 9T model (with camera 48 Mpx, video 4K, 8 core and display 6.39). For a better interpretation of the previous comparative results between experiments it is important to remark that YOLOv4, YOLOv5 and YOLOv4-tiny architectures have respectively the following approximate number of training parameters: 6×10^7, 4.6×10^7 and 6×10^6.

Fig. 1.12 Qualitative detections for each of the six experiments on the same test image

Table 1.11 Respective network confidence and IoU values for the six experiments computed on the same test image shown in Fig. 1.11

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Confidence	**0.88**	0.64	0.49	0.87	0.77	0.70
IoU	0.76	0.85	0.84	0.75	**0.92**	0.45

1.6 Conclusion

Object detection is a task that has received a lot of attention in recent years, especially with the development of extremely fast networks, arising from deep learning and traditional convolutional neural networks. This task consists of locating and classifying objects in static or dynamic images (videos). The location is done through the coordinates of a bounding box that contains at least one instance of the object, and the classification consists of producing a confidence value for a certain category of objects.

This work dealt with the detection in static images of art graffiti objects, which are artistic manifestations considered as popular art. These paintings have different techniques and styles, having a great potential to beautify cities, in addition to becoming local tourist spots. Another interesting point is that this form of expression is legal, unlike vandalism graffiti. To carry out the detections of art graffiti, three YOLO network models were used (YOLOv4, YOLOv5 and YOLOv4-tiny, respectively).

Six tests were performed, the first three using YOLOv4 at image dimensions of 416×416, 512×512 and 608×608, respectively. Test 4 also used YOLOv4 at resolution 416×416 but applying a sharpening preprocessing to the training and validation images. Tests 5 and 6 used YOLOv5 and YOLOv4-tiny, respectively. To carry out the training, validation and testing, we used a database created with images of our own authorship and also from sites with free copyright, with a total of 522 images, containing 603 annotations of art graffiti objects.

Among all the tests performed, the one with the highest global score (F1-Score) was Test 4, that is 416×416 image resolutions with YOLOv4 and applying sharpness to the training and validation images. This test has an accuracy of 94.3% and a recall of 86.5%. The test with YOLOv5 obtained good evaluation metrics, but it was inferior to YOLOv4. YOLOv4-tiny also achieved a close accuracy to all tests, however it had a low recall.

As future work we consider the application of YOLOv4-tiny to mobile devices, using also the geolocation of images, in order to generate relevant information (i.e., for art graffiti maintenance purposes). For this task, it will be necessary to improve the accuracy of this reduced detector. Another future work will consist in extending our detection problem considering together art, vandalism and other variants of graffiti classes (i.e., a multiclass detection problem).

Acknowledgements We acknowledge to the Spanish Ministry of Science and Innovation, under RETOS Programme, with Grant No.: RTI2018-098019-B-I00; and also to the CYTED Network "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559).

References

1. Alfarrarjeh, A., Trivedi, D., Kim, S.H., Park, H., Huang, C., Shahabi, C.: Recognizing material of a covered object: a case study with graffiti. In: 2019 IEEE International Conference on Image Processing (ICIP), pp. 2491–2495 (2019)
2. Bochkovskiy, A., Wang, C.-Y., Liao, H.-y.: Yolov4: optimal speed and accuracy of object detection (2020)
3. Collins, A.: Graffiti: Vandalism or Art? Greenhaven Publishing LLC (2017)
4. Forte, F., Paola, D.: Pierfrancesco: how can street art have economic value? Sustainability **11**(3), 580 (2019)
5. Gómez, M.A.: The writing on our walls: finding solutions through distinguishing graffiti art from graffiti vandalism. U. Mich. JL Reform **26**, 633 (1992)
6. Hatir, M.E., Barstuğan, M., Ince, I.: Deep learning-based weathering type recognition in historical stone monuments. J. Cult. Herit. **45**, 193–203 (2020)
7. Jiang, Z., Zhao, L., Li, S., Jia, Y.: Real-time object detection method based on improved yolov4-tiny (2020)
8. Karthi, M., Muthulakshmi, V., Priscilla, R., Praveen, P., Vanisri, K.: Evolution of yolo-v5 algorithm for object detection: automated detection of library books and performace validation of dataset, pp. 1–6 (2021)
9. Khatoun, R., Zeadally, S.: Smart cities: concepts, architectures, research opportunities. Commun. ACM **59**(8), 46–57 (2016)
10. Merrill, S.O.C.: Graffiti at heritage places: vandalism as cultural significance or conservation sacrilege? Time Mind **4**(1), 59–75 (2011)

11. Munsberg, G.R., Ballester, P., Birck, M.F., Correa, U.B., Andersson, V.O., Araujo, R.M.: Towards graffiti classification in weakly labeled images using convolutional neural networks. In: Latin American Workshop on Computational Neuroscience, pp. 39–48. Springer (2017)
12. Padilla, R., Netto, S.L., da Silva, E.A.B.: A survey on performance metrics for object-detection algorithms, pp. 237–242 (2020)
13. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection, pp. 779–788 (2016)
14. Tokuda, E.K., Cesar, R.M., Silva, C.T.: Quantifying the presence of graffiti in urban environments. In: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 1–4. IEEE (2019)
15. Tong, K., Wu, Y., Zhou, F.: Recent advances in small object detection based on deep learning: a review. Image Vis. Comput. **97**, 103910 (2020)
16. Wang, J., Zhijie, X., O'Grady, M.: Head curve matching and graffiti detection. Int. J. Comput. Vis. **14**, 9–14 (2010)
17. Yang, C., Wong, P.C., Ribarsky, W., Fan, J.: Efficient graffiti image retrieval. In: Proceedings of the 2nd ACM International Conference on Multimedia Retrieval, pp. 1–8 (2012)
18. Zou, Z., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: a survey (2019). arXiv:1905.05055

Chapter 2
Deep Neural Networks for Passengers' Density Estimation and Face Mask Detection for COVID-19 in Public Transportation Services

Rogelio Hasimoto-Beltran, Odin F. Eufracio-Vazquez, and Berenice Calderon-Damian

Abstract Public Transportation Networks (PTN) are considered as one of the essential commuting systems worldwide. Under good planification strategies, PTN improves population mobility, reduces environmental pollution, traffic congestion, and commuting time. Optimal management, control, and tracking of public transportation systems are desirable for improving service quality and users' wellness. This work proposes a management and control system for public transportation based on passengers' density estimation. Our final goal is to develop a communication network architecture and a Deep Neural Network for face detection to quantify the density and number of people being transported as a function of time. This collected information (density and number of passengers) is sent to public transport administrators (municipal authorities and bus owners) to take actions for adding/removing busses to/from specific route circuits and improve the users' commuting quality of service (that is, reducing traveling times, waiting time on the bus stop, and not oversaturated units). Additionally, to control the spread of COVID-19 on PTN, our algorithm can detect passengers with or without a face mask. We provide an in-depth description of our proposed Deep Neural Network and its implementation in Python programming language.

R. Hasimoto-Beltran (✉)
Centro de Investigación en Matemáticas (CIMAT) A.C., Jalisco S/N, Col. Valenciana CP: 36023, Apartado Postal 402, CP 36000, Guanajuato, Gto, Mexico
e-mail: hasimoto@cimat.mx

O. F. Eufracio-Vazquez · B. Calderon-Damian
CIMAT, Guanajuato, Gto, Mexico
e-mail: odin.eufracio@cimat.mx

B. Calderon-Damian
e-mail: berenice.calderon@cimat.mx

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_2

2.1 Introduction

Public Transportation (PT) service (on-road buses in this work) is transforming the way people commute within their surroundings (rural or urban) or even across cities. It represents an essential alternative to a personal automobile with additional benefits, safety (10 times safer per mile in the US), affordability, environmentally friendly (it reduces gasoline consumption and carbon footprint), among others [2]. PT demand is constantly increasing around the world, with an actual user's penetration (available number of users) of 6.6% in 2022 and expected to increase to 6.8% by 2026, providing transport services to 537 million users worldwide [23]. Additionally, the United Nations estimates a population increase of 20% by 2050 [17], representing almost 10 billion people demanding new mobility solutions, such as infrastructure, quality of service, facilities for special-need groups (children, persons with disabilities, older persons, etc.), and affordable rates. Demands for improving PT services are being supported by The Sustainable Mobility for All (SuM4All^{TM}) initiative [5], which is working on an ambitious project for transforming the future of the transport sector to achieve four main objectives for meritorious mobility: (1) *Universal Access*, it should be equitable (no one behind); (2) *Efficient*, in a subjective user' perspective; (3) *Safe* (avoid transport accidents); and (4) *Green Mobility*, to reduce both air and noise pollution.

Among these objectives, Transport Efficiency (TE) seems to be the most crucial variable to ensure that users' demands are met effectively. It takes into account metrics that usually involve transport cost, in-vehicle travel time, waiting time, and crowding [3]. There is a close and straight relationship between crowding and users' comfort level. It has been found that, as the number of passengers increases to the available transport capacity (crowding), so does the users' level of stress, anxiety, and feeling of invasion to privacy, affecting self-comfort and personal wellbeing [11, 24]. A recent study [1] shows an average reduction of 56.8% in accessibility to jobs in a regular workday morning peak due to crowding discomfort. It is concluded that the level of passengers' discomfort produced by crowding is, in general, the main decision-maker for not using PT services. Therefore, crowding should be considered as a key metric to be taken into account for concessionaires, city-transport planners, and policymakers to determine the effectiveness of bus scheduling choices (circuit frequency and the number of busses per city or urban route) to enhance passengers' travel experience and transport usage.

Transport crowding detection or Passengers' Density Estimation (PDE) has been studied from different perspectives, going from laborious and tedious on field investigations, up to the application of surveillance cameras and automatic techniques for scenes processing and analysis such as image processing and Deep Neural Networks (DNNs), respectively. Oberli et al. [18] evaluated RFID technology tags for passenger tracking and counting, concluding that recognition percentages are susceptible by the antenna position and radiation pattern. Zhang et al. [29] made use of smartcard fare collection systems and GPS to derive the location where passengers get on and off the bus to estimate the passenger density in every bus. There are hundreds of cities

worldwide where smartcards are not used yet. In Handte et al. [8] a WLAN access point is used to detect the number of onboard passengers. Users can then retrieve the crowd density estimations for the public transportation system from their mobile devices. It is reported that the number of passengers can be underestimated because some users do not connect to the public IP, Wi-Fi, or their cellphones are off. Zhan and Liu [30] present a more theoretical method based on the quantification of passengers' probability density function (taken from a subsample of the card payment system) along with multi-objective optimization (Particle Swarm Optimization) to estimate passengers' density. As mentioned by the authors, the results are idealizations.

Passengers' density estimation can also be viewed as an object detection problem. Object detection is a widely studied topic in the literature since it represents an essential component in many computer vision problems such as object tracking, pedestrian detection, people counting, and autonomous vehicles [12]. The methods to solve this object detection problem can be divided into two groups: those that use classical techniques (image processing) and those that apply modern techniques such as deep learning. Each group has specific techniques to solve the problem. Transport Crowding is an object detection problem that considers a fixed camera taking images of the bus's interior, so it is commonly assumed that images will have a static background. However, images will experience moderate to significant illumination changes, e.g., the bus may pass through a building or enter a tunnel, causing variable illumination changes, one of the main challenges of image processing techniques.

Given the above conditions, classical methods such as Background Modeling (BM) [27] or Background Subtraction (BS) [21] can be applied to PDE. With this approach, the passengers will be in the foreground, i.e., they represent the difference concerning the fixed background. The idea is to use this difference to quantify the density of passengers present in the images. The main advantages of this approach are relatively simple methods with low computational cost. On the other hand, the challenges faced by this approach to PDE are rapid and local illumination changes (shadows, light switching, etc.) and a partially dynamic background (on-road bus vibration) [21]. Further image processing techniques have been used in [4, 16, 28], with the inconvenience of being vulnerable to drastic illumination changes.

Recently, DNNs have dominated the object detection task [12]. One of the main advantages of DNNs is that they can automatically extract nonlinear features from the data, exhibiting high levels of robustness to illumination changes and transformations such as translation and rotation. In addition, the networks are complete models, i.e., the same classification model is responsible for extracting the optimal features for a specific task. To perform PDE in transport units, the task is to identify passengers and their location inside the bus (will see later that the user's location is important in current pandemic times). Object detection methods using deep learning can be classified into two-stage and one-stage methods. Two-stage methods first generate object proposals and then classify these proposals, e.g., the R-CNN family of methods [6, 7, 20]. One-stage methods simultaneously extract and classify object proposals, e.g., the YOLO family of methods [19]. Two-stage methods have relatively slower detection speed and higher accuracy, while one-stage methods have much faster detection speed and comparable detection. Liu-Yin et al. [15], proposed

a Convolutional Neural Network (CNN) to detect passengers, along with a Spatio-temporal context model (that addresses the problem of low resolution, variation of illumination, pose and scale) to track the moving head of each passenger. Wang et al. [26], present a lightweight CNN (reduced number of layers) to be run on an embedded system reporting accuracy of 99%. Hsu et al. [10, 12], present a more complex scheme, where two deep learning methods were implemented to extract features from crowds of passengers. The first method determines the number of people in a crowd, while the second detects the area in which head features are clear on a bus.

Even though the problem of counting people is mature, there still exist new challenges in the PT arena, particularly under novel COVID-19 pandemic times. It is important to develop health-safe public transportation by restricting and controlling packed people on-board and avoiding the spread of the virus through social distance. It has been found that the spread of infectious diseases grows abruptly with the number of passengers (without the need of reaching total transport capacity) [13], which in turn modifies the concept or perception of crowding before and during COVID-19. Perception of crowding during the pandemic is based on Public Health Agencies' recommendation for indoor and outdoor social distances, producing negative side effects for passengers such as longer traveling times and higher cost (since buses are not riding at a total capacity). Despite these downside effects, passengers are willing to assume their responsibility for uncrowded and more spacious transportation in exchange for health-safer commutes [24].

In this work, we undertake the task of improving public transportation quality-of-service. We consider two significant user-comfort-related metrics, crowding and users' waiting time. For this, we propose a system architecture with two main components:

1. PDE based on DNN, to be used by public transport authorities to satisfy users' demand and reduce bus crowding efficiently. Our PDE development goes one step further than just detecting faces; we also detect users wearing or not face masks to avoid spreading COVID-19 and generate alerts.
2. Mobile user app for Android OS provides information on the totality of city bus routes, real-time view of all busses on a selected route, estimated time of arrival at a specific stop point, and seating information for each bus selected.

Our final goal is to motivate the use of PT services (independently of the passenger economic level), improve user utilization of time (they know the arrival time of the transport), health-safe commutes, avoid traffic congestion, and decrease pollution. The rest of this work is organized as follows. In the next section, we describe the major components of our system architecture. Experimental setup and results are discussed in Sect. 2.3. The conclusions and future work are presented in Sect. 2.4.

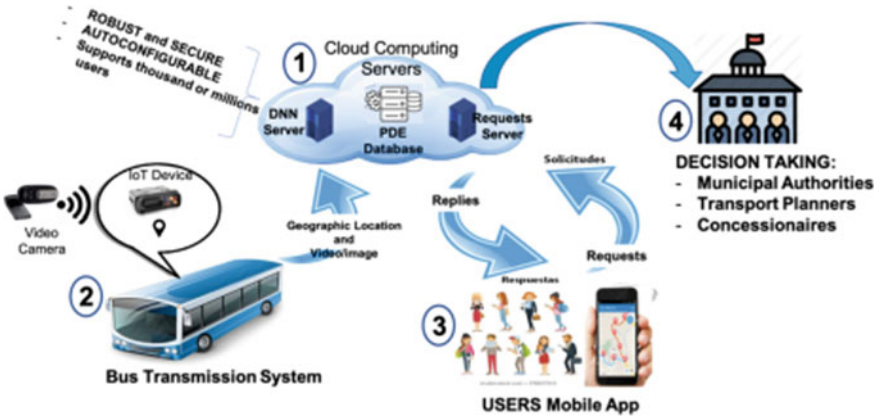

Fig. 2.1 System architecture for passengers' Density Estimation (PDE)

2.2 Proposed Architecture

Efficient urban mobility is still a challenge that needs urgent solutions, particularly in crowded cities. As part of this solution, we proposed a global system architecture that improves and incentivizes the use of the public transportation network. Our proposal provides, on one side, real-time information to transport planners about on-bus crowding and users' waiting-time for controlling and managing the number of busses per route basis (which directly influences the number of people choosing the option of public transportation). On the other side, it offers a mobile user application with information about transport routes, bus location (real-time visualization), estimated arriving times, and passengers' density per transport unit for a more comfortable commuting experience. Users can now make better use of their time and get to the bus stop minutes before the bus's arrival. Details of our system architecture, mobile App, and PDE are provided in the following subsections.

Our system architecture follows a distributed model to manage the public transportation network and provide service to thousands or even millions of users. We will concentrate on the PDE distributed module that efficiently reports, receives, and relays passengers' density information from busses in transit to users and transportation planners. Our system architecture consists of 4 main layers (Fig. 2.1): (1) *Cloud Computing Servers*; (2) *Bus Transmission System*; (3) *User Mobile App*, and (4) *Authorities (Planners) Decision Making*.

1. **Cloud Computing Servers (CCS)**: This layer is responsible for receiving information from bus units and forwarding it to all users and transport planners under explicit request. It consists of three main components: DNN Server (DS), Database Manager (DBM), and Request Server (RS). DS receives image/video information from the bus's interior to estimate the total number of persons and the percentage of those wearing face masks in the specified unit (see Sect. 2.3).

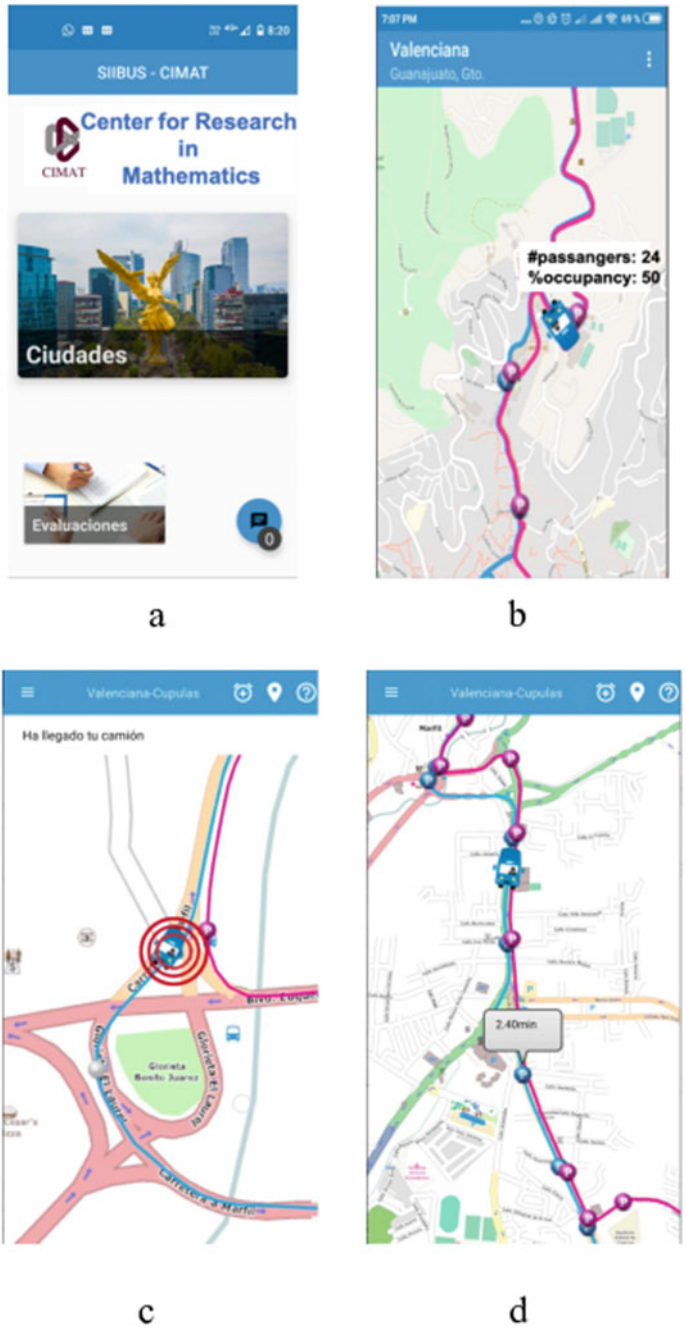

Fig. 2.2 User mobile application. **a** Mobile App cover image, **b** Bus monitoring on route and official stops (circles in red and blue), **c** Bus alarm notification, **d** Estimated arriving time to selected stop

Our DNN implementation can process images (or frames) of 640 × 640 pixels at a speed of 18 images/s; that is, one server can provide service to 1000 busses transmitting one image every 60 s. DNN estimated output is then transmitted to the DBM and made available to users and transport planners through the RS. For every bus and all city routes, the DBM collects and stores the following information: passengers' density estimation, Bus ID, Bus Owner, City ID, Route ID, and date and time. RS is responsible for serving requests to thousand or even million users who connect to the system through the Mobile App (MA); see the Mobile App section layer 3 below.

2. **Bus Transmission System (BTS)**: This system consists of an on-board Linux-based IoT device in charge of collecting geographical location (GPS) and image/video data from all busses in transit. An on-board camera wirelessly communicates image/video data to the IoT device every N s., which is the minimum travel time between two bus stops. N is computed as per-route basis, so in our particular case $30 \leq N \leq 60$ (City of Guanajuato, México). The IoT device is connected to a cellular network to transmit all the information to the DDN server.
3. **Users Mobile App (UMA)**: Developed for Android OS platforms (Smartphones) (Fig. 2.2a), it offers complete visualization of the PT routes and official stops from the comfort of the user's house, office, school, etc. (Fig. 2.2b). It prevents long waiting times and distress produced by crowded, cold, rainy, or insecure bus stops. UMA connects to RS to offer several services, among the most important (to this work) are (a) Real-time bus visualization (Fig. 2.2b); (b) User Notifications, in which users can activate bus-stop alarms to receive a notification when the bus arrives in a specified stop, even when the application has been closed (Fig. 2.2c); (c) Average waiting-time to a selected bus stop (Fig. 2.2b); and (d) PDE, which is activated by clicking on a specific bus icon, reporting the number of passengers and density in percentage (Fig. 2.2d).
4. **Authorities Decision Making (ADM)**: This is a database client for retrieving information from the DBM by selecting date and time. PDE can be retrieved per bus, route, or at a city level in averaged intervals of 15, 30, and 60 min. PDE at route and city level represents the sum of the number of passengers for all busses in a specified route and the number of passengers in all routes, respectively. In this way, transport managers can decide to remove or add bus units to specific routes.

2.3 Deep Learning for Object Detection

The deep learning approach is a type of machine learning which is characterized by the use of a large number of layers with nonlinear activations or processing units [12]. In the context of supervised learning, a traditional machine learning approach requires two main tasks: first, the manual design of an algorithm for feature extraction, and second, the application of a classification algorithm. In comparison, modern machine learning approaches such as deep learning are considered end-to-end learn-

ing systems because a single model or architecture can perform the tasks of feature extraction and data classification tasks. It is important to note that machine learning models can extract or learn nonlinear features suitable for the classification task at hand.

Deep learning methods, in particular convolutional neural networks (CNNs), have become the main focus in tasks related to image processing and computer vision [12]. CNNs exhibit significant similarity to standard neural networks, except that the operations performed on their layers exploit the natural structure in images, i.e., neighboring pixels are related. In addition, CNNs tend to have a lower computational cost by imposing sparse and shared weights on their layers. The four types of layers that are commonly present in CNNs are convolutional layers (Conv2d), subsampling or scaling layers (Pooling), rectification layers (ReLU), and dense layers (Fully-Connected).

The general operation of CNNs in image classification can be divided into two stages. The first stage involves feature extraction and involves a convolutional layer repetition, scaling, and rectification. The second stage functions as a non-linear classifier and involves a dense layer repetition. One of the first successful architectures in CNN where these two stages are visualized is LeNet [14]. This network was used in banks to identify handwritten numbers on checks. LeNet contains two convolutional layers, two subsampling layers, and three fully-connected layers.

New architectures take LeNet's capabilities further but share its general design. Two modern architectures that are commonly used as a first step in more complex tasks, such as object detection, are the VGG [22] and ResNet [9] networks. The VGG network follows the trend of increasing the number of layers and introduces principles that serve as a guideline for future architectures: it prefers the use of smaller convolution filters but increases the number of convolutional layers and doubles the number of convolution filters after each subsampling or scaling layer.

The ResNet network continues and surpasses the trend of building deeper networks. A general principle is that more complex features can be extracted or learned by increasing the number of layers. But not all images or objects in images are complex enough to require many layers. With ResNet, the network can choose the number of layers needed in its processing by including skip connections. These shortcuts can serve in the outward flow of images to compute activations and the return flow to propagate the error, which achieves deep adaptive networks with effective training. It is essential to know these architectures because they are used as a first step in extracting features in more complex tasks. In addition, pre-trained models with large databases serve as a starting point for tuning the model to the specific problem to be solved, i.e., transfer learning.

Object detection performance depends mainly on the quality of the extracted features and the robustness of the selected classifier. In particular, one of the main advantages of CNNs is that they can extract nonlinear features from the data, where such features present a high level of robustness to illumination changes and transformations such as translation.

One of the first successful models to incorporate deep learning techniques in object detection is the Region-based convolutional neural network (R-CNN) model

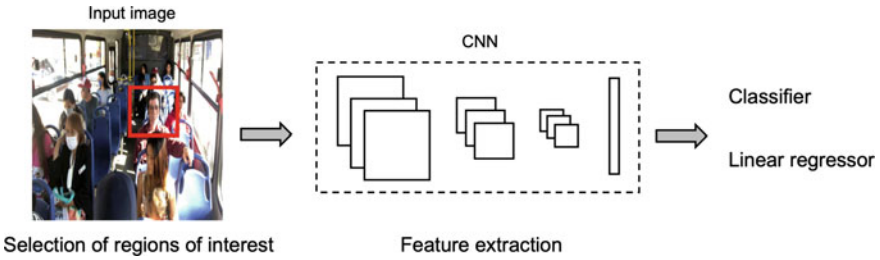

Fig. 2.3 R-CNN architecture, based on [7]

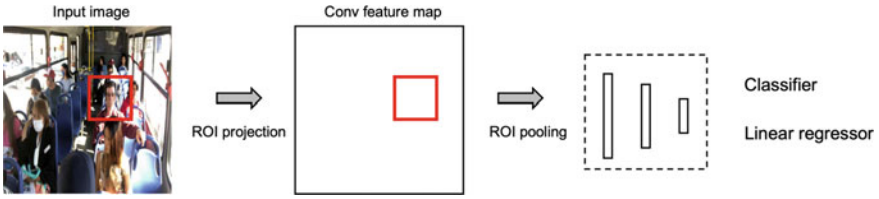

Fig. 2.4 Fast R-CNN architecture, based on [6]

[7]. R-CNN is a two-stage model where a series of proposals are generated and then processed for their correct classification and localization. The R-CNN model consists of the following steps:

1. **Selection of regions of interest**. Regions of interest (patches) are proposed and extracted. In the original work, these patches are selected using the selective search algorithm [25].
2. **Feature extraction**. Each region of interest has a feature vector extracted using a pre-trained CNN.
3. **Region classification**. The CNN feature vector is used to train a classifier and a linear regressor to estimate the class and location of each region.

Figure 2.3 shows the series of steps or modules performed by the R-CNN model.

One of the main problems of the R-CNN model is the high computational cost since each region of interest has features extracted independently. Fast R-CNN [6] solves this problem by computing the feature map of the input image only once, and now all regions of interest extract the features from this single field. Similarly, each region of interest can be of different size, which are standardize by a new Region of Interest Pooling layer before proceeding with the classifier and regressor steps, similar to R-CNN. Figure 2.4 depicts the Fast R-CNN architecture.

In [19], a selective search algorithm is used to propose regions of interest, but requieres a large number of proposals to obtain good results. Ren et al. [20] proposed a Faster R-CNN algorithm by introducing a Region Proposal Network (RPN), that shares image convolutional features with the Fast R-CNN detection network, enabling nearly cost-free region of interest proposals. The RPN is a network that

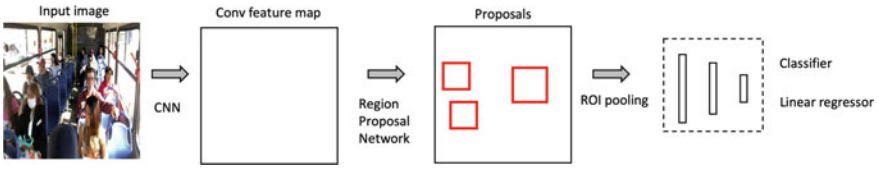

Fig. 2.5 Faster R-CNN architecture, based on [20]

takes as input the feature map of the previous CNN and uses each element of this map as the center of possible regions with different proportions. The RPN is trained to classify each element as an object or background and estimate the object's location. The non-maximum suppression (NMS) technique is applied to reduce similar proposed regions; see Fig. 2.5 for the Faster R-CNN model.

2.4 Experiments and Results

We used the two-stage detector Faster R-CNN model using RestNet50 as the backbone in our experimental setup. Pre-trained weights with the COCO 2017 dataset were used to initialize the model, and subsequently, the full model is fine-tuned with 1000 images collected from the local transportation system. Image data is collected from a single digital video camera installed up-front the interior of the bus and includes a great variety of crowdedness levels, going from emptiness to full-bus capacity (Fig. 2.6). Even though the number of users and active buses decreased by approximately 60% during pandemic times, we managed to select one of the most crowded routes crossing the city from side to side for the experiment. We video recorded a round-trip bus trajectory and split the entire data set in 80%-10%-10% for training, validation, and test, respectively. One of the most relevant parameters that affect face detection accuracy is illumination changes. Fortunately, the City of Guanajuato has a complex orography (hills, mountains, slopes, etc.), underground roads, and tunnels that produce sudden and drastic illumination changes during the bus trajectory that the camera was able to capture, providing us with sufficient information for the network training process. Different than previous schemes in the literature [10], our training process concentrates on face detection along with different distances in the bus (up to 10–12 Mts); in this way, we are able to consistently detect both small and large faces in the absence of occlusion. This training process of detecting faces at long distances helps in reducing the number of monitoring cameras in a tight-budget community.

Given the current COVID-19 health emergency, it was considered important not only to count passengers but also to identify whether or not they are wearing face masks. So, this classification problem was also included in our DNN training process. For passenger detection in the generated database, we obtained a mean Average Precision (intersection of our result and the ground truth) mAP@0.5 of 97.3% and

Fig. 2.6 Bus's interior with variety of crowdedness levels

a mAP@[0.5, 0.95] of 55.2%. Figure 2.7 shows the classification results. The left column represents the original images, and the right column the detected passengers. Each row includes images with different passenger densities where we can see that the network can locate and classify with high accuracy. The last row shows a substantial change of illumination (the truck entered a tunnel), and we can see that the network was able to identify the passengers. The green and red rectangular bounding boxes indicate if the passenger is wearing or not a mask, respectively. Figure 2.8 shows a subsample of 12 bus stops wherein the total number of passengers and the proportion of those passengers wearing or not face masks are calculated. Green and red bars represent those passengers wearing or not a face mask, respectively. In this case, it is clear that the use of masks is not unanimous: it happens in a proportion of almost 50%. Since bus drivers were not restricting passengers' access and use of face masks, this information is made available to city authorities and bus owners through our request

Fig. 2.7 Left original images, right detected passengers. The green bounding box implies that the passenger is wearing a mask and the red color means that they are not wearing a mask

server in order to control the number of busses and the number of passengers to provide an optimal transport service and to avoid the spread of contagious diseases.

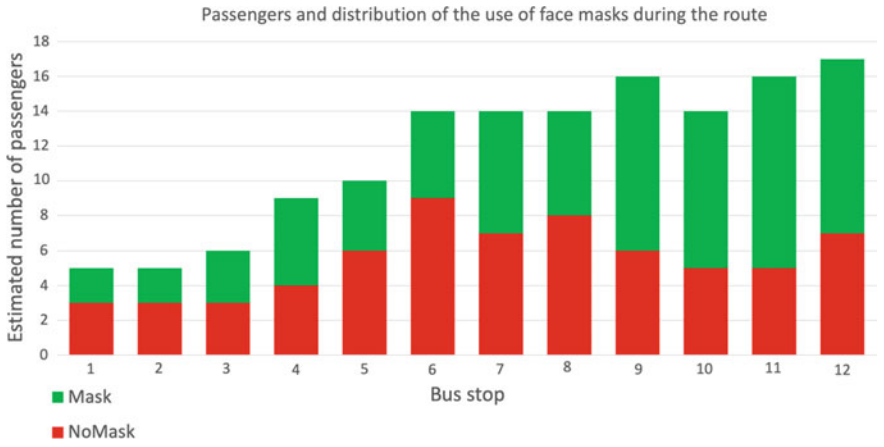

Fig. 2.8 Estimated total number of passengers and proportion of passengers wearing or not face masks

2.5 Conclusions and Future Work

Mobile technologies and machine learning applications become good allies for improving users' satisfaction in public transport services. Using these technologies, we have designed and implemented a transport system architecture that benefits both parts of the transport community, service providers (concessionaires and municipal authorities) and users of the service. This conjunction (providers and users) improves the population's mobility and incentivizes the use of public transportation, which impacts on reducing environmental pollution, traffic congestion, and commuting time [5]. Our architecture efficiently detects passengers' density estimation through a DNN and communicates this metric to transport planners through our cloud service. In this way, the number of transport units in transit can now be handled or controlled in real-time on a route basis. Additionally, a mobile user application has also been developed to provide users with the following commodities, bus geolocation in real-time for a selected route, high precision arriving-time estimates to bus stops, and information about the number of passengers per selected unit. Users can now manage their own time appropriately.

One of the main restrictions in our proposal is the use of only one video camera for estimating passengers density. This, of course, has several restrictions, such as passengers' occlusion and hidden faces (showing only the head of the passenger). To avoid these problems, it is recommendable to have 4 cameras on the buses with a standard length of 12–15 Mts, two on the front sides (left and right) and another two between the middle and backside of the bus, having all four cameras looking backward. In this way, passengers occlusions are decreased considerably since the DNN can detect partial face views.

References

1. Arbex, R., Cunha, C.B.: Estimating the influence of crowding and travel time variability on accessibility to jobs in a large public transport network using smart card big data. J. Trans. Geogr. **85**, 102671 (2020)
2. American Public Transportation Association. Public transportation facts (2019). https://www.apta.com/news-publications/public-transportation-facts. Accessed 15 Jan 2022
3. Batarce, M., Munoz, J.C., de Dios Ortuzar, J., Raveau, S., Mojica, C., Rios, R.A.: Evaluation of passenger comfort in bus rapid transit systems. Technical note no. idb-tn-770. Inter-American Development Bank, pp. 1–44 (2015)
4. Chen, J., Wen, Q., Zhuo, C., Mete, M.: Automatic head detection for passenger flow analysis in bus surveillance videos. In: 5th International Congress on Image and Signal Processing, pp. 143–147 (2012)
5. Sustainable Mobility for All. Global mobility report 2017 (2017). https://sum4all.org/publications/global-mobility-report-2017. Accessed 15 Jan 2022
6. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2015 Inter, pp. 1440–1448 (2015)
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
8. Handte, M., Iqbal, M.U., Wagner, S., Apolinarski, W., Marrón, P.J., Maria Muñoz Navarro, E., Martinez, S., Barthelemy, S.I., Fernández, M.G.: Crowd density estimation for public transport vehicles. In: Proceedings of the EDBT/ICDT 2014 Joint Conference, Athens, Greece (2014). ISSN 1613-0073
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Dec, pp. 770–778 (2016)
10. Hsu, Y.-W., Chen, Y.-W., Perng, J.-W.: Estimation of the number of passengers in a bus using deep learning. Sensors **20**, 2178 (2020)
11. Jaime Soza-Parra, J.C.M., Raveau, S., Cats, O.: The underlying effect of public transport reliability on users' t satisfaction. Trans. Res. Part A: Policy Practice **126**, 83-93 (2019)
12. Jiang, X., Hadid, A., Pang, Y., Granger, E., Feng, X.: Deep Learning in Object Detection and Recognition. Springer Singapore, Singapore (2019)
13. Kumar, P., Khani, A., Lind, E., Levin, J.: Estimation and mitigation of epidemic risk on a public transit route using automatic passenger count data. J. Transp. Res. Board **2675**(5), 94–106 (2021)
14. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
15. Liu, G., Yin, Z., Jia, Y., Xie, Y.: Passenger flow estimation based on convolutional neural network in public transportation system. Knowl.-Based Syst. **123**, 02 (2017)
16. Mukherjee, S., Saha, B., Jamal, I., Leclerc, R., Ray, N.: Anovel framework for automatic passenger counting. In: 18th IEEE International Conference on Image Processing, pp. 2969–2972 (2011)
17. United Nations. World population prospects (2019). https://population.un.org/wpp/Graphs/DemographicProfiles/900. Accessed 15 Jan 2022
18. Oberli, C., Torres-Torriti, M., Landau, D.: Performance evaluation of uhf rfid technologies for real-time passenger recognition in intelligent public transportation systems. IEEE Trans. Intell. Trans. Syst. **11**(3) (2010)
19. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Dec, pp. 779–788 (2016)
20. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017)

21. Shahbaz, A., Hariyono, J., Jo, K.H.: Evaluation of background subtraction algorithms for video surveillance. In: Frontiers of Computer Vision, FCV 2015 (2015)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, pp. 1–14 (2015)
23. Statista. Public transportation facts (2019). https://www.statista.com/outlook/mmo/mobility-services/public-transportation/worldwide. Accessed 15 Jan 2022
24. Tirachini, A., Hensher, D.A., Rose, J.M.: Crowding in public transport systems: effects on users, operation and implications for the estimation of demand. Transp. Res. Part A Policy Pract **53**, 36–52 (2013)
25. Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. Int. J. Comput. Vis. **104**(2), 154–171 (2013)
26. Wang, Z., Cai, G., Zheng, C., Fang, C.: Bus-crowdedness estimation by shallow convolutional neural network. In: International Conference on Sensor Networks and Signal Processing (SNSP), pp. 105–110 (2018)
27. Yong, X., Dong, J., Zhang, B., Daoyun, X.: Background modeling methods in video analysis: a review and comparative evaluation. CAAI Trans. Intell. Technol. **1**(1), 43–60 (2016)
28. Yang, T., Zhang, Y., Shao, D., Li, Y.: Clustering method for counting passengers getting in a bus with single camera. Opt. Eng. **49**(3), 037203 (2010)
29. Zhang, J., Yu, X., Tian, C., Zhang, F., Tu, L., Xu, C.: Analyzing passenger density for public bus: inference of crowdedness and evaluation of scheduling choices. In: 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), Qingdao, China
30. Zhao, J., Liu, Y., Gui, Y.: Multi-objective optimization of university bus based on passenger probability density estimation. Appl. Math. **8**, 621–629 (2017)

Chapter 3
Epistemic Uncertainty Quantification in Human Trajectory Prediction

Mario Canche, Ranganath Krishnan, and Jean-Bernard Hayet

Abstract Human Trajectory Prediction (HTP) is a critical technology in several areas related to the development of smart cities, such as video monitoring or autonomous driving. In the last years, there has been a leap forward in the state of the art of HTP, with great improvements observed in most of the classical benchmarks used in the related literature. This has been possible through the use of powerful data-driven deep learning techniques, coupled with probabilistic generative models and methodologies to cope with contextual information and social interactions between agents. In this chapter, we first show how incorporating Bayesian Deep Learning (BDL) techniques in Human Trajectory Prediction allows to provide realistic estimates of the epistemic and aleatoric uncertainties on the computed predictions. In addition, we also present an original methodology to assess the quality of the produced uncertainties (through BDL or other probabilistic approach).

3.1 Introduction

Video monitoring and autonomous driving are key technologies behind the future of Intelligent Transportation and, more generally, Smart Cities. For those applications and for many more, the developed systems critically need to predict the short-term motion of pedestrians, e.g. to anticipate anomalous crowd motion through a video monitoring system or to assess the safety of the motion an autonomous vehicle is about to undertake. In this context, Human Trajectory Prediction (HTP) has had a very fast development recently, in particular through the use of powerful, data-driven

M. Canche · J.-B. Hayet (✉)
Centro de Investigación en Matemáticas, Callejon Jalisco S/N, Guanajuato, México
e-mail: jbhayet@cimat.mx

M. Canche
e-mail: mario.canche@cimat.mx

R. Krishnan
Intel Labs, Hillsboro, OR, USA
e-mail: ranganath.krishnan@intel.com

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_3

deep learning techniques [10, 32]. Coupled with probabilistic generative models and dedicated methodologies to cope with contextual information or social interactions between agents, they have allowed a leap forward in the state of the art, with great improvements on classical HTP benchmarks used in the related literature.

However, a key element is still missing in most of the methods currently developed for guaranteeing some level of robustness in the way the forecast data can be used: A sound estimation of the uncertainties associated to the outputs produced by the predictors. As stated in several works before [16], this aspect is of paramount importance to ensure robustness in the use of HTP outputs, so that any decision system using the predicted trajectories of pedestrian could also be aware of how uncertain these estimates are. Note that, as highlighted in [16], there are two important kind of uncertainties to cope with: The *aleatoric* uncertainties are intrinsic to the data and reflect the inherent variations of the data, e.g. due to noise or to the inherently multimodal nature of the data; the *epistemic* uncertainties reflect the limits of the model, in particular due to the finiteness of the data we train the model with. If we had an infinite amount of data, such epistemic uncertainties could be zeroed, however, this limit case is in general far from being approached. In most systems, and in particular in the case of HTP systems, the quantity of training data is rather small. This chapter brings two contributions oriented to this issue: (1) we show how Bayesian Deep Learning (BDL) techniques allows us to provide realistic estimates of both epistemic and aleatoric uncertainties on the computed trajectory predictions; (2) we present an original methodology to calibrate the produced uncertainties. Note that the proposed methodology does *not* aim at producing more precise forecast but at producing more precise uncertainty estimates.

The Chapter is organized as follows: In Sect. 3.2, we describe related work and formalize the HTP problem we deal with; in Sect. 3.3, we show how BDL techniques can allow to estimate the epistemic uncertainties on the outputs on BDL systems; in Sect. 3.4, we describe a methodology that assess the quality of the estimated uncertainties; finally, in Sect. 3.5, we report results on standard HTP benchmarks that validate the whole approach.

3.2 Related Work and Problem Statement

Modeling human motion has a long history, taking its roots, in particular, in the simulation community, where researchers have proposed hand-tailored models such as the Social forces model [12], in which the motion of agents is seen has the result of attractive and repulsive forces acting on these agents. Many other approaches have been explored, such as cellular automata [3] or statistical models like Gaussian Processes [34, 35]. More recently, following the rise of deep learning techniques, data-driven methods have turned the field upside down and have allowed huge improvements in HTP benchmarks. Neural networks, and in particular recurrent neural networks, have been used intensively since the seminal work of Social LSTM [1] that uses LSTM (Long-Short Term Memory [13]) sub-networks to simul-

Fig. 3.1 Some examples of the raw data in ETH-UCY HTP datasets

taneously encode the observed trajectories and the social interactions between agents in the scene. In parallel, several architectures have also been proposed to handle the spatial context surrounding the agents [24, 36]. Most of these networks are *deterministic*, in the sense that, for one input (the history of one agent and some contextual data), they give as an output one single trajectory.

On the opposite, generative models are able to produce samples from the predictive distribution, at least in theory. They are more efficient in dealing with potentially multi-modal predictive distributions, e.g., when a person at a crossroad has two or three plausible plans ahead of him. Generative adversarial networks [8] (GAN) have been used for this purpose in HTP [10, 18, 31, 42]. Conditional Variational Autoencoders [33] (CVAE) [17, 22] is another technique used to design generative models, mapping realizations of a multivariate Gaussian distribution onto trajectory variations. The Trajectron [15] and Trajectron++ [32] systems, that are some of the most prominent systems in the last years, use CVAE as their backbones.

Another line of research has been the use of dedicated attention mechanisms [37], e.g. to identify the most relevant neighbors interacting with a given agent [38] or fully attention-based methods, i.e. Transformer networks (TF) [6] to replace the encoding/decoding recurrent networks in more traditional systems.

However, most of the aforementioned methods simply do *not* handle neither aleatoric nor epistemic uncertainties. A few techniques such as [1] include modeling choices that allow to represent aleatoric uncertainties, by defining the output of the model as the parameters of a Gaussian distribution; generative models [10, 32] implicitly encode the aleatoric uncertainties in a more general way. To the best of our knowledge, there is no system in the current literature that is able to cope with epistemic uncertainties, and the consequence is that the overall uncertainties in most system are probably under-evaluated.

Our forecasting problem is the following: Given an input sequence of T_{obs} pedestrian planar positions $\mathbf{p}_t$ for $t = 1 \dots T_{obs}$ in $\mathbb{R}^2$, we want to predict the sequence $\mathbf{p}_{T_{obs}+1:T_{pred}}$ of its future positions. In Fig. 3.2, a typical input (in blue) and target (in red) are shown. As it can be seen in Fig. 3.1, where some raw trajectories from typical datasets are illustrated, in general, more information can be available (maps, relative positions of other pedestrians…). However, here, we limit the scope of our approach to using only the past positions, expressed in world relative coordinates to the position at T_{obs}. We also focus on the case of parametric, deterministic regression

models, i.e. models π parameterized by parameters $\mathbf{w}$. Finally, we consider the case of forecasting systems producing an estimate $\hat{\Sigma}^{\mathbf{w}}_{T_{obs}+1:T_{pred}}$ of the associated aleatoric uncertainty. Hence, our target model has the form

$$\hat{\mathbf{p}}^{\mathbf{w}}_{T_{obs}+1:T_{pred}}, \hat{\Sigma}^{\mathbf{w}}_{T_{obs}+1:T_{pred}} = \pi(\mathbf{p}_{1:T_{obs}}; \mathbf{w}), \tag{3.1}$$

where each $(\hat{\mathbf{p}}^{\mathbf{w}}_t, \hat{\Sigma}^{\mathbf{w}}_t)$ gives the first two moments of the predictive distribution over the position at t. To alleviate the notation, when necessary, we will refer to the input sequence as $\mathbf{x} \triangleq \mathbf{p}_{1:T_{obs}}$ and to the output as $\mathbf{y} \triangleq \mathbf{p}_{T_{obs}+1:T_{pred}}$, $\Sigma \triangleq \Sigma_{T_{obs}+1:T_{pred}}$. An illustration of typical outputs from such a model is given in Fig. 3.2, left, with predicted uncertainties at each future timestep depicted in pink.

In typical setups, such a model is trained to determine optimal parameters $\mathbf{w}^*$ given a dataset $\mathbf{D}$ of K training examples $\{(\mathbf{x}_k, \mathbf{y}_k)\}$. The likelihood of the true $\mathbf{y}_k$ over the Gaussian predictive distribution defined by $\hat{\mathbf{y}}^{\mathbf{w}}_k$ and $\hat{\Sigma}^{\mathbf{w}}_k$ is maximized:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \prod_{k=1}^{K} \mathcal{N}(\mathbf{y}_k; \hat{\mathbf{y}}^{\mathbf{w}}_k(\mathbf{x}_k), \hat{\Sigma}^{\mathbf{w}}_k(\mathbf{x}_k)). \tag{3.2}$$

Here, in addition of the aleatoric uncertainties that the formulation above allows top cope with, we would also like to quantify the underlying epistemic uncertainties.

3.3 Quantification of Aleatoric and Epistemic Uncertainties

This section gives a short introduction to Bayesian deep learning and shows how it can be used to produce estimates of both aleatoric and epistemic uncertainties.

3.3.1 Bayesian Deep Learning

Bayesian deep learning provides a probabilistic interpretation of deep neural networks by inferring distributions over the model parameters $\mathbf{w}$. The initial idea of applying Bayesian principles in neural networks was in 1990's [25, 27]. With the advancements in deep learning yielding state-of-the-art results in various domains, there has been a resurgence of Bayesian neural networks with increasing interest among the AI research community to make use of them as a principled framework to quantify reliable uncertainty estimates in deep learning models. Given data $\mathbf{D} = \{(\mathbf{x}_k, \mathbf{y}_k)\}$, a model likelihood $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$ and prior distribution $p(\mathbf{w})$, we would like to learn the posterior distribution over the weights $p(\mathbf{w} \mid \mathbf{D})$:

$$p(\mathbf{w} \mid \mathbf{D}) = \frac{p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})\, p(\mathbf{w})}{\int p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})\, p(\mathbf{w})\, \mathbf{dw}}. \tag{3.3}$$

Since obtaining the posterior distribution above is analytically intractable, many approximate Bayesian inference methods have been proposed including stochastic gradient MCMC [4, 39, 41] and variational inference [2, 5, 9, 26] methods. Once the posterior distribution is learned, predictions for test data inputs $\mathbf{x}$ are obtained through marginalization of the weight posterior through Monte Carlo sampling:

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{D}) = \int p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})\, p(\mathbf{w} \mid \mathbf{D})\, d\mathbf{w} \approx \frac{1}{N} \sum_{i=1; \mathbf{w}_i \sim p(\mathbf{w}|\mathbf{D})}^{N} p\left(\mathbf{y} \mid \mathbf{x}, \mathbf{w}_i\right), \tag{3.4}$$

where the N weights $\mathbf{w}_i$ are sampled from the posterior over the weights $p(\mathbf{w} \mid \mathbf{D})$. In the trajectory prediction problem we described in the previous section, $\mathbf{x}$ is a time series of observed positions and $\mathbf{y}$ is a time series of future positions.

3.3.2 *Uncertainty Estimation Through Bayesian Deep Learning*

Suppose that we have designed a (deterministic) neural network π with parameters $\mathbf{w}$, able to output a prediction $\hat{\mathbf{y}}^{\mathbf{w}}$ for an input $\mathbf{x}$, as in Eq. 3.1, altogether with an estimate of the variance $\hat{\Sigma}^{\mathbf{w}}$ over these predictions. We also suppose that with some Bayesian deep learning method as the ones exposed above, we are able to sample parameters $\mathbf{w}_i$ from a posterior distribution over the network parameters, given the training dataset $\mathbf{D}$:

$$\mathbf{w}_i \sim p(\mathbf{w}|\mathbf{D}). \tag{3.5}$$

Note that, as commented in the previous section, performing this sampling on the posterior distribution of the weights gives a way to estimate epistemic uncertainties, i.e. variations over the model parameters themselves. By using N samples $\mathbf{w}_i$ from this posterior $p(\mathbf{w}|\mathbf{D})$ over the model parameters, we produce a mixture of N Gaussian distributions:

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{D}) = \frac{1}{N} \sum_{i=1; \mathbf{w}_i \sim p(\mathbf{w}|\mathbf{D})}^{N} \mathcal{N}\left(\mathbf{y}; \hat{\mathbf{y}}^{\mathbf{w}_i}(\mathbf{x}), \hat{\Sigma}^{\mathbf{w}_i}(\mathbf{x})\right). \tag{3.6}$$

From this representation estimates for the first two moments of the posterior distribution over $\mathbf{y}$, given $\mathbf{x}$ and $\mathbf{D}$, as

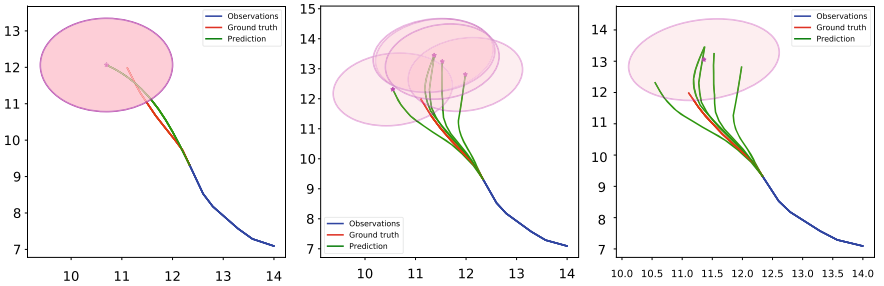

Fig. 3.2 Uncertainty representation over the predicted position at timestep $T_{pred} = 12$. The blue trajectory corresponds to the T_{obs} observed positions (used as an input for the prediction system), the red trajectory corresponds to the T_{pred} ground truth future positions, the green trajectories are the predictions and the corresponding covariances (estimated as in Eq. 3.1 at T_{pred}) are depicted in pink. Left: Output from a deterministic model. Middle and Right: Outputs from a Bayesian model (in this case, a variational model described in Sect. 3.5.1.2). The right subfigure depicts the covariance resulting from the mixture in the middle

$$\hat{\mathbf{y}}(\mathbf{x}) \quad \triangleq \frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{y}}^{\mathbf{w}_i}(\mathbf{x}), \tag{3.7}$$

$$\hat{\Sigma}_{\mathbf{y}}(\mathbf{x}) \quad \triangleq \frac{1}{N}\sum_{i=1}^{N}\hat{\Sigma}^{\mathbf{w}_i}(\mathbf{x}) + \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{y}}^{\mathbf{w}_i}(\mathbf{x}) - \hat{\mathbf{y}}(\mathbf{x}))(\hat{\mathbf{y}}^{\mathbf{w}_i}(\mathbf{x}) - \hat{\mathbf{y}}(\mathbf{x}))^T, \tag{3.8}$$

where $\hat{\mathbf{y}}^{\mathbf{w}_i}(\mathbf{x})$, $\hat{\Sigma}^{\mathbf{w}_i}(\mathbf{x})$ is the output of the forecasting model to the input $\mathbf{x}$ at the specific operation point $\mathbf{w}_i$. The volume of the variance matrix $\hat{\Sigma}_{\mathbf{y}}(\mathbf{x})$ gives us a first rough way to quantify the uncertainty over our output.

To get a more fine-grained representation of the uncertainty, we can also simply keep the mixture of Gaussian of Eq. 3.6.

As an illustration, in Fig. 3.2, we depict both uncertainty representations for Eqs. 3.6 (middle), 3.7 and 3.8 (right) for one single sampled predicted trajectory.

To ease the depiction and the explanations about the uncertainty calibration process, we will only consider the uncertainty at the *last* predicted position T_{pred}, i.e. we will consider the 2×2 variance matrix

$$\hat{\Sigma}^{\mathbf{w}}_{T_{pred}}. \tag{3.9}$$

3.4 Evaluating and Calibrating Uncertainties

In practice, uncertainty estimates produced by the aforementioned methods may not always capture the true variability in the output data. They often tend to be miscalibrated and they end up not representing correctly the model errors. Uncertainty calibration has been extensively studied in niche areas such as weather forecasting [7].

However, it is not easy to generalize or extrapolate these methods to other areas. In [20], a method is proposed to re-calibrate the output of any 1D regression algorithm, including Bayesian neural networks. It is inspired by the Platt scaling [30] and uses a specific subset of recalibration data with isotonic regression to correct the estimate for uncertainty. However, like other uncertainty calibration methods, only the case of regression with a single independent variable to adjust is presented. Nevertheless, our trajectory prediction problem outputs sequences of pairs of variables, which represent the position of the pedestrian in a 2D plane. In the following Sect. 3.4.1, we first recall the case of 1D regression as presented in Kuleshov et al. [20]. The generalization of the uncertainty calibration method to the case of multi-variate regression, with several independent variables to adjust, is described in Sect. 3.4.2.

3.4.1 Calibration of Uncertainties in 1D Regression

We first consider the case of the 1D regression problem. Let $\{\mathbf{x}_k, y_k\}_{k=1}^K$ a labeled dataset, where $\mathbf{x}_k \in \mathbb{R}^n$ and $y_k \in \mathbb{R}$ are identically distributed independent samples (i.i.d) from some data distribution. We assume that our regression model gives us an uncertain output in the form of a function

$$\pi : \mathbb{R}^n \to (\mathbb{R} \to [0, 1]), \tag{3.10}$$

which associates to each input data $\mathbf{x}_k$ a 1D cumulative distribution function $F_k(y)$ targeting y_k, that is $\pi(\mathbf{x}_k) = F_k$. For example, if the output were very precise around a predicted value $\hat{y}_k$, the corresponding density function would have a lone peak at that value and F_k would be such that

$$F_k(y) = \begin{cases} 1 & \text{if } y \geq \hat{y}_k \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

We also define $F_k^{-1} : [0, 1] \to \mathbb{R}$ as the quantile function, in such way that, for a probability value $\alpha \in [0, 1]$,

$$F_k^{-1}(\alpha) = \inf\{y \ : \ \alpha \leq F_k(y)\}. \tag{3.12}$$

In this 1D regression case, having a model well calibrated means that, for example, ground truth values y_k should fall within our 90% confidence intervals (as given by our uncertainty estimates) approximately 90% of the cases. This can be written as

$$\frac{1}{K} \sum_{k=1}^{K} \mathbb{I}\{y_k \leq F_k^{-1}(\alpha)\} \xrightarrow{K \to \inf} \alpha \qquad \text{for any } \alpha \in [0, 1]. \tag{3.13}$$

An out-of-the-box system will typically not satisfy the property above and needs to be calibrated. Calibration methods train an auxiliary regression model $\gamma : [0, 1] \rightarrow [0, 1]$ that modifies the outputs (the probabilities) of the originally trained predictor π of Eq. 3.10. That is, γ is trained so that the model composed of $\gamma \circ \pi$ is well calibrated. Kuleshov et al. [20] propose to use isotonic regression as the auxiliary calibration function γ. It is trained by fitting it with pairs of outputs/empirical evaluations of cumulated distributive function $\{F_k(y_k), \hat{P}(F_k(y_k))\}_{k=1}^K$ on a calibration set, where y_k is the true output and

$$\hat{P}(\alpha) = \frac{1}{K} |\{y_k \mid F_k(y_k) \leq \alpha,\ k = 1, \ldots, K\}| \tag{3.14}$$

is the proportion of the data within the calibration set where y_k is less than the α-th quantile of F_k. The uncertainty calibration pseudocode of this regression is presented in the Algorithm 3.1, and for more references, we urge the reader to consult [20].

Algorithm 3.1 Calibration for 1D regression

1: **Input:** Uncalibrated model $\pi : X \rightarrow (Y \rightarrow [0, 1])$ and independent calibration dataset $\mathbf{S} = \{(\mathbf{x}_k, y_k)\}_{k=1}^K$.
2: Construct the dataset:

$$\mathbf{C} = \{([\pi(\mathbf{x}_k)](y_k), \hat{P}([\pi(\mathbf{x}_k)](y_k)))\}_{k=1}^K, \tag{3.15}$$

where

$$\hat{P}(\alpha) = \frac{1}{K} |\{y_k \mid [\pi(\mathbf{x}_k)](y_k) \leq \alpha, k = 1, \ldots, K\}|. \tag{3.16}$$

3: Train an isotonic regression model γ on $\mathbf{C}$.
4: **Output:** Auxiliary regression model $\gamma : [0, 1] \rightarrow [0, 1]$.

3.4.2 Highest Density Regions

The notion of quantile used in the 1D regression case presented above is specific to the 1D case, hence we propose here to extend it to higher dimensions. For this purpose, we make use of the α-highest density regions (HDR) concept [14], defined for a probability density function f as follows

$$R(f_\alpha) = \{\mathbf{y} \text{ s.t. } f(\mathbf{y}) \geq f_\alpha\}$$

where $1 - \alpha$ defines an accumulated density over this region, that defines f_α:

$$\mathbb{P}(\mathbf{y} \in R(f_\alpha)) = 1 - \alpha. \tag{3.17}$$

The interpretation is quite straightforward: for a value α, the HDR $R(f_\alpha)$ is the region corresponding to all values of $\mathbf{y}$ where the density is superior to f_α, where f_α is such that the total density summed up in this region is exactly $1-\alpha$. When $\alpha \to 0$, $f_\alpha \to 0$ and the HDR tends to the support of the density function. When $\alpha \to 1$, then $f_\alpha \to \max_{\mathbf{y}} f(\mathbf{y})$ and the HDR tends to the empty set.

As proposed in [14], we can evaluate the mapping $\alpha \to f_\alpha$ by using a Monte Carlo approach. Suppose that we get n samples from f, then we can sort these samples through their density values as

$$f_{(1)} < f_{(2)} < \cdots < f_{(n)}, \tag{3.18}$$

and estimate f_α as $f_{(a)}$ where

$$a = \arg\max_k k \text{ s.t. } \frac{1}{\sum_{l=1}^{n} f_{(l)}} \sum_{l=k}^{n} f_{(l)} \geq 1-\alpha. \tag{3.19}$$

This means that we add the values $f_{(i)}$ (Eq. 3.18) in descending order until the sum is greater than or equal to $1-\alpha$. The last value $f_{(i)}$ added to the descending sum, will correspond to the estimation value of f_α. This method to estimate f_α is the one used in the experiments of Sect. 3.5.

Suppose that for each example k, our model outputs a probability density function f_k representing the uncertainty on our answer. Then we can generalize the notion of a well-calibrated output by stating that the model will be well calibrated, when for any $\alpha \in [0, 1]$, the proportion of ground-truth data falling into the $100(1-\alpha)\%$-HDR is roughly $1-\alpha$. Translating this into equations:

$$\frac{\sum_{k=1}^{K} \mathbb{I}\{\mathbf{y}_k \in R(f_\alpha)\}}{K} = \frac{\sum_{k=1}^{K} \mathbb{I}\{f_k(\mathbf{y}_k) \geq f_\alpha\}}{K} \xrightarrow{K \to \inf} 1-\alpha \quad \forall \alpha \in [0, 1]. \tag{3.20}$$

3.4.3 HDR-Based Calibration

We propose a recalibration scheme for 2D regression similar to 1D regression case. We will use it for calibrating the uncertainties over the last predicted position in HTP (hence, to predict a 2D vector). As explained hereafter, we use HDR and probability density functions instead of quantile functions and cumulative distribution functions.

We are given a calibration dataset $\{\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^{K}$ with $\mathbf{x}_k$ the inputs data and $\mathbf{y}_k$ the target data. These pairs are identically distributed independent samples (i.i.d) from some data distribution (pairs inputs/ground-truth inputs). In our application in trajectory prediction, we limit ourselves to consider as a target the last predicted position ($t = T_{pred}$), which means that $\mathbf{y}_k \in \mathbb{R}^2$. However, the extension to the whole predicted trajectory should be straightforward.

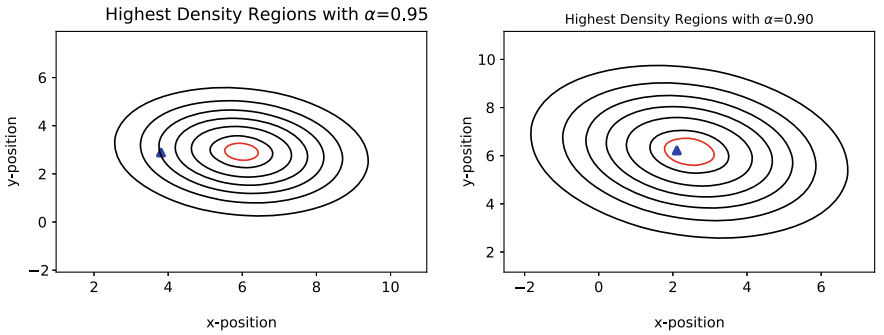

Fig. 3.3 Two illustrations of the use of HDR regions, delimited by the red curves, with other level curves for the probability density functions in black. The ground-truth points appear in blue

Algorithm 3.2 Calibration for Multi-dimensional regression

1: **Input:** Uncalibrated model $\pi : X \to (Y \to \mathbb{R}^+)$ giving as an output a p.d.f. $f(\mathbf{x}_k)$ and independent calibration dataset $\mathbf{S} = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^K$.
2: For all k, find the value of $\hat{\alpha}_k$ that satisfies

$$\mathbb{P}(\ \mathbf{y} \in R(f_k(\mathbf{y}_k))\ ) = 1 - \hat{\alpha}_k. \qquad (3.21)$$

3: Construct the dataset:

$$\mathbf{C} = \{\hat{\alpha}_k,\ \hat{P}_\alpha(\hat{\alpha}_k)\}_{k=1}^K \qquad (3.22)$$

where

$$\hat{P}_\alpha(\alpha) = \frac{|\{y_k \mid f_k(\mathbf{y}_k) \geq f_\alpha,\ k = 1, \ldots, K\}|}{K} \qquad (3.23)$$

4: Train an isotonic regression model γ on $\mathbf{C}$.
5: **Output:** Auxiliary regression model $\gamma : [0, 1] \to [0, 1]$.

We have seen in Sect. 3.3 that we can estimate the probability density function f_k with the output of the Bayesian model. The probability density function, in our case, has the form of a mixture of Gaussians, as explained in Sect. 3.3; more generally, it can be estimated from a set of samples generated from the Bayesian model by using Kernel density estimation (KDE). Next, we evaluate the corresponding ground truth value $\mathbf{y}_k$ in the probability density function and find the value of $\hat{\alpha}_k$ that satisfies

$$\mathbb{P}(\ \mathbf{y} \in R(f_k(\mathbf{y}_k))\ ) = 1 - \hat{\alpha}_k. \qquad (3.24)$$

where

$$R(f_k(\mathbf{y}_k)) = \{\mathbf{y}\ \text{s.t.}\ f_k(\mathbf{y}) \geq f_k(\mathbf{y}_k)\} \qquad (3.25)$$

thus defining an $100(1 - \hat{\alpha}_k)\%$ HDR where $f_k(\mathbf{y}_k)$ occupies the role of f_α in the formal definition of HDR. Therefore, we compute the $\hat{\alpha}_k$ for each $\mathbf{y}_k$ of the calibration

set and then estimate the empirical probability that each calibration data $\mathbf{y}_k$ falls within the $100(1-\hat{\alpha}_k)\%$ HDR, for each $\hat{\alpha}_k$ possible.

In order to calibrate, be the calibration set $\{\hat{\alpha}_k, \hat{P}_\alpha(\hat{\alpha}_k)\}_{k=1}^K$, where

$$\hat{P}_\alpha(\alpha) = \frac{|\{y_k \mid f_k(\mathbf{y}_k) \geq f_\alpha,\ k = 1, \ldots, K\}|}{K} \tag{3.26}$$

denotes the fraction of data belonging to the $100(1-\alpha)\%$ HDR. In Fig. 3.3, we give a couple of examples of ground truth data $\mathbf{y}_k$ altogether with the predictive density f_k. As in the case of 1D regression, we use isotonic regression as the regression model to calibrate on these data. This calibration model corrects the α values that define the HDR so that the condition of Eq. 3.20 is fulfilled. The whole algorithm is described in Algorithm 3.2.

3.5 Experiments

3.5.1 Trajectory Prediction Base Model and Bayesian Variants

For all the experiments described in this section, we use, as the base deterministic model, a simple LSTM-based sequence-to-sequence model that we describe hereafter. This model outputs a single trajectory prediction with its associated uncertainty.

3.5.1.1 Base Model

Inputs are pedestrian position increments: $\mathbf{x} \triangleq \delta_{1:T_{obs}} \triangleq \mathbf{p}_{1:T_{obs}} - \mathbf{p}_{0:T_{obs}-1} \in \mathbb{R}^{2T_{obs}}$, each of which being mapped through a linear embedding into a vector $\mathbf{e}_t \in \mathbb{R}^{d_e}$, for all the timesteps of the observed sequence $t = 1, \ldots, T_{obs}$

$$\mathbf{e}_t = \mathbf{W}_{emb}\delta_t, \tag{3.27}$$

which is fed to a first encoding LSTM layer referred to as $\mathcal{L}_{enc}$, which encodes the sequence $\mathbf{e}_{1:T_{obs}}$. Its hidden state is denoted as $\mathbf{h}_t^e \in \mathbb{R}^{d_c}$ and is updated recursively according to the following scheme

$$\mathbf{h}_t^e = \mathcal{L}_{enc}(\mathbf{h}_{t-1}^e, \mathbf{e}_t; \mathbf{W}_{enc}). \tag{3.28}$$

On the decoding side, another LSTM layer $\mathcal{L}_{dec}$ is used to perform the prediction of the subsequent steps, for $t = T_{obs} + 1, \ldots, T_{pred}$

$$\mathbf{h}^d_{T_{obs}} = \mathbf{h}^e_{T_{obs}}, \tag{3.29}$$

$$\mathbf{h}^d_t = \mathcal{L}_{dec}(\mathbf{h}^d_{t-1}, \mathbf{e}_t; \mathbf{W}_{dec}) \text{ for } t = T_{obs}+1, \dots T_{pred}, \tag{3.30}$$

$$[\hat{\delta}^T_t, \hat{\kappa}^T_t]^T = \mathbf{W}_{out}\mathbf{h}^d_t \text{ for } t = T_{obs}+1, \dots T_{pred}, \tag{3.31}$$

$$\mathbf{e}_t = \mathbf{W}_{emb}\hat{\delta}_t, \tag{3.32}$$

where the encodings $\mathbf{e}_t$ for $t > T_{obs}$ encode the previously predicted $\hat{\delta}_t$, in testing, or the ground truth $\delta_t \triangleq \mathbf{p}_t - \mathbf{p}_{t-1}$ for $t > T_{obs}$ (principle of *teacher forcing*), in training. The outputs $\hat{\kappa}_t \in \mathbb{R}^3$ (Eq. 3.31) are used to define a 2×2 variance matrix $\hat{\Sigma}_t$ over the result, so that the parameter optimization (corresponding to Eq. 3.2) is written as the minimization of the following log-likelihood

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{k=1}^{K} \sum_{t=T_{obs}+1}^{T_{pred}} -\log \mathcal{N}\left(\mathbf{p}_{k,t}; \mathbf{p}_{k,T_{obs}} + \sum_{\tau=1}^{t} \hat{\delta}^{\mathbf{w}}_{k,\tau}, \sum_{\tau=1}^{t} \hat{\Sigma}^{\mathbf{w}}_{k,\tau}\right), \tag{3.33}$$

where k spans the indices in our training dataset (or mini-batch), that contains K data. This deterministic model has as its parameters $\mathbf{w} \triangleq (\mathbf{W}_{emb}, \mathbf{W}_{enc}, \mathbf{W}_{dec}, \mathbf{W}_{out})$.

Finally, to reconstruct the trajectory, for $t > T_{obs}$ we simply apply

$$\hat{\mathbf{p}}_t = \mathbf{p}_{T_{obs}} + \sum_{\tau=1}^{t} \hat{\delta}_\tau, \tag{3.34}$$

and the corresponding predictive model given as the following density

$$f(\mathbf{p}_t) = \mathcal{N}\left(\mathbf{p}_t; \mathbf{p}_{T_{obs}} + \sum_{\tau=1}^{t} \hat{\delta}_\tau(\mathbf{p}_{1:T_{obs}}), \sum_{\tau=1}^{t} \hat{\Sigma}_\tau(\mathbf{p}_{1:T_{obs}})\right). \tag{3.35}$$

3.5.1.2 Bayesian Variants

In addition to this base deterministic model, we consider several *Bayesian variants* described hereafter:

- *Deep ensembles* [21]: Ensembles can be considered as a form of Bayesian Model Averaging [11, 40]. We train N instances of the base model described above by using different initialization values for the parameters $\mathbf{w}$. This way, we obtain a set $\{\mathbf{w}_i\}_{1 \le i \le N}$ of network parameters that approximate the posterior $p(\mathbf{w}|\mathbf{D})$. Obtaining such an ensemble gives a simple way to explore the landscape of the objective function of Eq. 3.33 but at a rather high cost, since it also implies to execute N different training processes.
- *Variational Bayesian Inference* [2]: the posterior $p(\mathbf{w}|\mathbf{D})$ is modeled through a Gaussian distribution with diagonal variance matrix. At training, we learn the mean and variance of each weight by maximizing the likelihood of the ground

truth data under the predictive distribution given by Eq. 3.35. The evaluation of this likelihood is done by sampling only one set of weights from the current posterior.

- *Dropout* [5]: It extends the use of the classical dropout technique (used typically as a regularizer) as a very easy way to produce samples from the posterior $p(\mathbf{w}|\mathbf{D})$, i.e. by sampling multiple Bernoulli distributions associated to the weights. During training, 5 samples are used to evaluate the likelihood.

Note that, in the three cases above, we use the sampled weights (i.e. instances of our networks) to produce the posterior predictive distribution as the probability density function required in the developments we proposed in Sect. 3.4. For an input $\mathbf{x}_k$ and given a training dataset $\mathbf{D}$, it is defined as

$$f_k(\mathbf{y}) \approx \frac{1}{N} \sum_{i=1; \mathbf{w}_i \sim p(\mathbf{w}|\mathbf{D})}^{N} p\left(\mathbf{y} \mid \mathbf{x}_k, \mathbf{w}_i\right), \tag{3.36}$$

where $\mathbf{y} \triangleq \mathbf{p}_{T_{pred}} \in \mathbb{R}^2$ is the last predicted position. This means that, at testing times, the computational complexity is multiplied by N. In the case of our simple prediction network, this is not a problem in practice for values of N inferior to 10.

3.5.2 Implementation Details

For our experiments, we have used the five sub-datasets forming the ETH-UCY dataset [23, 29], which is a standard benchmark in HTP. These sub-datasets are referred to as ETH-Univ, ETH-Hotel, UCY-Zara01, UCY-Zara02, UCY-Univ. In all our experiments, we use the Leave-One-Out methodology: Four of the sub-datasets provide the training/validation data and the remaining one is used for testing.

The base model described above, and its Bayesian variants, is implemented with $T_{obs} = 8, T_{pred} = 12$, embeddings of dimension $d_e = 128$ and hidden size for LSTMs $d_c = 256$. The mini-batch size was chosen as 64. Early stopping was used on the validation data to stop training.

All models are implemented with PyTorch [28]. The implementation of the ensembles does not require additional software layers. This is not the case for the other two Bayesian techniques (Variational and Dropout), that require significant modifications to the standard code. Hence, we have make use of the Bayesian-Torch [19] library that implements Bayesian versions of the neural networks required for our model (LSTMs and Dense layers). All Bayesian variants have been used with $N = 5$.

We refer the reader to our code repository[1] for more implementation details.

[1] https://github.com/cimat-ris/trajpred-bdl.

Table 3.1 Negative log-likelihoods (lower is better) over the different sub-datasets for deterministic, ensemble, dropout and variational models

	Deterministic	Ensemble	Dropout	Variational
Eth-Hotel	14.96	10.92	6.84	6.86
Eth-Univ	9.02	12.16	8.04	8.00
Ucy-Zara01	16.77	12.33	8.99	10.79
Ucy-Zara02	22.06	12.80	9.54	8.65
Ucy-Univ	15.87	10.70	7.30	7.08

Table 3.2 Calibration metrics before applying the calibration process (lower is better). Deterministic model vs. Dropout as the Bayesian approach

	MACE		RMSCE		MA	
	Det.	Bayes	Det.	Bayes	Det.	Bayes
Eth-Hotel	0.126	0.120	0.144	0.152	0.133	0.126
Eth-Univ	0.069	0.044	0.081	0.059	0.071	0.045
Ucy-Zara01	0.063	0.082	0.075	0.113	0.066	0.086
Ucy-Zara02	0.110	0.079	0.145	0.109	0.114	0.082
Ucy-Univ	0.099	0.104	0.115	0.112	0.104	0.109

3.5.3 Evaluation of the Uncertainties Quality

First, we compare the quality of the uncertainties coming from the deterministic network and from its Bayesian counterparts. To do so, we evaluate, for each sub-dataset, the average Negative Log Likelihood (NLL) of the ground truth data over the output predictive distribution. This likelihood is a Gaussian in the deterministic case, corresponding to Eq. 3.35, and a mixture of Gaussians in the Bayesian variants, corresponding to Eq. 3.36. The results are presented in Table 3.1. As it can be seen, the Bayesian variants tend to give significantly lower values for the NLL, which indicates that they represent more faithfully the uncertainties over their outputs.

Then, we compare the uncertainty calibration quality between the outputs of the deterministic model and of a Bayesian model, here the Dropout implementation.

To describe the quality of the calibration quantitatively as a numerical score, we use the mean absolute calibration error (MACE), root mean squared calibration error (RMSCE) and miscalibration area (MA). These metrics provide an estimate of the level of predictive error, and the lower the better. For MA, the integration is taken by tracing the area between curves and in the limit of number discretizations for the probability space $[0, 1]$ of $1 - \alpha$, MA and MACE will converge to the same value.

In Table 3.2, we compare the values of these three metrics, before applying the calibration process, between the deterministic algorithm (Sect. 3.5.1.1) and the Dropout Bayesian variant. In a majority of cases, before the calibration process is applied, the Bayesian variant brings a better calibrated uncertainty model.

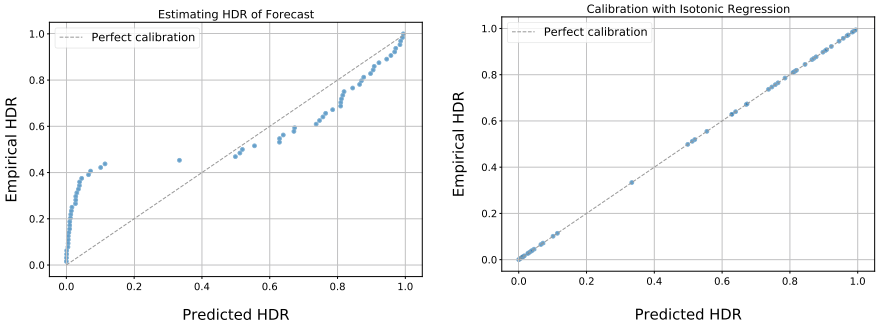

Fig. 3.4 Illustration of the calibration process under a LOO scheme on the ETH-UCY dataset, with ETH-Hotel as the testing data: On the left is the dataset $\{\hat{\alpha}_t, \hat{P}_\alpha(\hat{\alpha}_t)\}_{t=1}^{T}$, estimated as described in Sect. 3.4.3, that we use for calibration. Each blue dot corresponds to one of these t. The $\hat{\alpha}_t$, on the x axis tells us the level of confidence measured for ground truth data over the predicted distribution, while the $\hat{P}_\alpha(\hat{\alpha}_t)$ on the y axis gives an empirical estimate of the proportion of cases for which this level $\hat{\alpha}_t$ is passed. For the data presented here the system is under-confident for values of $\alpha < 0.5$ and over-confident for values of $\alpha > 0.5$

3.5.4 Evaluation of the Re-calibration Process

We follow the calibration process proposed in Sect. 3.4.3 and use the three Bayesian variants described above to perform the experiments. In Fig. 3.4, we depict an example of how the calibration is applied on the ETH-UCY dataset, with ETH-Hotel as the testing data. The left side shows the raw calibration dataset $\mathbf{C}$ (see Algorithm 3.2) before calibration, and the right side depicts the same data after applying calibration, i.e. determining the transformation γ.

Then, in Figs. 3.5, 3.6 and 3.7, we show the application of the function γ on the data that are not part of the calibration dataset. Here the curves are constructed by sampling regularly α and by evaluating $\hat{\mathbb{P}}_\alpha(\alpha)$ according to Eq. 3.23. As it can be seen, the calibration process results in having well-calibrated uncertainties, in the sense that for any $\alpha \in [0, 1]$, the proportion of ground-truth data falling into the $100(1-\alpha)\%$-HDR is roughly $1-\alpha$. Note that, before calibration, the purple curves indicate that in most of the cases (excepting UCY-Zara02), the uncertainties before calibration were in general under-estimating the true uncertainties. This means that, for a given value of α, the rate of ground truth points $\mathbf{y}$ at which the output p.d.f. satisfies $f(\mathbf{y}) > f_\alpha$ is significantly lower than $1-\alpha$ (i.e. the purple curve is well under the diagonal).

Finally, in Tables 3.3 and 3.4, we depict the effect of this re-calibration process on the calibration metrics introduced above, for the case of ensembles and dropout Bayesian variants. In both cases, the models end up very well calibrated.

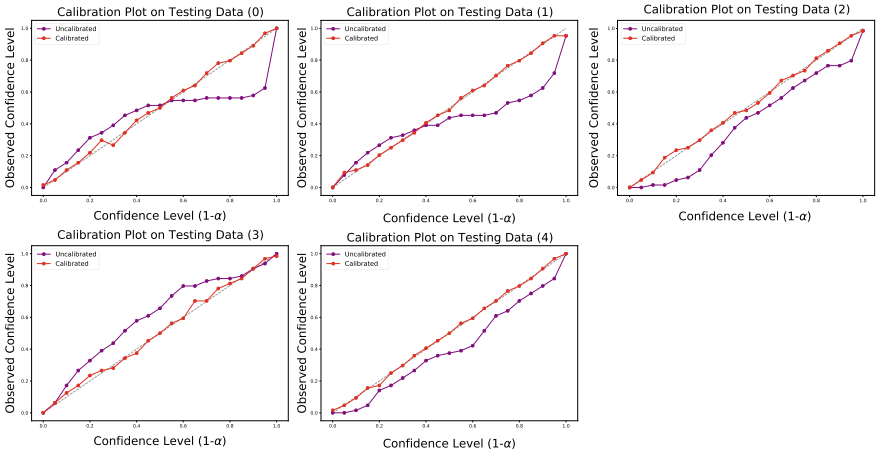

Fig. 3.5 Application of the calibration process with the Ensemble-based Bayesian approach (see Sect. 3.5.1.2) on the 5 datasets considered in the Leave-One-Out testing process: from left to right and top to bottom, ETH-Univ, ETH-Hotel, UCY-Zara01, UCY-Zara02, UCY-Univ. On the x axis is the confidence level $1 - \alpha$, sampled regularly on the [0, 1] interval, and on the y axis is the observed proportion of data that correspond to this confidence region R_α

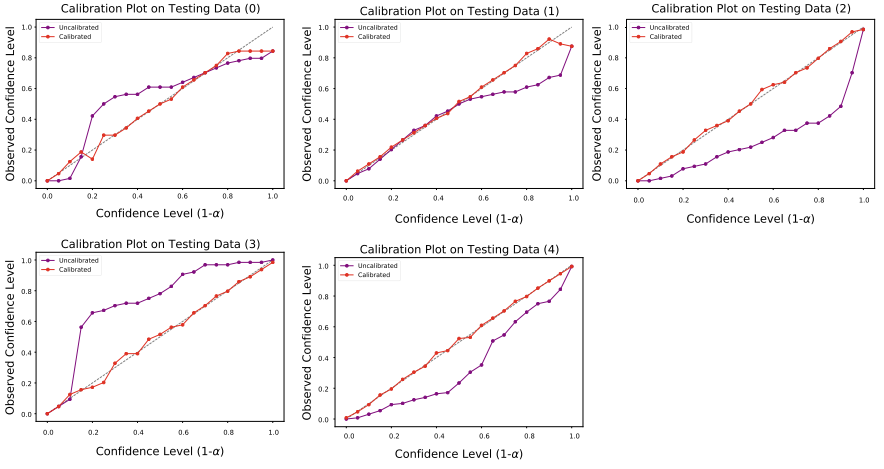

Fig. 3.6 Application of the calibration process with the Variational Bayesian approach (see Sect. 3.5.1.2) on the 5 datasets considered in the Leave-One-Out testing process: from left to right and top to bottom, ETH-Univ, ETH-Hotel, UCY-Zara01, UCY-Zara02, UCY-Univ. On the x axis is the confidence level $1 - \alpha$, sampled regularly on the [0, 1] interval, and on the y axis is the observed proportion of data that correspond to this confidence region R_α

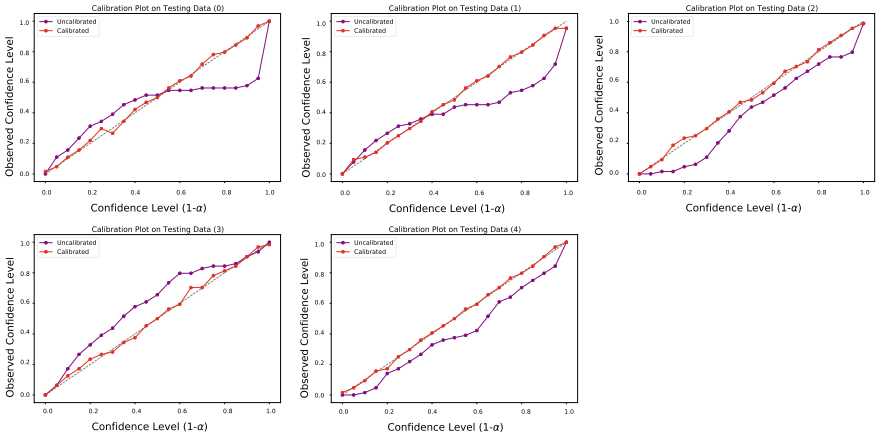

Fig. 3.7 Application of the calibration process with the Dropout Bayesian approach (see Sect. 3.5.1.2) on the 5 datasets considered in the Leave-One-Out testing process: from left to right and top to bottom, ETH-Univ, ETH-Hotel, UCY-Zara01, UCY-Zara02, UCY-Univ. On the x axis is the confidence level $1 - \alpha$, sampled regularly on the $[0, 1]$ interval, and on the y axis is the observed proportion of data that correspond to this confidence region R_α

Table 3.3 Calibration metrics before/after calibration (lower is better), for Deep ensembles

	MACE	RMSCE	MA
Eth-Hotel	0.146/0.017	0.174/0.020	0.152/0.015
Eth-Univ	0.069/0.014	0.081/0.018	0.072/0.012
Ucy-Zara01	0.076/0.019	0.092/0.026	0.080/0.017
Ucy-Zara02	0.122/0.019	0.162/0.024	0.128/0.016
Ucy-Univ	0.140/0.011	0.156/0.013	0.147/0.008

Table 3.4 Calibration metrics before/after calibration (lower is better) for Dropout

	MACE	RMSCE	MA
Eth-Hotel	0.120/0.013	0.152/0.018	0.126/0.011
Eth-Univ	0.044/0.011	0.059/0.014	0.045/0.008
Ucy-Zara01	0.082/0.015	0.113/0.018	0.086/0.013
Ucy-Zara02	0.079/0.020	0.109/0.026	0.082/0.017
Ucy-Univ	0.104/0.015	0.112/0.020	0.109/0.013

3.6 Conclusions

We have presented in this Chapter what is, to the best of our knowledge, the first intent to introduce Bayesian Deep Learning in the problem of Human Trajectory Prediction. We use it in order for the prediction system to cope not only with aleatoric uncertainties over its prediction, but also with epistemic uncertainties related, among others, to the finiteness of the training data. We have also proposed a novel approach to calibrate the resulting uncertainties by using the concept of Highest Density Region; we have shown that this re-calibration process is simple and produces well calibrated uncertainties.

Our future work includes the handling of aleatoric and epistemic uncertainties for more complex networks, i.e. networks using more advanced kinds of architectures (e.g. attention-based networks instead of recurrent networks) and using more contextual data (e.g., map of the surrounding area).

Acknowledgements The authors acknowledge the support of CYTED Network "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559), of Conacyt through its project funding 319584 and of Intel through its "Probabilistic computing" Science and Technology Center.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: human trajectory prediction in crowded spaces. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 961–971 (2016)
2. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: Proceedings of the International Conference on Machine Learning, pp. 1613–1622 (2015)
3. Burstedde, C., Klauck, K., Schadschneider, A., Zittartz, J.: Simulation of pedestrian dynamics using a two-dimensional cellular automaton. Physica A **295**(3–4), 507–525 (2001)
4. Chen, T., Fox, E., Guestrin, C.: Stochastic gradient hamiltonian monte carlo. In: Proceedings of the International Conference on Machine Learning, pp. 1683–1691 (2014)
5. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation. In: Proceedings of the International Conference on Machine Learning (ICML), vol. 3, pp. 1661–1680 (2016)
6. Giuliari, F., Hasan, I., Cristani, M., Galasso, F.: Transformer networks for trajectory forecasting. In: Proceedings of the IEEE International Conference on Pattern Recognition (ICPR), pp. 10335–10342 (2020)
7. Gneiting, T., Raftery, A.E.: Weather forecasting with ensemble methods. Science **310**(5746), 248–249 (2005)
8. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Bing, X., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
9. Graves, A.: Practical variational inference for neural networks. In: Advances in Neural Information Processing Systems, pp. 2348–2356 (2011)
10. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social gan: socially acceptable trajectories with generative adversarial networks. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2255–2264 (2018)

11. He, B., Lakshminarayanan, B., Teh, Y.W.: Bayesian deep ensembles via the neural tangent kernel. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1010–1022 (2020)
12. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**(5), 4282–4286 (1995)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
14. Hyndman, R.J.: Computing and graphing highest density regions. Amer. Stat. **50**, 120–126 (1996)
15. Ivanovic, B., Pavone, M.: The trajectron: probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2375–2384 (2019)
16. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? In: Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS), pp. 5580–5590. Curran Associates Inc. (2017)
17. Kong, J., Pfeiffer, M., Schildbach, G., Borrelli, F.: Kinematic and dynamic vehicle models for autonomous driving control design. In: 2015 IEEE Intelligent Vehicles Symposium (IV), pp. 1094–1099 (2015)
18. Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I.D., Rezatofighi, H., Savarese, S.: Social-bigat: multimodal trajectory forecasting using bicycle-gan and graph attention networks. In: Advances in Neural Information Processing Systems, pp. 137–146 (2019)
19. Krishnan, R., Esposito, P., Subedar, M.: Bayesian-torch: Bayesian neural network layers for uncertainty estimation (2022). https://github.com/IntelLabs/bayesian-torch
20. Kuleshov, V., Fenner, N., Ermon, S.: Accurate uncertainties for deep learning using calibrated regression. In: Proceedings of the International Conference on Machine Learning (ICML), vol. 80, pp. 2801–2809. PMLR (2018)
21. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: Proceedings of the International Conference on Neural Information Processing Systems, NIPS'17, pp. 6405–6416. Curran Associates Inc. (2017)
22. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H.S., Chandraker, M.K.: DESIRE: distant future prediction in dynamic scenes with interacting agents (2017). arXiv:1704.04394
23. Lerner, A., Chrysanthou, Y.L., Lischinski, D.: Crowds by example. Comput. Graph. Forum **26**, 655–664
24. Lisotto, M., Coscia, P., Ballan, L.: Social and scene-aware trajectory prediction in crowded spaces. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCV), pp. 2567–2574 (2019)
25. MacKay, D.J.C.: A practical bayesian framework for backpropagation networks. Neural Comput. **4**(3), 448–472 (1992)
26. Maddox, W.J., Izmailov, P., Garipov, T., Vetrov, D.P., Wilson, A.G.: A simple baseline for bayesian uncertainty in deep learning. In: Advances in Neural Information Processing Systems, pp. 13153–13164 (2019)
27. Neal, R.M.: Bayesian learning for neural networks. Ph.D. Thesis, University of Toronto (1995)
28. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates, Inc. (2019)
29. Pellegrini, S., Ess, A., Schindler, K., van Gool, L.: You'll never walk alone: modeling social behavior for multi-target tracking. In: Proceedings of IEEE International Conference on Computer Vision, pp. 261–268 (2009)
30. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Adv. Large Margin Class. **10**(3) (1999)
31. Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., Savarese, S.: Sophie: an attentive gan for predicting paths compliant to social and physical constraints. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1349–1358 (2019)

32. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: dynamically-feasible trajectory forecasting with heterogeneous data. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 683–700, Springer, Berlin, Heidelberg (2020)
33. Sohn, K., Yan, X., Lee, H.: Learning structured output representation using deep conditional generative models. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, NIPS'15, pp. 3483–3491. MIT Press (2015)
34. Tay, C., Laugier, C.: Modelling smooth paths using Gaussian processes. In: Proceedings of the International Conference on Field and Service Robotics, Chamonix, France (2007)
35. Trautman, P., Ma, J., Murray, R.M., Krause, A.: Robot navigation in dense human crowds: statistical models and experimental studies of human-robot cooperation. Int. J. Robot. Res. **34**(3), 335–356 (2015)
36. Varshneya, D., Srinivasaraghavan, G.: Human trajectory prediction using spatially aware deep attention models (2017). arXiv:1705.09436
37. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
38. Vemula, A., Muelling, K., Oh, J.: Social attention: modeling attention in human crowds. In: Proceedings of the International Conference on Robotics and Automation (ICRA), pp. 1–7 (2018)
39. Welling, M., Teh, Y.W.: Bayesian learning via stochastic gradient langevin dynamics. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 681–688 (2011)
40. Wilson, A.G., Izmailov, P.: Bayesian deep learning and a probabilistic perspective of generalization. In: Advances in Neural Information Processing Systems, vol. 33, pp. 4697–4708 (2020)
41. Zhang, R., Li, C., Zhang, J., Chen, C., Wilson, A.G.: Cyclical stochastic gradient mcmc for bayesian deep learning. In: International Conference on Learning Representations (2020)
42. Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C.L., Zhao, Y., Wang, Y., Wu, Y.N.: Multi-agent tensor fusion for contextual trajectory prediction. In: Proceedings of the IEEE/CVF Conerence on Computer Vision and Pattern Recognition (CVPR), pp. 12126–12134 (2019)

Chapter 4
Automatic Detection of Knives in Complex Scenes

Maira Moran, Aura Conci, and Ángel Sánchez

Abstract Smart Cities use a variety of Information and Communication Technologies (ICT) and databases to improve the efficiency and efficacy of city services. Security is one of the main topics of interest in this context. The increase in crime rates demands the development of new solutions for detecting possible violent situations. Video surveillance (CCTV) cameras can provide a large amount of valuable information contained in images which can be difficult to be analyzed by humans in an efficient form. Identifying and classifying weapons in such images is a challenging problem that can be driven by the application of Deep Learning techniques. Object detection algorithms, especially advanced Machine Learning ones, have demonstrated impressive results in a wide range of applications. However, they can fail in certain application scenarios. This work describes a novel proposal for knife detection in complex images. This is a challenging problem due to the multiple variabilities of these objects in scenes (i.e., changing shapes, sizes and illumination conditions, among others), which can negatively impact the performance of mentioned algorithms. Our approach analyzed the combination two super-resolution techniques (as a preprocessing stage) with one object detection network to effectively solve the considered problem. The results of our experiments show that the proposed methodology can produce better results when detecting small objects having reflecting surfaces (i.e., knives) in scenes. Moreover, the approach could be adapted for surveillance applications that need real-time detection of knives in places monitored by cameras.

M. Moran · A. Conci
IC - UFF, Niterói-RJ 24210-310, Brazil
e-mail: mhernandez@id.uff.br

A. Conci
e-mail: aconci@ic.uff.br

Á. Sánchez (✉)
ETSII - URJC, 28933 Móstoles (Madrid), Spain
e-mail: angel.sanchez@urjc.es

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_4

4.1 Introduction

New Smart City (SC) technologies are helping cities to maximize their resources and increase efficiencies in all facets of urban life. A SC consists of an urban space where Information and Communication Technologies (ICT) are extensively used to improve the quality and performance of services such as transportation, energy, water or infrastructures, in order to reduce resource energy consumption, wastage and overall costs [11].

One of the relevant areas in the SC is guarantying the security of their citizens. Video surveillance CCTV cameras, which are commonly used by urban police departments, can be part of these "smart" technologies in combination with video analytics software. Video recordings contain a wealth of valuable information that can be automatically analyzed to detect anomalous (and even dangerous) events from multiple cameras. Commonly, in security centers work human operators that are in charge of a large number of CCTV cameras, capturing multiple city views, operating in real-time. Due to the difficulty of humans for being able to keep their attention during several hours in front of many cameras (usually, more than 16), it is desirable that the video surveillance system could be automatically able to recognize potentially critical security events in specific video frames and cameras. In such cases, the system can notify an alert to the human operators to focus his/her attention on a concrete camera. Image-content analytics technology can help solving the event detection problem by processing video frames and identifying, classifying and indexing some types of targets objects (e.g., cars, motorcycles, persons or animals) [19]. Driven by Artificial Intelligence techniques, surveillance software can also make these images (or frames) in videos as searchable, actionable and quantifiable.

In this context, this work presents a study of applying deep networks to the problem of automatically detecting knives (and related objects) in images. This is a challenging problem due to the multiple variabilities of these targets when appearing in scenes. In particular, the changing shapes of knives, their relatively small sizes in images, the possibility of being partially occluded, being carried by a person (or being free) in a location, the changing illumination conditions in scenes, among other difficulties. All these involved variabilities (which can also appear combined), can produce a negative impact over the performance of the detection algorithms. The extension of this work to detect firearms like guns would not be difficult, since the used models are configurable for including additional object classes.

This paper describes a research on the application of combining super-resolution techniques with deep neural networks to effectively handle the knife detection problem in complex images. Our results show that the proposed methodology produces accurate results when detecting this special type of objects.

This paper is organized as follows. Section 4.2 summarizes the related work on the considered knife detection problem. The aspects of small-object detection (and, in particular, knives), as well as the description of the YOLOv4 model used in this work, are described in Sect. 4.3. In Sects. 4.4 and 4.5, we respectively describe the dataset

used in the experiment and some related pre-processing on it. The experiments carried out and their analysis appear in Sect. 4.6. Finally, Sect. 4.7 concludes this work.

4.2 Related Work

The problem of small-sized object detection in labeled datasets is still not solved at all [16]. In this problem, very few image pixels represent the whole objects of interest, which make it difficult to detect and classify them. The use of super-resolution to increase the object size in order to compensate for the loss of object information can help to the detection task [17].

One specific use case of small-sized object detection consist in the detection of knives. As for other types of weapons, carrying knives in public is either forbidden or restricted in many countries. Since knives are both widely available and can be used as weapons, their detection is of high importance for security personnel [8].

One of the first works on automatic detection of knives was presented by Kmiec and Glowacz in 2011 [12]. These authors compute a set of image descriptors using Histograms of Oriented Gradients (HOG). These descriptors, that are invariant to geometric and photometric transformations, are used with a SVM for the detection task.

Glowacz and collaborators [8] propose an Active Appearance Model (AAM) to detect knives in images. As the knife-blade has usually an uniform texture, using an AAM could contribute to improve detections, since the model would not converge to other objects having a similar shape.

In 2016 Grega et al. [10] publish a highly-cited work on detection of firearms and knives from CCTV images. Their goal is to reduce the number of false alarms in detections. These authors use a modified sliding window technique to determine the approximate position of the knife in an image. Then, they extract edge histograms and texture descriptors to create feature vectors for training a SVM able to classify the detected objects as knives.

Buckchash and Raman [2] have proposed in 2017 a method to detect visual knives in images. Their approach has three stages: foreground segmentation, feature extraction using the FAST (Feature Accelerated Segment Test) corner detector, and Multi-Resolution Analysis (MRA) for classification and target confirmation.

More recent works make use of deep networks. Castillo et al. [3] presented a system to locate cold steel weapons in images. (such as knives). These weapons have a reflecting surface that under different light conditions can distort and/or blur their shape in the frames. To solve the problem, the authors propose the combination of a contrast-enhancement brightness-guided preprocessing procedure with the use of different types of Convolutional Neural Networks (CNN).

Other authors have experimented with infrarred images (IR) to detect not visible (i.e., hidden) knives [18]. A type of deep neural network (GoogleNet), that was trained on natural images, was fine-tuned to classify the IR images as people or as people carrying a hidden knife.

A very comprehensive survey on the progress of Computer Vision-based concepts, methodologies, analysis and applications for automatic knife detection has been published recently showing the state-of-the-art of vision-based detection systems [4]. The authors define a taxonomy based on the state-of-the-art methods for knife detection. They analyzed several image features used in the considered works for this task. The challenges regarding weapon detection and new-frontier in weapon detection are included, as well. This survey references more then 80 works, and concludes pointing out some possible research gaps in the problem and related ones.

Another brief review of the state-of-the-art approaches of knife identification and classification was published very recently [5]. Although, this article is not a review paper, it presents a broad analysis of recent works using Convolutional Neural Network (CNN), Recurrent Convolutional Neural Network (R-CNN), Faster R-CNN, and Overfeat Network, that is most of deep learning methods used up now for the considered problem.

4.3 YOLOv4 Architecture for Detection of Knives

This section summarizes the object detection problem particularized for the case of knives, and the features of YOLOv4 model used in our experiments.

4.3.1 Detection of Knives

Object detection is a challenging task in Computer Vision that has received large attention in last years, especially with the development of Deep Learning [16, 19]. It presents many applications related with video surveillance, automated vehicle system robot vision or machine inspection, among many others. The problem consists in recognizing and localizing some classes of objects present in a static image or in a video. Recognizing (or classifying) means determining the categories (from a given set of classes) of all object instances present in the scene together with their respective network confidence values on these detections. Localizing consists in returning the coordinates of each bounding box containing any considered object instance in the scene. The detection problem is different from (semantic) instance segmentation where the goal is identifying for each pixel of the image the object instance (for every considered type of object) to which the pixel belongs. Some difficulties in the object detection problem include aspects such as geometrical variations like scale changes (e.g., small size ratio between the object and the image containing it) and rotations of the objects (e.g., due to scene perspective the objects may not appear as frontal); partial occlusion of objects by other elements in the scene; illumination conditions (i.e., changes due to weather conditions, natural or artificial light); among others but not limited to these ones. Note that some images may contain several combined variabilities (e.g., small, rotated and partially occluded objects). In addition

to detection accuracy, another important aspect to consider is how to speed up the detection task.

Detecting knives in images (and also in videos) is a challenging problem. The images where these objects can present several extrinsic and intrinsic variabilities due to the size of the target object (in general, its size ratio is very small when compared to the image size), the possibility of the weapon being carried by a person or appearing freely placed in a location, the illumination conditions of the scene (which could produce a very low contrast between the knife and the surrounding background), among other real difficulties.

4.3.2 YOLOv4

Redmon and collaborators have proposed in 2016 the new object detector model called YOLO (acronym of "You Only Look Once") [15], which handles the object detection as a one-stage regression problem by taking an input image and learning simultaneously the class probabilities and the bounding box object coordinates. This first version of YOLO was also called YOLOv1, and since them the successive improved versions of this architecture (YOLOv2, YOLOv3, YOLOv4, and YOLOv5, respectively) have gained much popularity within the Computer Vision community.

Different from previous two-stage detection networks, like R-CNN and faster R-CNN, the YOLO model used only one-stage detection. That is, it can make predictions with only one "pass" in the network. This feature made the YOLO architecture extremely fast, at least 1000 times faster than R-CNN and 100 times faster than Fast R-CNN.

The architecture of all YOLO models have some similar components which are summarized next:

- *Backbone*: A convolutional neural network that produces and accumulates visual features with different shapes and sizes. Classification models like ResNet, VGG, and EfficientNet are used as feature extractors.
- *Neck*: This component consists in a set of layers that receive the output features extracted by the Backbone (at different resolutions), and integrate and blend these characteristics before passing them on to the prediction layer. For example, models like Feature Pyramid Networks (FPN) or Path Aggregation networks (PAN) have been used for such purpose.
- *Head*: This component takes in features from the Neck along with the bounding box predictions. It performs the classification along with regression on the features and produces the bounding box coordinates to complete the detection process. Generally, it produces four output values per detection: the x and y center coordinates, and width and height of detected object, respectively.

Next, we summarize the main specific features of YOLOv4 architecture that were used in our experiments. YOLOV4 was released by Alexey Bochkovskiy et al. in

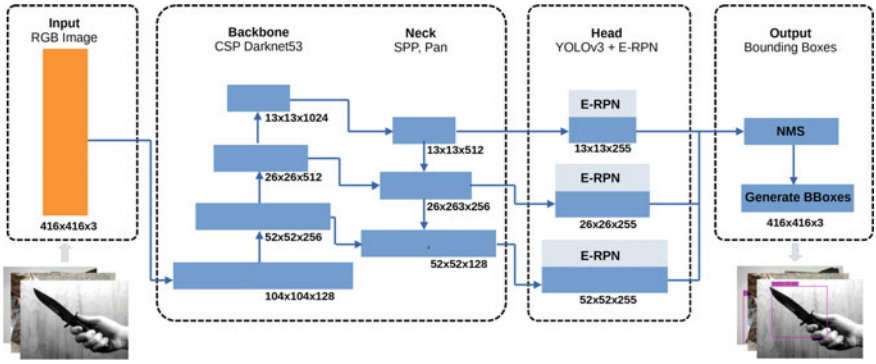

Fig. 4.1 Schematic representation of YOLOv4 architecture

their 2020 paper "YOLOv4: Optimal Speed and Accuracy of Object Detection" [1]. This model is ahead in performance on other convolutional detection models like EfficientNet and ResNext50. Like YOLOv3, it has the Darknet53 model as Backbone component. It has a speed of 62 frames per second with an mAP of 43.5% on the MS COCO dataset.

As technical improvements with respect to YOLOv3, YOLOv4 introduces as new elements the bag of freebies and the bag of specials.

Bag of Freebies (BoF) are a set of techniques enabling an improvement of the model in performance without increasing the inference cost. In particular:

- *Data augmentation techniques*: CutMix, MixUp, CutOut, …
- *Bounding box regression loss types*: MSE, IoU, CIoU, DIoU, …
- *Regularization techniques*: Dropout, DropPath, DropBlock, …
- *Normalization techniques*: Mini-batch, Iteration-batch, GPU normalization, …

Bag of Specials (BoS) consist in techniques that increase accuracy while slightly increasing the computation cost. In particular:

- *Spatial Attention Modules* (SAM): Spatial Attention (SA), Channel-wise Attention (CA), …
- *Non-Max Suppression modules* (NMS)
- *Non-linear activation functions*: ReLU, SELU, Leaky, Mish, …
- *Skip-Connections*: Weighted Residual Connections(WRC), Cross-Stage Partial connections (CSP), …

Figure 4.1 illustrates the layer structure of YOLOv4 network used in our experiments.

4.4 Datasets

The success of the proposed method is highly related to the quality of the data used to train the supervised algorithm. One of the main applications for the proposed problem is its inclusion in surveillance system. To our knowledge there are no current publicly-available CCTV datasets. The datasets used in similar works consist of images captured by the authors, and many of them are taken from the Internet. In this section, we present two main datasets in this field, which are also used in our work for training and testing the models.

4.4.1 DaSCI Dataset

The DaSCI knives dataset [14] is a subset of a more general weapon detection dataset. It was created by people from University of Granada as an open data repository, and designed for the object detection task. The annotation files describe the image region where each knife is located, by defining a correspondent bounding box. It is composed of 2,078 images, each one of them containing at least one knife, resulting a total of 2,155 objects. The dataset was created considering the diversity of the objects (i.e., the images were selected in order to provide samples with different visual features), resulting in a robust challenge dataset. Some considered visual features of knives are: types, shapes, colors, sizes, materials, locations, positions in relation to other scene objects, indoor/outdoor scenarios, and so on. The images were extracted mostly from the Internet, and the main sources were free image stocks and YouTube videos, from which frames were extracted, considering the criteria previously mentioned. The dataset is divided into 15 subsets (referred as DS1-DS15) according with their image sources. Each one is composed by: 8, 130, 16, 12, 188, 242, 11, 36, 49, 130, 603, 29, 143, 108, and 83 images, respectively. Table 4.1 summarizes the information about these subsets. Figure 4.2 shows some examples of images extracted from some of these sources.

As previously mentioned, the size, position and location if the objects varies in the dataset. This way, the area that the each knife covers in the image also differs (although it is often very small). Figure 4.3 shows histograms of these proportions. Even considering that the dataset was designed to present a high heterogeneity in this aspect, it can be observed that many of the objects (i.e., around 50%) only cover between 1 and 20% of the image size. The remaining objects are more equally distributed, occupying different portions of their respective images.

The fact that knives in this dataset tend to occupy a small area over the images (and consequently, present a low spatial resolution) is a challenging issue for the detection task, that can be assessed in the pipeline of possible solutions to be developed.

It is important to mention that the annotations are not completely uniform, in the sense that for some cases the knife area described in the annotation file covers the

Table 4.1 Information in DaSCI subsets

Source type	Video frames	DS1, DS2, DS3, DS4, DS5, DS6, DS7, DS8, DS9, DS12, DS13, DS14, DS15
	Internet images	DS11
	Captured by authors	DS10
Objects per image	One	DS1, DS2, DS3, DS4, DS5, DS6, DS7, DS8
	Multiple	DS9, DS10, DS11, DS12, DS13, DS14, DS15
Multiple scenarios	Yes	DS1, DS2, DS3, DS4, DS5, DS6, DS7, DS8, DS9
	No	DS10, DS11, DS12, DS13, DS14, DS15

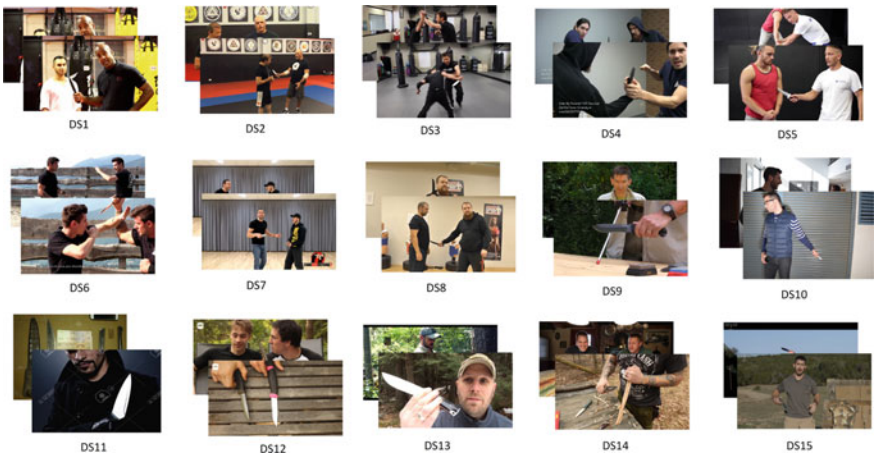

Fig. 4.2 Samples of each DaSCI subset

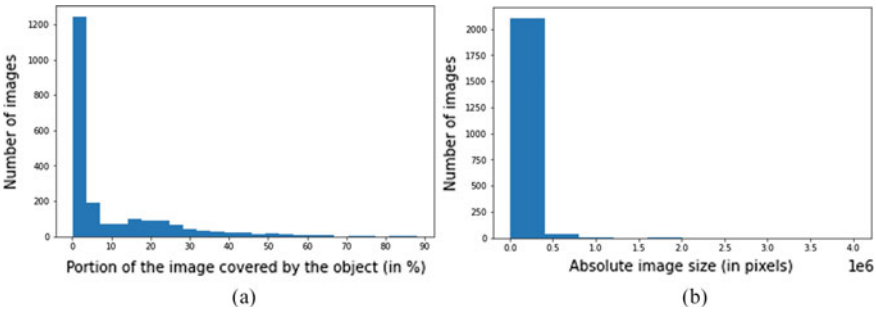

Fig. 4.3 Histogram of object sizes composing the knives samples in DaSCI dataset: **a** relative object vs image size proportions and **b** absolute object size (spatial resolution)

Fig. 4.4 Example of images that composed the DaSCI dataset and their respective annotations

Fig. 4.5 Example of images that composed the COCO dataset and their respective annotations

whole knife (i.e., both blade and handle), and for other cases the described knife are cover only the knife blade (Fig. 4.4).

The annotation formats describe each image and the positions of the associated objects. Firstly, the image information is detailed, including its file name, path and dimensions (width, height and depth, being this last one related to the number of channels, mostly 3 since RGB color images are used). Then, the information of objects is listed (always 'knife' in this work), and its respective region, which is described as a bounding box denoted by coordinates of its top left (x_{min}, y_{min}) and bottom right (x_{max}, y_{max}) corners.

4.4.2 MS COCO Dataset

The MS COCO (Microsoft Common Objects in Context) dataset [6] is widely used in Computer Vision literature for object detection and segmentation tasks. Since the appearance of its first version, other upgraded versions from this dataset have been published. In this work, we consider the MS COCO 2017 dataset. It consists of a very large and complete dataset, composed of 330,000 images with 1.5 million objects. This dataset has 80 different classes, and class 'knife' is one of them with 7,770 labeled objects from 4,326 images. Since the MS COCO dataset was initially designed to encompass objects of 80 different classes, the images selected to compose it mostly portrait scenes crowded with different objects, and knives are not the main object of interest in the scene. This can also be considered as a challenging issue for the problem assessed in this study. Figure 4.5 shows some samples of the MS COCO dataset.

Also, as similarly to DaSCI, in this dataset the knives mainly present a very low spatial resolution, which is another aspect to be handled in this study. Figure 4.6 shows an histogram of the object area vs image area ratio for the knives samples.

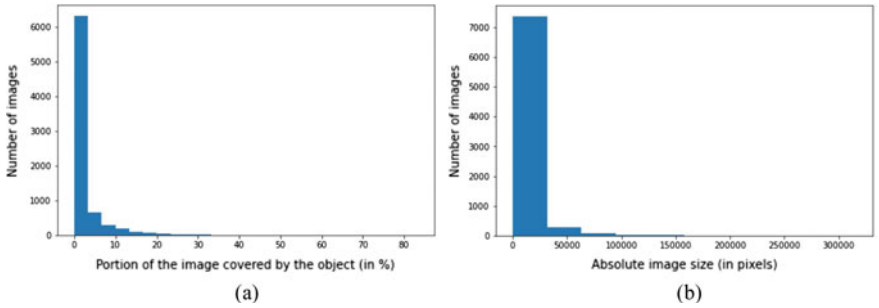

Fig. 4.6 Histogram of object sizes for the objects that compose the knives samples in MS COCO dataset: **a** relative object vs image size proportions and **b** absolute object size (spatial resolution)

The object bounding boxes in MS COCO annotations are described by the x and y coordinates of the top left corner, and the object's width and height, respectively.

4.4.3 Knife Classification Datasets

The knife detection task have been previously assessed in the literature (see survey work [4]). However, the number of public datasets available is still very limited. Regarding datasets that include knives in images, there are some available options that were initially proposed for classification tasks. Although their annotation should be expanded in order to be employed in a detection task, it is important to consider that such datasets are also available.

There is another dataset provided by DaSCI that could be employed for the knife classification task, composed of 10,039 images, which were extracted from the Internet. The annotations cover 100 object classes, being 'knife' the target one, with 635 images. Among the others classes included are: 'car', 'plant', 'pen', 'smartphone', 'cigar', etc.

Grega et al. [9] also proposed a method for knife classification. Their dataset consists of 12,899 images at 100×100 resolution, from which 9,340 are negative samples, and 3,559 are positive ones. The positive samples consist of a scene with a knife held in a hand, and the negative samples consists of scenes with no knife. Concerning the environment, the scenes in the images can be indoor and outdoor.

4.5 Pre-processings on Dataset

4.5.1 Dataset Preparation

In the YOLOv4 model each annotated file presents the following structure: object class, object coordinates (x and y), $width$ and $height$, separated by a simple space:

```
0 x y width height
```

In a YOLOv4 annotation file, each line corresponds to an object. An example of annotation in this format is shown next:

```
0 25 40 100 120
0 30 15 80 50
```

Note that each annotation file refers to an image, that contains one or more objects. In the example above, the first line describes the first object, that is the object class 'knife' (denoted by '0'). Also, the upper left corner of this first object's bounding box is in the position $x = 25$ and $y = 40$. Finally, this first object has a width of 100 and a height of 120. Similarly, for the second object in the example annotation.

As previously mentioned, the object regions in the DaSCI annotations are described as bounding boxes defined by the coordinates of the top left (x_{min}, y_{min}) and bottom right (x_{max}, y_{max}) corners. In this way, the values to compose these annotations can be easily calculated from the DaSCI annotations:

```
x = xmin
y = ymin
width = xmax-xmin
height = ymax-ymin
```

This way, YOLOv4 annotation obtained from the DaSCI XML annotation is composed of:

```
0 xmin ymin xmax-xmin ymax-ymin
```

As described in Sect. 4.5, the object's bounding box in the MS COCO annotation is also defined by the x and y coordinates of the upper left corner, and the object's width and height, so as in the YOLOv4 annotation format. The information to compose the annotations are directly transcribed from the MS COCO to a JSON annotation file. Note that, in this structure, each object annotation refers to an object, not to an image.

4.5.1.1 Image Pre-processing

The images to be used as input of the YOLOv4 algorithm must present a spatial resolution of 416×416. In this sense, the images of both MS COCO and DaSCI datasets must be resized to meet this condition. As previously mentioned, both datasets are composed by images with different sizes (i.e., spatial resolutions), so for some images the re-scale would result in an decrease of the image size, and for others this resizing would enlarge the original images. Increasing the image size, can be specially critic, since the methods commonly used for this task consist of interpolations that frequently lead to effects like blur, aliasing, etc., degrading the quality of the resulting image.

In order to observe the impact of the resizing part of the preprocessing, two alternative resizing operations were performed. The first one is bilinear interpolation, commonly used as a "black box" operation in most machine learning libraries, including the PyTorch Python library used in this work. The second one is SRGAN (Generative Adversarial Network for single image Super-Resolution) [13], which consists of a machine learning supervised algorithm. The SRGAN, more specifically one of its variations, is currently state of the art for some widely known challenges. Considering that the SRGAN uses a generative network G to create high-resolution images which are so similar to the original ones, that can mislead the differentiable discriminator D, which is trained to distinguish between the generated and the real super-resolution image. In this process, the D network demands an evolution of G during the training process, leading to perceptually superior solutions [13]. In this work, the SRGAN training was performed using the ImageNet dataset

On the other hand, the bilinear interpolation calculates the values of the new interpolated points based on a weighted mean of their surrounding points (four neighbors) in the original image. The weight assigned to each neighbor point is based on its distance to the new point. Consequently, the value of the new point is mostly influenced by the values of closer neighbors.

In this experiment, we analyze the impact of using super-resolution as a preprocessing step of the object detection algorithm. For such purpose, we have adopted a cross-dataset evaluation approach. Evaluations configured in an in-domain setting, which is defined by using the samples from the same dataset for training and testing the algorithms, tend to bias and affects negatively the generalization of machine learning algorithms. Moreover, the transfer learning technique was also assessed, as described in Sect. 4.5.3.

4.5.2 Dataset Variabilities

As previously mentioned, several factors can affect the performance of the proposed algorithms, as the illumination conditions, object size, perspective, visibility, etc. In this sense, we created subsets of interest from the original test set. Each of these

test sets presents an special condition, so one can observe how a particular condition affects to the results of the models. Next, the subsets are listed next:

1. Outdoor: it covers all the images that denote outdoors scenes, related mostly to a higher luminosity.
2. Indoor: composed of images that denote indoor scenes, mostly presenting a lower luminosity.
3. Occluded: composed of images in which the knives are being handled by a person, remaining partially occluded.
4. Not occluded: the object is lying on a surface and it is not held by anyone.

These subsets are not exclusive (i.e., the same image can belong to more than one subset), except when the conditions where defined subsets are excluding (e.g., subsets 1 and 2).

Also, the ratio between object size and image size is a factor that can affect the models' performance, specially considering that the use of super-resolution as pre-processing step may influence the results for small objects. As presented in the histogram of Sect. 4.5 (see Fig. 4.2), most of the objects that compose the DaSCI database, which is used as test set in our experiments, cover less than 20% of the corresponding images.

4.5.3 Transfer Learning

Along with the previously mentioned super-resolution pre-processing, another technique employed and analyzed in the performed experiments is transfer learning.

The transfer learning applied in this work consisted basically of using weights obtained from a task in a different domain to initialize the object detection algorithm before performing the actual training using the samples of the actual domain (in order to promote a faster convergence of the model). In this work, the initialization of weights was carried out by training a YOLOv4 algorithm using the Pascal VOC dataset. Until the 105th convolutional layer, the weights obtained by the transfer learning were used, and the remaining layers were re-trained using our final task.

The PASCAL VOC dataset [7] is widely used for supervised tasks such a as classification, detection and segmentation, being employed in benchmark comparisons for such tasks. It is composed of a wide range of images in realistic scenes. Their annotation associate them with twenty different classes. The class 'knife' is not present in this dataset. Three subsets compose it: train, validation and test. The first subset (train) is composed of 1.464 images, the validation set is composed of 1,449 images, and the test set consists of a private set. Figure 4.7 shows some examples of images from the PASCAL VOC dataset. Even considering that there are other image datasets widely known in literature, such as the ImageNet dataset, we decided to use the PASCAL VOC dataset since their annotations include bounding boxes designed for a detection task.

Fig. 4.7 Example of images that composed the PASCAL VOC dataset

4.6 Experimental Results

In this section, we present the results of performed experiments, which analyze the impact of using transfer learning and super-resolution techniques in the training process of the object detection network (YOLOv4). In Sect. 4.6.1, we summarize the metrics used in this analysis. Then, Sect. 4.6.2 presents the results of experiments, comparing the results obtained by using each of the mentioned techniques, in general and associated with different aspects of the test dataset such as object size, visibility and illumination.

4.6.1 Description of Performance Metrics

The evaluation is based on true positives TP (i.e., regions correctly detected as regions containing knives); false negatives FN (i.e., non detected regions containing knives); and false positives FP (i.e., regions incorrectly detected as regions containing knives). From these results some metrics can calculated such as Precision (Prec), Recall (Rec), and F1-Score, using the following equations:

$$Prec = \frac{TP}{TP + FP} \qquad Rec = \frac{TP}{TP + FN} \qquad F1 = 2\frac{Prec \times Rec}{Prec + Rec} \tag{4.1}$$

The Jaccard index or Intersection over Union (IoU) is also used in this analysis. This metric computes the areas of the bounding boxes denoting the detected knives and the corresponding ground truths.

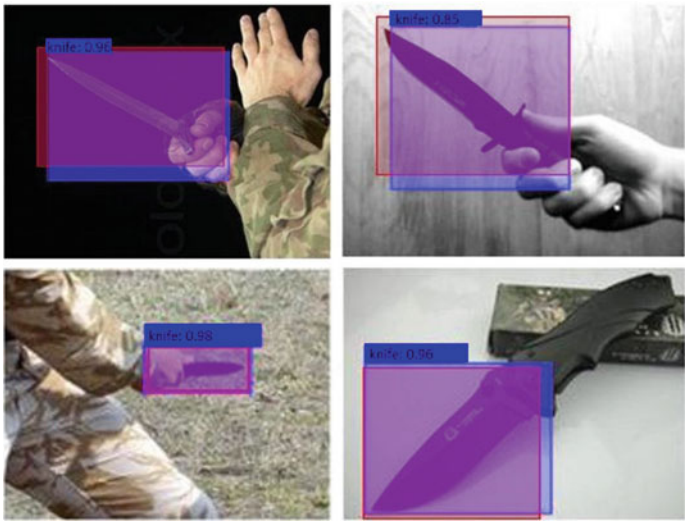

Fig. 4.8 Examples of bounding boxes: ground truth (violet) and algorithm result (blue)

Table 4.2 Training variations

Model	Training process	
	Tranfer learning	Pre-processing
M1	No	Bilinear interpolation
M2	Yes	Bilinear interpolation
M3	No	SRGAN
M4	Yes	SRGAN

Figure 4.8 exemplifies the mentioned IoU areas for several test images. The area in blue represents the bounding box obtained by one of the proposed algorithms, and the area in violet shows the bounding box defined by the ground truth.

4.6.2 Experimental Results

Next, we compare the results of different YOLOv4 models trained using the considered approaches. These training models are characterized as shown in Table 4.2.

Table 4.3 General results of the models

Model	TP	FP	FN	IoU (mean)	Precision	Recall	F1-Score
M1	2,057	143	98	0.776	0.935	0.955	0.945
M2	1,071	774	1,084	0.269	0.580	0.497	0.535
M3	2,064	32	91	0.756	0.985	0.958	0.971
M4	727	1,211	1,428	0.141	0.375	0.337	0.355

4.6.3 General Results

As described in Sect. 4.5.1.1, we used the cross-dataset approach to train and test all the models. The test dataset (DaSCI) is composed of 2,078 images, which cover 2,155 objects. The main hyper-parameters used in the training process are: confidence prediction threshold = 0.25, IoU threshold = 0.5, and batch size = 1.

Table 4.3 shows the values obtained for the selected metrics considering the whole dataset. It is possible that not all models detected most of the objects. The best overall performance was achieved by M3. One can notice that the use of transfer learning promotes a worse overall performance for models M2 and M4, compared with M1 and M3. Also, the results suggest that using the super-resolution pre-processing affects the models performance in different ways depending on whether it is combined with transfer learning or not.

For the models not trained with transfer learning (M1 and M3), the SRGAN subtly improved the results, increasing the number of TP in 7 cases and reducing the number of FN in 7 cases. The number of FP was substantially reduced (−111 cases). On the other hand, for the models trained with transfer learning (M2 and M4), the results using SRGAN were substantially worse. This difference is of −344 (−32.12%) for TP, +437 (56.46%) for FP, and +344 (31.73%) for FN.

Concerning the other performance metrics, the M1 model presented the best average IoU values, and the M3 model presented the best Precision, Recall and F1-score values. In general, the use of the super-resolution pre-processing had a negative impact in both metrics.

The differences in the IoU values achieved by each model can also be observed in the histograms presented in Fig. 4.9, where it is possible to observe that models M1 and M3 achieved IoU values that lay in mostly in the 70–100% interval. On the other hand, the IoU values that models M2 and M4 achieved lay in mostly in the 1–20% interval.

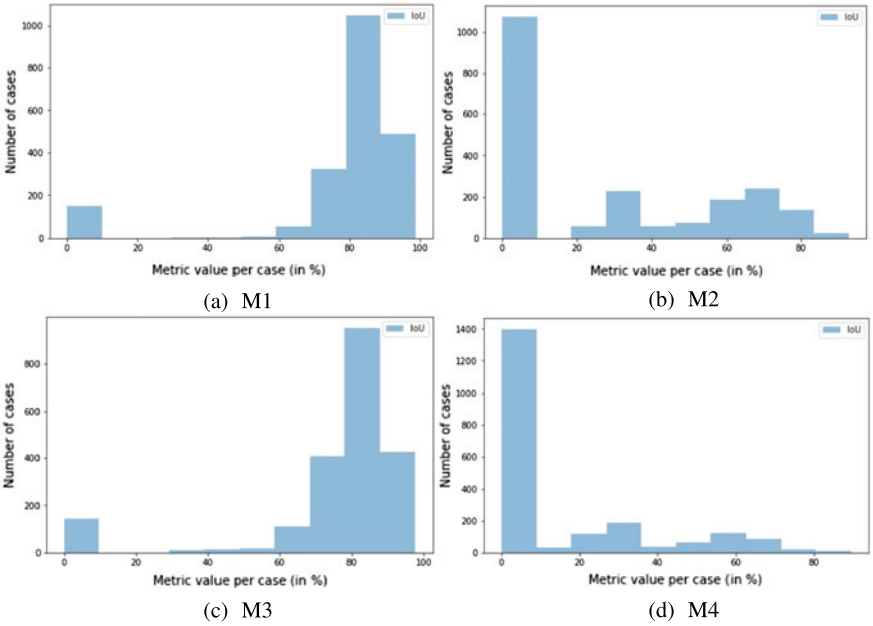

Fig. 4.9 Histograms of the IoU distributions achieved by each model

4.6.4 Results Considering Variabilities in Images

4.6.4.1 Results Considering the Sizes of Objects

The plots presented in Fig. 4.10 shows the performance variations associated to the relative object size in their respective images. As mentioned in Sect. 4.5, most of objects in test set are very small in relation of their respective images. This way, the performance of the models for the relatively small objects represent a large part of the overall results. Also, it is expected that in real-world detection applications, such as surveillance videos, the objects would also cover a very small portion of the images. Therefore, the results for these cases are specially important in our assessment.

In Fig. 4.10, it is possible to observe that the performance of models M1 and M3 remains similar for most relative object sizes. On the other hand, model M2 and M4 present a better performance for objects having a relative size around less than 30% of the image.

4.6.4.2 Results Considering Partial Occlusions

Another factor that may affect the detection performance is occlusion, which in the considered context is defined by the object being handled by a person, whose hand

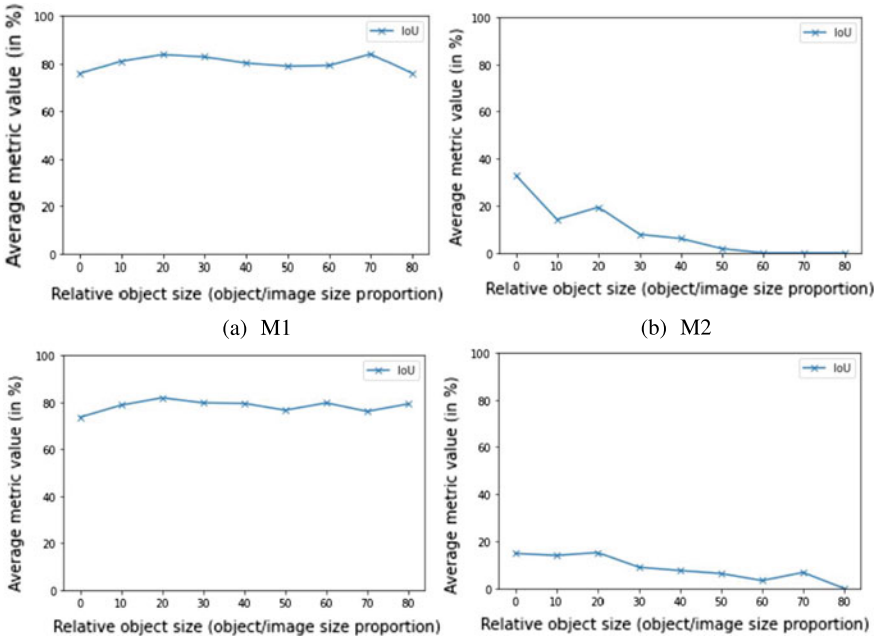

Fig. 4.10 IoU variations associated to relative object sizes for each model

Table 4.4 Results for partially occluded and visible knives

Model	Occlusion	TP	FP	FN	IoU	Precision	Recall	F1-Score
M1	No	226	3	41	0.811	0.987	0.846	0.911
M1	Yes	1,830	140	58	0.773	0.929	0.969	0.949
M2	No	40	4	227	0.115	0.909	0.154	0.263
M2	Yes	1,031	770	851	0.287	0.572	0.548	0.560
M3	No	236	9	31	0.805	0.963	0.884	0.922
M3	Yes	1,828	23	60	0.750	0.986	0.968	0.977
M4	No	74	94	193	0.131	0.440	0.277	0.340
M4	Yes	653	1,117	1,235	0.142	0.369	0.346	0.357

consequently occludes the knife blade. Table 4.4 compares the results between the portion of the dataset in which the objects are partially occluded as described, and the case in which the objects are completely visible (i.e., placed in some flat surface).

Note that results significantly differ, especially for the M2 model, which suggests that this aspect clearly affects the models performance. Overall, all models presented better results in cases in which the object was occluded. Similar to the overall trend pointed out for the general results, models trained without transfer learning achieved

Table 4.5 Models results for both indoor and outdoor cases

Model	Natural illumination	TP	FP	FN	IoU	Precision	Recall	F1-Score
M1	Indoor	981	139	55	0.732	0.876	0.947	0.910
M1	Outdoor	1,002	4	40	0.828	0.996	0.962	0.979
M2	Indoor	457	301	579	0.241	0.603	0.441	0.509
M2	Outdoor	576	473	466	0.301	0.550	0.553	0.551
M3	Indoor	992	15	44	0.717	0.985	0.958	0.971
M3	Outdoor	999	17	43	0.780	0.983	0.959	0.971
M4	Indoor	339	598	697	0.125	0.362	0.327	0.344
M4	Outdoor	361	613	681	0.159	0.371	0.346	0.358

better results, being M1 the best model for occluded objects and M3 the best model for non occluded objects.

4.6.4.3 Results Considering Natural Illumination

Finally, another factor considered in our evaluation is the natural illumination of the scene for each image. More specifically, we compare the models results for indoor and outdoor scenes, since this change of natural illumination may present some impact in the detection performance. Table 4.5 summarize these results.

According to the results, the natural illumination seems not to be a particular challenging factor for the detection models, since the results for all models tend to be similar for both indoor and outdoors scenes. It is possible to observe that the models achieved slightly better results with outdoor scenes. In contrast with the occlusion factor, the natural illumination variations is more equally represented in the test dataset (i.e., the number of images with indoor and outdoor scenes are relatively close).

4.7 Conclusion

In this work, we evaluated the performance of the YOLOv4 deep neural architecture for detecting knives in natural images. In the performed experiments, two other conditions were assessed: (a) the use of a super-resolution algorithm as pre-processing step and (b) the application of a transfer learning technique. The evaluation of results not only considers the whole test dataset, but also specific subsets, in order to evaluate if there are specific conditions that can affect the results, such as object sizes, natural illumination and partial occlusions. The results have shown that using a super-

resolution pre-processing algorithm only promotes better results if it is not combined with transfer learning. Moreover, the use of the proposed transfer learning technique reduced the overall performance of our YOLOv4 models.

In future works, we aim to evaluate other pre-processing techniques to be combined with new deep object detection approaches with the goal to achieve real-time processing performance, suitable for CCTV monitoring systems. Finally, we will also explore the classification aspect of object detection algorithms (i.e., including the detection of different classes of knives by considering their specific features), and extending this framework to detect also some types of firearms like guns.

Acknowledgements We acknowledge to the CYTED Network "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559). Ángel Sánchez acknowledges to the Spanish Ministry of Science and Innovation, under RETOS Programme, with Grant No.: RTI2018-098019-B-I00. Aura Conci and Maira Moran express their gratitude to FAPERJ, CAPES and CNPq Brazilian Agencies.

References

1. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: Yolov4: Optimal speed and accuracy of object detection (2020)
2. Buckchash, H., Raman, B.: A robust object detector: application to detection of visual knives. In: 2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW), pp. 633–638 (2017)
3. Castillo, A., Tabik, S., Pérez, F., Olmos, R., Herrera, F.: Brightness guided preprocessing for automatic cold steel weapon detection in surveillance videos with deep learning. Neurocomputing **330**, 151–161 (2019)
4. Debnath, R., Bhowmik, M.K.: A comprehensive survey on computer vision based concepts, methodologies, analysis and applications for automatic gun/knife detection. J. Vis. Commun. Image Represent. **79** (2021)
5. Dwivedi, N., Singh, D.K., Kushwaha, D.S.: Employing data generation for visual weapon identification using convolutional neural networks. Multimedia Syst. **28**(10), 347–360 (2022)
6. Lin, T.-Y., et al.: Microsoft coco: common objects in context. In: Computer Vision – ECCV 2014, pp. 740–755. Springer International Publishing (2014)
7. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
8. Glowacz, A., Kmieć, M., Dziech, A.: Visual detection of knives in security applications using active appearance models. Multimedia Tools Appl. **74**(12), 56416–56429 (2015)
9. Grega, M., Lach, S., Sieradzki, R.: Automated recognition of firearms in surveillance video. In: 2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), pp. 45–50 (2013)
10. Grega, M., Matiolański, A., Guzik, P., Leszczuk, M.: Automated detection of firearms and knives in a cctv image. Sensors **16**(1) (2016)
11. Khatoun, R., Zeadally, S.: Smart cities: concepts, architectures, research opportunities. Commun. ACM **59**(8), 46–57 (2016)
12. Kmiec, M., Glowacz, A.: An approach to robust visual knife detection. Mach. Graph. & Vis. Int. J. **20**(2), 215–227 (2011)
13. Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., Shi, W.: Photo-realistic single image super-resolution using a generative adversarial network (2017)

14. Olmos, R., Tabik, S., Herrera, F.: Automatic handgun detection alarm in videos using deep learning. Neurocomputing **275**, 66–72 (2018)
15. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: Unified, real-time object detection, You only look once (2016)
16. Tong, K., Wu, Y., Zhou, F.: Recent advances in small object detection based on deep learning: a review. Image Vis. Comput. **97**, 103910 (2020)
17. Wang, Z.-Z., Xie, K., Zhang, X.-Y., Chen, H.-Q., Wen, C., He, J.-B.: Small-object detection based on yolo and dense block via image super-resolution. IEEE Access **9**, 56416–56429 (2021)
18. Yuenyong, S., Hnoohom, N., Wongpatikaseree, K.: Automatic detection of knives in infrared images, pp. 65–68 (2018)
19. Zou, Z., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: a survey (2019). arXiv:1905.05055

Chapter 5
Human Body Pose Estimation in Multi-view Environments

Jorge L. Charco, Angel D. Sappa, Boris X. Vintimilla, and Henry O. Velesaca

Abstract This chapter tackles the challenging problem of human pose estimation in multi-view environments to handle scenes with self-occlusions. The proposed approach starts by first estimating the camera pose—extrinsic parameters—in multi-view scenarios; due to few real image datasets, different virtual scenes are generated by using a special simulator, for training and testing the proposed convolutional neural network based approaches. Then, these extrinsic parameters are used to establish the relation between different cameras into the multi-view scheme, which captures the pose of the person from different points of view at the same time. The proposed multi-view scheme allows to robustly estimate human body joints' position even in situations where they are occluded. This would help to avoid possible false alarms in behavioral analysis systems of smart cities, as well as applications for physical therapy, safe moving assistance for the elderly among other. The chapter concludes by presenting experimental results in real scenes by using state-of-the-art and the proposed multi-view approaches.

J. L. Charco (✉) · A. D. Sappa · B. X. Vintimilla · H. O. Velesaca
Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo Km., 30.5 Vía Perimetral, P.O. Box 09-01-5863 Guayaquil, Ecuador
e-mail: jlcharco@espol.edu.ec; jorge.charcoa@ug.edu.ec

A. D. Sappa
e-mail: asappa@espol.edu.ec; asappa@cvc.uab.es

B. X. Vintimilla
e-mail: boris.vintimilla@espol.edu.ec

H. O. Velesaca
e-mail: hvelesac@espol.edu.ec

J. L. Charco
Universidad de Guayaquil, Delta and Kennedy Av., Guayaquil, Ecuador

A. D. Sappa
Computer Vision Center, Campus UAB, 08193 Bellaterra, Barcelona, Spain

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_5

5.1 Introduction

During the last decade, several works have focused on the Human Pose Estimation (HPE) problem, which is usually tackled by detecting human body joints such as wrist, head, elbow, etc. to build the human body skeleton by the connection of all detected joints. The solutions proposed in the state-of-art are robust when all body joints are visible by the detector. However, when joints are occluded, due to moving objects in the scene (i.e., bicycles, cars) or natural human body pose self-occlusions, it becomes a challenging problem, particularly, in monocular vision system scenarios. An important aspect to take into account is that other areas of research take advantage of the accuracy of human pose estimation to develop on top of it different solutions, for instance: human action recognition, gaming, surveillance, just to mention a few. Nowadays, most of computer vision tasks (e.g., segmentation, camera pose, object detection), including the human pose estimation problem, are tackled by convolutional neural networks (CNN), reaching a better performance with respect to classical approaches (e.g., [2, 6, 10, 22, 26, 29, 30]).

CNN architectures have been used in monocular vision system scenarios to solve the human pose estimation problem, using as an input a set of images with single or multiple-persons from one camera to feed the architecture. Regarding this latter point of multiple-persons input data, the computational cost could be increased due to the number of body joints of each subject that the architecture has to detect in the image. Although the obtained results are appealing, the complex poses are a challenging problem since certain parts of the body joints could be occluded, including when other moving objects are part of the scene. This problem could be overcome by the multi-view approaches since the human body can be captured from different points of view at the same time. This could allow to recover body joints occluded in one view by using information from other cameras, from another point of view, where these body joints are not occluded.

Some tasks such as camera pose, 3D-reconstruction, object detection (e.g., [5, 25, 27, 31], just to mention a few), where the principal problem is the occluded regions, have been tackled by multi-view approaches. However, few works have been proposed to tackle the human pose estimation problem by using the approach mentioned above. Some works propose to fuse features from both views (i.e., two cameras located in different points of view) to predict the human pose (e.g., [13, 23]), through either the epipolar lines of the images across all different views or using the intermediate layers in early stages of the architecture to find corresponding points in other views.

On the contrary to previous works, a compact architecture, which has been proposed for monocular scenarios [16], is adapted in the current chapter to leverage the relative camera pose in multi-view scenarios, and thus to find the relationship between the different features of the images, which are acquired at the same time from different views, to tackle the self-occlusion problems of the body joints.

This chapter is organized as follows. Section 5.2 summarizes deep learning based camera pose estimation methods used to obtain the relative rotation and translation

between two cameras, which includes experimental results on real-world datasets and transfer learning from virtual environments to real-world. Then, Sect. 5.3 describes the architecture proposed to estimate the human body pose in multi-view scenarios, including experimental results and comparisons with other approaches of the state-of-art. Finally, conclusions and future works are provided in Sect. 5.4.

5.2 Camera Pose Estimation

Most computer vision tasks require a calibration process to get camera intrinsic and extrinsic parameters, which are used to have correspondences between the camera and the world reference system. In general, these calibration processes are performed by using special calibration patterns. However, during the last decade, different proposals have been developed to estimate these parameters in an automatic way, more specifically the extrinsic camera parameters (relative translation and rotation), by using just the context of the images without considering special calibration patterns.

In the process of calibration there are difficulties such as illumination, low resolution, few images features, which turn this a challenging problem to be solved. For doing it, features detection is a key point to consider since they are used in any camera calibration process. Different proposals have been used for feature point detection, for instance SURF [1], ORB [24], SIFT [20] just to mention a few. However, the accuracy of these algorithms decreases when there are not enough features points to be matched. In order to overcome this problem, different CNN architectures have been used in this process due to the power to extract features on images, showing better results than classical approaches. Nowadays, methods used for camera calibration could be divided into two categories: single or multi-view environments.

In the single view approaches (e.g. [17, 18]), a sequence of images are used, which are captured from the same angle and point of view. The main limitation of these approaches is that some important features could be occluded due their dependence of the angle and position of the camera in the world coordinate system.

In the second category, i.e., multi-view approaches, the occluded feature problem could be solved since the scene is captured from different points of view at the same time. However, it is necessary to ensure the overlap between the acquired images. A few works have been proposed considering multi-view scenarios; it could be mentioned the approaches proposed by [6, 9, 19], where a set of pairs of images is used to feed a Siamese CNN architecture for relative camera pose estimation.

In this chapter, the multi-view environment problem is considered, where a Siamese CNN architecture is used to estimate the relative camera pose. This architecture and its main features are presented in Sect. 5.2.1. Then, experimental results from this Siamese architecture, on real world scenarios, are presented in Sect. 5.2.2. Finally, a transfer learning strategy, which takes advantage of virtual scenarios to initialize the network weights, is introduced in Sect. 5.2.3.

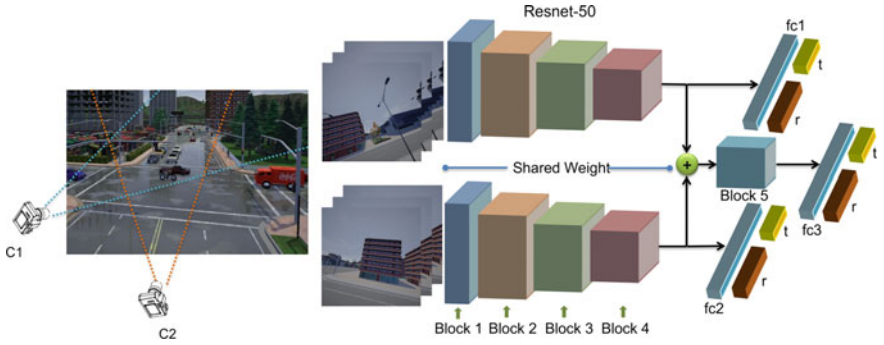

Fig. 5.1 Siamese CNN architecture fed with pairs of images of the same scene captured from different points of views at the same time. The extrinsic camera parameter estimation is obtained by using the regression part, which contains three fully-connected layers

5.2.1 Siamese Network Architecture

This section summarizes the Siamese CNN architecture, referred to as RelPoseTL, that has already been presented in our previous work [4]. It is used to estimate the relative pose between two synchronized cameras (Fig. 5.1 shows an illustration of this architecture). The RelPoseTL is based on a modified Resnet-50, proposed in [12], which has two identical branches with shared weights up to the fourth residual block. The output of each branch is concatenated to feed the fifth residual block. ELUs activation functions are used instead of RELUs in the residual block structures, since it helps to speed up convergence and to avoid the vanishing gradient, as it was mentioned in [7]. A pair of images are used to feed the architecture. These images are acquired at the same time, but from different point of views, i.e., different positions and orientations with respect to the real-world system. Three fully connected (*fc*) layers are added after to the fourth and fifth residual block. The first two fully connected layers *fc1* and *fc2* have a dimension of 1024 each one and are added after the fourth residual block to predict the **global camera pose** followed by two regressors of (3×1) and (4×1), which correspond to the global translation and rotation concerning the real-world system respectively. The last fully-connected layer *fc3* is added to the fifth residual block. Similarly to mentioned above, this fully connected layer has a dimension of 1024, but it is used to predict the **relative camera pose**, followed by two regressors of (3×1) and (4×1) to predict the relative translation and rotation between the given pair of images.

Global camera pose is represented by: $\Delta p = [\hat{t}, \hat{r}]$, where $\hat{t}$ represents the translation as a 3-dimension vector and $\hat{r}$ represents the quaternion values of the rotation as a 4-dimension unitary vector. The Euclidean distance is used to estimate them:

$$T_{Global}(I) = \left\| t - \hat{t} \right\|_{\gamma}, \tag{5.1}$$

$$R_{Global}(I) = \left\| r - \frac{\hat{r}}{\|\hat{r}\|} \right\|_{\gamma}, \tag{5.2}$$

where t and $\hat{t}$ represent the ground truth and prediction of the translation respectively, and r is the ground truth rotation represented as quaternion values, and $\hat{r}$ denotes its prediction. The predicted rotation is normalized to a unit length as $\frac{\hat{r}}{\|\hat{r}\|}$. $L2$ Euclidean norm is defined as γ. Due to the difference in scale between both terms (translation and rotation), the authors in [17] propose to use two learnable parameters called $\hat{s}_x$ and $\hat{s}_y$ to balance translation and rotation terms. The effect of these parameters are similar to the one proposed in [18], where a β parameter is used to balance both terms; for more details see the work proposed by [17, 18]. In RelPoseTL, a modified loss function that uses $\hat{s}_y$ as learnable parameter is used:

$$Loss_{Global}(I) = T_{Global} + (exp(\hat{s}_y) * R_{Global} + \hat{s}_y). \tag{5.3}$$

On the other hand, the relative pose is estimated from the output of the fifth residual block. The relative translation and rotation are obtained similarly to the global pose estimation, which is defined as:

$$T_{Relative}(I) = \left\| t_{rel} - \hat{t}_{rel} \right\|_{\gamma}, \tag{5.4}$$

$$R_{Relative}(I) = \left\| r_{rel} - \frac{\hat{r}_{rel}}{\|\hat{r}_{rel}\|} \right\|_{\gamma}, \tag{5.5}$$

where $T_{Relative}$ and $R_{Relative}$ are the differences between the ground truth (t_{rel} and r_{rel}) and the prediction from the trained model ($\hat{t}_{rel}$ and $\hat{r}_{rel}$). As the rotation ($\hat{r}_{rel}$) is predicted directly from the trained model, then it has to be normalized before. In order to get t_{rel} and r_{rel}, the following equations are used:

$$t_{rel} = t_{C1} - t_{C2}, \tag{5.6}$$

$$r_{rel} = r^*_{C2} * r_{C1}, \tag{5.7}$$

where Ci corresponds to the pose parameters of the camera i (i.e., rotation and translation); these parameters are referred to the real world system; r^*_{C2} is the conjugate quaternion of r_{C2}. Similarly to the Eq. (5.3), the loss function used to obtain the relative pose is defined as:

$$Loss_{Relative}(I) = T_{Rel} + (exp(\hat{s}_y) * R_{Rel} + \hat{s}_y). \tag{5.8}$$

Fig. 5.2 Real world images of Cambridge Landmarks dataset [18] used for training and evaluating the RelPoseTL architecture

Note that $Loss_{Global}$ in Eq. (5.3) and $Loss_{Relative}$ in Eq. (5.8) are applied for different purposes. The first one is used to predict global pose through each branch of the trained model, where each branch is fed by images captured at the same scenario from different points of view. The second one predicts the relative pose between pairs of images by concatenating the Siamese Network (see Fig. 5.1). The RelPoseTL was jointly trained with Global and Relative Loss, as shown in Eq. (5.9):

$$L = Loss_{Global} + Loss_{Relative}. \tag{5.9}$$

5.2.2 *Results from Real World Datasets*

The RelPoseTL architecture presented in the previous section has been trained and evaluated using the Cambridge Landmarks dataset [18]; Fig. 5.2 shows some images of this dataset, which were captured in outdoor environments. All images are resized to 224 pixels along the shorter side; then, the mean intensity value is computed and subtracted from the images. For the training process, the images are randomly cropped at 224×224 pixels. On the contrary to the training stage, during the evaluation process, a central crop is used instead of a random crop.

For the training process, the weights of Resnet-50 pretrained on ImageNet were used to initialize layers of RelPoseTL up to the fourth residual block, for the remaining layers samples from the normal distribution were used. The RelPoseTL was trained on ShopFacade and OldHospital of Cambridge Landmarks dataset. A set of 5900 pairs of images of OldHospital dataset was used for the training process. A similar process was performed but with 1300 pairs of images of ShopFacade dataset.

Table 5.1 Comparison of average errors (extrinsic parameters) of relative camera pose between RelPoseTL and Pose-MV architectures on ShopFacade and OldHospital of Cambridge Landmarks dataset

Scene/Models	Pose-MV [6]	RelPoseTL [4]
ShopFacade	1.126 m, 6.021°	**1,002 m, 3.655°**
OldHospital	5.849 m, 7.546°	**3.792 m, 2.721°**
Average	3.487 m, 6.783°	**2.397 m, 3.188°**

The architecture was trained during 500 epoch for both datasets, which approximately took 7 h and 3 h respectively. For the evaluation process, a set of 2100 pairs of images from the OldHospital dataset and a set of 250 pairs of images from ShopFacade dataset have been considered.

In order to evaluate the performance of RelPoseTL, angular error and Euclidean distance error are considered. The first is used to compute the rotation error; it is computed with a 4-dimensional vector. The second one is used to computed the distance error between ground truth and the estimated value using a 3-dimensional vector. Table 5.1 depicts average errors on rotation and translation for both datasets, obtained with the RelPoseTL network and with the Pose-MV network [6]. It can be appreciated that the translation error obtained by the RelPoseTL improves the results of Pose-MV in about 32%, and about 53% when the rotation error is compared between the RelPoseTL and Pose-MV.

5.2.3 From Virtual Environments to Real World

As an extension to the results obtained in the previous section, where the RelPoseTL architecture has been trained with real data, in this section a novel training strategy is proposed. It consists on first training the network with synthetic images obtained from a virtual environment and then, use this weights as an initialization when training again the network but with images from real scenarios. This strategy is intended to improve results from real data since the training of the network does not start from scratch but from a pre-trained set. An advantage of using virtual environments is the possibility of generating an almost unlimited set of synthetic images considering different conditions, actors and scenarios, i.e., weather, illumination, pedestrian, road, building, vehicles. Furthermore, an additional advantage lies on the fact that the ground truth is automatically obtained, reducing human error when the datasets are manually annotated. Different engines could be used to design virtual environments and aquire such a kind of synthetic pairs of images (e.g., CARLA Simulator [8], Virtual KITTI [11], Video Game engines, just to mention a few). In the current work different synthetic datasets have been generated using the CARLA simulator, an open-source software tool [8]. Among the tools offered by CARLA simulator, there is an editor that allows you to modify existing virtual worlds, as well as to create

new scenarios from scratch; this editor integrates both CARLA simulator and Unreal Engine, which is a video game engine that CARLA is based on.

It should be mentioned that synthetic images generated from virtual environments and the real images used for final training could have different features spaces and distribution, hence a Domain Adaptation (DA) strategy should be used to transfer the knowledge from one domain to other. Depending on the relationship between both domains, transferring the knowledge learned from virtual to real environments can be performed in one-step or multi-step DA. For the first case, both domains are directly related since the features spaces are similar. In the second case, an intermediate domain, which should be highly related with both domains, is necessary. In this section, the strategy proposed in [4] will be followed. It consists on creating 3D virtual scenarios with a similar structure to the datatsets of real images, in our case OldHospital and KingsCollege datasets. This similarity should be not only on the shape and distribution of objects in the scene, but also on the way images are acquired (i.e., similar relative distance and orientation between the cameras and objects in the scene for the generation of synthetic images).

In order to generate the two datasets with synthetic images similar to OldHospital and KingsCollege datasets, two 3D virtual scenarios are considered. These 3D virtual scenarios have similar structure to the corresponding real scenarios, i.e., they have the same feature spaces (objects' geometry and camera point of view); for more details about the relationship between geometric similarity of real and virtual scenario, as well as camera pose similarity between synthetic and real sequence, see the work proposed by [5]. Figure 5.3 shows illustrations of the virtual scenarios generated from the CARLA Simulator tool. The Dataset 1 and Dataset 2 are similar to the OldHospital and KingsColleges datasets respectively. Additionally, two virtual cameras were also considered. The cameras move in these virtual scenarios and acquire pairs of images. The cameras start the trajectories with a given initial position and orientation with respect to the world reference system. Then, the pair of cameras moves together randomly and change their relative orientation also randomly. The images are simultaneously acquired for each camera when there is enough overlap between their field of view. This process generates about 3000 synthetic images from both synchronized cameras, for each scenario; it takes about three hours for each scenario. The overlap between pairs of synthetic images is computed with OpenMVG [21], where a minimum of 60% overlap between acquired synthetic images is imposed.

Once datasets with synthetic images have been generated as mentioned above, the RelPoseTL model is trained by initializing all layers of the architecture as presented in Sect. 5.2.2. The Adam optimizer is used for the training process with batch size of 32 and learning rate of 10^{-4}. The synthetic images were resized to 224×224 pixels, including data normalization for training process. A set of 8124 pairs of images was considered to train the architecture, which took about 30 hour per dataset till convergence was reached. During the evaluation phase, images were pre-processed as mentioned above. A set of 2048 pairs of images from each of the two datasets was used.

Finally, in order to transfer the knowledge learned in the synthetic image domain to the new domain (i.e., real-world), the Domain Adaptation (DA) strategy is applied.

Fig. 5.3 Synthetic images generated using CARLA Simulator tool. (*2nd row*) Virtual scenario similar to OldHospital dataset. (*4th row*) Virtual scenario similar to KingsCollege dataset

It consists on training again the RelPoseTL architecture but now with just a few pairs of real-world images; the weights from the synthetic image domain are used as initialization. It helps to speed up the training process as well as to avoid the time consuming ground truth generation on real-world images. This strategy is also referred in the literature to as transfer learning. Figure 5.4 shows an illustration of this DA strategy. In details, each layer of RelPoseTL is initialized with the learned weights from the training process using synthetic images, which should be similar to the real-world scenarios. In order to show the advantages of this strategy, in this section, sets of (256, 512 and 1024) pairs of images from the real-world scenario are considered. These images are from the OldHospital and KingsCollege of Cambridge Landmark dataset [18]. They are used to train the model and results are compared with those obtained by initializing the network with the weights obtained from the synthetic image domain. In order to compare the results obtained when the network is initialized with ImageNet weights or with those from the synthetic scenario—the DA

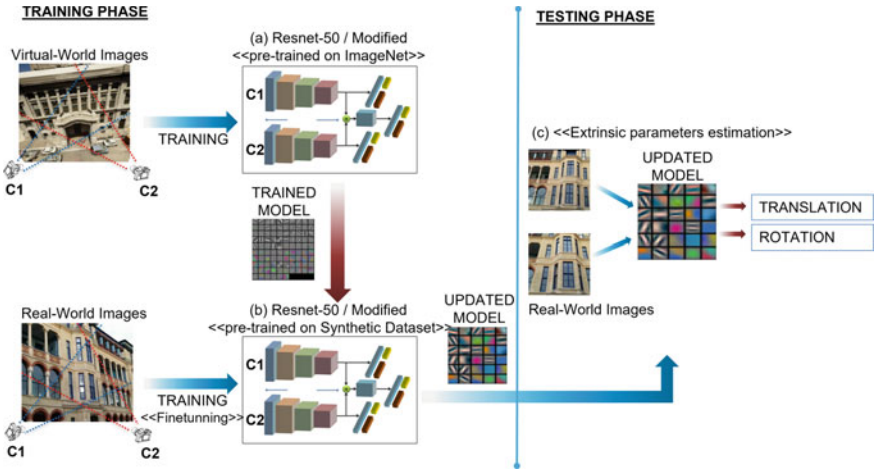

Fig. 5.4 **a** RelPoseTL is trained using synthetic images generated from CARLA Simulator. **b** The learned knowledge is used to apply DA strategy using real images. **c** Updated weights after DA strategy are used to estimate relative camera pose (i.e., relative translation and rotation)

strategy—three sets of 64, 128 and 256 pairs of real-world images are considered. Quantitative results are presented in Tables 5.2 and 5.3. Angular error is used to evaluate the rotation error from the estimated quaternion. On the other hand Euclidean error is used to measure the error between the estimated translation and ground truth value. In the case of OldHospital dataset, the DA strategy, which consists on using initially the Dataset 1 for training the model from scratch and then transferring the learned knowledge for retraining the model with just few real images, reaches the best result. This best result has been obtained in all cases, even if they are compared with the model trained using just the real images dataset. A similar process was performed with KingsCollege dataset using Dataset 2 where the results are the best even if the model is just trained using real images dataset, showing that the similarity of features spaces (i.e., 3D scenario used to represent the real environments) as well as in the camera pose (i.e., relative distance between the cameras and the objects in the scene as well as their relative orientation) helps to improve the camera pose estimation when DA is applied. Note that the accuracy of this estimation is important to relate the information of human body joints between the different cameras, which will be shown in the following section.

5.3 Human Pose Estimation

Once the relative pose between the different cameras is estimated as mentioned in the previous section, it is used as an input to tackle the 2D human pose estimation in multi-view environments. The 2D human pose is estimated by detecting human

Table 5.2 Angular and Euclidian distance errors of RelPoseTL [4] trained with real datat (RD) and with the domain adaptation (DA) strategy on pairs of images (PoI) from OldHospital dataset; on the first row RelPoseTL is initialized with ImageNet weights, while in the second row the weights are obtained by pre-training RelPoseTL with synthetic datatset (SD1)

Trained with	DA strategy on OldHospital dataset		
	Train: 256 PoI	Train: 512 PoI	Train: 1024 PoI
	Test: 64 PoI	Test: 128 PoI	Test: 256 PoI
RD (Init. ImageNet)	4.29 m, 5.72°	3.93m, 4.04°	3.48 m, 3.95°
RD (Init. SD1)	3.55 m, 5.59°	3.40m, 3.70°	3.20 m, 3.54°

Table 5.3 Angular and Euclidian distance errors of RelPoseTL [4] trained with real datat (RD) and with the domain adaptation (DA) strategy on pairs of images (PoI) from KingsCollege dataset; on the first row RelPoseTL is initialized with ImageNet weights, while in the second row the weights are obtained by pre-training RelPoseTL with synthetic datatset (SD2)

Trained with	DA strategy on KingsCollege dataset		
	Train: 256 PoI	Train: 512 PoI	Train: 1024 PoI
	Test: 64 PoI	Test: 128 PoI	Test: 256 PoI
RD (Init. ImageNet)	5.28 m, 5.29°	3.86 m, 5.08°	2.95 m, 4.06°
RD (Init. SD2)	4.89 m, 4.96°	3.13 m, 4.18°	2.35 m, 3.32°

body joints (e.g., elbow, wrist, head, etc.) from images and then connecting them to build the human body figure. During last years, different approaches have been proposed for the human pose estimation (HPE), for instance OpenPose [2], DeepPose [28], Convolutional pose machine [29], just to mentioned a few; appealing results have been shown from these approaches, mainly when all body joints are detected. However, the natural pose of human body generally involves self-occlusions that make the HPE a challenging problem in monocular vision system scenarios.

An alternative to overcome the problems of monocular vision systems could be by considering multi-view approaches. In these approaches, since the human body is captured from different points of view at the same time, occluded joints from one view can be observed from another view. The multi-view approaches have been already used to tackle the region occlusion problem in certain tasks such as 3D-reconstruction, camera pose, autonomous driving, object detection (e.g., [5, 14, 25, 27, 31]). However, few works have leveraged the advantages of multi-view approaches to overcome the occlusion problem in the human pose estimation. Some works exploit the epipolar geometry of multi-view approaches to solve the region occlusion problem as the authors in [23]. In the current work, a CNN architecture is used to fuse all features on epipolar line of the images of all points of view as previous step to get the predicted joints. Another approach has been proposed in [13], where the author leveraged the extracted feature of intermediate layers to find its corresponding points in a neighboring view to combine and robustly extract features of each view.

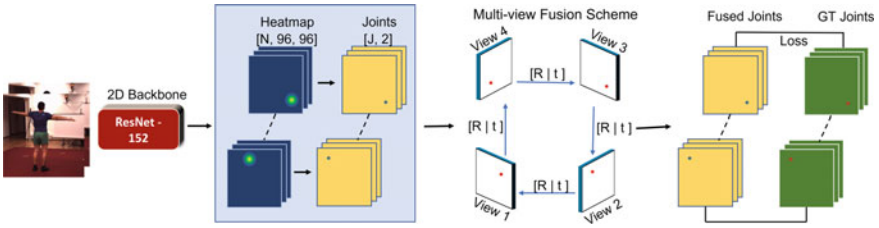

Fig. 5.5 CNN backbone is fed with a set of pairs of images of the same scene simultaneously acquired from different points of view. The multi-view fusion scheme allows to estimate occluded joints with information from other views across of the relative camera pose

The performance of these approaches allow that certain applications such as action recognition, healthcare, or augmented reality, take advantage of the obtained accuracy to develop different solutions. In this section a novel multi-view scheme is presented to robustly estimate the human body pose. The architecture to tackle the human pose estimation problem from a multi-view scheme is presented in Sect. 5.3.1. Then, experimental results are presented in Sect. 5.3.2.

5.3.1 Multi-view Scheme

The multi-view scheme presented in [3], which is referred to as Mview-Joints, tackles the self-occlusion problem in the 2D human pose estimation by considering at least two views. It uses the CNN backbone proposed by [16], which is a variant of Resnet-152 with learnable weights, and as output, the number of body joints is considered.

The model is fed by a set of images containing just a single-person, which has been captured from a multi-view system of C calibrated and synchronized cameras with known parameters. The images captured by the multi-view system are organized in pairs of images using two different close views, namely, reference view Im^{ref} and source view Im^{src}. Heatmaps obtained from each image, like those resulting from the usage of the backbone [16], are fused across source view considering the confidence of each joint and the relative camera pose, improving the accuracy of joints from each image (see Fig. 5.5).

In order to estimate the 2D position of the joints ($p_{(x,y)}$) in the image plane, the center of mass of each heatmap is computed as follow:

$$p_{(x,y)} = \sum_{u=1}^{W} \sum_{v=1}^{H} h_{i_{(u,v)}} . (\zeta_{\Theta}(h_{i_{(u,v)}})), \tag{5.10}$$

where ζ_{Θ} represents the function softmax; h represents the ROI of the heatmaps of ith joint and W and H correspond to the size of the ROI heatmap. For each 2D

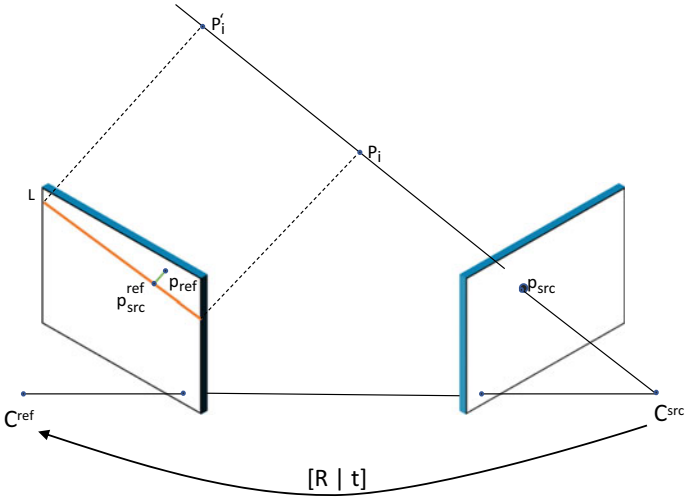

Fig. 5.6 An image point p_{src} back-projects to a ray in 3D defined by the point p_{src} and two depth values p_i and p_i'; it is then projected to the image plane of reference view to generate the epipolar line (L)

position of each joint obtained using Eq. (5.10), its position in the world coordinate system $P = (X, Y, Z)$ is obtained, as shown below:

$$x_i = f\frac{X}{Z} \qquad y_i = f\frac{Y}{Z}, \tag{5.11}$$

where (x, y) is the 2D position of the ith joint obtained in Eq. (5.10). The focal length of the camera is defined as f. Since the depth (Z) of the joint is unknown, two values are empirically defined to solve the Eq. (5.11). The first one corresponds to a depth value close to zero while the second value corresponds to a depth near to the size of the scene, in our case it has been set to 10 m.

The position in the world coordinate system of each joint is transformed using the relative camera pose between both points of view (see Fig. 5.6), i.e, source and reference view, and then, projected to the image plane, as shown below:

$$T_{rel} = Rot_{src} \cdot (T_{ref} - T_{src}), \tag{5.12}$$

$$Rot_{rel} = Q(Rot_{ref}.T)^{-1} * Q(Rot_{src}.T), \tag{5.13}$$

$$p^{ref}_{src_{(x,y)}} = \Delta 2D_{ref}(Rot_{rel} \cdot (P_i - T_{rel})), \tag{5.14}$$

where $Q(.)$ represents the quaternion. The rotation matrix and the translation vector are defined as $Rot \in \mathbb{R}^{3x3}$ and $T \in \mathbb{R}^{3x1}$ respectively. P_i corresponds at ith joint in

the world coordinate system obtained in Eq. (5.11). $\Delta 2D_{ref}(.)$ represents the back-projection of 3D position of ith joint from camera coordinate system to image plane in the reference view using the intrinsic parameters. The projected line on image plane of reference view L is obtained using the linear equation and the point computed in the Eq. (5.14). In order to obtain the depth of ith joint of the source view, which should be on the projected line on image plane of reference view, the intersection between the projected line and the 2D-point of the joint computed in the reference view $p_{ref_{(x,y)}}$ is performed.

The confidence of the two different 2D positions in the plane of the reference image of the ith joint, where the first corresponds to the reference view $p_{ref_{(x,y)}}$ and the second, a projected joint from source to reference view $p^{ref}_{src_{(x,y)}}$, is computed as the distance between the ground truth of 2D position of ith joint and the estimated 2D positions of ith joints. These confidence values are used as shown in Eq. (5.16).

$$\omega = 1 - \left| \frac{D_{\Delta}(\hat{\gamma}_i, \gamma_i)}{\sum D_{\Delta}(\hat{\gamma}_i, \gamma_i)} \right|, \tag{5.15}$$

$$\delta_{upd_{i_{(x,y)}}} = \omega * p_{i_{(x,y)}}, \tag{5.16}$$

where $(\hat{\gamma}\ ,\ \gamma)$ represent the ground truth and prediction of 2D position of ith joint respectively, and ω corresponds to the confidence of the points of ith joints in the reference view.

The enhanced 2D position of ith joint is denoted as $\delta_{upd_{i_{(x,y)}}}$, which considers the information and confidence of ith joint projected from the source to reference view. In order to minimize the error between the enhanced 2D position of ith joint and its ground truth in the learning process of the proposed multi-view scheme, a loss function is defined as:

$$Loss = \sum_{i=1}^{N} \left\| \delta_{upd_{i_{(x,y)}}} - \hat{p}_{i_{(x,y)}} \right\|_2, \tag{5.17}$$

where N corresponds to the number of joints, and $\hat{p}_{i_{(x,y)}}$ is the ground-truth of ith joint in image plane.

5.3.2 *Results from Multi-view Approach*

The Mview-Joints architecture presented in the previous section has been trained and evaluated using the Human3.6m dataset [15]. Human3.6m is one of the largest publicly available benchmark, with a multi-view setup, for human pose estimation. In details, four synchronized and calibrated cameras are considered to generate a

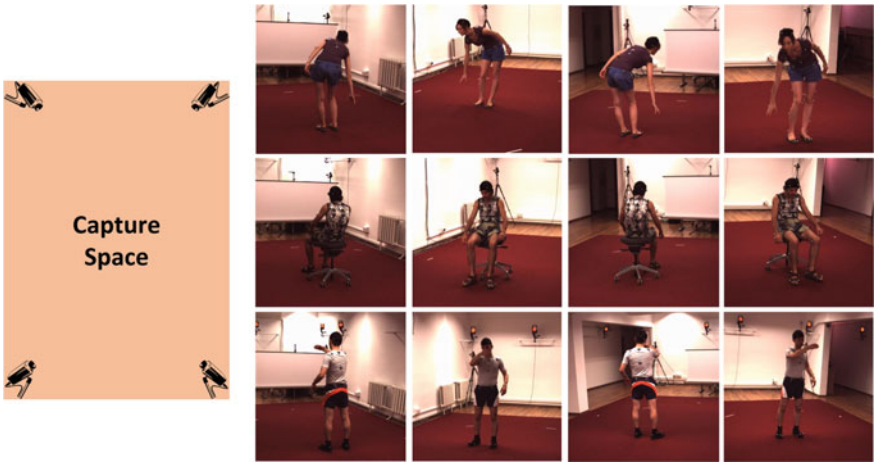

Fig. 5.7 Human3.6m dataset used for the training and evaluation processes. The subject is captured from different point of views considering the four calibrated and synchronized cameras located in each corner of the room

set of images with a single-person doing different activities (Fig. 5.7 shows three different captures from the four cameras).

Sets of images of Human3.6m dataset were cropped according to the bounding box of the person and resized to 384×384 pixels as a previous step for training the model. Mview-Joints was firstly initialized with the weight pretrained by [16] and trained on a set of 60 k pre-processed images of Human3.6m dataset, in an end-to-end way, until 20 epochs. This training process takes about 120 h. The same pre-processing is used during the evaluation phase with a set of 8k images. In order to evaluate the performance of Mview-Joints architecture for 2D human pose estimation, the Joint Detection Rate (JDR) is used. This metric measures the percentage of $successfully$ detected joints, which uses a threshold to validate if a joint has been $successfully$ detected. The threshold is defined as half of the head size. In addition, the Euclidean distance error is used to compute the accuracy of each human body joint estimated by the network.

Results and comparisons with state-of-the-art CNN-based approaches are presented in Tables 5.4 and 5.5 using JDR metric. As it can be appreciated, the improvement is most significant for shoulder, elbow and ankle joints, which increment from 96.44% to 99.65%, from 95.00% to 97.31% and from 96.62% to 97.45%, respectively. In term of average JDR, the results obtained by Mview-Joints improves the Epipolar transformer model [13] about 1%, and with respect to Cross-View fusion [23] about 3%.

Table 5.5 shows the Euclidean distance errors. In this case the accuracy of estimated body joints using the Mview-Joints architecture is compared with respect to the CNN backbone proposed by [16]. As it can be appreciated, body joints such as elbow, wrist, knee, nose, head improve by 15.88%, 4.32%, 8.46%, 18.11% and

Table 5.4 Comparison of 2D pose estimation accuracy on Human3.6m dataset using JDR(%) as metric. "−": these entries were absent. ∗: approach presented in [23]. ϒ trained again by [13]. ψ approach presented in [13]. R50 and R152 are ResNet-50 and ResNet-152 respectively. Scale is the input resolution of the network

	Net	scale	shlder	elb	wri	hip	knee	ankle	root	neck	head	Avg
Sum epipolar line *	R152	320	91.36	91.23	89.63	96.19	94.14	90.38	–	–	–	–
Max epipolar line *	R152	320	92.67	92.45	91.57	97.69	95.01	91.88	–	–	–	–
Cross-View fusion *ϒ	R50	320	95.6	95.0	**93.7**	96.6	95.5	92.8	96.7	96.5	96.2	95.9
Cross-View fusion *ϒ	R50	256	86.1	86.5	82.4	96.7	91.5	79.0	**100**	93.7	95.5	95.1
Epipolar trans-former ψ	R50	256	96.44	94.16	92.16	98.95	**97.26**	96.62	99.89	99.68	99.63	97.01
Mview-Joints	R152	384	**99.65**	**97.31**	**93.70**	**99.22**	97.24	**97.45**	99.83	**99.82**	**99.75**	**98.22**

Table 5.5 Comparison of average Euclidean distance error (pixels) between Mview-Joints and Learning triangulation backbone proposed by [16] on Human3.6m dataset (*Backbone*: Resnet 152 with pretrained weight [16])

	Net	shlder	elb	wri	hip	knee	ankle	root	neck	nose	belly	head	Avg
Learning triangulation	Backbone	**7.84**	8.00	7.40	**7.55**	7.45	9.70	5.75	**5.86**	6.46	6.47	6.57	7.18
Mview-Joints	Backbone + Multi-view	7.88	**6.73**	**7.08**	7.62	**6.82**	**9.19**	**5.24**	6.05	**5.29**	**6.15**	**3.25**	**6.48**

Fig. 5.8 Qualitative results on challenging scenarios—Mview-Joints architecture obtains better estimations than the backbone proposed by [16]

50.53% respectively, when compared to the results obtained with CNN backbone proposed by [16]. Since the multi-view scheme leverages the different views of the scenario, the challenging human body pose are better estimated. Figure 5.8 shows some scenes with self-occlusions where it can be appreciated that Mview-Joints architecture is able to estimate the human body pose better than single view approach [16].

5.4 Conclusions

This chapter focuses on the challenging problem of human body pose estimation in multi-view scenarios. It is intended to tackle self-occlusion problems by accurately estimating the human body pose. Firstly, in order to put the different views in a common framework, the relative position and orientation—extrinsic camera parameters—between the different cameras is estimated by using a deep learning based strategy, instead of classical calibration-pattern based approaches. In order to train this extrinsic camera calibration network, synthetic datasets of outdoor scenarios are generated overcoming the limitation of lack of annotated real-world data. Then, once relative pose between cameras is estimated, human body pose in the multi-view scenario is obtained by using an adaptation of an architecture initially intended for single view scenarios. Experimental results of estimated human pose and comparisons with state-of-the-art approaches are provided showing improvements on challenging scenarios. This chapter shows how information from different views can be fused in order to reach a more accurate representation than single view approaches, in particular when self-occlusions are considered. An important aspect to consider is that the precision of body joint estimations is the base to solve other related problems such as action recognition, surveillance, 3D human pose estimation among other. Future work will be focused on extending the usage of multi-view environments to leverage the geometry of the scene, and thus, improve the 3D human pose. Additionally, the usage of attention modules will also be considered to tackle the occluded regions into the multi-view scheme.

Acknowledgements This work has been partially supported by the ESPOL projects EPASI (CIDIS-01-2018), TICs4CI (FIEC-16-2018) and PhysicalDistancing (CIDIS-56-2020); and the "CERCA Programme/Generalitat de Catalunya". The authors acknowledge the support of CYTED Network: "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559) and the NVIDIA Corporation for the donation of the Titan Xp GPU. The first author has been supported by Ecuador government under a SENESCYT scholarship contract CZ05-000040-2018.

References

1. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
2. Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., Sheikh, Y.: Openpose: realtime multi-person 2d pose estimation using part affinity fields. IEEE Trans. Pattern Anal. Mach. Intell. **43**(1), 172–186 (2019)
3. Charco, J.L., Sappa, A.D., Vintimilla, B.X.: Human pose estimation through a novel multi-view scheme. In: Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP, pp. 855–862. INSTICC, SciTePress (2022)
4. Charco, J.L., Sappa, A.D., Vintimilla, B.X., Velesaca, H.O.: Transfer learning from synthetic data in the camera pose estimation problem. In: Proceedings of the 15th International Joint

Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, pp. 498–505. INSTICC, SciTePress (2020)
5. Charco, J.L., Sappa, A.D., Vintimilla, B.X., Velesaca, H.O.: Camera pose estimation in multi-view environments: from virtual scenarios to the real world. Image Vis. Comput. **110**, 104182 (2021)
6. Charco, J.L., Vintimilla, B.X., Sappa, A.D.: Deep learning based camera pose estimation in multi-view environment. In: 2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 224–228. IEEE (2018)
7. Clevert, D.-A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus) (2015). arXiv:1511.07289
8. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: an open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16 (2017)
9. En, S., Lechervy, A., Jurie, F.: Rpnet: an end-to-end network for relative camera pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)
10. Fang, H.-S., Xie, S., Tai, Y.-W., Lu, C.: Rmpe: regional multi-person pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2334–2343 (2017)
11. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 4340–4349 (2016)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
13. He, Y., Yan, R., Fragkiadaki, K., Yu, S.-I.: Epipolar transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition, pp. 7779–7788 (2020)
14. Hofbauer, M., Kuhn, C.B., Meng, J., Petrovic, G., Steinbach, E.: Multi-view region of interest prediction for autonomous driving using semi-supervised labeling. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6. IEEE (2020)
15. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6m: large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Trans. Pattern Anal. Mach. Intell. **36**(7), 1325–1339 (2014)
16. Iskakov, K., Burkov, E., Lempitsky, V., Malkov, Y.: Learnable triangulation of human pose. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7718–7727 (2019)
17. Kendall, A., Cipolla, R., et al.: Geometric loss functions for camera pose regression with deep learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 3, p. 8 (2017)
18. Kendall, A., Grimes, M., Cipolla, R.: Posenet: a convolutional network for real-time 6-dof camera relocalization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2938–2946 (2015)
19. Melekhov, I., Ylioinas, J., Kannala, J., Rahtu, E.: Relative camera pose estimation using convolutional neural networks. In: International Conference on Advanced Concepts for Intelligent Vision Systems, pp. 675–687. Springer (2017)
20. Mortensen, E.N., Deng, H., Shapiro, L.: A sift descriptor with global context. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, vol. 1, pp. 184–190. IEEE (2005)
21. Moulon, P., Monasse, P., Marlet, R., et al.: Openmvg. an open multiple view geometry library (2016). https://github.com/openMVG/openMVG
22. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision, pp. 483–499. Springer (2016)
23. Qiu, H., Wang, C., Wang, J., Wang, N., Zeng, W.: Cross view fusion for 3d human pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4342–4351 (2019)

24. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: an efficient alternative to sift or surf. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571. IEEE (2011)
25. Sarmadi, H., Muñoz-Salinas, R., Berbís, M.A.: RJIA Medina-Carnicer: Simultaneous multi-view camera pose estimation and object tracking with squared planar markers. IEEE Access **7**, 22927–22940 (2019)
26. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5693–5703 (2019)
27. Tang, C., Ling, Y., Yang, X., Jin, W., Zheng, C.: Multi-view object detection based on deep learning. Appl. Sci. **8**(9), 1423 (2018)
28. Toshev, A., Szegedy, C.: Deeppose: human pose estimation via deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1653–1660 (2014)
29. Wei, S.-E., Ramakrishna, V., Kanade, T., Sheikh, Y.: Convolutional pose machines. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4724–4732 (2016)
30. Minghu, W., Yue, H., Wang, J., Huang, Y., Liu, M., Jiang, Y., Ke, C., Zeng, C.: Object detection based on rgc mask r-cnn. IET Image Proc. **14**(8), 1502–1508 (2020)
31. Xie, H., Yao, H., Sun, X., Zhou, S., Zhang, S.: Pix2vox: context-aware 3d reconstruction from single and multi-view images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2690–2698 (2019)

Chapter 6
Video Analytics in Urban Environments: Challenges and Approaches

Henry O. Velesaca, Patricia L. Suárez, Dario Carpio, Rafael E. Rivadeneira, Ángel Sánchez, and Angel D. Sappa

Abstract This chapter reviews state-of-the-art approaches generally present in the pipeline of video analytics on urban scenarios. A typical pipeline is used to cluster approaches in the literature, including image preprocessing, object detection, object classification, and object tracking modules. Then, a review of recent approaches for each module is given. Additionally, applications and datasets generally used for training and evaluating the performance of these approaches are included. This chapter does not pretend to be an exhaustive review of state-of-the-art video analytics in urban environments but rather an illustration of some of the different recent contributions. The chapter concludes by presenting current trends in video analytics in the urban scenario field.

H. O. Velesaca (✉) · P. L. Suárez · D. Carpio · R. E. Rivadeneira · A. D. Sappa
Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo Km., 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador
e-mail: hvelesac@espol.edu.ec

P. L. Suárez
e-mail: plsuarez@espol.edu.ec

D. Carpio
e-mail: dncarpio@espol.edu.ec

R. E. Rivadeneira
e-mail: rrivaden@espol.edu.ec

A. D. Sappa
e-mail: asappa@espol.edu.ec

Á. Sánchez
E.T.S. Ingeniería Informática, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid, Spain
e-mail: angel.sanchez@urjc.es

A. D. Sappa
Computer Vision Center, Campus UAB, Bellaterra, 08193 Barcelona, Spain

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_6

6.1 Introduction

The reduction in the price of cameras has led to the extension of camera networks to different applications in urban environments. Nowadays, cameras have become a ubiquitous technology in our everyday life. Although they are mainly used for video surveillance applications, growing in popularity opens new possibilities for smart cities (e.g., from detection of available parking to road maintenance applications). In terms of economy, the global video surveillance market size is expected to grow from USD 45.5 billion in 2020 to USD 74.6 billion by 2025.[1] This growth is attributed to the increasing concerns about public safety and security, the growing adoption of IP cameras, and the rising demand for wireless sensors.

This huge amount of data is generally used for real-time monitoring by video-surveillance operators or for offline review if something needs to be investigated. All in all, after a user-defined time, which depends on the infrastructure, this information is eliminated from storing systems, missing the opportunity to transform video signals into a powerful source of information. Video analytics engines perform this transformation of signal into information. Video analytics automatically enhances video surveillance systems by performing the tasks of real-time event detection, post-event analysis, and extraction of statistical data while saving human resource costs and increasing the effectiveness of the surveillance system operation.

Developments related to technological advancements like deep learning have increased the feasibility of launching new and innovative products for urban video analytics. Among the extensive and varied computer vision research topics in the video/scene comprehension domain, essential needs continue to focus on object detection and tracking, license plate recognition, person re-identification, face and action recognition, among others. In many cases, the mentioned techniques must be executed in nature, which means a wide variety of scene conditions since achieving a suitable and robust solution is not trivial. Although robust object detection and tracking are crucial for any automated surveillance application, this can be seen as a preliminary step to unlock various use cases.

Video surveillance with machine learning can be a very cost-effective solution for different applications building on top of video analytics of urban environments. Although each one of these applications requires their specific implementations, there are common processing modules that are generally used, for instance, image preprocessing, including camera calibration and image enhancement, object detection and recognition, tracking, to mention a few. Hence, in this chapter, to review the state-of-the-art approaches, a pipeline with different modules is considered, which clusters the most common approaches in the literature to solve specific tasks (see Fig. 6.1). This module-based pipeline allows the analysis of state-of-the-art approaches and reviews them in a common framework. The current chapter is organized as follows. First, Sect. 6.2 reviews common image preprocessing techniques generally used in this framework. Section 6.3 summarizes object detection approaches, while Sect. 6.4 focuses on classification techniques from the state-of-the-art. In Sect. 6.5 track-

[1] https://www.statista.com/statistics/864838/video-surveillance-market-size-worldwide/.

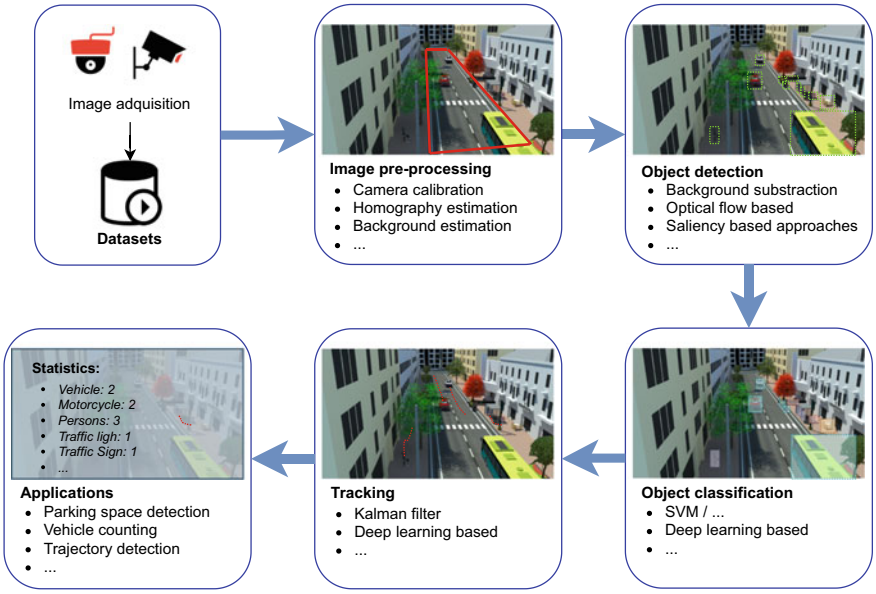

Fig. 6.1 Pipeline with the modules reviewed in the current work

ing approaches are reviewed. Video analytics applications of urban environments are presented in Sect. 6.6. Although in this chapter there is not a review on image acquisition hardware, common datasets generally used as benchmarks to evaluate different approaches are depicted in Sect. 6.7. Finally, discussions on challenges and opportunities from the reviewed approaches are given in Sect. 6.8

6.2 Image Preprocessing

An important stage to obtain more accurate solutions is the image preprocessing applied to the given input image before performing any computer vision approach. This section reviews only the most commonly used image processing techniques in video analysis of urban environments. It is not meant to be an exhaustive review but just a list of common approaches proposed to address recurring issues such as camera calibration, background subtraction, and image enhancement.

(a) (b)

Fig. 6.2 Example of inverse perspective mapping from **a** oblique view; into **b** top view

6.2.1 Camera Calibration

The most common image processing approach is the camera calibration, needed to accurately map 3D points to the image plane and define the region of the image to be analyzed or the relative distance between objects in the scene. In general, camera calibration is performed at the initial setup using a calibration pattern. The need for specific calibration patterns with known sizes to estimate the 2D-3D correspondences is a disadvantage for real-world applications. In order to overcome this limitation, there are some approaches that exploit the geometry from the images to perform the calibration, for instance, using vanishing point and lines (e.g., [16, 23, 68, 72]). Once extrinsic and intrinsic camera parameters are estimated, homography can be applied to reproject pixels from a given image region to the 3D real-world scenario. For instance, Noh et al. [55] propose an inverse perspective mapping to obtain a top view of crosswalk regions used for potential pedestrian risky event analysis. The system detects vehicles and pedestrians and estimates their trajectories; then, a decision tree is used to analyze whether there is or not a risk for the pedestrian at the crosswalk (Fig. 6.2 shows an illustration of the top view computed after camera calibration at a crosswalk). Another camera calibration-based approach, but in this case for inverse perspective mapping from on-board camera systems, is considered by Grassi et al. [33], where top view images together with GPS location (smartphone information) are used to identify parking spaces in urban scenarios. A more elaborated approach has been recently presented by Liu et al. [49], where also an inverse perspective mapping approach is proposed for vehicle counting, speed estimation, and classification. The proposed approach combines a convolutional neural network classifier with projective geometry information to classify vehicles.

On the contrary to previous approaches where bird's-eye views, obtained from inverse perspective mapping, Grents et al. [35] consider that just the oblique view is used to obtain the polygonal region of interest from the given image. A local reference frame is placed over this polygonal ROI to estimate the pose of the objects in the scene. The approach allows to count, classify, and determine the speed of vehicles in the annotated ROI. Ling et al. [48] use perspective images for identifying free

parking spaces in a low-cost IoT device. The system can provide real-time traffic and parking management information for smart cities.

6.2.2 Background Subtraction

Background estimation is another image processing approach generally used with static cameras. Background models are used to subtract from a given frame and easily detect the foreground objects (moving objects). The main limitation with these approaches is the threshold definition; it can be a constant value or dynamically updated [37]. Different background models have been proposed in the literature, one of the simplest and easy to compute is the temporal averaging of consecutive frames [44] or the temporal median [34]. These methods were widely used in the 1990s for traffic surveillance due to their simplicity. However, they are not robust to challenging scenarios, including fast changes in lighting conditions, camera jitter, or dynamic backgrounds. More elaborated background models have been proposed by using statistical models such as Mixture of Gaussians (MOG) [62]. In these methods, each pixel is modeled as a mixture of two or more Gaussians and online updates. These methods allow modeling scenarios with dynamic backgrounds or with lighting changes. Their main limitation is the computational complexity, resulting in a higher processing time than simpler background subtraction approaches. Recently, deep learning-based approaches have been proposed for the background subtraction problem. For instance, Lim et al. [47] present an encoder-decoder structured CNN to segment moving objects from the background. The network models image background while extracting foreground information as a high-level feature vector used to compute a segmentation map. Deep learning-based approaches have shown a considerable performance improvement compared with conventional unsupervised approaches; actually, top background subtraction methods evaluated in public benchmarks (e.g., CDnet 2012, CDnet 2014) are deep learning-based approaches. An extensive study on background subtraction methods based on deep neural networks is presented by Bouwmans et al. [11], where more than four hundred papers are reviewed.

6.2.3 Image and Content Enhancement

Finally, other image processing approaches generally applied in the video analysis of urban environments are related to the image enhancement processes; these image processing approaches are intended to improve the quality of the image in order to reach a better final result. Qu et al. [57] propose an image enhancement processing stage that is performed to reduce the influence of light changes. The proposed approach enhances the image's contrast and highlights the object's color. The approach is tested on pedestrian detection in different scenarios, improving the accu-

racy of the results. Another image enhancement approach, also focusing on lighting problems, more intended explicitly for nighttime low illumination scenarios, has been proposed by Shi et al. [61]. The authors propose a dual-channel prior-based method—dark and bright channels—with a single image. The proposed approach corrects the lack of illumination as well as gets well-exposed images. Mapping the given image to another color space is a common image processing approach, generally used to enhance the representation of the image to facilitate further processing. For instance, Zotin et al. [74] present an image enhancement algorithm based on multi-scale Retinex in HSV color model. The authors propose a novel approach that requires a less computational cost to convert from the RGB to HSV and back again to RGB; this mapping to the HSV space is required to correct the V component. The proposed approach has been tested in different outdoor scenarios showing appealing results. Finally, Cucchiara et al. [21] propose an image processing approach to detect and suppress shadows in a surveillance system. The technique is performed in the HSV color space, which improves the precision of the location of the shadows, favors the extraction of the most relevant information, and facilitates further processing.

Although not related to image enhancement but content manipulation, we can mention some approaches in the literature proposed for privacy protection. The ubiquitous deployment of vision systems in public areas has increased privacy and security concerns. Hence, data protection strategies need to be appropriately designed and correctly implemented in order to mitigate this problem. An example of a strategy implemented to preserve identity at a scale is the face and license plate blurring process in Google Street View [28]. New intelligent surveillance systems are being designed to face up to this concern; for instance, Tu et al. [63] propose a privacy-preserving video streaming strategy. It consists of face detection and removal strategy; hence, videos can be later on distributed for further processing.

6.3 Detection

Once the information has been preprocessed, the next step, according to the pipeline presented in Fig. 6.1, is the detection of objects before their classification. Different approaches have been proposed in the literature. Fan et al. [27] propose a technique to detect abandoned objects in public places, for which a temporary static objects model is taken as a basis to simulate in an urban environment the complete cycle of people or cars stopped for momentary causes such as traffic lights or signals using a finite state machine. This approach allows detecting the object over a given region directly without applying a classification approach. This model mitigates the problem of detecting false positives in urban environments. Avola et al. [6] present an approach to detect anomalous security behaviors in public or private environments. The detection technique uses histogram equalization and RGB local binary pattern operator based on images' changes. High-resolution images taken at low altitudes by unmanned aerial vehicles have been used to demonstrate the approach. Another approach to detecting objects is presented by Chen et al. [18], where it is proposed

to use gigapixel video; this type of high-resolution image provides local and global information simultaneously to obtain in real-time the location of the object globally by regions and then using descriptors the location of the object is complemented locally.

Junos et al. [40] propose an approach based on convolutional networks, where a lightweight model has been designed that focuses on increasing the receptive field of the network through the use of a pyramidal combination of the layers of the model. The experiments were favorable to be implemented in real-time applications. Chandrakar et al. [17] propose a technique for detecting moving objects and subsequent monitoring. The deep learning model proposed by the authors called RBF-FDLNN has been used. This model allows it to extract the necessary characteristics to detect moving objects, allowing their subsequent monitoring efficiently. The datasets used in the experiments are PASCAL and KITTI, which contain objects in urban settings. Another approach to object detection is presented by Shi et al. [60]; the authors propose a convolutional network model using the geometric information of stereo images. This approach does not require anchor boxes or depth estimation, it adaptively performs the detection of objects, achieving speed and precision in the obtained results. Bochkovskiy et al. [9] propose an object detection model capable of supporting real-world environments efficiently, for which they have designed an architecture that focuses on enhancing the receptive fields of each layer, also using drop blocks as a regularization method. They have also applied data augmentation through the mosaic of 4 images. The blocks have been designed for punctual attention. The COCO and Imagenet datasets have been used for the experiments, obtaining better statistics than their predecessor, YOLOv3.

Wei et al. [67] propose an approach to detect and classify people. For the detection step, several operations have been applied to reduce the computational load, for which each image is reduced four times and the colors are converted to only luminance. It applies the difference between consecutive images to detect moving areas and identify people, which are then classified with a convolutional network model. Another approach is presented by Cao et al. [14], where the authors present a method to analyze videos of traffic scenarios. It is based on a three-dimensional reconstruction model to extract geometric features, which is later on used for vehicle detection. Aslani et al. [4] present another approach that performs the real-time detection of moving objects from aerial images. It is based on the usage of optical flow and a filtering step to remove non-relevant information; a morphological filter is applied to extract the most significant features of the regions of interest and discard occlusions.

Gao et al. [30] present a method to detect and count people; this approach is a combination of two techniques since it uses a convolutional network for the extraction of characteristics, in this case, the MASK R-CNN and the linear support vector machine to perform the detection of people. As input for the deep learning model, the regions of people's heads obtained from the images by means of the Adaboost algorithm are used. The authors generate their data set taken from real classroom surveillance scenes for the experiments. Another method that uses convolutional networks is presented by Zhang et al. [71], where a method to detect objects (e.g., buses, cars, and motorcycles) in traffic scenarios is presented. The focus is on two

optimized components that allow real-time object detection. The first component generates the candidate regions, and the second component detects the objects in the previously selected regions. To improve the performance of the method, global and multi-scale information is incorporated from the processed videos. Three datasets have been used to validate the proposed method: MS COCO, PASCAL VOC07, KITTI.

Object detection is one of the most widespread techniques in computer vision, and there is a huge number of applications combining detection with segmentation and/or classification. Like in most of the topics of computer vision, classical approaches for object detection have been overcome by CNN-based approaches. These models facilitate the automatic extraction of features and achieve the best object detection results. In fact, there are already some CNN-based real-time embedded object detection solutions on the market, which are useful for building different applications with them.

6.4 Classification

Once the regions of interest have been detected, they need to be analyzed and classified according to the categories defined in the given problem. For decades, object classification has been an active research topic; several approaches have been proposed based on different machine learning techniques. This section covers both classical and deep-learning-based techniques. Silva et al. [26] propose an approach to detect and classify motorcycle helmets using a multi-layer perceptron network. Also focusing on the classification problem, Buch et al. [15] propose a novel approach to visualize the trajectory of vehicles in an augmented reality framework by reconstructing objects classified as vehicles in 3D. Another technique for urban traffic surveillance is presented by Cao et al. [15], where the authors propose a robust algorithm to detect and classify vehicles in environments of variation of lighting and complicated scenes, to maintain the most representative global characteristics to accelerate the detection and classification of the vehicles. Low altitude aerial images have been processed to carry out the experiments applying the gradient histogram method. Makhmutova et al. [52] also tackling the traffic management problem, an optical flow-based approach is proposed by using the cameras of the public traffic control system. The main idea is to classify the types of cars to determine the location and count them later.

Continuing with the classification of objects in videos, a method is proposed by Bose et al. [10], which allows the use of low-resolution images with projective distortion to classify objects present in the far-field. This method makes it possible to classify pedestrians and vehicles invariant to changes in the scenes through contextual functions based on the position and direction of movements of the objects present in the videos. Another classification technique is presented by Hamida et al. [38], where the authors propose an architecture that can scale according to the client's requirement. This scheme does not affect the temporal, spatial, or qualitative aspects

and focuses on extracting the information from the region of interest, which is filtered to maintain only the most representative characteristics. Canel et al. [13] present a novel method to perform intelligent filtering. It is referred to as FilterForward, which using micro-classifiers takes only relevant frames from the urban traffic videos, with which coinciding events (i.e., objects of specific colors carried by pedestrians) are detected. Also, an approach to classify vehicles (e.g., buses, cars, motorcycles, vans) is presented by Buch et al. [12]; this method is based on 3D models of the silhouette of each of the vehicle models present in the video. To validate the efficacy of the designed model, the UK reference i-LIDS dataset was used.

On the contrary to previous approaches, Gautamin et al. [31] propose a face detection method that is invariant to changes in illumination, orientation and supports partial occlusions. The approach is based on training with authorized faces calculating their weight; then, in the validation process, the weights are compared to determine whether or not the faces match. The process of classifying the faces is carried out by applying the Haar cascade classifier combined with the skin detector. Additionally, Xu et al. [70] present an approach to classify in real-time objects present in a surveillance video. Linear support vector machine and the histogram of oriented gradients have been used for the feature extraction. Li et al. [46] propose a threshold filtering method adaptable to variations in correlation between the types of features presented over a period of time; this approach is based on clustering, which allows to carry out the real-time analysis of classified vehicles.

In addition to the classical approaches mentioned above, many convolutional neural network-based approaches have been proposed in recent years. Vishnu et al. [65] present a technique to detect motorcycle offenders for not wearing a helmet. The model is based on a convolutional network model to perform the classification and recognition of drivers without a helmet. Kumar et al. [43] present a simplified approach to process a large amount of traffic data to make the prediction and forecast of its probable behavior. The authors use a convolutional neural network model to predict and forecast urban traffic behavior. Li et al. [45] propose a network, which extends the framework of the Faster R-CNN architecture, to detect and classify any object presented in each frame of the urban video sequence. Another approach that presents a method for estimating the volume of urban traffic is described by Zhu et al. [73]. This technique is based on a deep learning model that works with high-resolution images taken from an unmanned aerial vehicle, which were manually tagged. This model recognizes and classifies vehicle types in trucks, cars, or buses. Similarly, Wang et al. [66] present an approach based on a convolutional network called NormalNet, which has been designed to extract the characteristics of images using 3D information combined with normal vectors to achieve an efficient classification. Another CNN approach is presented by Naik et al. [53], where the authors implement a customization work of the YOLOv4 model to perform multiple object detection using a low-resolution real-time video, with which nine different types of vehicles are classified. Continuing with deep learning, Arinaldi et al. [3] present a technique to collect information from traffic videos in real-time to estimate the type of vehicle, its speed, and the respective count. A convolutional network-based on regions of interest known as Faster RCNN has been designed for this approach.

More recently, Dey et al. [24] propose a technique based on transfer learning. It obtains the information from previously trained models to be able to classify the vehicles present in the urban traffic videos in real-time. Foreground region detection is applied to group vehicle patterns to achieve vehicle classification. Object detection is one of the most widespread techniques in the field of computer vision and has had wide applications combined with segmentation and/or classification. Traditional detection techniques are mostly based on machine language algorithms and convolutional networks have already been used to describe their location in images. These models facilitate the extraction of features automatically.

The object classification problem has been largely studied in the computer vision domain. Most of the techniques reviewed in this section tackle the object classification, combined with the object detection problem, by means of efficient convolutional neural networks; this combination allows to implement solutions that can be deployed in real-time on embedded platforms. As an additional conclusion, it could be mentioned that a large number of references made use of transfer learning to train the proposed architectures. This strategy improves classification results, particularly in challenging scenarios that may include occlusions, crowdedness, or poor lighting conditions.

6.5 Tracking

The last stage to obtain an accurate solution is the object tracking task using the information obtained in the previous stage of the pipeline. Since the objective of urban video analytics is related to counting, identifying, or re-identifying objects in urban environments (e.g., cars, trucks, buses, motorcycles, bicycles, pedestrians, and other moving objects), tracking the object is needed once it appears in the image and is correctly detected to avoid duplicity. Generally, it is not only necessary to track a single object in the sequence of frames, so Multiple Object Tracking (MOT) is a task that currently plays an essential role in computer vision. MOT task is mainly divided into locating multiple objects, maintaining their identities, and performing their trajectories given a video input.

Different types of probabilistic inference models are used in different works to carry out the MOT task. The most widely used technique is the Kalman Filter [41], which is a recursive predictive filter that is based on the use of state-space techniques and recursive algorithms. To carry out the tracking, Chen et al. [19] use the centroid of each detected blob using a constant velocity Kalman Filter model. The state of the filter is the centroid location and velocity, and the measurement is an estimate of this entire state. A variation of the original Kalman Filter technique called Extended Kalman Filter (EFK) is used in other works. In the work of Cho et al. [20] the measurements obtained from the different sensors (e.g., cameras, radars, and lidars) are merged to track neighboring objects; and the EFK implementation uses the sequential sensor method that treats the observations of individual sensors independently and feeds them sequentially to the EKF estimation process. Also, based on EFK,

Dyckmanns et al. [25] present an approach to object tracking in urban intersections, demonstrating that maneuvers can be recognized earlier, and incorrect maneuver detection can be avoided by incorporating prior knowledge into the filtering process. The active multiple model filter approach helps to compensate for unavoidable measurement errors and thus enables robust and stable estimations of the position and velocity of tracked objects. On the other hand, the work presented by Nguyen et al. [54] uses an Interacting Multiple Model (IMM) [8] because objects such as cars, trucks, bikers, etc., change their behavior so frequently that one motion model alone is not sufficient to track their movements reliably. Structurally, the IMM tracking framework consists of several EKF's with different motion models, and the track estimates are mixed statistically.

Another strategy used for object tracking is the Particle Filter, a method used to estimate the state of a system that changes over time. More specifically, it is a Monte Carlo method commonly used in computer vision for tracking objects in image sequences. Rehman et al. [5] present an anomaly detection framework based on particle filtering for online video surveillance, where they use characteristics of size, location, and movement to develop prediction models and estimate the future probability of activities in video sequences. On the other hand, Gaddigoudar et al. [29] use the particle filter algorithm to track fast-moving human targets; mainly, it performs better with pedestrians with occlusions.

Other works, such as those presented by Jodoin et al. [39] use classical techniques based on binary descriptors and background subtraction, the combination of both methods allows us to track road users independently of their size and type. Another recent technique in the object tracking domain is Simple Online and Real-Time Tracking (SORT) [7]. This algorithm is based on the Kalman Filter [41] and Hungarian Algorithm [42]. SORT compares the positions of the objects detected in the current frame with assumptions for the positions of the objects detected in previous frames. Grents et al. [35] present a study that addresses the problem of estimating traffic flows from data from video surveillance cameras; in this work, the Fast R-CNN is used to detect objects, and later on, the SORT is considered to estimate the total number of vehicles in an avenue in real-time. On the other hand, Velesaca et al. [64] consider the use of SORT to generate general statistics of the different objects in urban environments. Among the main categories include persons and vehicles.

Contrary to the traditional techniques explained above, in the last decade, techniques based on deep learning have been used to increase the performance of object tracking in urban environments significantly. Del Pino et al. [22] propose the use of convolutional neural networks to estimate the actual position and velocities of the detected vehicles. On the other hand, Liu et al. [50] propose a deep learning-based progressive vehicle re-identification approach, which uses a convolutional neural network to extract appearance attributes as the first filter, and a siamese neural network-based license plate verification as a second filter. The SORT algorithm presented above was later enhanced to add a deep association metric, and this new approach was named DeepSORT [69]. This model adds visual information extracted by a custom residual CNN. CNN provides a normalized vector with 128 features as output, and the cosine distance between those vectors was added to the affinity

scores used in SORT. In the work presented by Praveenkumar et al. [56], DeepSORT is used to track multiple pedestrians in real-time, using motion representation and data association algorithms. The virtual block counting method is implemented for efficient tracking and counting of multiple pedestrian objects, in which the occlusion problem is effectively solved. In other recent works, DeepSORT was used to track vehicles, thus determining the number and flow of cars that travel on a particular avenue [59].

All approaches presented above focus their efforts on tracking multiple objects in 2D, that is, in the image plane. Contrary to this approach, 3D tracking could provide more precise information about the position, size estimation, and effective occlusion management; this approach is potentially more helpful for high-level machine vision tasks. An example of this approach is the work presented by Nguyen et al. [54], which uses a stereo camera system to get a cloud of 3D points and map an occupation grid using an inverse sensor model to track different objects within the range of the sensors. However, despite all these benefits, 3D tracking is still developing to solve camera calibration, pose estimation, and scene layout problems.

According to the reviewed literature, among the different types of probabilistic inference models, it was found that the most used were Kalman Filter, Extended Kalman Filter, and Particle Filter. On the other hand, other widely used approaches in recent years are those based on deep learning, where convolutional neural networks are frequently used to perform multiple object tracking tasks. A widely used deep learning-based approach is the DeepSORT network, which is an improved version of the SORT algorithm. In addition to the approaches based on tracking in 2D, in recent years, there has been a growing interest in the development of 3D object tracking methods, mainly using point clouds. It was possible to show that recent works are working on deep learning and 3D tracking approaches.

6.6 Applications

The application of video analytics in urban environments has received increasing attention in the past few years. This is because video analytics can be used to improve the understanding of the dynamics of urban scenes. According to the modules reviewed above, different video content analysis applications can be applied to process urban environment videos in real-time. A wide range of applications can benefit from the results of detection, classification, and tracking techniques; just some examples of these applications are Parking Space Detection, Vehicle or People Counting, Trajectory Detection, Face recognition, Vehicle License Plate Recognition, among others. This section provides a brief yet concise survey of state-of-the-art applications.

Fig. 6.3 Illustration of parking space detection in a parking lot (10 free space highlighted in red box)

6.6.1 Traffic Scenarios

License plate recognition is largely used worldwide for vehicle identification, access control, and security, generally in controlled scenarios and stationary vehicles. It is commonly referred to as *automatic number-plate recognition* (ANPR) or *automatic license-plate recognition* (ALPR). This process, as shown in Fig. 6.4, consists in detecting the license plate in a video frame and then recognizing the number plate of vehicles in real-time videos. Finally, the recognized information is used to check if the vehicle is registered or licensed or even grants access control to private places. The ANPR/ALPR systems have been used in a wide range of applications, places, and institutions, such as shopping malls, police forces, tolls, private places in general, car parking management, and other data collection reasons (e.g., [2, 50, 75]). The main purpose of these systems is to identify vehicles and their registered owners by reading the license plates. The technology is used by police forces to identify stolen cars, by customs to detect illegal immigrants, and by parking companies to issue parking tickets. Another urban environment application based on video analytics is the identification of free parking space (as shown in Fig. 6.3), in general, using low-cost IoT devices. These techniques aim to detect free parking slots in real-time, count the number of places for parking management systems, and provide this information to the drivers. This task is difficult because of the large number of parking spaces and the complexity of the parking environments. License plate detection is also required for privacy protection [28], detecting and blurring the plate number for confidentiality reasons. Furthermore, in [49], vehicle counting, speed estimation, and classification is proposed, which combines object detection and multiple object tracking to count and classify vehicles, pedestrians, and cyclist from video captured by a single camera. Finally, another application for traffic control is the vehicle classification (e.g., cars, buses, vans, motorcycles, bikes) detected in a scene [12]; where counting vehicles by lane and estimating vehicle length and speed in real-world units is possible.

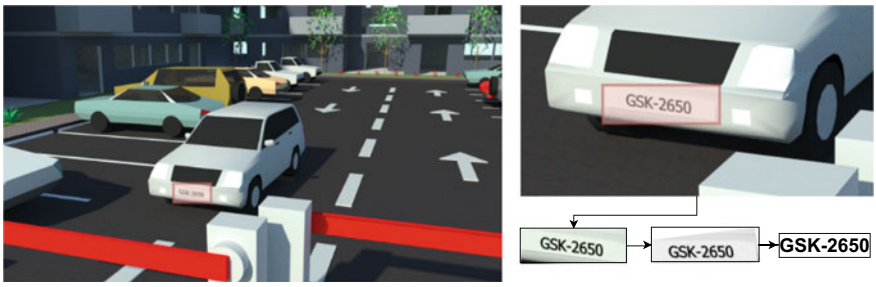

Fig. 6.4 Illustration of vehicle license plate recognition process for access control

6.6.2 Pedestrian-Oriented Applications

Pedestrian Detection and Tracking are another classical application in urban video analytics environments that has attracted much attention in recent years due to the increase in computational power of intelligent systems. On top of pedestrian detection and tracking, other approaches can be developed, for instance counting people (Fig. 6.5) in specific places or human behavior analysis (e.g., how long they stay in a particular area, where they go, etc.) in intelligent video surveillance systems (e.g., [29, 56]). Still, a limitation of these approaches is to deal with challenging crowded places (occlusions or low-resolution images) or false positives detections; despite these limitations, some applications can get good performance [51]. More recently, due to the COVID-19 outbreak, some approaches for pedestrian detection and tracking for social distance have been proposed; for instance, [1] proposes to use an overhead perspective. This method can estimate social distance violations between people using an approximation of physical distance, making use of the fact that, in general, people walk in a more or less systematic way, and they tend to move in the same direction as the rest of the people in the scene. Another challenging application is the estimation of pedestrian actions on streets [58]. It can be applied for safety issues on smart cars. The estimation of pedestrian actions can provide information such as the intention of the pedestrian (e.g., to cross the street, to stop at a traffic light, etc.), which can be used by smart cars to make decisions about how to behave.

Face detection (e.g., [31, 63]) is one of the most common applications of video analytics in urban environments. It can be used to identify people who are walking or driving in the area. This information can be used to improve security by tracking specific individuals' movements or improving traffic flow by identifying choke points. Furthermore, face detection can be used to count the number of people in an area, which can be used to estimate the number of people who are likely to visit an area at a given time.

As a conclusion from this section, including traffic scenarios and pedestrian-oriented applications, it could be mentioned that more and more video analytics applications will be offered in the context of smart cities in the next few years. This growth will happen due to the expansion of camera networks, which is a ubiqui-

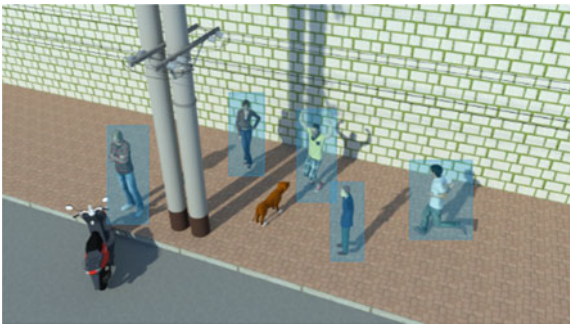

Fig. 6.5 Illustration of detecting and counting people in the street (5 persons detected)

tous technology in any urban scenario, the increase in computational capabilities, and the deep-learning strategy generally used to implement the different solutions. These three elements are inspiring the development of novel applications; some of them were unthinkable just a few years ago. These applications will address open problems in urban scenarios, just some of them are: action recognition, early event detection, path prediction (in both pedestrian and vehicle), crowd behavior analysis, multimodal architectures (using video analytics together with the social media analysis), crowdsourcing solutions using cameras from a mobile phone, and many other applications that could include cameras from remote sensing devices (e.g., drone, satellite, and so on). This list does not pretend to be an exhaustive list of further novel applications but just an illustration of open problems that may be tackled and is a good starting point to think about the possibilities that video analytics can offer to smart cities in the near future.

6.7 Datasets

This section details some of the different datasets used in the state-of-the-art video analytics in urban environments to detect, classify, and track different types of objects (e.g., pedestrians, vehicles, license plates, traffic signs). It does not pretend to be an exhaustive list of datasets but rather an illustration of the different benchmarks available for evaluation and comparisons. One of the most used and popular datasets is the **KITTI**[2] dataset (e.g., [17, 22, 60, 71]) that allows tracking of both people and vehicles. The dataset has been collected by driving a car around a city. It consists of 21 training videos and 29 test ones, with a total of about 19000 frames. Item includes detections obtained using the DPM and RegionLets detectors, as well as stereo and laser information. Another widely used dataset is **MS COCO**[3] (Microsoft Common Objects in Context) dataset (e.g., [9, 71]), which is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K

[2] http://www.cvlibs.net/datasets/kitti/eval_tracking.php.

[3] https://cocodataset.org/#home.

images. **PASCAL VOC 2007**[4] dataset (e.g., [17, 71]) is generally used for training image recognition approaches. This dataset includes twenty object classes to find persons, animals, vehicles, and others. The dataset can also be used for image classification and object detection tasks. Another popular dataset is **ImageNet**,[5] which is an image database (e.g., [9, 24]) organized according to the WordNet hierarchy (currently only the nouns), in which hundreds and thousands of images depict each node of the hierarchy. The project has been instrumental in advancing computer vision and deep learning research. Another dataset used in the literature is **UA-DETRAC**[6] (e.g., [45]), which consists of 100 video sequences captured from real-world traffic scenes (over 140000 frames with rich annotations, including occlusion, weather, vehicle category, truncation, and vehicle bounding boxes) for object detection, object tracking, and MOT system.

More specifically for urban environments, **VeRi-776**[7] is a vehicle re-identification dataset (e.g., [50]) that contains 49357 images of 776 vehicles from 20 cameras. The dataset is collected in the real traffic scenario, which is close to the setting of CityFlow. The dataset contains bounding boxes, types, colors, and brands of each vehicle. Another interesting dataset, which involve vehicle images captured across day and night, is the **Vehicle-1M**[8] dataset (e.g., [36]). It uses multiple surveillance cameras installed in cities. There are 936051 images from 55527 vehicles and 400 vehicle models in the dataset. Each image is attached with a vehicle ID label denoting its identity in the real world and a vehicle model label indicating the vehicle's car maker, model, and year. On the contrary to previous datasets **AVSS-2007**[9] dataset (e.g., [27]) is focused on subway environments. It includes three video clips corresponding to low, intermediate, and high activity in a subway scenario. Each clip is about 2 to 3 minutes long, with one staged true drop. Another of the datasets widely used for evaluations and comparisons is the **CDnet**[10] dataset, which is an expanded change detection benchmark dataset (e.g., [47]). Identifying changing or moving areas in the field of view of a camera is a fundamental preprocessing step in computer vision and video processing. Example applications include visual surveillance (e.g., people counting, action recognition, anomaly detection, post-event forensics), smart environments (e.g., room occupancy monitoring, fall detection, parking occupancy detection), and video retrieval (e.g., activity localization and tracking). On the other hand, **PANDA**[11] is the first gigaPixel-level human-centric video dataset (e.g., [18]), for large-scale, long-term, and multi-object visual analysis. The videos in PANDA were captured by a gigapixel camera and cover real-world large-scale scenes

[4] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/.

[5] https://www.image-net.org/.

[6] https://detrac-db.rit.albany.edu/.

[7] https://vehiclereid.github.io/VeRi/.

[8] http://www.nlpr.ia.ac.cn/iva/homepage/jqwang/Vehicle1M.htm.

[9] http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007.html.

[10] http://www.changedetection.net/.

[11] http://www.panda-dataset.com/.

with both wide field-of-view and high-resolution details (gigapixel-level/frame). The scenes may contain 4K headcounts with over 100× scale variation.

As mentioned above, this is just an illustration of many benchmarks available for the research community. Table 6.1 presents a summary of the approaches found in the literature for each of the modules in Fig. 6.1; furthermore, this table includes datasets and research papers that use them.

6.8 Conclusions

Novel solutions for smart cities are increasingly required in our everyday life, looking for efficient management of different services. This manuscript reviews computer vision-based approaches needed for applications, from counting people in crowded scenarios to vehicle classification or free parking slot detection. In recent years, due to the improvements in technology and the increase in computing power, a large number of these approaches have been implemented in real-time or even embedded in low-cost hardware. This chapter starts by proposing a pipeline to perform the review on a common framework. Initially, image processing techniques generally used on urban video analytics are reviewed. Then, object detection and classification approaches are summarized, highlighting that in several recent approaches, both tasks are merged in a single module. Once objects are detected and classified, an important component to be considered is the tracking, which was initially performed by Kalman filters, but recently, deep learning-based approaches have given the best results. Finally, the chapter reviews applications built on top of the modules of the proposed pipeline, as well as datasets generally used as benchmarks for evaluations and comparisons. As mentioned in several sections of the chapter, this work does not pretend to be an exhaustive list of approaches from the state-of-the-art for each module of the considered pipeline, but rather an illustration of some of the different recent contributions. As a first general conclusion, it could state that deep learning-based solutions are the best option for any of the pipeline's modules. As a second conclusion, it is clear that there are many opportunities to be explored based on video analytics of urban environments. Depending on the complexity of the solution, low-cost hardware can be considered. Most of the applications reviewed in this chapter are standalone solutions; the challenge for the future is to interconnect these applications to benefit each other and develop more robust solutions for smart cities.

Table 6.1 Papers summarized for each stage of the pipeline and technique used

Dataset	Preprocessing	Object detection	Object classification	Tracking	Applications
KITTI: [17, 22, 60, 71]	Camera calibration: [16, 23, 68, 72]	Faster R-CNN: [32]	Optical flow: [52]	Kalman Filter: [19]	Pedestrian Detection/Tracking: [1, 29, 56, 58]
PASCAL VOC 2007: [17, 71]	Inverse Perspective Mapping: [33, 49, 55]	Finite State Machine: [27]	Proyective Distortion: [10]	Extended Kalman Filter: [8, 20, 25]	License Plate Recognition: [2, 28, 50, 75]
MS COCO: [9, 71]	Perspective View: [35, 48]	Histogram equalization and RGB-Local Binary Pattern (RGB-LBP): [6]	MLP: [26]	Particle Filter: [5, 29]	Vehicle Counting: [49]
ImageNet: [9, 24]	Background Subtraction: [34, 37, 44, 47]	Optical Flow: [4]	Custom Filtering: [13, 46]	SORT: [35, 64]	Vehicle Classification: [10, 12]
UA-DETRAC: [45]	Mixture of Gaussians (MOG): [62]	CenterNet-3D object detection: [60]	Haar cascade, HOG and SVM: [31, 70]	Custom CNN: [22, 50]	Vehicle Trajectory: [15]
AVSS-2007: [27]	Privacy-preserving strategies: [28, 63]	Custom CNN: [17, 18, 30, 40, 71]	Faster R-CNN: [50]	DeepSORT: [56, 59]	Parking space: [33, 48]
VeRi-776: [50]	Image enhancement: [21, 57, 61, 74]	YOLOv4: [9]	Custom CNN: [24, 43, 65, 66, 73]	3D Tracking: [3, 53, 54]	Face detection: [31, 63]
CDnet: [47]					
Vehicle-1M: [36]					
PANDA: [18]					

Acknowledgements This work has been partially supported by the ESPOL projects TICs4CI (FIEC-16-2018) and PhysicalDistancing (CIDIS-56-2020); and the "CERCA Programme/ Generalitat de Catalunya". The authors acknowledge the support of CYTED Network: "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559).

References

1. Ahmed, I., Ahmad, M., Rodrigues, J.J., Jeon, G., Din, S.: A deep learning-based social distance monitoring framework for covid-19. Sustain. Cities Soc. **65**, 1–12 (2021)
2. Md Arafat, M.Y., Khairuddin, A.S.M., Paramesran, R.: Connected component analysis integrated edge based technique for automatic vehicular license plate recognition framework. Intell. Transp. Syst. **14**(7), 712–723 (2020)
3. Arafat, M.Y., Khairuddin, A.S.M., Paramesran, R.: Detection and classification of vehicles for traffic video analytics. Procedia Comput. Sci. **144**, 259–268 (2018)
4. Aslani, S., Mahdavi-Nasab, H.: Optical flow based moving object detection and tracking for traffic surveillance. Int. J. Electr., Comput., Energ., Electron. Commun. Eng. **7**(9), 1252–1256 (2013)
5. Tariq, S., Farooq, H., Jaleel, A., Wasif, S.M.: Anomaly detection with particle filtering for online video surveillance. IEEE Access **9**, 19457–19468 (2021)
6. Avola, D., Foresti, G.L., Martinel, N., Micheloni, C., Pannone, D., Piciarelli, C.: Aerial video surveillance system for small-scale uav environment monitoring. In: 14th International Conference on Advanced Video and Signal Based Surveillance, pp. 1–6. IEEE (2017)
7. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: International Conference on Image Processing, pp. 3464–3468. IEEE (2016)
8. Blom, H.A.P., Bar-Shalom, Y.: The interacting multiple model algorithm for systems with markovian switching coefficients. Trans. Autom. Control **33**(8), 780–783 (1988)
9. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: YOLOv4: optimal speed and accuracy of object detection (2020)
10. Bose, B., Grimson, E.: Improving object classification in far-field video. In: Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1–8. IEEE (2004)
11. Bouwmans, T., Javed, S., Sultana, M., Jung, S.K.: Deep neural network concepts for background subtraction: a systematic review and comparative evaluation. Neural Netw. **117**, 8–66 (2019)
12. Buch, N., Cracknell, M., Orwell, J., Velastin, S.A.: Vehicle localisation and classification in urban CCTV streams. In: ITS World Congress, pp. 1–8 (2009)
13. Canel, C., Kim, T., Zhou, G., Li, C., Lim, H., Andersen, D.G., Kaminsky, M., Dulloor, S.: Scaling video analytics on constrained edge nodes (2019). arXiv:1905.13536
14. Cao, Mingwei, Zheng, Liping, Jia, Wei, Liu, Xiaoping: Joint 3D reconstruction and object tracking for traffic video analysis under Jov environment. Trans. Intell. Transp. Syst. **22**(6), 3577–3591 (2020)
15. Cao, X., Changxia, W., Lan, J., Yan, P., Li, X.: Vehicle detection and motion analysis in low-altitude airborne video under urban environment. Trans. Circuits Syst. Video Technol. **21**(10), 1522–1533 (2011)
16. Caprile, B., Torre, V.: Using vanishing points for camera calibration. Int. J. Comput. Vision **4**(2), 127–139 (1990)
17. Chandrakar, R., Raja, R., Miri, R., Sinha, U., Kushwaha, A.K.S., Raja, H.: Enhanced the moving object detection and object tracking for traffic surveillance using RBF-FDLNN and CBF algorithm. Expert Syst. Appl. **191**, 1–15 (2022)
18. Chen, K., Wang, Z., Wang, X., Gong, D., Yu, L., Guo, Y., Ding, G.: Towards real-time object detection in gigapixel-level video. Neurocomputing (2021)
19. Chen, Z., Ellis, T., Velastin, S.A.: Vehicle detection, tracking and classification in urban traffic. In: 15th International Conference on Intelligent Transportation Systems, pp. 951–956. IEEE (2012)
20. Cho, H., Seo, Y.W., Kumar, B.V., Rajkumar, R.R.: A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In: International Conference on Robotics and Automation, pp. 1836–1843. IEEE (2014)
21. Cucchiara, R., Grana, C., Piccardi, M., Prati, A., Sirotti, S.: Improving shadow suppression in moving object detection with HSV color information. In: Intelligent Transportation Systems, pp. 334–339. IEEE (2001)

22. Pino, I.D., Vaquero, V., Masini, B., Sola, J., Moreno-Noguer, F., Sanfeliu, A., Andrade-Cetto, J.: Low resolution lidar-based multi-object tracking for driving applications. In: Iberian Robotics Conference, pp. 287–298. Springer (2017)
23. Deutscher, J., Isard, M., MacCormick, J.: Automatic camera calibration from a single manhattan image. In: European Conference on Computer Vision, pp. 175–188. Springer (2002)
24. Dey, B., Kundu, M.K.: Turning video into traffic data-an application to urban intersection analysis using transfer learning. IET Image Process. **13**(4), 673–679 (2019)
25. Dyckmanns, H., Matthaei, R., Maurer, M., Lichte, B., Effertz, J., Stüker, D.: Object tracking in urban intersections based on active use of a priori knowledge: active interacting multi model filter. In: Intelligent Vehicles Symposium, pp. 625–630. IEEE (2011)
26. Silva, R.R., Aires, K.R., Veras, R.D.: Detection of helmets on motorcyclists. Multimed. Tools Appl. **77**(5), 5659–5683 (2018)
27. Fan, Q., Pankanti, S.: Modeling of temporarily static objects for robust abandoned object detection in urban surveillance. In: 8th International Conference on Advanced Video and Signal Based Surveillance, pp. 36–41. IEEE (2011)
28. Frome, A., Cheung, G., Abdulkader, A., Zennaro, M., Wu, B., Bissacco, A., Adam, H., Neven, H., Vincent, L.: Large-scale privacy protection in google street view. In: 12th International Conference on Computer Vision, pp. 2373–2380. IEEE (2009)
29. Gaddigoudar, P.K., Balihalli, T.R., Ijantkar, S.S., Iyer, N.C., Maralappanavar, S.: Pedestrian detection and tracking using particle filtering. In: International Conference on Computing, Communication and Automation, pp. 110–115 (2017)
30. Gao, C., Li, P., Zhang, Y., Liu, J., Wang, L.: People counting based on head detection combining Adaboost and CNN in crowded surveillance environment. Neurocomputing **208**, 108–116 (2016)
31. Gautam, K.S., Thangavel, S.K.: Video analytics-based intelligent surveillance system for smart buildings. Soft. Comput. **23**(8), 2813–2837 (2019)
32. Gavrilescu, R., Zet, C., Foşalău, C., Skoczylas, M., Cotovanu, D.: Faster R-CNN: an approach to real-time object detection. In: International Conference and Exposition on Electrical and Power Engineering, pp. 165–168 (2018)
33. Grassi, G., Jamieson, K., Bahl, P., Pau, G.: Parkmaster: an in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments. In: Proceedings of Symposium on Edge Computing, pp. 1–14 (2017)
34. Graszka, P.: Median mixture model for background–foreground segmentation in video sequences 103–110 (2014)
35. Grents, A., Varkentin, V., Goryaev, N.: Determining vehicle speed based on video using convolutional neural network. Transp. Res. Procedia **50**, 192–200 (2020)
36. Guo, H., Zhao, C., Liu, Z., Wang, J., Hanqing, L.: Learning coarse-to-fine structured feature embedding for vehicle re-identification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, pp. 1–8 (2018)
37. Gupte, S., Masoud, O., Martin, R.F., Papanikolopoulos, N.P.: Detection and classification of vehicles. Trans. Intell. Transp. Syst. **3**(1), 37–47 (2002)
38. Hamida, A.B., Koubaa, M., Amar, C.B., Nicolas, H.: Toward scalable application-oriented video surveillance systems. In: Science and Information Conference, pp. 384–388. IEEE (2014)
39. Jodoin, J.P., Bilodeau, G.A., Saunier, N.: Urban tracker: multiple object tracking in urban mixed traffic. In: Winter Conference on Applications of Computer Vision, pp. 885–892. IEEE (2014)
40. Junos, M.H., Khairuddin, A.S.M., Dahari, M.: Automated object detection on aerial images for limited capacity embedded device using a lightweight CNN model. Alex. Eng. J. (2021)
41. Kalman, R.E.: A new approach to linear filtering and prediction problems. J. Basic Eng. **82**(1), 35–45 (1960)
42. Kuhn, H.W.: The hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1–2), 83–97 (1955)
43. Kumar, T.S.: Video based traffic forecasting using convolution neural network model and transfer learning techniques. J. Innov. Image Process. **2**(03), 128–134 (2020)

44. Lee, B., Hedley, M.: Background estimation for video surveillance. Image Vis. Comput. N. Z. 315–320 (2002)
45. Li, C., Dobler, G., Feng, X., Wang, Y.: Tracknet: simultaneous object detection and tracking and its application in traffic video analysis, pp. 1–10 (2019). arXiv:1902.01466
46. Li, Y., Padmanabhan, A., Zhao, P., Wang, Y., Xu, G.H., Netravali, R.: Reducto: on-camera filtering for resource-efficient real-time video analytics. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 359–376 (2020)
47. Lim, K., Jang, W.D., Kim, C.S.: Background subtraction using encoder-decoder structured convolutional neural network. In: 14th International Conference on Advanced Video and Signal Based Surveillance, pp. 1–6. IEEE (2017)
48. Ling, X., Sheng, J., Baiocchi, O., Liu, X., Tolentino, M.E.: Identifying parking spaces & detecting occupancy using vision-based IoT devices. In: Global Internet of Things Summit, pp. 1–6. IEEE (2017)
49. Liu, C., Huynh, D.Q., Sun, Y., Reynolds, M., Atkinson, S.: A vision-based pipeline for vehicle counting, speed estimation, and classification. Trans. Intell. Transp. Syst. (2020)
50. Liu, X., Liu, W., Mei, T., Ma, H.: A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In: European Conference on Computer Vision, pp. 869–884. Springer (2016)
51. Liu, X., Sang, J., Weiqun, W., Liu, K., Liu, Q., Xia, X.: Density-aware and background-aware network for crowd counting via multi-task learning. Pattern Recogn. Lett. **150**, 221–227 (2021)
52. Makhmutova, A., Anikin, I.V., Dagaeva, M.: Object tracking method for videomonitoring in intelligent transport systems. In: International Russian Automation Conference, pp. 535–540. IEEE (2020)
53. Naik, U.P., Rajesh, V., Kumar, R., et al.: Implementation of YOLOv4 algorithm for multiple object detection in image and video dataset using deep learning and artificial intelligence for urban traffic video surveillance application. In: International Conference on Electrical, Computer and Communication Technologies, pp. 1–6. IEEE (2021)
54. Nguyen, T.-N., Michaelis, B., Al-Hamadi, A., Tornow, M., Meinecke, M.-M.: Stereo-camera-based urban environment perception using occupancy grid and object tracking. Trans. Intell. Transp. Syst. **13**(1), 154–165 (2011)
55. Noh, B., No, W., Lee, J., Lee, D.: Vision-based potential pedestrian risk analysis on unsignalized crosswalk using data mining techniques. Appl. Sci. **10**(3), 1–21 (2020)
56. Praveenkumar, S.M., Patil, P., Hiremath, P.S.: Real-time multi-object tracking of pedestrians in a video using convolution neural network and Deep SORT. In: ICT Systems and Sustainability, pp. 725–736. Springer (2022)
57. Qu, H., Yuan, T., Sheng, Z., Zhang, Y.: A pedestrian detection method based on YOLOv3 model and image enhanced by retinex. In: 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, pp. 1–5. IEEE (2018)
58. Ridel, D., Rehder, E., Lauer, M., Stiller, C., Wolf, D.: A literature review on the prediction of pedestrian behavior in urban scenarios. In: 21st International Conference on Intelligent Transportation Systems, pp. 3105–3112. IEEE (2018)
59. Santos, A.M., Bastos-Filho, C.J., Maciel, A.M., Lima, E.. Counting vehicle with high-precision in Brazilian roads using YOLOv3 and Deep SORT. In: 33rd Conference on Graphics, Patterns and Images, pp. 69–76. IEEE (2020)
60. Yuguang Shi, Yu., Guo, Z.M., Li, X.: Stereo CenterNet-based 3D object detection for autonomous driving. Neurocomputing **471**, 219–229 (2022)
61. Shi, Z., Guo, B., Zhao, M., Zhang, C., et al.: Nighttime low illumination image enhancement with single image using bright/dark channel prior. J. Image Video Process. **2018**(1), 1–15 (2018)
62. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: Proceedings of Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 246–252. IEEE (1999)

63. Tu, N.A., Wong, K.S., Demirci, M.F., Lee, Y.K., et al.: Toward efficient and intelligent video analytics with visual privacy protection for large-scale surveillance. J. Supercomput. 1–31 (2021)
64. Velesaca, H.O., Araujo, S., Suárez, P.L., Sánchez, A., Sappa, A.D.: Off-the-shelf based system for urban environment video analytics. In: International Conference on Systems, Signals and Image Processing, pp. 459–464 (2020)
65. Vishnu, C., Singh, D., Mohan, C.K., Babu, S.: Detection of motorcyclists without helmet in videos using convolutional neural network. In: International Joint Conference on Neural Networks, pp. 3036–3041. IEEE (2017)
66. Wang, C., Cheng, M., Sohel, F., Bennamoun, M., Li, J.: NormalNet: a voxel-based CNN for 3D object classification and retrieval. Neurocomputing **323**, 139–147 (2019)
67. Wei, H., Laszewski, M., Kehtarnavaz, N.: Deep learning-based person detection and classification for far field video surveillance. In: 13th Dallas Circuits and Systems Conference, pp. 1–4. IEEE (2018)
68. Wildenauer, H., Micusik, B.: Closed form solution for radial distortion estimation from a single vanishing point. In: BMVC, vol. 1, pp. 1–11 (2013)
69. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: International Conference on Image Processing, pp. 3645–3649. IEEE (2017)
70. Xu, R., Nikouei, S.Y., Chen, Y., Polunchenko, A., Song, S., Deng, C., Faughnan, T.R.: Real-time human objects tracking for smart surveillance at the edge. In: International Conference on Communications, pp. 1–6. IEEE (2018)
71. Zhang, H., Wang, K., Tian, Y., Gou, C., Wang, F.-Y.: MFR-CNN: incorporating multi-scale features and global information for traffic object detection. Trans. Veh. Technol. **67**(9), 8019–8030 (2018)
72. Zhang, M., Yao, J., Xia, M., Li, K., Zhang, Y., Liu, Y.: Line-based multi-label energy optimization for fisheye image rectification and calibration. In: Proceedings of Conference on Computer Vision and Pattern Recognition, pp. 4137–4145. IEEE Computer Society (2015)
73. Zhu, J., Sun, K., Jia, S., Li, Q., Hou, X., Lin, W., Liu, B., Qiu, G.: Urban traffic density estimation based on ultrahigh-resolution UAV video and deep neural network. J. Sel. Top. Appl. Earth Obs. Remote. Sens. **11**(12), 4968–4981 (2018)
74. Zotin, A.: Fast algorithm of image enhancement based on multi-scale retinex. Procedia Comput. Sci. **131**, 6–14 (2018)
75. Zou, Y., Zhang, Y., Yan, J., Jiang, X., Huang, T., Fan, H., Cui, Z.: Zhongwei: license plate detection and recognition based on YOLOv3 and ILPRNET, pp. 1–8. Signal, Image and Video Processing (2021)

Chapter 7
Multimodal Sensor Calibration Approaches in the ATLASCAR Project

Vitor Santos, Paulo Dias, Miguel Oliveira, and Daniela Rato

Abstract An important issue in smart cities is intelligent mobility. In this line, the ATLASCAR project started in 2009 as a challenge to transpose to a real car many perception and navigation techniques developed for small scale vehicles in robotic competitions. Among many others, that challenge brought the need to use multiple sensors from different modalities for a richer and more robust perception of the road and its agents. The first issues concerned the proper registration and calibration of one LiDAR sensor and one visual camera, but rapidly evolved to multiple LiDARs and cameras of variate natures. In this paper, we present several calibration techniques that were proposed, developed, applied and tested along the several years of the project providing interesting practical solutions and the means and tools to merge and combine sensorial data from multiple sources. Based on the multiple detection of specific targets, like cones, spheres and traditional checkerboards or charuco boards, the techniques range from point cloud and data matching, using several techniques, including deep learning trained classifiers, up to holistic optimization techniques. The developments have also been applied in other contexts such as 3D scene reconstruction or in industrial manufacturing work cells.

7.1 Introduction

The ATLASCAR project appeared after several years of research and development in small scale mobile robots for autonomous driving competitions [30]. The successes and results obtained lead to a full scale challenge in a real automobile [43, 44]. The

V. Santos (✉) · P. Dias · M. Oliveira · D. Rato
IEETA, Universidade de Aveiro, 3810 Aveiro, Portugal
e-mail: vitor@ua.pt

P. Dias
e-mail: paulo.dias@ua.pt

M. Oliveira
e-mail: mriem@ua.pt

D. Rato
e-mail: danielarato@ua.pt

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_7

Fig. 7.1 The vehicles used in the ATLASCAR project. The first on the left (ATLASCAR1) was used between 2010 and 2014 and vehicle on the right (ATLASCAR2) has been in use since 2015

main purpose of the project is to carry out studies and developments in perception and navigation in the domain of autonomous driving and driver assistance systems. Figure 7.1 shows the two cars used in the project since 2010 equipped with several sensors, namely cameras and (LiDAR) sensors.

Locomotion and control issues are very important for autonomous and semi-autonomous navigation, but perception has been the greatest concern of the authors since the beginning of project. Moreover, in that line, sensorial multi-modality was a strategic decision by integrating from the early beginning of the project monocular, stereo and thermal cameras and 2D and 3D LiDARs. The merging and combination of such a wealth of data sources requires obviously the proper sensor calibration, which, in this context, refers to the estimation of a geometric registration of all individual sensor reference frames in the car.

From all the modal combinations possible, the first endeavour was the pair 3D LiDAR—2D LiDAR registration, because of its high novelty at the time since 3D LiDARs were still very rare and absolutely not off-the-shelf components. Indeed, a 3D LiDAR sensor had been developed within the group since the year 2003 [26] and later applied and improved in several contexts [12, 31, 32].

Although sensor calibration is traditionally a relevant problem that has been solved in several ways in the literature for image cameras, it was not so common for LiDARs at the early stages of the project. Moreover, most solutions require a high degree of human intervention to assist the calibration process. That may not be an issue if calibration is to be performed once, but on a mobile platform sensors are subject to mechanical perturbations and calibration may be required to be performed more often, and a swift process is welcome for the operators.

This concern was also a drive for this research line, and this paper describes four main phases of this process throughout the years, both in the usage of specific detection targets, up to computational and algorithmic approaches to perform the calibration. In brief, and to be detailed further, the main topics are the following:

- Calibration based on point clouds from direct target detection;
- Calibration based on derived point clouds after the detection of a spherical target;
- Enhanced techniques for the detection of target in cameras using deep learning;

- Holistic optimization for the registration of sensors.

As a critical component of intelligent or robotic systems, the topic of extrinsic calibration has been frequently addressed in the literature over the past decades. The large bulk of these works has been focused on a particular case of sensor to sensor calibration problems, which is the case of systems containing a single pair of sensors, i.e. pairwise calibrations. In this regard, the RGB to RGB camera calibration has been the most covered case: [13, 24, 27, 35, 41, 49], but other combinations of modalities exist, such as RGB to depth camera calibration [4, 7, 19, 22, 36, 52], RGB camera to 2D LiDAR [9, 16, 17, 35, 37, 47, 52], 2D LiDAR to 3D LiDAR [2]; RGB camera to 3D LiDAR [16, 23], RGB camera to radar [11], etc.

The problem of RGB to RGB camera has been thoroughly addressed by the research community, in particular in stereo systems configurations [13, 24, 27, 41, 49]. Typically, these approaches function by carrying out an iterative procedure which estimates the transformation between the cameras using the reprojection error as guidance. The optimization procedure may also include the estimation of the intrinsic parameters as in [49]. These calibration methodologies have also been proposed to address the problem of online calibration [27, 41] as well as markerless calibration [24].

Calibration procedures often require the use of targets such as the well-known chessboard pattern [37] widely used in several libraries for camera calibration. However other targets may also be used, in particular when involving the calibration of depth information that can be significantly simplified with the use of predefined geometries in the scene, such as spheres [22, 35, 39, 42], or cones [2, 25].

Finding a visible calibration pattern is also a challenge when calibrating thermal cameras with other modalities. To address this, [48] proposes a custom-built checkerboard with square holes where calibration points in each frame are located with a pattern-finding algorithm. A near-optimal subset of valid frames is selected using an implementation of the enhanced Monte Carlo method followed by an optimization algorithm is used to fit a distortion model (for intrinsic calibration) or a six-parameter pose model (for extrinsic calibration) to the data.

More recent systems that use deep learning are even aiming for a targetless calibration, from which several examples already started appearing in the community [3, 18, 46, 53].

The paper follows with four main sections, addressing the four topics presented before, with a detailed description of the approach along with some results: Sects. 7.2, 7.3, 7.4 and 7.5. In the end, a comparative summary of the methods and techniques is given and some perspectives for the future are drawn.

7.2 Calibration of 3D and 2D LiDARs with Conical Targets

In 2010, the first attempt to improve the multimodal calibration process used in the ATLASCAR project was performed on an instrumented real automobile. The initial

Fig. 7.2 ATLASCAR1 and the LiDAR sensors on-board (left). Conic calibration target (right)

vehicle was equipped with a 3D laser scanner based on a rotating 2D LiDAR [26] and two 2D SICK LMS 151 LiDARs mounted on the front bumpers (Fig. 7.2 left).

A calibration object was used to assist the unambiguous determination of the position and orientation of the 2D laser footprints regarding the 3D point cloud. A conic geometry was selected given its ability to remove ambiguity in height. The combination with a second cone provided enough information to determine the orientation around the vertical axis. An intermediate plane was added to keep the cones at a fixed known distance while providing a platform to add calibration patterns if necessary. The final calibration object can be seen in the right part of Fig. 7.2 during an acquisition. Two conic shapes connected by a planar surface complete the target to be detected simultaneously by 3D and 2D LiDARs. The process of calibration involves the steps described in the next sections.

7.2.1 Proposed Calibration Process

The next subsections describe the calibration procedure. First we describe how the initial estimates are defined by the user, then we discuss the fitting of the 2D ellipses into the observed data, and finally we present the methodology to estimate the geometric transformation between the sensors.

7.2.1.1 Manual Input of Initial Estimates

The process of calibration is semi-automatic as it requires to specify manually the position of the calibration object in the 3D point cloud. The user is required to select a point near the center of the calibration object (P_c in Fig. 7.3) and an additional point in the plane between the two cones (P_b in Fig. 7.3). Using this initial approximation, a 3D point cloud of the virtual calibration object is registered using the Iterative

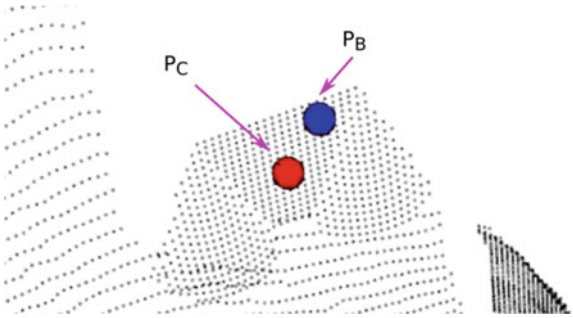

Fig. 7.3 User initial estimate of the target in the 3D point cloud with the manual selection of 2 points (a point near the center of the calibration object P_C and a second point on the plane P_B)

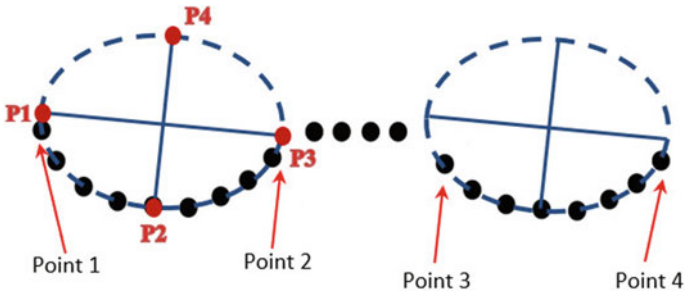

Fig. 7.4 User initial estimate and ellipse fitting operations in 2D point cloud: 4 points (Point 1 to Point 4) are selected by the user consisting of the extremes of the ellipses captured in the two conical surfaces of the pattern. The estimated ellipses are also presented

Closest Point providing an initial estimate of the position of the calibration object in the dataset.

Given the position of the calibration object in the 3D data, it is necessary to determine the location of the calibration object in the 2D LiDARs on which the cones in the calibration object produce two ellipse sections. Once again, an initial estimate is provided by the user who selects four points corresponding to the beginning and the end of the two ellipses in each 2D data sets (Points 1–4 in Fig. 7.4).

7.2.1.2 Estimation of 2D Ellipses pose Relatively to the Calibration Object

Based on the initial user selection, an ellipse fitting algorithm estimates the characteristics of the ellipse that best fits the data (center, major and minor axis length, and the angle of the major axis). Since a specific ellipse has only one way to fit geometrically into a cone in terms of height and inclination, the only undetermined variable left would be the rotation angle around the vertical axis of the cone that

corresponds to the major axis between the ellipse and the plane of the calibration object. Mathematical details on the computation of the 2D ellipse fitting algorithm can be found in [2].

7.2.1.3 2D Laser Transform Calculation

From the analytical model of each ellipse, four 2D points (P_1 to P_4) are computed for each ellipse at the intersections of the ellipse with the major and minor axes, as shown in Figs. 7.4 and 7.5. The coordinates are computed according to Eqs. (7.1), (7.2), (7.3) and (7.4), where R_x and R_y are the major and minor axis of the ellipse, C_x and C_y are the coordinates of the ellipse center and α the ellipse rotation angle in the 2D plane.

$$P_1 = \left(C_x + \frac{R_x}{2}\cos(\alpha), C_y + \frac{R_x}{2}\sin(\alpha) \right) \tag{7.1}$$

$$P_2 = \left(C_x + \frac{R_y}{2}\cos\left(\frac{\pi}{2}+\alpha\right), C_y + \frac{R_y}{2}\sin\left(\frac{\pi}{2}+\alpha\right) \right) \tag{7.2}$$

$$P_3 = \left(C_x - \frac{R_x}{2}\cos(\alpha), C_y - \frac{R_x}{2}\sin(\alpha) \right) \tag{7.3}$$

$$P_4 = \left(C_x - \frac{R_y}{2}\cos\left(\frac{\pi}{2}+\alpha\right), C_y - \frac{R_y}{2}\sin\left(\frac{\pi}{2}+\alpha\right) \right) \tag{7.4}$$

Since the process is repeated for the two ellipses, 8 corresponding points are obtained for each dataset making it possible to estimate the rigid body transformation that best fits those 8 points from the 2D laser data into the 8 transformed points in the 3D laser data. This transformation corresponds to the 2D laser calibration matrix.

7.2.2 Results

A real scale prototype of the calibration object was built to perform real data acquisition (see Fig. 7.2 right). To avoid the construction of several calibration objects, we simulate various object locations by capturing several times the same scene with the calibration object at different positions and merging the acquisitions into a single point cloud with the required number of calibration objects. Tests were made using three and four objects.

Results of the calibration using four calibration objects are shown in Fig. 7.6. In green we present the initial calibration based on manual measurements and in purple the result using the proposed calibration process. An improvement of the alignment with the 3D laser data is obvious (see the details of the alignment with the calibration object in the red spheres in Fig. 7.7). The main limitation of the proposed approach is visible in the red sphere in Fig. 7.6: the resulting calibration (purple) although

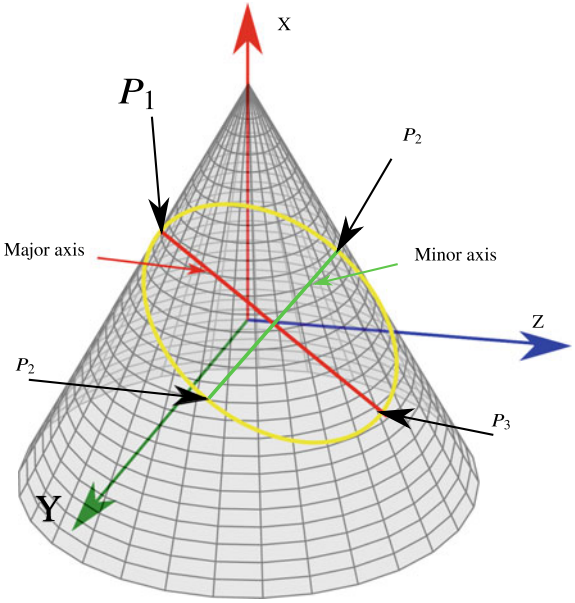

Fig. 7.5 Ellipse 3D fitting in the pattern cone

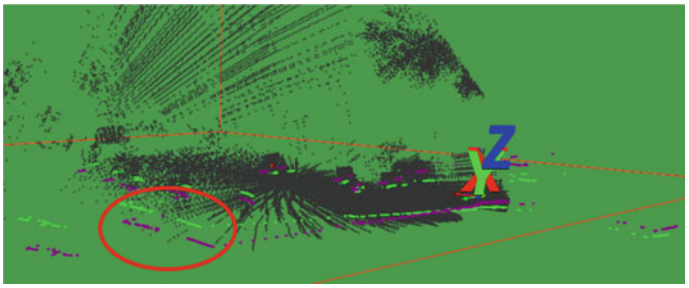

Fig. 7.6 Results of the proposed calibration method: overview of the scene

much better close to the calibration objects suffers from a rotation around the X axis (direction forward of the 3D LiDAR) noticeable in points far from the calibration objects. This issue is mainly due to the limited resolution of the 2D LiDARs that provides insufficient data for a precise determination of the ellipse attributes (in some cases the fitting was computed with only 8 points per ellipse). Errors in the ellipse parameters results in errors in the fitting on the cone (in particular height position and rotation) and thus significant rotation errors in the calibration process.

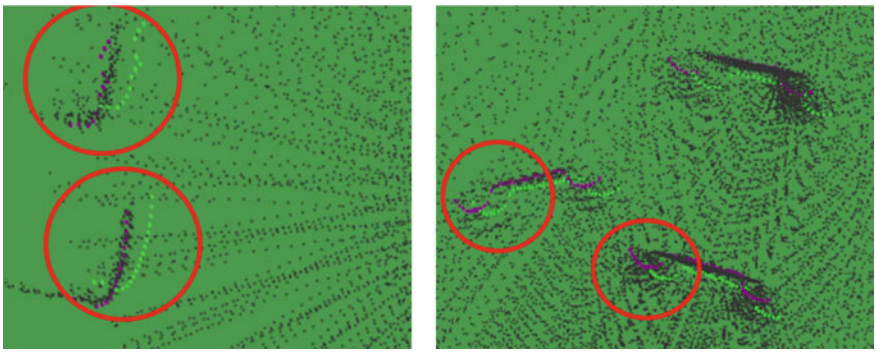

Fig. 7.7 Details of the scene (left) and of the calibration objects (right)

7.3 Spherical Target to Calibrate LiDARs and Cameras

The calibration of sensors with different principles and data representations requires a target detectable by all sensors so the registration can be made on each data capture independently of the point of view or placement of the sensor with respect to that target. So, this requires the definition of a target with those properties and, if possible, easy to replicate and manage, and prone for automatic or advanced semi-automatic calibration techniques.

7.3.1 Principles of the Calibration Approach

The target geometry that best fits the mentioned requirements is a sphere. The projection of a sphere in a plane on a pinhole camera model will always be a circle or an ellipse, and any cross section of the sphere with a plane will also always be a circle, as occurs in the planar LiDAR range measurements.

Detecting a sphere with known dimensions allows to obtain its center in a 3D coordinate frame. If multiple positions of a moving sphere are captured synchronously by multiple sensors, their corresponding centers will define a set of points following the motion of the sphere. Each sensor will then have gathered a set of points from its own point of view. If those sets of points are matched by some mathematical procedure, it is possible to obtain the relative positions of these sensors among them.

The sensors involved in the ATLASCAR are mainly cameras that provide images, and LiDARs that provide point clouds obtained after range data.

Detection of a sphere in visual data can be more or less challenging depending on the illumination conditions. A colored target can help the detection procedure. Also, the circular (or nearly circular) shape can be detected by circle-oriented Hough transforms in edge images obtained, for example, by Canny edge detectors.

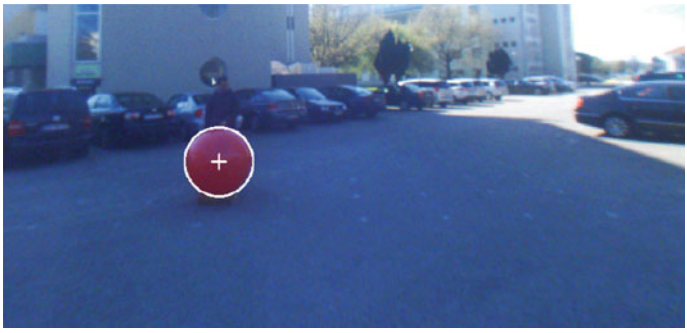

Fig. 7.8 Ball detected in the image using classical computer vision techniques

Detection of the sphere in point clouds has two main fronts: single plane LiDAR scan (2D laser range finder) or multiplane LiDAR scan (3D laser range finder).

Next sections explain these approaches in more detail.

7.3.1.1 Detecting a Sphere in an Image

The projection of a sphere in an image is either a circle or an ellipse. If the optical axis intersects the sphere center, we have a perfect circle, but away from that, ellipses appear in the projected image. Their shape or eccentricity is affected by the focal distance of the camera and, in many cases, it is hardly noticeable at naked eye when the sphere center remains close enough to the optical axis of the camera. Anyway, the center point of the ellipse (projection of sphere) corresponds to the center of the actual sphere.

Detecting the sphere in the image is then reduced to the detection of a circular or, in the limit, elliptical shape in the image. In both cases, the center corresponds to the projection of the actual ball center.

As described in [35], several techniques were used to detect the ball with a mix of color segmentation, edge detection and Hough Transform for circles (albeit with poorer results) and also with an approach based on the Ramer-Douglas–Peucker algorithm [14, 38], that is available in the OpenCV library as the `approxPolyDP` function. Results were reasonable and provided a good estimation of the ball (Fig. 7.8), but the dependency on illumination conditions led later to a more robust technique as described ahead in Sect. 7.4, which is also applicable to other types of images.

7.3.1.2 Detecting a Sphere in a Point Cloud

LiDAR sensors generate two possibilities of point clouds: 2D and 3D. In 2D point clouds, the sphere will appear as the cross section made by a plane, and we have

points of a circular arc. In 3D, the point cloud is richer, and we have points from a spherical cap.

Detecting the circular arcs or the spherical caps can be done with appropriate algorithms. For the spherical cap the best solution was based on RANSAC [15] techniques to fit the points to a spherical model by using functions available in Point Cloud Library (PCL).

To detect arcs, the number of points is too small (sometimes less than 10 points) to use RANSAC techniques. So, for arcs in 2D point clouds, first a cluster segmentation based on Nearest Neighbour (NN) techniques is used to isolate segments; then these segments are tested to check whether they define an arc or not. The work of Pereira et al. [35], inspired on the earlier works of Xavier et al. [51], explains in detail the technique with the mathematics formulation, using the Internal Angle Variance (IAV), that was successfully applied with quite acceptable results in detecting the center and radius of the circle, even with a limited number of points.

7.3.2 Geometric Transformations from Sets of Ball Centers

The center of the ball is discovered on the data from the several sensors (visual cameras and 2D and 3D range sensors) for a set of different positions. This generates as many sets of points as the sensors in the setup. Each set of points (ball centers) can be seen as a "temporal point cloud" (TPC) and then a pairwise match of these temporal point clouds is made to extract the mutual geometric transformations. The mathematical procedure can use Iterative Closest Point (ICP) [5, 8] or, if the points of the cloud are named and clouds are of the same size (which is mostly the case in this problem), SVD techniques are suited and effective. The technique is explained ahead in Sect. 7.4.3 along with the main equations used.

Figure 7.9 shows the temporal point clouds from four sensors in ATLASCAR1 [45]. The points in each temporal point cloud represent the successive positions of the ball (its center) during a motion operated by a user that tried to collect data sufficiently representative for the expectation of good geometric transformations between the reference sensor and all the remainders. The point clouds shown in Fig. 7.9 are represented from the respective sensor point of view. As we know that all point clouds originated at the same points (the ball centers), this allows the calculation of the mutual geometric transformations, hence the extrinsic calibration of all the sensors.

7.3.3 Results

The technique of using a sphere as target was successful because it is relatively easy and practical to use. The more ball positions, the more accurate can the calibration be. One traditional issue in assessing the quantitative results of calibration techniques

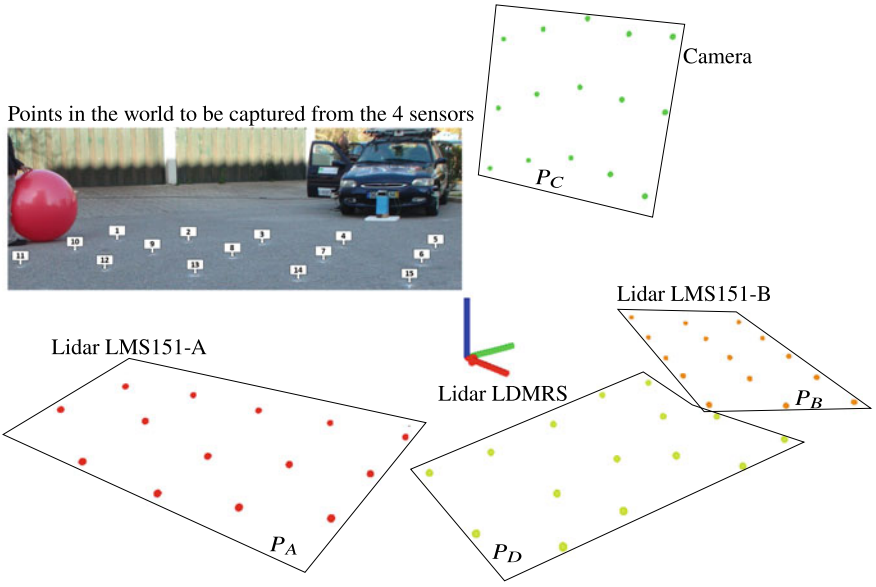

Fig. 7.9 Example of temporal point clouds from 4 different sensors in ATLASCAR1. Although not apparent in the illustration, the temporal point clouds P_A, P_B, P_C, P_D are actually sets of points in 3D and not points laying necessarily in a plane

in experimental setups is the absence of a Ground Truth. Therefore, studies are made by varying some parameters as well as performing reprojection of points with the calibrated frames and assess some error or variation in the expected results.

In this approach, those issues also occur. So, two types of results were used to assess the performance of the entire workflow with a real setup: consistency of ball detection and consistency of the calculated geometric transformations.

7.3.3.1 Consistency of Ball Detection

In this test, the ball was placed steady at different distances from the range sensors and, for each position, a few hundred samples of coordinates of the ball center were acquired from which a mean and a standard deviation were calculated. A standard deviation of less than 20 mm was observed for 2D Sick LMS151 sensors up to 6 meters, and also for a 3D SwissRanger sensor up to 3 meters, as shown in Fig. 7.10.

7.3.3.2 Consistency of the Geometric Transformation

A second experiment was made to test the consistency of the geometric transformation obtained, which is the actual purpose of the calibration process. Keeping the

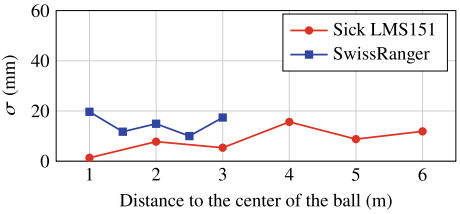

Fig. 7.10 Standard deviation of the detection of the ball center for several ranges for one 2D LiDAR and for a 3D range finder (SwissRanger)

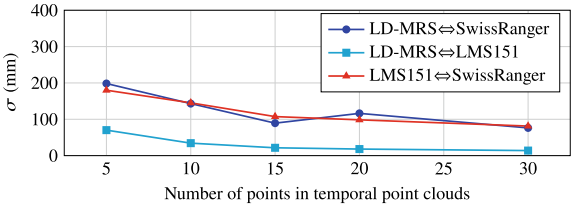

Fig. 7.11 Standard deviation of the translation between the three sensor pairs for 20 experiments of the calibration process

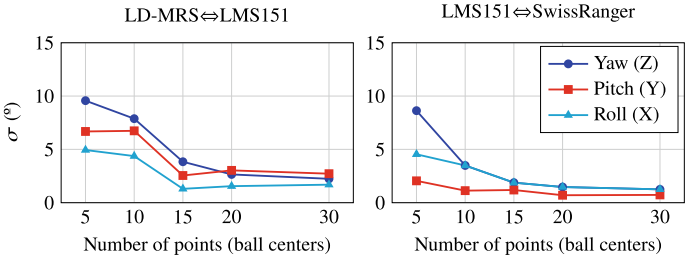

Fig. 7.12 Standard deviation of the Euler angles between two pairs of the sensors used

sensors in the same place, 20 experiments of calibration were executed. Figure 7.11 shows the standard deviation of the translation absolute errors when comparing three range sensors. Different sizes of temporal point clouds (number of ball positions) were used. As expected, the error decreases for larger numbers of ball positions. Also, the results with the 3D sensor (SwissRange) are less accurate than with the other LIDAR sensors.

The Euler angles variation was also tested and the effect of the number of points is even more pertinent. A minimum of 15 or 20 points ensures a smaller error in the Euler angles as can be seen in Fig. 7.12.

7.4 Deep Learning to Detect a Sphere in Multimodal Images

One limitation found in the detection of the ball occurs in visual images where the illumination conditions can seriously influence the quality and precision of the detection. Despite some techniques based on colour and shape, and some other based on ground subtraction, the results of ball segmentation in images were far from perfect because of the dependence with environment illumination. A solution had to be devised to make the calibration process more straightforward for visual images. The calibration philosophy is the same as earlier, but the detection methodology is completely new and takes advantage of a learning based approach to detect the ball in images.

The approach begins with the intrinsic calibration of each sensor using the ROS camera calibration tools. After retrieving the sensor data for the chosen system, the RGB information is converted to monochromatic images, and the depth map is used as an input for depth cameras. The pre-trained RCNN network with the Open Images Dataset [20, 21] operates on each captured image and detects the ball's position. Images with partial or no detection are not considered for calibration.

The coordinates of the ball in the image in pixels,(x_{pix}, y_{pix}), are obtained by applying the equations of the pinhole model. The distance from the ball centre to the camera, Z_c, is also retrieved from the pinhole model equations, and consequently, we can calculate the remainder X_c and Y_c coordinates in meters. At this point, the correction of the ball apparent diameter is necessary to compensate for the illusion that the camera is seeing the entire ball when it only sees a fraction of it. Finally, a SVD algorithm calculates the transformations between coordinate frame systems.

Figure 7.13 shows a diagram of the main steps of the methodology developed for the extrinsic calibration with a spherical target detected with a neural network trained using deep learning.

7.4.1 Detection of the Calibration Target using the Same Neural Network

To train the neural network, we converted the class "Ball" images of the Open Images Dataset [20, 21] to monochromatic encoding. This presents a significant advantage because it allows using the same trained network to access images from different modalities with the same accuracy. In addition, color images can be easily converted to monochrome, making the methodology independent of colors. It also helps to lower the effects of illumination factors. This has been tested in RGB, monochromatic, depth and IR modalities.

This was tested using Detectron2 framework [50] with a Faster R-CNN architecture [40]. The selected network is a R50-C4 1x with a training time of 0.551 s/iteration, an inference time of 0.102 s/image and a training memory of 4.8GB. The

network was configured to train with the grayscale converted "Ball" class, using 7000 iterations with a learning rate of 0.00025 and a batch size of 512 per image. A total of 6845 images were used to train. Figure 7.14 shows an example of the spherical target being detected with the trained network in the visual and depth images.

7.4.2 *Conversion of the Coordinates of the Center of the Ball to Meters*

This method is developed under the assumption that the centre of the bounding box is coincident with the centre of the ball. If the bounding box is not approximately square, that frame is discarded because either it corresponds to a bad or partial detection of the ball.

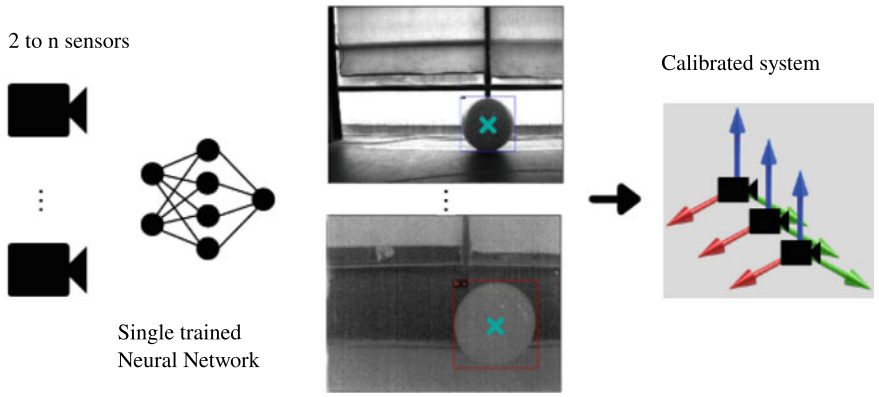

Fig. 7.13 Main steps of the methodology to detect a ball in multimodal image sources: visual (top) and thermal (bottom) images are shown

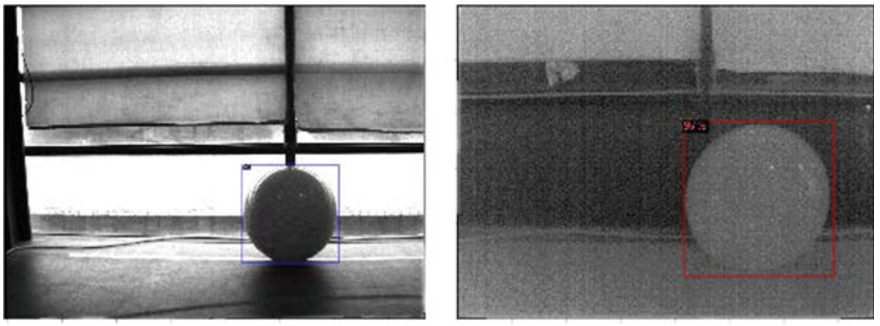

Fig. 7.14 Example of the spherical target being detected in the visual (right) and depth (left) images

Equation (7.5) represents the well-known pinhole model equation that relates the pixel coordinates in an image and the corresponding coordinates in the world coordinate system in meters. In this particular case, (x_{pix}, y_{pix}) represents the centre of the ball in the image in pixels, and the point (X_c, Y_c, Z_c) represents the centre of the ball in meters. f_x and f_y are the camera's focal lengths in both directions, and (C_x, C_y) is the camera principal point, both obtained in the initial intrinsic calibration.

$$Z_c \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{7.5}$$

By applying Eq. (7.5) to two diagonal points in opposite sides of a bounding box and knowing that Z_c is the same for both points, Eqs. (7.6) can be applied:

$$Z_c \begin{bmatrix} x_{pix_i} \\ y_{pix_i} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{c_i} \\ Y_{c_i} \\ Z_c \end{bmatrix} = \begin{cases} Z_c x_{pix_i} = f_x X_{c_i} + C_x Z_c \\ Z_c y_{pix_i} = f_y Y_{c_i} + C_y Z_c \\ Z_c = Z_c \end{cases} \tag{7.6}$$

Subtracting the equations for both points results in Eq. (7.7). $x_{pix_2} - x_{pix_1}$ corresponds to the bounding box width (W) and $y_{pix_2} - y_{pix_1}$ to the bounding box height (H). In a similar way, $X_{c_2} - X_{c_1}$ and $Y_{c_2} - Y_{c_1}$ are the bounding box rectangle's width and height in the camera's frame, which is equal to the ball diameter, Φ_{real}.

$$\begin{cases} Z_c(x_{pix_2} - x_{pix_1}) = f_x(X_{c_2} - X_{c_1}) \\ Z_c(y_{pix_2} - y_{pix_1}) = f_x(Y_{c_2} - Y_{c_1}) \end{cases} \Leftrightarrow \begin{cases} Z_c W = f_x \Phi_{real} \\ Z_c H = f_y \Phi_{real} \end{cases} \tag{7.7}$$

By replacing the known Eq. (7.8) in Eq. (7.7), Eq. (7.9) is obtained. By replacing the second equation from Eq. (7.7) in Eq. (7.9), the final expression presented in equation (7.10) is obtained.

$$r_{pix} = \frac{\Phi_{pix}}{2} = \frac{W + H}{4} \Leftrightarrow W = 2\Phi_{pix} - H \tag{7.8}$$

$$Z_c(2\phi_{pix} - H) = f_x \Phi_{real} \tag{7.9}$$

$$Z_c = \frac{f_x + f_y}{2} \frac{\Phi_{real}}{\Phi_{pix}} \tag{7.10}$$

The diameter of the ball in pixels, Φ_{pix}, is obtained by calculating the mean of the height and width of the bounding box of the detected ball (assuming that the bounding box is approximately square). Once the value of Z_c is known, it can be replaced in Eq. (7.5) and get the remainder coordinates.

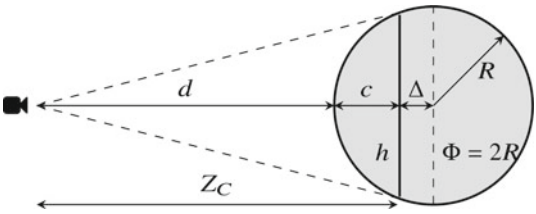

Fig. 7.15 Apparent diameter h of a ball with radius R at distance d

7.4.2.1 Apparent Diameter Correction

The apparent diameter of a spherical object is the diameter measured by an observer at a specific distance from the object. As illustrated in Fig 7.15, the fraction of diameter seen by the camera depends on its distance to the object and the object's radius. Equation (7.11) describes the relation between these two variables and the fraction.

$$F = \frac{h}{\Phi_{real}} = \frac{h}{2R} \tag{7.11}$$

To correct this phenomenon, Z_c has to be recalculated using Eq. (7.12), which compensates the fraction of the diameter that can be seen by the camera, correcting it by the actual diameter of the spherical target.

$$Z_{corr} = Z_C + \Delta = \frac{f_x + f_y}{2} \frac{\Phi_{real}}{\Phi_{pix}} + R\sqrt{1 - F^2} \tag{7.12}$$

7.4.3 Transformation Matrix Calculation

The sets of points are sorted with a one-to-one correspondence and relocated around the origin by subtracting their respective centroids from them (7.13). A Singular Value Decomposition (SVD) (7.15) method is applied to the 3×3 covariance matrix (7.14) of the coordinates of points.

$$\begin{cases} A_c = A - centroid_A \\ B_c = B - centroid_B \end{cases} \tag{7.13}$$

$$H = A_c^{\intercal} \cdot B_c \tag{7.14}$$

$$\left[U, S, V^{\intercal}\right] = \text{SVD}(H) \tag{7.15}$$

Table 7.1 Pair-wise percentage error for calibration of 3 sensors

Source	Destination		
	RGB_1 (%)	RGB_2 (%)	$Depth_2$ (%)
RGB_1	–	1.07	2.06
RGB_2	1.24	–	1.98
$Depth_2$	2.38	1.97	–

ft/ft

The rotation matrix, R, and the translation vector, t, between the point clouds are consequently calculated using the expressions in (7.16).

$$\begin{cases} R = V^{\intercal} \cdot U^{\intercal} \\ t = centroid_B{}^{\intercal} - R \cdot centroid_A{}^{\intercal} \end{cases} \tag{7.16}$$

7.4.4 Results

We performed some tests with three sensors to prove this methodology: two RGB cameras and one depth sensor. Table 7.1 shows the percentage error for that experiment. As can be concluded, the experiment presented low errors (less than 2% between two RGB cameras and less that 3% between depth and RGB).

7.5 Optimization Approach for Calibration

In order to calibrate the RGB cameras and the 3D LiDAR sensors, we propose an optimization based approach. The geometric transformations that encode the poses of the sensors, w.r.t. each other or w.r.t. a global coordinate system, are iteratively estimated in order to minimize the error produced by an objective function.

7.5.1 The Methodology

Classical approaches define the objective function $f(\cdot)$ in such a way that, for each data point associated detected in both sensors, an error e is computed, using as input a tandem of sensors s_i and s_j:

$$e = f\left(\mathbf{T}_{[s_i \to s_j]}, x_{[s_i]}, x_{[s_j]}, \{\lambda_{[s_i]}\}, \{\lambda_{[s_j]}\}\right), \tag{7.17}$$

where $\mathbf{T}_{[s_i \to s_j]}$ is the geometric transformation from the coordinate frame of sensor s_i to the coordinate frame of sensor s_j, $x_{[s]}$ are the coordinates of the detection as viewed in the data of sensor s, which may be 2D pixel coordinates or 3D point coordinates depending on the sensor's modality. Finally, $\{\lambda_{[s]}\}$ are the set of additional parameters that are required by the objective function to compute the error. There are several problems with these approaches. First, they require that the key point detected in both sensors in order to operate, i.e. $x_{[s_i]}$, $x_{[s_j]}$ must exist. Secondly, the design of the function is dependent on the combination of modalities of the sensor tandem. For example, of both s_i and s_j are RGB cameras, $f(\cdot)$ would be defined very differently when compared to a case where s_i is an RGB camera and s_j is a 3DLiDAR. The fact that the objective function depends on the pairwise combination of sensor modalities makes this approach difficult to scale to multiple modalities. Thirdly, the design of objective function based on sensor tandems is a cumbersome procedure.

Because of the shortcomings stated above, we propose to design the objective function in such a way that it considers only a single sensor and not a pair of sensors. Additionally, the pose of the calibration pattern is given to the objective function and the residual is computed from the association of the key point as viewed in the sensor (the detection) and as defined in the pattern (which we know because the pattern has know dimensions). The sensor to pattern paradigm based objective function can be defined as the sum of independent contributions for all sensors in the sensors set $\mathcal{S}$:

$$e = \sum_{s \in \mathcal{S}} f\Big(\mathbf{T}_{[s \to p]}, x_{[s]}, \{\lambda_{[s]}\}, \{\lambda_{[p]}\}\Big), \tag{7.18}$$

where $\mathbf{T}_{[s \to p]}$ is the geometric transformation from the coordinate frame of sensor s to the coordinate frame of the pattern p, and $\{\lambda_{[p]}\}$ are the parameters of the pattern from which it is possible to compute the properties of the key point. This includes the number of corners in the pattern, the 3D position of those corners w.r.t. the coordinate frame of the pattern, and the physical dimensions of the pattern, to be used by LiDAR data. The design of the objective function $f(\cdot)$ will depend on the modality of a single sensor, instead of depending on the pairwise combination of sensor modalities. This considerably facilitates the designing of the objective function as well as its scalability. To denote that the modality of the sensor must be taken into account, we propose to extend (7.18) to:

$$e = \sum_{s \in \mathcal{S}} f\Big(\mathbf{T}_{[s \to p]}, x_{[s]}, \{\lambda_{[s]}\}, \{\lambda_{[p]}\}, m(s)\Big), \tag{7.19}$$

where $m(\cdot)$ is a function that retrieves the modality of a sensor. Having established the basic structure of the objective function, which is defined in (7.19) for a single key point association between the detection in the sensor data and know key point the pattern, it is now possible to extend the formulation to accommodate several key points for each image or point cloud:

$$e = \sum_{s \in \mathcal{S}} \sum_{d \in \mathcal{D}_{[s]}} f\Big(\mathbf{T}_{[s \to p]}, x_{[s,d]}, \{\lambda_{[s]}\}, \{\lambda_{[p]}\}, m(s)\Big), \tag{7.20}$$

where $\mathcal{D}_{[s]}$ denotes the set detections produced by sensor s. Also, in order to be accurate, the approach must consider several images and point clouds for each RGB or 3D LiDAR sensor, respectively. We define a collection c as an instant in time where the system acquired data from all sensors in the system, in order to write the contribution of all collections c, in the set of collections $\mathcal{C}$:

$$e = \sum_{c \in \mathcal{C}} \sum_{s \in \mathcal{S}} \sum_{d \in \mathcal{D}_{[c,s]}} f\Big(\mathbf{T}_{[c,s \to p]}, x_{[c,s,d]}, \{\lambda_{[s]}\}, \{\lambda_{[p]}\}, m(s)\Big), \tag{7.21}$$

where the detections $\mathcal{D}_{[c,s]}$ now depend not only on the sensor s, but also on the collection c. The transformation from the sensor to the pattern coordinate frames can be represented in a more advantageous form. Since that:

$$\mathbf{T}_{[c,s \to p]} = \mathbf{T}_{[s \to w]} \cdot \mathbf{T}_{[c,w \to p]} \tag{7.22}$$

where $\mathbf{T}_{[s \to w]}$ denotes the transformation from the sensor coordinate frame to a common global reference frame, which we call *world*, and $\mathbf{T}_{[c,w \to p]}$ is the transformation from the world to the pattern coordinate frame. Note that the transformation that encodes the pose of the pattern, $\mathbf{T}_{[c,w \to p]}$ is dependent on the collection, because the pattern is repositioned before each collection, i.e., it is important that the several images and point clouds have variability, which is achieved by moving the pattern to different positions for each collection. Conversely, the transformation that encodes the pose of the sensor, $\mathbf{T}_{[s \to w]}$, does not depend on the collection. This is mandatory, since the sensor is rigidly attached to the vehicle's chassis.

The optimization framework makes use of the errors produced by the objective function expressed in (7.21). In order to estimate the poses of the several sensors, as well as the several poses of the calibration pattern for each collection, the procedure minimizes the error computed by the objective function. This can be written as:

$$\underset{\{\mathbf{T}_{[s \to w]}\}, \{\mathbf{T}_{[c,w \to p]}\}}{\arg\min} \sum_{c \in \mathcal{C}} \sum_{s \in \mathcal{S}} \sum_{d \in \mathcal{D}_{[c,s]}} f\Big(\mathbf{T}_{[s \to w]}, \mathbf{T}_{[c,w \to p]}, x_{[c,s,d]}, \{\lambda_{[s]}\}, \{\lambda_{[p]}\}, m(s)\Big), \tag{7.23}$$

where $\{\mathbf{T}_{[s \to w]}\}$ denotes the set of transformations from each sensor to the world frame, and $\{\mathbf{T}_{[c,w \to p]}\}$ is the set of poses of the pattern for each collection c.

This is the expression that represents the calibration procedure, formulated as an optimization problem. We use a *least squares* methodology, that tries to minimize the squared residuals (i.e. the predicted error) of the objective function in (7.23) with respect to its parameters. The optimization is solved with the trust region reflective algorithm [6, 10], which is a suitable method for large bounded sparse problems.

For the RGB sensor modality, i.e., when $m(s)$ is an rgb camera modality, we propose to use the reprojection error, computed by projecting the known corners of the calibration pattern $\mathbf{x}$ to the rgb camera image, and evaluating the distance between these projected coordinates with the corresponding detections in image $\mathbf{x}$:

$$f_{m(s)=\text{rgb}} = \left\| x_{[c,s,d]} - \mathbf{K}_{[s]} \cdot \mathbf{T}_{[s \to w]} \cdot \mathbf{T}_{[c,w \to p]} \cdot x_{[p,d]} \right\|, \tag{7.24}$$

where $x_{[p,d]}$ are the 3D coordinates defined in the pattern coordinate frame of the key point which corresponds to detection d, which are retrieved from the additional parameters of the pattern $\{\lambda_{[p]}\}$, $\mathbf{K}_{[s]}$ is the intrinsic matrix of the rgb camera sensor s, extracted from the additional parameters of the sensor $\{\lambda_{[s]}\}$, and $x_{[c,s,d]}$ denotes the pixel coordinates of the corner of detection d, in the image of collection c and for the rgb camera sensor s, and finally $\|\cdot\|$ denotes the norm.

The inclusion of additional modalities would require the definition of other cost functions, but their integration in the proposed framework is straightforward. Besides the RGB camera modality, we have already conducted successful experiments with 3D-LiDARS, which are not presented here because the AtlasCar2 does not have this types of sensors. Currently we are working on the inclusion of both depth and 2D-LiDAR modalities.

7.5.2 Results

In this section, results are presented that show that the proposed method is able to calibrate the three RGB cameras onboard the AtlasCar2 (Fig. 7.16).

The experiments were carried out using a training dataset containing 39 sets of images. For testing, we use a different dataset containing 21 sets of images. Each set contains one image from each camera, and the goal of the calibration is to estimate geometric transformations between the cameras. We compare our approach with Open Source Computer Vision Library (OpenCV) stereo camera calibration algorithm. However, since that method cannot tackle more than two camera at once, we carried out the calibration of the three pairwise combinations of cameras.

Notice that we used our approach to calibrate the complete system (all three cameras in a single optimization). Then, results are provided in pairs of sensors so that we may compare against other methods and use well established evaluation metrics.

Three evaluation metrics are used in the evaluation: the mean rotation error (ϵ_R, in radians), the mean translation error (ϵ_t, in meters) and the reprojection error (ϵ_{rms}, in pixels). See [33] for further details. Results in Table 7.2 demonstrate that our approach achieves similar accuracies when compared to OpenCV's method, despite the fact that it is calibrating the complete system (all the sensors simultaneously).

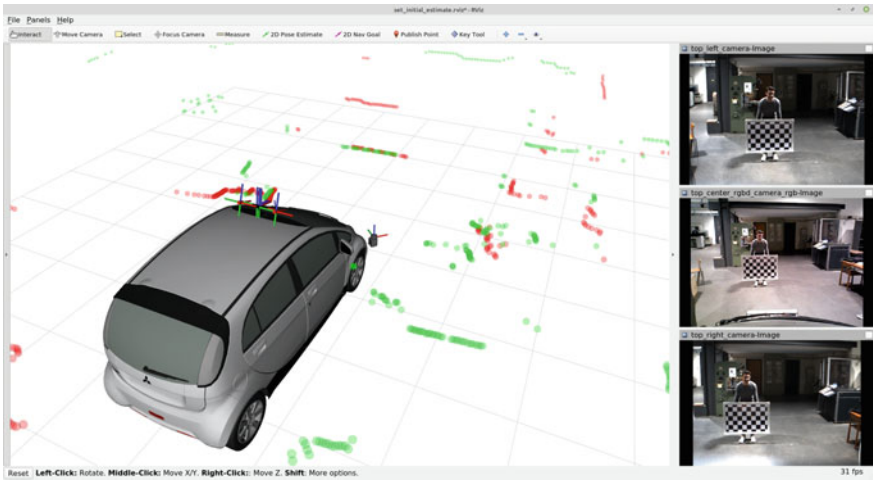

Fig. 7.16 Visualizing the data produced by the AtlasCar2 vehicle. Green and Red dots are the point clouds produced by the left and right 2D-LiDARs, respectivelly. On the right, the images produced by the three onboard cameras are displayed

Table 7.2 Performance comparison of methods. Best values highlighted in bold

Method	Sensor pair	$\epsilon_{\mathbf{R}}$ (rad)	$\epsilon_{\mathbf{t}}$ (m)	ϵ_{rms} (pix)
OpenCV	*left camera* to	0.018	0.013	**1.157**
Our approach	*right camera*	0.027	0.032	1.198
OpenCV	*right camera* to	0.244	0.081	3.336
Our approach	*center camera*	0.096	0.074	**2.375**
OpenCV	*center camera* to	0.078	0.490	**3.000**
Our approach	*left camera*	0.090	0.045	3.283

ft/ft

7.6 Comparative Analysis of the Techniques

The techniques and approaches described in the paper do not have exactly the same scope and framework, but are rather complementary and not necessarily mutually exclusive. Therefore, it is not possible to conduct a fair comparison between these approaches. The shared functionalities include: the usage of specific calibration targets (cones, spheres or checkerboards or charuco boards); the techniques to detect the targets in data (geometric matching of points, classic computer vision, learning-based detection of a target's area); or the algorithmic technique to perform the calibration (matching point sets with SVD, minimization of error function by iterative computation using optimization techniques).

For a clearer overview of the differences and similarities among the techniques, Table 7.3 summarises their relevant properties and distinguishable features.

Table 7.3 Main distinguishing properties of the approaches/techniques described in the paper

Prop./Techn.	Conic target	Spherical target	Deep learning	Optimization based
Application domain	Calibration of LiDAR 2D and 3D	Calibration LiDAR 2D, 3D, visual, thermal and depth	Detector of spheres in visual, thermal and depth* images	Universal optimization of geometric transformations
Data to process	Point clouds of measurements	Sets of points which are the centers of multiple ball positions	Gray level images with balls on them	Annotated information of specific targets in captured images or segmented from point clouds
Outputs	Geometric transformation between a pair of sensors	Geometric transformations between pairs of sensors	Set of points of ball centers in the images	Optimal geometric transformations for a group of coordinate frames
Algorithmic basis	Iterative Closest Point and ellipse fitting	Matching of 3D points sets by Least Square Fit (SVD)	R-CNN detection after a specific training set	Definition of error functions and their minimization by optimization
Level of human intervention	Indicate 2 points in the 3D point cloud and 4 points in the 2D point cloud	Move the ball around to create sets of centers	Virtually none. Relies on acquired images	Move the chess board around to create collections. Give first guesses or auxiliary points
Year and reference of publication	2012 [2]	2015, 2016 [34, 35]	2020 [39] *	2019–2021 [1, 28, 29, 33]

ft* Calibration with depth images is in a pending publication for 2022/ft

An analysis of Table 7.3 shows that we have worked on a large range of calibration problems. Overall, the RGB, 2D-LiDAR, 3D-LiDAR, Depth and Thermal sensor modalities have been addressed, which is a considerable portion of the sensor modalities typically used in autonomous vehicles. Concerning the methods used for carrying out the estimation of the transformation between sensors, we have also explored several possibilities, from closed form solutions such as SVD or to iterative ones such as ICP. Most of the approaches require human intervention to produce or review annotations in the data, but we have also worked on completely automatic calibration methodologies. Finally, the table also highlights that the authors have been actively working on the extrinsic calibration of multimodality sensors onboard autonomous vehicles since 2012.

7.7 Conclusion and Future Perspectives

This paper describes several techniques used in the ATLASCAR project for the calibration of multi modal sensors present on the car. Although the techniques appeared in sequence in the history of the project, none of them is expected to completely replace the others, but rather offer the possibility to complement each other with their own advantages and applications. For example, the optimization technique described in Sect. 7.5 faces some challenges when performing the detection and labeling of target patterns present in data (visual, point clouds, depth images) but offers an excellent context for calibration by optimization of a virtually unlimited set of sensors. Alternative targets, such as spheres, possibly detected by other approaches such as semantic classifiers may provide a very advantageous combination for future developments.

Besides these potential developments on the integration of several techniques, these approaches may also continue to evolve individually. Examples are generalizing the Deep Learning based classification to LiDAR data, which is now trending in the state of the art, allowing alternative ways of faster detection of the calibration targets in the raw data for further calibration procedures.

Apart from all the integrated and individual developments, it is also worth mention that multi modal calibration is not a problem exclusive to autonomous vehicles. There are several other scenarios where the calibration of intelligent systems with multiple sensors of various modalities is also of paramount importance. One such examples is the collaborative industrial manufacturing cells that are expected to monitor the workspace with high detail and robustness, which will also require large sets of sensors and modalities.

Acknowledgements The authors acknowledge the support of CYTED Network: "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559). This work was also partially funded by National Funds through the FCT—Foundation for Science and Technology, in the context of the project UIDB/00127/2020. Daniela Rato acknowledges the support of the Foundation of Science and Technology (FCT) in the context of the scholarship 2021.04792.BD.

References

1. Aguiar, A., Oliveira, M., Pedrosa, E., Santos, F.: A camera to lidar calibration approach through the optimization of atomic transformations. Expert Syst. Appl. 114894 (2021)
2. Almeida, M., Dias, P., Oliveira, M., Santos, V.: 3D-2D laser range finder calibration using a conic based geometry shape. In: Image Analysis and Recognition—9th International Conference, ICIAR 2012, Aveiro, Portugal, June 25–27, 2012. Proceedings, Part I. Lecture Notes in Computer Science, vol. 7324, pp. 312–319. Springer (2012)
3. Bai, Z., Jiang, G., Xu, A.: Lidar-camera calibration using line correspondences. Sensors **20**(21) (2020)
4. Basso, F., Menegatti, E., Pretto, A.: Robust intrinsic and extrinsic calibration of RGB-D cameras. IEEE Trans. Rob. **34**(5), 1315–1332 (2018)
5. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)

6. Branch, M.A., Coleman, T.F., Li, Y.: A subspace, interior, and conjugate gradient method for Large-Scale Bound-Constrained minimization problems. SIAM J. Sci. Comput. **21**(1), 1–23 (1999)
7. Chen, G., Cui, G., Jin, Z., Wu, F., Chen, X.: Accurate intrinsic and extrinsic calibration of RGB-D cameras with GP-based depth correction. IEEE Sens. J. **19**(7), 2685–2694 (2019)
8. Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image Vis. Comput. **10**(3), 145–155 (1992). Range Image Understanding
9. Chen, Z., Yang, X., Zhang, C., Jiang, S.: Extrinsic calibration of a laser range finder and a camera based on the automatic detection of line feature. In: 2016 9th Int. Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 448–453 (2016)
10. Coleman, Thomas F., Li, Yuying: An interior trust region approach for nonlinear minimization subject to bounds. SIAM J. Optim. **6**(2), 418–445 (1996)
11. Gao, D., Duan, J., Yang, X., Zheng, B.: A method of spatial calibration for camera and radar. In: 2010 8th World Congress on Intelligent Control and Automation, pp. 6211–6215 (2010)
12. Dias, P., Campos, G., Santos, V., Casaleiro, R., Seco, R., Santos, B.S.: 3D reconstruction and spatial auralization of the painted dolmen of Antelas. In: Corner, B.D., Mochimaru, M., Sitnik, R. (eds.) Three-Dimensional Image Capture and Applications 2008, vol. 6805, pp. 272–281. International Society for Optics and Photonics, SPIE (2008)
13. Dinh, V.Q., Nguyen, T.P., Jeon, J.W.: Rectification using different types of cameras attached to a vehicle. IEEE Trans. Image Process. **28**(2), 815–826 (2019)
14. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartogr.: Int. J. Geogr. Inf. Geovisualization **10**(2), 112–122 (1973)
15. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
16. Guindel, C., Beltran, J., Martin, D., Garcia, F.: Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE (2017)
17. Häselich, M., Bing, R., Paulus, D.: Calibration of multiple cameras to a 3D laser range finder. In 2012 IEEE International Conference on Emerging Signal Processing Applications, pp. 25–28 (2012)
18. Jiang, P., Osteen, P., Saripalli, S.: Semcal: Semantic lidar-camera calibration using neural mutual information estimator. In: 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 1–7 (2021)
19. Khan, A., Aragon-Camarasa, G., Sun, L., Siebert, J.P.: On the calibration of active binocular and RGBD vision systems for dual-arm robots. In: 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1960–1965 (2016)
20. Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D., Murphy, K.: Openimages: a public dataset for large-scale multi-label and multi-class image classification (2017). https://github.com/openimages
21. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V.: The open images dataset v4: unified image classification, object detection, and visual relationship detection at scale. IJCV (2020)
22. Kwon, Y.C., Jang, J.W., Choi, O.: Automatic sphere detection for extrinsic calibration of multiple RGBD cameras. In: 2018 18th International Conference on Control, Automation and Systems (ICCAS), pp. 1451–1454 (2018)
23. Lee, D., Lee, J., Park, S.: Calibration of VLP-16 lidar and multi-view cameras using a ball for 360 degree 3D color map acquisition. In: 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 64–69 (2017)
24. Ling, Y., Shen, S.: High-precision online markerless stereo extrinsic calibration. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1771–1778 (2016)

25. Álvarez, H., Alonso, M., Sánchez, J.R., Izaguirre, A.: A multi camera and multi laser calibration method for 3D reconstruction of revolution parts. Sensors **21**(3) (2021)
26. Matos, M., Santos, V., Dias, P.: A fast low-cost 3D scanner for navigation and modelling. In: ROBOTICA2004—Proceedings of the Scientific Meeting, pp. 69–74. Porto, Portugal (2004)
27. Mueller, G.R., Wuensche, H.: Continuous stereo camera calibration in urban scenarios. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6 (2017)
28. Oliveira, M., Castro, A., Madeira, T., Dias, P., Santos, V.: A general approach to the extrinsic calibration of intelligent vehicles using ROS. In: Robot 2019: Fourth Iberian Robotics Conference, pp. 203–215. Springer International Publishing, Cham (2020)
29. Oliveira, M., Castro, A., Madeira, T., Pedrosa, E., Dias, P., Santos, V.: A ROS framework for the extrinsic calibration of intelligent vehicles: a multi-sensor, multi-modal approach. Robot. Auton. Syst. 103558 (2020)
30. Oliveira, M., Santos, V.: Autonomous navigation for robots with road-like challenges: perception approaches used in the ATLAS project. In: 11th International Conference on Mobile Robots and Competitions (ROBOTICA2011), pp. 58–63. Lisboa, Portugal (2011)
31. Pascoal, R., Santos, V.: Compensation of azimuthal distortions on a free spinning 2D laser range finder for 3D data set generation. In: Proceedings of the 10th International Conference on Mobile Robots and Competitions (ROBOTICA2010), pp. 41–46. Leiria, Portugal (2010)
32. Pascoal, Ricardo, Santos, Vitor, Premebida, Cristiano, Nunes, Urbano: Simultaneous segmentation and superquadrics fitting in laser-range data. IEEE Trans. Veh. Technol. **64**(2), 441–452 (2015)
33. Pedrosa, Eurico, Oliveira, Miguel, Lau, Nuno, Santos, Vítor.: A general approach to hand-eye calibration through the optimization of atomic transformations. IEEE Trans. Rob. **37**(5), 1619–1633 (2021)
34. Pereira, M., Santos, V., Dias, P.: Automatic calibration of multiple lidar sensors using a moving sphere as target. In: Robot 2015: Second Iberian Robotics Conference, pp. 477–489. Springer International Publishing (2016)
35. Pereira, Marcelo, Silva, David, Santos, Vitor, Dias, Paulo: Self calibration of multiple lidars and cameras on autonomous vehicles. Robot. Auton. Syst. **83**, 326–337 (2016)
36. Qiao, Y., Tang, B., Wang, Y., Peng, L.: A new approach to self-calibration of hand-eye vision systems. In: 2013 International Conference on Computational Problem-Solving (ICCP), pp. 253–256 (2013)
37. Zhang, Q., Pless, R.: Extrinsic calibration of a camera and laser range finder (improves camera calibration). In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), vol. 3, pp. 2301–2306 (2004)
38. Ramer, Urs: An iterative procedure for the polygonal approximation of plane curves. Comput. Graphics Image Process. **1**(3), 244–256 (1972)
39. Rato, D., Santos, V.: Automatic registration of IR and RGB cameras using a target detected with deep learning. In: 2020 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2020, Ponta Delgada, Portugal, April 15–17, 2020, pp. 287–293. IEEE (2020)
40. Ren, S., He, K., Girshick, R., Sun, J.: Towards real-time object detection with region proposal networks, Faster R-CNN (2015)
41. Su, R., Zhong, J., Li, Q., Qi, S., Zhang, H., Wang, T.: An automatic calibration system for binocular stereo imaging. In: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp. 896–900 (2016)
42. Ruan, M., Huber, D.: Calibration of 3D sensors using a spherical target. In: 2014 2nd International Conference on 3D Vision, vol. 1, pp. 187–193 (2014)
43. Santos, V.: ATLASCAR: a sample of the quests and concerns for autonomous cars. In: Informatics in Control, Automation and Robotics, pp. 355–375. Springer International Publishing (2020)
44. Santos, V., Almeida, J., Avila, E., Gameiro, D., Oliveira, M., Pascoal, R., Sabino, R., Stein, P.: ATLASCAR—technologies for a computer assisted driving system on board a common

automobile. In: 13th International IEEE Conference on Intelligent Transportation Systems, pp. 1421–1427 (2010)
45. Santos, V., Rato, D., Dias, P., Oliveira, M.: Multi-sensor extrinsic calibration using an extended set of pairwise geometric transformations. Sensors **20**(23) (2020)
46. Van Crombrugge, I., Penne, R., Vanlanduit, S.: Extrinsic camera calibration with line-laser projection. Sensors **21**(4) (2021)
47. Vasconcelos, F., Barreto, J.P., Nunes, U.: A minimal solution for the extrinsic calibration of a camera and a laser-rangefinder. IEEE Trans. Pattern Anal. Mach. Intell. **34**(11), 2097–2107 (2012)
48. Vidas, S., Lakemond, R., Denman, S., Fookes, C., Sridharan, S., Wark, T.: A mask-based approach for the geometric calibration of thermal-infrared cameras. IEEE Trans. Instrum. Meas. **61**(6), 1625–1635 (2012)
49. Wu, L., Zhu, B.: Binocular stereovision camera calibration. In: 2015 IEEE International Conference on Mechatronics and Automation (ICMA), pp. 2638–2642 (2015)
50. Kirillov, A., Wu, Y., He, K., Girshick, R.: Detectron2 (2019). https://github.com/facebookresearch/detectron2
51. Xavier, J., Pacheco, M., Castro, D., Ruano, A., Nunes, U.: Fast line, arc/circle and leg detection from laser scan data in a player driver. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 3930–3935 (2005)
52. Zhang, C., Zhang, Z.: Calibration between depth and color sensors for commodity depth cameras. In: 2011 IEEE International Conference on Multimedia and Expo, pp. 1–6 (2011)
53. Zhao, G., Hu, J., You, S., Kuo, C.C.J.: CalibDNN: multimodal sensor calibration for perception using deep neural networks. In: Kadar, I., Blasch, E.P., Grewe, L.L. (eds.) Signal Processing. Sensor/Information Fusion, and Target Recognition XXX, vol. 11756, pp. 324–335. International Society for Optics and Photonics, SPIE (2021)

Chapter 8
Early Computer-Aided Diagnose in Medical Environments: A Deep Learning Based Lightweight Solution

Miguel Nehmad Alche, Daniel Acevedo, and Marta Mejail

Abstract The use of artificial intelligence in healthcare systems has helped doctors to automate many tasks and has avoided unnecessary patient hospitalizations. Mobile devices have grown in computing capacity and enhanced image acquisition capabilities, which enable the implementation of more powerful outpatient services. In this chapter we propose a lightweight solution to the diagnose of skin lesions. Specifically, we focus on the melanoma classification whose early diagnosis is crucial to increase the chances of its cure. Computer vision algorithms can be used to analyze dermoscopic images of skin lesions and decide if these correspond to benign or malignant tumors. We propose a deep learning solution by means of the adaptation of the attention residual learning designed for ResNets to the EfficientNet networks which are suitable for mobile devices. A comparison is made of this mechanism with other attention mechanisms that these networks already have incorporated. We maintain the efficiency of these networks since only one extra parameter per stage needs to be trained. We also test several pre-processing methods that perform color corrections of skin images and sharpens its details improving the final performance. The proposed methodology can be extended to the early detection and recognition of other forms of skin lesions.

M. Nehmad Alche
Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales, Departamento de Computación, Buenos Aires, Argentina
e-mail: mikealche@gmail.com

D. Acevedo (✉) · M. Mejail
CONICET-UBA. Instituto de Investigación en Cs. de la Computación (ICC), Buenos Aires, Argentina
e-mail: dacevedo@dc.uba.ar

M. Mejail
e-mail: marta@dc.uba.ar

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_8

8.1 Introduction

In order to promote healthcare systematization, researchers and practitioners are challenged to find efficient technological solutions to meet the needs of citizens. In particular, remote care technology might promote healthcare personalization and the self-management of patients with specific chronic conditions. The recent COVID-19 pandemic has accelerated advances in telemedicine and particularly teledermatology, which has allowed dermatologists to provide dermatological care remotely [14]. For this, faster solutions are desirable for the early diagnosis of skin diseases in the transition to health systems in smart cities.

Melanoma is one of the forms of skin cancer with the highest mortality rates and have been increasing for the past decades, as can be seen in countries such as UK, where the rate of melanoma has increased 119% since the 1990s, or USA (from 27,600 cases in 1990 to 91,270 in 2018) [34]. That is why it is vitally important that proper treatment is carried out as soon as possible. Fortunately, dermatologist are able to utilize non-invasive techniques to obtain color images of skin lesions. These images can either be macroscopic or dermoscopic depending on the acquisition setup. Macroscopic images are acquired using standard cameras or mobile phones, while dermoscopy images are obtained with specific magnification devices and an oil interface (immersion contact dermoscopy) or using cross-polarizing light filters (non-contact dermoscopy) [16]. Nowadays, most research use of dermoscopy images since large datasets are available and provide for a better visualization of color and pattern properties that improve diagnostic accuracy.

Some computer vision techniques have emerged in order to detect this type of skin lesions in an early stage. The purpose of these algorithms is to be able to bring as many people as possible a reliable way to carry out periodic checks: either by distributing them in mobile applications for end users, as well as the creation of specific tools for health professionals specialized in skin treatment.

The development of computer-aided diagnosis (CAD) systems that can be used by dermatologists generally follow a pipeline. Each stage of this pipeline encompasses research topics of great importance: image pre-proccesing, lesion segmentation, feature extraction, feature selection (optional), and classification. These stages may be merged with each other and some systems may not have all of them present.

Image pre-processing is a required step for images that come from different sources, acquired under irregular conditions, and need for certain enhancement. Contrast enhancement, for example, is a popular technique to improve the performance of lesion segmentation methods, increasing the contrast between the lesion border and its surroundings [31]. Another task associated with pre-processing is artifact removal. They may be present in different forms and are related to image acquisition (air bubbles, ruler and ink markings, etc.) or cutaneous properties (skin lines, blood vessels and hair). One of the most frequently addressed problems is hair removal by means of the Dullrazor method [22], which is based on morphological operations, bilinear interpolation and adaptive median filtering (see [1] for a comparative study). The variety of types of digital cameras and illumination conditions introduces signif-

icant variability in the color properties of the images. For that, color normalization and calibration are another important operation that must be performed during image pre-processing so as to make diagnoses more reliable. Quintana et al. [26] addresses this problem taking into account the spectral distribution of the dermoscope lighting system, while Iyatomi et al. [21] develop color calibration filters based on the HSV color system. Color constancy algorithms are investigated by Barata et al. [7] for color normalization.

Although this work does not include image segmentation for melanoma diagnosis, lesion segmentation has an important role in the process of automating skin lesion diagnosis and analysis. As in any image segmentation process, a region of interest is obtained by separating diseased area from the healthy region. These techniques include handcrafted features such as threshold-based [13], edge and region-based methods [44] and the most widely used methods based on deep learning architectures such as U-Net [39], Fully Convolutional [9], Deep Fully Convolutional Residual Network [41] and Convolutional De-Convolutional Neural Networks [42]. For a thorough review of recent segmentation methodologies see [36].

Early approaches for the diagnosis of dermoscopy images were mostly based on scoring rules which rely on dermoscopic criteria. Some of these methods aim at recognizing only criteria that is associated with melanoma such as the 7-point checklist [6] and the Menzies method [23]. Other methods focus on a broader analysis of the lesion taking into account the degree of asymmetry, border sharpness, lesion architecture and color distribution (ABCD [24, 33] and CASH [19] rules). All this knowledge that comes from the methodologies designed by dermatologists, led to the formulation of hand-crafted features. For instance, the ABCD rule of dermoscopy assigns the highest weight to the asymmetry criterion, making it a relevant cue for melanoma diagnosis. Experts consider that asymmetry of lesions should be evaluated with respect to its shapes, colors and patterns. Several papers have designed features based on this criteria [10, 25, 28]. The D in the ABCD rule stands for the identification of dermoscopic structures that exhibit specific visual patterns (e.g., pigment network is a network like structure with dark lines over a lighter background, and dots/globules are round or oval structures of dark coloration and variable size). This has motivated the design of descriptors that characterize the texture of a lesion with techniques based Wavelet and Fourier transforms [29] among others. A complete analysis of feature extraction can be found on the survey by Barata et al. [8].

The last stage of a CAD system deals with its outcome: a diagnose. Most of the CAD systems focus on the distinction between melanoma and benign or atypical nevi, due to high degree of malignancy associated with the former type of cancer. Several classifiers have been used for the diagnosis task: instance-based, decision trees, Bayesian classifiers, artificial neural networks (ANNs), support vector machines (SVMs), and ensemble methods.

The increase in computing resources has lead to the rapid development of deep learning models. Most of the image-based problems are solved with convolutional networks, and achieve state-of-the-arts performances in processing and classifying images. Also, it has benefited from international competitions [11] which make dermoscopic image datasets with associated classification labels available to researchers.

In the following we enumerate several works based on CNN architectures aiming at classifying skin lesion images. GoogleNet and AlexNet architectures were employed for classification of skin lesions by Alqudah et al. [5]. ISIC dataset was used to classify images into three classes of benign, melanoma, and seborrheic keratosis and classification was carried out on both segmented and non-segmented images. Ratul and Mozaffari [27] developed an automated CAD system for early diagnosis of malignant skin lesions by means of dilated convolution with four different architectures such as VGG16, VGG19, MobileNet, and InceptionV3. Gessert et al. [15] also performed skin lesion classification with an ensemble of deep learning models including EfficientNets, SENet, and ResNeXt WSL by utilizing a search strategy.

A deep learning framework that integrates deep features information to generate most discriminant feature vector was developed by Akram et al. [3]; they employed Entropy-Controlled Neighborhood Component Analysis (ECNCA) for discriminant feature selection and dimensionality reduction, and selected layers of deep architectures (DenseNet 201, Inception-ResNet-v2, and Inception-V3) were employed as classifiers in the system. El-Khatib et al. [12] also developed a system that is able to diagnose skin lesions using deep learning-based methods, aiming to differentiate melanoma from benign nevus based on both deep neural networks and feature-based methods. They employed architectures such as GoogleNet, ResNet-101, and NasNet-Large already pretrained on large ImageNet and Places365 datasets. Almaraz-Damian et al. [4] utilized a fusion of ABCD rule into deep learning architecture for efficient preprocessing, feature extraction, feature fusion, and classification for detection of melanoma skin cancer. The ABCD rule was employed for handcrafted feature extraction and deep learning features were extracted by the CNN architecture. The proposed framework was evaluated on ISIC 2018 and the performance was compared with other techniques such as Linear Regression (LR), SVMs, and Relevant Vector Machines (RVMs). For further references of state-of-the-art deep learning techniques see the survey by Adegun and Viriri [2].

Regarding image classification algorithms, after the success of ResNet networks [18], several improvements and variations have been tested. Among them, the EfficientNet [38] points to a good trade-off between precision and computational cost.

The attention mechanism has been applied so as to strengthen the discriminative ability of a convolutional network. Zhang et al. [43] proposed the addition of a small number of parameters to the ResNet that allows a simple but powerful attention mechanism with low computational cost. Another form of attention in the channel dimension has also been introduced by Hu et al. [20] to improve performance of convolutional networks.

Techniques that combine several models (ensemble models) were studied by Xie et al. [40], however this type of models usually require high computational costs that are to be avoided if a lightweight implementation is desired such as mobile apps.

In this paper we improve state of the art results on melanoma image classification. Inspired by the work of Zhang et al. [43], where an attention residual learning mechanism is added to the Resnet, we incorporate a similar mechanism into the EfficientNet. In line with the economy of resources posed by the EfficientNet, this mechanism uses very few parameters. Also, it has been shown that skin lesion images

benefit from color preprocessing [7]. For that, we apply preprocessing algorithms that normalize the colors of the images by applying color constancy algorithms, as well as removing variations in hue from images by applying Ben Graham preprocessing [17].

Our results show that significant improvements are achieved when both the Ben Graham preprocessing method as well as the addition of the attention residual learning mechanism to the EfficientNet networks are used.

The paper is organized as follows. In Sect. 8.2 the proposed methodology is presented. Next on Sect. 8.3 we present and analyze the results that verify our hypothesis. Concluding remarks are presented in Sect. 8.4.

8.2 Methodology

The improvements that we obtain on skin lesion classification are mainly achieved by the introduction of the attention residual mechanism on the EfficientNet along with image preprocessing. For that, in this section we describe the base methods of our proposal. First, preprocessing techniques are introduced, and then describe the attention residual mechanism on ResNets followed by the EfficientNets and how we insert this attention mechanism on them.

8.2.1 Preprocessing

We have previously reviewed in our introduction the importance of pre-processing which are the steps taken to correct, enhance and format images before they are used by model training and inference. This step should be differentiated from image augmentation which is a process applied to training data only so as to create different versions of images in order to expose the model to a wider array of training examples (e.g., randomly altering rotation, brightness, or scale). Still, image augmentation manipulations are sometimes forms of image preprocessing. In this work we utilize two main techniques that deal with color correction and with an enhancement that reveals details of the skin lesion image.

8.2.1.1 Color Correction and Color Constancy

Color preprocessing methods aim to achieve greater uniformity in the images without discarding valuable and particular information about each one which facilitates the task of classification for neural networks. Color constancy methods [7] transform the colors of an image that have been captured under an unknown light source, so that the image appears to have been obtained under a canonical light source. The implementation of this transformation consists of two steps.

First, it is necessary to estimate the light source under which the image was taken, called estimated illuminant and represented by a vector $\mathbf{e} = [e_R\ e_G\ e_B]^T$. Two algorithms are implemented:

Max-RGB: This algorithm forms the estimated illuminant by selecting for each channel of an image the maximum value that appears in it by the equation $\max_{\mathbf{x}} I_c(\mathbf{x}) = ke_c$.

Shades of Gray: This method computes the estimated illuminant from the equation $\left(\frac{\int (I_c(\mathbf{x}))^p d\mathbf{x}}{\int d\mathbf{x}}\right)^{1/p} = ke_c$, where I_c represents the channel c of the image; $\mathbf{x} = (x, y)$ is the spatial position of the pixel and k is a normalization constant.

As a second step, the image colors have to be recalibrated by means of the equation $I_c^t = I_c * 1/(\sqrt{3}e_c)$ for each channel c, once the estimated illuminant vector $\mathbf{e} = [e_R\ e_G\ e_B]^t$ is computed.

In this work, the color constancy transformation is performed prior to any other data augmentation transformation.

8.2.1.2 Ben Graham Preprocessing

This preprocessing comes from the winner of the diabetic retinopathy competition on the Kaggle platform [17]. It resembles the unsharp masking method for image sharpening, since it is based on obtaining the unsharp mask from an image. It can be summarized on Eq. (8.1) where the input image I_{in} is subtracted from its convolved version with the Gaussian kernel G (whose variance is determined automatically from the image size); then it is scaled and shifted.

$$I_{out} = 4(I_{in} - G * I_{in}) + 128 \tag{8.1}$$

8.2.2 Attention Residual Learning

In the paper by Zhang et al. [43] authors are able to simulate the effect of an attention layer without the extra computational cost that comes from the addition of significant number of new parameters. This technique is based on adding a second skip connection to the ResNet blocks, where the original input is multiplied point-wise by a Softmax version of the block's output.

The final output of the block is then formed by the typical output of a ResNet block (with its regular skip-connection) added to this new Attention Residual Learning (ARL) aggregate which is controlled by a scalar α that regulates its effect. This can be seen formally in Eq. (8.2) where x is the input, F is the convolutional block, α is the scalar that regulates the intensity of the effect, '$\cdot$' is the point wise multiplication and finally $\mathfrak{R}$ is the spatially applied softmax function.

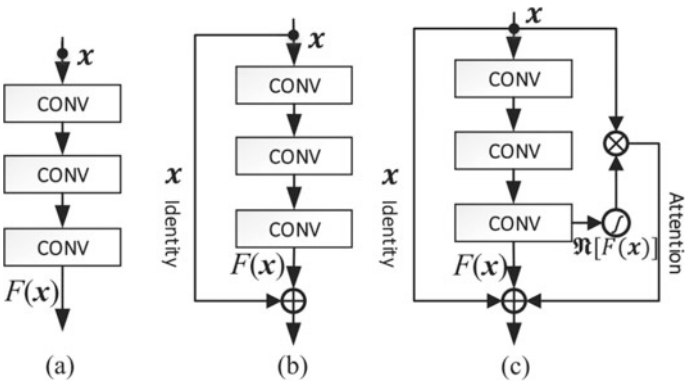

Fig. 8.1 Comparison among different blocks used in deep neural networks: **a** classical convolutional block; **b** a block with the skip-connection as used by ResNets; **c** the block with the skip-connection and the added attention residual learning [43]

$$y = x + F(x) + \alpha \cdot \Re[F(x)] \cdot x \tag{8.2}$$

The softmax function $\Re^S$ is defined in Eq. (8.3) where O is the input, $m^c_{i,j}$ refers to the value at position (i, j) from channel c of output m.

$$\Re^S(O) = \left\{ m \mid m^c_{i,j} = \frac{e^{o^c_{i,j}}}{\sum_{i',j'} e^{o^c_{i',j'}}} \right\} \tag{8.3}$$

The classical residual block and the ARL is shown in Fig. 8.1. It is worth mentioning that only one parameter α is added to the training process for each ResNet block.

8.2.3 *EfficientNets*

EfficientNet architectures give a defined way of how to scale the architecture models when more computing power is available. These networks are comparable to ResNets and outperform them in several tasks [38].

The decision of either choosing to add channels to the layers (width scaling), or to add layers to the model (depth scaling), or choosing to add resolution to the layers (resolution scaling) is specified by a constraint that involves taking full advantage of the computational power available. This restriction is defined by the following inequalities:

$$\begin{aligned} \text{depth: } d &= \alpha^{\phi} \\ \text{width: } w &= \beta^{\phi} \\ \text{resolution: } r &= \gamma^{\phi} \\ \text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\ \alpha \geq 1, \beta &\geq 1, \gamma \geq 1 \end{aligned}$$

where ϕ is a user-adjustable parameter that regulates available resources and α, β, γ are parameters determined by a small grid search. According to the convention used by the authors, a value of $d = 1$ implies 18 convolutional layers and a value of $r = 1$ implies a size of 224×224 for the images. This restriction leads to a family of networks called EfficientNet-b1 to EfficientNet-b7, depending on their capacity and complexity. In this work we use the EfficientNet-b0 model. The choice of scaling that defines this model is done using Neural Architecture Search similar to [37] optimizing an objective function that is defined by $ACC(m) \cdot (FLOPS(m)/T)^w$ where $ACC(m)$ and $FLOPS(m)$ refer to the accuracy and the number of FLOPS of the m model respectively, T is the target of FLOPS to achieve and w is a parameter that controls the importance of the FLOPS in the optimization.

The base element of the EfficientNet-b0 is the MBConv block [30]. The first appearance of these blocks in neural networks occurs with the MobileNet architecture [30]. By means of Depth-wise Separable Convolutions they reduce the amount of FLOPS required, without significantly impairing model accuracy. It is based on splitting the standard convolution layer in two: the first layer, called a *depth-wise convolution*, performs a lightweight filtering by applying a single convolutional filter per input channel; the second layer is a 1×1 convolution, called a *point-wise convolution*, which is responsible for building new features through computing linear combinations of the input channels. Another significant change that MobileNet brings is the use of inverted residuals. In the bottleneck blocks of the ResNet, a channel reduction is made prior to applying the convolution operation. However, in MobileNet this dynamic is reversed: the channels are expanded prior to perform the convolution operation and then reduced again, instead of reducing the channels and then expanding them. See [30] for details.

The EfficientNet-b0 baseline network is mainly built from inverted bottlenecks MBConv. As in ResNets, these blocks have skip connections. In line with the computational savings that these networks put forward, we propose the addition of a light attention mechanism as in attention residual learning, i.e., we add the last term of Eq. (8.2) to each MBConv. It should be noticed that, as with ResNets, each MBconv block now only has a single extra parameter α to be trained.

The ARL attention mechanism implemented on EfficientNet-b0 can be introduced where skip-connections exist. The skip-connection only appears in the MBConv blocks which make up 16 of the 18 layers. However, the skip-connection cannot be applied in the 16 existing layers since 7 of them double the number of channels of the output with respect to the input (making it impossible to match input and output dimensions). That is why we will apply the ARL mechanism in only $16 - 7 = 9$

Table 8.1 EfficientNet-B0 Arquitecture. Each row describes a layer of type $\hat{\mathcal{F}}_i$, with $\langle \hat{H}_i, \hat{W}_i \rangle$ input resolution, and $\hat{C}_i$ output channels. The last column indicates with symbol '$\star$' the layers where the insertion of the ARL mechanism is possible

i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	ARL
1	Conv3 × 3	224 × 224	32	
2	MBConv1, k3 × 3	112 × 112	16	
3	MBConv6, k3 × 3	112 × 112	24	
	MBConv6, k3 × 3	112 × 112	24	$\star$
4	MBConv6, k5 × 5	56 × 56	40	
	MBConv6, k5 × 5	56 × 56	40	$\star$
5	MBConv6, k3 × 3	28 × 28	80	
	MBConv6, k3 × 3	28 × 28	80	$\star$
	MBConv6, k3 × 3	28 × 28	80	$\star$
6	MBConv6, k5 × 5	14 × 14	112	
	MBConv6, k5 × 5	14 × 14	112	$\star$
	MBConv6, k5 × 5	14 × 14	112	$\star$
7	MBConv6, k5 × 5	14 × 14	192	
	MBConv6, k5 × 5	14 × 14	192	$\star$
	MBConv6, k5 × 5	14 × 14	192	$\star$
	MBConv6, k5 × 5	14 × 14	192	$\star$
8	MBConv6, k3 × 3	7 × 7	320	
9	Conv1 × 1 & Pooling & FC	7 × 7	1280	

layers and we will then have 9 new α parameters to train. This can be clearly seen in Table 8.1 where the ARL column marks with the symbol '$\star$' the specific layers that have the ARL mechanism incorporated.

The EfficientNet-b0 baseline network also has an attention mechanism added to it called Squeeze & Excitation [20], which can be combined with our proposed mechanism. In the experiments we test the inclusion and removal of both mechanisms.

8.3 Experiments and Results

We work with ISIC's 2017 dataset [11] which consists of 2000 dermoscopic images. The images have been resized to a size of 224 × 224 pixels. The following data augmentation techniques are applied as a basis: rotation of 180 degrees for both sides, zoom of up to 30% in different regions of the image and changes in the luminosity. A batch size of 16 is used and Oversampling is used as a mechanism to solve the class imbalance.

The trainings are carried out in 2 stages. In the first stage, only the head of the model (the fully connected layers) is trained. The head has an output size of length 2 corresponding to the 2 classes that we are trying to predict: melanoma or others. This training is carried out for 4 epochs, applying a learning rate with a One Cycle [35] policy of using a maximum value of $3 \cdot 10^{-3}$. In the second stage, the entire model is trained for 20 epochs, applying the One Cycle policy with a maximum value of $3 \cdot 10^{-4}$. This process is repeated 10 times, each time with a different seed, in order to have robust data.

8.3.1 Impact of the Preprocessing Methods in the Classification

In this experiment we hypothesize that the differences in luminosity on which the pictures were taken introduces noise into the classification process which hardens the task for the neural network. Therefore, we try to verify if the application of color constancy algorithms -Max RGB and Shades of Gray- in the process of homogenizing the dataset images, also facilitates the task of classification.

In turn, in addition to the color constancy algorithms, a third method is tried: Ben Graham preprocessing. This preprocessing is rather something close to a high pass filter. However, in the process of suppressing the low frequencies, it is believed that the differences introduced by the different illuminations when the images have been taken will also be attenuated.

We train 4 ResNet-50 networks which were pretrained in ImageNet. The first will serve as a baseline for comparison, while the second one will be trained on the dataset corrected with Max-RGB, the third one on the dataset corrected with Shades of Gray and the fourth on the dataset preprocessed with the Ben Graham method.

8.3.1.1 Results and Interpretation

Both Tables 8.2 and 8.3 are formed by taking the maximum accuracy and AUROC respectively reached by each run and then calculating the mean, variance and quartiles of them. As can be seen, the training using Ben Graham's method surpasses all the other methods, by giving a maximum average accuracy between all runs of 0.887, while the training without any processing obtains 0.874.

Likewise, the Max RGB method also seems to have a positive impact on the results with average maximum accuracy between runs of 0.877, albeit to a lesser extent than the Ben Graham method. However, the Shades of Gray method seems in this case to harm performance.

It is understood that this may be due to the way each preprocessing method corrects images. In Fig. 8.2 it can be seen how the images processed by Shades of Gray are inclined -in some cases- towards a blue tint, while those processed by

Table 8.2 Mean, standard deviation and quartiles accuracies for 10 runs on the dataset without any color correction, and with the 3 preprocessing described algorithms

Dataset	Mean	Std	25%	50%	75%	Max
Baseline	0.874000	**0.006630**	0.868333	0.873333	0.878333	0.886667
Max RGB	0.876667	0.009558	0.868333	0.880000	0.885000	0.886667
Shades of Gray	0.858667	0.011244	0.853333	0.860000	0.865000	0.880000
Ben Graham	**0.887333**	0.012746	**0.881667**	**0.886667**	**0.893333**	**0.906667**

Table 8.3 Mean, standard deviation and quartiles AUROC values for 10 runs on the dataset without any color correction, and with the 3 preprocessing algorithms

Dataset	Mean	Std	25%	50%	75%	Max
Baseline	0.885389	0.014831	0.880555	0.889861	0.894861	0.905000
Max RGB	0.891306	0.011410	0.882153	0.891528	0.900695	0.908333
Shades of Gray	0.861222	0.018135	0.849931	0.859028	0.866111	0.903056
Ben Graham	**0.898667**	0.016924	**0.883681**	**0.900833**	**0.907153**	**0.928333**

Max-RGB maintain the reddish tone that characterizes them at the same time as they amalgamate. Regarding the reason of why the Ben Graham method gives such good results, it can be hypothesized that it is because its action of filtering the low frequencies of the image allows the network to focus on the finer details of the lesions.

As a second reason, it can be seen from the comparison images that the result of applying the Ben Graham method also achieves a certain color correction effect: all images (not just some as in the Shades of Gray method) are now found leaning towards blue and brown tints. In other words, the global information on the tone of the image is homogenized together with the reduction of low frequencies.

8.3.2 Attention Residual Learning on EfficientNet

In this experiment we try to study the effect of adding the Attention Residual Learning (ARL) mechanism on the Efficient-Net models. Specifically, it is not only interesting the addition of the mechanism to the base model, but we also study the way it relates to the attention mechanism already present: Squeeze & Excitation (SE). For this, we study how four EfficientNet variants behave (corresponding to the possible combinations of having these two attention mechanisms activated or not).

We employ the EfficientNet-b0 variant. Each one of the four networks is pretrained on ImageNet. The first one will serve as a baseline for comparison, on the second

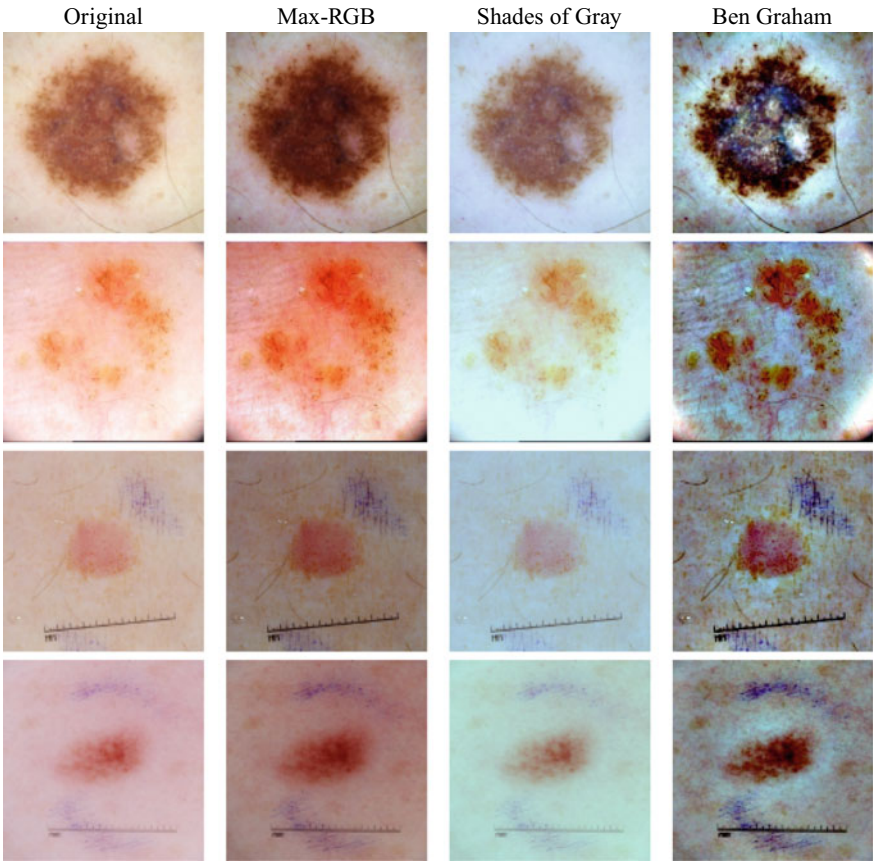

Fig. 8.2 Comparison between different image preprocessing methods

one we will suppress the SE mechanism, on the third one not only we will suppress the SE but will also add the ARL mechanism and finally the fourth and last network will have both attention mechanisms activated. EfficientNet-b0 has only 9 blocks on which a skip connections is used, therefore when adding ARL we will have only 9 new α parameters to train.

8.3.2.1 Results and Interpretation

As can be seen on Tables 8.4 and 8.5, after 10 runs with different initial seeds, the maximum average accuracy reached by the model using ARL and SE outperforms the base model. It can be seen a considerable difference between models that have SE enabled from the models that don't. This gives us the insight that a big part of

Table 8.4 Accuracy results on the baseline EfficientNet-b0 and their combinations of attention mechanisms

Dataset	Mean	Std	25%	50%	75%	Max
Baseline(SE)	0.857333	0.007166	0.853333	0.860000	0.860000	0.866667
No attention	0.834667	0.007569	0.833333	0.833333	0.840000	0.846667
ARL	0.829333	0.012649	0.821667	0.833333	0.838333	0.846667
SE and ARL	**0.862000**	0.007730	**0.860000**	**0.863333**	**0.866667**	**0.873333**

Table 8.5 AUC-ROC results on the baseline and their combinations of att. mechanisms

Dataset	Mean	Std	25%	50%	75%	Max
Baseline(SE)	0.852667	0.009516	0.848472	0.850972	0.861042	0.865278
No attention	0.788500	0.016842	0.780347	0.794306	0.797639	0.813056
ARL	0.786250	0.012818	0.769722	0.785555	0.796875	0.804445
SE and ARL	**0.853611**	0.009903	**0.848958**	**0.855694**	**0.861250**	**0.866667**

EfficientNet's great performance comes from the attention mechanism rather than from the new architecture.

8.3.2.2 Qualitative Analysis with GradCAM

In addition to the metrics that allow a quantitative analysis, it is possible to perform a qualitative analysis of the models by observing the result of GradCAM [32]. The GradCAM technique allows viewing heatmaps on images to understand which sections of the images have the greatest influence on the classification. Fig. 8.3 shows a comparison between the heatmaps generated by the original model and those generated by the model with ARL.

Looking at Fig. 8.3 the third row is the one that seems to expose the differences most clearly. The introduction of ARL appears to significantly reduce the area of the image that the network considers relevant for classification. That is, one can visually see that the attention mechanism is working properly. This is especially noticeable in the model that has ARL as the only attention method (EfficientNet model from which the SE mechanism is removed), although it is also observed in a more subtle way in the model that it has both ARL and SE mechanisms.

A second striking aspect to notice appears in the example in the second row where one can see how all the models seem to focus on the area surrounding the injury rather than the injury itself. This would be an indication that the tissue surrounding the lesion also provides valuable information. Particularly in the original image corresponding to the example of the second row, the surrounding tissue is covered with red marks, which does not appear to be information that can be ruled out.

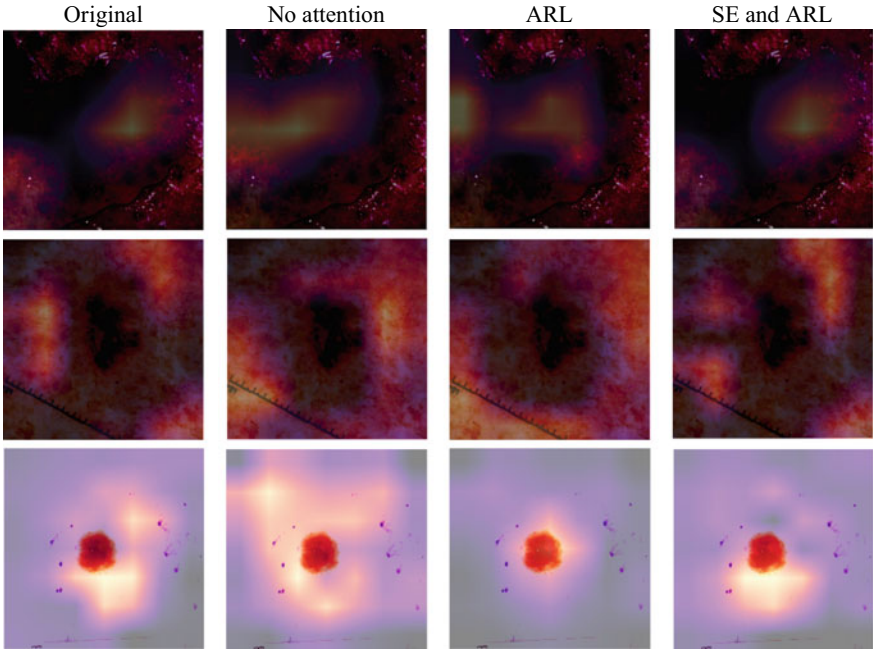

Fig. 8.3 GradCAM heatmap comparison for models with or without ARL. In the first and second row examples, only the model with SE and ARL correctly predict the presence of melanoma. In the example in the third row, all models correctly predict the absence of melanoma

8.4 Conclusions

In the present work, various mechanisms have been studied to improve the performance of the classification of skin lesions using convolutional neural networks.

Regarding the experiments on dataset preprocessing in Sect. 8.3.1 it has been found that Ben Graham's preprocessing notably improves the accuracy of the classification. This is not the case with the Max RGB and Shades of Gray color correction algorithms, which give no considerable increases or even inferior results respectively to the original control dataset.

Regarding the experiment in Sect. 8.3.2 which studies the impact of introducing attention mechanisms in networks -particularly the ARL mechanism- it has been found that the introduction of ARL improves the accuracy in the models of the EfficientNet family.

References

1. Abbas, Q., Celebi, M.E., García, I.F.: Hair removal methods: a comparative study for dermoscopy images. Biomed. Signal Process. Control **6**(4), 395–404 (2011)

2. Adegun, A., Viriri, S.: Deep learning techniques for skin lesion analysis and melanoma cancer detection: a survey of state-of-the-art. Artif. Intell. Rev. **54**(02), 811–841 (2021)
3. Akram, T., Lodhi, H.M.J., Naqvi, S.R., Naeem, S., Alhaisoni, M., Ali, M., Haider, S.A., Qadri, N.N.: A multilevel features selection framework for skin lesion classification. Hum.-Centric Comput. Inf. Sci. **10**, 1–26 (2020)
4. Almaraz-Damian, J.A., Ponomaryov, V., Sadovnychiy, S., Castillejos-Fernandez, H.: Melanoma and nevus skin lesion classification using handcraft and deep learning feature fusion via mutual information measures. Entropy **22**(4) (2020)
5. Alqudah, A.M., Alquraan, H., Qasmieh, I.A.: Segmented and non-segmented skin lesions classification using transfer learning and adaptive moment learning rate technique using pretrained convolutional neural network. J. Biomim., Biomater. Biomed. Eng. **42**(8), 67–78 (2019)
6. Argenziano, G., Fabbrocini, G., Carli, P., de Giorgi, V., Sammarco, E., Delfino, M.: Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions: comparison of the ABCD rule of dermatoscopy and a new 7-point checklist based on pattern analysis. Arch. Dermatol. **134**(12), 1563–70 (1998)
7. Barata, C., Celebi, M.E., Marques, J.S.: Improving dermoscopy image classification using color constancy. IEEE J. Biomed. Health Inform. **19**(3), 1146–1152 (2015)
8. Barata, C., Celebi, M.E., Marques, J.S.: A survey of feature extraction in dermoscopy image analysis of skin cancer. IEEE J. Biomed. Health Inform. **23**(3), 1096–1109 (2019)
9. Bi, L., Kim, J., Ahn, E., Kumar, A., Fulham, M., Feng, D.: Dermoscopic image segmentation via multistage fully convolutional networks. IEEE Trans. Biomed. Eng. **64**(9), 2065–2074 (2017)
10. Clawson, K.M., Morrow, P.J., Scotney, B.W., McKenna, D.J., Dolan, O.M.: Determination of optimal axes for skin lesion asymmetry quantification. In 2007 IEEE International Conference on Image Processing, vol. 2, pp. II–453–II–456 (2007)
11. Codella, N.C.F., Celebi, M.E., Dana, K., Gutman, D., Helba, B., Kittler, H., Tschandl, P., Halpern, A., Rotemberg, V., Malvehy, J., Combalia, M.: International Skin Imaging Collaboration (ISIC) Challenge: using dermoscopic image context to diagnose melanoma (2020)
12. El-Khatib, H., Popescu, D., Ichim, L.: Deep learningbased methods for automatic diagnosis of skin lesions. Sensors **20**(6) (2020)
13. Celebi, E.M., Wen, Q., Hwang, S., Iyatomi, H., Schaefer, G.: Lesion border detection in dermoscopy images using ensembles of thresholding methods. Ski. Res. Technol. **19**(1), e252–e258 (2013)
14. Farr, M.A., Duvic, M., Joshi, T.P.: Teledermatology during covid-19: an updated review. Am. J. Clin. Derm. 1–9 (2021)
15. Gessert, N., Nielsen, M., Shaikh, M., Werner, R., Schlaefer, A.: Alexander: skin lesion classification using ensembles of multi-resolution EfficientNets with meta data. MethodsX **7**, 100864 (2020)
16. Agnessa Gadeliya Goodson and Douglas Grossman: Strategies for early melanoma detection: approaches to the patient with nevi. J. Am. Acad. Dermatol. **60**(5), 719–735 (2009)
17. Graham, G.: Kaggle diabetic retinopathy detection competition report (2015)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
19. Henning, J.S., Dusza, S.W., Wang, S.Q., Marghoob, A.A., Rabinovitz, H.S., Polsky, D., Kopf, A.W.: The cash (color, architecture, symmetry, and homogeneity) algorithm for dermoscopy. J. Am. Acad. Derm. **56**(1), 45–52 (2007)
20. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
21. Iyatomi, H., Celebi, M.E., Schaefer, G., Tanaka, M.: Automated color calibration method for dermoscopy images. Comput. Med. Imaging Graph. **35**(2), 89–98 (2011)
22. Lee, T., Ng, V., Gallagher, R., Coldman, A., McLean, D.: Dullrazor: a software approach to hair removal from images. Comput. Biol. Med. **27**(6), 533–543 (1997)
23. Menzies, S.W.: An Atlas of Surface Microscopy of Pigmented Skin Lesions: Dermoscopy. McGraw-Hill (2003)

24. Nachbar, F., Stolz, W., Merkle, T., Cognetta, A.B., Vogt, T., Landthaler, M., Bilek, P., Braun-Falco, O., Plewig, G.: The ABCD rule of dermatoscopy: high prospective value in the diagnosis of doubtful melanocytic skin lesions. J. Am. Acad. Dermatol. **30**(4), 551–559 (1994)
25. Ng, V.T., Fung, B.Y., Lee, T.K.: Determining the asymmetry of skin lesion with fuzzy borders. Comput. Biol. Med. **35**(2), 103–120 (2005)
26. Quintana, J., Garcia, R., Neumann, L.: A novel method for color correction in epiluminescence microscopy. Comput. Med. Imaging Graph. **35**(7), 646–652 (2011)
27. Ratul, A.R., Mozaffari, M.H., Lee, W.S., Parimbelli, E.: Skin lesions classification using deep learning based on dilated convolution (2019)
28. Ruela, M., Barata, C., Marques, J.S., Rozeira, J.: A system for the detection of melanomas in dermoscopy images using shape and symmetry features. Comput. Methods Biomech. Biomed. Eng.: Imaging Vis. **5**(2), 127–137 (2017)
29. Sadri, A.R., Azarianpour, S., Zekri, M., Celebi, M.E., Sadri, S.: Wn-based approach to melanoma diagnosis from dermoscopy images. IET Image Process. 11(7):475–482 (2017)
30. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.: Mobilenetv2: inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
31. Schaefer, G., Rajab, M.I., Celebi, M.E., Iyatomi, H.: Colour and contrast enhancement for improved skin lesion segmentation. Comput. Med. Imaging Graph. **35**(2), 99–104 (2011)
32. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: visual explanations from deep networks via gradient-based localization. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 618–626 (2017)
33. Senan, E.M., Jadhav, M.E.: Analysis of dermoscopy images by using ABCD rule for early detection of skin cancer. Glob. Transit. Proc. **2**(1), 1–7 (2021). 1st International Conference on Advances in Information, Computing and Trends in Data Engineering (AICDE–2020)
34. Siegel, R.L., Miller, K.D., Jemal, A.: Cancer statistics. CA: Cancer J. Clin. **68**(1), 7–30 (2018)
35. Smith, L.N.: A disciplined approach to neural network hyper-parameters: part 1—learning rate, batch size, momentum, and weight decay (2018)
36. Stofa, M.M., Zulkifley, M.A., Zainuri, M.A.A.M.: Skin lesions classification and segmentation: a review. Int. J. Adv. Comput. Sci. Appl. **12**(10) (2021)
37. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: platform-aware neural architecture search for mobile. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2815–2823 (2019)
38. Tan, M., Le, Q.E.: EfficientNet: rethinking model scaling for convolutional neural networks. In: Proceedings of the International Conference on Machine Learning, vol. 97, pp. 6105–6114 (2019)
39. Vesal, S., Ravikumar, N., Maier, A.: Skinnet: a deep learning framework for skin lesion segmentation. In: 2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC), pp. 1–3 (2018)
40. Xie, F., Fan, H., Li, Y., Jiang, Z., Meng, R., Bovik, A.: Melanoma classification on dermoscopy images using a neural network ensemble model. IEEE Trans. Med. Imaging **36**(3), 849–858 (2017)
41. Lequan, Yu., Chen, H., Dou, Q., Qin, J., Heng, P.-A.: Automated melanoma recognition in dermoscopy images via very deep residual networks. IEEE Trans. Med. Imaging **36**(4), 994–1004 (2017)
42. Yuan, Y., Lo, Y.-C.: Improving dermoscopic image segmentation with enhanced convolutional-deconvolutional networks. IEEE J. Biomed. Health Inform. **23**(2), 519–526 (2019)
43. Zhang, J., Xie, Y., Xia, Y., Shen, C.: Attention residual learning for skin lesion classification. IEEE Trans. Med. Imaging **38**(9), 2092–2103 (2019)
44. Zhou, H., Schaefer, G., Celebi, M.E., Lin, F., and Liu, T.: Gradient vector flow with mean shift for skin lesion segmentation. Comput. Med. Imaging Graph. **35**(2), 121–127 (2011). Advances in Skin Cancer Image Analysis

Chapter 9
Melamine Faced Panel Inspection, Towards an Efficient Use of Natural Resources

Fernando P. G. de Sá, Cristhian Aguilera, Cristhian A. Aguilera, and Aura Conci

Abstract Today an adequate use of natural resources respecting the extraction, transformation, and consumption limits considering that we live in an already lacking in natural reserves planet is paramount. The growing environmental consciousness inserted in the new economy encourages the industry to adopt new technologies in order to optimize production processes. The timber industry deals with a feed-stock sensitive to environmental appeals, which demand tight control over the origin of the processed wood. The adoption of automatic and non-destructive techniques to find defects in wood products improve the use of several natural resources such as soil, water, and others environmental provisions. In this work, we present a new image processing technique for quality control tasks of melamine boards. This only uses information from one channel acquired in the near-infrared spectrum. We carry out a classification study of defects present in the manufacture of melamine boards. We consider the most important type of defects: stains, paper (displacement or) detachment, (paper wrapped or) attached material, wrinkled paper, and folded paper. Each type of defects presents different cardinalities, demanding a number of considerations for set normalization and definition of proper techniques for adjustments in

F. P. G. de Sá (✉) · A. Conci
Institute of Computing—IC, Federal Fluminense University, Gal. Milton Tavares de Souza s/n, Niterói, 24210-310, (Rio de Janeiro - RJ), Brazil
e-mail: fernandosa@id.uff.br
URL: http://www.ic.uff.br/

A. Conci
e-mail: aconci@id.uff.br

C. Aguilera
Department of Electrical and Electronics Engineering, University of Bío-Bío, Collao 1202, Concepción 4051381, Chile
e-mail: cristhia@ubiobio.cl
URL: http://www.biobio.cl/

C. A. Aguilera
Departamento de Ciencias de la Ingeniería, Universidad de los Lagos, Av. Fuchslocher 1305, Osorno 5290000, Chile
e-mail: cristhian.aguilera@ulagos.cl
URL: http://www.ulagos.cl/

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_9

the data mining process of unbalanced sets. The results outperform the literature's previous work on the classification of defects of the same type of boards.

Keywords Surface fault detection · Near-infrared images · Melamine faced panels

9.1 Introduction

The best use of wood and the fabrication of defect-free boards for the construction and furniture industry are key elements for the construction of houses and furniture industry in an environment friendly cities. In recent years, the manufacture of boards for use in the creation of modern spaces in cities has undergone a significant change from the point of view of new designs and materials. These bring new challenges in terms of final quality. In the case of melamine boards, quality control is a critical task that requires an increasingly intelligent inspection system to identify defects. Advances in technology (particularly in the field of Industry 4.0) enable the generation and integration of intelligent systems that are consistent with production processes. Final products without defects, compliance with regulations, and quality standards do not only affect the productivity rates of companies but also play a notable role in positioning of companies in the market, due to the consequent rejection reductions and especially in customer loyalty.

The wood industry has significant global growth and a high projection. The wood-based panel market was estimated at over 1000 million cubic meters in 2021, and the market is projected to register a CAGR (Compound Annual Growth Rate) of over 6% during the forecast period (2021–2026), according to Mordor Intelligence (https://www.mordorintelligence.com/industry-reports/wood-based-panel-market).

This growth also includes different types of products such as: boards for the construction industry; medium density fiberboard (MDF); high-density fiberboards (HDF); oriented strand board (OSB) and melamine boards. These new products are also characterized by new designs, beyond whiteboards or solid colors, generating boards with complex multicolor designs that make the fault detection difficult. The detection and classification of superficial defects, particularly in melamine boards, is one of the most important tasks in the quality control phase of manufacturing companies. Since it allows generating and evaluating statistics that help to correct possible errors in the manufacturing process and also allows the panels to be classified into various classes according to the destination market.

Different defects may appear in the board manufacturing process. Some of those defects, such as stains, for example, are very common in wood products, but others, such as glued paper, are more rare [1]. Defects result in products with different qualities and consequently marked valor (and prices). In the commerce of objects made of them, it is necessary to identify what type the fault could be present in it, to guarantee: (1) better use of natural resources in the wood industry, (2) the correct pricing and (3) adequate quality information to the customers. Moreover, identifying a type of defect sometimes can be very difficult, because some of them need very

knowledgeable experts. Due to this a number of imaging have been used to help with defect identification by using computer vision and pattern recognition techniques. Techniques like x-ray scanning, visible light optical acquisitions (by RGB cameras), ultrasound, microwave sensors, terahertz scanners, and even moisture of separated parts of the electromagnetic spectrum have been investigated in terms of possibilities to provide complementary information for quality control because each one presents advantages and difficulties on these tasks.

In this work, an exploratory study is presented for the classification of defects in melamine boards using the Near-Infrared Images (NIR) wavelength. The used data-set was obtained in an industrial production line of melamine boards, by means of an array of multiple cameras, capable of taking information in real-time from boards. In this development, a total of 2,778 different samples of infrared images, each one stored in Portable Network Graphics (PNG) format are used. The complete labeled data available consists of 1,919 samples for training, 572 for validation, and 287 for test, all from 5 types of surface defects and a group of perfect boards. The data set was obtained under normal operating conditions of the plant, generating an unbalanced data set naturally related to the frequency of occurrence of defects in the industry on observation [1].

Melamine is a board made of chipboard particles, covered with a sheet of melamine resin named ***paper*** in industrial plant. The board is composed of small fragments of pinewood, pressed and selected to later be mixed with special adhesives; generally based on water, resin, and chemical hardeners. After the particleboard and the sheet are fused together using a heating and pressing system, completing the manufacture of the board.

In the board manufacturing process, faults can occur in each of the production stages and they can be reflected in the final quality of the board. A critical aspect in the quality of the boards is the appearance of stains or foreign objects on the board that alterates its visual and aesthetic condition, a factor that is very sensitive for end customers. Another common problem is the sliding of the melamine thin plate on the board. Although this mistake may not be critical (depending on the displacement), it is difficult to detect by manual and automatic inspection systems.

For this work, five common surface defects in the fusion of the sheet and the board are considered. In Fig. 9.1 we can see the appearance as NIR images of defects in melamine boards: (a) Normal is a board without defect; (b) Detachment of material, occurs when the board is damaged and has some exposed parts without melamine laminate; (c) Stained or smudged paper, appears when stains appear on the final product, due to poor handling of the sheet or the presence of dust and suspended particles; (d) Displaced or wrinkled paper, appears when an unwanted displacement or wrinkle of the sheet occurs, preventing the board from being completely flat; (e) Paper broken (or folded), appears when the sheet breaks (or folds) and part of the board are exposed; and (f) Attached or pasted paper appears when an unwanted piece of sheet appears below the melamine thin plate to be glued. Such a description of the defect is very important because their names could vary among the different plants, inspection organizations, or institutions. The defects described above represent more than 97% of defective boards in production. This is why the timely detection and

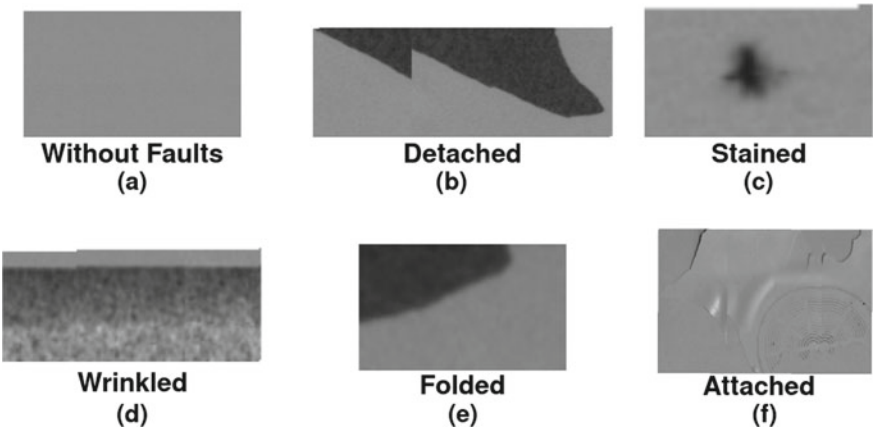

Fig. 9.1 NIR appearance of the types of defect on melamine board surfaces considered in this work

classification of those defects is crucial for the commercialization of boards in the international market.

9.2 Related Works

The rapid development of artificial intelligence technology has lead to its increasing application in different fields. The use of computer-aided diagnosis employing data driven techniques was the object of recent works for wood defect detection. In these works one remarkable element is the adoption of nondestructive testing to reach economic and sustainability requirements. A wide variety of inspection techniques meet these requirements, including laser testing, acoustic-emission technology, computer tomography, high-speed camera, among others. In general, the application of these techniques generates an amount of data which are later processed to find worthy insights.

For example, in computer vision-related works using data from images taken by visible-light (RGB) cameras, we found in the literature the use of the gray-level co-occurrence matrix (GLCM) method for segmentation and classification of knots on wooden surfaces [25]. Tamura et al. [20] presented a system based on texture parameters. In another work presented by Yuce et al. [26], the aforementioned approach is adopted to classify different types of defects in wood plates using artificial neural networks (ANN).

Others authors used the support vector machine (SVM) classification algorithm in their works [6] combined with different feature extraction methods. Mahram et al. [13] focused on the intensity and the size of different types of knots on wooden

surfaces. Cover et al. [7] developed an approach merging texture descriptors and K-nearest neighbor (KNN) in addition to SVM [19].

Brahnam et al. [3] proposed a method of detection and classification of knots and cracks using local binary patterns (LBP) as feature descriptors [3]. Prasitmeeboon et al. [16] analyze bivariate histograms and Nurthohari et al. [14] use the histogram of oriented gradients (HOG) as feature vectors in their analysis. These and other works present different methods to detect common defects on the surface of wood such as knots, cracks, stains or holes [2, 18].

On the other hand, many applications based on image processing have explored the use of information in the near-infrared (NIR) spectrum. This allows obtaining important results because it opens the possibility of processing information that is not available when working only in the visible spectrum (VIS). Semantic segmentation projects, for example, have been presented by Salamati et al. [18] and Bigdeli et al. [2] using information from the usual R, G, B channels plus the near infrared channel. Works such as those developed by Sharma et al. [19], Lee et al. [11] and Zhang [27] using different algorithms enhance images in terms of reducing light influence, perception of details, and texture evaluations.

The detection of defects in production lines has also been favored by contributions that make use of images in the infrared spectrum. For instance, Yean et al. [24] perform an analysis of the NIR spectrum to detect micro defects in silicon wafers on solar cells (that by the particular nature of the material are very difficult to detect). Hamdi et al. [8] propose a system for detecting defects in a production line of textile products by overcoming the lighting disturbances typical of the industrial process. In wood knot detection works, such as the one presented by Hamdi et al. [8], the aim is to improve performance indices as in Aguilera et al. [1], where a classification system is evaluated through the use of visible and near-infrared images, applying LBP, and using a bag-of-words (BOW) representation and the speed-up robust features (SURF) method for grouping the feature with similar representation.

9.3 Theoretical Background

From the point of view of algorithms, as seen above, the development of new methods in the area of machine learning has opened the door to new approaches to address the superficial defect detection problem. In this work, we adopt two common approaches of image analysis and multiclass classification processing. The following subsections are dedicated to the theoretical foundation of the used local binary pattern (LBP) and support vector machine (SVM) methods.

9.3.1 Local Binary Pattern (LBP)

The local binary pattern (LBP) was proposed as a technique for texture representation by [15]. It has a number of variations on the distance from the evaluated pixel neighborhood and directions of computation [21].

The LBP general feature computation generates a code of each pixel considering its values and those of its surroundings using a 2 step algorithm. In the first step the pixel in analysis for generation the binary code is compared with the neighbor pixels. In case of its value being smaller than those of any neighborhood position, then such a position is set as 1, otherwise as 0. The third image of Fig. 9.2 (top-right) shows an illustration of a Local Binary Pattern.

In the second step, a binary code is formed considering the set of zeros and ones formed in the fist step of the LBP feature computation. This code gives the name to the technique and it is attributed to the central pixel, the named g0 in Fig. 9.3. However, for each position where the binary number formation begins a different value can be formed. This image shows the binary code and its value (as a decimal number) when the initial position is the top left of the comparison neighborhood achieving: $(10000111)_B = (135)_D$.

Of course even using this clockwise orientation and formation sequence this is only one of 7 other possibilities: $(00001111)_B = (15)_D$; $(00011110)_B = (30)_D$; $(00111100)_B = (60)_D$; $(01111000)_B = (120)_D$; $(11110000)_B = (240)_D$; $(11100001)_B = (225)_D$; and $(11000011)_B = (195)_D$. In other words the 8 bits number formation can begins in any of the g1.... g8 position showed in Fig. 9.3.

Moreover, an anticlockwise orientation (as for instance g8, g7, g6, g5, g4, g3, g2, g1) can be used and other number formation sequence as well (as for instance g7, g6, g5, g4, g8, g1, g2, g3). Some of such order present special meaning as a

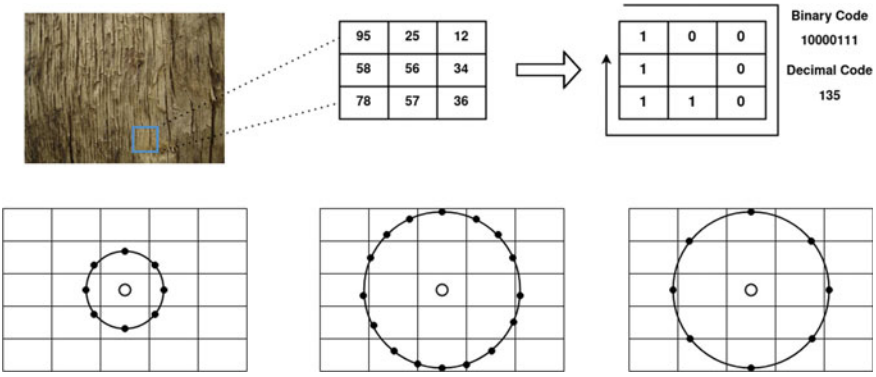

Fig. 9.2 A neighborhood of an image for distance 1, its gray scale pixel intensities and binary result after comparison between central point and each neighbor (when codification begins in the top left point). Configurations for distance 1 from central pixel and for distance 2 (considering 16 consecutive or 8 alternated values)

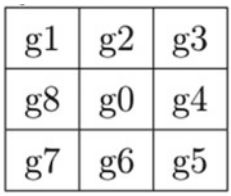

Fig. 9.3 Reference for the positions at neighborhood 1 to aid on explain the calculation of Eqs. (9.1, 9.2)

representation invariant to reflection or rotations and can be used in order to improve the identification of elements like a constant strap or edge defined by the level or tones.

After the definition of this order, a new position for the set of zeros and ones (of the first step) appears and the named g' representation of Fig. 9.3 is obtained. This new order can be considered in the LBP Eq. (9.1):

$$lbp(g_0) = \sum_{n=1}^{N-1} g'_n 2^{n-1} \tag{9.1}$$

where g'_n={0,1}, g'_n=1 if the pixel value in position n is greater then the pixel value in g_0 and g'_n=0 otherwise.

There are a number of possible variations related to what can be understood as neighborhood, that is the open or close ball for each point of the image or pixel [10] (p. 102). The Euclidean distance or d_2 is the usual one. This distance in $\Re^2$ is defined by Eq. (9.2) considering $p = 2$. However, for the same pixels organization, other distances can be obtained depending on what is the value of p fixed in the general expression used to compute the distance. p defines in this way d_1, d_2 and d_∞, based on Eq. (9.2) [10] (p. 88):

$$d_p(g'_0, g'_n) = \left(|g'_{0x} - g'_{nx}|^p + |g'_{0y} - g'_{ny}|^p\right)^{1/p} \tag{9.2}$$

where $p = \{1, 2, \ldots, \infty\}$, x, y are the pixel axial orientation for $\Re^2$ (that is : $g'_n = (g'_{nx}, g'_{ny})$ and $g'_0 = (g'_{0x}, g'_{0y})$). These x, y coordinates are related to the central point depicted in Fig. 9.4. When $p = 2$ it is named Euclidean distance of $\mathbb{R}^2$, the distance from each position g'_n to the central g'_0 are presented in Fig. 9.5. The borders of the neighborhood up to a defined distance have the appearance of circle (Fig. 9.2).

There are other appearances for such balls or neighborhoods depending on the distance function. Figure 9.6 shows 3 neighborhood appearance for $\rho = \{1, 2, \infty\}$. The numbers present in each position by different colors are the distance computed using Eq. (9.2): from each pixel to the central one. The central image on Fig. 9.6 represents the results using the same distance function of Fig. 9.5 but now with an approximation of 2 decimals only.

Finally, it is important to realize that not all the pixels compounding the ball border up to a given distance need to be used to represent the LBP of an image on analysis.

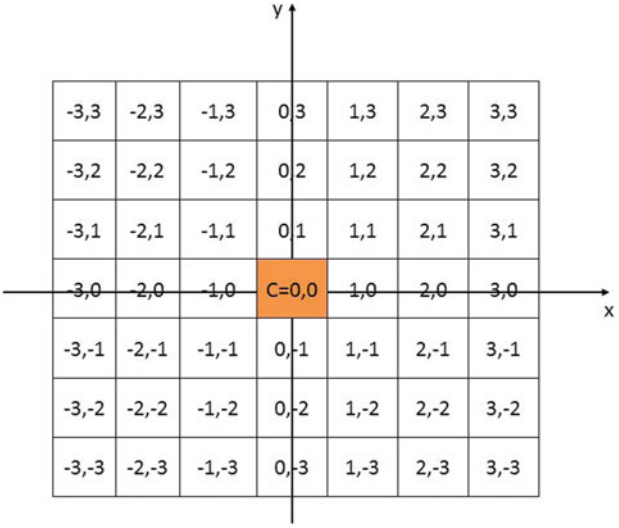

Fig. 9.4 Coordinates of horizontal and vertical directions 1 and 2 for pixels x, y distance computation

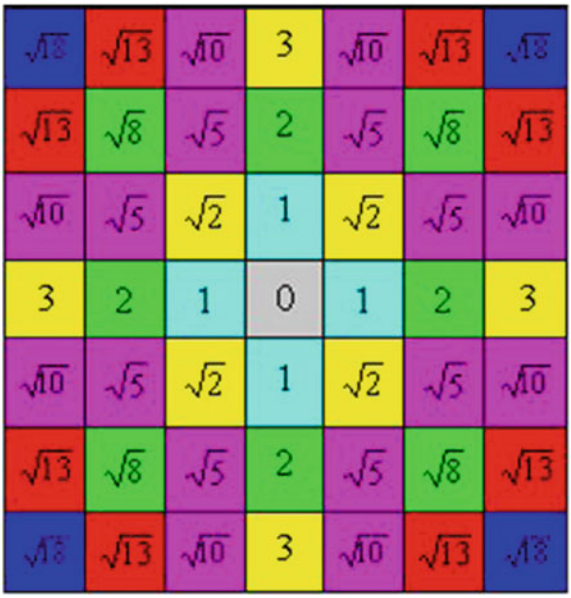

Fig. 9.5 Equation (9.2) related to $p = 2$ (or Euclidean distance d_2) up 3 neighbors for LBP computation

Fig. 9.6 Open ball appearance related to $p = \{1, 2, \infty\}$ on Eq. (9.2) computations, considering neighbourhood of 7 for d_1 and d_∞

For instance for Euclidean distance 2 where a neighborhood of more than 8 pixels around a central one is the usual result (see Fig. 9.2 bottom row) the LBP can be computed using only 8 values, when a number up to 1 byte is enough to store these LBP possible values. However, when greater numbers of 0 and 1 are employed for the formation of the LBP more bits must be necessary to store this binary number (that will be greater than a value stored in 8 bits, resulting in a need of 2 bytes or more than 3 bytes). That is: in some cases, the maximum number generated for the LBP code can be up to $255 = 2^8 - 1$, or $65,535 = 2^{16} - 1$ or $2^{32} - 1$, and so on. Consequently, depending on the LBP implementation this 1 or 0 can be used up to any number for the composition of a binary number and this can be a parameter to be decided considering the problem under study. The central image in Fig. 9.2 bottom row illustrates the case where 16 is used and, the right image shows the option for the case when only 8 is used, both when Euclidean distance is 2.

9.3.2 Support Vector Machine (SVM)

Support vector machines are a class of statistical models with extensive use in pattern recognition problems. It attempts to find a suitable hypothesis to the complexity of the training data while minimizing the upper bound of the generalization error [22]. In classification problems, a SVM aims to create a maximum-margin hyperplane to separate the data in distinct classes by choosing a function that transforms the original feature space ensuring the structural risk minimization principle [22].

Let $\mathcal{S}$, the training set of features of N classes, be represented by a D dimensional feature vector, that is $\mathcal{S} = \{(\boldsymbol{x}_n, y_n) \mid \boldsymbol{x}_n \in \Re^D,\ y_n \in \mathbb{Z},\ n = 1, \ldots, N\}$. By a kernel function ϕ, the input data is projected into a high dimensional feature space to maximize the power of the separating hyperplane. This optimization problem is solved using quadratic programming, whose model is defined below:

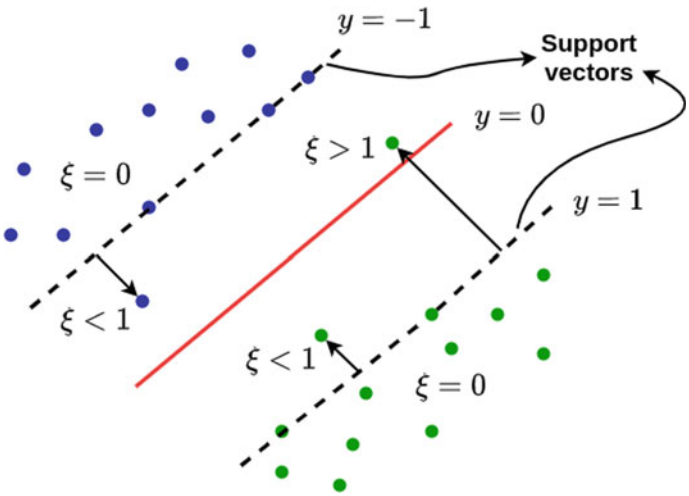

Fig. 9.7 Geometric representation of SVM: a red hyperplane discriminates blue and green classes

$$\begin{aligned} &\min_{\boldsymbol{w},b} && \tfrac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ &\text{s.t.} && y_n(\boldsymbol{w}^T\phi(x_n) + b) \geq 1 - \xi_n, \\ & && \xi_n \geq 0,\ n = 1, \ldots N. \end{aligned} \tag{9.3}$$

where $\boldsymbol{w}$ and b are the model parameters, ϕ is kernel function, and C is a hyper-parameter that controls the trade-off between the model complexity and the margin width.

For binary SVM classification, the ground truth labeling is done using $y_n = \pm 1$, with the sign determining if a training instance n is from the positive or the negative class. Once training is complete, the result of Eq. (9.3) induces a straight line separating both classes. Figure 9.7 shows the maximum hyperplane where $\boldsymbol{w}x + b = 0$ discriminates the classes as follows: $\boldsymbol{w}x + b \geq 1$ when $y_n = +1$, and $\boldsymbol{w}x + b \leq 1$ when $y_n = -1$. Data points over the dotted lines are known as the support vector for satisfying the equality in the equations. As can be seen in the figure, the slack variables ξ are highlighted to show their values when violated.

Although SVM is inherently a binary classifier, it fashions two possible arrangements in order to support multiclass classification: one-against-one method and one-against-the-rest method [23]. The one-against-one approach is adopted to handle the multiclass classification in our results. The main characteristic of this method is to break down the multiclass problem into multiple binary classification problems. Thus, let m be the number of classes, the one-against-one approach can construct $\frac{m(m-1)}{2}$ hyperplanes respecting the optimized separation between each one of the m classes.

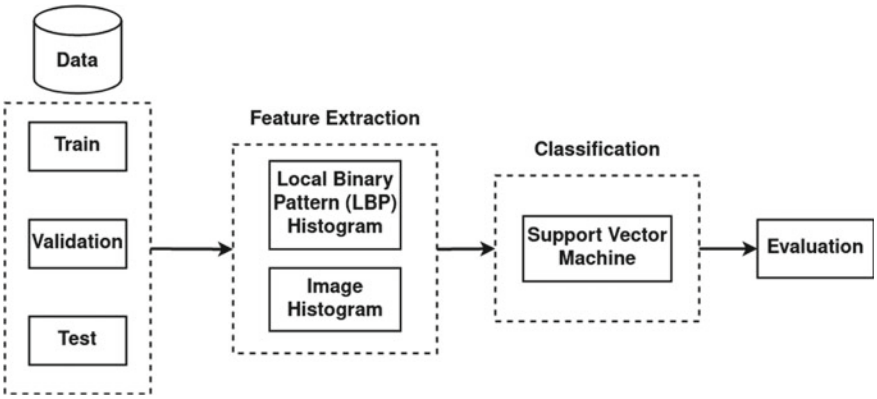

Fig. 9.8 General approach

9.4 Used Methodology for Surface Classification

Before discussing independently each part of the proposed methodology, let us present the general used approach, which consists in feature extraction and classification learning, as presented in Fig. 9.8. It is important to note that in this type of application, i.e. a continuous inspection [4], differently of the usual image processing for object classification there is not the phase of segmentation [17], because there is not objects to be separate of the background in the acquired scene to be analysed.

As can be seen in the Fig. 9.8, from the available data set we extract the features of the gray level counting directly the number of each level present in the images or organizing the level according to their position by using local binary patterns (LBPs). After, these features feed the classification algorithm of support vector machine (SVM) aiming to identify each sample as with or without defects. Finally, numerical indexes are responsible to analyse the quality of the obtained results.

9.4.1 Used Dataset

Here, the near-infrared spectrum is used because this type of imaging presents advantages on the identification of superficial irregularities [1]. For the image acquisition, a system of multiple cameras and industrial controllers was designed and constructed allowing the simultaneous shooting of each of the cameras to cover a complete board. Such boards have real dimensions of 1.83 × 2.50 m. A total of 3 infrared spectrum cameras were used in a melamine panel manufacturing line operating at a speed of 1 m/s. All cameras are connected via a Giga Ethernet network. Basler Aca 1300-60 gm NIR cameras of 1,280 × 1,024 pixels were used, generating images of complete boards with a resolution of 3,570 × 5,770 pixels [1].

Table 9.1 Original data set composition for all phases

Classes	Training	Validation	Test	Total
Without faults	685	249	240	1,174
Detached	122	33	2	158
Stained	246	66	14	326
Wrinkled	587	149	19	755
Attached	157	41	3	201
Folded	124	34	8	164
Total	1,919	572	287	2,778

The NIR information of each board is stored as a single-channel. This channel can be considered as a grey scale image. Some of these are depicted in Fig. 9.1. They are used to compose the classes of the data set, each image having different numbers of pixels. However, their pixels are always described in 256 levels of the channel range. The data set is organized in a way that an image belongs only to one group. That is, the data set presents previous separation for training and validation of the learning phase and then final tests: this structure ensures that there is no sample present at the same time in more than one of these sets. The data set composition can be seen in Table 9.1.

9.4.2 Dealing with Imbalanced Data

Correct identification of the types of melamine board superficial issue can be very difficult, because most of the faults present very similar regions over some part of the sample (Fig. 9.1). This is specially complex for the groups of without faults, detached, stained, and folded types of defects. A total of 2,778 NIR spectrum images of melamine panels with defects and without defects were captured, generating a data set with the showed distribution (Table 9.1).

Once the occurrence of some types of defects is low, the acquired images during the inspection tasks compose a dataset naturally imbalanced, with the predominance of samples from the class 'without faults'. We adopt a simple under-sampling technique that selects random samples from the m classes to build a new training dataset whose size is limited by the size of minority class [9].

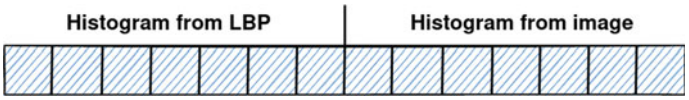

Fig. 9.9 Histograms compounding the feature vector

9.4.3 Used Programming Tools

We use the Python 3.8 computer language and its common libraries are adopted for data manipulation and machine learning development in this work. More specifically the following programming tools are used:

- Image processing: Scikit-image version 0.19.1, OpenCV version 4.5.5
- Machine learning: Scikit-learn version 1.0.2
- Array manipulation: NumPy version 1.22.1

9.4.4 Features Computation

The general techniques of image analysis can be helpful for the task of features extraction. The proposed approach has two groups of features. The first group represents features related to the pixel intensity and the second group is texture related, that is it considers the pattern distribution of the intensity in a considerable position of a given area (or texel) of each samples.

The Local Binary Pattern (LBP) approach is used for textural evaluation (that is the second group of features). This computation result can be changed by the selection of the parameters: `method`, `number of points` and `radius` of the used library.[1]

The parameter `radius` define the distance of the considered neighborhood, as represented in Fig. 9.2, top row. The parameter `method` has the options: 'default'; 'ror'; 'uniform' and 'var'. They mean the use of a different organization of the binary number to be created by implementation, they modify the order of the basic algorithm (or 'default'). This is employed in the second step of the LBP computation as explained in Sect. 9.3.1 and in the used library documentation. The `number of points` corresponds to the maximum possible code to be generated and attributed to the pixel as the LBP code.

After the computation of these features, the number of pixels in each possibility is used to form a histogram of the LBP group that represents the image as a feature vector in the next steps of learning.

[1] https://scikit-image.org/docs/stable/api/skimage.feature.html#skimage.feature.local_binary_pattern.

Both LBP and intensity level histograms can present some variations in their computations. The parameters radius, points and methods allow some adjustment in the LBP computations, related to the size of the neighborhood considered and the way the binary number representing the texture is formed, as already mentioned. In this work 8, 13, and 18 are the numbers of bits allowed for the final LBP binary code formation. The used distance was fixed as 3 and 8 (see Fig. 9.2). And the possible methods for organizing the code are considered.

For the intensity level histogram, the number of bins used is a possible variation. The variation of parameters from both histograms as grouped in 8 or 32 bins of division of the maximum values are considered in the classification results to be adjusted by the grid search approach for improving the next classification phase. Figure 9.9 exemplifies the used vector of features obtained by the histogram combinations. Table 9.2 shows these possible parameters to be investigates.

9.4.5 Classification Training and Testing

The next step is learning how to do the classification for each class. An important aspect in this is to choose a classifier that matches better the faults in order it will be used in a future implementation. In this article, the feature vectors are used as input for the Support Vector Machine (SVM) classification algorithm. There are elements for variation in the cost functions C, γ and the used kernel, these hyper-parameter possibilities are evaluated using Accuracy and F-score.

The fine tuning of SVM and feature parameters is necessary in order to better explore the search space. To accomplish this effort, the Grid Search was employed for an iterative evaluation of the parameters related to the ways the feature vector is built (that is the elements of possible variation in the histograms). The evaluation goes up to the end using the validation to identify the best possible features after training them with some infrared images labeled as with a specific fault type. It is employed in order to evaluate the SVM hyper-parameters to better predict the type of defect from new images in an industrial plant in the future. Figure 9.10 presents

Table 9.2 Parameter and hyper parameter search space

	Parameters/Hyperparameters	Used values				
Feature extraction	LBP method	'default'	'ror'	'uniform'		'var'
	Radius	3		8		
	Number of points	8		13	18	
	Interval of bins	8		32		
Classification	C	1	10	100	120	150
	γ	0.001		0.0001		
	SVM kernel (ϕ)	rbf		Linear	Polynomial	

details of in this work adopted strategy to find the best characteristics of the training set.

In the classification step, the validation set has the features used to evaluate the hyper-parameters (cost C, γ and kernel ϕ) according the performed classification. Finally, the test set, which also had its characteristics extracted, is used to evaluate the built model. Processing continues until the entire space of the grid search has been explored.

Table 9.2 presents the search space for grid search parameters and hyper-parameters during the validation processing. Note that the range of the histogram is evaluated only under two settings, 8 to 8 or 32 to 32 levels. Meanwhile, the search space for LBP (radius and number of points and the binary formation) are more extensive.

9.5 Results

The multi class classification (one vs all) performed by SVM was implemented using the workflow described in Sect. 9.4. The purpose of this classification is to distinguish normal samples from those identified as different types of failures, as

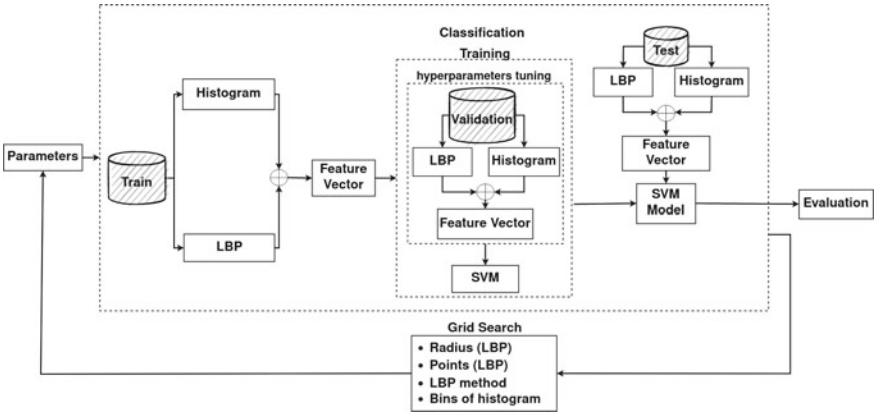

Fig. 9.10 Adopted processing strategy

Table 9.3 Optimal parameters from feature extraction phase

Parameters	Value
Radius	28
Number of points	8
Method	Default
Bins of histogram	8

Table 9.4 Optimal hyperparameters from the SVM classifier

Hyperparameters	Value
C	150
kernel (ϕ)	linear

Table 9.5 Resulting multi class confusion matrix

	Without faults	Detached	Stained	Wrinkled	Attached	Folded
Without faults	242	0	1	0	6	0
Detached	0	24	2	2	0	5
Stained	7	0	58	0	1	0
Wrinkled	0	3	12	129	0	5
Attached	2	0	1	0	38	0
Folded	0	3	1	1	0	29

described in Table 9.1. The complete exploration of the parameters in the search space was made by grid search and it identified the optimal parameters involved in the feature extraction presented in Table 9.3. The model validation (after training) determined the optimal SVM hyper parameters described in Table 9.4.

Table 9.5 presents the fine-tuned set of parameters and hyper parameters results fitted for each class. The respective averages of Precision, Recall and F-scores obtained in the classification process are presented in Table 9.6. As you can see, the order from best to worst (or vice versa) varies according to the "metric" used for comparison. For example, for the F-score: normal samples were identified more correctly than any type of defect, reaching a value of 97%, and wrinkled paper is the best identified type of defect (92%). On the opposite side, Detached and Folded faults are the ones with the least satisfactory F-score results. This presents higher values for the class of wrinkled defects and worse for folded papers.

Although, we are collectively calling Accuracy, Precision, Recall and F-scores "metric", note that we are not in fact actually processing them as metric in the metric space sense. Moreover, we are not even concerned that any of them are in fact a metric, or even a semi-metric in a topological space [10]. They are in fact only being here used as a comparative index of performance of the final process, in absence of a more proper option [5].

9.6 Comparisons with Previous Work

It is important to evaluate our work in the light of other solutions for the same problem. The work developed by [1] treats similar problem of detection of defects

Table 9.6 Results for each class considering various "metrics"

Class	Precision	Recall	F-score	Overall accuracy	Macro F-score
Without faults	0.96	0.97	0.97	**0.91**	**0.86**
Detached	0.80	0.73	0.76		
Stained	0.77	0.88	0.82		
Wrinkled	0.98	0.87	0.92		
Attached	0.84	0.93	0.88		
Folded	0.74	0.85	0.79		

on melamine-faced board. There are three classes of faults equivalent there with the present work. The Table 9.7 presents those that enable comparison between the two works based on results of classification accuracy of the classes 'Stained', 'Wrinkled' and 'Without Faults'.

As seen in Table 9.7, both solutions present equivalent results for 'Wrinkled' and 'Without Faults'. The class 'Stained' presents the higher gain of classification performance using the methodology discussed in this work. Future works can improve the actual results using the approach of feature construction discussed by [28].

9.7 Conclusion

This work investigates the possibilities of using a single infrared images on the classification of melamine surfaces as with or without defect, considering data set with six (6) possible groups and evaluating 2778 samples of surface, from five (5) types of faults (Table 9.1). The used data set present originally unbalanced numbers of samples in each type of class. Two types of features are considered in the SVM using grid search and hyper-parameters fine tuning evaluation. In order to train the classifier, the original data set was separated for learning, validation, and test. To promote better balance among classes, each of the classes is adjusted to present the same number of elements where each element has been selected at random.

Comparisons with previous work were investigated. Aguilera et al. [1] obtained theirs results after a standardization of the sizes of the analyzed samples and adopting

Table 9.7 Comparing average classification considering equivalent classes with previous work

Classes	Aguilera et al. [1]	Our work
Stained	0.84	0.96
Wrinkled	0.97	0.96
Without faults	0.95	0.97

Extended Local Binary Pattern (E-LBP) [12] to extract the features from images. Other details as the SVM model (to perform multi-class classification) used are just like our methodology [1].

Observing Table 9.7, we can state that the achieved results have been very good, obtaining acceptable degrees of certainty, specially because this is an initial step where there are a lot of possible improvements. These come from reorganizations of the number of samples used in each set of faults and standardization of the sizes of their samples, passing trough the inclusions of other types of feature to be computed and included in the feature vector, other presentation of it to the classifiers (not only histograms) up to more classifiers then only SVM to be included, and even other "metrics" to be tested. All these facts allow the say that NIR presents great possibilities for inspections in this type of continuous classification problem.

Acknowledgements We acknowledge to the CYTED Network "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559) C.A. and C.A.A. acknowledge projects 194810GI/VC and project Fondef ID21I10256. A.C. express their gratitude to the Brazilian Agencies FAPERJ, CAPES and CNPq.

References

1. Aguilera, C.A., Aguilera, C., Sappa, A.D.: Melamine faced panels defect classification beyond the visible spectrum. Sensors **18**(11), 3644 (2018)
2. Bigdeli, S., Süsstrunk, S.: Deep semantic segmentation using NIR as extra physical information. In: 2019 IEEE International Conference on Image Processing (ICIP), pp. 2439–2443. IEEE (2019)
3. Brahnam, S., Jain, L.C., Nanni, L., Lumini, A., et al.: Local Binary Patterns: New Variants and Applications, vol. 506. Springer (2014)
4. Conci, A., Proença, C.B.: A comparison between image-processing approaches to textile inspection. J. Text. Inst. **91**(2), 317–323 (2000)
5. Conci, A., Stephenson, S.L., Galvão, S.S., Sequeiros, G.O., Saade, D.C., MacHenry, T.: A new measure for comparing biomedical regions of interest in segmentation of digital images. Discret. Appl. Math. **197**, 103–113 (2015). Distance Geometry and Applications
6. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
7. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
8. Hamdi, A.A., Fouad, M.M., Sayed, M.S., Hadhoud, M.M.: Patterned fabric defect detection system using near infrared imaging. In: 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), pp. 111–117. IEEE (2017)
9. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)
10. Kubrusly, C.S.: The Elements of Operator Theory. Birkhauser, 2nd edn (2010)
11. Lee, J., Park, Y., Jeon, B.: Low intensity RGB texture enhancement based on near infrared image using perceptual information. In: 2018 10th International Conference on Communications, Circuits and Systems (ICCCAS), pp. 422–425. IEEE (2018)
12. Liu, Li., Zhao, Lingjun, Long, Yunli, Kuang, Gangyao, Fieguth, Paul: Extended local binary patterns for texture classification. Image Vis. Comput. **30**(2), 86–99 (2012)

13. Mahram, A., Shayesteh, M.G., Jafarpour, S.: Classification of wood surface defects with hybrid usage of statistical and textural features. In: 2012 35th International Conference on Telecommunications and Signal Processing (TSP), pp. 749–752. IEEE (2012)
14. Nurthohari, Z., Murti, M.A., Setianingsih, C.: Wood quality classification based on texture and fiber pattern recognition using hog feature and SVM classifier. In: 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), pp. 123–128. IEEE (2019)
15. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution grayscale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 971987 (2002)
16. Nurthohari, Z., Murti, M.A., Setianingsih, C.: Defect detection of particleboards by visual analysis and machine learning. In: 2019 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST), pp. 1–4. IEEE (2019)
17. Rodrigues, E.O., Conci, A., Liatsis, P.: Element: multi-modal retinal vessel segmentation based on a coupled region growing and machine learning approach. IEEE J. Biomed. Health Inform. **24**(12), 3519–3597 (2020)
18. Salamati, N., Larlus, D., Csurka, G., Süsstrunk, S.:. Semantic image segmentation using visible and near-infrared channels. In: European Conference on Computer Vision, pp. 461–471. Springer (2012)
19. Vivek Sharma, Jon Yngve Hardeberg, and Sony George. Rgb-nir image enhancement by fusing bilateral and weighted least squares filters. *Journal of Imaging Science and Technology*, 61(4):40409–1, 2017
20. Tamura, Hideyuki, Mori, Shunji, Yamawaki, Takashi: Textural features corresponding to visual perception. IEEE Trans. Syst. Man Cybern. **8**(6), 460–473 (1978)
21. Tan, X., Triggs, B.: Enhanced local texture feature sets for face recognition under difficult lighting conditions. IEEE Trans. Image Process. **19**, 16351650 (2010)
22. Vapnik, V.N.: The nature of statistical learning. Theory (1995)
23. Weston, J., Watkins, C.: Multi-class support vector machines. Technical report, Citeseer (1998)
24. Yean, Jeong-Seung., Kim, Gyung-Bum.: Investigation of laser scattering pattern and defect detection based on Rayleigh criterion for crystalline silicon wafer used in solar cell. J. Korean Soc. Precis. Eng. **28**(5), 606–613 (2011)
25. YongHua, Xie, Jin-Cong, Wang: Study on the identification of the wood surface defects based on texture features. Opt.-Int. J. Light. Electron Opt. **126**(19), 2231–2235 (2015)
26. Yuce, B., Mastrocinque, E., Packianather, M.S., Pham, D., Lambiase, A., Fruggiero, F.: Neural network design and feature selection using principal component analysis and Taguchi method for identifying wood veneer defects. Prod. Manuf. Res. **2**(1), 291–308 (2014)
27. Zhang, X., Sim, T., Miao, X.: Enhancing photographs with near infra-red images. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)
28. Zhang, Z., Luo, C., Wu, H., Chen, Y., Wang, N., Song, C.: From individual to whole: reducing intra-class variance by feature aggregation. Int. J. Comput. Vis. (2022)

Chapter 10
Waste Classification with Small Datasets and Limited Resources

Victoria Ruiz, Ángel Sánchez, José F. Vélez, and Bogdan Raducanu

Abstract Automatic waste recycling has become a very important societal challenge nowadays, raising people's awareness for a cleaner environment and a more sustainable lifestyle. With the transition to Smart Cities, and thanks to advanced ICT solutions, this problem has received a new impulse. The waste recycling focus has shifted from general waste treating facilities to an individual responsibility, where each person should become aware of selective waste separation. The surge of the mobile devices, accompanied by a significant increase in computation power, has potentiated and facilitated this individual role. An automated image-based waste classification mechanism can help with a more efficient recycling and a reduction of contamination from residuals. Despite the good results achieved with the deep learning methodologies for this task, the Achille's heel is that they require large neural networks which need significant computational resources for training and therefore are not suitable for mobile devices. To circumvent this apparently intractable problem, we will rely on knowledge distillation in order to transfer the network's knowledge from a larger network (called 'teacher') to a smaller, more compact one, (referred as 'student') and thus making it possible the task of image classification on a device with limited resources. For evaluation, we considered as 'teachers' large architectures such as InceptionResNet or DenseNet and as 'students', several configurations of the MobileNets. We used the publicly available TrashNet dataset to demonstrate that the distillation process does not significantly affect system's performance (e.g. classification accuracy) of the student network.

V. Ruiz (✉) · Á. Sánchez · J. F. Vélez
ETSII—URJC, 28933 Móstoles (Madrid), Spain
e-mail: victoria.ruiz.parrado@urjc.es

Á. Sánchez
e-mail: angel.sanchez@urjc.es

J. F. Vélez
e-mail: jose.velez@urjc.es

B. Raducanu
CVC—UAB, 08193 Cerdanyola (Barcelona), Spain
e-mail: bogdan@cvc.uab.es

A. D. Sappa (ed.), *ICT Applications for Smart Cities*, Intelligent Systems Reference Library 224, https://doi.org/10.1007/978-3-031-06307-7_10

10.1 Introduction

The recently coined concept of Smart Cities refers to the use of ICT-based solutions to improve life quality in urban areas in several domains: waste management, safety, transportation, mobility, etc. Nowadays, selective waste classification for recycling (paper, glass, metal, plastic, etc.) has become a very important aspect of our everyday life. With the significant increase in large-scale production and consumption, waste recycling has become a huge societal challenge and therefore it is imperative to come up with new solutions to guarantee a sustainable economy and improve the quality of the living environment. According to a recent report released by the World Bank [16], in 2016, 242 million tons of plastic waste was generated worldwide, representing 12% of the total solid waste. The same report forecasts that the waste generation will drastically outpace population growth by more than double by 2050. Without any measures, this will affect not only living conditions, but also will have a negative impact on the people's health being the main cause for skin, respiratory and cardio-vascular system diseases.

Previous waste selection systems took place in recycling plants and were mainly based on manual operations which were not only expensive and inefficient, but also dangerous for human operators [29]. Therefore, the automatization of this process, which could guarantee a safe, clean and efficient waste classification process was an urgent necessity for local communities and city managers. In the recent years, with the development of robotics and artificial intelligence, computer-vision based automatic waste classification has become fundamental in waste recycling. In this direction, solutions proposed by companies such as Picvisa1 or Sadako Technologies2 are only a few examples.

Successful computer vision-based approaches (ranging from image segmentation, object detection, visual servoing) are largely possible due to recent progress in deep learning techniques and powerful computational resources (GPUs). In particular, methods based on convolutional neural networks (CNN) are responsible for the high accuracy on image classification task, achieving 95% in the case of ImageNet (while the human visual system reaches 99% [30]). However, in the case of waste classification, this value is lower. The limitation in exploiting the full power of deep learning architectures is mainly due to lack of large trash datasets. A variety of CNNs models have been proposed to tackle the problem of waste classification, including ResNext [35], Xception [3], Faster R-CNN [17], EfficientNet [21], YOLO [9, 20], etc. More recently, the focus have been shifted towards Transformers [34], an architecture employing a self-attention model which helps building distant dependencies in the pattern, a mechanism which CNNs is lacking. For this reason, Transformers are capable of extracting more powerful features than CNNs, with stronger discriminative properties. Recently, an approach for selective waste classification based on Transformers has been proposed in [13].

A drawback of the previous mentioned systems is that they are based on large deep network architectures which require significant computational resources, which are costly to deploy to recycling facilities. With the rise of mobile computing, the focus

has been shifted towards more compact solutions that require less computational resources and could be used on portable devices. The ubiquity of mobile phones and tablets determined a migration of waste classification applications from clusters with powerful GPUs to mobile devices. However, deploying deep learning models to mobile devices with limited computational capabilities is accompanied by some challenges, as identified by [5]. First challenge is represented by memory limitation. It is known that large networks (with a large number of parameters) usually guarantee a low generalization error. However, on a device with limited memory, the challenge is to fit a model with much less parameters (one or two orders of magnitude) without any significant loss in accuracy. The second challenge is represented by energy consumption. Backpropagation is one of the most used operations in the network training. Due to its iterative character it is very time-consuming. On the other hand, continuous weights update is a computationally intensive operation. Thus, by combining time and computation complexity we come up with a serious energy challenge that we need to overcome. So far, researchers have paid more attention to optimizing the training and inference errors, rather than addressing the computational cost.

In this paper, in order to address the aforementioned challenges, we propose a computational efficient method to deploy deep learning models to mobile devices. We use network distillation as a strategy for knowledge transfer between a large deep learning model and a compact model, suitable for mobile devices. Thus, network distillation could take place on a standard PC with sufficient memory and computation resources, and at the end of the process, the resulted compact model could be easily deployed to a mobile device. As a case study, we will refer to the aforementioned problem of automatic waste classification. This way, we achieve a trade-off between computational complexity and accuracy.

The paper is structured as follows. In the next section, we present the related work in automatic waste classification as well as in network distillation. In section 3 we review the network distillation fundamentals. In section 4 we present our validation protocol and experimental results. Finally, in section 5 we draw our conclusions and present the guidelines for future work.

10.2 Related Work

The first part of this section reviews recent work on trash classification based on deep learning models in general, and lightweight architectures in particular. The second part reviews recent works on the network distillation process, as a methodology for knowledge transfer from large deep models to more compact ones.

10.2.1 Waste Recycling

Nowadays, many waste classification systems have been proposed. Most of them are focused on both detection and classification of garbage. Moreover, some recent works are also focused on the usage of lightweight architectures for mobile devices with limited resources. Next, we summarize the most related works with ours.

Concerning large deep architectures, De Cariolis et al. [8] trained an improved YOLOv3 neural network for waste detection and recognition. The authors created their own dataset by downloading Google images. Vo et al. [30] proposed a model called Deep Neural Networks for Trash Classification (DNN-TC), which is an improvement of ResNext model [26]. The authors achieved a 94% of accuracy on TrashNet dataset. Kulkarni and Sundara [15] proposed a hybrid transfer learning and faster R-CNN for region proposal for object location and classification. To train the model, the authors trained GANs models to create collages of images from TrashNet dataset. Zhang et al. [40] developed an attention-based branch expansion network whose objective was to distinguish confounding features. The authors achieved 94.53% on TrashNet dataset. Shi et al. [27] proposed a neural network architecture based on channel expansions rather than short-circuit connections.

Liu et al. [18] used YOLOv2 networks with box dimension clustering and pretrained weights in order to improve system's performance for trash detection and recognition. Moreover, the feature extractor part was replaced by the lightweight network MobileNet, and therefore, the final network can be used on low resources devices. Aral et al. [3] trained several deep learning models for classification. Some of them, such as MobileNet and Xception, have a lower number of parameters and can be considered lightweight models. Yang and Li [36] designed WasNet, a lightweight neural network, based on convolutional layers and attention mechanism achieving a 96.10% of accuracy on TrashNet dataset. Bircanoglu et al. [4] developed the lightweight CNN RecycleNet. This network, which achieves a 81% of accuracy on TrashNet, alternates the skip connections in dense layers in order to optimized prediction times. Therefore they reduce the number of network's parameters from 7 millions to 3 millions. White et al. [33] also worked on convolutional neural networks for trash classification that can be deployed on low power devices. Specifically, the authors designed WasteNet using a pretrained DesNet architecture combined with hybrid transfer learning and data augmentation. Huang et al. [12] trained a CNN-based network with the self-attention mechanism called Vision Transformer. They achieved 96.98% of accuracy on TrashNet. The model can be used on portable devices. Huynh et al. [13] combined three CNN (Efficient-B0, Efficient-B1, ResNet101), and added images from an external dataset to include the organic category. They achieved 94.11% of accuracy. Wang et al. [32] tested several state-of-the-arts convolutional neural networks, and concluded that MobileNetv3, with an accuracy of 94.36% was one of the architectures with better performance. In addition, this neural network had the advantage of having a small memory overhead (49.5 MB) and fast computing speed (261.7 ms). Finally, Masand et al. [19] designed ScrapNet, an EfficientNet based architecture. To train Scrapnet, the authors

developed a new dataset, as the result of combining four different datasets. To test the results over the TrashNet dataset, a transfer learning process was performed. As a result, the authors achieved 98% of accuracy on the TrashNet dataset.

10.2.2 Network Distillation

Deploying directly large models to mobile devices do not meet the energy efficiency and computational complexity requirements. To cope with this limitation, network distillation has been proposed as an alternative to perform knowledge transfer from a large network, referred as 'teacher', to a more compact network, referred as 'student' [10, 23]. From this point of view, network distillation could also be seen as a form of model compression. The concept behind network distillation is that the student model replicates teacher model in order to obtain similar performance. A network distillation system consists of three elements: knowledge, distillation algorithm and teacher-student architecture.

When we talk about network distillation, we could consider three categories of knowledge transfer: logits [20], activations or features of intermediate layers [1, 37] and relationships between them [29]. In the first category, the 'student' model should replicate the output of the last layer from the 'teacher' model. This is a simple, yet effective, knowledge distillation strategy and it has been used in several applications such as object detection [6] and semantic landmark detection for human pose estimation [39]. One drawback of this strategy is represented by the fact that is limited to supervised learning, since logits represent class probability distribution. In the second category, the output of the intermediate layers is considered to supervise the training of the 'student' model. This strategy is especially useful in the case of thinner and deeper networks. It was first proposed in [23], whose idea was to match the feature activations of the 'teacher' and the 'student'. However, choosing which layers from the 'teacher' model could be used as 'hints' and which layers from the 'student' model to be guided is still an open question. There were some attempts to address this problem by considering an attention mechanism in order to match the features indirectly [5, 37]. Finally, the last category exploits the relationship between different layers or samples. In [17], they proposed to use single value decomposition in order to extract relevant information from the correlations between feature maps. Another approach considers knowledge distillation from two 'teachers' [38]. They formed two graphs whose nodes represent the logits and the feature maps of each 'teacher' model. On a different direction, which takes into account the relationship between samples, [22] proposes to transfer the mutual relations of the data, expressed in terms of distance-wise and angle-wise information. A similar approach has been pursued in [5], where the 'student' network is learned by feature embedding, trying to preserve the feature similarities of the samples from the 'teacher' network.

Some work has also been devoted to better understand the underlying mechanism behind knowledge distillation by quantifying the extracted visual concepts from the intermediate layers [7]. On the other hand, [14] explained knowledge distillation

from the perspective of risk bound, data efficiency and imperfect teacher. Recently, besides image classification, knowledge distillation methods have been applied to a wide variety of applications, such as: adversarial attacks [21], data augmentation [16], data privacy and security [31] and person re-identification [24].

10.3 Reviewing Network Distillation

In Sect. 10.2 it is shown that, to develop a deep learning model for trash classification, most related works train large neural network architectures. Other related works have also dealt with the limited resource problem and trained lightweight neural network architectures. In this context, the architectures trained upon could be both, standard architectures as MobileNet, Xception, or ShuffleNet, or custom designed lightweights architectures. In both cases, the pre-trained weights from ImageNet were used.

However, other methods to transfer knowledge have been developed. One of the most efficient ones is the distillation method. The distillation method transfers the knowledge of a complex pre-trained network (called teacher) to a lightweight network (called student). Specifically, this process adds to the targets provided from the ground truth (called hard targets) the outputs of the teacher network (called soft targets) [10]. Therefore, the information provided by the teacher network will be better as more entropy would have the model. For example, if the teacher network has low entropy, its outputs will follow a Dirac distribution. That is, the correct class will have a probability near 1, and the rest of the classes, a probability near 0. However, if the teacher network has high entropy, the probabilities of the network will be more similar and less information will be provided. Therefore, the higher entropy would have the teacher model, the higher information will be provided.

To increase a model entropy Hinton et al. [10] propose divide the output logits (z_i) by a constant called *temperature* (T). Then, these soft logits are transformed in a probability distribution through a softmax transformation as usual. Note that in the case of hard outputs, the temperature is $T = 1$, and as a result, the probability is not altered. Formally, let be z_i the logit of the model i, and T constant which represent the temperature, then we define the soft probabilities from the model i with a temperature T, $p_i(T)$ as:

$$p_i(T) = \frac{e^{\frac{z_i}{T}}}{\sum_i e^{\frac{z_i}{T}}} \tag{10.1}$$

Next, to transfer the knowledge the loss function is redefined as the sum of a soft term and a hard term. The soft term will add the knowledge of the teacher network, while the hard term will add the knowledge of the student network that has learned so far. Specifically, the soft term will compare the outputs of the teacher network with the output of the student network. As it is mentioned before, the entropy of the

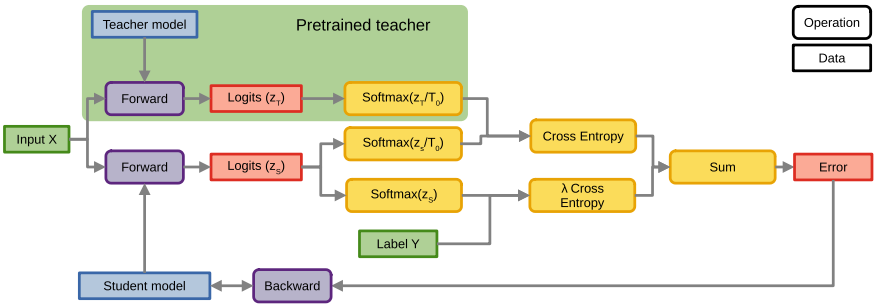

Fig. 10.1 Destillation process flow

teacher architecture will be increased by recalculating the probability of the output with the constant of temperature. To achieve comparable results, the output of the student network will be also recalculated by the same temperature. On the other hand, the hard term will compare the ground truth with the outputs of the student network as a classic classification model. As a result, no temperature correction will be made in this term.

Formally, let be $H(x, y)$ the categorical cross entropy function between the probabilities distributions x and y. We denote $L_{student}$ the loss function, L_{soft} the loss associated to the soft term, L_{hard} the loss associated to the hard term, and y the ground truth. Then, these functions are define as:

$$L_{soft} = H(p_{teacher}(T = T_0), p_{student}(T = T_0)) \tag{10.2}$$

$$L_{hard} = H(y, p_{student}(T = 1)) \tag{10.3}$$

$$L_{student} = L_{soft} + \lambda L_{hard} \tag{10.4}$$

where $\lambda \in (0, 1)$ is a regularization parameter. Figure 10.1 shows this methodology.

10.4 Proposed Validation Protocol and Experimental Results

This section details our evaluation protocol and experimental results. Specifically, the dataset used and the preprocessing process are presented in the first place. Then, the evaluation and the implementation details are fixed. Finally, the results are presented and discussed.

10.4.1 The TrashNet Dataset

The TrashNet dataset [35] consists of a collection of 2,527 RGB images divided in six classes of waste: 'glass', 'paper', 'cardboard', 'plastic', 'metal', and 'general trash', respectively. The image distribution per class is the following: 501 for the 'glass' category, 594 for 'paper', 403 for 'cardboard', 482 for 'plastic', 410 of 'metal', and 137 for 'general trash', respectively. The images were collected by placing the object in front of a white background and using sunlight and/or ambient lighting. Figure 10.2 illustrates the six classes that constitute the TrashNet dataset.

10.4.2 Data Preprocessing

To develop a deep learning model which classifies the waste images, all the images have been resized down to a spatial resolution of 197×283 pixels, in order to overcome the computational resources of our GPU. Moreover, for training the neural networks models, the pixel values of the images have been normalized into the range between 0 and 1.

Since the TrashNet dataset consists of a small number of images to train our models, and deep learning neural networks require large datasets, the common practice of data augmentation has been applied in order to increase the number of images. In consequence, since the new data is generated during the training process, an unlimited number of augmented samples is possible to obtain.

Specifically, the new images were generated through a random subset of the following transformations:

- Rotation between 0 and 40° (see Fig. 10.3a).
- Width changes between 0 and 20% (see Fig. 10.3b).
- Height changes between 0 and 20% (see Fig. 10.3b).
- Shear between 0 and 20% (see Fig. 10.3c).
- Horizontal flip (see Fig. 10.3d).
- Translation between 0 and 12.5% over horizontal axis (see Fig. 10.3e).
- Cutout: masking with a random square of half image size (see Fig. 10.3f).
- Brightness changes between –0.5 and 0.5 (see Fig. 10.3g).
- Contrast changes between 0.5 and 1.5 (see Fig. 10.3h).
- Saturation changes between 0 and 2 (see Fig. 10.3i).

10.4.3 Evaluation

To evaluate the models, some of the classic metrics in classification problems have been used: accuracy and confusion matrix.

Fig. 10.2 **a** Plastic; **b** Metal; **c** Cardboard; **d** Paper; **e** Glass; **f** Trash

On one hand, the accuracy measures the global goodness of the model. This metric is defined as the percentage of images correctly classified vs the total number of images.

On the other hand, it is also important to know when the classification model is performing well and what types of errors is doing. The confusion matrix gives us all this information. Formally, it is a matrix where each row represents the actual classes and each column the predicted classes. Therefore, each cell represents the

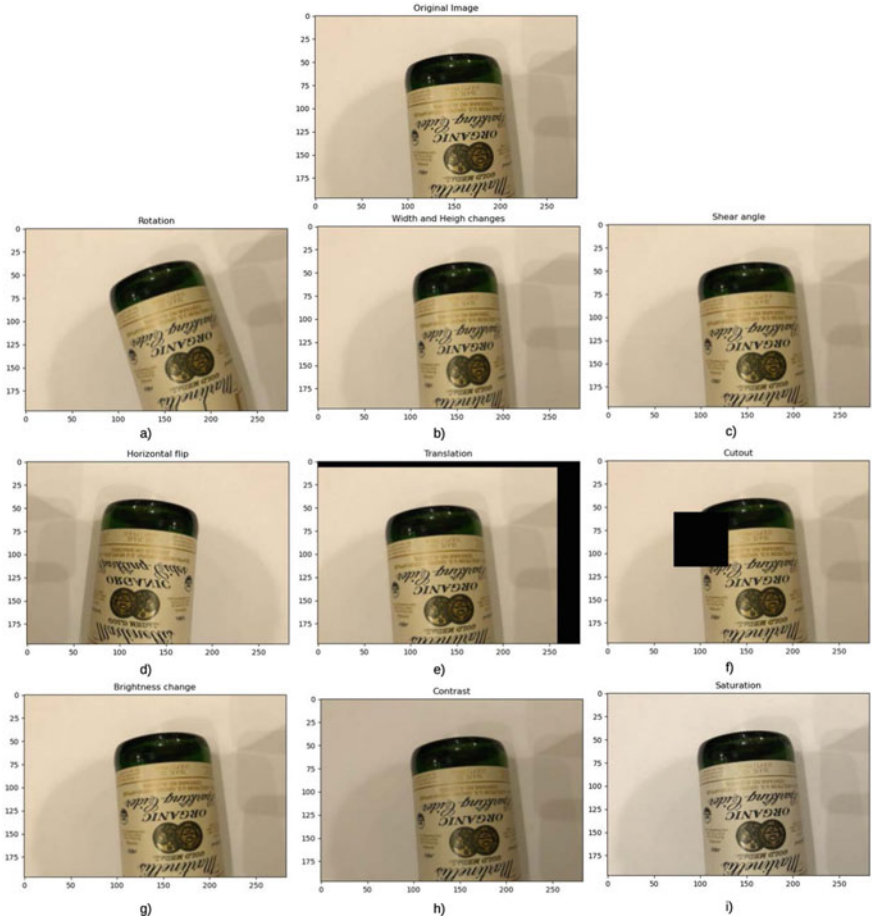

Fig. 10.3 Examples of transformations for data augmentation

percentage of instances of a specific class predicted as another specific class. The diagonal of the matrix represents, as a result, the accuracy of each class.

10.4.4 Implementation Details

This work is the continuation of our paper presented at IWINAC 2019 [25]. In that work, several large neural networks architectures were trained to classify the six classes of TrashNet. Now, the objective of the current work is to transfer the knowledge from large models to lightweight ones which are more suitable to be deployed to mobile devices. One of the models with the best results was an InceptionResNet

architecture. Therefore, two teacher models have been considered for the distillation process: InceptionResNetV2 [28], and DenseNet201 [11].

InceptionResnetV2 is a neural network architecture based on inception modules and residual connections. On one hand, inception modules concatenate several branches of stacked convolutional layers in order to achieve a multiresolution analysis. On the other hand, residual connections connect a layer with the previous one in order to achieve a denser architecture. The other teacher model considered is DenseNet201, a CNN-based architecture where each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. As a result, a more 'knowledgeable' neural network with better performance is achieved.

As student models, two lightweight architectures have been considered: MobileNet [2], and ShuffleNet [34]. MobileNets architectures consists of depth-wise separable convolutions. Depth-wise convolutions stack a $K_1 \times K_2$ convolution layer with a 1×1 convolution layer (called pointwise convolution). These types of layers reduce significantly the complexity of the neural network without degrading its performance. In MobileNet, the width of the architecture (i.e., the number of input channels in each layer) can be controlled by the α multiplier. As a result, applying this coefficient, the number of input channels N becomes $\alpha \cdot N$. In order to distinguish the width of the network, this architecture is called MobileNet-α. In this work, we consider the architectures MobileNet-1 and MobileNet-0.25. ShuffleNet architecture is based on two new operations, pointwise group convolutions and channel shuffle operations. Specifically, pointwise group convolutions are proposed to reduce computation complexity of 1×1 convolutions. To overcome the side effects brought by group convolutions, the channel shuffle operation helps the information flow across feature channels.

Table 10.1 compiles the number of parameters of each architecture. Focusing on teacher architectures, it can be appreciated that InceptionResNetV2 have more than double of parameters than DenseNet201. Concerning the student architectures, the three considered models have a wide range of parameters between 470,000 and 4,000,000 of parameters. Therefore, in any combination, the number of parameters of the student architectures are at least five times lower than the teachers'. In the extreme case, where the InceptionResNetV2 model acts as a teacher, and the MobileNet-0.25 acts as a student, the number of parameters is reduced by more than 100 hundred times.

The two teacher models and the two MobileNet architectures have been initialized using pre-trained weights from ImageNet [9], while ShuffleNet was initialized from scratch, as we could not find any pretrained ShuffleNet model in our framework[1]. For training all these models we fixed a learning rate of 0.0002 and a batchsize of 16 samples. Also, the model is trained with the stochastic gradient descent (SGD) as an optimizer during 1000 epoch. To avoid overfitting, an early stopping technique was applied. As a result, the training process is stopped if the validation loss does not improve during 50 epochs. The GPU used was a GPU 2 RTX 5000 with 16 GB of memory.

[1] A pre-trained ShuffleNet model would have led to improved results.

Table 10.1 Neural networks architectures complexity

Model	Num. trainable parameters
DenseNet201 [11]	20,013,928
InceptionResNetV2 [28]	55,813,192
MobileNet-1 [2]	4,231,976
MobileNet-0.25 [2]	470,072
ShuffleNet [34]	1,531,888

Table 10.2 Experiments description

Experiment name	Teacher model	Student model	Num. train parameters
D-M-1	DenseNet201	MobileNet-1	3,213,126
D-M-0.25		MobileNet-0.25	214,614
IR-M-1	InceptionResNetV2	MobileNet-1	3,213,126
IR-M-0.25	InceptionResNetV2	MobileNet-0.25	214,614
D-S	DenseNet201	ShuffleNet	958,350

To analyze the knowledge distillation process, the specific teacher and student combinations are detailed in Table 10.2. It can be appreciated that there is a small difference between the number of trainable parameters of the student architectures shown in Table 10.1, and those parameters shown in Table 10.2, although the architectures are the same. This difference is due to neurons of the last dense layer, which originally were 1000, and now they are substituted by 6 neurons, in order to adapt the network to the TrashNet dataset.

In order to study the effect of the temperature T and λ parameters, on one hand, the temperature range was fixed between 1 and 30. Specifically, we considered the following values for T: 1, 2, 5, 10, 20, and 30. On the other hand, the following λ values were considered: 1, 0.1, 0.01, and 0.001. These coefficients have been chosen during the training process, analyzing the contribution of the soft and the hard losses.

10.4.5 Results

Finally, the achieved results are presented. Firstly, the performance of the teacher models is detailed. Next, the performance of the students' models is discussed, and the influence of the temperature (T) and the regularization term (λ) is analyzed. Finally, a brief comparison with the state of the art is made.

Table 10.3 Performance of teacher models

Model	Accuracy (%)
DenseNet201	96.75
InceptionResNetV2	95.90

Table 10.4 Performance of distilled models

Experiment name	T	λ	Accuracy (%)
D-M-1	5	1	93.67
D-M-0.25	2	1	91.69
IR-M-1	2	1	92.88
IR-M-0.25	1	1	90.11
D-S	2	1	83.00

Teacher models performance analysis
Prior the knowledge distillation process, the teacher models were trained first. Table 10.3 shows the performance of these models. It could be appreciated that with the DenseNet201 architecture, the best result was achieved with a 96.75% of accuracy. On the other hand, InceptionResNetV2 performance is close to the previous one, as this model achieves 95.90% of accuracy.

Student models performance analysis
Once the teachers' models have been learnt, the students models were trained with the knowledge distillation process. Table 10.4 shows the best performance for each experiment mentioned previously. First of all, in all the models, the best results were achieved for $\lambda = 1$. As a result, the hard and soft losses have the same weight during training. Secondly, the best results have been achieved for the combination of architectures DenseNet201 and MobileNet-1, with a 93.97% of accuracy. We have noticed that by, reducing the number of parameters 5 times approximately, the accuracy has decreased by 3.18% only. In the extreme case of the experiment, IR-M-0.25, corresponding to a reduction of more than 100 times of the number of parameters of the student model compared with the teacher model, a 6.86% decrease in accuracy is obtained.

Moreover, it could be appreciated from the experiments that for those cases where a better teacher model has been used tend to obtain better performance. Indeed, D-M-1 has better accuracy than IR-M-1, and D-M-0.25 has better accuracy than IR-M-0.25. Furthermore, focusing on the number of parameters, Table 10.4 shows that the more parameters the student architecture has, a higher accuracy is achieved. Indeed, MobileNet-1 gets higher accuracy than MobileNet-0.25. Concerning the D-S experiment, which corresponds to the combination between DenseNet201 and ShuffleNet architectures, Table 10.4 shows that this experiment gets a significant lower accuracy. However, we would like to recall that the ShuffleNet model was trained from scratch.

Table 10.5 Temperatures analysis for D-M-1 experiment with $\lambda = 1$

Experiment	$T = 1$	$T = 2$	$T = 5$	$T = 10$	$T = 20$	$T = 30$
D-M-1	91.69%	92.88%	93.67%	89.32%	92.88%	92.09

Analyzing the confusion matrix, Fig. 10.4 shows which classes are mistakenly classified by other classes. For all the trained models, the class with less accuracy is '*trash*'. As this category corresponds to all the waste which is not 'glass', 'paper', 'cardboard', 'plastic' or 'metal', it is the most challenging one and it explains its lower performance. Focusing on D-M-1 and IR-M-1 experiments, 'plastic' is the second category with lower results. This class is also the second one with more confusion for D-M-0.25, IR-M-0.25 and D-S experiments. Indeed, in these confusion matrix, 'plastic' is often confused by 'metal' and 'paper'.

Performance analysis of T and λ parameters

As it was mentioned in Sect. 10.4.4, the 5 experiments were performed with different values for the temperature T and the regularization parameter λ, in order to assess their influence in the distillation process.

Concerning the performance of the T parameter, Table 10.5 shows that best performance is achieved for low temperature values (between 1 and 30). However, the difference between these temperatures, and the higher ones, are not very significant, as is shown in Table 10.5. Indeed, for temperatures of $T = 2$ and $T = 20$ we obtained the same performance. As a result, we conclude that for the range of temperatures analyzed, this parameter does not affect the performance.

Also, for the selected range of temperatures used in the distillation process, we analyzed the difference between the probabilities assigned to the correct class and the probabilities of the incorrect classes for all the images in the test dataset. Fig. 10.5 shows an example of the evolution of the soft probability distribution over a test image when the temperature is increased. In particular, these probabilities are achieved when an image belonging to the 'glass' category is predicted. As it is shown in the Fig. 10.5, in the first plot (upper left), which corresponds to $T = 1$, the 'glass' class (which is the correct one) has a probability near to 1, and the other classes (the incorrect ones) have probabilities near to 0. In the other extreme, the plot at bottom right, which corresponds to $T = 30$, shows that the correct class glass has a probability near to 0.275, while the incorrect classes have probabilities values between 0,125 and 0.175. Therefore, applying a high temperature coefficient has as result a reduction in the difference between the probability of the correct class, and the probabilities of the wrong classes. In between plots depict the results for the remaining values of the T parameter (from left to right and top to bottom).

Regarding the regularization parameter λ, Table 10.6 shows the performance on the D-M-1 experiment with $T = 5$ when the regularization parameter is modified. It is shown that for higher λ, higher accuracy is achieved.

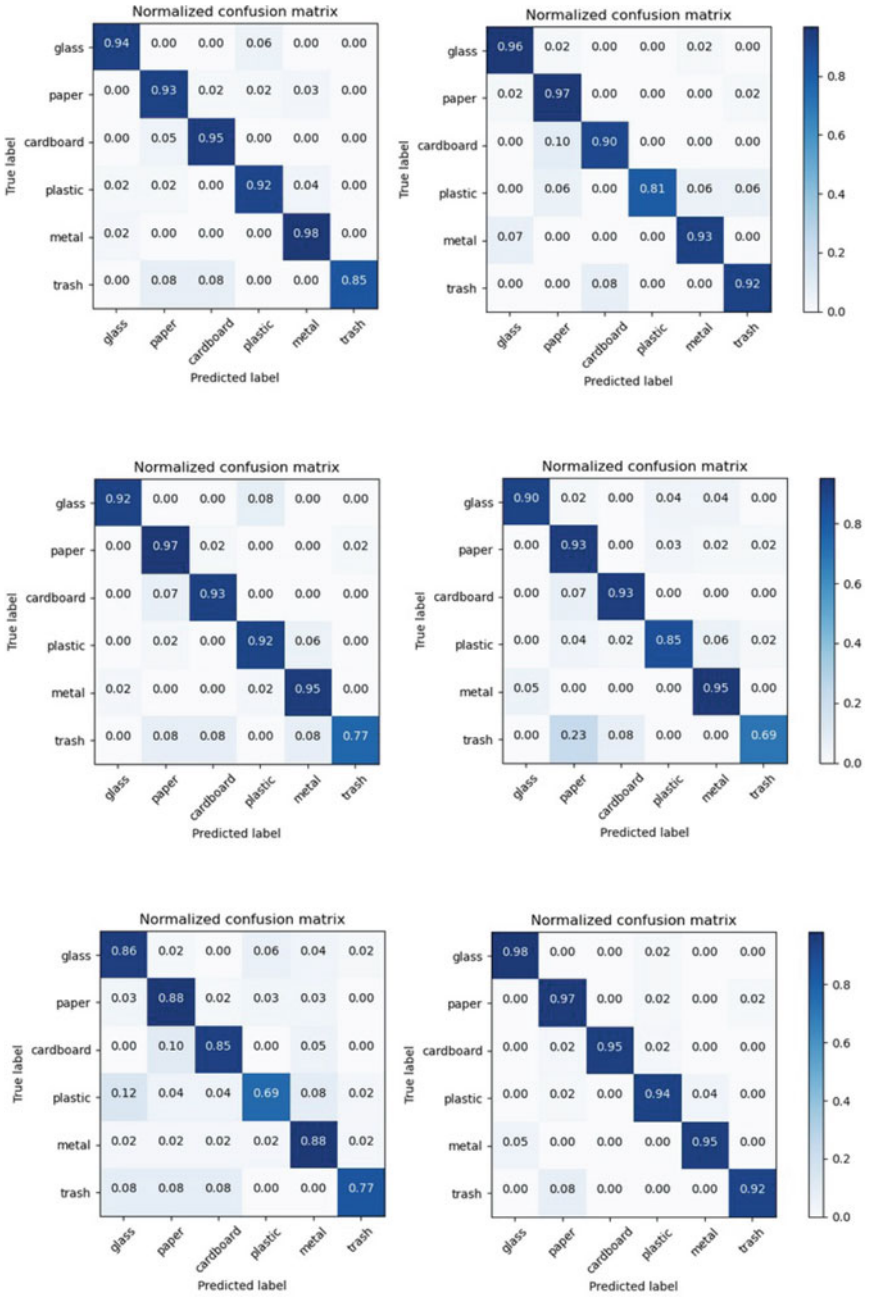

Fig. 10.4 Confusion matrix comparative

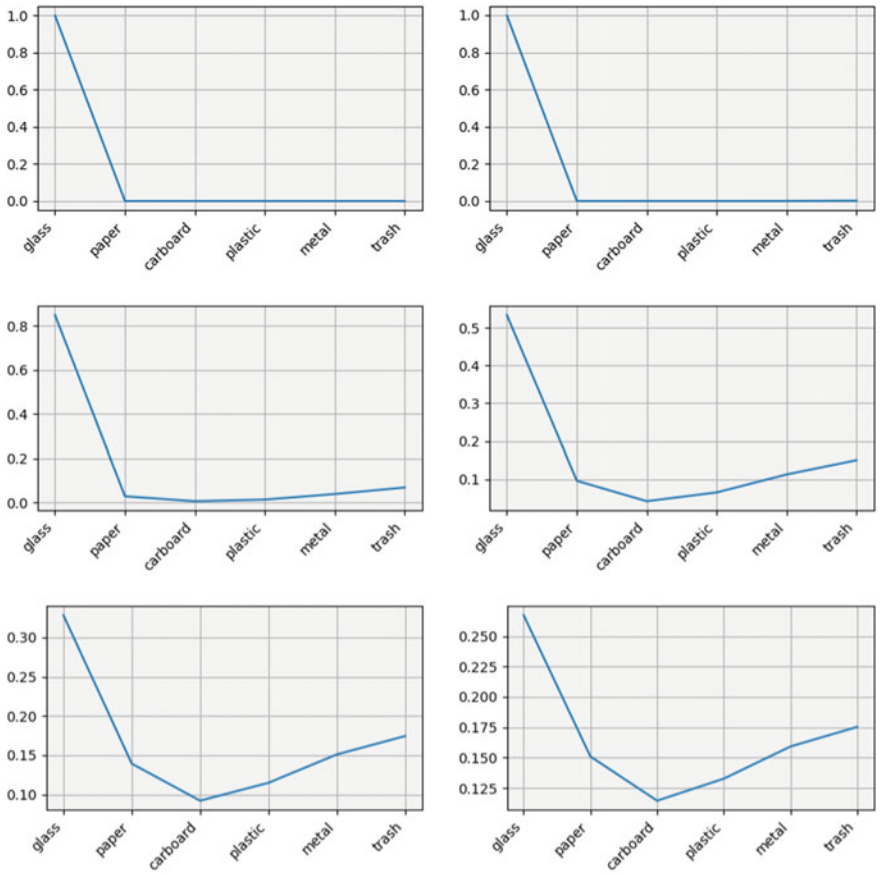

Fig. 10.5 Original and softened probability distributions generated by the teacher network for temperature values of (from left to right and top to bottom) $T = 1, 2, 5, 10, 20, 30$

Table 10.6 λ analysis for D-M-1 experiment with $T = 5$

Experiment	T	$\lambda = 1$	$\lambda = 0.1$	$\lambda = 0.01$	$\lambda = 0.001$	$\lambda = 0.0001$
D-M-1	5	93.67%	90.90%	87.35%	68.37%	39.13%

Comparison with state of the art methods

Finally, a brief comparison with the state of the art methods is provided. Table 10.7 shows recent works related to lightweight architectures for trash waste classification. All the presented results are reported on the TrashNet datasets except for two which also include a custom dataset. Although the comparison is not easy in many cases as they use other datasets, it is shown that our results are better than some works as [4, 18], and they are close to [13, 32]. Moreover, the performance is significantly better than [3], where the same MobileNet architecture has been used.

Table 10.7 Different state of the art systems for waste classification for mobile devices with limited resources. *Result obtained using a large network (teacher). This is shown for comparison purposes only

Author (Year)	Dataset	Architecture	Accuracy (%)
Liu et al. (2018) [18]	Own dataset (1 cat.)	YOLOv2+ MobileNet	89.1
Bircanoglu et al. (2018) [4]	TrashNet	RecycleNet	81
Aral et al. (2019) [3]	TrashNet (6 cat.)	MobileNet	84
Yang and Li (2020) [36]	TrashNet	WasNet	96.10
White et al. (2020) [33]	TrashNet	WasteNet	97
Huynh et al. (2020) [13]	TrashNet + own dataset (7 cat)	Efficient-B0+ Efficient-B1+ResNet101	94.11
Huang et al. (2021) [12]	TrashNet	CNN+Vision Trasnformer	96.98
Wang et al. (2021) [32]	TrashNet + others (9 cat)	MobileNetV3	94.26
Masand et al. (2021) [19]	TrashNet + Openrecycle + TACO + Waste Classification	ScrapNet	98
*Our solution	TrashNet	DenseNet201	96.75
Our solution	**TrashNet**	**D-M-1**	**93.67**

10.5 Conclusions and Future Work

Selective waste classification for recycling has become a very important aspect of our everyday life. In this paper, we addressed this problem by proposing a computer vision-based solution suitable for mobile devices. For this purpose, we used network distillation as a strategy for knowledge transfer from a large deep learning model to a more compact one, suitable for mobile devices. Thus, network distillation could take place on a standard PC with sufficient memory and computation resources, and at the end of the process, the resulting lightweight model could be easily deployed to a mobile device. Extensive experiments run on the TrashNet dataset, with a variety of 'teacher' and 'student' models, demonstrated that the results obtained with the lightweight models through the distillation process are computationally efficient, while maintaining a competitive accuracy level compared to large models and the current state-of-the art.

Future work will be devoted to studying new data augmentation strategies in order to enhance the TrashNet dataset. Additionally, we will also consider to study the problem of domain adaptation, as an alternative knowledge transfer strategy, when new datasets become publicly available.

Acknowledgements The authors gratefully acknowledge the financial support of the CYTED Network entitled: "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559). We also acknowledge to the Spanish Ministry of Science and Innovation Project RTI2018-098019-B-I00.

References

1. Ahn, S., Hu, S. X., Damianou, A., Lawrence, N. D., Dai, Z.: Variational information distillation for knowledge transfer. In: Proceedings of CVPR, pp. 9163–9171 (2019)
2. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Adam, H.: Mobilenets: efficient convolutional neural networks for mobile vision applications (2017) arxiv:abs/1704.04861
3. Aral, R.A., Keskin, S.R., Kaya, M., Haciömeroğlu, M.: Classification of TrashNet dataset based on deep learning models. In: Proceedings—2018 IEEE International Conference on Big Data, Big Data 2018, pp. 2058–2062 (2019)
4. Bircanoglu, C., Atay, M., Beser, F., Genç, O., Kizrak, M.A.: Recyclenet: intelligent waste sorting using deep neural networks. In: Conference: 2018 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA) (2019)
5. Chen, D., Mei, J.P., Zhang, Y., Wang, C., Wang, Z., Feng, Y., Chen, Y.: Cross-layer distillation with semantic calibration. In: Proceedings of AAAI, pp. 7028–7036 (2021)
6. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge. In: Proceedings of NeurIPS (2017)
7. Cheng, X., Rao, Z., Chen, Y., Zhang, Q.: Explaining knowledge distillation by quantifying the knowledge. In: Proceedings of CVPR, pp. 12925–12935 (2020)
8. De Carolis, B., Ladogana, F., MacChiarulo, N.: Yolo TrashNet: garbage detection in video streams. In: IEEE Conference on Evolving and Adaptive Intelligent Systems (2020)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009, pp. 248–255. IEEE (2009)
10. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In Proceedings of NIPS (2014)
11. Huang, G., Li, Z., Van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2261–2269 (2016)
12. Huang, K., Lei, K.Q., Jiao, Z., Zhong, Z.: Recycling waste classification using vision transformer on portable device. Sustainability (Switzerland) **13**(21) (2021)
13. Huynh, M.H., Pham-Hoai, P.T., Tran, A.K., Nguyen, T.D.: Automated waste sorting using convolutional neural network. In: 2020 7th NAFOSTED Conference on Information and Computer Science (NICS), pp. 102–107 (2020)
14. Ji, G., Zhu, Z.: Knowledge distillation in wide neural networks: risk bound, data efficiency and imperfect teacher. In: Proceedings of NeurIPS (2020)
15. Kulkarni, H.N., Raman, N.K.S.: Waste object detection and classification. In: Technical Report. Stanford University (CA), USA (2019)
16. Lee, L., Hwang, S.J., Shin, J.: Self-supervised label augmentation via input transformations. In: Proceedings of ICML (2020)
17. Lee, S.H., Kim, D.H., Song, B.C.: Self-supervised knowledge distillation using singular value decomposition. In: Proceedings of ECCV (2018)
18. Liu, Y., Ge, Z., Lv, G., Wang, S.: Research on automatic garbage detection system based on deep learning and narrowband internet of things. In: Journal of Physics: Conference Series, vol. 1069 (2018)
19. Masand, A., Chauhan, S., Jangid, M., Kumar, R., Roy, S.: Scrapnet: an efficient approach to trash classification. IEEE Access **9**, 130947–130958 (2021)

20. Mirzadeh, S.I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., Ghasemzadeh, H.: Improved knowledge distillation via teacher assistant. In: Proceedings of AAAI, pp. 5191–5198 (2020)
21. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 582–597 (2016)
22. Park, W., Kim, D., Lu, Y., Cho, M.: Relational knowledge distillation. In: Proceedings of CVPR, pp. 3967–3976 (2019)
23. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: hints for thin deep nets. In: Proceedings of ICLR (2015)
24. Ruiz, I., Raducanu, B., Mehta, R., Amores, J.: Optimizing speed/accuracy trade-off for person re-identification via knowledge distillation. Eng. Appl. Artif. Intell. **87** (2020)
25. Ruiz, V., Sanchez, A., Velez, J.F., Raducanu, B.: Automatic image-based waste classification. In: Ferrandez Vicente, J.M., Álvarez-Sánchez, J.R., de la Paz López, F., Moreo, J.T., Adeli, H. (eds.) From Bioinspired Systems and Biomedical Applications to Machine Learning, pp. 422–431. Springer International Publishing (2019)
26. Saining, X., Ross, G., Piotr, D., Zhuowen, D., Kaiming, H.: Aggregated residual transformations for deep neural networks (2016). arxiv:161105431v2
27. Shi, C., Xia, R., Wang, L.: A novel multi-branch channel expansion network for garbage image classification. IEEE Access **8**, 154436–154452 (2020)
28. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17, p. 42784284 (2017)
29. Tung, F., Mori, G.: Similarity-preserving knowledge distillation. In: Proceedings of ICCV, pp. 1365–1374 (2019)
30. Vo, A.H., Son, L.H., Vo, M.T., Le, T.: A novel framework for trash classification using deep transfer learning. IEEE Access **7**, 178631–178639 (2019)
31. Wang, C., Lan, C., Zhang, Y.: Model distillation with knowledge transfer from face classification to alignment and verification (2017). arXiv:1709.02929
32. Wang, C., Qin, J., Qu, C., Ran, X., Liu, C., Chen, B.: A smart municipal waste management system based on deep-learning and internet of things. Waste Manage. **135**, 20–29 (2021)
33. White, G., Cabrera, C., Palade, C., Li, C., Clarke, S.: Wastenet: waste classification at the edge for smart bins (2020). arxiv:2006.05873
34. Xiangyu, Z., Xinyu, Z., Mengxiao, L., Jian, S.: Shufflenet: an extremely efficient convolutional neural network for mobile devices (2017). arxiv:1707.01083
35. Yang, M., Thung, G.: Classification of trash for recyclability status. CS229 Project Report (2016)
36. Yang, Z., Li, D.: Wasnet: a neural network-based garbage collection management system. IEEE Access **8**, 103984–103993 (2020)
37. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In: Proceedings of ICLR (2017)
38. Zhang, C., Peng, Y.: Better and faster: knowledge transfer from multiple self-supervised learning tasks via graph distillation for video classification. In: Proceedings of IJCAI, pp. 1135–1141 (2018)
39. Zhang, F., Zhu, X., Ye, M.: Fast human pose estimation. In: Proceedings of CVPR, pp. 3517–3526 (2019)
40. Zhang, Y., Sun, S., Liu, H., Lei, L., Liu, G., Lu, D.: Target state classification by attention-based branch expansion network. Appl. Sci. (Switzerland) **11**(21) (2021)