Computers & Graphics 37 (2013) 496–508

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on 3D Object Retrieval

Efficient 3D object recognition using foveated point clouds ☆

Rafael Beserra Gomes [a], Bruno Marques Ferreira da Silva [a], Lourena Karin de Medeiros Rocha [a,*], Rafael Vidal Aroca [a], Luiz Carlos Pacheco Rodrigues Velho [b], Luiz Marcos Garcia Gonçalves [a]

[a] Universidade Federal do Rio Grande do Norte (UFRN), Natal, Brazil
[b] Instituto Nacional de Matemática Pura e Aplicada (IMPA), Rio de Janeiro, Brazil

ARTICLE INFO

Article history:
Received 30 October 2012
Received in revised form
18 March 2013
Accepted 27 March 2013
Available online 1 May 2013

Keywords:
Point cloud
3D object recognition
Moving fovea

ABSTRACT

Recent hardware technologies have enabled acquisition of 3D point clouds from real world scenes in real time. A variety of interactive applications with the 3D world can be developed on top of this new technological scenario. However, a main problem that still remains is that most processing techniques for such 3D point clouds are computationally intensive, requiring optimized approaches to handle such images, especially when real time performance is required. As a possible solution, we propose the use of a 3D moving fovea based on a multiresolution technique that processes parts of the acquired scene using multiple levels of resolution. Such approach can be used to identify objects in point clouds with efficient timing. Experiments show that the use of the moving fovea shows a seven fold performance gain in processing time while keeping 91.6% of true recognition rate in comparison with state-of-the-art 3D object recognition methods.

1. Introduction

With current developments experienced in hardware technologies, computer vision systems would be ideally able to capture 3D data of the world and process this data in order to take advantage of their inherent depth information. However, nowadays, most current computer vision systems are still based on 2D images while the use of 3D data can offer more details about geometric and shape information of captured scenes and consequently, of general objects of interest. In this way, the development of 3D object recognition systems has been an active research topic over the last years [1].

Recent technology advances have enabled the construction of devices, as for example the Microsoft Kinect [2], that capture 3D data from the real world. The Kinect is a consumer grade RGB-D sensor originally developed for entertainment that has enabled several novel works for research including robotics, commercial, and gaming applications. Mobile phone manufactures have also started to shipping smartphones with stereo vision cameras in the recent years. Other manufacturers already announced camera sensors with depth information as a 4th channel. Furthermore, the price reduction of equipment is driving a wide adoption of 3D capture systems.

Although the amount of data provided by 3D point clouds is very attractive for object recognition, it requires intensive computing algorithms that could render systems based on this type of data computationally prohibitive, mainly if real time interaction is needed. Hardware accelerators and optimizations are frequently used for real time computing, however object recognition is still an open research field with several challenging research opportunities, especially when real time performance is desired. One software solution consists in processing point clouds efficiently using algorithms that compute local geometric traits. One example of such system is depicted in Section 4, which enumerates advantages of correspondence grouping algorithms.

We are interested on accelerating object retrieval using 3D perception tools and data acquisition from real images (not synthetic images). For this purpose, we propose the usage of a moving fovea approach to downsample 3D data and reduce the processing of the object retrieval system from point clouds. An example of foveated cloud can be seen in Fig. 1. Experimental results show that our system offers up to seven times faster recognition time computing without compromising recognition performance. We also provide two web based tools to interactively view and manipulate point clouds and to capture Kinect point clouds without the need to install any software, which has been used within the Collage Authoring System.

This article is structured as follows: Section 2 presents the theoretical background used in this work with reviews of some related works on 3D object retrieval and the moving fovea approach. Section 3 describes 3D moving fovea applied to the object recognition problem and its formulation in the context of our work. Section 4 depicts both the system that forms the base of our implementation and also the proposed scheme, along with implementation considerations. Section 5 describes the experiments, including a performance

☆To comment on this article, please join the discussion on the Collage Authoring Environment Google Group https://groups.google.com/group/collage-authoring-environment.

* Corresponding author. Tel.: +55 84 32153771.
E-mail address: lourena@gmail.com (L.K.d.M. Rocha).

0097-8493
http://dx.doi.org/10.1016/j.cag.2013.03.005

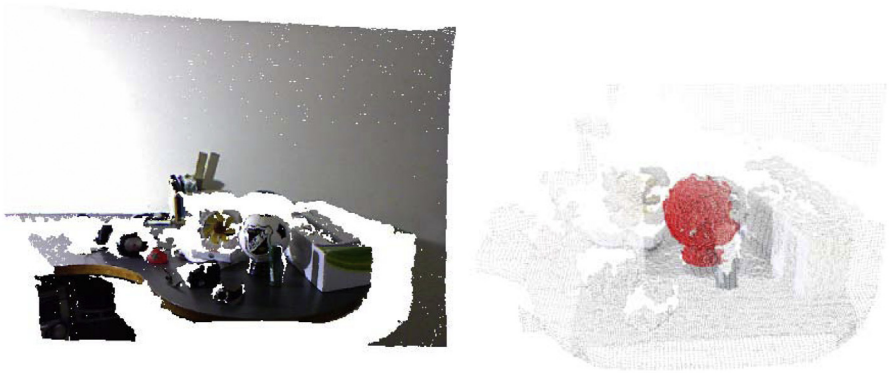

Fig. 1. Example of an original point cloud and object detection in the foveated cloud.

evaluation that can be executed with the Collage authoring environment, while Section 6 closes the article with our final remarks.

2. Theoretical background

Three-dimensional object recognition is a multi-disciplinary area of research, with major contributions originating from the Pattern Recognition, Computer Vision, Robotics and Computer Graphics communities. In this section, relevant contributions from each of these subareas are briefly enumerated, emphasizing the data acquisition method employed in each of them.

2.1. 3D multiresolution approaches

Vision is so far the most important sensing resource for robotics tasks that can be executed based on devices like web cameras and depth sensors. Unfortunately, the huge amount of data to be processed is limited by the processing time that is a restriction for doing reactive robotics. Several approaches represent image with non-uniform density using a foveated model that mimics the retina mapping to the visual cortex in order to deal with this amount of data [3–9]. The fovea is the area of retina with greatest visual acuity, so foveated models have high resolution nearby the fovea and decrease the resolution according to the distance from the fovea.

The foveation process is performed either by subsampling in software [3,4], by hardware with reduced sampling [10] or by using a system with 2 or more cameras, where one is used for peripheral vision and another one is used for foveated vision [11,12]. The software foveation allows greater ease of modification and easily implementable in conventional hardware, but is slower than hardware solutions which are usually more expensive and difficult to change. In terms of coverage, solutions that use specific cameras to peripheral and foveated vision are more open to stimuli of the whole environment by using a wide angle peripheral camera, what would require a huge resolution camera in the case of a single camera system due to high resolution fovea needs. However, a camera specific for foveated vision requires movement of physical devices and a large difference between peripheral and fovea cameras suppress stimuli appearing on a intermediate level of resolution because these are not in the fovea camera field of view neither in the peripheral camera. In this work, the foveation is performed by software. It is important to note that most of these models allow free movement of the fovea, what does not happen at the biological eye's retina. Otherwise, all the vision resources should be moved in order to keep the object at foveal region.

In a dynamic and cluttered world, all information needed to perform complex tasks are not completely available and not processed at once. Information gathered from a single eye fixation is not enough to complete these tasks. In this way, in order to efficiently and rapidly acquire visual information, our brain decides not only where we should look but also what is the sequence of fixations [13]. This sequence of fixations, and therefore the way the fovea is guided, is related to cognition mechanisms controlled by our visual attention mechanism. Several works propose saliency maps from which fixations can be extracted [14].

It is also known that the human vision system has two major visual attention behaviors or dichotomies [15]. In the top-down attention approach, the task in hand guides attention processing. On the other hand, in bottom-up attention, external stimuli drive attention. Text reading is an example of the top-down behavior of attention, where visual fixations are done systematically, passing through the paper in a character by character and line by line movement. On the opposite, if a ball is thrown toward the same reader, this bottom-up stimulus will make the reader to switch attention to the dangerous situation.

Besides in robotic vision, several foveated systems are proposed in order to reduce the amount of data to be coded/decoded also in real-time video transmission [8,6]. In this kind of application, an image should be encoded with foveation thus keeping higher resolution in regions of interests. In a similar way, Basu [16] proposes a foveated system to 3D visualization with limited bandwidth restriction, where the fovea position controls the objects' texture quality and resolution.

2.2. 3D object recognition

Early object recognition systems acquired data from expensive and rarely available range sensors, such as laser scanners [17,18] and structured light patterns [19]. Ashbrook et al. [17] describe an object recognition system that relies on similarities between geometric histograms extracted from the 3D data and the Hough Transform [20]. Johnson and Hebert popularized the Spin Images descriptor [18,19], which was used as the basis to an object recognition algorithm that groups correspondences of Spin Images extracted in a given query model and those extracted in the scene data that share a similar rigid transformation between the model and the scene [18]. Data from 3D scanners and also from synthetic CAD 3D models are employed in the work of Mian et al. [21].

Until recently, 3D object recognition systems processed data mostly in an off-line fashion, due to long computing times involved [22]. This paradigm has started to shift as algorithms have been proposed in the Robotics community [23,24] to enable real-time manipulation and grasping for robotic manipulators. In fact, algorithms designed to describe 3D surfaces through histograms of various local geometric traits evaluated on point clouds became a major trend in the last years [25–27,23]. Consequently, faster and more accurate 3D object recognition systems based on keypoint matching and descriptors extracted in the scene and in

the sought object point clouds were developed. After being established, point correspondences are grouped by hypotheses sharing a common transformation, which is estimated by voting [28,29], multi-dimensional clustering [30,31] or RANSAC [32] (also used to detect shapes on 3D data [33]). The presence of the object of interest is then inferred if certain conditions are met, such is the number of votes, cluster size, or the number of RANSAC inliers.

With the wider availability of consumer-grade depth sensors such as the Microsoft Kinect, several works on 3D object recognition are proposed employing this class of sensor [34–38,24]. Aldoma et al. [24] proposed the global feature coined Clustered Viewpoint Feature Histogram (CVFH) to improve performance of object recognition for robotics. Machine learning based approaches [37,38] were formulated to perform 3D object recognition making heavy use of depth information, without any computation on point clouds involved.

Aldoma et al. [34] highlight how algorithms that are part of the *Point Clouds Library* (PCL) software package could be used to form 3D object recognition systems based on local and global features. There are also 3D object classification/categorization systems, as in the works of Wohlkinger et al. [35,36] and of Lai et al. [37]. In this latter class of systems, every chair in a scene should be labeled as the object of type "chair", whereas in object recognition only the specific chair being sought should be retrieved from the scene.

3. Foveated point cloud

This work proposes the use of a foveated point cloud in order to reduce the processing time of object detection. The idea is that the point density is higher nearby the fovea and that this density decreases according to the distance from the fovea. In this way, it is possible to reduce the total number of the points reducing also the processing time at the same time that the density around the fovea is enough to keep feasible the object detection. Parts of the point cloud with reduced density may be useful in providing other stimuli which may be part of a context of visual attention. For example, a saliency map can be computed in the foveated cloud in order to drive bottom-up or top-down stimulus. This can be very useful in the context of robotic vision, since the robot can be aware to multiple simultaneous stimuli in the environment.

3.1. Foveated point cloud model

The foveated point cloud proposed here is based on the 2D foveated model proposed by Gomes [4]. This model transforms an image into a set of smaller images with same size but with different resolutions. In order to achieve that, the model defines image patches from the original image that are arranged in a sequence of levels. The first level is a mapping of the whole original image while the last one is a mapping of a patch placed at the original image centered at a fovea. This patch has the same size of each image level. The result is a set of small images that composes a foveated image.

In the 3D case, instead of resampling concentric image patches, the foveated point cloud is achieved by downsampling the original point cloud using concentric boxes, each one representing a level as shown in Fig. 2. Each box specifies a point cloud crop each one with a different point cloud density. The outer box has one of its corners placed at a specific 3D coordinate and it defines the model coordinate system. See the axes in Fig. 2. All points outside this box are discarded. Inside it, smaller boxes are linearly placed. The smallest box is centered at a parameter called *fovea*: a 3D coordinate where the point cloud density is maximum. A downsampling schema is applied in this smallest box. Each bigger box is also downsampled but with a level by level decreasing point cloud density up to the outer box, where the point cloud density is minimum.

The proposed foveated point cloud is formalized as follows. We define m+1 3D boxes of size $\mathbf{S_k}\in\mathbb{R}^3$, with $k=0,1,\ldots,m$ representing each level. Each level of the foveated point cloud model changes the point cloud density. The first level (level 0) has a density reduction by d_0 and the last one (level m) has a density reduction by d_m. The density reduction of intermediate levels is given by linear interpolation between d_0 and d_m.

The largest box has three parameters: size (S_0), orientation and position (denoted by $\boldsymbol{\Delta}$). Usually, if the whole point cloud should be covered, it is possible to automatically set these three parameters as the bounding box of the entire scene. However in some applications, it could be interesting to place it in a part of a huge point cloud. The last two parameters determine the model coordinate system.

The smallest box is guided by a fovea $\mathbf{F}$ at that box center. For formalization convenience, the fovea coordinate system origin is (0, 0, 0) at the largest box center. In this way, $\mathbf{F}=\mathbf{F}'-\mathbf{S_0}/2$, where $\mathbf{F}'$ is the fovea at model coordinate system.

Let $\boldsymbol{\delta}_k\in\mathbb{R}^3$ be the displacement of box at level k, then $\boldsymbol{\delta}_0=(0,0,0)$ and $\boldsymbol{\delta}_m+\mathbf{S_m}/2=\mathbf{F}'$.

The displacement of each box using linear interpolation is given by

$$\boldsymbol{\delta}_k=\frac{k(\mathbf{S_0}-\mathbf{S_m}+2\mathbf{F})}{2m} \tag{1}$$

Note that $\boldsymbol{\delta}_k$ is defined only for $m>0$; in other words, the foveated model should have at least 2 levels.

The size of each k-th box using linear interpolation is given by

$$\mathbf{S_k}=\frac{k\mathbf{S_m}-k\mathbf{S_0}+m\mathbf{S_0}}{m} \tag{2}$$

3.2. Fovea growth factor

Here, we introduce a fovea growth factor $\mathbf{G}=(s_x,s_y,s_z)\in\mathbb{R}^3$, where s_x,s_y,s_z are the scale factors applied to directions x, y and z, respectively (see Fig. 3). As detailed in Section 4, this factor increases the number of points by enlarging levels volumes. Observe that this model behaves like there is no foveation when $\mathbf{G}$ goes to ∞.

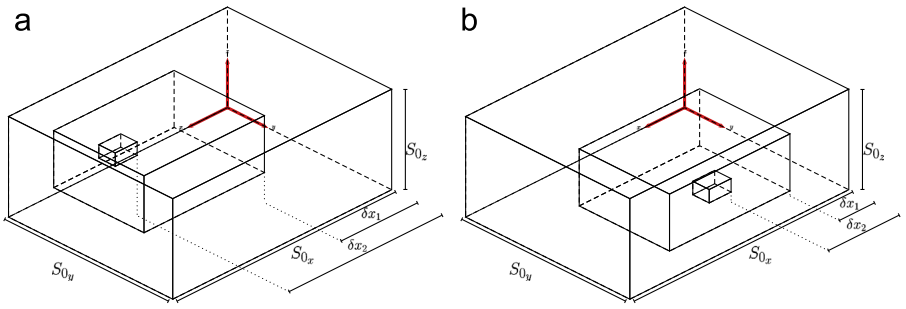

Fig. 2. Foveated model with 3 levels. Two different placements for the fovea were used in (a) and (b).

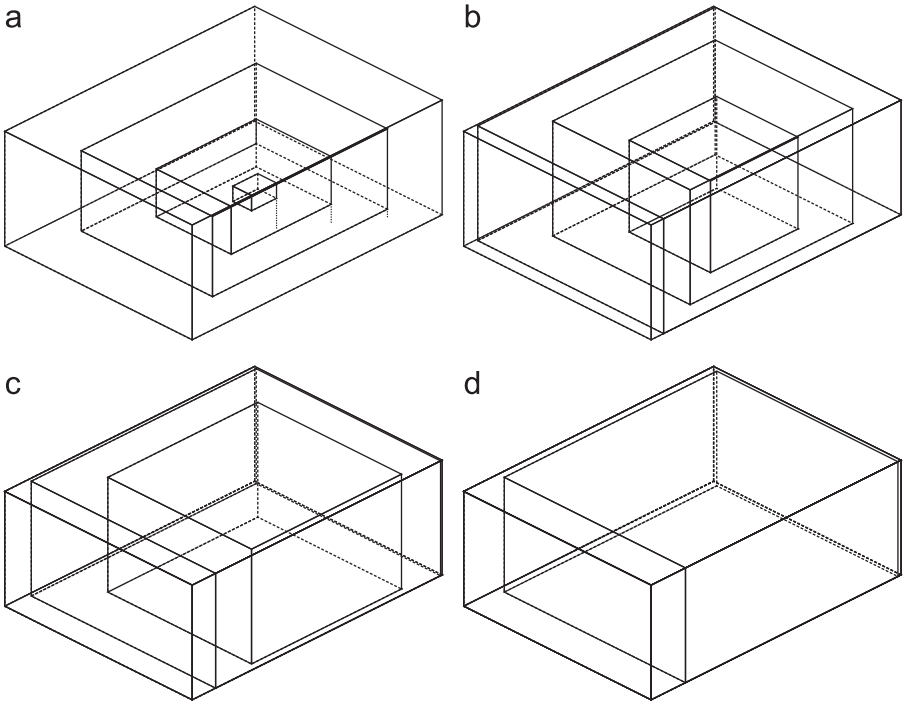

Fig. 3. Different values for the fovea growth factor. (a) G=(0, 0, 0), (b) G=(20, 20, 20), (c) G=(40, 40, 40) and (d) G=(60, 60, 60).

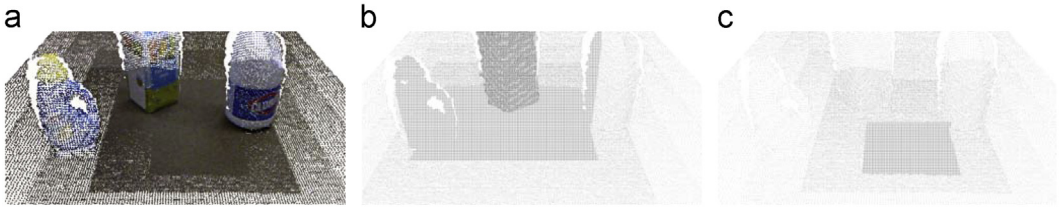

Fig. 4. Example of application of the foveated model: (a) original point cloud, (b) foveated with $m=4$, $\mathbf{S}_m=(0.4,\ 0.4,\ 0.4)$, $\mathbf{S}_0=(1,\ 1,\ 1)$ and $\mathbf{F}=(-0.06,\ 0.11,\ -0.75)$ and (c) foveated with $m=4$, $\mathbf{S}_m=(0.2, 0.2, 0.2)$, $\mathbf{S}_0=(1, 1, 1)$ and $\mathbf{F}=(0.05, -0.11, -0.75)$.

In this way, each level is bounded by the lower limit of maximum between $\delta_k-\mathbf{G}$ and (0, 0, 0) and the upper limit of the minimum between $\delta_k+\mathbf{S_k}+\mathbf{G}$ and S_0. These minimum and maximum limit the levels to the original point cloud boundary.

3.3. Downsampling

After foveated levels boundaries computation, the point cloud is downsampled in order to change the point cloud density. In this step, there are two possibilities of point cloud storage: to create a single point cloud joining all the downsampled points from each level or to store each downsampled point cloud from each level independently. Note that both ways can be adopted simultaneously.

However, by joining all points in a single cloud leads to geometric distortions, probably imperceptible on a visual inspection, if the downsampling algorithm modifies the points coordinates. A possible solution to this issue is to join all points from a level that do not belong to an inner level. This way points from a level do not mix with points from another one. In order to ensure the disjunction between levels, it is enough to test if the point from a level k to be inserted in the foveated point cloud is not inside the box $k+1$ ($k\neq m$) as depicted in Algorithm 1. Example of a foveated cloud point can be seen in Fig. 4

Algorithm 1. Processing steps applied to foveate a point cloud.

Input: point cloud $\{P\}_{in}$
Input: minimum density parameter d_0
Input: maximum density parameter d_m
Output: point cloud $\{P\}_{out}$
Output: point cloud vector $\{L\}$ of size $m+1$
// `Foveated Point Cloud`
foreach *level k* **do**
  $d_k = d_0 + k(d_m - d_0)/m$;
  downsample $\{P\}_{in}$ into $\{L\}_k$ using d_k as parameter;
  foreach *point p of* $\{L\}_k$ **do**
    if *p is inside box k and is not inside box* $k+1$ **then**
      add p to point cloud $\{P\}_{out}$;
    end
  end
end

3.4. Fovea position

As explained before, one of the parameters of the foveated point cloud is the fovea position vector. A dense point cloud is more suitable to successful correspondence matching. If the object

is far from the fovea then fewer keypoints are extracted and less correspondences are found. Thus, it is desirable to keep the fovea near the object. In order to achieve better results, the proposed architecture includes a visual attention module that guides the fovea placement.

First, a function evaluates if the object is detected by the system. If the object is detected, then the fovea is moved to the object's centroid. Otherwise, if the object is not detected, then some strategy may be applied in order to recover the fovea position. A sequence of fixations can also be used along the time or at once in order to detect where the object is. Once the object is detected, a tracking module can be applied so that the fovea is always in the desirable place.

One straightforward strategy is to disable foveation until the object is found. This temporarily increases the processing time, but the original point cloud is used and, then, the object can be found at foveated peripheral areas. Another strategy is to gradually increase the growth fovea factor. By using this strategy, it is possible to gradually increase the number of cloud points and thus avoiding having a processing time peak. Another possible strategy is to use a bottom-up attention strategy. In this case, the fovea is moved to the most salient region, which can be computed considering the class of objects to be found.

If the scene has more than one object, then it is possible to foveate each object at a time and process them in sequence. In other words, if two objects, for example, ask for top-down attention, then the visual process pay attention to one in a frame and to the another one in the next frame.

As some of these issues are not the main contribution of the current work, we neglect it to be treated in a future work. We just wanted to remark that it is possible to apply several strategies based on visual attention in order to properly place the fovea.

4. Proposed object recognition scheme

In this section, we discuss the core framework that our system is based, the *correspondence grouping* algorithm [29]. After showing the standard method, the foveated scheme to recognize objects is presented, along with the modifications and implications that were needed to maximize performance using multiresolution data.

4.1. Searching objects in 3D point clouds

The proposed object recognition scheme works with point clouds (set of 3D points referenced in a fixed frame) representing the *object model* and the *scene* to be processed. Positions in this reference frame supposedly having an instance of the sought object are given as outputs. We note here that our system recognizes objects in a scene if and only if the model of the query object is available, implying that it does not perform object classification/categorization or retrieve similar objects from a previously computed database (as is the case of some systems enumerated on the work of Tangelder and Veltkamp [22]). Put differently, the query object is searched in the scene and not vice versa. As a consequence, this allows the system to find multiple instances of the same object in a single scene.

We have chosen to build our system based on the local 3D features framework, which exploits local geometric traits at key positions in point clouds in order to extract discriminative descriptors employed to establish point correspondences between keypoints from the model and from the scene. These point correspondences are further processed to infer possible presences of the object. Moreover, this class of system presents some desirable and important properties, such as robustness to occlusion and scene clutter, dispensing the need to elaborate extensive and cumbersome training stages (mandatory for machine learning approaches) and ability to process point clouds acquired from RGB-D sensors like the Kinect in an efficient manner.

The system is based on the correspondence grouping approach of Tombari and Di Stefano [29] (with implementation publicly available [39]), in which a model object is recognized in the scene if, after keypoint correspondences being established, enough evidence for its presence in a given position is gathered. This scheme is shown in Fig. 5a. For the sake of completeness, every step of the system is described as follows.

4.1.1. Extracting local 3D descriptors

The first step in the correspondence grouping algorithm is to describe both the scene and model point clouds. For this, the normal vector for each point is computed considering a surface generated by a neighborhood of size k_n around each point. Then, a uniform downsampling algorithm is applied to extract keypoints as the centroid of all points contained within a radius r_k. After this,

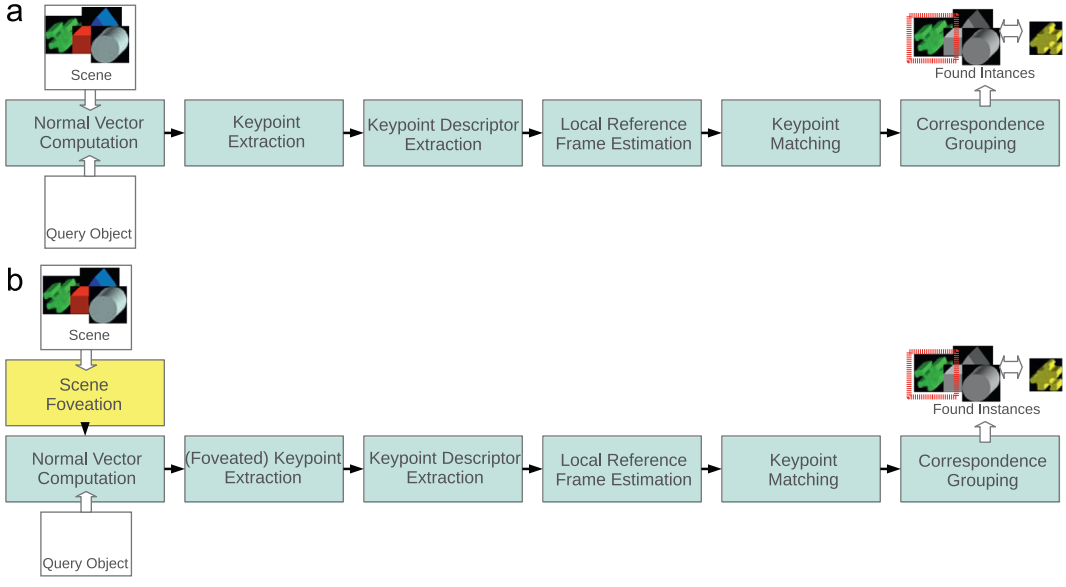

Fig. 5. Object recognition algorithm based on correspondence grouping: (a) Original scheme and (b) proposed foveated object recognition scheme. In (b), the scene is downsampled through a foveation step, reducing considerably the number of points to be processed without compromising overall accuracy (see text).

SHOT (*Signature of Histograms of OrienTations*) descriptors [27] are computed, assembling a histogram of the normals within a neighborhood of radius r_s as the signature of a keypoint. The last step to fully describe point clouds is a very important stage encompassing the estimation of a *Local Reference Frame* (LRF) for keypoints of the model and scene. Thus, the principal axes spanning a LRF within a neighborhood of radius r_l in each keypoint position are estimated robustly by the algorithm of Petrelli and Di Stefano [40]. The result of this computation (three unit vectors for each principal direction) is associated with each keypoint and will be employed in the final stage of the object recognition scheme. Different values for the parameters are set in the scene and in the model, allowing more precise recognition tasks. For clarity, the process which extracts descriptors for the model and scene is shown in Algorithms 2 and 3 respectively, with parameters $k_{nm}, r_{km}, r_{sm}, r_{lm}$ used for the model and $k_{ns}, r_{ks}, r_{ss}, r_{ls}$ used for the scene.

Algorithm 2. Processing steps applied to the model point cloud. See text for parameter details.

Input: model point cloud $\{M\}$
Output: model keypoint $\{K\}_m$
Output: model descriptors $\{D\}_m$
Output: model LRFs $\{L\}_m$
// `Process model`
foreach *point of* $\{M\}$ **do**
  compute the normal vectors $\{N\}_m$ within a neighborhood of size k_{nm};
end
from $\{M\}$, extract model keypoints $\{K\}_m$ by uniform downsampling with radius r_{km};
foreach *keypoint of* $\{K\}_m$ **do**
  compute the SHOT descriptors $\{D\}_m$ within a neighborhood of size r_{sm} using normals $\{N\}_m$;
  compute the set of LRFs $\{L\}_m$ within a neighborhood of size r_{lm};
end

Algorithm 3. Processing steps applied to the scene point cloud. See text for parameter details.

Input: scene point cloud $\{S\}$
Output: scene keypoint $\{K\}_s$
Output: scene descriptors $\{D\}_s$
Output: scene LRFs $\{L\}_s$
// `Process scene`
foreach *point of* $\{S\}$ **do**
  compute the normal vectors $\{N\}_s$ within a neighborhood of size k_{ns};
end
from $\{S\}$, extract model keypoints $\{K\}_s$ by uniform downsampling with radius r_{ks};
foreach *keypoint of* $\{K\}_s$ **do**
  compute the SHOT descriptors $\{D\}_s$ within a neighborhood of size r_{ss} using normals $\{N\}_s$;
  compute the set of LRFs $\{L\}_s$ within a neighborhood of size r_{ls};
end

4.1.2. Keypoint matching

For each keypoint and respective descriptor and LRF in the model a match in the scene is searched by finding the closest scene point (in the Euclidean sense) in the n-dimensional space containing the SHOT descriptors. Search procedures are employed in a kd-tree to handle the cumbersome routine involved. If the squared distance between the SHOT descriptors is smaller than a threshold d^2_{max}, a point correspondence is established. This process is highlighted in Algorithm 4.

Algorithm 4. Keypoint matching. See text for parameter details.

Input: model and scene keypoints $\{K\}_m$ and $\{K\}_s$
Input: model and scene descriptors $\{D\}_m$ and $\{D\}_s$
Input: model and scene LRFs $\{L\}_m$ and $\{L\}_s$
Output: set of keypoint correspondences $\{C\}$
initialize correspondences $\{C\}$ as empty;
// `Find keypoint correspondences`
foreach *keypoint of* $\{K\}_m$ **do**
  let d_{mi} be the descriptor of the current keypoint k_{mi};
  using $\{K\}_s$ and $\{D\}_s$, find the nearest keypoint k_{sj} with descriptor d_{sj};
  if $euclidian_{distance}(d_{mi}, d_{sj}) < d_{max}$ **then**
    add pair of triplets $(k_{mi}, d_{mi}, lrf_{mi})$ and $(k_{sj}, d_{sj}, lrf_{sj})$ to $\{C\}$;
  end
end

4.1.3. Correspondence grouping

Since each model and scene keypoint have a LRF associated, a full rigid body transformation modeled by a rotation and a translation can be estimated between the LRFs associated with each keypoint correspondence. Accordingly, a 3D Hough Space is used to gather evidence of the object presence through a voting process. After the rigid body transformation is applied, the cell of the Hough Space containing this 3D position is calculated and its accumulator incremented. Finally, after repeating these steps for all correspondences, object instances are deemed found at each cell having the number of votes larger than a predefined threshold V_h. The size of each cell is controlled by a parameter L_h. Algorithm 5 illustrates the object recognition scheme through correspondence grouping.

Algorithm 5. Object recognition based on correspondence grouping. See text for parameter details.

Input: Correspondences set $\{C\}$
Output: set of object instances $\{I\}$
initialize Hough accumulator $\{H\}$ with cell size L_h;
// `Group correspondences`
foreach *correspondence* $\{C\}$ **do**
  estimate transformation T which aligns lrf_{mi} to lrf_{sj};
  evaluate $B = Tk_{mi}$ and find the cell h in $\{H\}$ in which lies B;
  increment number of votes of h;
end
foreach *cell h of* $\{H\}$ **do**
  if *h has at least* V_h *votes* **then**
    add the object instance located in the scene at h to $\{I\}$;
  end
end

4.1.4. Saving computation time

We note that the normal vectors are evaluated considering neighborhoods formed by all points in the clouds, whereas the SHOT descriptors and LRFs are computed at the neighborhoods of the keypoints. In this way, computation time is saved, while the local 3D geometric traits are still kept discriminative.

4.2. Object recognition in foveated point clouds

To enhance the 3D object recognition capabilities of the correspondence grouping approach, the cloud foveation algorithm

is employed after some adaptations. A complete scheme of the proposed 3D object recognition system is shown in Fig. 5b.

A foveated model is applied to the cloud acquired from the depth sensor according to Algorithm 1 and the parameters of Table 2. Normal estimation can be done before or after foveation. In the first case, the computation is more expensive, but the captured geometric traits of the scene are less distorted. In the current version of the system, we opted to conserve scene geometry.

Since the keypoint extraction (uniform downsampling) would extract keypoints with a single radius (originally r_{ks}), the multi-resolution of the scene cloud would not be respected, as shows Fig. 6b. Consequently, we adapted the keypoint extraction to be also dependent on different and specified levels of resolution, possibly differing of the used downsampling radii $d_0 \ldots d_m$. The correspondence grouping algorithm was then modified to accommodate keypoint extraction with foveated point clouds. The scene points are downsampled using a different radius r_k for each level $k = 0, \ldots, m$. The first level (level 0) uses a radius of r_0 and the last (level m) uses a radius of r_m. All other radii from the intermediate levels are linearly interpolated. Thus, keypoints can be extracted respecting each level of the foveated scene resolution, as shows Fig. 6c.

There are two major consequences about this approach. First, there is a considerable time saving due to the keypoint reduction both in descriptors computation and correspondence step, since different values for the extraction radius are employed instead of a (possibly small) single value. Second, it is possible to greatly increase the keypoint density near the fovea position without significantly increasing the total number of original scene points. This peripheral decrease and foveal increase in the number of points also reduces the number of false descriptors correspondences, improving thus the object detection success rate if the fovea is properly placed.

In the foveated version of the recognition scheme, Algorithm 3 (scene processing) would be modified to include the scene foveation after the normal estimation and to extract keypoints using a radius value r_k for each resolution level instead of the r_{ks} (see Fig. 5b).

5. Experiments and results

5.1. Implementation details

The proposed object recognition system is implemented in C++ on an Ubuntu Linux environment, making use of the functionalities provided by the *Point Cloud Library* (PCL) [34]. The experiments are evaluated on a non-optimized version of the system running at a laptop PC with an Intel Core i5 2.5 Ghz processor and 4 GB of RAM.

5.2. Ground truth generation

The availability of ground truth data allows a more in depth evaluation of the proposed object recognition method by directly comparing the object data with the algorithm output. Hence, through the use of a model object that can be described analytically and also be accurately and easily identifiable on a given scene, we can proceed with a more thorough analysis regarding the number of instances of the query object found and its retrieved position in the scene. To cope with this, an object model in the form of a sphere is utilized as ground truth after some computation steps are executed in each scene to collect its actual position in the global reference frame and also its radius. For this, a RANSAC [41] procedure is applied aiming to fit a 3D sphere model in each scene point cloud. Accordingly, this simple yet efficient procedure is able to correctly identify the sphere in scenes with variable levels of clutter and occlusion accurately enough to suffice our needs. After carefully placing the sphere in the scene, we manually

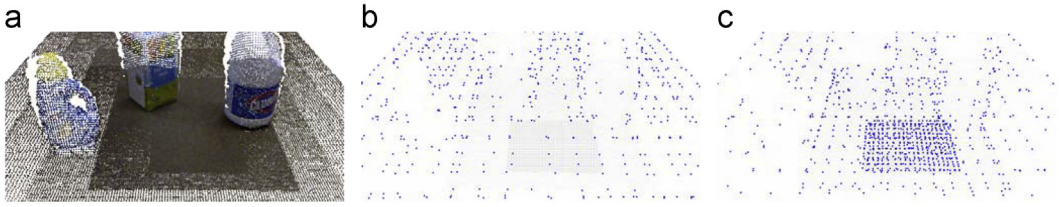

Fig. 6. Keypoint extraction: (a) Foveated point cloud, (b) keypoints by uniform sampling and (c) keypoints by uniform sampling using different radii for each level.

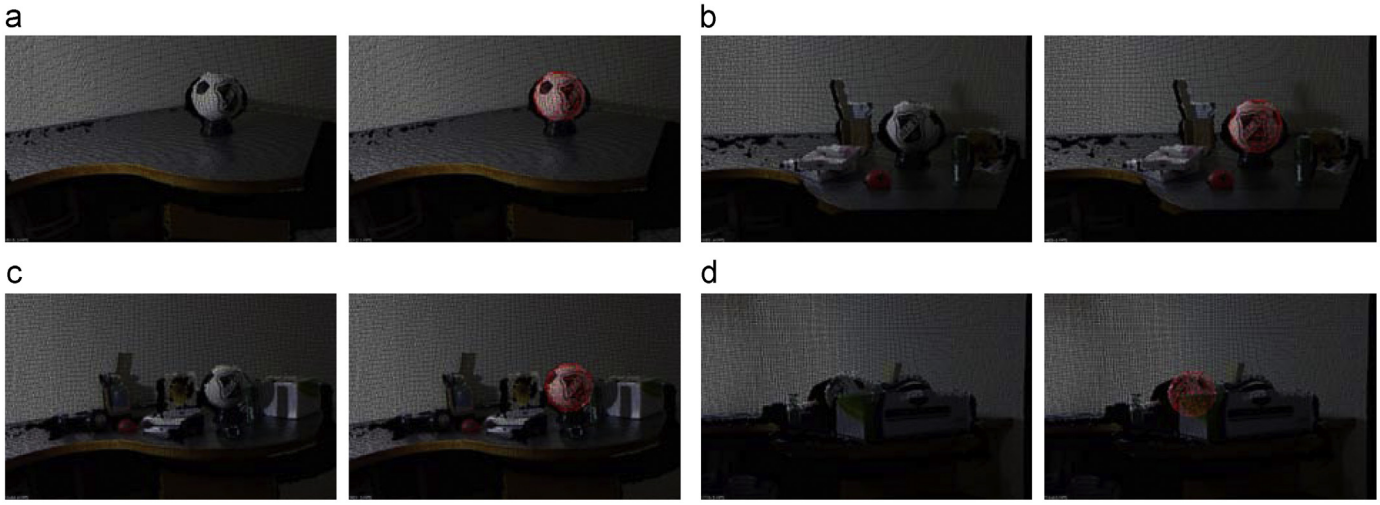

Fig. 7. Samples from different scene arrangements containing a sphere and the extracted sphere parameters (used as ground truth) highlighted in red. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.) From (a) to (d), each scene has an increasing level of clutter.

move a Kinect sensor, capturing a total of 12 frames of scenes with increasingly level of clutter from different views, at the same time that the parameters of the detected sphere are gathered and annotated in a ground truth file. Fig. 7 shows 4 samples of the acquired scenes and their respective ground truth.

5.3. Foveated object recognition experiment

In this experiment, the correspondence grouping 3D object recognition method is compared with our proposed moving fovea object recognition. As can be seen in Section 4.1, the standard correspondence grouping algorithm has several parameters which should be properly tuned according to the operation scenario for a successful recognition procedure. This can be explained because the scene/object dimensions should be taken into account and also because there are not (yet) any methods allowing automatically parameterization. Table 1 shows all parameters involved in the process and their respective default values. These used default values are determined empirically after executing the algorithm for various scenes gathered with a similar sensor (3D Kinect), setup (objects on a table top) and distances between the sensor and the scene.

The foveated recognition scheme has parameters to be set, specifying the region in 3D space and the desired resolution levels. The moving fovea parameters are shown in Table 2.

We aim to show experimentally that the foveated approach decreases the total running time per frame of the correspondence grouping process while keeping high successful recognition rates. For this, various foveated and non-foveated setups are instantiated by varying the different parameters involved. These different configurations are divided into 4 groups. The first group uses three different sizes for the highest resolution level $\mathbf{S}_m$ keeping constant values for the downsampling radii, $r_0 = 0.01$ and $r_m = 0.5$. All three setups are shown in Table 3. The second group (Table 4) employs three possibilities for r_m with $\mathbf{S}_m = (0.1, 0.1, 0.1)$ and $r_0 = 0.01$. Three different configurations varying r_0, with $\mathbf{S}_m = (0.25, 0.25, 0.25)$ and $r_m = 0.3$ were used for the third group of fovea settings, as shows Table 5. All these foveated configurations are built with three resolution levels ($m = 2$). The last group is composed by five non-foveated setups. We decide to keep all the parameters of the standard correspondence grouping constant, employing different values only for the radius to choose the scene keypoints, r_{ks}. Table 6 summarizes the last group.

Since no object recognition methods are free of resulting in false detections (false positives), some criterion to classify a detection as successful (true positive) or not has to be established. To comply with this, the Euclidean distance between the ground truth sphere position and the detected sphere position is measured. The sphere is stated as detected if this distance is below a given threshold. It is important to note that the used correspondence grouping algorithm (standard and foveated) is able to detect multiple instances of the same object in the scene. Thus, if multiple detections are triggered within a given radius around the ground truth position, only one is counted. Both distance thresholds were set with a value of 8 cm. The object detection configurations are run in all 12 scenes with the number of true positives and false positives per configuration being kept. There is only one instance of the sought object in each scene, summing up a total of 12 true positives.

The results for all the 14 configurations in the 12 scenes are shown in Table 7. For the first group (having varying fovea size $\mathbf{S}_m$), only the configurations with ID 1 and 2 were able to successfully detect the object. The small fovea size (10 cm in each box dimension) and large r_m (50 cm) used in the configuration 0 severely degrades the scene resolution to a point that becomes impossible to extract relevant local features and descriptors.

Table 1
List of parameters of the correspondence grouping algorithm and their default values (see text for details). All parameters are given in scene units (meters).

Parameter	Descriptor	Default value
k_{ns}	K-Neighbors normals scene	10
r_{ks}	Radius keypoints scene	0.03
r_{ss}	Radius SHOT scene	0.02
r_{ls}	Radius LRF scene	0.015
k_{nm}	K-Neighbors normals model	10
r_{km}	Radius keypoint model	0.01
r_{sm}	Radius SHOT model	0.02
r_{lm}	Radius LRF model	0.015
d^2_{max}	Correspondence threshold	0.25
L_h	Size of Hough cell	0.01
V_h	Voting threshold	5

Table 2
List of parameters of the moving fovea and their default values (see text for details). All parameters are given in scene units (meters).

Parameter	Descriptor	Default value
m	Number of res. levels - 1	3
$\mathbf{S}_0$	Lowest density box size	(3.0, 3.0, 3.0)
$\mathbf{S}_m$	Highest density box size	(0.5, 0.5, 0.5)
$\mathbf{F}$	Fovea position	(−0.07, 0.02, 0.6)
$\mathbf{\Delta}$	Outer box position	(−2.9, −1.9, −1.3)
r_0	Lowest density keypoint radius	0.08
r_m	Highest density keypoint radius	0.02

Table 3
Group 1 of settings for the foveated recognition—change in $\mathbf{S}_m$.

ID	$\mathbf{S}_m$	r_0	r_m
0	(0.1, 0.1, 0.1)	0.01	0.5
1	(0.25, 0.25, 0.25)	0.01	0.5
2	(0.4, 0.4, 0.4)	0.01	0.5

Table 4
Group 2 of settings for the foveated recognition—change in r_m.

ID	$\mathbf{S}_m$	r_0	r_m
3	(0.1, 0.1, 0.1)	0.01	0.02
4	(0.1, 0.1, 0.1)	0.01	0.08
5	(0.1, 0.1, 0.1)	0.01	0.14

Table 5
Group 3 of settings for the foveated recognition—change in r_0.

ID	$\mathbf{S}_m$	r_0	r_m
6	(0.25, 0.25, 0.25)	0.02	0.3
7	(0.25, 0.25, 0.25)	0.08	0.3
8	(0.25, 0.25, 0.25)	0.14	0.3

Table 6
Group 4 of settings for the non-foveated recognition—change in r_{ks}.

ID	r_{ks}
9	0.01
10	0.02
11	0.05
12	0.08
13	0.14

Consequently, not enough point correspondences could be stabilized to detect the object.

Though we use the smallest fovea size of the first group in all configurations of the second group, smaller values for r_m are set, enabling correct detections, except for the configuration 5. A relatively large (14 cm) for r_m explains this fact. The setup number 3 detected the sphere in all 12 scenes and the setup 4 in only 3 of them. This is expected because a smaller r_m value decreases the uniform sampling radius from intermediate levels (interpolated between r_0 and r_m). Also, it increases the number of keypoints throughout the levels and thus increases the number of successful detections.

In the third group, only configuration number 6 triggered the detection of the sought object in the scenes, though in not all of them (11 out of 12). A larger r_0 was set, resulting in overly downsampled point clouds and thus, no detections for setups 7 and 8.

Table 7
Results for a total of 14 configurations for the foveated (0–8) and non-foveated (9–13) recognition methods, after recognizing the ground truth objects in 12 scenes with varying levels of clutter. The average processing time is shown, along with the number of true positive detections (with distance to the true position below 0.08 m).

ID	Time (s)	True positive	False positive
0	0.166	0	0
1	1.058	12	1
2	2.612	12	2
3	1.582	12	1
4	0.364	3	0
5	0.251	0	0
6	0.507	11	1
7	0.244	0	0
8	0.225	0	0
9	25.748	12	2
10	7.201	12	0
11	1.935	0	0
12	1.341	0	0
13	1.004	0	0

Only the non-foveated configurations 9 and 10 detect the object. This latter setup detect the sphere in all scenes without any false detections, and hence can be elected the *gold standard* against which we can compare our algorithm. Curiously, the configuration that results in the largest number of false detections is the non-foveated with the highest resolution (configuration 9) and the foveated with the largest fovea size (configuration 2). In some sense, we could state that the overall downsampling process applied by the fovea plays an important role possibly removing redundancy and undesired features from the scene point cloud. The largest recognition rate achieved by foveated setups is also 12 true detections, but with one false positive (setups 1 and 3) and two false positives (setup 2).

Although the configuration number 10 has had the best recognition performance in terms of detection ratio, it is the second slowest of all setups, with an average of 7.201 s per scene. A successful recognition in the foveated scenario is achieved with an average of 0.364 s per frame by the setup 4, but with only 0.33% of success ratio. The fastest average processing time achieved (0.507 s) with recognition rates higher than 50% is computed by the setup 6, with 91.6% of true positive rate and only 8.3% of false positive rate. This setup is also the fastest showing the smallest false positive rate with at least 50% of success. The setup showing the fastest computing times with true positive rate at 100% is the setup 1, with an average of 1.058 s per frame and false positive rate of 8.3%. Of all foveated configurations, the worst average belongs to the setup 2. In terms of gain in computation performance, the fastest setup shows an increase of 19.78*x*, while the most accurate offers 6.8*x* and the slowest is still 2.75*x* faster than the better standard correspondence grouping configuration.

Obviously, the slowest average computation time of all configurations is computed with the non-foveated which has used the smallest radius to extract the scene keypoints. Also, we can see that as the scene resolution degrades, what is dictated mainly by the two interpolating radii rather than the fovea size, the computation times and recognition rate decrease.

These results emphasize that using the moving fovea, one can opt to dramatically decrease recognition times providing a small increase in false detections (if acceptable) or to decrease recognition times keeping false detections to a minimum.

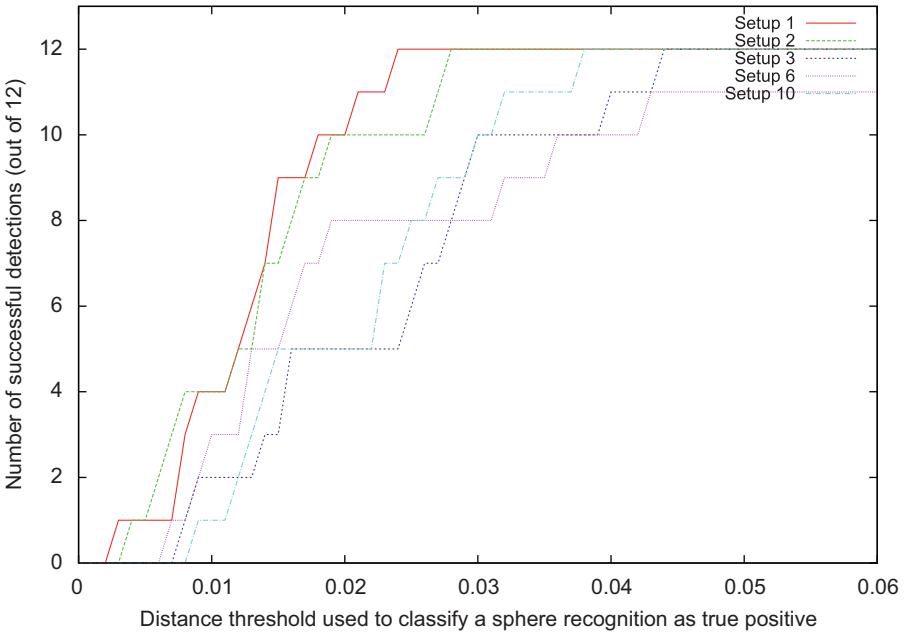

Fig. 8. The number of true positives detected by each setup in function of the threshold (meters) considered to classify a detection as successful.

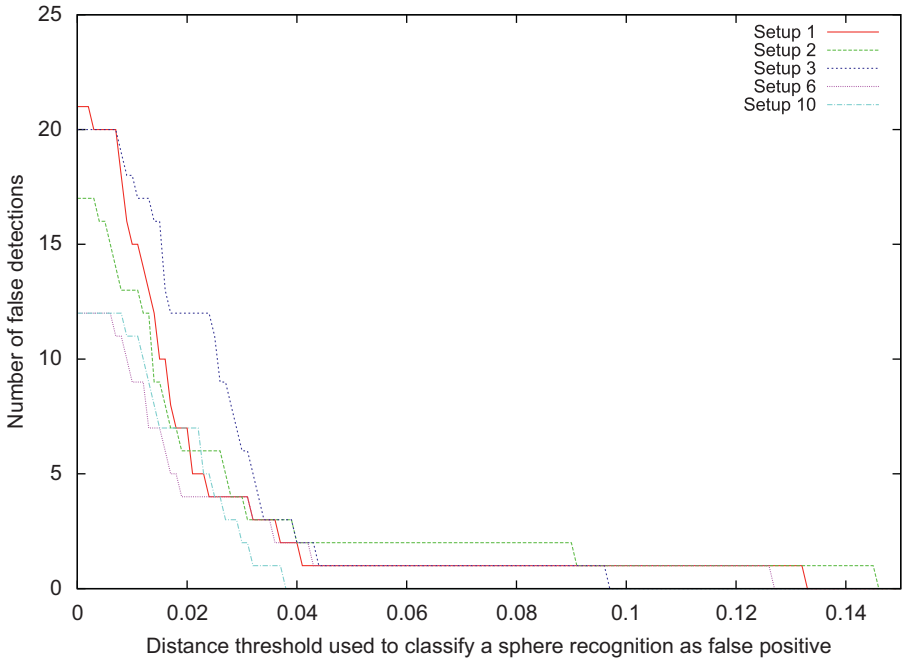

Fig. 9. The number of false positives detected by each setup in function of the threshold (meters) considered to classify a detection as false.

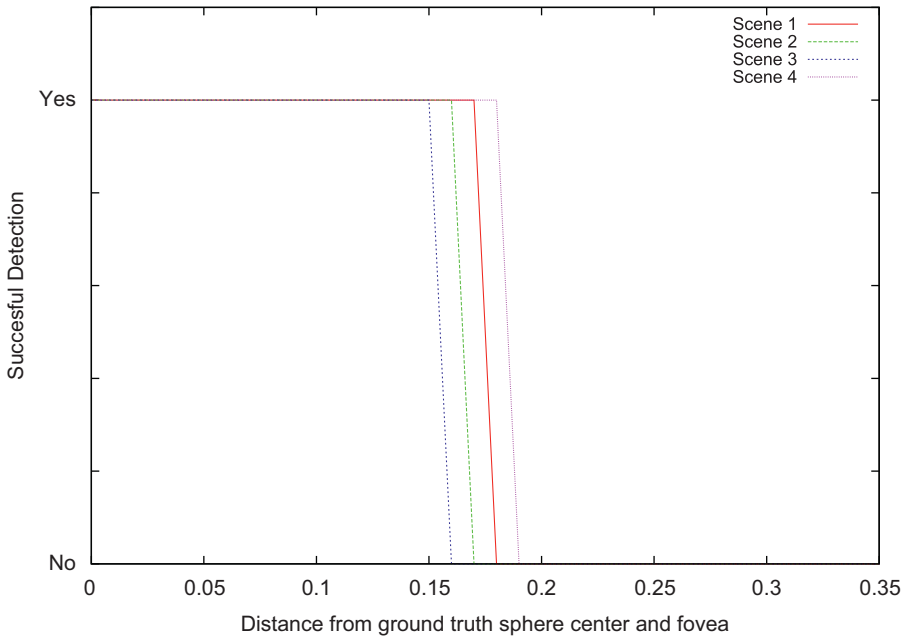

Fig. 10. Successful detections in each of four scenes by varying the distance between the fovea and the ground truth position.

5.4. *Detection sensitiveness*

Due to the criterion used in stabilizing an object detection as true positive and false positive, we show in Figs. 8 and 9 how sensitive to the detection threshold (distance between the detected object and the ground truth) are the best four foveated setups (IDs 1, 2, 3 and 6) and the best non-foveated one (10). As it can be seen in Fig. 8, the best foveated setup in terms of overall confiability is also the most capable to detect the object closer to the ground truth position. In fact, it is even less sensitive to the threshold than the *gold standard* configuration, being able to recognize the object in all 12 scenes using as threshold 2.5 cm, while the best non-foveated only recognized the same number with the threshold set as 3.8 cm. In Fig. 9, we analyze the threshold employed to classify a detection as false positive also for the top five configurations. We note that using a low value (e.g. 1 cm), all object detection setups find multiple instances of the same object around its true position. The non-foveated setup required the smallest threshold (4 cm) to avoid false detections while the setup number three was the foveated configuration which required the smallest value. The most reliable foveated setup stopped triggering false detections only after 13.5 cm.

5.5. *Varying the fovea distance to the object of interest*

We also analyze how the fovea placement in the scene would influence the overall object detection performance. For this, we have

chosen 4 samples scenes with varying amount of clutter and run the foveated correspondence grouping algorithm with varying distances between the ground truth position and the fovea position along one axis. Distances in the range of 0–35 cm were used. The fixed foveated parameters were $m=2$, $\mathbf{S}_m=(0.3,0.3,0.3)$, $\mathbf{S}_0=(1.0,1.0,1.0)$, $r_0=0.01$ and $r_m=0.06$. The results of the experiments can be seen in Fig. 10. For each scene, the figure shows if detection of the sphere succeeded or not, according to the distance being discussed. Clearly, as the fovea moves far from the (true) object center, less correct correspondences are found, resulting also in less true object detections. Nevertheless, with all other parameters kept constant, the moving fovea method correctly recognizes the object if it is positioned within a distance of at most 18 cm.

5.6. Experiments available at the collage authoring environment

Experiments available to the user are built using the Collage Authoring Environment for Executable Papers from Elsevier, allowing a complete and interactive experience of the proposed 3D object recognition method directly from the web. Functionalities such as data acquisition, data visualization through an interactive 3D point cloud visualizer, modification of algorithm parameters, execution of the object recognition system and code editing are readily available. If desired, the source codes can be downloaded to be used with other systems.

We verify that users frequently find difficulties while trying to execute their first tests with 3D models, specially related to capture and visualization. In light of this, besides the executable system that is available from Collage, we also offered two minor contributions to this authoring environment to help users to start experiment with 3D object capture and manipulation quickly and easily: a data acquisition module and a tool to manipulate and visualize 3D point clouds.

5.6.1. Point cloud acquisition

We have developed a data acquisition module that allows visualization and capture of 3D point clouds using the Microsoft Kinect Sensor by simply using a web browser, so users do not need any specific software installed on their machine (with the exception of the Kinect sensor device drivers). This system can be easily used to generate 3D models of the scenes and objects of interest for further processing.

In order to be able to capture data from the Kinect Sensor, we developed a web based Kinect capture system, which relies on the Java Web Start (JWS) technology to allow anyone to use it with a single click on a web page. We used the OpenNI library [42] and Java Native Interface (JNI) abstractions to be able to connect to the Kinect sensor plugged on the user's computer.

The system generates 3D point clouds in the PCD (*point cloud data*) file format, a portable, simple and open format for 3D data introduced by the *Point Cloud Library* (PCL) [34]. It consists of a standard ASCII text file containing a 3D point information per each line in a format separated by spaces (X Y Z RGB), so any point and its color can be represented by a single line of ASCII numbers.

5.7. Point cloud manipulation and visualization

Another contribution is the PCD point cloud opener for Collage. We have adapted and worked with the Collage team to deploy a PCL web viewer (kindly granted to use by the PCL development team) on Collage, so any Collage Experiment can open and manipulate point clouds with a simple and intuitive interface. The PCL viewer uses the WebGL system, a recent standard for 3D object rendering directly on web pages. WebGL is supported on most modern browsers, thus, in order to view and interact with the point clouds, a browser with WegGL support must be used.

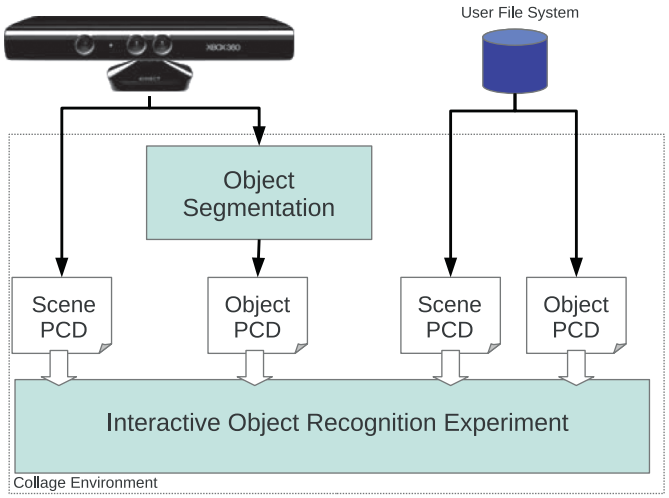

Fig. 11. Workflow of the object recognition experiment available at the Collage Authoring Environment.

5.8. Interactive object recognition

The object recognition experiment available (Fig. 11) to the user in the Collage Environment is now explained in detail.

In order to provide his/her own input point clouds to our experiment, users are able to use the data acquisition module described in Section 5.6.1. Observe that this module is completely independent from the Collage Authoring Environment.

Our Collage Experiment contains two experiments: Model Segmentation Experiment and Object Recognition Experiment. The first one is responsible for taking one point cloud representing a scene composed by only one object lying on a table. The Model Segmentation Experiment then generates the Model Point Cloud, containing only the point cloud representing the object. The Object Recognition Experiment takes as input the Model Point Cloud generated by the previous experiment and a point cloud of a scene containing this object, among others. Then, it executes the Correspondence Grouping and the Foveated Point Cloud approaches. The user is able to see the visual result of this execution and also text files with details about it.

The workflow presented in the Model Segmentation Experiment will be followed, in order to capture and segment the object model (according to Sections 5.6.1 and 5.8.1 respectively). A *readme1.txt* text file with general instructions on how to use the system is available as Experiment Data Item 1, while a *readme2.txt* (Experiment Data Item 2) explains how to proper acquire and segment the object model. The scene captured containing only the object of interest upon a table is then shown in the Experiment Data Item 3, an interactive point cloud visualizer (see Section 5.7). Experiment Data Item 4 shows the source code files. The Experiment Code Item 1 contains Bash commands used to compile and execute the source code. After performing the actual segmentation process, the resulting point cloud containing the model data is shown in another interactive window at the Experiment Data Item 5, with textual output from the program shown at the Experiment Data Item 6. If desired, the source code (Experiment Data Item 4) of the segmentation procedure can be edited (or downloaded). In the case that the user chooses not to capture data, a previously specified PCD file *penguin.pcd* is used as default value for the Experiment Data Item 3. Alternatively, a PCD file assumed to be already stored in the user's file system can also be supplied by modifying the Experiment Data Item 4. We note here that this PCD file should contain the object point cloud already segmented.

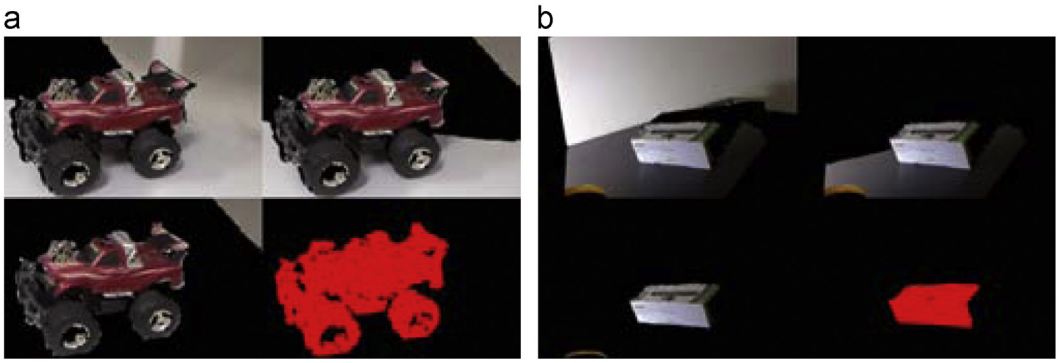

Fig. 12. Algorithm showing each step of the object segmentation scheme, used to extract points in a scene point cloud belonging to an object of interest. From top to bottom, left to right, the two images show the original scene, points after filtering by distance, points after plane removal and object points. (a) A toy car is extracted and (b) a box is segmented.

After the model acquisition and segmentation, the actual Object Recognition Experiment can be executed. A text file *readme3.txt* (Experiment Data Item 7) instructs the user on how to proceed, while the source code can be visualized, edited or downloaded by manipulation of the Experiment Data Item 9. The user can supply a scene containing the object to be recognized or use the default *scene.pcd* through the Experiment Data Item 8. The experiment executes the *correspondence grouping* algorithms in its default version (Section 4.1) and foveated version (Section 4.2), resulting in two outputs presented for each execution: a point cloud visualization of the recognition result, highlighting the recognized instances in the scene and also the object of interest and a textual output, printing important information of the execution such as correspondences founds and execution time. These two outputs can be inspected in the Experiment Data Item 10 and Experiment Data Item 11 (default visualization and text output) and Experiment Data Item 12 and Experiment Data Item 13 (foveated visualization and text output).

5.8.1. User supplied object segmentation

In the case that the user opts to supply the object point cloud via the Kinect, an object segmentation procedure has to be applied to the data, aiming to extract only the points belonging to the object of interest. To this end, after positioning the model object over a plain surface (e.g. a table) without any other object on the field of view of the sensor, the point cloud is processed by:

1. removing points that are at a distance z_{max} from the sensor;
2. removing remaining points classified as contained on a plane until a given fraction F_{points} of all points is reached through RANSAC [41] plane segmentation;
3. clustering remaining points by Euclidean Distance, implying that starting from a single point, points with distance smaller than a threshold $d_{cluster}$ are grouped together.

This simple procedure was verified experimentally to work well for several objects, as can be seen in Fig. 12a and b.

User collaboration in placing the object as indicated is necessary to reduce the number of points to be processed and also to avoid points from the object model being incorrectly discarded. Some shortcomings such as incorrect point clouds as results can easily be solved with proper parameter tuning.

6. Conclusions

We have presented in this article the usage of the moving fovea approach to efficiently execute 3D object recognition tasks. We note that although this paper explores the usage of the foveated approach to a specific object recognition algorithm, the proposed technique is well suitable to be used with any other 3D object recognition or retrieval system.

One key feature of the proposed approach is the usage of a multiresolution scheme, which uses several levels of resolution ranging from the highest resolution, possibly equal to the resolution of the original scene, to lower resolutions. Lower resolutions are obtained by reducing the point cloud density according to the fovea level, which consequently reduces the processing time. This setup is similar to the human vision system, which focus attention and processing on objects of interests at the same time that keeps attention and processing to peripheral objects, but with lower resolution.

Experimental analysis shows that the proposed method dramatically decreases the processing time used to recognize 3D objects on scenes with considerable level of clutter, while keeping accuracy loss to a minimum. In comparison to a state-of-the-art recognition method, a true positive recognition rate of 91.6% was achieved with an improvement of seven fold performance gain in terms of average recognition time per frame. These results are well suitable for usage on mobile and embedded systems with low computational resources or on applications that need faster object recognition processing time, such as robotics.

As a future work, we plan to explore the usage of the foveated multiresolution system to best find the fovea position according to possible objects identified on lower scales. As the object is found on the lower scale, then the fovea can focus and process detailed information of the object at the best fovea position.

Acknowledgements

The authors would like to thank the support from the National Research Council (CNPq), Brazilian sponsoring Agency for research and also the PCL development team that has allowed the use of PCL web viewer tool.

Appendix. Supporting information

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cag.2013.03.005.

Note from publisher: this material was originally submitted as part of the Collage Executable Paper pilot, please visit http://www.elsevier.com/executablepaper for more information.

References

[1] Campbell RJ, Flynn PJ. A survey of free-form object representation and recognition techniques. Comput Vis Image Underst 2001;81(2):166–210, http://dx.doi.org/10.1006/cviu.2000.0889.

[2] Microsoft. Microsoft kinect. ⟨http://www.xbox.com/KINECT⟩; 2012. Online; accessed 26-October-2012.

[3] Chang EC, Yap CK. A wavelet approach to foveating images. In: Proceedings of the thirteenth annual symposium on computational geometry. New York, NY, USA: ACM. ISBN 0-89791-878-9; 1997. p. 397–9. http://doi.acm.org/10.1145/262839.263024.

[4] Gomes RB, Goncalves LMG, Carvalho BM. Real time vision for robotics using a moving fovea approach with multi resolution. In: IEEE international conference on robotics and automation; 2008. p. 2404–9.

[5] Gomes HM, Fisher R. Learning and extracting primal-sketch features in a log-polar image representation. In: Proceedings of Brazilian symposium on computer graphics and image processing. IEEE Computer Society; 2001. p. 338–45.

[6] Çagatay Dikici, Bozma HI. Attention-based video streaming. Signal Process: Image Commun 2010;25(10):745–60, http://dx.doi.org/10.1016/j.image.2010.08.002.

[7] Lee S, Bovik A. Fast algorithms for foveated video processing. IEEE Trans Circuits Syst Video Technol 2003;13(2):149–62, http://dx.doi.org/10.1109/TCSVT.2002.808441.

[8] Itti L. Automatic foveation for video compression using a neurobiological model of visual attention. IEEE Trans Image Process 2004;13(10):1304–18, http://dx.doi.org/10.1109/TIP.2004.834657.

[9] Wang Z, Lu L, Bovik A. Foveation scalable video coding with automatic fixation selection. IEEE Trans Image Process 2003;12(2):243–54, http://dx.doi.org/10.1109/TIP.2003.809015.

[10] Bernardino A, Santos-Victor J. A binocular stereo algorithm for log-polar foveated systems. In: Proceedings of the second international workshop on biologically motivated computer vision. London, UK: Springer-Verlag. ISBN 3-540-00174-3; 2002. p. 127–36.

[11] Olof Eklundh J. Attending, foveating and recognizing objects in real world scenes. In: British machine vision conference – bmvc04; 2004.

[12] Chen K, Lin C, Chiu T, Chen M, Hung Y. Multi-resolution design for large scale and high-resolution monitoring. IEEE Trans Multimedia 2011;PP(99):1256–68, http://dx.doi.org/10.1109/TMM.2011.2165055.

[13] Wang W, Chen C, Wang Y, Jiang T, Fang F, Yao Y. Simulating human saccadic scanpaths on natural images. In: IEEE conference on computer vision and pattern recognition (CVPR); June 2011. p. 441–8. http://dx.doi.org/10.1109/CVPR.2011.5995423.

[14] Butko N, Movellan JR. Infomax control of eye movements. IEEE Trans Autonomous Mental Dev June; 2(2):91–107. http://dx.doi.org/10.1109/TAMD.2010.2051029.

[15] Gonçalves LMG, Grupen RA, Oliveira AA, Wheeler D, Fagg A. Tracing patterns and attention: humanoid robot cognition. Intelligent Syst Appl 2000;15(4):70–7.

[16] Basu A, Cheng I, Pan Y. Foveated online 3d visualization. In: Proceedings of the 16th international conference on pattern recognition, vol. 3; 2002. p. 944–7. http://dx.doi.org/10.1109/ICPR.2002.1048193.

[17] Ashbrook AP, Fisher RB, Robertson C, Werghi N. Finding surface correspondance for object recognition and registration using pairwise geometric histograms. In: Proceedings of the 5th European conference on computer vision-volume II–volume II. ECCV '98; London, UK, UK: Springer-Verlag. ISBN 3-540-64613-2; 1998. p. 674–86. URL ⟨http://dl.acm.org/citation.cfm?id=645312.648923⟩.

[18] Johnson AE, Hebert M. Surface matching for object recognition in complex 3-d scenes. Image Vision Comput 1998;16:635–51.

[19] Johnson A, Hebert M. Using spin images for efficient object recognition in cluttered 3d scenes. IEEE Trans Pattern Anal Mach Intell 1999;21(5):433–49, http://dx.doi.org/10.1109/34.765655.

[20] Hough P. Methods and means for recognizing complex patterns. 1962. US Patent 3069654.

[21] Mian A, Bennamoun M, Owens R. Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans Pattern Anal Mach Intell 2006;28(10):1584–601, http://dx.doi.org/10.1109/TPAMI.2006.213.

[22] Tangelder JW, Veltkamp RC. A survey of content based 3d shape retrieval methods. Multimedia Tools Appl 2008;39(3):441–71, http://dx.doi.org/10.1007/s11042-007-0181-0.

[23] Rusu R, Bradski G, Thibaux R, Hsu J. Fast 3d recognition and pose using the viewpoint feature histogram. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), 2010; 2010. p. 2155–62. http://dx.doi.org/10.1109/IROS.2010.5651280.

[24] Aldoma A, Vincze M, Blodow N, Gossow D, Gedikli S, Rusu R, et al. Cad-model recognition and 6dof pose estimation using 3d cues. In: 2011 IEEE international conference on computer vision workshops (ICCV workshops); 2011. p. 585–92. http://dx.doi.org/10.1109/ICCVW.2011.6130296.

[25] Hetzel G, Leibe B, Levi P, Schiele B. 3d object recognition from range images using local feature histograms. In: Proceedings of the 2001 IEEE Computer Society conference on computer vision and pattern recognition, 2001. CVPR 2001; vol. 2; 2001. p. II–394–9. http://dx.doi.org/10.1109/CVPR.2001.990988.

[26] Rusu R, Blodow N, Marton Z, Beetz M. Aligning point cloud views using persistent feature histograms. In: IEEE/RSJ international conference on intelligent robots and systems, 2008. IROS 2008; 2008. p. 3384–91. http://dx.doi.org/10.1109/IROS.2008.4650967.

[27] Tombari F, Salti S, Di Stefano L. Unique signatures of histograms for local surface description. In: Proceedings of the 11th European conference on computer vision conference on computer vision: part III. ECCV'10; Berlin, Heidelberg: Springer-Verlag. ISBN 3-642-15557-X, 978-3-642-15557-4; 2010. p. 356–69. URL ⟨http://dl.acm.org/citation.cfm?id=1927006.1927035⟩.

[28] Chen H, Bhanu B. 3d free-form object recognition in range images using local surface patches. Pattern Recogn Lett 2007;28(10):1252–62, http://dx.doi.org/10.1016/j.patrec.2007.02.009.

[29] Tombari F, Di Stefano L. Object recognition in 3d scenes with occlusions and clutter by Hough voting. In: 2010 Fourth pacific-rim symposium on image and video technology (PSIVT); 2010. p. 349–55. http://dx.doi.org/10.1109/PSIVT.2010.65.

[30] Mian A, Bennamoun M, Owens R. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. Int J Comput Vision 2010;89(2–3):348–61, http://dx.doi.org/10.1007/s11263-009-0296-z.

[31] Zhong Y. Intrinsic shape signatures: a shape descriptor for 3d object recognition. In: 2009 IEEE 12th international conference on computer vision workshops (ICCV Workshops); 2009. p. 689–96. http://dx.doi.org/10.1109/ICCVW.2009.5457637.

[32] Papazov C, Burschka D. An efficient RANSAC for 3d object recognition in noisy and occluded scenes. In: Proceedings of the 10th Asian conference on computer vision – volume part I. ACCV'10; Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-642-19314-9; 2011. p. 135–48. URL ⟨http://dl.acm.org/citation.cfm?id=1964320.1964334⟩.

[33] Schnabel R, Wahl R, Klein R. Efficient ransac for point-cloud shape detection. Comput Graphics Forum 2007;26(2):214–26.

[34] Aldoma A, Marton Z, Tombari F, Wohlkinger W, Potthast C, Zeisl B, et al. Tutorial: point cloud library: three-dimensional object recognition and 6 dof pose estimation. IEEE Robotics Autom Mag 2012;19(3):80–91, http://dx.doi.org/10.1109/MRA.2012.2206675.

[35] Wohlkinger W, Vincze M. Shape-based depth image to 3d model matching and classification with inter-view similarity. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), 2011; 2011. p. 4865–70. http://dx.doi.org/10.1109/IROS.2011.6094808.

[36] Wohlkinger W, Aldoma A, Rusu R, Vincze M. 3dnet: large-scale object class recognition from cad models. In: 2012 IEEE international conference on robotics and automation (ICRA); 2012. p. 5384–91. http://dx.doi.org/10.1109/ICRA.2012.6225116.

[37] Lai K, Bo L, Ren X, Fox D. A large-scale hierarchical multi-view rgb-d object dataset. In: 2011 IEEE international conference on robotics and automation (ICRA); 2011. p. 1817–24. http://dx.doi.org/10.1109/ICRA.2011.5980382.

[38] Blum M, Springenberg J, Wulfing J, Riedmiller M. A learned feature descriptor for object recognition in rgb-d data. In: 2012 IEEE international conference on robotics and automation (ICRA); 2012. p. 1298–303. http://dx.doi.org/10.1109/ICRA.2012.6225188.

[39] Point Cloud Library. 3d object recognition based on correspondence grouping. ⟨http://pointclouds.org/documentation/tutorials/correspondence_grouping.php#correspondence-grouping⟩; 2012. Online; accessed 26-October-2012.

[40] Petrelli A, Di Stefano L. On the repeatability of the local reference frame for partial shape matching. In: 2011 IEEE international conference on computer vision (ICCV); 2011. p. 2244–51. http://dx.doi.org/10.1109/ICCV.2011.6126503.

[41] Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 1981;24(6):381–95, http://dx.doi.org/10.1145/358669.358692.

[42] OpenNI. OpenNI library. ⟨http://openni.org⟩; 2012. Online; accessed 26-October-2012.