Better features to track by estimating the tracking convergence region

Zoran Zivkovic, Ferdinand van der Heijden

Laboratory for Measurement and Instrumentation
Faculty of Electrical Engineering, University of Twente
P.O. box 217, 7500 AE Enschede, The Netherlands
E-mail: `Z.Zivkovic@el.utwente.nl`

Abstract

Reliably tracking key points and textured patches from frame to frame is the basic requirement for many bottom-up computer vision algorithms. The problem of selecting the features that can be tracked well is addressed here. The Lucas-Kanade tracking procedure is commonly used. We propose a method to estimate the size of the tracking procedure convergence region for each feature. The features that have a wider convergence region around them should be tracked better by the tracker. The size of the convergence region as a new feature goodness measure is compared with the widely accepted Shi-Tomasi feature selection criteria.

1. Introduction

Many computer vision algorithms rely on feature point tracking through the image sequences (just to mention a few [7, 2, 6, 5]). In the initialization stage the features to be tracked are chosen. The main topic of this paper is the problem of how to select the ”good” features that can be reliably tracked. A new feature goodness measure is proposed that is more global in nature then the widely accepted local measure proposed by Shi and Tomasi in [8].

The paper is organized as follows. First, the Lucas-Kanade procedure for feature tracking and the Shi-Tomasi method for feature selection are briefly described. Further, a method for estimating the convergence region is given. The implementation issues are also discussed in detail. Finally, some experiments are presented comparing the two feature quality measures and some conclusions are drawn.

2. Image Motion Model

Away from the occluding boundaries the changes between two frames in an image sequence can be described as:

$$I_1(\vec{x}_{im}) = I_0(\vec{x}_{im} + \vec{\delta}(\vec{x}_{im})) \tag{1}$$

The new image $I_1(\vec{x})$ can be reconstructed by moving the points from the initial image $I_0(\vec{x}_{im})$ by $\vec{\delta}(\vec{x}_{im})$. Here vector $\vec{x}_{im} = [\ x_{im} \quad y_{im}\ ]^T$ presents the 2D image coordinates. The image motion displacement between the two frames can be written as:

$$\vec{\delta}(\vec{x}_{im}) = \vec{d} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{2}$$

which is a simple pure translation image motion model.

3. Computing Image Motion

To compute image motion we need some additional assumptions since we have just one equation and two unknowns in $\vec{d}$ (see [1]). The simplest situation is the one when the flow is the same everywhere. This can be a reasonable assumption if we consider a patch from an image that is small enough.

We use the standard measure of dissimilarity:

$$J(\vec{d}) = \iint_W [I_1(\vec{x}_{im}) - I_0(\vec{x}_{im} + \vec{d})]^2 d\vec{x}_{im} \tag{3}$$

where W is the window of the given feature patch. In practice the integration denotes simply summing over all the image pixels within the patch. The feature tracking problem is then equal to the problem of finding $\vec{d}$ that minimizes (3).

The equation (3) can be linearized with respect to $\vec{d}$ by a truncated Taylor expansion approximation:

$$I_0(\vec{x}_{im} + \vec{d}) = I_0(\vec{x}_{im}) + \vec{g}(\vec{x}_{im})^T \vec{d} \tag{4}$$

where

$$\vec{g}(\vec{x}_{im}) = \begin{bmatrix} g_x(\vec{x}_{im}) \\ g_y(\vec{x}_{im}) \end{bmatrix} \tag{5}$$

Here, $g_x(\overrightarrow{x}_{im})$ and $g_y(\overrightarrow{x}_{im})$ are the image derivatives in the x_{im} and the y_{im} direction at the image point $\overrightarrow{x}_{im}$.

The $\overrightarrow{d}$ that minimizes linearized version of (3) is the solution of the following system:

$$Z\overrightarrow{d} = \overrightarrow{e} \tag{6}$$

where

$$Z = \iint_W \left[\begin{array}{cc} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{array} \right] d\overrightarrow{x}_{im} \tag{7}$$

and

$$\overrightarrow{e} = \iint_W (I_0 - I_1) \left[\begin{array}{c} g_x \\ g_y \end{array} \right] d\overrightarrow{x}_{im} \tag{8}$$

The dependency on $\overrightarrow{x}_{im}$ is left out for simplicity.

Note that we are talking about image motion. However, in most of the practical situations we hope that the image motion corresponds to real world 3D motion projected into the image.

4. Texturedness

Not every image patch is suitable for tracking (for example, for an intensity edge we can estimate only normal velocity component). Various measures of ”texturedness” or ”cornerness” are proposed to avoid *the aperture problem*. If the displacement $\overrightarrow{d}$ is small, the linear approximation can be considered as good enough. Tracking a feature is then equivalent to solving the system given by (6). Consequently, the matrix Z must be both above noise level and well-conditioned. Therefore, eigenvalues of Z should be large and they should not differ by several orders of magnitude. Since the pixels have some maximum value the greater eigenvalue is bounded. In conclusion, a feature patch can be accepted if for some predefined λ we have

$$\min(\lambda_1, \lambda_2) > \lambda \tag{9}$$

where λ_1 and λ_2 are the eigenvalues of the matrix Z. This is the widely accepted Shi-Tomasi feature goodness measure [8].

5. Iterative search

The Lucas-Kanade procedure [4] minimizes (3) iteratively. The solution of the linearized system (6) is used to warp the new image I_1 and the procedure is repeated. This can be written as:

$$\overrightarrow{d}(k+1) = \overrightarrow{d}(k) + Z^{-1}\overrightarrow{e}(k), \text{with } \overrightarrow{d}(0) = 0 \tag{10}$$

where $\overrightarrow{d}(k)$ presents the estimated displacement at k-th iteration. The equation (8) with the image I_1 warped using $\overrightarrow{d}(k)$ gives us $\overrightarrow{e}(k)$. The described algorithm is the well known *Gauss-Newton minimization procedure* for sums of squares (see [3], chapter 6).

6. Convergence region estimation

If we denote the true displacement by $\overrightarrow{d}^*$ and define $\overrightarrow{x}(k) = \overrightarrow{d}^* - \overrightarrow{d}(k)$, the iteration equation (10) can be rewritten as:

$$\overrightarrow{x}(k+1) = \overrightarrow{x}(k) - Z^{-1}\overrightarrow{e}(k), \text{with } \overrightarrow{x}(0) = \overrightarrow{d}^* \tag{11}$$

First we define:

$$V(\overrightarrow{x}) = \|\overrightarrow{x}\| \tag{12}$$

Successful tracking would mean that

$$V(\overrightarrow{x}(k)) = \|\overrightarrow{x}(k)\| \to 0 \text{ for } k \to \infty \tag{13}$$

Convergence region is the domain where for each initial displacement $\overrightarrow{x}(0)$ the tracking process converges. The size of this region would be an appropriate criterion to define how well the feature could be tracked.

Suppose that we can find a domain S with the following properties:

$$\forall \overrightarrow{x}(k) \in S, \dot{V}(\overrightarrow{x}(k)) < 0 \text{ and } \overrightarrow{x}(k+1) \in S \tag{14}$$

with $\dot{V}(\overrightarrow{x}(k)) = V(\overrightarrow{x}(k+1)) - V(\overrightarrow{x}(k))$. Convergence is guaranteed within S since what we state is simply that we want to always move closer to the solution. Our function $V(\overrightarrow{x})$ is symmetric and monotonously increasing with respect to $\overrightarrow{x}$. If we find the closest point $\overrightarrow{x}_c$ for which $\dot{V}(\overrightarrow{x}_c) \geqslant 0$, the region $\|\overrightarrow{x}\| < \|\overrightarrow{x}_c\|$ will have the mentioned properties. The distance $\|\overrightarrow{x}_c\|$ can be used to describe the estimated convergence region size and consequently it is a proper feature goodness measure.

The theory presented here is inspired by the nonlinear system analysis methods and in this sense $V(\overrightarrow{x})$ corresponds to *the Lyapunov function* [9].

7. Implementation

The highly nonlinear function $\dot{V}(\overrightarrow{x})$ depends on the local neighborhood of the feature. As an example see figure 1. The function $\dot{V}(\overrightarrow{x})$ was sampled using a 0.5 pixel grid. We show also the estimated convergence region $\|\overrightarrow{x}\| < \|\overrightarrow{x}_c\|$.

In practical implementation for each feature we compute $\dot{V}(\overrightarrow{x})$ for some discrete displacements around the feature till we find the first $\dot{V}(\overrightarrow{x}) \geqslant 0$:

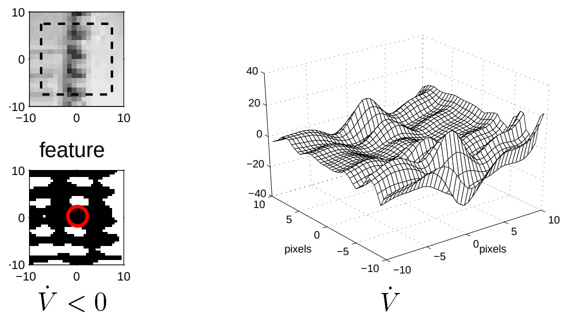

Figure 1. Estimated convergence region

```
Input: $I_0$, $g_x$, $g_y$ (image derivatives), $W$ (feature window),
    $Z^{-1}$, $SS$ (search strategy - an array of displacements
    $\vec{d}$ with nondecreasing $\left\|\vec{d}\right\|$- we use 8 points (every
    $45^o$) on concentric circles with radiuses increasing in
    $0.5$ pixel steps starting from initial $0.5$ pixels)

 1. $\vec{x}(0) = (\vec{d}^* =)SS(i)$
 2. Calculate $\vec{e}$ (window $W$ from $I_1$ = window $W$
    shifted for $\vec{d}^*$ from $I_0$)
 3. $\vec{x}(1) = \vec{x}(0) - Z^{-1}\vec{e}$ (Lucas-Kanade iteration)
 4. If $||\vec{x}(1)|| \geqslant ||\vec{x}(0)||(\dot{V} \geqslant 0)$
        return $||\vec{x}_c|| = ||\vec{x}(0)||$
    else
        $\{i = i + 1$; goto 1$\}$

Output: $||\vec{x}_c||$(estimated convergence region radius - dis-
    crete in 0.5 pixel steps in our case)
```

Computation costs are comparable with the computations needed for tracking the feature. In our case, average number of iterations, that are similar to the Lucas-Kanade iterations, is $8 * 2*$average$||\vec{x}_c||$.

8. Experiments

In our experiments we use the functions provided in public available OpenCV library (Intel Corporation). Therefore, we try to mention further all the parameters we used so that the experiments can be repeated. Our additional function to compute the $||\vec{x}_c||$ for the selected features can be downloaded at: `www.mi.el.utwente.nl/zzz`

8.1. Data set

In order to evaluate the proposed method we tried to choose a representative data set. The image sequences we used are presented in figure 2. Most of the sequences are taken from the CMU VASC Image Database (`vasc.ri.cmu.edu /idb`). The sequence ”marbled-block” from University of Karlsruhe (`i21www.ira.uka.de /image_sequences`) is added (complex motion - both camera and the object are moving).

The sequences exhibit a variety of camera movements, object textures and scene depth variations. Also, different cameras were used and different environment conditions were present. All the sequences have 512×480 format except the sequences ”house” 576×384 and ”marbled-block” 512×512. Some further motivation for our choices is given in the next section.

8.2. Real data experiment

For each sequence we take the initial image and select 200 best features according to the Shi-Tomasi criterion. The features are selected to be at a distance of at least 15 pixels from each other (10 pixels for the ’house’ sequence because of the large black background area) and at least 10 pixels from the image border. We calculate the convergence region estimates for the features. Further, we calculate the displacement between the initial and i-th image in the sequence using the Lucas-Kanade procedure with fixed 20 iterations (the image derivatives are calculated using the 3×3 Sobel operator). We choose i so that for at least 10% of the features the displacement is erroneously calculated. If we denote the initial image as the 0-th then for our sequences we had i values of 6,2,2,1,3,4,6,7,3,5 (written in the order as presented in figure 2) and the number of lost features was 45,147,64,83,102,40,57,20,38,54.

The ground-truth for detecting erroneously calculated displacements is generated using the Lucas-Kanade tracker but now using all the frames between the initial and the i-th frame. Therefore, when we were choosing our test sequences, we had to take care that the displacements between the consecutive frames are small. We also use a *coarse to fine* approach by blurring the images and calculating the displacement for larger windows first (helpful for almost flat scenes and camera translation only - for example ’building01’). Our ground truth is not perfect so finally we have checked most of the features by visual inspection.

In the initialization phase we are trying to select the features that can be tracked well. We have two criteria: minimum eigenvalue (’Good features’) and the new defined radius of the estimated convergence region (’Better features’). For both criteria we choose a threshold and throw away the features below the threshold. For our data set (total of 2000 ”corner-like” features selected from the sequences), the Receiver Operator Characteristic (ROC) for both criteria is presented in figure 3. A feature belongs to true-positive if it was selected and it was well tracked. False-positive are the features selected but lost. Clearly the new feature goodness criteria carries more information about how well

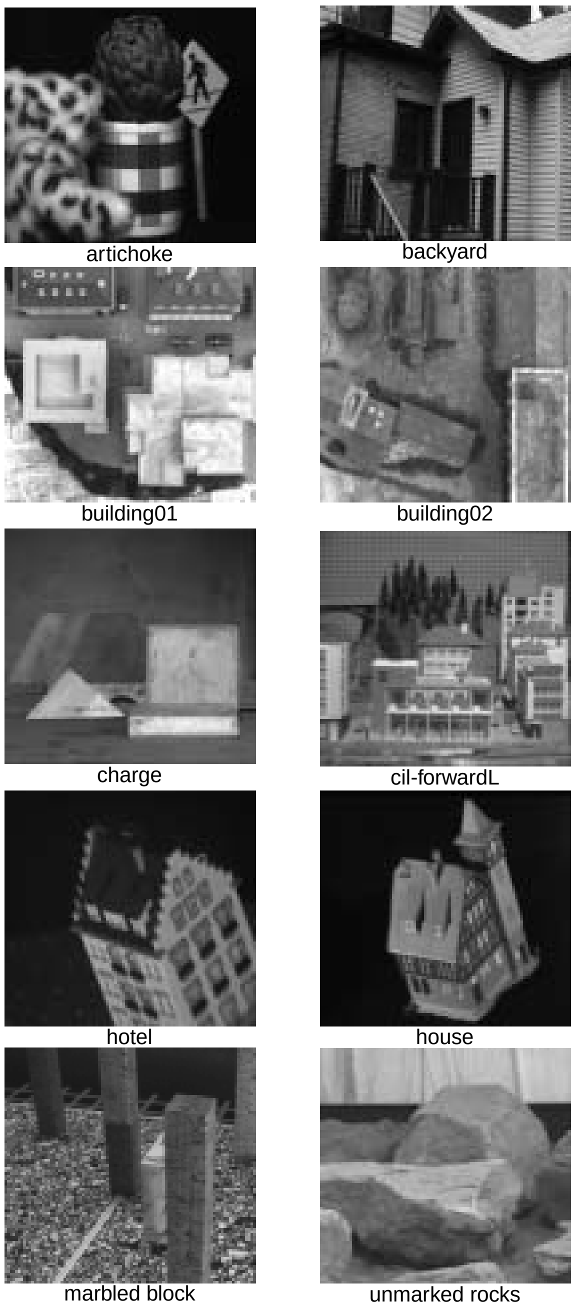

Figure 2. Image sequences

the features can be tracked. Local neighborhood is important for local search methods like the Lucas-Kanade tracker used here.

9. Conclusions

A new feature goodness measure is presented based on the Lucas-Kanade tracker performance. Selecting first a larger number of features and then throwing away the ones with small convergence region could be very useful for

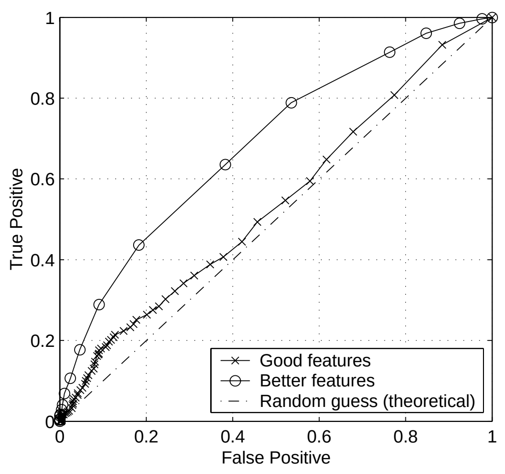

Figure 3. ROC - real sequences

many bottom-up computer vision algorithms that rely on the well tracked features. For example, if from our data set (the selected 2000 features) we keep the features having larger then average convergence region we get 708 features from which 118 erroneously tracked. Selecting the 708 features directly using the Shi-Tomasi ”cornerness” criterion gives 207 wrong features and that is almost two times more.

References

[1] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, Sept. 1995.

[2] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. *In Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 1997.

[3] R. Fletcher. *Practical Methods of Optimization*. J. Wiley, 1987.

[4] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *In Proceedings IJCAI81*, pages 674–679, 1981.

[5] Y. Song, X. Feng, and P. Perona. Towards detection of human motion. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 810–817, June 2000.

[6] J. Strom, T. Jebara, S. Basu, and A. Pentland. Real time tracking and modeling of faces: An ekf-based analysis by synthesis approach. *In Proceedings of International Conference on Computer Vision: Workshop on Modelling People*, September 1999.

[7] C. Tomasi. Pictures and trails: a new framework for the computation of shape and motion from perspective image sequences. *In Proceedings of CVPR*, pages 913–918, 1994.

[8] C. Tomasi and J. Shi. Good features to track. *In Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[9] M. Vidyasagar. *Nonlinear Systems Analysis*. Prentice-Hall, 1993.