INTECH
open science | open minds

OPEN ACCESS
ARTICLE

International Journal of Advanced Robotic Systems

A Fast Feature Points-Based Object Tracking Method for Robot Grasp

Regular Paper

Yang Yang[1,*] and Qixin Cao[1]

1 Research Institute of Robotics, Shanghai Jiao Tong University, Shanghai, China
* Corresponding author E-mail: yangyangcv@sjtu.edu.cn

Received 29 May 2012; Accepted 24 Jan 2013

DOI: 10.5772/55951

Abstract In this paper, we propose a fast feature points-based object tracking method for robot grasp. In the detection phase, we detect the object with SIFT feature points extraction and matching. Then we compute the object's image position with homography constraints and set up an interest window to accommodate the object. In the tracking phase, we only focus on the interest window, detecting feature points from the window and updating the window's position and size. Our method is of special practical meaning in the case of service robot grasp. Because when the robot grasps the object, the object's image size is usually small relative to the whole image, it is unnecessary to detect the whole image. On the other hand, the object is partially occluded by the robot gripper. SIFT is good at dealing with occlusion, but it is time consuming. Hence, by combining SIFT and an interest window, our method gains the ability to deal with occlusion and can satisfy the real-time requirements at the same time. Experiments show that our method exceeds several leading feature points-based object tracking methods in real-time performance.

Keywords SIFT, Interest Window, Homography Constraint, Tracking, Grasp

1. Introduction

Accompanying the development in robotics and computer vision, robot grasp in unstructured environments attracts more and more attention. Unlike industrial robots' grasps, autonomous robots' grasps in unstructured environments often suffer from a cluttered environment, partial occlusion and illumination changes. To deal with these challenges, some researchers propose to treat the robot grasp problem as a problem of object detection and pose estimation and propose to use features points and local descriptors to solve this problem [1]. A complete grasp process (a fetch and carry task) should contain four stages: object detection, pose estimation, grasp confirmation and evaluation of the grasp's quality. The last step involves judgment of the grasp's quality while the robot is holding the object. While most of the literature focuses on the first three stages, little attention is given to the final stage. In fact, solely focusing on grasp, or picking up an object is not enough for grasp tasks, especially for home service robots. For example, when the robot picks up a cup, the angle between the cup and the vertical axis is needed to avoid water spilling. When placing the object back on a table, the current position of the object must be known in order to place it correctly. Therefore, the last stage is vital

for the success of the full grasp task. Based on these considerations, we propose to use object tracking to monitor the whole robot grasp process. There is a huge amount of literature introducing all kinds of tracking approaches. [2] gives an extensive review of these approaches. According to [2], the tracking algorithms can be divided into three categories based on the use of object representations, namely methods establishing point correspondence, methods using primitive geometric models and methods using contour evolution. Among them, methods using point correspondences have been long used in both object detection and tracking. In 2004, David Lowe proposed an amazing local descriptor extraction algorithm named SIFT (scale invariant feature transform) [3], which possesses the merits of being invariant to scale, rotation/translation, or even illumination changes. Along with each feature point, there is a corresponding vector describing the feature of the point's neighbour and hence each point is distinctive and matching between points in different images much easier. Inspired by Lowe, many other feature point extraction methods have emerged, such as SURF [4] and YAPE [5], to mention a few. The speed of these feature point extraction methods is also accelerated with the evolution of hardware such as GPU [6, 7] and multi-core CPU [8]. Collet et al. [1] used SIFT features to recognize objects and to compute their 6D pose. In the training stage, dozens of images of the target object are taken from different viewpoints, the SIFT features are then extracted and these features are matched to build a sparse 3D model of the object with these feature point correspondences. In the online object recognition and pose estimation stage one single image is taken, the key points are extracted and then matched to the points stored in the training stage and the object's current pose is estimated from these 2D-3D point correspondences. With this method quite accurate results can be obtained, but as mentioned above, looking and grasping is only carried out once and isn't treated as a tracking procedure, which is not enough for a full robot grasp. [9] and [10] proposed that for robot grasp, it is necessary to monitor the grasping process and track the object during execution. They proposed an approach for object recognition and pose estimation based on colour co-occurrence histograms and a geometric model. They embedded the colour co-occurrence histograms in a learning framework that facilitated a "winner–takes–all" strategy across different scales. The hypotheses generated in the recognition stage provided the basis to estimate the orientation of the object around the vertical axis. The geometric model of the object is used to estimate and continuously track the complete 6 DOF pose of the object. However, because it relies on colour co-occurrence histograms, its performance will degrade in a cluttered environment, especially when the background has a similar colour to the target object. [5] suggested that, since we are going to recognize and track a previously known object, there is usually time to train the system beforehand and we can transfer the time from online recognition to offline training to improve the run-time performance. They generated thousands of synthetic views of the target object from one single image and cast the run-time key point recognition problem as a more generic classification problem and solved this classification problem with randomized trees [5] or random ferns [11]. Their methods remarkably improve the real-time performance for feature points-based object tracking. However, the training stage is time consuming. For a typical image of 640 by 480 pixels, several hours are usually needed to generate the detector and tracker data. Encouraged by the development of feature points, we have decided to adopt the feature points-based tracking method. [12] gave a comprehensive comparison of local descriptors and concluded that SIFT performs the best. However, the extraction of SIFT feature points is time consuming, which prohibits it from real-time applications. Although there are many GPU based SIFT methods [6, 7] as previously mentioned, they are still not fast enough. To promote real-time performance, in this paper we propose a fast SIFT feature points-based object tracking method especially for planar object tracking. In the initialization stage, we extract SIFT features, match them to our stored feature points and then setup an interest window containing the object using homography constraint. For the next incoming frames, we only focus on the interest window and hence greatly increase the speed.

2. Method Overview

The SIFT feature is robust and powerful for object recognition in a cluttered environment, even when the object is partially occluded and is invariant to scale, viewpoint or illumination changes, but it is time consuming. As mentioned in [3], an image of the size 500 by 500 pixels generally gives rise to about 2000 stable features. However, we usually don't need so many feature points for object detection and pose estimation. For example, we only need 5 point correspondences to recover the object's 6D pose given the camera's intrinsic parameters, according to [13]. For planar homography matrix estimation, theoretically we only need four point correspondences [14]. The time cost of SIFT feature point extraction has a close relationship with the image size. According to our test with Lowe's SIFT demo software (http://www.cs.ubc.ca/~lowe/keypoints/), the average time for extracting SIFT feature points from an 640 by 480 pixel image is 1.75 seconds, while the time is 0.63 seconds for an 320 by 240 pixel image. For object detection in the robot grasp situation, generally the target object only occupies a small part of the image. Hence, it is not necessary to detect the whole image. On the contrary, we can concentrate on a small window, which contains the object and update this window's property within a

tracking framework using the information from the previous frame. That is, after detecting the object, we can calculate its image position and size. For the incoming frame, only feature points from this window will be extracted. If the object is successfully detected, the position and size of this window will be updated to continue the tracking procedure. If the window's size becomes too small, which usually means the loss of tracking, the system can automatically reinitialize the tracking process by detecting the whole image and locate the window containing the object once again. A flow chart of the whole tracking process is shown in Figure 1.

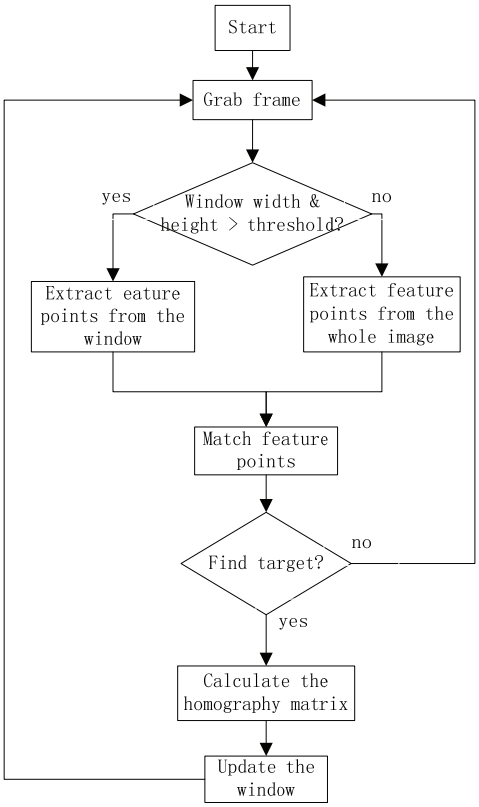

Figure 1. Flow chart of the tracking procedure.

3. SIFT Feature Point Extraction and Matching

First we will give a brief introduction to SIFT. Scale-invariant feature transform (SIFT) was first introduced by D. Lowe in 1999 [15] and enriched in 2004 [3]. Although the effect is surprisingly good, the thinking behind the process is quite simple. That is, to find a stable key point invariant to scale, viewpoint or illumination changes, first build an image pyramid in a scale space constituted by images convolved with Gaussian filters at different scales and then take the difference between successive Gaussian-blurred images. If the input image is represented as $I(x,y)$ and the Gaussian filter with standard deviation σ is $G(x,y,\sigma)$, then we can get a Gaussian-blurred scale space:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{1}$$

Where * is the convolution operation and $G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$. The difference of Gaussian (DoG) is:

$$\begin{aligned} DoG(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned} \tag{2}$$

Hence, DoG can be computed by simple image subtraction. The maxima or minima of the DoG images are just the candidates for feature points. To select points insensitive to noise and edge responses, discard those with low contrast or along an edge. After interpolation for a more accurate position, assign one or more orientations to each feature point. Finally, to make these candidates distinctive and stable for matching and recognition, assign a descriptor that is a 128 length vector depicting the features of the point's neighbourhood.

The feature point matching is performed with the Best Bin First algorithm [16], which is a variation of the kd-tree search algorithm designed to efficiently find an approximate solution to the nearest neighbour in high-dimensional spaces.

4. Robust Homography Estimation

Let $p_1(x_1,y_1,1)$ and $p_2(x_2,y_2,1)$ be a point correspondence from two different image frames over time. They are constrained by a homography matrix:

$$p_2 = H_{3\times3} \cdot p_1 \tag{3}$$

To convert (3) to the homogeneous form, make a cross product with p_2 at both sides of (3) to get:

$$0 = p_2 \times p_2 = H_{3\times3} \cdot p_1 \times p_2 \tag{4}$$

If we denote each row of $H_{3\times3}$ as h^{jT}, where $j = 1,2,3$ and T stands for transpose, we have:

$$p_2 \times (H_{3\times3} \cdot p_1) = \begin{bmatrix} y_2 \cdot h^{3T} \cdot p_1 - 1 \cdot h^{2T} \cdot p_1 \\ 1 \cdot h^{1T} \cdot p_1 - x_2 \cdot h^{3T} \cdot p_1 \\ x_2 \cdot h^{2T} \cdot p_1 - y_2 \cdot h^{1T} \cdot p_1 \end{bmatrix} \tag{5}$$

Because $h^{jT} \cdot p_1 = p_1^T \cdot h^j$ we have:

$$\begin{bmatrix} 0_{1\times3} & -1 \cdot p_1^T & y_2 \cdot p_1^T \\ 1 \cdot p_1^T & 0_{1\times3} & -x_2 \cdot p_1^T \\ -y_2 \cdot p_1^T & x_2 \cdot p_1^T & 0_{1\times3} \end{bmatrix} \cdot \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = A_{3\times9} \cdot h_{9\times1} = 0_{3\times1} \tag{6}$$

where $h_{9\times1}$ is a column vector composed by each element of $H_{3\times3}$:

$$h_{9\times1} = \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{21} & H_{22} & H_{23} & H_{31} & H_{32} & H_{33} \end{bmatrix}$$

Equation (6) is a simple homogeneous linear equation system and $h_{9\times1}$ can be calculated by the SVD decomposition of matrix A. According to [14], H can be uniquely determined by four non-collinear point correspondences. However, because of the imperfections in feature point matching, there must be some outliers, or wrong point correspondences, which will affect the accuracy of H. To reduce the outliers' impact, we use the RANSAC algorithm [17] to compute H robustly. We randomly choose four point correspondences and use equation (6) to compute a homography matrix. After that, we re-project points from the second image onto the first using equation (3) and compute the error $e = (x_2 - \frac{h_{11}x_1 + h_{12}y_1 + h_{13}}{h_{31}x_1 + h_{32}y_1 + h_{33}})^2 + (y_2 - \frac{h_{21}x_1 + h_{22}y_1 + h_{23}}{h_{31}x_1 + h_{32}y_1 + h_{33}})^2$. If this error is less than a threshold, then we say this point correspondence satisfies the homography matrix and is an inlier. This procedure is repeated until we find a homography matrix with the most inliers, which is the robust estimation of the true homography matrix.

5. Update the Interest Window

After obtaining the homography matrix, each corner of the target object is back projected onto the current frame using Equation (3). A minimum window is found that can accommodate the four corners and store its position and size. This process is continued for each incoming frame. This is illustrated in Figure 2. The right image shows the target object and serves as the template. We extract and store its SIFT feature points before the tracking begins. The left image is one frame from the camera. Blue lines link the corresponding feature points in the two images. After obtaining the homography matrix, all four edges of the object are located and they are labelled by four different colours for visual effect. The interest window containing the object is labelled by a cyan rectangle.

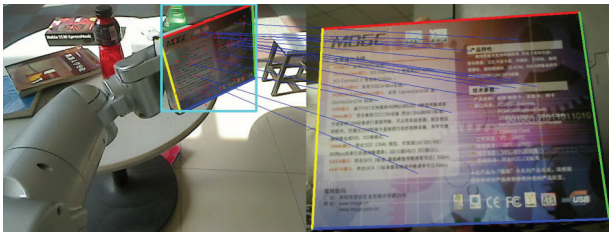

Figure 2. One frame of the tracking process.

6. Experiment

To verify the effectiveness of our method, we conducted some experiments in our laboratory. The environment and the robot arm are shown in Figure 3. As shown in Figure 3, to demonstrate our method's ability to cope with a cluttered environment we put some books, boxes and bottles on the table and there is strong sunshine from the right side. The robot arm is part of the robot SmartPal IV provided by Yaskawa Electric Corporation. The arm has 7 DOF and the gripper has one. The camera is Logitech pro 9000 and the image size is 640 by 480 pixels. Our computer's CPU is Intel E4600, with 2GB ram. The graphical card is NVIDIA GeForce 8500GT. We realize our method with C++ on the base of siftgpu [18] and utilizing OpenCV. Figure 4 shows some frames from the tracking video. In the video, the robot picks up the target object, holds it for ten seconds and then puts it down.

Figure 3. The experiment environment and the robot arm.

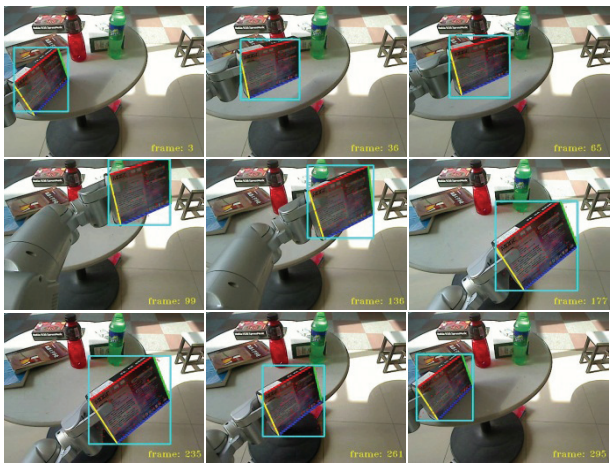

Figure 4. Frames from the tracking video.

With the same video, we compare our method with that of [11] and that of [18], which are named as fern and siftgpu respectively. Figure 5 shows the processing time for each frame of the three methods.

From Figure 5, we can see clearly that fern performs better than siftgpu, while our method performs better than fern. The average time of the three methods are listed in Table 1. According to our test, the computational complexity of our proposed method is $O(n)$, where n is the square image size.

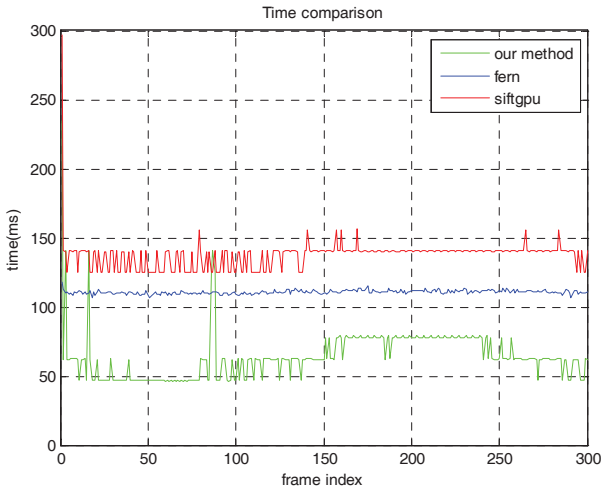

Figure 5. Time comparison of the three methods.

method	our method	fern	siftgpu
average time (ms)	64.3667	110.7633	137.7000

Table 1. Average time of the three methods

To show our method's robustness against occlusion, we made another grasping test. In some frames of the test, the target object is only partially visible either because the object is occluded by other objects, or because part of the object goes out of the camera's field of view. Figure 6 shows some frames of the tracking video. From Figure 6 we can see that our algorithm can correctly track the object even when only half of the object is visible.

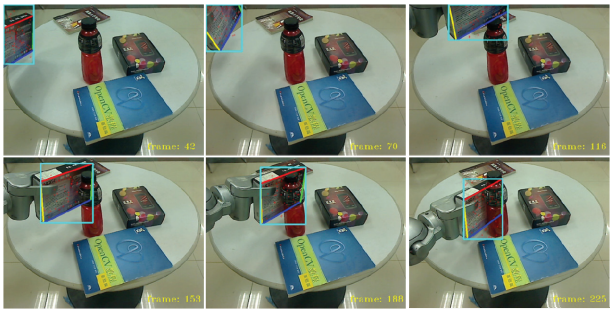

Figure 6. Frames showing our method's robustness against occlusion.

For our tracking case, the perturbation mainly comes from occlusion. To show the perturbation, we define the following error, which is commonly used in object detection and tracking:

$$e = 1 - \frac{A_p \cap A_g}{A_p \cup A_g} \tag{7}$$

where A_p is the area of the boundary box predicted by our method, A_g is the area of the ground truth bounding box, $A_p \cap A_g$ is the interaction of A_p and A_g and $A_p \cup A_g$ is the union of A_p and A_g. Figure 7 shows the error of each frame for the video used in Figure 6.

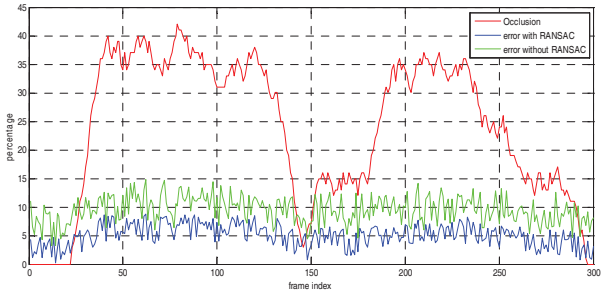

Figure 7. Tracking error w/ and w/o RANSAC.

From Figure 7 we can see that more occlusion results in a larger error as a whole. The error without RANSAC is on average 4.7% larger than the error with RANSAC.

7. Conclusion

In this paper we present a fast feature points-based object tracking method that is especially powerful in confronting the challenges of robot grasp, such as a cluttered environment, partial occlusion and scale, viewpoint or illumination changes. With the SIFT feature point correspondences and the homography constraint, we estimate the object's image position and find a window containing the object. Then we concentrate on this window, extract the SIFT features from this window only and update the window's size and position in each frame to continue this tracking process. The whole tracking procedure is completely automatic and can recover by itself when the track is lost. Experiments show that our method outperforms the methods in [11] and [18] in real-time performance. Although this method is equally effective in common object tracking tasks as well as that of robot grasp, our method requires that the object is rich in texture. For textureless objects or objects with specular surfaces, our method is inclined to fail.

8. Acknowledgements

This research is partially funded by Yaskawa Electrical Corporation. We would like to express our grateful thanks to Yaskawa Electrical Corporation for providing the robot arm and gripper of SmartPal IV. Special thanks go to Mr. Masaru Adachi for all the kind help.

9. References

[1] Collet, A., Berenson, D., Srinivasa, S. S. & Ferguson, D. (2009) Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation, *IEEE International Conference on Robotics and Automation*: 3534-3541.

[2] Yilmaz, A., Javed, O. & Shah, M. (2006) Object tracking: A survey, *ACM Journal of Computing Surveys*, 38: 1-45.

[3] Lowe, D. (2004) Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, 60: 91-110.

[4] Bay, H., Ess, A., Tuytelaars, T. & Van Gool, L. (2008) Speeded-Up Robust Features (SURF), *Computer Vision and Image Understanding*, 110: 346-359.

[5] Lepetit, V. & Fua, P. (2006) Keypoint recognition using randomized trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28: 1465-1479.

[6] Heymann, S. et al. (2007) SIFT Implementation and Optimization for General-Purpose GPU, *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*: 317-322.

[7] Cornelis, N. & Van Gool, L. (2008) Fast Scale Invariant Feature Detection and Matching on Programmable Graphics Hardware, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*: 1013-1020.

[8] Zhang, N. (2010) Computing Optimised Parallel Speeded-Up Robust Features (P-SURF) on Multi-Core Processors, *International Journal of Parallel Programming*, 38: 138-158.

[9] Ekvall, S., Kragic, D. & Hoffmann, F. (2005) Object recognition and pose estimation using colour cooccurrence histograms and geometric modeling, *Image and Vision Computing*, 23: 943-955.

[10] Kragic, D. et al. (2005) Vision for robotic object manipulation in domestic settings, *Robotics and Autonomous Systems*, 52: 85-100.

[11] Ozuysal, M., Calonder, M., Lepetit, V. & Fua, P. (2010) Fast Keypoint Recognition Using Random Ferns, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32: 448-461.

[12] Mikolajczyk, K. & Schmid, C. (2005) A performance evaluation of local descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27: 1615-1630.

[13] Nister, D. (2004) An efficient solution to the five-point relative pose problem, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26: 756-770.

[14] Hartley, R. & Zisserman, A. (2003) Multiple View Geometry in Computer Vision, 2nd ed. UK: Cambridge Univ. Press.

[15] Lowe, D. (1999) Object Recognition from Local Scale-Invariant Features, *International Conference on Computer Vision*: 1150-1157.

[16] Beis, J. S. & Lowe, D. G. (1997) Shape indexing using approximate nearest-neighbour search in high-dimensional spaces, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*: 1000-1006.

[17] Fischler, M. A. & Bolles, R. C. (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, 24: 381-395.

[18] Wu, C. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT), 2007, available at: http://cs.unc.edu/~ccwu/siftgpu