Fast Hierarchical Template Matching Strategy for Real-Time Pose Estimation of Texture-Less Objects

Chaoqiang Ye, Kai Li, Lei Jia, Chungang Zhuang, and Zhenhua Xiong(✉)

State Key Laboratory of Mechanical System and Vibration, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
{649182890,leica8244,jerryjia,cgzhuang,mexiong}@sjtu.edu.cn

Abstract. This paper proposes a fast template matching strategy for real-time pose estimation of texture-less objects in a single camera image. The key novelty is the hierarchical searching strategy through a template pyramid. Firstly, a model database whose templates are stored in a hierarchical pyramid need to be generated offline. The online hierarchical searching through the template pyramid is computed to collect all candidate templates which pass the similarity measure criterion. All of the template candidates and their child neighbor templates are tracked down to the next lower level of pyramid. This hierarchical tracking process is repeated until all template candidates have been tracked down to the lowest level of pyramid. The experimental result shows that the runtime of template matching procedure in the state-of-the-art LINE2D approach [1] after applying our fast hierarchical searching strategy is 44 times faster than the original LINE2D searching method while the database contains 15120 templates.

Keywords: Template matching · Template searching · Real-time · Pose estimation · Hierarchical searching · LINE2D

1 Introduction

Object recognition and pose estimation in three-dimensional space plays a significant role in robotic applications that range from home-service robots to intelligent manipulation and assembly. In most of industrial applications, pose of objects ought to be estimated precisely in advance so that they can be operated by a robotic end effector. Typically, industrial objects are texture-less and usually have a highly reflective surfaces, together with changing conditions in lightning, scale, rotation or occlusion, which significantly increase the difficulty in object recognition and pose estimation.

In recent years, many researchers around the world have paid their attention to the recognition of textured objects for which discriminative appearance features can be extracted. For 2D image, SIFT [2], SURF [3], ORB [4], FREAK, [5] and HOG [6] are the most popular features which is invariant to changes in scale, rotation and illumination. The position and orientation matrix of the objects in the scene can be computed by the pairs of feature points between the object model and the scene. However, when applied to texture-less objects, discriminative appearance features detectors typically

N. Kubota et al. (Eds.): ICIRA 2016, Part I, LNAI 9834, pp. 225–236, 2016.
DOI: 10.1007/978-3-319-43506-0_19

fail to extract enough feature points to establish the object model and find the corresponding feature points in the scene.

Template matching is a very popular approach related with recognition and pose estimation of texture-less objects. In this approach, object images in various poses are captured in advance to establish the object template database. For online recognition, we can sweep sliding windows of several discrete sizes over the entire scene image with a small pixel step and search for a match against all stored object template images in the database. Different from the discriminative appearance, these template images do not have scale, rotation invariance. As a result, template matching approach usually needs a large amount of template images to fulfill the whole pose space, which causes poor real-time performance of this approach.

Hinterstoisser et al. [1] have proposed an efficient template matching based method named LINE2D/LINE3D/LINE-MOD to recognize the pose of texture-less objects. The LINE2D uses the image gradients cue only, while LINE3D uses the surface normals cue in depth image only, and LINE-MOD uses both. This method follows the template matching procedure to get initial estimated pose corresponding to the template with highest matching score. Then this initial estimated pose is used as the starting iterative pose for subsequent pose refinement with the Iterative Closest Point (ICP) algorithm [7]. With data structures optimized for fast memory access and a highly linearized and parallelized implementation using special SSE hardware instructions, the method can achieve very fast matching for every single template to the scene. As a result, this method is capable of real-time matching with thousands of templates to get the initial estimated pose. The total time is the sum of time consumed both in template matching procedure and ICP algorithm. The less difference between initial estimated pose and the real pose, the less computational time by the ICP algorithm to get convergence. However, the precision of the initial estimated pose depends on the distribution density of the sample template. The real-time template matching performance is expected to degrade noticeably for a large object database, since its time complexity is linear with the number of loaded templates in the database. In [8], Hinterstoisser et al. have pointed out that the LINE-method needs about 3000 templates per object to fulfill the entire recognition space with the scale step of 10 cm and the rotation step of 15°. This template distribution is sparse and limits the precision of the initial estimated pose.

In this paper, we propose a fast hierarchical search strategy based on the basic template matching procedure in LINE2D to reconcile the conflict between the precision of initial estimated pose and the number of templates. With this strategy, we can find the template that is most consistent to the ground truth in a hierarchical way rather than go through all templates in the dataset one by one, which benefits us for significantly shortening the runtime required in template matching and helps us to estimate object pose more precisely in a larger template database without the loss of real-time performance. This fast hierarchical search strategy can also easily extend to LINE3D and LINEMOD to speed up the template matching and promote the precision of estimated pose.

The outline of this paper is as follows: Sect. 2 gives a review of the background knowledge and notations related with LINE2D method. In Sect. 3, fast hierarchical template matching strategy is introduced in detail. We have applied the proposed strategy to real experiments to recognize a common industrial workpiece and estimate its pose. The results are showed in Sect. 4. Finally, Sect. 5 draws some conclusive remarks.

2 Background and Notations

2.1 Robust Similarity Measure

A template T can be defined as a pair $T = (O, P)$, where O is the reference image of an object to detect and P is a feature list that specifies the detail information of every single feature and its location relative to reference image O. Steger [9] proposed a similarity measure which is robust to small translation and deformations to compare a template with a region at location c in a scene image I:

$$\varepsilon_{steger}(I, T, c) = \sum_{r \in P} |\cos(ori(O, r) - ori(I, c + r))| \tag{1}$$

where $ori(O, r)$ and $ori(I, c + r)$ are the gradient orientations at location r in image O and $c + r$ in image I respectively. Based on this idea, Hinterstoisser et al. [1] proposed a more efficient similarity measure by using the maximum over a small neighborhood:

$$\varepsilon(I, T, c) = \sum_{r \in P} \left(\max_{t \in R(c+r)} |\cos(ori(O, r) - ori(I, t))| \right) \tag{2}$$

where $R(c + r)$ defines the neighborhood of size T centered on location $c + r$ in the scene image. By finding this maximum, the local neighborhood of each gradient in the template and scene image gets better aligned.

The similarity score S of the reference image O at the location c in image I is defined as follows:

$$S(I, T, c) = \frac{\varepsilon(I, T, c)}{\mathrm{size}(P)} \tag{3}$$

where size(P) is the number of features in feature list P. By finding the similarity score S which is exceeding a given threshold, the matching position of a template in an image is found.

2.2 Computing the Gradient Orientations

Just as mentioned in [1], LINE2D method computes the gradients on three channels of every single pixel in the input image separately and only take pixels whose maximum gradient norm exceeds a given threshold into consideration. These pixels can be regarded as edge pixels and represented as follows:

$$C(x)\colon \ \underset{C \in \{R,G,B\}}{\arg\max} \left\| \frac{\partial C}{\partial x} \right\| > t \tag{4}$$

where *R, G, B* are the RGB channels of the corresponding colorful input image and *t* is the related threshold.

However, the number of pixel points that pass the criterion expressed in Eq. (4) is usually much larger than that of pixel points required to represent edges. Consequently, the thick edges in the input image are detected, which increases the robustness of algorithm to small deformations and scale difference to a certain extent on one hand. On the other hand, it also magnifies the disturbance aroused by the cluttered surroundings and causes some false positive results. In order to detect edge pixel and get its corresponding gradient orientation precisely, we utilize CANNY [10] operator to detect edge pixels in advance. Then we compute the gradient orientation map $Ig(x)$ at edge pixel detected by CANNY operator which satisfies Eq. (4):

$$Ig(x) = ori(C(x)) \tag{5}$$

We only consider the orientation rather than the direction of gradient in edge pixels to quantize the orientation map and divide the orientation space into 8 equal subspaces as shown in Fig. 1.

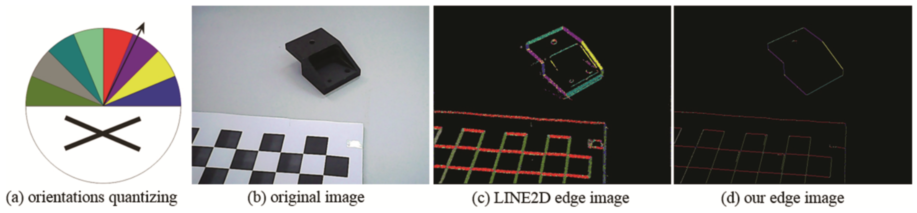

Fig. 1. **(a):** Quantizing the gradient orientations. Every colorful sector represents one orientation. The black arrow direction belongs to the third orientation. **(b):** A scene image which contains a common black industrial part and a calibration pattern. **(c):** The edge pixels and their corresponding orientation detected by LINE2D. Different color in edge pixels represents different orientations. Obviously, the detected edge is very thick. **(d):** The edge pixels and their corresponding orientation detected with our method. The precise thin edges are detected successfully. (Color figure online)

2.3 Spreading the Orientations

In order to avoid applying the max operator in Eq. (2), LINE2D adopts a new binary representation of the gradients around each edge pixel location to construct orientation map. We use a binary string with 8 bits to represent the gradient orientation and each individual bit of this string corresponds to one quantized orientation. Then we can spread each gradient orientation within a neighborhood around every major gradient orientation in the input scene image to obtain a new orientation map. The spreading operation is to spread the central orientation to every pixel location in the neighborhood and can be computed quickly by using the logical *OR* operation. With this spreading operation, the template matching method increases its robustness to scale changes and rotations. The neighborhood range depends on the magnitude *T* of the size. This magnitude *T* is useful for our online fast hierarchical searching. See more details in Sect. 3. Figure 2 shows the spreading process in detail when $T = 3$.

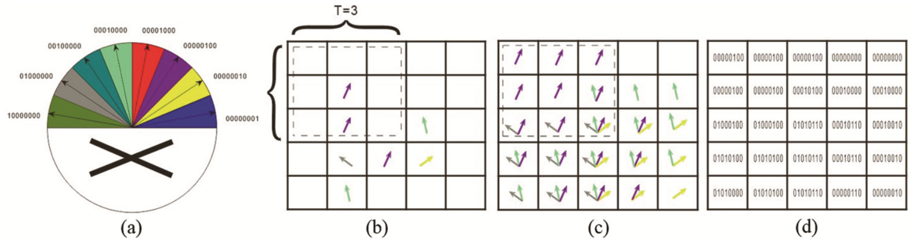

Fig. 2. Spreading process with $T = 3$. **(a)** The binary representation of every orientation. **(b)** The major orientations detected in the input scene image before spreading. **(c)** The orientations in the input scene image after spreading. The spreading operation is occurred in the neighborhood with size $T = 3$ of every major orientation. **(d)** The corresponding binary representation of the orientations after the spreading operations. The spreading operations can be computed quickly with logical *OR* operations by using this binary representation.

3 Fast Hierarchical Template Matching Strategy

3.1 Hierarchical Template Database Generation

The hierarchical template database is automatically generated from CAD model of the object to be detected. Every single template image is created by placing a virtual camera around the CAD model at a given viewpoint and projecting the model into the image plane of the virtual camera. The intrinsic parameters of the virtual camera are consistent with the real camera so that the virtual template projecting process can precisely simulate the real imaging process. The viewpoints location is defined under a spherical coordinates system by giving the spherical parameters α (latitude), β (longitude) and r (radius). To simplify the representation of pose and create a more vivid demonstration of the

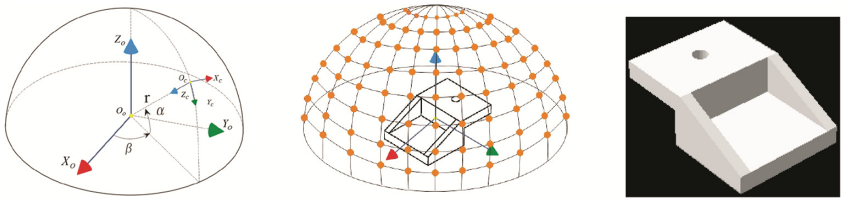

Fig. 3. **Left:** Spherical coordinates system of viewpoints location. The pose of objects in object coordinate system (X_o, Y_o, Z_o) with respect to the camera coordinate system (X_c, Y_c, Z_c) is defined by sphere radius (r), latitude (α) and longitude (β). **Middle:** Example of viewpoints distribution at a given sphere radius r: yellow vertices represent the virtual camera centers used to generate templates. **Right:** A template image generated by our method with $r = 430$ mm, $\alpha = 36°$, $\beta = 130°$. This CAD model is the prototype of the industrial workpiece used in our experiment (See more details in Sect. 4). (Color figure online)

hierarchical template database generating process, we only take above three parameters related with poses into consideration and just set the in-plane rotation angle to zero. Figure 3 shows the definition of the spherical coordinates system and an example of viewpoints distribution at a given sphere radius r.

We generate templates by using a uniform subdivision of the pose space defined by the intervals of three parameters: sphere radius (r), latitude (α) and longitude (β). The smaller the subdivision angle and distance of the three parameters, the larger the number of templates and eventually the higher the precision of final estimated pose produced by template matching. To increase the precision of estimated pose as much as possible without degrading real-time performance, we set up a hierarchical template database to reduce the total number of templates required for each estimation. The basic ideas are as follows.

Three parameters related to the pose of template can be regarded as the three dimensions in the 3D pose space. Each pose defined by its sphere radius (r), latitude (α) and longitude (β) can be regarded as a single point in the 3D pose space. For each point, its adjacent points have similar parameters. As a result, they will have similar matching scores in the template matching. Based on this fact, we select the cubic area in the 3D pose space every 27 pose points on one level template pyramid and just keep the center pose point of the cubic area to generate one pose point on the next level template pyramid. This step achieves a substantial reduction in the number of templates from one level template pyramid to the next level. In our implementation, we only need to construct 3 level template pyramids to reduce the number of templates from 15120 to 126. Figure 4 illustrates the generation process of hierarchical template database.

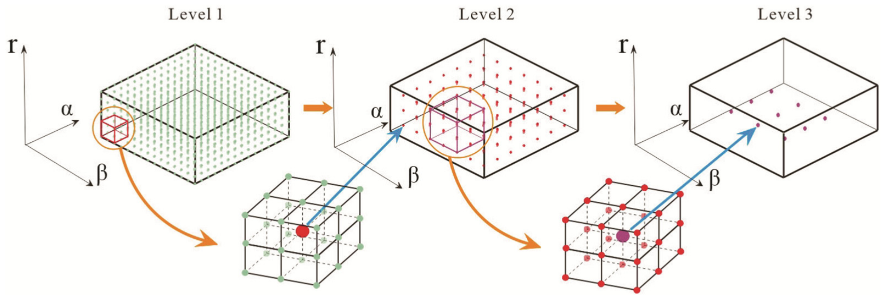

Fig. 4. Hierarchical template database generation process from level 1 to level 3. Every template pose is drawn as a point in α-β-r 3D pose space. We only keep the center point of every 27 pose points in cubic area to generate one pose point in next pyramid. The number of template points is significantly reduced after several level pyramid generation.

3.2 Online Fast Hierarchical Search

For the template pyramid generated as shown in Fig. 4, the template density of different level is different. The higher the pyramid level is, the sparser the template is. Just as Sect. 2.3 mentioned, the operation of spreading gradient orientations increases robustness to scale changes and rotations. In order to increase the robustness of the hierarchical search and to ensure the convergence of the algorithm, we propose different scene orientation images for corresponding level of template pyramid by spreading the gradient orientations with different neighborhood size T. The matching of templates on each template pyramid level is computed at its corresponding scene orientation image. We generate the orientation image with large neighborhood size T of spreading operation for templates on high pyramid level and small neighborhood size T for templates on low pyramid level. Figure 5 shows the orientation images with different neighborhood size T for corresponding level pyramid of template database.

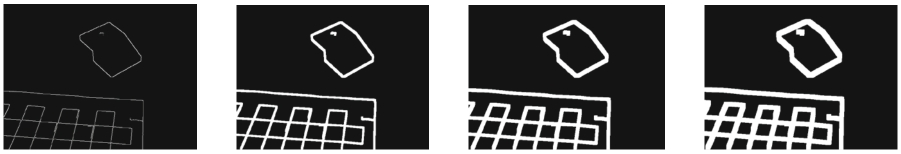

Fig. 5. Spreading the orientations with different neighborhood size T for corresponding level pyramid of template database. **(a)** Original orientation image without spreading the orientations. **(b)** Orientation image with spreading neighborhood size $T = 7$ for level 1 pyramid. **(c)** Orientation image with spreading neighborhood size $T = 11$ for level 2 pyramid. **(d)** Orientation image with spreading neighborhood size $T = 15$ for level 3 pyramid.

The online hierarchical searching is computed from the highest pyramid level to the lowest pyramid level and starts at the highest level. All templates on the highest pyramid are matched by applying the similarity measure (2) between the template and the corresponding orientation image. We set several similarity score threshold Si for each pyramid level to pick the candidate templates. The subscript i corresponds to index i of

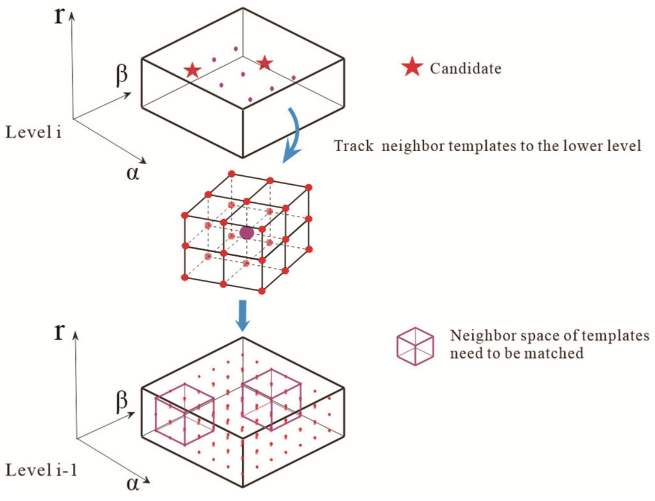
r
β
α
Level i
★ Candidate
Track neighbor templates to the lower level
r
β
α
Level i-1
Neighbor space of templates need to be matched

Fig. 6. Illustration of the fast hierarchical search strategy.

pyramid level. All templates with similarity score that exceeds its corresponding similarity score threshold Si are stored in a list of candidates. We track all of the candidate templates on the next lower level of pyramid and measure the similarity between all neighbor templates of every candidate and the corresponding orientation image. We collect all the templates whose similarity score exceeds its corresponding Similarity score threshold Si to build a new candidate list and track these candidates on the next lower level pyramid again. This process is repeated again and again until all template candidates have been tracked down to the lowest pyramid level or no templates with similarity score exceeding the corresponding Similarity score threshold are collected. Figure 6 draws the process of this hierarchical search strategy.

This kind of hierarchical searching is very fast because each tracking down to next lower pyramid only needs to match a small amount of template within the neighborhood of candidates to achieve a much higher recognition accuracy of the corresponding pyramid level. That is to say, in one of our implementation, the lowest pyramid level contains 15120 templates. If we want to get the estimated pose with the accuracy of the lowest pyramid level, we can apply this hierarchical searching strategy and only need to match a few hundred templates rather than match all of the 15120 templates one by one. Obviously, the runtime of this hierarchical searching strategy is related with the number of candidates on every pyramid level. Similarity score threshold of every pyramid level is the critical parameter which determines the searching runtime and should be chosen carefully. Smaller threshold score means that more candidates are needed to be collected on every pyramid level and the corresponding searching runtime would become longer. A good set of thresholds is that collects false candidate templates as little as possible and preserve all right candidate templates at the same time. Note that this fast hierarchical search strategy is essentially independent of the number of parameters related with pose. If we take the in-plane rotation angle into consideration, we can generate the hierarchical template database in 4D pose space whose dimensions are represented by four parameters related with pose and apply the online fast hierarchical search strategy similarly.

4 Experimental Results

We have applied the proposed hierarchical template matching strategy to real experiments to estimate the pose of 4 types of objects based on basic similarity measurements of LINE2D. The pictures and CAD model of these objects are shown in Fig. 7.

Firstly, we generate the template image uniformly in the pose range of 0–90° tilt rotation (α), 0–360° inclination rotation (β) and 500 mm–680 mm scaling (r). To cover this range, we collect 15120 templates with a tilt rotation step of 3° ($\Delta\alpha$), an inclination rotation step of 5° ($\Delta\beta$) and a scale step of 30 mm (Δr). Compared to the original LINE2D which only contains 3115 templates [8], the templates in our database is a denser distribution of viewpoints.

Our hierarchical template matching strategy is essentially by reducing the searching space to speed up the template matching process. For a given number of templates, the

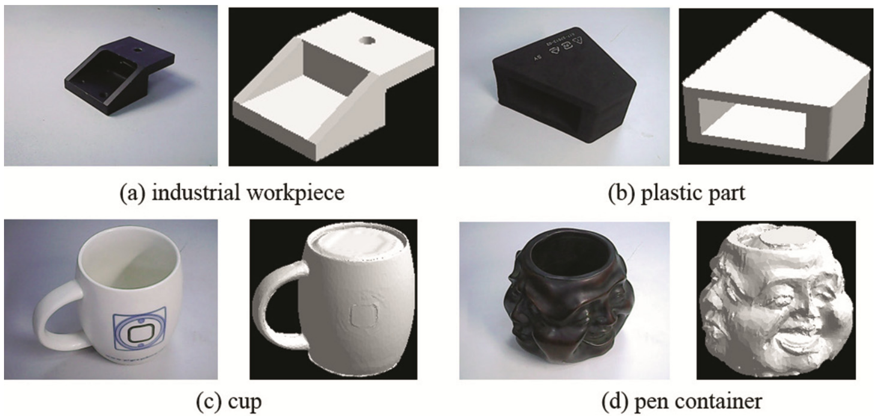

Fig. 7. 4 types of objects used in the experiments.

more layers in pyramid, the smaller the searching space. To estimate the proposed strategy performance with different size of searching space and find the suitable number of layers in pyramid, we constructed five different template pyramids with different layers to perform template matching. There are 15120 templates in the lowest layer of these five pyramid. The number of templates in the highest layer is different. There are 4 templates on the highest layer for 5 layer-height pyramid, 27 templates for 4 layer-height pyramid, 126 templates for 3 layer-height pyramid, 1620 templates for 2 layer-height pyramid and 15120 templates for 1 layer-height pyramid. Note that the template matching in 1 layer-height pyramid is actually the original strategy applied in LINE2D.

For color gradient features extraction towards each template image, we keep 63 features uniformly located on the contour of the object silhouette by adopting the gradient feature extraction strategy introduced in [8]. Furthermore, we make the most of SSE hardware instructions to compute the response maps and linearize the parallelized memory for matching each single template to scene image which is proposed in [1]. For the online hierarchical searching, we compute five scene orientation images with spreading neighborhood size $T = 7$, 11, 15, 15, 15 for corresponding layer 1 to 5 of pyramid, respectively. The Similarity score threshold Si is set to 80 %, 75 %, 60 %, 50 % and 50 % for corresponding layers to pick the candidates need to be tracked in next layer of pyramid. Considering there are only one object in every picture, we sort the candidate list and only track top 4 candidates with high similarity score on the next lower layer. This step helps set the number of templates that need to be matched for each downward-searching and stabilize the runtime in every frame to be compared with original LINE2D method.

For every object, we randomly capture 20 pictures with different poses and match the templates in these pictures through above five types of template pyramid with different height. Then, we record the average matching score and average runtime of five types of searching pyramid. Figure 8 shows the results.

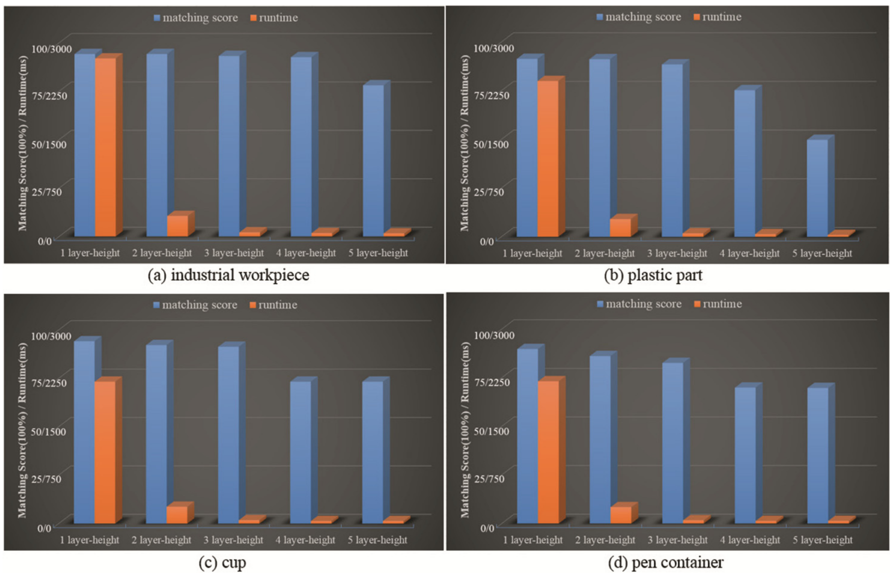

Fig. 8. Average matching score and runtime of five types of pyramid with different layers.

The pyramid with only 1 layer-height represents the original searching strategy in LINE2D by matching 15120 templates one by one, which expectedly achieve the highest matching score and longest runtime. The pyramid with 5 layer-height indicates the smallest searching space. Though the runtime of this pyramid is shortest, the matching score is also the lowest. A good trade-off between speed and robustness is 3 layer-height pyramid. Let's just take the experiments of 3 layer-height pyramid into discussion. In LINE2D searching method, we need to match 15120 templates to the scene image one by one. However, we only need to match $126 + 4 \times 27 + 4 \times 27 = 342$ templates out of 15120 templates with our hierarchical searching strategy. As a result, the average runtime of template matching procedure in LINE2D after applying our fast hierarchical searching strategy is about 44 times faster than the original LINE2D searching method while the database contains 15120 templates.

We also conduct three types of pyramid to perform another experiment to estimate the relationships between the distribution density of templates and the accuracy of estimated poses. We set the number of templates on the lowest layer of pyramid as 126, 1620, 15120 respectively and keep the number of templates on the top layer to 126. That's to say, we construct a pyramid with 1 layer-height for experiment with 126 templates, a pyramid with 2 layer-height for 1620 templates and a pyramid with 3 layer-height for 15120 templates. The object that need to be recognized in this experiment is the industrial workpiece (see Fig. 7(a)). The templates matching results by applying our fast hierarchical searching strategy in different template density databases are shown in Fig. 9.

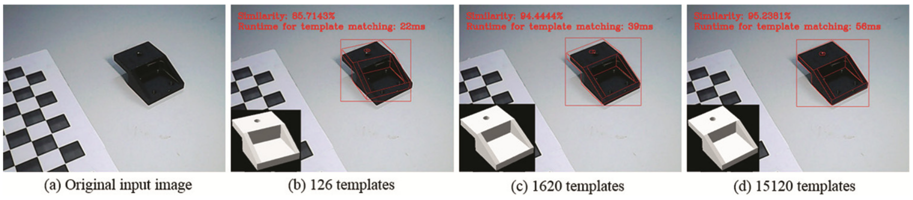

Fig. 9. Templates matching results for three kinds of databases with different template density. **(a)** The original input image for templates matching. **(b)** Templates matching result with 126 templates. The template with highest similarity score is presented in the bottom-left corner. **(c)** Templates matching result with 1620 templates. **(d)** Templates matching result with 15120 templates.

Figure 9 demonstrates that we can obtain the pose estimation results with higher accuracy by matching templates in a larger template database. Therefore, we can estimate object pose more precisely in a larger template database without the loss of real-time performance by applying our hierarchical template matching strategy.

Up to now, we didn't introduce any complicated clutter backgrounds to the experiments. Just as the explanation and discussions mentioned in [1], LINE2D will produce many false positive in cluttered environment. What we are concerned about is that how to speed up the templates matching procedure in a large database. If we are expected to recognize the pose of object in a cluttered environment, this fast hierarchical search strategy can also easily extend to a more robust algorithm named LINEMOD [1] to speed up the template matching and promote the precision of estimated pose.

The experiments were performed on standard computer with an Intel Processor Core(TM) with 3.2 GHz and 8 GB of RAM. The implementation of the original LINE2D, we use the publicly available source code in OpenCV 2.4.11.

5 Conclusions

We have proposed a fast hierarchical template matching strategy for real-time pose estimation of texture-less objects. This strategy is intended for speeding up the template matching based approaches that are applied in pose estimation of texture-less objects. We have introduced the hierarchical template database generation process and online fast hierarchical searching strategy through the template pyramid in detail. Then, we applied this strategy into the basic template matching procedure of a state-of-the-art pose estimation approach named LINE2D. In the experiment, we extracted 15120 templates to construct the templates database. The experiment result shows that the runtime of template matching procedure in LINE2D after applying our fast hierarchical searching strategy is 44 times faster than the original LINE2D searching method. In conclusion, this fast hierarchical template matching strategy can reconcile the conflict between the precision of estimated pose and the number of templates. We can construct a much denser template database which contains more templates covered in pose range to obtain more precise results without a typical loss in runtime performance by applying this fast hierarchical template matching strategy.

Acknowledgments. This research was supported in part by National Key Basic Research Program of China under Grant 2013CB035804 and National Natural Science Foundation of China under Grant U1201244.

References

1. Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., Lepetit, V.: Gradient response maps for real-time detection of textureless objects. IEEE Trans. Pattern Anal. Mach. Intell. **34**(5), 876–888 (2012)
2. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
3. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
4. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571. IEEE, November 2011
5. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: fast retina keypoint. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 510–517. IEEE, June 2012
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 1, pp. 886–893. IEEE, June 2005
7. Fitzgibbon, A.W.: Robust registration of 2D and 3D point sets. Image Vis. Comput. **21**(13), 1145–1153 (2003)
8. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) ACCV 2012, Part I. LNCS, vol. 7724, pp. 548–562. Springer, Heidelberg (2013)
10. Steger, C.: Occlusion, clutter, and illumination invariant object recognition. Int. Arch. Photogrammetry Remote Sens. Spat. Inf. Sci. **34**(3/A), 345–350 (2002)
11. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)