Model-based 3D Object Tracking with Online Texture Update

Keisuke Tateno Daisuke Kotake Shinji Uchiyama

Visual Information Technology Development Center, Canon Inc.
3-30-2 Shimomaruko, Ohta-ku, Tokyo 146-8501 JAPAN
{tateno.keisuke, kotake.daisuke, uchiyama.shinji}@canon.co.jp

Abstract

We propose a real-time 3D object tracking method robust to illumination change. Recent research studies on edge-based tracking using a 3D CAD model have mainly focused on improving robustness to rapid motion, with little attention given to illumination change. We tackle this problem by dynamically estimating the surface texture information of the 3D model of an object to be tracked, as well as its pose. The effectiveness of our method is shown through experiments in real settings.

1 Introduction

Vision-based 3D object tracking, which estimates the pose of an object in 3D space using image information obtained by camera, has various applications in a variety of fields such as machine vision and augmented reality. In this paper, we focus on tracking of artificial objects: industrial products and their parts. Tracking such objects is useful in machine vision applications such as autonomous assembly by robot. Typically, such artificial objects are poorly textured and consist of artificial line segments. Therefore, edge-based tracking [1] is suitable for such objects because artificial lines usually appear as "edges" on an image.

Edge-based tracking seems to be robust to illumination changes because an edge is defined as the only place where the intensity changes sharply and it can be stably detected under different lighting conditions. However, this invariance to illumination changes leads to a lack of identity information of an image edge. This means that it is difficult to correctly match an image edge and a 3D line segment after rapid motion, often leading to tracking failure. Some methods using additional information [2,3,4] are very robust to rapid motion, but cannot be applied in our scenario because one method can only track a moving camera on which an inertia sensor is attached [2], and others assume that the object to be tracked has a rich texture [3, 4].

In order to improve the matching performance without additional information, Wuest *et al.* proposed a method that uses local appearance information (profile) from around an image edge [5]. Reitmayr *et al.* also proposed a similar method, using the profile from around image edges in the rendered image of a 3D textured model [6]. These methods can significantly improve the matching performance. However, they are not robust to illumination change because it is assumed in [5] that the visibility of each 3D line segment never changes under different illumination conditions, and in [6] that the lighting conditions never change and therefore texture data is constant. In real indoor environments, the appearance of the object can easily change even with a slight movement of the object or the camera, because the distance between the object and the light source is short. In such environments, the visibility of each edge and its local appearance change dynamically and therefore they cannot be regarded as constant as assumed in [5] and [6].

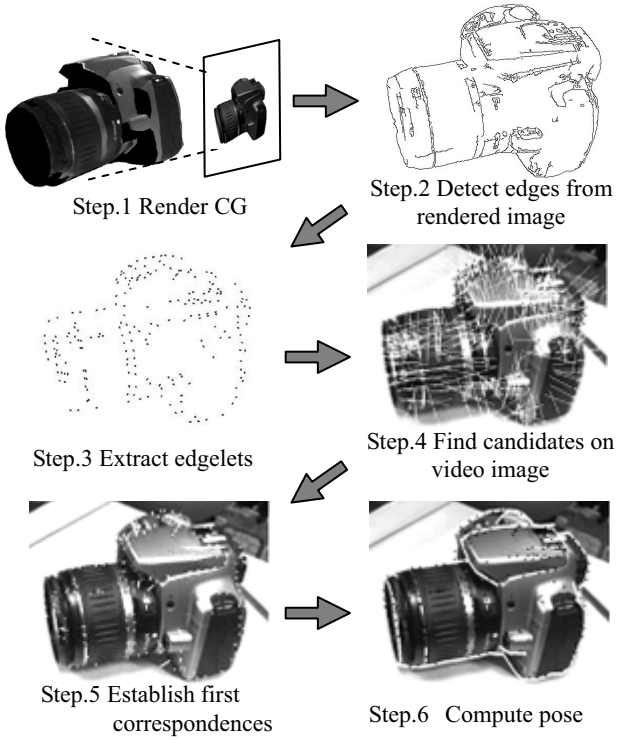

Figure 1: Pose estimation using textured 3D model.

In this paper, we propose a 3D object tracking method using a 3D CAD model composed of meshes and their texture images. To endure illumination change, our method dynamically estimates not only the pose of an object, but also the surface appearance information of the object. Different from the method in [5], the appearance information is saved in the 3D space as the texture image of the 3D CAD model.

2 Edge-Based Pose Estimation Using Textured 3D Model

In this section, we explain the 3D object tracking method based on the method described in [5]. This is the basis for our method. The 3D model of an object is a mesh model constructed from triangular patches. Each patch has a photorealistic texture image, making it possible to produce realistic CG images. It is assumed that the camera's intrinsic parameters such as focal length and

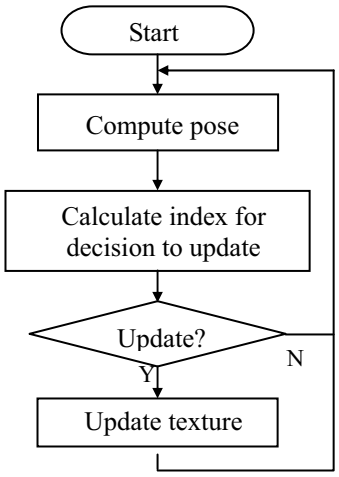

Figure 2: Flow of proposed method.

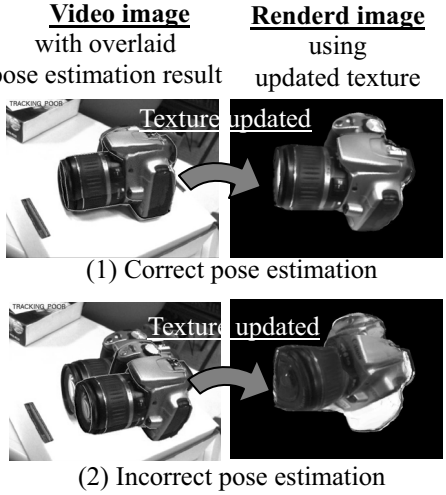

Figure 3: Texture updating affected by pose estimation results.

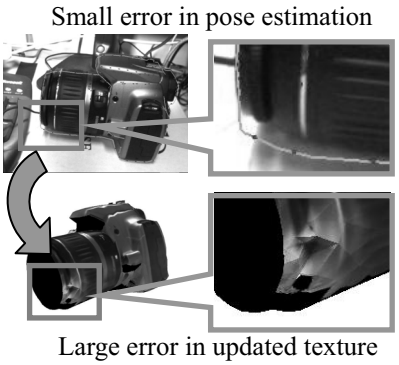

Figure 4: Texture in contour region affected by small error.

radial distortion are known. Below, we explain the pose estimation method shown in Fig. 1 step by step.

Step 1 – Step 3 "Extract 3D edgelets"

First, the CG of an object is rendered based on the pose in the previous frame (for the first frame, we use a pre-determined pose). Then, edges are detected from the rendered image using a Canny edge detector [7]. Among detected edges, some are selected so that they will be equidistant from each other in the image. Finally, the 3D position and orientation are assigned to each selected edge using the rendered depth image. Hereafter, we refer to such 3D edge having its position and orientation as "edgelet" [8].

Step 4 "Find candidates on video image"

In this step, some edges that can correspond to each edgelet are found on the video image. First, each edgelet is projected onto the image, and its 2D position and orientation are computed. Then, edges are searched on the image along the search line normal to the projected edgelet. For each edgelet, multiple detected edges are retained as candidate corresponding edges.

Step 5 "Establish first correspondences"

For each edgelet, the most likely corresponding edge is selected among the candidates using the "profile" around the edges. The edgelet profile is obtained from the rendered image as the 1D vector composed of the 15 intensity values, which is sampled along the line normal to the projected edgelet. Similarly, the profile for each candidate is obtained from the video image. Then, the correspondence is established based on the sum of squared difference (SSD) between the two profiles (one for the edgelet and the other for the candidate edge).

Step 6 "Compute the pose"

The pose of the object is calculated using an iterative calculation process. If there are at least six correct correspondences, it is possible to calculate the pose. Generally, the precision of the estimated pose increases as the number of correspondences increases. However, if some incorrect correspondences are included in the data, the precision of the estimated pose decreases, often leading to a tracking failure. In order to reduce the influence of such incorrect correspondences, an M- estimator is incorporated into the calculation process. In each iteration step, the 6D correction vector of the pose is computed in such a way as to minimize the sum of the squared 2D distances between the projected edgelets based on the pose and the edges that correspond to those edgelets. As described in [5], the correspondences obtained in Step 5 are used to compute the correction vector in the first step, whereas the correspondences are re-established in all subsequent steps by selecting the edge that is closest to the projected edgelet using the corrected pose.

3 Pose Estimation with Online Texture Updating

In this section, we propose a pose estimation method that dynamically updates the texture of the 3D model using the pose estimation results. Figure 2 shows the entire process flow of the proposed method. The flow can be divided into the following three steps:

1) Pose estimation
   The pose is estimated as described in Section 2.
2) Decision to update
   Following the pose estimation step, it is determined whether or not to update the texture of the 3D model by evaluating the estimated pose. The texture images are updated, using only those frames where the pose estimation is deemed successful.
3) Texture updating
   Based on the decision in 2), the texture images are updated using the estimated pose and captured image.

Below, we detail the Step 2) and 3) and omit the explanation of Step 1) because it is described in the previous section.

3.1 Decision to update

In our method, the estimated pose is completely trusted in the process of texture updating; therefore, if the pose is not correct, the texture images are incorrectly

updated as shown in Fig. 3. Hence, it must be first decided whether or not the pose has been successfully estimated, and then the texture images are updated only when the pose estimate is deemed successful.

To determine whether or not the pose estimate is successful, we use the *quality-of-tracking* value proposed in [6] as an index. This value is calculated based on the inlier rate of the pose estimation results. The inlier rate is the ratio of edgelets deemed to be correctly corresponding to the correct images edges, to all the edgelets created from the 3D model. The higher the index value is, the more accurately we consider the pose is estimated. The following are the specific procedures for determining whether or not to update.

1) Calculate the *quality-of-tracking* value

The inlier rate is computed as the percentage of edgelets considered to be inliers in an M-estimator among all the edgelets. Then, the *quality-of-tracking* value is calculated as described in [6].

2) Make a decision on updating

If the *quality-of-tracking* value exceeds a certain threshold, the system determines that the pose estimation is sufficiently accurate and the texture can be updated; otherwise, the system only proceeds to the pose estimation process in the next frame.

3.2 Texture updating

Our method updates the texture by mapping the pixels within the object area on the video image onto the texture image, based on the estimated pose. However, some part may not be suitable for updating. For example, the texture images that obtains pixels near the boundary between the object and the background may be significantly affected by a pose estimation error, as shown in Fig. 4. This can happen even when the pose estimation error is small. In our method, the decision to update the texture is a global decision, not caring about each local mismatch. Hence, updating texture images near the object's boundary is avoided by identifying the boundary contour of the 3D model. The following are the specific procedures for updating the texture.

1) Compute the planar homography between the video image and the texture image

First, the three vertices and centroid (center of gravity) of each triangular patch are projected onto the image plane based on the estimated pose. From these four corresponding points between the texture image coordinates and captured image coordinates, the planar homography is computed.

2) Identify boundary regions

Next, we determine if each triangular patch is located in a boundary region or not. To identify a boundary region, the depth buffer generated in the edgelet extraction step is used. A region where the value of this depth buffer changes significantly is identified as a boundary region.

3) Map the captured image onto the texture image

For patches other than boundary regions, the pixel values of the video image are mapped onto the texture image using the planar homography computed above. These procedures are carried out for each triangular patch.

4 Experiment

In this section, we evaluate the robustness of our method to illumination changes through experiments in real environments.

Overview of experiments

In this experiment, we use a Flea2 manufactured by Point Grey Research. The image size we use is 640×480. The internal parameters of the camera, including its radial distortion, are pre-calibrated. For processing, we use a desktop PC equipped with an Intel Pentium 4 3.8 GHz (CPU) and an nVIDIA GeForce7800GTX (GPU). Since the computational cost of the edgelet extraction and texture updating is too high to be done in real-time, such two processes are separated from the pose estimation process and executed in a different thread from the thread for pose estimation in our implementation. This means that the 3D edgelets used in pose estimation are different from the ones extracted in the previous frame. We measured the average frame rate of our method and the accuracy before actual experiments. Our method with the 2500 polygon model runs about at 20 fps on the PC mentioned above. The accuracy was measured for one selected frame by comparing the pose obtained by our method and the reference pose that is believed to be close to the true pose. The reference pose is obtained by manually establishing the correct correspondence between edgelets and image edges and refined a given prior pose using the method described in Section 2. The 3D positional and rotational errors are about 3*mm* and 0.2°, respectively.

Experiment on robustness to illumination change

We compare the proposed method with the method without texture updating (similar to the method in [6]). For the objects to be tracked, we use a commercial digital single-lens reflex camera and inkjet printer, which are placed against a simple, predominately white background. We use a pre-recorded sequence of images that includes a light source change. In this sequence, the object and the video camera (Flea2) move independently. We applied both methods to the same sequence. The initial texture images of the 3D models were created in advance from real images.

Figure 5 shows the results of pose estimation. The top row shows the common results of both methods in the frames before the light source changed, the middle one shows the results of the method without texture updating in a frame after the light source changed, and the bottom one shows the result of the proposed method in the same frames as in the middle row. The left two columns show the tracking results for the camera, and the right ones show the tracking results for the printer. The 3D models are overlaid onto each video image using the estimated

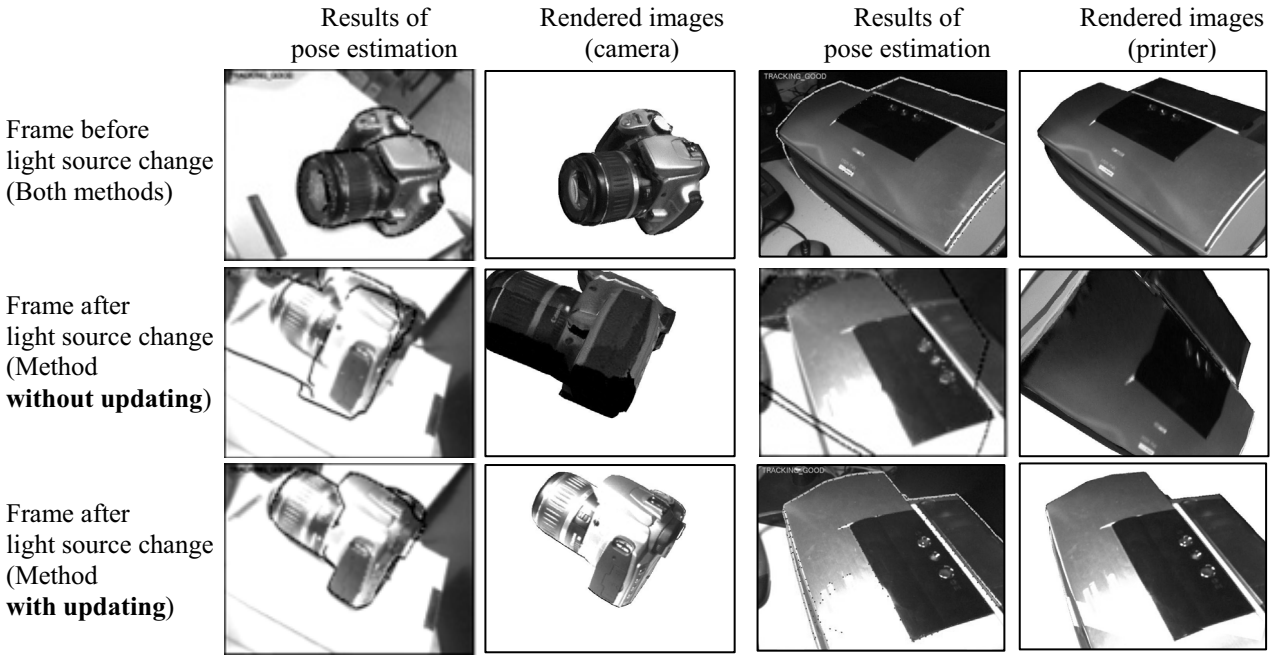

Figure 5: Results of experiments.

pose. If the rendered line segment matches the image edges of the object, the pose is considered to be accurately estimated.

Before the light source changed, the pose was correctly estimated by both methods. After the light source changed, however, the pose estimation method without texture updating failed because the rendered images of the 3D models were quite different from the object in the video images. On the other hand, the proposed method correctly estimated the pose, as can be seen by the fact that the lighting change accurately reflected on the texture images. These results show that the proposed method is robust in situations where the light source changes.

5 Conclusions

In this paper, we proposed a method that dynamically updates the 3D model as well as the pose. This method makes it possible to dynamically reflect the physical appearance changes of the object onto the 3D model by obtaining the texture images from the video images when the pose is correctly estimated. A comparative experiment under conditions of illumination change proved the validity of the proposed method.

The method proposed in [5] also updates the appearance information of the 3D model as in our method. Two methods differ in that the appearance information in [5] is held in 2D image space, while our method holds it in 3D space on the object. Since the 3D appearance information can be projected onto an image more correctly than 2D appearance information, the performance of matching of a 3D edgelet and its corresponding edge can be better than [5].

One of the future challenges is to cope with the error accumulation on the texture images. Since the error is always included in the estimated pose, the error on the update texture images accumulates as the texture is updated. This problem cannot be avoided as far as the timated pose is completely trusted in texture update. To solve this problem, it is necessary to introduce the pose estimation framework based on some absolute indices. For example, some 3D lines which is almost always salient and not updated can be used as absolute indices. Otherwise, as described in [9], only the brightness change information is updated while the texture information itself is fixed. The method in [9] is computationally complex compared to our method, but it is a good suggestion for future improvement.

References

[1] A. I. Comport, E. Marchand, and F. Chaumette, "A real-time tracker for markerless augmented reality," *Proc. ISMAR'03*, pp. 36-45, 2003.

[2] G. Klein and T. Drummond, "Tightly integrated sensor fusion for robust visual tracking," *Proc. BMVC'02*, pp.787 -796, 2002.

[3] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," *Proc. ICCV'05*, pp.1508-1515, 2005.

[4] M. Pressigout and E. Marchand, "Real-time 3D model-based tracking: combining edge and texture information," *Proc. ICRA'06*, pp.2726-2731, 2006.

[5] H. Wuest, F. Vial, and D. Stricker, "Adaptive line tracking with multiple hypotheses for augmented reality," *Proc. ISMAR'05*, pp.62-69, 2005.

[6] G. Reitmayr and T. W. Drummond, "Going out: robust model-based tracking for outdoor augmented reality," *Proc. ISMAR'06*, pp.109-118, 2006.

[7] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. on PAMI,* vol.8, no.6, pp.679-698, 1986.

[8] E. Eade and T. Drummond, "Edge landmarks in monocular SLAM," *Proc. BMVC'06*, pp.7-16, 2006.

[9] H. Yang, G. Welch, and M. Pollefeys, "Illumination insensitive model-based 3D object tracking and texture refinement, " *Proc. 3DPVT'06*, pp.869-876, 2006.