Mechanical assembly assistance using marker-less augmented reality system

Yue Wang, Shusheng Zhang, Sen Yang, Weiping He and Xiaoliang Bai

Key Laboratory of Contemporary Design and Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, Xi’an, China

Abstract

Purpose – This paper aims to propose a real-time augmented reality (AR)-based assembly assistance system using a coarse-to-fine marker-less tracking strategy. The system automatically adapts to tracking requirement when the topological structure of the assembly changes after each assembly step.

Design/methodology/approach – The prototype system’s process can be divided into two stages: the offline preparation stage and online execution stage. In the offline preparation stage, planning results (assembly sequence, parts position, rotation, etc.) and image features [gradient and oriented FAST and rotated BRIEF (ORB)features] are extracted automatically from the assembly planning process. In the online execution stage, too, image features are extracted and matched with those generated offline to compute the camera pose, and planning results stored in XML files are parsed to generate the assembly instructions for manipulators. In the prototype system, the working range of template matching algorithm, LINE-MOD, is first extended by using depth information; then, a fast and robust marker-less tracker that combines the modified LINE-MOD algorithm and ORB tracker is designed to update the camera pose continuously. Furthermore, to track the camera pose stably, a tracking strategy according to the characteristic of assembly is presented herein.

Findings – The tracking accuracy and time of the proposed marker-less tracking approach were evaluated, and the results showed that the tracking method could run at 30 fps and the position and pose tracking accuracy was slightly superior to ARToolKit.

Originality/value – The main contributions of this work are as follows: First, the authors present a coarse-to-fine marker-less tracking method that uses modified state-of-the-art template matching algorithm, LINE-MOD, to find the coarse camera pose. Then, a feature point tracker ORB is activated to calculate the accurate camera pose. The whole tracking pipeline needs, on average, 24.35 ms for each frame, which can satisfy the real-time requirement for AR assembly. On basis of this algorithm, the authors present a generic tracking strategy according to the characteristics of the assembly and develop a generic AR-based assembly assistance platform. Second, the authors present a feature point mismatch-eliminating rule based on the orientation vector. By obtaining stable matching feature points, the proposed system can achieve accurate tracking results. The evaluation of the camera position and pose tracking accuracy result show that the study’s method is slightly superior to ARToolKit markers.

Keywords Tracking, Augmented reality, Assembly, Marker-less, Line-mod, ORB

Paper type Research paper

1. Introduction

Augmented reality (AR) is a type of human–computer interaction technology that enhances the natural visual perception of a human user by means of computer-generated information (i.e. 3D models, annotation and text; Azuma, 1997). To augment the real world with this information seamlessly, the camera pose must be calculated. The techniques that achieve this goal can be classified into three groups: marker-based, sensor-based and marker-less tracking.

Marker-based tracking has been widely and deeply researched (Ng et al., 2013; Wang et al., 2013; Zhu et al., 2013). Westerfield et al. (2015) used ARToolkit markers to track the motherboard models so that the manipulator could assemble the real element in the correct position. Uchiyama and Marchand (2011) adopted random dot markers for tracking. Their approach was robust against occlusion and user interaction. Wang et al. (2005) explored tracking using color-based non-square visual markers for assembly guidance. Most prototype systems of AR used this marker-based technique for its low computational cost and good performance even under bad illumination. However, marker-based tracking approaches also suffer a certain drawback, namely, visual pollution or tracking jitter owing to the absence of adequate feature points.

In addition, external markers are not permitted for many industrial applications.

Sensor-based tracking is another popular tracking method (You and Neumann, 2001), but it can be easily interfered with in industrial scenarios. For instance, magnetic sensing tracking
Marker-less tracking is seen as a desirable tracking approach for AR assembly, which has undergone impressive improvements in recent years (Crivellaro et al., 2015; Mottaghi et al., 2015; Engel et al., 2014), such as feature point-based methods (Pauwels et al., 2013; Pressigout and Marchand, 2006; Kim et al., 2003), edge-based methods (Tombari et al., 2013; Damen et al., 2012; Payet and Todorovic, 2011), model-based methods (Wang et al., 2015; Hinterstoisser et al., 2012b; Prisacariu and Reid, 2012) and simultaneous localization and mapping (SLAM) methods (Engel et al., 2014; Caruso et al., 2015; Mur-Artal and Tardós, 2014). However, feature point-based methods are prone to tracking jitter or turbulence because mechanical parts for assembly are textured poorly. Edge-based methods are sensitive to cluttered background and not robust to occlusions during the interaction process. Model-based methods have been widely used for marker-less tracking in AR-based assembly, but retrieving data from a large number of reference images to find the key frame leads to fast searching space and heavy computational load, which greatly reduces real-time performance and system availability. SLAM-based methods are limited to estimating a relative camera pose only, which is not suitable for AR-based assembly; moreover, they are prone to tracking failure on dynamic scenes. Additionally, owing to the characteristic that the geometry appearance of assemblage changes with assembly processes, most existing marker-less tracking methods are no longer practical. Therefore, an immense body of research has focused on developing a specific ad hoc tracking method.

In this paper, we present a fast and robust marker-less tracking approach for AR mechanical assembly. The system generates instructions by superimposing a virtual model on the real parts, indicating what the next step is and the way it assembles. The online assistance provided can reduce the time spent by technicians searching for information in paper-based documentation, and thus, it enables faster and more reliable operation.

The main contributions of our work are as follows: First, we present a coarse-to-fine marker-less tracking method that uses modified state-of-the-art template matching algorithm, LINE-MOD (Hinterstoisser et al., 2012a), to find the coarse camera pose. Then we use a feature point tracker Oriented FAST and Rotated BRIEF (ORB) (Rublee et al., 2011) to calculate the accurate camera pose. The whole tracking pipeline needs, on average, 24.35 ms for each frame, which can satisfy the real-time requirement for AR assembly. On the basis of this algorithm, we present a generic tracking strategy according to the characteristic of assembly and develop an AR assembly assistance platform. Second, we present a feature point mismatch-eliminating rule based on the orientation vector. By obtaining stable matching feature points, our system can achieve accurate tracking results. The evaluation of the camera position and pose tracking accuracy results show that our method is slightly superior to ARToolKit markers.
using fiducial markers. However, for propagating the established coordinate systems, the workspace must be a flat surface that provided at least a dominant plane, which limited wider application. Alvarez et al. (2011) designed a hybrid tracking method that combined an edge tracker, a point-based tracker and a 3D particle filter to update the camera pose continuously. The AR disassembler could run in real-time and deal with scenes that lacked texture. Chen and Li (2010) developed a keypoint tracking method that used a FAST algorithm to extract keypoint features and descriptors. To improve the robustness, a Kanade–Lucas–Tomasi (KLT) tracker was added to deliver additional information for pose estimation. Nevertheless, their method could not distinguish different objects, an important issue for AR assembly system.

Although the above methods demonstrated a good tracking performance on their specific systems, most of them can easily result in tracking jitter or turbulence owing to the lack of significant texture information on mechanical parts. Moreover, to realize 3D tracking, feature-based methods also need to process a large amount of reference images obtained from multiple views in real time to acquire the accurate camera pose. Additionally, most of the above methods are fast enough to process one frame, but relying only on a feature-based method can hardly meet the real-time requirement of augmented reality owing to the vast searching space and heavy computational load.

3. Overview of the study’s approach

In this paper, we propose a marker-less tracking approach, and based on this methodology, an AR assembly assistance system is presented. The system’s process can be divided into two stages: the offline preparation stage and online execution stage.

In the offline preparation stage, we use CAD software for assembly sequence, generation as previously described (Barnes et al., 2004). In this process, the position and rotation of each part to be assembled is defined by the base part, which is the first node of the generated sequence. Moreover, assembly sequence, position and rotation of each part and tool associated with the project are all stored in an XML file. At the same time, multi-views after each assembly-planning step are generated automatically, and image features are extracted and stored in XML files (Section 3.1). In the online execution stage, the XML file of the assembly sequence is first parsed to identify the next part to be assembled and tools to be used, and then the image features of the base part are extracted and matched with those generated offline. In this manner, the camera pose is calculated (Section 3.2) and guiding information that loaded from the assembly management platform based on those parsing results is superimposed on the base part. Finally, the operator can follow this assistance information and install the part to the correct position on the base element. However, after the part is assembled on the base, the geometry of the assembly changes and the tracking process may fail. Therefore, we propose a tracking strategy according to the characteristic of assembly, when upon pressing the “Next Step” button the XML file that stores the image features changes accordingly (see Figure 1).

3.1 Offline preparation

After each step of assembly planning, multi-view images (called “reference images” later in this paper) are generated automatically using the virtual camera in the CAD software, which is much more convenient and efficient compared with an online learning method (Kalal et al., 2010). Furthermore, for modern manufacturing, 3D models often exist before the real objects are created, so requiring 3D models beforehand is not a disadvantage anymore.

The reference image sampling rules of our method in the CAD software are as follows. The assemblage is placed at the center of the icosahedron and sampled on each vertex of the triangles with a virtual camera (see Figure 2). To obtain all possible topology images of the assemblage, each triangle of an icosahedron is divided into four equilateral triangles and iterated several times. In our experiment, for a good trade-off, we stop at 91 vertices on the upper hemisphere. As a result, two adjacent vertices are less than 20 degrees apart. The virtual camera aims at the center all along the sampling process; it covers the pose range of $0^{\circ}$–$360^{\circ}$ around the assemblage, and $0^{\circ}$–$90^{\circ}$ tilt rotation. To overcome the lack of rotation invariance of the LINE-MOD algorithm (Hinterstoisser et al., 2012a), four angles of in-plane rotation between $-45^{\circ}$–$45^{\circ}$ are formed; therefore, 455 views are obtained.

For each reference image generated by the method described above, we compute its orientations of gradient, as described (Hinterstoisser et al., 2012a), and use standard ORB (Rublee et al., 2011) to extract 2D feature points and compute their 3D positions on the object model by back-projection method (Lepetit et al., 2005). Finally, reference images labeled with corresponding image features and poses are sorted in XML format for later use.

3.2 Camera-pose calculation

In the online execution stage, image features are extracted and matched with those generated offline for camera pose estimation. To realize fast and accurate pose estimation, a coarse-to-fine tracking strategy is proposed. First, we use a modified template matching method LINE-MOD to obtain the reference image that is similar to the current viewpoint and then an accurate pose refinement method using a point tracker is presented.

3.2.1 Camera-pose rough estimation

The idea of this section is that the key frame and its corresponding rough camera pose can be obtained through template matching. In our research, we base our work on the LINE-MOD algorithm, which has good precision recall performance and computes feature vectors very fast. The LINE-MOD algorithm proposes a similarity measure that, for each gradient orientation on the object, searches in a neighborhood of the associated gradient location for the most similar orientation in the test image. It can be summarized in equation (1):

$$
\varepsilon(I, T, c) = \sum_{r \in R} \max_{\mathbf{v} \in \mathbf{R}^{(i + r)}} |\cos(\text{ori}(O, r) - \text{ori}(I, t))|
$$

where $\text{ori}(O, r)$ is the gradient orientation in radians at location $r$ in a reference image $O$ of an object to detect, and
ori(I, t) is the gradient orientation at location t in input image I. 
\[ \tau \in R(c + r) = \left[ c + r - \frac{\tau}{2}, c + r + \frac{\tau}{2} \right] \times \left[ c + r - \frac{\tau}{2}, c + r + \frac{\tau}{2} \right], \]
which defines the neighborhood of size \( \tau \) centered on location \( c \) shifted by \( r \) in the input image \( I \). This setup makes the method robust to small shifts and deformations. 
p denotes the list of locations \( r \) to be considered in \( O \); therefore, template \( T \) is defined as a pair \( T = (O, p) \).

LINE-MOD uses a gradient feature for matching. At each location of a given image, LINE-MOD computes the gradient orientation of each color channel and picks the orientation with the greatest gradient magnitude. Then, the algorithm quantizes the gradient orientations into eight equal spacing to represent each gradient orientation in 8 bits. Moreover, LINE-MOD speeds up this similarity calculation by spreading gradient orientations and pre-computing response maps.

In the execution of a template matching process, a LINE-MOD model first learns a list of templates for each object. Then, the algorithm searches through the input image with a sliding window, which contains a matching object, if the similarity score between the gradient feature matrix of this window and that of a template is above a certain threshold. The preliminary bounding box is then determined by using the center of sliding window and the bounding box of the template. At the end of this search, non-max suppression is performed on all potential bounding boxes with an overlap threshold of 0.5. The remaining bounding boxes are the locations of the predicted objects.

However, the problem with the original LINE-MOD method is that although it is invariant to small distortions of images, it is not inherently scale invariant, and only works at \( 1 \pm 2 \) scale. For instance, if the object in the training image is fairly far away from the camera, while in a testing image it is much closer to the camera, then even if the object in the testing image has the same orientation as that in the training image, its gradient features around a certain location will be far more
spread out than those in the training image. This causes difficulties for the template-matching problem. Our goal for the following modification is to make LINE-MOD scalable by using depth information obtained from Softkinetic. The similarity measure equation is as follow:

\[
\mathcal{S}(c_0, \gamma) = c_0 + D(c_0) y
\]

(2)

where \( ori(O, S_0(c', \gamma')) \) is the gradient orientation in radians at location \( S_0(c', \gamma') \) in a reference image \( O \) of an object to detect. \( ori(I, t) \) is the gradient orientation at location \( t \) in input image \( I \). \( c' \) defines the shift from the center point of the matching object to the origin of the input image \( I \). \( c_0 \) defines the shift from the center point of the object to be matched to the origin of reference image \( O \). \( \gamma' \) denotes the shift from the patch center and \( D(c_0) \) is the depth value at location \( c_0 \), which can be acquired from a Softkinetic depth sensor.

In equation (2), we incorporated depth information obtained from Softkinetic and scaled the computed gradient feature matrix using this depth data. As objects are trained on a half-dome with fixed positions, the distance between the objects and camera in the training stage is known. Therefore, according to the depth information, the scale ratio of the template can be determined (Figure 3). We then store the scaled gradient feature matrices in each template and perform template matching with the input image. During this process, a sliding window is used to slide the input image for the similarity calculation between the target object and template. In the sliding window, we traverse through all locations in the location list \( P \) and sum up the similarity at each location. When the similarity value reaches maximum, this reference image is identified as a key frame. Our method enlarged the work space or scale space of LINE-MOD algorithm.

3.2.2 Camera-pose refinement

After template matching, the key frame similar to the current input image is obtained. To refine the camera pose further, a feature-based tracking method is incorporated into our algorithm. The feature descriptors of the input image are matched with that of the key frame. As a result, the corresponding 2D/3D point set is obtained, which provides the necessary initialization data for precise camera position computation.

Figure 3  Template matching process using the improved LINE-MOD algorithm

Notes: (a) Template extraction, the blue points represent the gradients that extracted from (b) and (c) template matching at different object scales; (d) template matching failed using the original LINE-MOD algorithm
Mechanical assembly assistance

Yue Wang et al.

Owing to the lack of adequate textures, point feature matching methods such as those found in a few studies (Lowe, 2004; Mikolajczyk and Schmid, 2002) often fail. Therefore, corner features on the assemblage are used in this paper. ORB is an excellent descriptor, which is a fusion of a FAST corner detector and a BRIEF descriptor with many modifications to enhance the performance. Readers can check other references (Rublee et al., 2011) for detailed information. The corresponding 2D/3D points are obtained by matching 2D feature points between the input image and the key frame. Correct points matching result is the most essential premise for pose estimation.

However, in practical use, mismatch happens between partial point pairs. To eliminate mismatch from noisy data and find the correct match for camera-pose calculation, hypothesis testing strategy, random sample consensus (RANSAC), is first used for preliminary screening, and then orientation vector of matching-point pair is adopted to further eliminate outliers. The orientation vector outlier rejection method can be seen as follows.

First, according to the characteristics of mechanical components, density-based spatial clustering method is used to cluster the matching point pairs. Then, for each matching-points cluster, the following steps are conducted.

\[ M = [m_1, m_2, m_3, \ldots, m_n]^T \]

The similarity between the direction vectors \( m_i \) and \( m_j \) is measured using the inner product:

\[ d_{ij} = \langle m_i, m_j \rangle \]

The similarity between \( m_{(1:n)} \) and all elements in \( M^{T}_{(2, n)} \) is defined as follows:

\[ G = m_i M^T / (n - 1) \]

All the similarity results in \( G \) are sorted in a descending order; if one element in \( G \) is smaller than a given threshold value \( \delta \) (in practice we use 0.985) the matching-point pair is treated as outlier.

In this paper, point pairs obtained from our outlier rejection method is compared with those obtained using the RANSAC method. The results can be seen in Figure 4. There are lots of mismatching points in Figure 4(a). After using the RANSAC algorithm, mismatching points reduce significantly but still exist [Figure 4(b)]. Figure 4(c) shows our method, which uses RANSAC and orientation vector to remove the outliers. It can be seen that one additional mismatching point pair is further removed, which makes the point pairs matching process more accurate. Besides, the time spent on the orientation vector outliers removal process is less than 2 ms.

After the above procedure, the correct matching-point pairs are obtained. Then, the perspective-n-point (PnP) algorithm is used to calculate the accurate camera pose. However, a traditional iterative PnP algorithm such as Lu Hage Mjolsness (LHM) algorithm (La et al., 2000) has very high computational complexity, which greatly reduces the efficiency of the algorithm. Besides, the calculation accuracy of the iterative methods degrades significantly when no redundant reference points can be used (\( n \leq 5 \)).

4 Experimental results

4.1 Experimental setup

Our experiments were implemented on a standard notebook with an Intel(R) Core(TM) i3-4130 CPU with 3.4GHz and 4GB of RAM. A Softkinetic DS325 depth sensor was used, which has a color sensor (resolution: 720P) and a depth sensor (resolution: 320 × 240). In our experiments, the image size was set to 640 × 480. The system setup can be seen in Figure 5. The software was developed based on the game engine “unity3D”, and virtual elements such as 3D models were generated by the 3D Studio Max modeling software. In this study, the tracking algorithm was written in C++ language and imported to unity3D as a Dynamic Link Library (DLL).

4.2 Assembly assistance experiments

In this section, we developed a precision machine tool assembly assistance system based on our marker-less tracking method. The first-person view of the machine tool assembly-guiding
task can be seen in Figure 6. When touching the “Start” button, the camera pose was calculated using our marker-less tracking method, while a 3D model of the part to be assembled was loaded from the remote server and superimposed on the real assembly scene to indicate where and how to assemble the part. The manipulator could, then, follow this guiding information and assemble the part in the correct position. After completing this step, the process is to trigger the “Next” button for the following assembly assistance information. If the manipulator wants to see this information again, pressing the “Prev” button would generate an animation replay. During this assembly process, our method still can automatically tracking when the part of interest changes its appearance [Figure 6(c)].

4.3 Evaluation of the algorithm
4.3.1 Scale-invariance evaluation
To evaluate the impact of depth factor on scale invariant property of our tracking method, a set of comparison tests were conducted and the results presented in Figure 7. Figure 7(a) to (d) represents tracking using our method, with the adaption of a depth factor, as described in Section 3.2.1. Figure 7(e) and (f) represents tracking without the above adaption. As can be seen from Figure 7(a) to (d), the scale of the object in the frame changes, but a good tracking performance is still achieved. This was mainly because of the fact that the LINE-MOD algorithm was adapted into a scale invariant descriptor with depth information, and its working range was expended, which could help to obtain the most similar key frame quickly with the current input frame through template matching; thereby, the coarse camera pose could be estimated. After that, an ORB descriptor of scale invariant property can help to calculate the accurate camera pose based on the key frame. However, owing to exceeding the working range of the original LINE-MOD algorithm (1~2 scale), the coarse camera pose estimation process cannot be completed, so the tracking process fails abysmally [Figure 7(e) and (f)].

4.3.2 Tracking accuracy evaluation
To evaluate the tracking accuracy, the most commonly used marker system ARToolKit was adopted for comparison with our method. It was reported by Abawi et al. (2004) in the International Symposium on Mixed and Augmented Reality (ISMAR) that ARToolKit showed high tracking accuracy with 30~80° in rotation, 20~90 cm in distance. Therefore, we conducted the comparative tests within those ranges. In the experiments, the camera was mounted on a high-precision mechanical pan and tilted and adjusted in a way that the center of the lens and the center of the marker were at the same height, as was previously done (Abawi et al., 2004). We adopted the “one-factor-at-a-time” approach. Here, the distance of the camera to the marker was changed from 20 up to 90 cm in steps of 10 cm, while the other input factor rotation angle was maintained at 35°. In another set of experiments, the angle of the camera was changed only around the y-axis in the range from 30° up to 80° in steps of 10°, while Z distance was maintained at 30 cm. The experimental results can be seen in Figure 8. As can be seen in Figure 8(a) and (b), the overall errors of translation and rotation of both our method and the ARToolKit increased with the increasing Z distance, but the ARToolKit’s overall errors of was greater than ours. From Figure 8(c) and (d), we can see that the overall errors of translation and rotation of the ARToolKit increase with the increase in the rotation angle. However, the translation and rotation errors in our method remain unchanged when the rotation angle increased. From the above four figures, we conclude that our tracking method has an advantage over the ARToolKit in the accuracy of position and pose tracking.

To prove the effectiveness of our proposed outlier point pair rejection method, a supplementary experiment was also conducted. Method 1 represents our method without orientation vector rejection process. From Figure 8, it is indicated that the orientation vector outlier rejection process plays a positive role in camera position and pose tracking accuracy.

4.3.3 Time analysis
Table 1 shows the computation time of our proposed method. From Table 1, we can see that the template matching process to find the key frame needs 41 m. After the key frame is obtained, the tracking speed can run at 30 fps until the matching points fall below a thread (in practice we use 5 matching points). Even upon considering the template matching process, our method can meet the real-time requirement (≥15 fps) for an AR system.

5. Conclusions and future work
Mechanical assembly using AR technology has been widely reported in the literature, but most are based on artificial markers, which suffer from not only visual pollution but also complex operation defects. In this paper, we proposed a coarse-to-fine marker-less tracking strategy. First, we adapted the LINE-MOD algorithm into a scale-invariant descriptor to find a key frame labeled with its corresponding pose; then, feature-based tracking method using an ORB descriptor was exploited to refine the pose. To eliminate mismatching points in feature-based tracking, an orientation vector outlier rejection method was presented, which could further remove outlier-
mismatching points after using the RANSAC algorithm in lesser computational time. An AR assembly assistance system was developed with our proposed marker-less tracking approach. The assembly result showed that our method automatically adapted to tracking requirement when the part of interest “changes” its appearance. In addition, the tracking accuracy and real time property of the algorithm were investigated. The results showed that our tracking method could run at 30 fps and
that the camera position and pose tracking accuracy was slightly superior to ARToolKit. Because of the working distance and FOV range of depth sensor Softkinetic, the working scope and user experience of our approach is limited. In our future research work, we will dedicate our efforts to solving the scale invariant problem of LINE-MOD with an image processing approach. Furthermore, 3D feature descriptors will be explored to improve the robustness of our method.

References


Mechanical assembly assistance

Yue Wang et al.


Further reading


Corresponding author

Shusheng Zhang can be contacted at: zssnet@sina.cn