Natural Feature Tracking Augmented Reality for On-Site Assembly Assistance Systems

Rafael Radkowski and James Oliver

Iowa State University, Virtual Reality Applications Center,
1620 Howe Hall, Ames, IA, 50011, USA
{rafael,oliver}@iastate.edu

Abstract. We introduce a natural feature tracking approach that facilitates the tracking of rigid objects for an on-site assembly assistance system. The tracking system must track multiple circuit boards without added fiducial markers, and they are manipulated by the user. We use a common SIFT feature matching detector enhanced with a probability search. This search estimates how likely a set of query descriptors belongs to a particular object. The method was realized and tested. The results show that the probability search enhanced the identification of different circuit boards.

Keywords: Augmented Reality, Natural Feature Tracking, Assembly Assistance.

1 Introduction

An assembly assistance system is a computer terminal, which provides assembly work instructions such as the assembly sequence, the components needed for a product, the handling of tools, etc. They are located at assembly stations on a factory floor and are commonly used in a variety of industries. These systems are critical for novice assemblers who typically refer to them regularly. However, even experienced assemblers are required to use assembly assistance systems because product variants are difficult to memorize and these systems are also used to track production efficiency. Most assembly assistance systems are comprised of simple alphanumeric lists of instructions with perhaps links to associated 2D schematic drawings. To enhance the effectiveness of such systems, we developed an Augmented Reality (AR) assembly assistance system for a major manufacturer of electrical components that superimposes a live video image of a manual assembly station with 3D models, 2D texts, and annotations. It tracks the parts to assemble, shows the assembly sequence, and provides information about the assembly method.

The assembly assistance system must track multiple rigid objects; in our case, planar printed circuit boards that are manipulated by the user (Figure 1). Therefore, we have developed a natural feature tracking (NFT) system. It relies on the so-called SIFT feature tracker [1]: feature maps are created that represent the objects to track. To identify and track an object, features need to be identified in a video stream and

R. Shumaker (Ed.): VAMR/HCII 2013, Part II, LNCS 8022, pp. 281–290, 2013.

compared with the stored feature database. This and similar techniques are well known for image tracking. For example, today they are employed in magazines and newspapers.

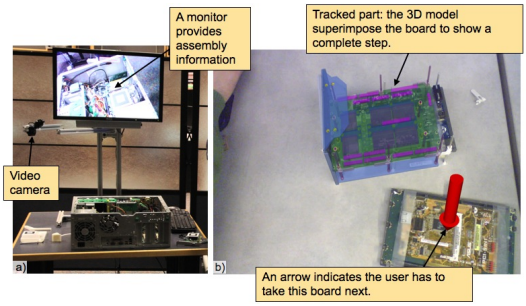

Fig. 1. a) A prototype of the AR on-site assistance system. b) The application tracks multiple circuit boards and adds assembly instructions.

Despite its ubiquity, the SIFT method presents several challenges when tracking multiple circuit boards. In general, circuit boards are difficult to track: they look similar to on another, provide only a limited number of good feature points, and they are 3D objects, even though they are flat.

Our contribution is a method to distinguish multiple circuit boards and to identify the related feature map in real time as well as an optimized feature map structure for rigid object feature maps. We employ a probability search that estimates how likely a set of given query features belongs to a particular circuit board. The method utilizes statistical similarity of features and clustering methods to calculate a probability value. We use a tree data structure to compare all relevant feature maps on a different level of complexity.

The next section reviews the relevant related work that drives our method. In section 3, we present our realization of the tracking system and explain the probability search. We present an application example in section 4 and close the paper with a summary and an outlook.

2 Related Work

In general, natural feature tracking (NFT) is a vision-based tracking approach. According to Zhou [2], vision-based tracking can be classified in model-based tracking techniques and feature-based techniques. This classification considers the amount of previous knowledge the tracking system needs to have about the scene. NFT belongs to the feature-based techniques, which relies on natural features. A large share of research is devoted to NFT for AR applications since AR relies on tracking and NFT facilitates the usage of physical objects in the environment. Thus, the review will only highlight some research that fosters our approach.

Lepetit et al. [3] introduced a keypoint-based tracking method that automatically builds different view sets of a training image in order to improve performance and robustness. Multiple keypoints are extracted from these images and stored as a classification database. They use a randomized *kd*-tree to classify the feature points of a sample image. The method works robustly, it facilitates tracking of a wide range of images, and also copes with cluttered and distorted objects. Nevertheless, it is trained for only one object.

Klein et al. [4] has developed a method that simultaneously estimates the pose of a camera and creates a feature map. The idea is to split tracking and mapping into two different threads. This enables the use of computationally expensive optimization methods in order to build an optimized feature map. The approach is robust and works with a large set of keypoints. Nevertheless, it is intended for navigation purposes and cannot identify particular objects.

Chen et al. [5] have developed a keypoint tracking system that copes with different lighting conditions. The authors employ a FAST algorithm to extract keypoint features and descriptors. The descriptors are organized in a *kd*-tree for fast keypoint retrieval. To improve the robustness, a Kanade-Lucas-Tomasi (KLT) tracker [6] has been added that delivers additional information for pose estimation. This enhances the probability of obtaining good features to track. The method utilizes an additional matching algorithm to improve the robustness. Nevertheless, their method does not distinguish different objects.

Cagalaban et al. [7] introduce a tracking method that allows tracking of multiple 3D objects in unprepared environments. The method incorporates a KLT tracking and color tracking to detect multiple moving objects. However, the authors' test objects were relatively simple (cars), object segmentation relies on background separation and the tracked objects cannot be identified.

Uchiyama et al. [8] present a tracking method that relies on a method called locally likely arrangement hashing. The authors intend to track 2D maps, which are difficult to track because the arrangement of a map looks similar from different viewpoints. Their tracking approach utilizes the intersections on maps to retrieve a robust feature map. In addition, the authors use online learning to be able to cover a large map.

The Fours Eyes Lab conducted research in keypoint optimization and keypoint selection in order to optimize the keypoint database in such a way that only the best, most robust, features maintain tracking (i.e., [9], [10]). For instance, they explore the effect of different texture characteristics on tracking. They also evaluate the influence of different tracking parameters using a large database of 2D images that show different light conditions and geometric changes. Their research is aimed at developing a robust tracking system. Nevertheless, the research does not address circuit boards, particularly, nor in particular, the problems associated with tracking physical tracking targets with keypoints.

3 Natural Feature Tracking for Circuit Boards

The objective of the tracking system is to determine the pose of a video camera to enable spatial registration for AR. Our tracking system relies, like many others, on matching keypoints from training images with keypoints obtained from a run-time video stream. Usually, all training keypoints are stored in one database, and the query set is matched against this database. The challenge, when tracking physical objects like circuit boards, is to receive a sufficient number of correctly matched feature points to enable pose estimation and tracking. Since circuit boards roughly look similar, the number of false matches increases with the overall number of features. In this section, we first explain the suggested tracking method. Afterwards, we introduce our feature map optimization strategy.

3.1 Feature Point Tracking and Matching

Figure 2 presents an overview of the natural feature tracking method. The method implements the functions keypoint extractions, descriptor computation, descriptor matching, and pose estimation. Our implementation is based on OpenCV [11], an open source computer vision library that provides the required core image processing functionality. In addition, a rendering function generates the output image. However, since the rendering function is not part of part of the tracking system, it is not explained.

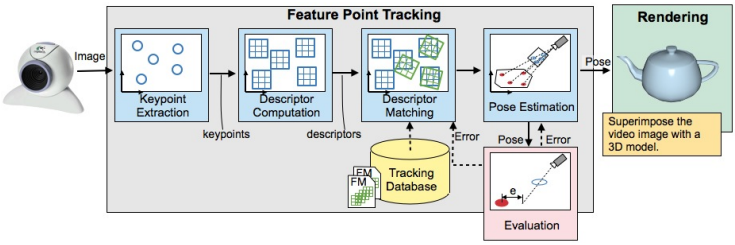

Fig. 2. Overview about the entire natural feature tracking system

Consider a feature map $F_i = \{k_1, ..., k_N \mid d_0...d_N\}$, with N keypoints $k \in K_i$ and N associated descriptors $d \in D_i$. Each set F_i enables tracking of a physical object O_i, with i, the index of the object, also referred to as tracking targets. We use SIFT feature points and descriptors [1]. A keypoint describes the location of a feature. The descriptor is a vector with 128 values that represent the magnitude and the orientation of gradient vectors that surrounding the keypoint. A training database DB_{ref} with $DB_{ref} = \{F_0, \ldots, F_N\}$ stores all feature keypoints K_N and descriptors D_N of all physical objects to track.

For object tracking, let $I(x, y)$ be the input image fetched from a video camera. First, we identify a set of keypoints K^* and extract keypoint descriptors D^* in I. Initially, the keypoints and descriptors are unidentified; they may belong to one or to multiple tracking targets. To identify them, descriptors D^* are matched against the reference descriptors $D_N \in DB_{ref}$. A k-nearest-neighbors method (KNN) is a

common way to match data. The descriptors in DB_{ref} are organized as a randomized *kd*-tree [12]. A *kd*-tree is a space-partitioning data structure [13]. It splits *k*-dimensional data into half-spaces considering the variance of each dimension. Randomized *kd*-trees use a limited number of dimensions to split the state-space of data. The *kd*-tree is trained in advance. The nearest neighbors are retrieved by traversing the *kd*-tree.

We added a probability search in order to facilitate the tracking of circuit boards. During our work on the assembly assistance station, with increasing number of descriptors in the database we encountered too many false-matches. The probability value is added in order to eliminate a set of descriptors. Thus, the number of descriptors that need to be considered is reduced. Therefore, we use a limited number of descriptors with a limited complexity. Figure 3depicts an abstraction of the tree structure spaces utilized. The nodes carry the descriptors $d_{i,j}$, with i, the tracking target identifier, and j, the descriptor number. Each node splits the search space into half-spaces. The upper nodes of the tree use the probability search. A probability value estimates how likely a descriptor belongs to a particular feature map.

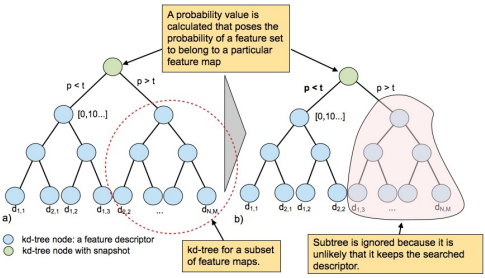

Fig. 3. kd-trees are used for a fast k-nearest neighbor search. An additional probability search allows determination of a subset of descriptors that likely belong to the searched feature map.

Let d_t^* be the query descriptor that needs to be matched. First, a probability value is calculated. We compare the vectors of each descriptor with a feature map signature $S = \{r_0, r_1, ..., r_N\}$ and the signature centroid Sc_0 and Sc_1. The signature is a composed descriptor vector that contains the most robust gradient vector direction and magnitude ranges $r = \{r_{min}, r_{max}\}$ of a feature map. The centroids are the center of the majority of descriptor directions. The probability value is calculated using:

$$\forall d_t^* \in D^* : \Delta s = min(\frac{|Sc(d_t^*)_i^* - Sc_0|}{var(S)}, \frac{|Sc(d_t^*)_i^* - Sc_1|}{var(S)}) \tag{1}$$

with Δs, the distance value, and Sc_i, the centroid of the descriptor d_t^*, and *var(S)*, the variance of the signature. The calculation is carried out for all feature maps in the database. Using this approach, one descriptor can be associated to more than one feature map. Thus, we consider all query descriptors to calculate the probability value by counting the possible assignment of a descriptor to a particular feature map:

$$P_i = \frac{1}{N} \sum_{j=0}^{N} k | \begin{cases} k = 1 & if \Delta S_i \in FM_i \\ k = 0 & if \Delta S_i \notin FM_i \end{cases} \tag{2}$$

With *P* the probability value, *N*, the number of descriptors. This method facilitates the determination of outliers. All descriptors of the training database that belong to a feature map with a low probability value are not considered in the following step.

The probability search is carried out only when the ratio between matched and unmatched descriptors of a scene is below a threshold *T*. We empirically determined a *T* = 0.4 considering three circuit boards at once in the scene. If a major set of descriptors cannot be matched, this ratio indicates that a new circuit board is in the scene. As long as the tracking quality is acceptable we assume that only the identified subset is needed. The *kd*-tree was adapted to consider the probability value; we remove the nodes and retrain the sub-tree.

To match the feature d_t^*, we employ a *k*-nearest-neighbor (KNN) matching to find the best match in the training database [14]. The KNN method is a non-parametric method. It calculates *k*-distances for each vector of the input data to the reference data, where *k* represents the number of neighbors the method returns when calculating the output distance. We calculate the *k*=2 distances, thus, for each query descriptor we find the two best matches in the reference dataset D_{ref}:

$$dist_i(d_i, d_{ref}) = \sqrt{(d_{i,1} - d_{ref,1})^2 + \cdots + (d_{i,N} - d_{ref,N})^2} \tag{3}$$

$$y = \frac{dist_1 + dist_2}{2} \tag{4}$$

with $dist_i$, the distance between each input descriptor and reference descriptor, and *y*, the final output calculated from the two best matches $dist_1$ and $dist_2$. The KNN method returns the two nearest neighbors for each query descriptor; the matching set is denoted as *M*. We utilize the OpenCV implementation of the Fast Library for Approximate Nearest Neighbors for KNN matching [15].

Next, a ratio test is employed to find the best matches of the entire output set *M*. The ratio test checks whether the matches found violate a threshold [6]:

$$r < r_t \tag{5}$$

$$r = \frac{dist_1}{dist_2} \tag{6}$$

With *r*, the ratio, and r_t, the threshold. Usually, the ratio ranges from 0.4 to 0.6. Only high quality matches pass this test. All other matches are deleted.

Finally, we employ an epipolar search to find feature points that meet the fundamental epipolar constraint of a 3D projection: all keypoints of the query set must lie on the epipolar line of the reference keypoint set. We use an 8-point RANSAC algorithm to calculate them. The RANSAC algorithm is an iterative technique to estimate the parameters of a model from a set of given input points [16]. The inliers of the RANSAC test comply with this constraint and are used to estimate the pose.

To estimate the pose, we calculate the extrinsic camera parameters using a textbook technique, namely, Direct Linear Transformation (DLT), solved with Singular Value Decomposition [15]. The DLT uses corresponding keypoints of the tracking targets and the image to solve a camera model equation. Figure 4 presents a result. The left image shows the training image and its keypoints. The right image depicts the video stream and the matching keypoints. The lines indicate the corresponding keypoints.

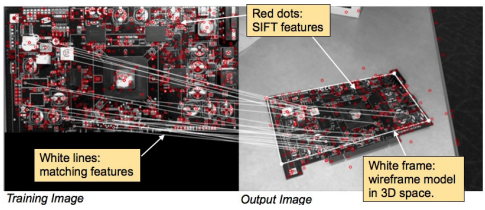

Fig. 4. Tracking of a circuit board with SIFT feature tracking

3.2 Feature Map Signature

The feature map signature is a vector with the most robust descriptors. It facilitates fast recognition and classification of feature descriptors, which are determined by counting. A reference image is used and its feature points and descriptors are extracted. To get query images, multiple affine transformations are applied. Also noise is added to several images and the resolution is changed. The feature descriptors of these images are extracted and matched against the reference image descriptors. Each match in the reference image is counted. After N samples (N=400 for our tests) a matching count for each reference descriptor available. For the further processing, we select 20% of the most matching features when they occur in 50% of all frames.

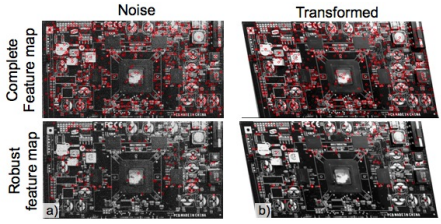

Fig. 5. Feature maps after optimization

The idea of the probability search is to determine a probability value for each query descriptor considering the majority of all robust descriptors of a feature map and their variance. Therefore, we calculate a signature and the feature map centroids. Both are determined using a k-means clustering method. Let $D = \{D_0, D_1, D_2, ..., D_N\}$ be the set of all robust descriptors with $D_i = \{d_0, d_1, d_2, ..., d_{127}\}$. First we cluster the descriptors with the k-mean clustering with k=2:

$$argmin \sum_{i=1}^{k} \sum_{d_j \in D} |d_j - \mu_i|^2 \tag{8}$$

with μ, the mean of the descriptor values. The clustering results in a set of descriptors, which are most similar, and the cluster centroids. Figure 6 shows a result generated from three circuit boards[1].

[1] The circuit boards used for testing are computer graphics cards, network adapters, etc. The application has been developed for circuit boards of controllers. The board layout cannot be published due to a confidential declaration.

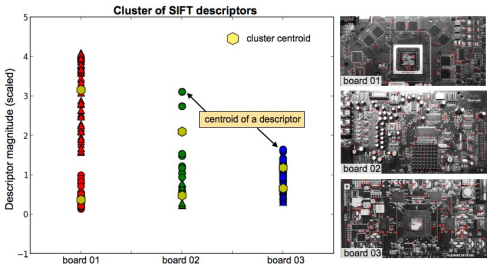

Fig. 6. Cluster of centroids of descriptor magnitudes

This approach relies on different cluster centroids and different descriptor distributions that result in different variances. Thus, we can use a statistical approach to anticipate a best match. As shown in Figure 6, the clusters are distinguishable. Each of the clusters poses one signature *S*. The centroids are used as mean values μ for each feature map. As presented before, Equation 1 and 2 are used to calculate the probability of new descriptors to belong to a particular feature map.

The signature S of a feature map is a 2xN vector that keeps the range of the majority of all descriptor values per descriptor element:

$$S = \{s_0, s_1, s_2, \ldots, s_N\},\ with\ s_i = (d_{max}, d_{min}) \tag{9}$$

with d_{max} and d_{min}, the min and max values for every descriptor element d_i.

4 Results

We compared the tracking performance with a regular natural feature tracking that works without a probability search. The same SIFT feature tracker was applied and the *k*-nearest neighbor method was used for feature matching. For repeatability, the tests have been carried out with testing software and images of circuit boards and not with a live video and manual operated camera. We used eight computer-related circuit boards (graphics card, network adapter etc.) as reference data. The query dataset used eight images of the same boards plus eight additional colored advertisements from journals. All query images have been automatically transformed (size, rotation, perspective, shear) in order to get five query images for each sample. All sample images have been matched against the training database. The tests have been carried out on a Dell Precision T3500 computer, with an Intel Xeon 3.47 GHz processor, NVidia Quadro 5000 GPU, and 6GB RAM.

Table 1 shows the results. The first column shows the number of tracking targets in the database, the second the overall number of descriptors in the database. The next three columns, the percentage of matched features and the percentage of false matched features for the standard KNN algorithm. The ratio is the quotient of false to all matches. The last three rows show the same for suggested method.

Table 1. Results from feature matching experiments

No. DB	Overall descriptors	KNN	ratio	KNN false	KNN + P	KNN + p false	ratio
1	145	95%	0%	0%	94%	0%	0%
2	538	94%	5%	5%	90%	4%	4%
4	2211	75%	59%	44%	80%	13%	16%
6	3689	72%	76%	55%	79%	10%	13%
8	4946	82%	60%	49%	82%	7%	9%

The results show that the optimized descriptor matching algorithm yields a higher matching ratio when the number of tracking targets, i.e., the number of descriptors in the database, is increased. The usual matching algorithm provides too many false matches, matches that belong to different tracking targets. The subsequent applied data tests cannot remove all of these false matches. Thus, the pose estimation does not work correctly. The probability method helps to remove data from the database before the KNN matching is applied. Thus, the number of wrong matches were reduced.

However, we also used both methods to track regular 2D color images. In this case that the suggested algorithm does not yield significant advantage. Regular color images produce many more feature descriptors than circuit boards do. The feature descriptors are also more distributed; a major descriptor direction often does not exist. Thus, the *k*-means clustering cannot identify descriptor clusters because the variance of descriptors is similar. In general, circuit boards provide fewer good feature descriptors than color images do. Thus we assume that cluster centroids and a majority of descriptor directions and magnitudes can be identified.

5 Summary and Outlook

This paper introduces an augmented reality on-site assembly assistance system that helps assemblers during the assembly process. The system identifies and tracks the objects to assemble and provides assistance if necessary. For our purpose, the assembly assistance system needs to track and to identify physical planar circuit boards. We encountered problems with regular approaches when more than two circuit boards must be tracked. Circuit boards do not provide a sufficient number of distinguishable features points. Thus, we enhanced the identification probability by introducing a probability value. It relies on the sparse descriptor set of circuit boards: it is possible to identify clusters and to distinguish boards using the cluster centroids. It can be used to enhance the probability of identifying the correct feature keypoints and, finally, the correct tracking target.

However, the method is limited to tracking targets with a sparse descriptor set. We conducted a limited set of experiments. We know, the approach works for our system and the product we need to track. In the future, we seek to analyze the overall capability of the approach presented. Therefore, we will work on a larger dataset for training and query.

References

1. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
2. Zhou, F., Duh, H., Billinghurst, M.: Trends in Augmented Reality Tracking Interaction and Display: A Review of Ten Years of ISMAR. In: International Symposium on Mixed and Augmented Reality (ISMAR 2008), Cambridge, UK, vol. 2008, p. 10 (July 2008)
3. Lepetit, V., Fua, P.: Keypoint Recognition using Randomized Trees. Transactions on Pattern Analysis and Machine Intelligence 28(9), 1465–1479 (2006)
4. Klein, G., Murray, D.: Parallel Tracking and Mapping for Small AR Workspaces. In: Proc. International Symposium on Mixed and Augmented Reality (ISMAR 2007, Nara) (2007)
5. Chen, Z., Li, X.: Tracking based on Natural Feature for Augmented Reality. In: International Coriference on Educational and Information Technology, ICEIT 2010 (2010)
6. Laganière, R.: OpenCV Computer Vision. Packt-Publishing, Birmingham (2011)
7. Cagalaban, G., Kim, S.: Multiple Object Tracking in Unprepared Environments Using Combined Feature for Augmented Reality Applications. In: Kim, T.-h., Chang, A.C.-C., Li, M., Rong, C., Patrikakis, C.Z., Ślęzak, D. (eds.) FGCN 2010. Communications in Computer and Information Science, vol. 119, pp. 1–9. Springer, Heidelberg (2010)
8. Uchiyama, H., Saito, H., Servières, M., Moreau, G.: Camera tracking by online learning of keypoint arrangements using LLAH in augmented reality applications. In: Virtual Reality, vol. 15(2-3), pp. 109–117 (2011)
9. Gruber, L., Zollmann, S., Wagner, D., Schmalstieg, D., Höllerer, T.: Optimization of Target Objects for Natural Feature Tracking. In: Proc. IAPR/IEEE ICPR (20th International Conference on Pattern Recognition), 2010, Istanbul, Turkey, August 23–26, pp. 3607–3610 (2010)
10. Gauglitz, S., Höllerer, T., Turk, M.: Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking. International Journal of Computer Vision (IJCV) 94(3), 335–360 (2011)
11. Webpage of the OpenCV Wiki: last seen: (February 22, 2013), `http://opencv.willowgarage.com/wiki/`
12. Silpa-Anan, C., Hartley, R.: Localization using an imagemap. In: Australasian Conference on Robotics and Automation, Australia, December 6 - 8 (2004)
13. Freidman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. 3, 209–226 (1977)
14. Jacob, E.G., O'Rourke, J., Indyk, P.(ed.): Nearest neighbours in high-dimensional spaces. In: Handbook of Discrete and Computational Geometry, 2nd edn. CRC Press (2004)
15. Muja, M., Lowe, D.G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: International Conference on Computer Vision Theory and Applications, VISAPP 2009 (2009)
16. Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM 24(6), 381–395 (1981)