2014

Comparison of Natural Feature Descriptors for Rigid-Object Tracking for Real-Time Augmented Reality

France Franco Bermudez
*Inter American University of Puerto Rico*

Sheneeka Ward
*Georgia State University*

Christian Santana Diaz
*Polytechnic University of Puerto Rico*

Rafael Radkowski
*Iowa State University, rafael@iastate.edu*

Timothy Garrett
*Iowa State University, garrettt@iastate.edu*

*See next page for additional authors*

Follow this and additional works at: [http://lib.dr.iastate.edu/me_conf](http://lib.dr.iastate.edu/me_conf)

Part of the *Electrical and Computer Engineering Commons, Materials Science and Engineering Commons*, and the *Mechanical Engineering Commons*

**Recommended Citation**

Bermudez, France Franco; Ward, Sheneeka; Diaz, Christian Santana; Radkowski, Rafael; Garrett, Timothy; and Oliver, James H., "Comparison of Natural Feature Descriptors for Rigid-Object Tracking for Real-Time Augmented Reality" (2014). *Mechanical Engineering Conference Presentations, Papers, and Proceedings*. 188.

[http://lib.dr.iastate.edu/me_conf/188](http://lib.dr.iastate.edu/me_conf/188)

This Conference Proceeding is brought to you for free and open access by the Mechanical Engineering at Iowa State University Digital Repository. It has been accepted for inclusion in Mechanical Engineering Conference Presentations, Papers, and Proceedings by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Comparison of Natural Feature Descriptors for Rigid-Object Tracking for Real-Time Augmented Reality

Abstract
This paper presents a comparison of natural feature descriptors for rigid object tracking for augmented reality (AR) applications. AR relies on object tracking in order to identify a physical object and to superimpose a virtual object on an object. Natural feature tracking (NFT) is one approach for computer vision-based object tracking. NFT utilizes interest points of a physical object, represents them as descriptors, and matches the descriptors against reference descriptors in order to identify a physical object to track. In this research, we investigate four different natural feature descriptors (SIFT, SURF, FREAK, ORB) and their capability to track rigid objects. Rigid objects need robust descriptors since they need to describe the objects in a 3D space. AR applications are also real-time applications, thus, fast feature matching is mandatory. FREAK and ORB are binary descriptors, which promise a higher performance in comparison to SIFT and SURF. We deployed a test in which we match feature descriptors to artificial rigid objects. The results indicate that the SIFT descriptor is the most promising solution in our addressed domain, AR-based assembly training.

Disciplines
Electrical and Computer Engineering | Materials Science and Engineering | Mechanical Engineering

Comments

Authors
France Franco Bermudez, Sheneeka Ward, Christian Santana Diaz, Rafael Radkowski, Timothy Garrett, and James H. Oliver

This conference proceeding is available at Iowa State University Digital Repository: http://lib.dr.iastate.edu/me_conf/188
ABSTRACT

This paper presents a comparison of natural feature descriptors for rigid object tracking for augmented reality (AR) applications. AR relies on object tracking in order to identify a physical object and to superimpose virtual object on an object. Natural feature tracking (NFT) is one approach for computer vision-based object tracking. NFT utilizes interest points of a physical object, represents them as descriptors, and matches the descriptors against reference descriptors in order to identify a phsical object to track. In this research, we investigate four different natural feature descriptors (SIFT, SURF, FREAK, ORB) and their capability to track rigid objects. Rigid objects need robust descriptors since they need to describe the objects in a 3D space. AR applications are also real-time application, thus, fast feature matching is mandatory. FREAK and ORB are binary descriptors, which promise a higher performance in comparison to SIFT and SURF. We deployed a test in which we match feature descriptors to artificial rigid objects. The results indicate that the SIFT descriptor is the most promising solution in our addressed domain, AR-based assembly training.

1 Introduction

Augmented Reality (AR) technology is a type of human-computer interaction that enhances the natural visual perception of a human user with computer-generated information (i.e., 3D models, annotation, and text) [1]. AR presents this information in a context-sensitive way that is appropriate for a specific task, and typically, relative to the users physical location. Special viewing devices are necessary to use AR. A common viewing device is the so-called head mounted display (HMD), a device similar to eyeglasses that use small displays instead of lenses. The user sees the physical environment in a video image that is shown on the displays. This video image is superimposed with computer-generated information that are aligned with the physical environment.

AR is used as tool in several areas of mechanical engineering. Our research addresses the areas of AR-based assembly training and maintenance in mechanical engineering. For this purpose, an AR application must be able to track mechanical parts, to identify them, and to tell mechanics how to assemble these parts. A typical component is a rigid objects with a com-
plex spatial structure.

An AR application for AR-based assembly training relies on robust object tracking - the application must be able to follow the position of an particular object and to identify it. Since no artificial markers can be attached on a product on a factory floor, one possible approaches is natural feature tracking (NFT). NFT utilizes interest points on the surface of an object (i.e., color patches, edges, grey-scale gradients) to create a so-called feature map; a sparse computer-internal representation of feature descriptors [2]. During runtime the feature descriptors are compared with descriptors found in a live video stream. If they match, the pose of the camera can be estimated subsequently.

NFT facilitates the tracking of planar objects. Since the feature descriptors are represented in image space, they can be easily matched against reference features which are extracted from planar physical objects. The geometrical relations between both sets of descriptors can be represented as homography with 2D to 2D correspondences. The pose of a video camera can be estimated from the homography, which allows proper augmentation. This can be called the standard approach. Spatial rigid objects of mechanical engineering need to be described in a 3D domain instead of a 2D domain which is typically done with a 6-degree-of-freedom transformation matrix for each descriptor. Nevertheless, distortions of planar feature descriptors cannot be avoided. Thus, the more robust the feature descriptors are against distortions, the better the matching, and subsequently the tracking, works. In addition, a 2D to 3D correspondence needs to be computed to estimate the camera pose.

In this research, we compared four different natural feature descriptors and their feasibility to describe and track rigid objects in 3D space. We compared the descriptors SIFT [3], SURF [4], FREAK [5], and ORB [6]. Since the SIFT feature descriptor is a well-known solution, we use it as reference. For testing, we used a video of several artificial rigid objects and matched reference descriptors against descriptors found in the video. To assess the tracking quality we counted the positive matches between the reference image and the video image features. The paper is structured as follows. In the next section, we introduce the related work. The third section explains our implementation of the NFT application. Section 4 describes the test methods and presents the results. The last section closes the paper with a summary and an outlook.

2 Related Work

This section presents an overview about the related work in this area. We first introduce several NFT approaches for object tracking. Second, we introduce research that has already compared feature descriptors. We close this section with a summary.

2.1 Natural Feature Tracking for Augmented Reality

In general, NFT is a vision-based tracking technique. 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. [7] 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. [8] have 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. [2] have developed a keypoint tracking system that copes with different lighting conditions. The authors employ a FAST algorithm [6] 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 [9] 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.

Cagalab et al. [10] introduce a tracking method that allows tracking of multiple 3D objects in unprepared environments. The method incorporates 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. [11] 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., [12], [13]). 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 augmented reality and does not incorporate ORB and FREAK feature descriptors.

Kim et al. [14] present a 3D object recognition and tracking method which relies on Zernike moments. They obtain interest points from a live video using Harris corners and the characteristics of the location of the Harris corner are described with Zernike moments. The method matches the moments with reference moments in order to identify an object. For tracking, they use a Lie group method to calculate the homography between an initial 3D model and the object in an video image. The advantage of their method is an insensitivity against light changes. To verify the results, the authors tested the identification and tracking with 20 different models. The success rate is approx. 90%.

Okuma et al. [15] introduces a hybrid tracking system for 2D objects that incorporates two natural feature approaches and an inertial sensor. A Lucas-Kanade tracking method with Good Features To Track is their main tracking method. If this method fails to track an object, they analysis the feature density in the video image to detect motion. To show the feasibility, the authors conducted an experiment in which they track an object. However, the approach only tracks 2D objects.

2.2 Comparison of Natural Feature Tracking Methods

Several authors already compared different feature descriptors and assessed their performance. Gauglitz et al. [16] present a study that compares several planar feature descriptors concerning various geometric changes, lighting conditions, and levels of motion blur. They perform a quantitative evaluation of a huge amount of feature descriptors including SIFT and SURF. In total, they assessed 30 feature descriptor methods. The results are ambiguous and depend on the test parameter. In general, the results show advantages of the SIFT descriptor. Nevertheless, the authors also reported performance issues when using SIFT.

Schaeffer [17] presents a comparison of FREAK vs. SURF vs. BRISK feature detectors. The author conducted a quantitative study and recorded runtimes and accuracy. The results show that the FREAK detector results in the highest matching accuracy and the fastest runtime.

Redondi et al. [18] introduce an optimization and present a comparison of binary feature descriptors. The authors propose an entropy coding scheme that analyzes the ordering of the descriptor array in order to minimize the number of bits necessary to represent it. In addition, they analyzed the BRISK and FREAK feature detector and compared it against the SIFT detector. To our knowledge, this research is the closest to our work. We also incorporate FREAK in our comparison and we obtained similar results. Nevertheless, we also analyzed the ORB feature detector and integrated the tracking method into an AR application to allow a qualitative assessment.

2.3 Summary

The review of the related work shows that NFT is widely used for tracking of planar objects in AR applications. Most of the objects have been planar. Even 3D objects have been split up to several planar descriptions. A single feature map is usually planar. Nevertheless, the review show that several applications already utilized NFT for rigid object tracking. The introduced applications could track rigid objects and the research shows its feasibility. However, the authors did not compare different feature descriptors. The presented comparisons in literature did not address FREAK and ORB feature descriptors for AR. Nevertheless, previous research indicates the tracking quality. However, the research uses databases of feature descriptors and do not integrate the features into AR applications. Thus, the feasibility of this feature detection methods for AR has not been evaluated yet. In addition, FREAK and ORB are also binary descriptors and promises a better performance in comparison to SIFT; the high computational cost is a disadvantage of SIFT.

3 Natural Feature Tracking for Augmented Reality

The goal of the tracking system is to determine the pose of a video camera with respect to a particular physical object in order to superimpose computer-generated data. Our NFT tracking system relies, like many others, on aligning interest points from training images with interest points obtained from a run-time video stream. An interest point is represented by a keypoint location and a feature descriptor, which describes the surrounding of a particular keypoint. All training keypoints are stored in one database, and the query descriptor set is matched against this database. Since feature descriptors represent the object in a 2D domain, the challenge when tracking physical rigid objects is to receive a sufficient number of correctly matched feature points to enable pose estimation and tracking. In this section, we first present an overview of our implementation of NFT before we explain its single steps.

3.1 Overview

Figure 1 presents an overview of the natural feature tracking method. The method implements the functions keypoint and descriptor extraction, descriptor matching, as well as pose estimation.

**Keypoint and descriptor extraction:** the goal of the first step is to identify keypoints in the run-time video stream and to extract descriptors that represents the surrounding of the keypoints. The output are descriptors that have been found in the video image.

**Descriptor matching:** the keypoint descriptors are matched with reference descriptors of all objects to track. The reference
descriptors are generated during initialization from a set of photos of the objects to track and stored in a database. The outputs are matching descriptors and an object id.

**Pose estimation**: the matching descriptors are used to estimate the pose of the video camera with respect to the physical object. The output is the pose of the camera.

Our implementation is based on OpenCV [11], an open source computer vision library that provides the required core image processing functionality, the feature descriptors, the matching algorithm, and the pose estimation method.

For rendering, we use OpenSceneGraph (OSG), an open source scene graph programming toolkit. OSG provides functions for scene graph creation and management, to load and write 3D files for interaction, as well as for 3D scene and video rendering. Since the rendering functionality is not part of this research we will not further address it.

### 3.2 Feature Map Database

Consider a feature map $F_i = \{k_1, ..., k_N | d_1, ..., d_N\}$, with $N$ keypoints $k_i \in K_i$ and $N$ associated descriptors $d_i \in D_i$. Each set $F_i$ is associated with a physical object to track and enables tracking of this object $O_i$, with $i$, the index of the object, also referred to as a tracking target. A keypoint describes the location of an interest point on the surface of the tracking target. A keypoint descriptor the surrounding of this keypoint. Our implementation supports SIFT, SURF, FREAK, and ORB feature descriptors. For instance, the descriptor SIFT 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 $|DB_{ref}| = \{F_0, ..., F_N\}$ stores all feature keypoints $K_N$ and descriptors $D_N$ of all physical objects to track.

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. One kd-tree stores only the descriptor of one particular feature type (SIFT, SURF, FREAK, ORB). The descriptors are labeled with a label $i$, the index of a particular rigid objects. This label is necessary to identify an object. Several trees are required if more than one feature type should be used. Our implementation supports this, however, we usually do not use more that one feature type at the same time.

### 3.3 Keypoint and Descriptor Extraction

**3.3.1 Scale-invariant feature transform** The scale-invariant feature transform (SIFT) was published by Lowe [3]. Lowe’s method defines interest points in an image as maxima and minima of the result of difference of Gaussians function applied to a series of resampled images. Only interest points with a high contrast are used; interest point candidates with low contrast are discarded. The feature descriptor is a feature vector with 128 elements that represents a histogram of the surrounding pixels. SIFT features are invariant to image scale and rotation. The are also robust to illumination and viewpoint changes. The large feature descriptor vector is an issue which affects the performance. On modern computers, SIFT features can be extracted and matched in real-time or close to real-time.

**3.3.2 Speeded Up Robust Features** Speeded Up Robust Features (SURF) [4] was inspired by SIFT and works similarly, however, the feature detection and representation were enhanced. It uses an integer approximation to the determinant of Hessian blob detector to detect interest points. To describe features, is uses the sum of a Haar wavelet response around the interest point candidate. The descriptor is stored in a vector with 64 elements. The author’s experiments show that SURF works faster than SIFT.

**3.3.3 Oriented FAST and rotated BRIEF** ORB (Oriented FAST and rotated BRIEF) [6] is a combination of the so-called FAST feature detector [19] and the BRIEF feature detector [20]. FAST searches for corners as interest points. At corners, it considers the intensity threshold between the center of a pixel and those in a circular ring around this pixels. The FAST feature is fast and accurate. Originally, it is not scale invariant. The developers of ORB added this feature by employing an image pyramid. The BRIEF descriptor is a bit array that represents the intensity of an image patch. A bit for a certain pixel is calculated using a threshold test. ORB incorporates both, thus, it gains the advantages of FAST and BRIEF which results in improved runtime performance and more accuracy.
3.3.4 Fast Retina Keypoint

Fast retina keypoint (FREAK) was introduced by Alahi et al. [5] and was inspired by the human visual system. It describes an interest point as a binary string that represents the intensity of an image patch. The image patch is a so-called retina sampling pattern due to its similarity to a retina structure. It is a circular sampling pattern with more feature points in the center in comparison to rectangular sampling grids. Experiments from Alahi et al. show that FREAK features can be computed and matched faster than SIFT features.

3.4 Descriptor Matching

The goal of the descriptor matching is to identify the found descriptors and keypoints. To identify them, descriptors \( D^*_r \) are matched against the reference descriptors \( D^*_N \in DB_{ref} \). To match the feature \( d^*_i \), we employ a k-nearest-neighbor (KNN) search 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 \( D_{ref} \). Best matches are considered the database descriptors with the closest distance. The distance function is either a Euclidian distance for real-value feature descriptors (SIFT, SURF)

\[
dist_k(d_i, d_{ref}) = \sqrt{(d_{i,1} - d_{ref,1})^2 + ... + (d_{i,N} - d_{ref,N})^2}
\]

(1)

or a Hamming distance for binary feature descriptors (BRIEF, ORB).

\[
dist_k(d_i, d_{ref}) = \min(H(d_i), H(d_{ref}))
\]

(2)

with \( dist_k \) and \( k = 1,2 \), the two closest feature descriptors, and \( H \), the Hamming operator. The search function returns the two nearest neighbors for each query descriptor. These are the two best matches based on the distance of the feature descriptors. The entire matching set is denoted as \( M \).

The reference dataset \( D_{ref} \) and the single feature descriptors are organized in a randomized kd-tree [12]. This accelerates the matching process. We utilize the OpenCV implementation of the Fast Library for Approximate Nearest Neighbors for KNN matching [15].

Next, a ratio test is employed to reject all ambiguous matches and to keep only reliable matches. The knn-algorithm returns the two best matches. If the distance between two descriptors of the first match is close, and the distance between the descriptors of the second match is large, we can simply accept the first match as a good match or vice versa. The second match is rejected. If the two best matches are close, they are ambiguous and will likely result in an error by selecting one of them. In this case, both matches are rejected. The ratio test finds the best matches of the entire output set \( M \). It is a ratio test that checks whether the matches found violate a threshold \( r < r_t \) [6]:

\[
r = \frac{dist_1}{dist_2}
\]

(3)

With \( r \), the ratio, and \( r_t \), the threshold. Usually, the threshold ranges from 0.6 to 0.8. Only high quality matches pass this test. All other matches are deleted.

The last step employs an epipolar test to check whether the feature points meet the fundamental constraint of a 3D projection: all matching keypoints of the query set must lie within a certain distance to the epipolar line of the reference set’s points. All matches that do not obey this fundamental constraint are rejected. To calculate this, the fundamental matrix is computed:

\[
x' = Fx
\]

(4)

with \( F \), the fundamental matrix, a 3x3 matrix that encode the epipolar geometry of two views. We use the 8-point algorithm to determine the fundamental matrix \( F \). The 8-point algorithm assumes that we have matches that fulfill the constraint:

\[
(x')^TFx = 0
\]

(5)

With nine unknowns for \( F \), this results in the following system of linear equations:

\[
\begin{bmatrix}
x_1x'_1 & y_1x'_1 & x'_1 & y_1 & y'_1 & x'_1 & x_1 & y_1 & 1 \\
x_2x'_2 & y_2x'_2 & x'_2 & y_2 & y'_2 & x'_2 & x_2 & y_2 & 1 \\
... \\
x_nx'_n & y_nx'_n & x'_n & y_n & y'_n & x'_n & x_n & y_n & 1 \\
\end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0
\]

(6)

To solve this equation, 8-points are required. We use a RANSAC algorithm with Singular Value Decomposition to calculate the solution. The RANSAC algorithm is an iterative technique to estimate the parameters of a equations [16]. Firstly, it uses eight points to obtain an initial solution. All the remaining
point of the matching set are subsequently tested against the solution in order to check whether they support the solution. These supporting points are inliers, all others are considered as outliers. Finally, all inliers of the RANSAC test are used to estimate the pose.

Figure 2 presents a result of the matching step. The left image shows the training image and its keypoints. The right image depicts the video stream and the keypoints found in the video image. The lines indicate the corresponding keypoints.

![Figure 2](image)

FIGURE 2. Tracking of a circuit board with SIFT feature tracking

Note, this matching strategy differs from the recommended or originally used strategy, which have been introduced by the developers of the particular descriptors. For instance, the developers of FREAK introduced a so-called Saccadic search, which parses the descriptors in several steps [5]. We did not follow the recommendations since we are interested in obtaining data to assess the tracking quality. This particular recommendation will only increase its performance.

### 3.5 Pose Estimation

Goal of the pose estimation step is to calculate the pose of the video camera in object coordinate system with respect to the object to track. The correspondence between a point \( P \) on the surface of the object to track and the image keypoints \( k \) is described as:

\[
  k(u, v, w) = K(RP + t)
\]

with \( K \), a camera calibration matrix, \( R \), a 3x3 rotation matrix that represents the orientation of the camera, and \( t \), the translation of the camera. Extending \( R \) with \( t \) results in \( [R|t] \), the camera transformation matrix \( CM \) which needs to be determined. Therefore, we use a PnP-algorithm described in [21] and [22]. To solve the equation, a correspondence between image points and object points is necessary, which were obtained during the matching step. The matches are used to describe the geometric relation between image and physical object: the distance between all points in the image and all points on the object must be calculated. Following the PnP-algorithm, this can be done using the cosines law.

\[
  \cos \alpha_{i,j} = \frac{P_i P_j}{||P_i|| ||P_j||} = \frac{K^{-1} p_i, K^{-1} p_j}{||K^{-1} p_i|| ||K^{-1} p_j||}
\]

To solve the cos-law, it has to be represented as a system of linear equations. This system can be solved using Singular Value Decomposition, which is a standard textbook technique from this point. The output of this algorithm is the camera matrix \( CM \), which represents the pose of the video camera with respect to the physical object.

The camera matrix is passed to the renderer. Figure 3 shows a set of rendering examples, screenshots from our AR application. In this case, the object to track is a circuit board. The AR application superimposes two virtual probes and a virtual connection between them.

![Figure 3](image)

FIGURE 3. Sample screenshots from an AR application that use NFT to track a circuit board

### 4 Feature Comparison

Thus section describes the comparison of the four different natural feature descriptors techniques, in particular, of the capability to track rigid objects in AR applications. We compared SIFT, SURF, ORB, and FREAK where SIFT and SURF can be considered as a baseline of our experiment. Both are well known, have been analyzed several times, and have already
proven their feasibility for AR applications. The following subsections present the test method, the results, and a discussion of the results.

4.1 Test method

To evaluate the different feature detection methods, we set up an experiment in which we applied the different features to track to rigid objects. We counted the number of good matches and assessed the tracking quality and performance.

Figure 4 shows the objects that were used for this test. We assembled a set of Lego brick objects for this purpose. All objects vary in size, spatial structure, and depth; depth means the distance between the highest and lowest point when considering a imaginary plane as depth zero (an even plane would be the optimal object to track). The dimensions of the objects are (dimensions of a rectangular bounding volume, height, wide, depth):

- object 1: 175 x 140 x 45mm
- object 2: 130 x 130 x 40mm
- object 3: 75 x 90 x 35mm
- object 4: 170 x 100 x 30mm

To have comparable test conditions, videos of all objects have been taken. Photos of the objects have been used to train a reference database DBref.

Figure 4. Artificial rigid objects, which has been used for the evaluation

To assess the tracking capability, we recorded the true-positive and the false-positive matches. The true-positive matches indicate the number or correct matching, the false-positive matches indicate the matches that are declared as true, however, they are false. We used a two-step approach to count the number of true-positives and false-positives matches in the video image. Firstly, we determined all positive matches within the region of interest. The region of interest was a rectangular area that encases the tracking target in image coordinates. All matches within the region of interest were considered as candidates for true positive matches, all remaining matches have be rejected as false-positive matches. The bounding box was placed at the last known position of the object.

Secondly, we refined the initial assessment using a projection of the remaining true-positive matches. True-positive matches, in particular the keypoints, can be back-projected from object coordinates to image coordinates using the camera projection matrix.

\[ k = Kp \]  

with \( p \), an interest point on the rigid object that is associated to a certain feature point \( k \) in image coordinates, and \( K \), the camera projection matrix. If the distance between a keypoint representation in 2D coordinates and its associated interest point in 3D coordinates violates a threshold, they are out of alignment and cannot match, even so the matching step identified them as similar. These points have also been rejected and counted as false positives. In addition, the computation time of all feature descriptors has been recorded.

4.2 Results

Figure 5 to Figure 8 show the results. The results are presented as Receiver Operating Characteristics (ROC) graphs [23]. The abscissa shows the false-positive matches and the ordinate the true-positive matches. The graph indicates the ratio between good matches and false matches for the four test objects. Note, the individual points along the graph indicate the recorded data. Every point refers to a different parameter set. The values are shown in Figure 9; the indexes in Figure 5 are indicate the test numbers (test 1 is the left, the indexes go along with the line). The red area highlights the region in which decent pose estimation is possible.

The results show that all feature tracking methods are feasible to track rigid objects. For augmented reality applications, a true-positive rate of 0.6 with a false-positive rate of 0.2 is the working area in which we obtain a decent object alignment. Note, this area relies on our observation of the tracking experiments. It may not be fixed and can vary in a certain range when tracking different rigid objects.

For each experiment, the processing time for each step in feature extraction, matching, and pose estimation has been determined: for the SIFT feature, 30.1 ms, for SURF 16.4ms, for FREAK 8.8 ms, and for ORB 15.3 ms. Note that this times did not consider the rendering time, which is required by the AR application to generate the output. Each time is the average time of all experiments for a particular feature detection method.
4.3 Discussion

The results show that all feature detection methods are feasible for AR applications. However, the results show that the SIFT feature detection and description method provides the highest number of true-positive matches which is essential for a high-fidelity tracking, regardless of the shape of the object. Considering the data, we can say that the SIFT feature detector is capable to track rigid objects in a 3D space. This result is no surprise, it complies with the results of both Redondi [18] and Gauglitz et al. [13], who also reported a high matching accuracy for SIFT. Since SIFT is also known for being less affected by image distortions and viewpoint rotations, it is also plausible that it tracks 3D objects with keypoints on different locations on the surface of a 3D object; keypoints appear as distorted patches if they are not aligned towards the video camera. However, the current implementation of SIFT runs slower than the implementations of SURF, FREAK, and ORB due to the larger vector size. For the SURF feature descriptor, we obtained similar results. However, the SURF feature detection and matching operates faster than SIFT. The ratio between true-positive and false-positive matches is lower. Nevertheless, we observed a similar tracking quality.

The results of ORB and FREAK are ambiguous. In general, the results show that both feature descriptors facilitate matching of rigid objects. The data for the FREAK detector shows the lowest ratio of true/false-positive matches. However, we can report that FREAK could match feature descriptors on object 2 and object 3 (Figure 4) and that the matching accuracy facilitates calculation of a camera pose. In contradiction to this, the matching quality for object 1 and object 4 was often too low: too many false-positives have been identified which result in an inaccu-
rate pose or no solution for the pose. The advantage is that the FREAK feature detector works faster than the other descriptors.

The data for the ORB feature detector shows an average ratio of true-positive and false-positive matches. In general, we observed a decent tracking quality when using the ORB feature. However, we observed problems when tracking object 3. Only a small number of features represented this object in the database. The NFT tracking algorithm was also not able to identify many features in the runtime video. In some frames, only 3 to 9 feature descriptors have been matched, which does not facilitate object tracking. In our opinion, the object size may have caused this problem. Object 3 is the smallest object. Since the video camera has always been at the same location, the distance between camera and object remained similar in all experiments. Thus, object 3 covers the smallest area of image pixels. We think the low resolution of the object in the video image caused this matching problem.

One limiting fact is the low number of test objects. We presented results for four random objects which does not allow general assessment of the feature descriptor quality and the matching outcome. Nevertheless, the results will direct future work and may direct if one has to decide for a particular feature detector and descriptor to implement.

In addition, the randomly selected objects are also a limiting factor. These are not real-world objects. They have been assembled to cover a wide range of different dimensions and shapes. Nevertheless, it is possible to assemble objects which cannot be tracked. For instance, we assembled several additional Lego brick objects and also tried to create a feature database to track them. For instance, one object was a 3D car made out of Lego bricks. The number of descriptors that we obtained was too little to consider it as test object for this experiment; zero of the four feature descriptors could track it. Since our tracking application works well for most other objects as well as for 2D images, we think the shape, color, and the appearance of the object in general caused this negative result. However, we did not address this and did not characterize the shape style of rigid objects.

The used region-of-interest and the back-projection may also do not find all valid true-positive matches. We did not find any wrongly assigned true-positives when we checked several images manually. Nevertheless, we cannot exclude it.

Finally, the tracking strategy presented in Section 3 does not comply 100% with the tracking methods the developers of these methods used and recommend. We did not change the representation or the distance values, however, there are changes that might affected the results.

5 Conclusion and Future Work

In this research, we assessed natural feature descriptors and their matching outcome when rigid objects need to be tracked. Rigid objects are those with a non-planar shape. When describing feature descriptors on their surface, using photos from multiple views, similar features can appear distorted when a video camera views them from different angles.

We investigated which feature descriptor can cope best with these challenges and track rigid objects. We compared four different feature descriptors, SIFT, SURF, FREAK, and ORB. From previous research, we already know that the SIFT feature facilitates the tracking of rigid objects since it is robust against image distortions and different viewing angles. Nevertheless, SIFT processes slowly due to its large feature vector. The FREAK and ORB feature detectors are binary feature detectors. This promises faster detection and matching. We compared them using videos of randomly assembled Lego brick objects and counted the number of true-positive and false-positive matches. The data supports that SIFT and SURF features have the best ratio of true-positive and false-positive matches. As expected, SIFT is slow. The FREAK feature detector is the fastest detector. Nevertheless, we obtained ambiguous results and also encountered several tracking problems. The ORB feature detector is a good balance between matching quality and run-time performance. The matching fidelity allows us to calculate a camera pose and align virtual objects with the rigid objects. From our point of view, the ORB feature detector is a promising option for AR applications.

In future work, we will investigate the limits of the ORB feature detector. The presented research was a first attempt to obtain knowledge about the tracking capabilities of the ORB feature descriptor. The next research will analyze details like the number of objects we can track and distinguish in one AR application, the maximum amount of features in a database, as well as different environment conditions such as changing light conditions and glares on object surfaces. We will also replace the Lego brick objects and incorporate real world objects.

6 Acknowledgement

This material is supported in part by the National Science Foundation Grant CNS-1156841.

REFERENCES


FIGURE 9. The parameter values for the experiments.