Fakultät für Mathematik, Informatik und Naturwissenschaften
Lehrstuhl für Informatik VIII (Computergraphik, Computer Vision und Multimedia)
Mobile Multimedia Processing
Prof. Dr. Bastian Leibe

Master Thesis

Multiple Target Tracking for Marker-less Augmented Reality

Francis Engelmann
Student Id.: 286130

January 2014

Erstgutachter: **Prof. Dr. Bastian Leibe**
Zweitgutachter: **Prof. Dr. Leif Kobbelt**

communicated by Prof. Dr. Bastian Leibe

Acknowledgment

I would like to thank everyone involved in this thesis for their support, fruitful discussions and helpful ideas. Primarily, I want to thank my supervisor Patrick Sudowe and Prof. Dr. Bastian Leibe for their valuable advice throughout the thesis, also all the members of the Computer Vision Group for a pleasant working atmosphere and last but not least, I would also like to thank my family and friends for their support in the past years.

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.

Hiermit versichere ich, diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht zu haben.

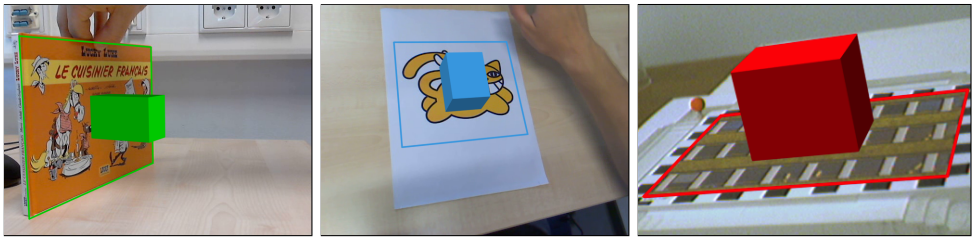
Aachen, 16^{th} of January, 2014

(Francis Engelmann)

Contents

Chapter 1

Introduction



Augmented reality (AR) is an increasingly recognized technique for information visualization. The general goal of AR applications is to simplify interactions with the real world and improve its understanding. This is achieved by displaying computer-generated content into the user's environment. Common tools for visualization are hand-held devices, head-mounted-displays (HMD), or even projectors [BR05]. The main challenges in AR are to find the correct pose of the real objects to augment, to render virtual content with geometrical consistency into the user's field of view, and to perform these steps in real time. Typically, AR applications demand accurate identification and localization of objects in a single frame, as well as 3D pose tracking over sequential images. Computer vision offers solutions to these requirements that are inexpensive, non-intrusive, and practical [LF05].

Typically, vision-based AR systems rely on a *learning phase*. During this phase, the system learns the physical objects (*targets*) that it should augment. Specifically, the system is provided with a normalized model for each target. The models are stored, using some canonical representation, in a database for availability during runtime. When the learned targets become visible to the system's camera, the system *detects* and *tracks* them in order to compute their exact location and finally render corresponding virtual objects over them. In the context of AR, detection refers to the process that *localizes* a learned target-object in the camera image and computes its pose with respect to the camera. If multiple targets are

stored in the database, detection also involves the *identification* of the current visible target. Furthermore, detection is the first step of *tracking*. In contrast to detection, which is applied to single images, tracking refers to camera pose estimation in temporal image sequences. Tracking is essential in AR applications as it dramatically increases the accuracy and robustness of the system [UM12].

The literature [YJS06, UM12, LF05] differentiates between two categories of tracking: (1) *Tracking-by-detection* tries to locate a target by matching each camera-image against all the model-images in the database. Put differently, the object detector is applied independently to each frame of the input video sequence. The single detections are then associated to form a track. (2) *Tracking-by-tracking* or also *frame-by-frame tracking* tries to find correspondences between the current camera-image and the previous camera-image in order to compute the pose difference between the two frames. In this work, we use a hybrid approach that combines these two methods.

The learned targets are subdivided into *planar* and *non-planar* objects [LF05]. While planar targets require only one single image to be learned, non-planar targets demand for a more elaborated model-representation. Our AR system will work on planar targets. Traditionally, one further differentiates between *marker-based* and *marker-less* AR systems. The first AR applications were marker-based. They relied on fiducial markers placed into the environment. Such markers are chosen to be easily detectable and identifiable with a camera. Even though they are inherently easier to track than natural markers, they are also more intrusive to the user's environment [UM12]. For marker-less approach, the approach is as follows: (1) Extract easily recognizable interest points, such as corners, in the model-image and the camera-image; (2) describe the extracted interest points using its surrounding image area; (3) find corresponding pairs of descriptors between the camera-image and the model-image. Searching for correspondences is either performed using nearest-neighbor-search and a distance function (such as the Hamming distance [WSB09a]), or by using randomized ferns or trees. In the latter case, matching is seen as a classification problem [WRM$^+$08]. Our implementation relies on natural feature-point matching using nearest-neighbor-search. For a more thorough introduction on current augmented reality techniques, we refer the reader to [NPS10], [UM12] and [Yua06]. They all give comprehensive surveys of recent approaches.

The goal of this master thesis is to implement and evaluate a marker-less augmented reality framework based on natural feature-point matching. The system should be able to differentiate between multiple planar targets and compute their pose relative to the camera. The pose estimation should be temporally smooth. Finally, the system should be able to cope with common problem cases that occur in practice, like objects entering or leaving the camera's field of view.

In this work, we implemented an AR framework for planar targets based on the ORB [RRKB11] feature-point descriptor. The implementation is written in C++ and uses the ORB implementation provided with the OpenCV library [Bra00]. The main components of the framework are a detector, a tracker and a graphical overlay. The detector returns a homography that maps the model-image onto the target in the camera-image. The homography is estimated from a set of feature-point correspondences using the *Direct Linear Transform* (DLT) algorithm and *Levenberg-Marquardt* (LM) optimization [HZ00]. The outliers in the set of feature-point correspondences are removed with an extended version of *Random Sample Consensus* (RANSAC) [FB81]. The tracker is based on the Kalman filter [FP02], which applies a consistent dynamic movement on the target. In a hierarchical matching scheme, we extract additional matches from consecutive frames and perspectively transformed model-images, which yields more accurate and jitter-free homography estimations. The graphical overlay computes the six-degree-of-freedom (6DoF) pose from the estimated homography. Finally, to visualize the computed pose, it draws a cube on the surface of the tracked target. In the evaluation part, we analyze the performance of our system by looking at the accuracy of the estimated homography and the ratio of correctly tracked frames. The evaluation is based on the groundtruth information provided by two datasets [LBMN09] and [GHT11]. We evaluate most components of the framework under different target movements and lighting conditions. In particular, we proof that our framework is robust against considerable perspective distortion and show the benefit of using the hierarchical matching scheme to minimize jitter and improve accuracy.

1.1 Related Work

The choice for a feature-point descriptor plays an important role in a AR framework: Some implementations such as [WRM+08] use a heavily-modified version of SIFT [Low99] for real-time capabilities on mobile phones. Some introduce new feature-point descriptors like [TCT+10] and others improve on existing approaches [RRKB11, TCT+12]. Our implementation is based on the ORB feature-descriptor [RRKB11]. The general trend however seems to optimize for speed rather than for reliability. Besides the choice of an appropriate feature-point descriptor, the target itself is also of great importance for a satisfying tracking result. Some effort was put into evaluating natural feature-point based tracking targets [EB08, GZWS09, Low04]. After establishing feature-point correspondences, an important step is the geometrical verification to remove outliers. A well recognized approach is RANSAC [FB81], which has seen many alternations and improvements over the years [CM05, SLK09, MBSS10]. We make use of the traditional RANSAC with sample selection constraints based on [MBSS10] and

additionally introduce a sanity check after each iteration that rejects physically impossible transformations.

It is not our goal to limit tracking to feature-points like [TK91] but rather combine tracking of feature-points and, on a higher level, homographies using their corner points. The higher level tracking has the benefit of avoiding the association problem since corners are directly identifiable. Similar to [WSB09b], on the feature-point level, we propagate inliers between consecutive frames. We combine tracking and detection in a similar fashion to [TCT+10].

The Kalman filter [FP02] is a popular and very general tool for tracking. In order to apply Kalman filtering to homography estimation, one needs to find a parameterization for the homography. [KSF07, CMFO07] perform this by reducing the degrees of freedom of the homography to their specific application scenery (road ground-plane estimation). Since a homography is fully determined (up to scale) by four points, we use the position of the corner points of the mapped target as parameters space for the Kalman filter. A similar approach is used by [SÖL+10], they consider the action of the homography on a unit square.

In this work, we limit ourselves to 2D planar targets. Thus, the transformation of the target into the scene image can be expressed using a homography. Other work use planar models to track 3D targets like [PLW08]. After finding the homography, the model-image is reinterpreted as the projection of a 3D model and each 2D feature-point is assigned a 3D coordinate.

After homography estimation, we still need to convert the perspective mapping into a 3D-2D projection. The relation between homography and projection matrix has been covered in the literature several times under different viewpoints [Zha00, LF05, HZ00, FL88, MV+07]. The main idea is, given a homography mapping a target object into the scene frame, compute the corresponding 6DoF pose. The approaches of [Zha00, Stu00] and [LF05] both describe homography decomposition and are similar to our work, but they omit to explain the important normalization step with respect to the determinant of the homography, which guarantees orientation preservation.

1.2 Contributions

In this work, we implemented a planar target tracker for marker-less augmented reality. In particular, our work includes the following aspects: (1) Detailed analysis of ORB [RRKB11] feature-points in the context of AR applications; (2) fast RANSAC extension that is guaranteed to return only homographies that correspond to rigid-body transformations; (3) hierarchic matching technique that

improves tracking and makes the system robust to large angles between the normal of the target-surface and the optical axes of the camera; (4) measure for the accuracy of the estimation homography based on the covariance matrix of the detected target-corner-points. We also show how to utilize the accuracy measure to improve tracking with a Kalman filter and to improve homography estimation accuracy; (5) we show empirically that short base-line matching between two frames is essential for jitter free augmentation and robust tracking; (6) direct solution to 6DoF pose estimation problem from a given homography.

1.3 Nomenclature

Throughout this thesis, we will make use of several specific names and terms, which are explained now. A *target* is the physical object that we want to detect and track in a video sequence. The *model-image* shows an orthogonal projection of target without rotation or translation with respect to the camera. It is given to the system in the learning phase. The *camera-image* (or *scene-image*) shows an arbitrary projective mapping of target as seen by the camera. In our case, it is a frame from an input video sequence and becomes available at run-time. A *match* or a *correspondence* connects two 2D image-points that correspond to the same 3D physical point. We use both terms as synonyms. *Degenerated* (or *invalid*) homographies correspond to physically impossible perspective transformations. The opposite are non-degenerated homographies, which represent *rigid-body*-transformations (see Figure 3.4).

1.4 Outline

This thesis is structured as follows. Chapter 2 starts by giving background information helpful to understand the following chapters. It includes the introduction of homography transformations as well as the DLT algorithm, which is employed to compute homographies. Moreover, it explains how the RANSAC algorithm is applied to robustly compute a homography from noisy data. Finally, the Kalman filter is shortly described in the context of our tracking approach. Chapter 3 is dedicated to target detection and robust homography estimation. It provides the fundamentals for our tracking framework, which is presented in chapter 4. The last step of the augmented reality pipeline is explained in chapter 5, it describes how to compute the 6DoF pose from a given homography so that we can overlay the scene image with additional artificial information in a geometrically consistent way. Chapter 6 gives an extensive evaluation of all the aforementioned components and justifies certain design decision. Finally, we conclude in chapter 7. Appendix A shows further result images and plots.

Chapter 2

Background

This chapter gives a brief overview of the fundamentals employed within this work. First of all, we explain *homographies* and how to estimate them using the *DLT algorithm*. We then move the focus to *RANSAC*, which is a well-known approach for dealing with outliers. Finally, we present the *Kalman filter*, which is a very prominent framework for probabilistic tracking.

2.1 Homography Estimation using DLT

This section explains the *Direct Linear Transformation* (DLT) algorithm for homography estimation. A homography is a projection from a plane onto another plane. In particular, given a set of 2D to 2D point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, the homography `H` represents a projective transformation that maps the points $\mathbf{x}_i$

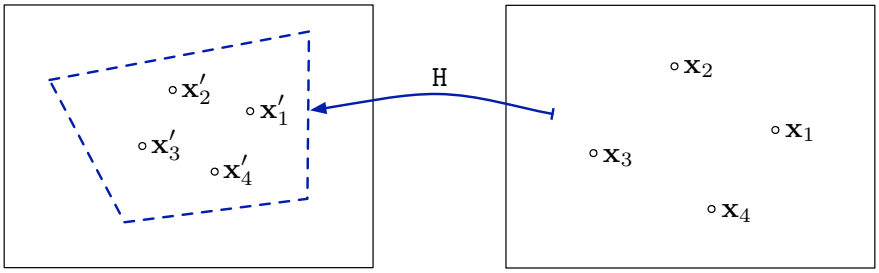


Figure 2.1: Schematic representation of a homography mapping. The homography `H` maps the points $\mathbf{x}_i$ to $\mathbf{x}'_i$. We will refer to the right image as the model-image showing an orthographic view the target and to the left image as the scene-image. The dashed blue line represents the mapping of the target after applying the perspective transformation.

onto $\mathbf{x}'_i$ as visualized in Figure 2.2. The transformation is given by the equation:

$$\begin{aligned} \mathbf{x}'_i &= \mathtt{H} \cdot \mathbf{x}_i \\ \begin{bmatrix} x'_i \\ y'_i \\ w'_i \end{bmatrix} &= \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} \end{aligned} \tag{2.1}$$

Note that this representation involves homogeneous coordinates. For notational simplicity and without loss of generality, we will assume in the following $w'_i = w_i = 1$. Equation (2.1) is reformulated into $\mathtt{A} \cdot \mathbf{h} = \mathbf{0}$. We refer the reader to [HZ00] for the complete derivation. To construct $\mathtt{A} \in \mathbb{R}^{2n \times 9}$ and $\mathbf{h} \in \mathbb{R}^9$ we proceed as follows: The vector $\mathbf{h}$ contains the entries of the homography $\mathtt{H}$. Then, each of the n point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ generates two rows in the matrix $\mathtt{A}$ as shown below:

$$\begin{bmatrix} x'_1 & y'_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -x_1y'_1 & -x_1 \\ 0 & 0 & 0 & x'_1 & y'_1 & 1 & -y_1x'_1 & -y_1y'_1 & -y_1 \\ & & & & & \cdots & & & \\ & & & & & \cdots & & & \\ & & & & & \cdots & & & \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \tag{2.2}$$

$\mathtt{A} \cdot \mathbf{h} = \mathbf{0}$

Four point correspondences are sufficient to fully constrain $\mathtt{H}$. Nevertheless, the more correspondences are available, the less sensitive the homography estimation becomes to measurement noise [GL11]. The solution of equation (2.2) is the null-space vector of $\mathtt{A}$ that can be obtained by computing the *singular value decomposition* (SVD) of $\mathtt{A}$:

$$\mathtt{A} = \mathtt{U} \cdot \mathtt{D} \cdot \mathtt{V}^\top = \mathtt{U} \cdot \begin{bmatrix} d_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_{99} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{91} \\ \vdots & \ddots & \vdots \\ v_{19} & \cdots & v_{99} \end{bmatrix}^\top \tag{2.3}$$

If the singular values (positive diagonal entries) of $\mathtt{D}$ are arranged in descending order, then $\mathbf{h}$ corresponds to the last column of $\mathtt{V}$. Finally, $\mathtt{H}$ is obtained by reordering $\mathbf{h}$ as described in equation (2.2).

Algorithm 2.1: RANSAC for robust homography estimation using DLT

Input : putative feature-point correspondences.
Output: homography H, partitioning of correspondences into inliers and outliers.

while $i < N$ **do**
  Sample random set of four correspondences.
  Compute homography $\mathtt{H}_i$ from random sample using DLT.
  Compute distance $d_\perp$ for each putative correspondence.
  Compute number of inliers for $\mathtt{H}_i$ by number of correspondences for which $d_\perp < \tau_{d_\perp}$.
end
Choose the homography $\mathtt{H}_i$ with the most inliers.
Reestimate the homography H using *all* inlier correspondences.

2.2 RANSAC

The previously presented DLT algorithm takes a set of feature-point correspondences as input to compute a homography. The feature-point correspondences can for instance be established using nearest-neighbor-search. In practice, the resulting matches will not always be correct because of noise and should therefore be seen as *putative* matches. However, the DLT algorithm assumes that these feature-point correspondences are correct, although this is rarely the case. Also, the DLT algorithm is sensitive to wrong matches [HZ00] so it is a good idea to first remove wrong matches. The RANSAC (Random Sample Consensus) algorithm is a popular tool for estimating geometric models, e.g. homographies, from datasets affected by noise. It partitions the putative matches into inliers (correct matches) and outliers (wrong or ambiguous matches). Additionally, it gives an estimate of the underlying geometric model. The algorithm works by randomly choosing a sample of 4 correspondences and computing the corresponding homography $\mathtt{H}_i$. It then uses some distance measure $d_\perp$ to decide for each putative correspondence whether it is consistent with the homography or not. If this distance is smaller than a given threshold $\tau_{d_\perp}$ the match is considered to be an inlier, if it is larger, the match is an outlier. These steps are repeated N times and finally the algorithm returns the homography with the highest inlier count. The resulting procedure is summarized in Algorithm 2.1. Section 3.3.2 presents the details of our implementation.

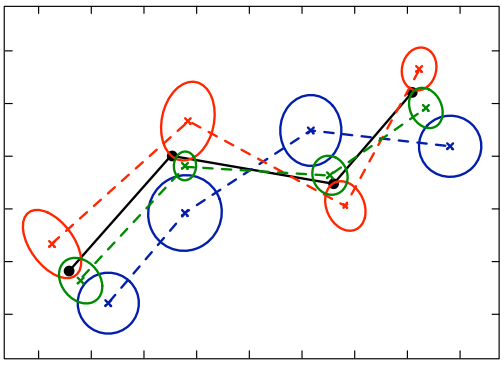


Figure 2.2: Illustration of a Kalman filter used to track a moving object. The black dots indicate the groundtruth movement of the object in a two-dimensional space at discrete time steps. The red crosses indicate the corresponding noisy measurements of the position, the blue crosses show to the predicted position and the green crosses show the corrected positions. The respective covariances are indicated by the ellipses that correspond to contours having one standard deviation. Note how the corrected position (green) is interpolating between the predicted (blue) and measured (red) positions.

2.3 Kalman Filter

This section introduces the Kalman filter, which is a probabilistic framework for tracking targets under the assumption of Gaussian noise. Kalman filters have been extensively covered by the available literature [FP02, BN06, JSM10, Dic10, Hor86]. Here we only present it in the form as it was used within this work. The general idea behind the Kalman filter is straightforward: A tracked target is represented by an internal state $\mathbf{x}_t$ and a covariance matrix $\Sigma_{\mathbf{x}_t}$. For example, the state vector can be composed of a position and a velocity.
The state $\mathbf{x}_t$ is assumed to evolve over time t according to a *dynamic model* $\mathtt{D}$ mapping $\mathbf{x}_{t-1}$ onto $\mathbf{x}_t$. On the one hand, the dynamic model $\mathtt{D}$ enforces a consistent movement on the target, such as a constant velocity model, while on the other hand it limits the influence of measurement noise $\Sigma_{\mathtt{M}_t}$. The measurements $\mathbf{y}_t$ are related to the hidden state $\mathbf{x}_t$ by a *measurement model* $\mathtt{M}$. The measurement model describes what parameters of the state $\mathbf{x}_t$ are observable. For example, given a single frame, a position is imminently measurable whereas a velocity is not. The Kalman filter assumes both the dynamic- and measurement-model to be affected by normally distributed noise with zero mean:

$$\mathbf{x}_t \sim \mathcal{N}(\mathtt{D} \cdot \mathbf{x}_{t-1}, \Sigma_{\mathtt{D}_t}) \tag{2.4}$$
$$\mathbf{y}_t \sim \mathcal{N}(\mathtt{M} \cdot \mathbf{x}_t, \Sigma_{\mathtt{M}_t}) \tag{2.5}$$

where $\Sigma_{\mathtt{D}_t}$ and $\Sigma_{\mathtt{M}_t}$ correspond respectively to the uncertainty of the dynamic model and the measurement model.

At each time step t, the Kalman filter first estimates a *prediction* $\mathbf{x}_t^-$ of the current state. In a second step, the prediction is refined by incorporating the measurement $\mathbf{y}_t$ of the current frame to yield a *corrected* state $\mathbf{x}_t^+$. Specifically, the predicted state $\mathbf{x}_t^-$ and its covariance matrix $\Sigma_{\mathbf{x}_t}^-$ are computed using the corrected state information from the previous time step and the underlying dynamic model $\mathtt{D}$ by:

$$\mathbf{x}_t^- = \mathtt{D} \cdot \mathbf{x}_{t-1}^+ \tag{2.6}$$

$$\Sigma_{\mathbf{x}_t}^- = \mathtt{D} \cdot \Sigma_{\mathbf{x}_{t-1}}^+ \cdot \mathtt{D}^\top + \Sigma_{\mathtt{D}_t} \tag{2.7}$$

where $\Sigma_{\mathbf{x}_{t-1}}^+$ is the corrected covariance matrix for the previous time step and $\Sigma_{\mathtt{D}_t}$ models the uncertainty that measures how well the motion model is respected in reality. The correction step is computed as follows:

$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathtt{K}_t \cdot \underbrace{(\mathbf{y}_t - \mathtt{M} \cdot \mathbf{x}_t^-)}_{\text{Residual}} \tag{2.8}$$

$$\Sigma_{\mathbf{x}_t}^+ = (\mathtt{I} - \mathtt{K}_t \cdot \mathtt{M}) \cdot \Sigma_{\mathbf{x}_t}^- \tag{2.9}$$

where the *Kalman gain* $\mathtt{K}_t$ is calculated as:

$$\mathtt{K}_t = \frac{\Sigma_{\mathbf{x}_t}^- \cdot \mathtt{M}^\top}{\mathtt{M} \cdot \Sigma_{\mathbf{x}_t}^- \cdot \mathtt{M}^\top + \Sigma_{\mathtt{M}_t}} \tag{2.10}$$

$$\tag{2.11}$$

where $\Sigma_{\mathtt{M}_t}$ corresponds to the measurement uncertainty at time t. Although the Kalman filter equations are quite straightforward, the difficulty lies in finding good values for the covariance matrices $\Sigma_{\mathtt{M}_t}$ and $\Sigma_{\mathtt{D}_t}$ [FP02]. This problem is very application specific and requires deeper understanding of the underlying bases. We will therefore postpone it to Section 4.5.3.

Chapter 3

Detection

This chapter covers how target detection and robust homography estimation are performed by our system. In particular, we want to decide whether a learned target is present in a given scene image and, if yes, where it is precisely located. We start by introducing state-of-the-art object-detection based on feature-point correspondences. Then, we concentrate on implementation specific details of the different components. Finally, we show how to compute the precise location of the identified target in the scene image using robust homography estimation.

3.1 Overview

This section gives an overview of the basic processing pipeline underlying state-of-the-art object identification and localization. We start by explaining the procedure that can detect a given target in a given scene image. This task is separated in three basic steps: (1) First, local feature-points are extracted from both the model image and the scene image independently. The corresponding descriptors are computed. (2) The feature-points are then matched, based on their descriptors, to find putative correspondences between the model image and the scene image. (3) Finally, it is verified that these tentative feature-point matches are non-ambiguous and occur in a consistent geometric configuration.

This procedure can be generalized to be able to differentiate between multiple different targets as we did in this work. In an offline-stage, we first construct the descriptor database. Since our system is currently limited to planar targets, it is sufficient to consider one single orthographic image per target. For each such model image, we extract the feature-points and compute their descriptors, which are then stored in the database with a reference to the original model image. After establishing the database, we proceed to the actual detection part. The descriptors of the feature-points from the scene image are matched against the descriptor database. This results in a set of putative correspondences, which are

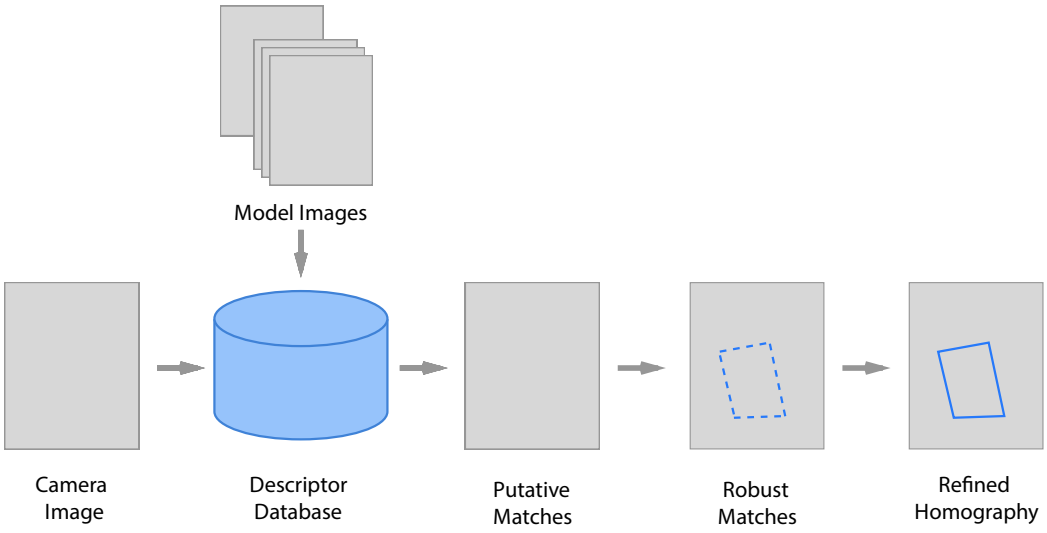


Figure 3.1: Schematic visualization of the detection pipeline.

filtered by performing a *Ratio-Test* (3.3.1) and applying the *RANSAC* Algorithm (3.3.2) for geometric validation. The remaining correspondences are then used to estimate a homography that maps the detected target into the scene image. An initial estimate is given by a normalized version of the DLT algorithm that can be further refined by nonlinear optimization, namely the *Levenberg-Marquardt* algorithm. The resulting detection pipeline is visualized in Figure 3.1.

In the following sections, we consider the different steps of the detection pipeline in more depth.

3.2 Feature Detection and Descriptor Extraction

Our system relies on the recently proposed ORB [RRKB11] feature detector and descriptor extractor. The ORB feature detector is based on FAST interest point detectors and the descriptors are based on BRIEF [RD05, RD06, CLO+12]. As the authors state [RRKB11, Low04], ORB is two orders of magnitude faster than SIFT, while performing equally well. ORB is part of the OpenCV library [Bra00]. This makes it a very promising choice for our implementation.

However, while working with ORB, we noticed a few peculiarities. When observing the spatial distribution of extracted ORB features, we noticed cluster-like distributions around certain regions. This is shown in Figure 3.2. Inside these clusters, the feature-points are spatially very close to each other, which might lead to ambiguous descriptor matches. Upon analyzing these phenomenon more in depth, we came to the following conclusions:

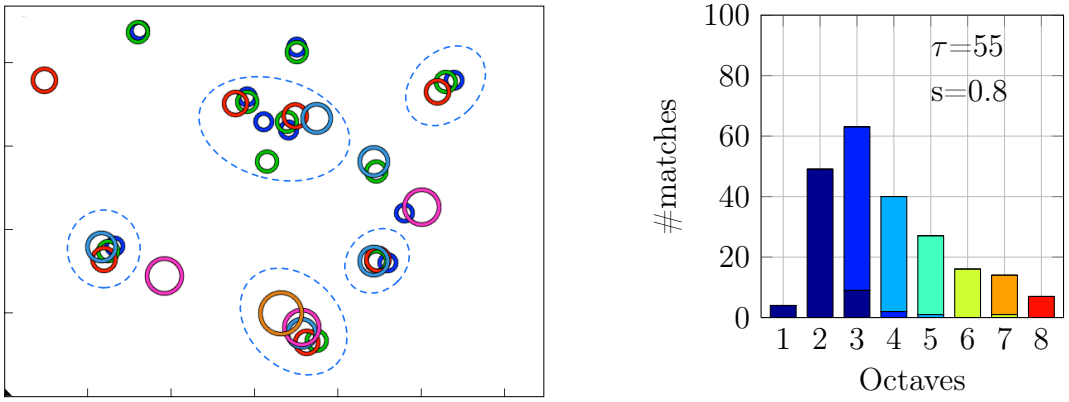


Figure 3.2: *Left:* extraction of ORB feature-points. The color indicates the octave from which the feature-point was extracted. The diameter shows the meaningful neighborhood. One recognizes the formation of clusters (dashed ellipses) as well as the (pairwise) different octave-membership of the feature-points inside a cluster. The grid indicates 10 px in each dimension. *Right:* all descriptors from one octave are matched against descriptors from scaled image. The height of a bar corresponds to the number of matches per octave from the original image, the color designates the octaves from the scaled image.

1. The feature-points inside a cluster origin from different scaling levels (octaves) of the image. This makes sense since the ORB feature extractor uses a scale pyramid of the image to produce multi-scale features.

2. The descriptors of the feature-points inside a cluster are distinctive enough to allow non-ambiguously matches over multiple scales. There are 8 octaves, each one scaled by the factor 1.2.

The first point becomes immediately clear by visually analyzing the octave membership of the extracted feature-points. The results can be seen in Figure 3.2.

For the second point, we performed descriptor matching between an image and a scaled version of the same image at multiple different scalings s. The *Hamming*-Distance was used as distance measure. All matches with a distance larger than a threshold τ were deleted.

If the descriptors inside a cluster are sufficiently discriminative, then the descriptors originating from one scaling level n in the original image should ideally be matched exclusively with descriptors from one scaling level m in the scaled image. Indeed, we could observe such a result.

The effect becomes less important if the chosen threshold value τ is too high. Empirically, we found 55bit to be a good threshold value for the 256bit ORB descriptors. The reader is also referred to Appendix A for more results.

3.3 Descriptor Matching and Outlier Removal

In this section, we describe how our system performs descriptor matching and outlier removal. At this stage the descriptor database of the target images is available as well as the descriptors from the camera-image. Descriptor matching is performed by computing the nearest neighbors using a brute-force strategy from the OpenCV library [Bra00]. Even though ORB performs almost as well as SIFT, which is known to be a very strong descriptor, we still have to deal with wrong matches before robustly estimating the homography [RRKB11]. The identification and removal of wrong matches is performed in two steps: (1) We use the well known ratio-test as proposed by [Low04]; (2) we employ RANSAC, which removes all the matches that cannot be explained by a consistent geometric configuration. Both outlier removal steps are now explained in more detail.

3.3.1 Ratio Test

The ratio-test allows to distinguish reliable matches from unreliable ones. For each descriptor x from the scene image we find the two nearest neighbors x_1 and x_2 originating from the training database. Then the ratio of the two distances gives insight in how similar the descriptors are and hence how reliable the match is. [Low04] recommends to classify all matches as outliers if the distance ratio is greater than 0.8:

$$f = \frac{d_{Hamming}(x, x_1)}{d_{Hamming}(x, x_2)} \begin{cases} \text{if } f < 0.8, & \text{classify as inlier} \\ \text{if } f > 0.8, & \text{classify as outlier} \end{cases} \tag{3.1}$$

According to Lowe, this simple test removes 90% of wrong matches while eliminating less than 5% of correct matches.

3.3.2 RANSAC

Our implementation of RANSAC follows to a great extend the description given in [HZ00] (p.123). We present our approach to the sample selection, which guarantees non-degenerate estimation results. In Chapter 6, we compare our implementation to the original one given in [HZ00].

Distance Measure

We use the *symmetric transfer error* to compute the error of a correspondence from a homography $\mathtt{H}$:

$$d_{\perp} = d(\mathbf{x}_i, \mathtt{H}^{-1}\mathbf{x}_i')^2 + d(\mathbf{x}_i', \mathtt{H}\mathbf{x}_i)^2 \tag{3.2}$$

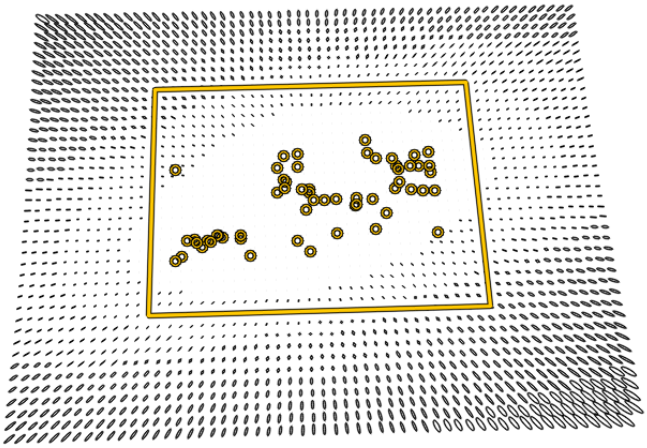


Figure 3.3: Visualization of the extrapolation problem. The accuracy of the transformed points is indicated by the covariance matrices. (See Section 4.5.3). The ellipses illustrate the covariances. The circles show the points used to compute the homography.

Extrapolation Problem An estimated homography accurately maps points from inside the region defined by the points used to compute it, but the accuracy will progressively deteriorate with the distance to this region. This effect is visualized in Figure 3.3.

Sample Selection

In each RANSAC iteration, a random sample of four correspondences is chosen to compute a homography $\mathtt{H}_i$. The sample selection procedure has a direct influence on the quality of the estimated homography. We want to avoid degenerate solutions by disregarding points that are collinear or in direct proximity of each other. Such situations will lead to numerical instabilities during homography estimation and are not unlikely, as we have seen in Section 3.2. Also, the selected points should have a good spatial distribution over the target in order to avoid the *extrapolation problem*. Finally, the homography mapping of the model-image must be convex. To obtain the mentioned characteristics, each random sample (consisting of four correspondences) must fulfill the following checks:

Collinearity While picking the 4 correspondences, we check after selecting each random match whether the newly selected point and the already chosen ones are collinear or not. We start the check as soon as we choose the third random point. Since a perspective transformation preserves parallelism, it is of no importance whether we perform this check on the model- or the scene-image points. Thus

we randomly choose four correspondences $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ one after the other. When the third point p_3 is chosen, we control if the three points (p_1, p_2, p_3) are collinear. When choosing the fourth point, we check if the points (p_1, p_2, p_4) and (p_1, p_3, p_4) are collinear. To perform the test whether three points $p_i = (x_i, y_i), i \in \{1, 2, 3\}$ are collinear we construct the matrix:

$$\mathtt{A} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \tag{3.3}$$

If the determinant of A is zero, then the points are collinear [CG67]. If the test shows that the newly selected point is collinear with the existing ones, a new random match is immediately sampled until the criterion is met. If no other point is available, i.e. all the putative matches are collinear, the ransack routine returns without a result.

Spatial Distribution To limit the effect of the *extrapolation problem* and to avoid spatially too close matches, each newly selected random point must be at least 9 px apart from the already picked random points. This value was chosen as it corresponds to the radius of the FAST feature extractor used by ORB [RRKB11]. As we have seen (Figure 3.2) it is not unlikely that two matches are spatially very close, i.e. less than a few pixels. If this happens, then two rows (or more) of the matrix A will be very similar, which results in an under-constrained homography estimation problem. This will not give a meaningful solution. Thus, it is crucial to avoid such a situation.

Rigidity Constraint After the four sample correspondences $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ are chosen, we can control whether the matches reflect a rigid-body transformation. The shape formed by four point correspondences either forms a fully convex shape or a shape with one concavity. For a homography to correspond to a rigid-body transformation it must map a convex shape to another convex shape and a shape with one concavity to another shape with one concavity. Additionally, in the case of shapes with one concavity, the point correspondences have to be correctly ordered for the concavity to be at the same place. According to [MBSS10], it is sufficient to consider only the direction of the turns along the path formed by the four points. The corresponding directions must be the same along the paths in the model-image and the scene-image. In particular, we consider the cross product of the vectors $\mathbf{V}_n$ and $\mathbf{V}_{n+1}$, which are respectively formed from point $\mathbf{x}_n$ to $\mathbf{x}_{n+1}$ and from point $\mathbf{x}_{n+1}$ to $\mathbf{x}_{n+2}$. The direction of the turn is then given by the sign of the cross product $\mathbf{V}_n \times \mathbf{V}_{n+1}$, which must be the same for corresponding points in both images. The rigidity constraint is then expressed by:

$$sign(\mathbf{V}_n \times \mathbf{V}_{n+1 \bmod 4}) = sign(\mathbf{V}'_n \times \mathbf{V}'_{n+1 \bmod 4}) \tag{3.4}$$

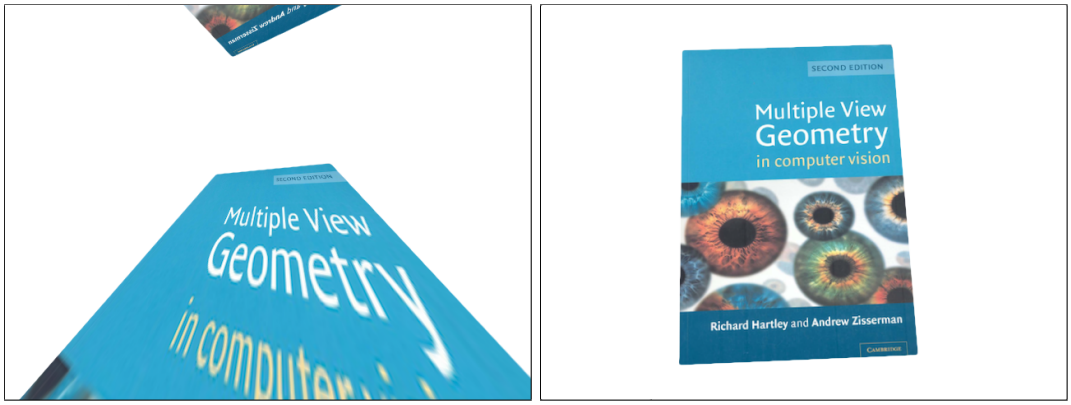


Figure 3.4: Illustration of a homography mappings. *Left:* degenerated mapping *Right:* rigid-body transformation.

Convexity Constraint Because of the extrapolation problem, it cannot be guaranteed that a homography H_i describing a rigid-body transformation for four sample points also describes one for the four corner points of the model-image. Thus, we apply the computed H_i to the corner-points of the model-image and control whether the projected corners still form a convex quadrangle. Since the convexity constraint is a special case of the rigid-body constraint, we can again make use of equation (3.4). Figure 3.4 shows an example of a degenerated homography mapping that is anticipated by this approach.

As this approach is no longer fully random, we will refer to it as *guided*-sample-selection as opposed to *random*-sample selection.

Number of Samples

The number N of samples is recomputed in each RANSAC iteration:

$$N = \frac{log(1-p)}{log(1-e^s)} \tag{3.5}$$

where $p = 0.995$ is the probability that at least one of the random samples is free from outliers, e is the probability that a randomly selected correspondence is an inlier and $s = 4$ is the minimum number of correspondences needed to compute a homography. It is equivalent to interpret e as the current best inlier ratio, which is recomputed whenever we find a homography with a larger consensus set, i.e. a higher inlier count.

Once we have identified all the outliers from the putative matches, we robustly estimate a homography `H` based on the inlier matches. This procedure is described in the next section.

3.4 Homography Estimation

In this section, we consider the problem of homography estimation. As we have seen in Section 2.1, a homography $\mathtt{H}$ projects a set of points $\mathbf{x}_i$ of one plane onto a set of points $\mathbf{x}'_i$ from another plane. See Figure 2.2 for a visualization. In the following we present two approaches for homography estimation: First, the *normalized DLT* algorithm, which gives a direct linear solution and second, an iterative non-linear estimation based on the *Levenberg-Marquardt* algorithm.

We will now motivate why we use these two different approaches. The advantage of DLT is a direct, linear (and thus unique) solution. If normalization is applied, the DLT does give very good results [HZ00]. The downside of the DLT algorithm is that it minimizes the *algebraic error* $\|\mathtt{A} \cdot \mathbf{h}\|$. The algebraic error is related to, but is not quite the same as a *geometric error*, which we actually want to minimize. In order to minimize an arbitrary geometric error function, one uses more elaborated iterative optimization algorithms. These methods usually need a good initialization. If not, the optimization procedure can get stuck in a non-optimal local minimum. A good initialization is usually given by a linear method such as the normalized DLT. We use the solution of the DLT algorithm as a first solution and LM for the final polish.

3.4.1 Direct Linear Method: Normalized DLT

Here we present the normalized DLT for 2D homographies. Data normalization gives dramatically better results [HZ00]. Instead of directly applying the DLT algorithm to the point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, a normalization process is first applied. In particular, the points are translated and scaled so that their centroid corresponds to the origin and their average distance to the origin is $\sqrt{2}$. This transformation is applied to each point set independently. Thus we obtain a transformation $\mathtt{T}$ that maps the points $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$ and a transformation $\mathtt{T}'$ that maps $\mathbf{x}'_i$ to $\tilde{\mathbf{x}}'_i$. We have:

$$\mathtt{T} = \begin{bmatrix} s & 0 & -t_x \\ 0 & s & -t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

with

$$t = (t_x, t_y) = \frac{1}{n} \sum_i (x_i, y_i) \tag{3.7}$$

$$s = \frac{\sqrt{2}}{\frac{1}{n} \sum_i \sqrt{(x_i - t_x)^2 + (y_i - t_y)^2}} \tag{3.8}$$

where n is the number of points.

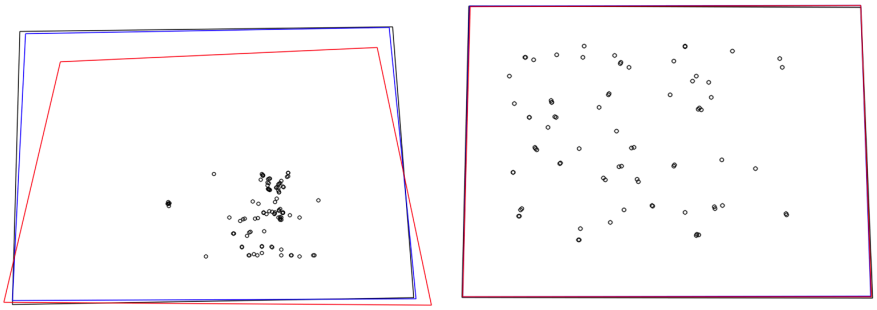


Figure 3.5: An exemplary illustration comparing the results of the normalized and unnormalized DLT algorithm. The polygons correspond to the contours of the transformed target in the scene image. The groundtruth is visualized in black, the normalized result in blue and the unnormalized in red. The difference is less notable if the feature-points *(black circles)* used to compute the homography are uniformly distributed *(compare left and right)*.

The transformation $\mathtt{T}'$ is constructed in the same way but we use $\mathbf{x}'_i = (x'_i, y'_i)$ instead of $\mathbf{x}_i = (x_i, y_i)$. After the normalization step, the DLT algorithm is applied to the resulting point correspondences $\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{x}}'_i$. This yields a homography $\tilde{\mathtt{H}}$ that is again denormalized to obtain the desired homography $\mathtt{H}$:

$$\mathtt{H} = \mathtt{T}'^{-1} \cdot \tilde{\mathtt{H}} \cdot \mathtt{T} \tag{3.9}$$

Figure 3.5 compares the normalized and unnormalized DLT algorithm. It also visualizes the influence that the geometric distribution of the feature-points has on the homography estimation and how the normalized DLT is more robust against such influence.

3.4.2 Iterative Non-Linear Method: Levenberg-Marquardt

The Levenberg-Marquardt (LM) algorithm is a non-linear iterative minimization method [HZ00] (p.600). Given an *initial solution*, a set of *parameters* is iteratively refined in order to minimize a certain *cost function*. Like any non-linear iterative optimization algorithm it has certain disadvantages compared to linear methods such as the previously presented normalized DLT algorithm. In general, iterative methods are slower due to their iterative nature. Also they need a good initialization in order to converge and to end up in a local minimum. The main advantage of non-linear approaches is a better solution in the sense that they minimize over a more meaningful error function. For example, in our case we minimize the *symmetric transfer error* (Equation 3.11, [HZ00] p.94), which has a direct geometric interpretation, whereas the DLT algorithm minimizes the algebraic error, which is not geometrically meaningful. Linear methods still give

useful results as we have seen in Section 3.4.1. We use the result from the normalized DLT as initialization for our non-linear optimization method. As parameters we choose the entries of $\mathtt{H}$, which are stored in the parameter vector $\mathbf{h}$. These parameters are optimized using the cost function given by:

$$\|\mathbf{X} - f(\mathbf{h})\|^2 \tag{3.10}$$

which is equal [HZ00] (p. 112) to the symmetric transfer error:

$$\sum_i d(\mathbf{x}_i, \mathtt{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, \mathtt{H}\mathbf{x}_i)^2 \tag{3.11}$$

where $\mathbf{X}$ is a $4n$-vector made up of the inhomogeneous coordinates of the points $\mathbf{x}_i$ followed by the inhomogeneous coordinates of the points $\mathbf{x}'_i$. The function f relates the parameters $\mathbf{h}$ to the costs:

$$f : \mathbf{h} \mapsto (\mathtt{H}^{-1}\mathbf{x}'_1, \ldots, \mathtt{H}^{-1}\mathbf{x}'_n, \mathtt{H}\mathbf{x}_1, \ldots, \mathtt{H}\mathbf{x}_n) \tag{3.12}$$

For the actual implementation of Levenberg-Marquardt we relied on the well-known Eigen library for linear algebra [GJ+10]. The Levenberg-Marquardt method is provided with a vector `fvec` representing equation (3.4) and its Jacobian `fjac` with respect to the parameters $\mathbf{h}$. The vector `fvec` is given by:

$$\mathtt{fvec} = \left[\begin{array}{c} x'_1 - \tilde{x}_1 \\ y'_1 - \tilde{y}_1 \\ \vdots \\ x'_n - \tilde{x}_n \\ y'_n - \tilde{y}_n \\ \hline x_1 - \tilde{x}'_1 \\ y_1 - \tilde{y}'_1 \\ \vdots \\ x_1 - \tilde{x}'_1 \\ y_1 - \tilde{y}'_1 \end{array}\right] \tag{3.13}$$

where $\tilde{\mathbf{x}}_i = (\tilde{x}_i, \tilde{y}_i)$ and $\tilde{\mathbf{x}}'_i = (\tilde{x}'_i, \tilde{y}'_i)$ are the inhomogeneous coordinates of $\mathtt{H}^{-1}\mathbf{x}'_i$ and $\mathtt{H}\mathbf{x}_i$ respectively. In this section $\mathbf{x}_i = (x_i, y_i)$ and $\mathbf{x}'_i = (x'_i, y'_i)$ represent the *inhomogeneous* coordinates of the measured points. The Jacobian `fjac` is computed as:

$$\mathtt{fjac} = \frac{\partial \mathtt{fvec}}{\partial \mathbf{h}} \tag{3.14}$$

Similar to the DLT algorithm, we can again apply data normalization. However, we could not observe a similar improvement as we did in the case of the normalized

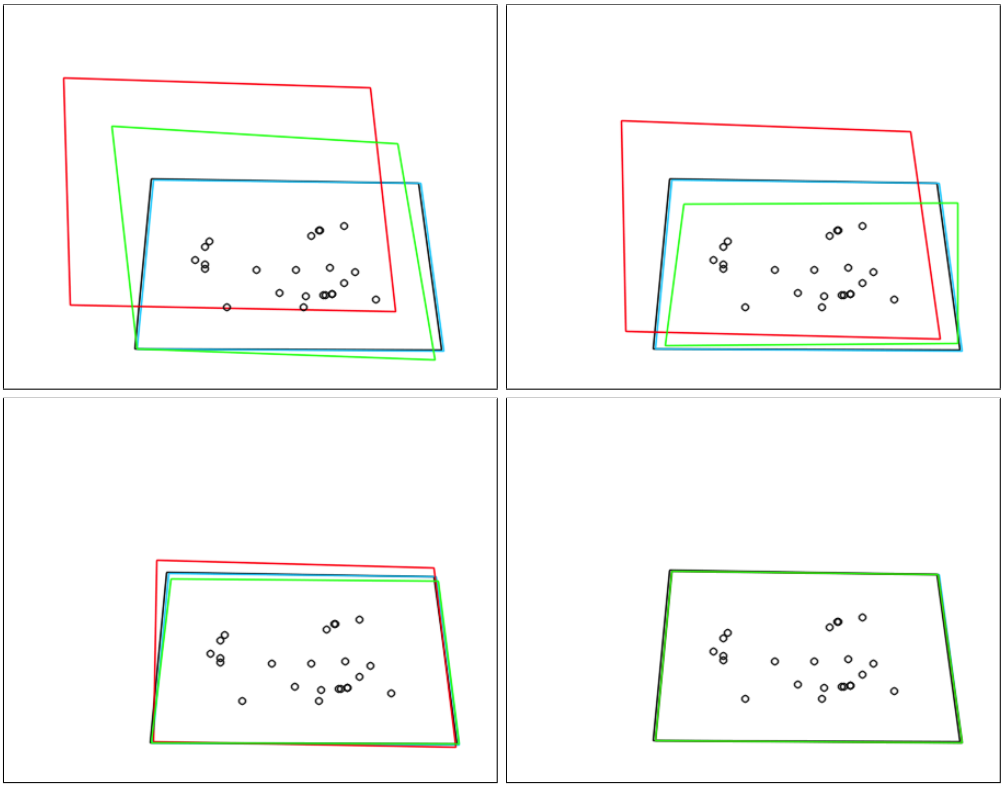


Figure 3.6: Illustration of consecutive LM iterations *(from left to right and top to bottom)* starting from the identity homography. It shows that the normalized LM *(green)* converges faster than the unnormalized LM *(red)*. Eventually, both solutions become visually indistinguishable from the solution given by the normalized DLT *(blue)*. The circles show the inliers used to compute the homographies. More extensive evaluations on DLT and LM are given in Section 6.4.

DLT algorithm. To our findings, data normalization has little to no impact on the accuracy of the LM estimator. We observed, however, a faster convergence if data normalization is performed. Figure 3.6 illustrates this observation. For a better visualization, we used the identity homography for initial parameter values. In practice, one would use the solution given by the normalized DLT algorithm.

Chapter 4

Tracking

Tracking considerably increases the robustness and accuracy of our system by exploiting temporal coherencies. During detection, we performed wide-baseline matching, i.e. establishing correspondences between a pair of images taken from different viewpoints. For tracking, we will also consider short-baseline matching, which finds correspondences between a pair of images taken from similar viewpoints as it is the case for consecutive frames in a video sequence. In general, short-baseline matching is easier to solve than wide-baseline matching: a quite good estimation for the location and appearance of the target is already given in the previous frame. This chapter explains how the system benefits from this additional information by using a *hierarchical* matching approach. Finally, the tracking setup imposes a smooth and consistent dynamic behavior on the tracked target by employing a Kalman filter with underlying constant-velocity model.

4.1 Overview

Our tracking approach aims to maximize the number of correct correspondences between the model-image and camera-image. This is achieved through a hierarchical matching strategy using different matching techniques: (1) *Inter-frame matching* (see Subsection 4.3), which finds corresponding feature-points in two consecutive camera-images from the input video sequence; (2) *warped-target matching* (see Subsection 4.3), which finds correspondences between the current frame and the model-image under perspective distortion; (3) *database matching* (see Subsection 3.2) finds matches between the camera-image and the model-image stored in the database. This is the same method as used during detection. Table 4.1 gives an overview of the matching hierarchy.

Since our goal is to establish feature-point correspondences between the current camera-image and the model-image we need to remap matched feature-points originating from the previous frame or from the warped-target frame back into

Matching	Appearance	Location	Homography
1. Inter-Frame	same	same	depends
2. Warped-Target	differs	similar	recent
3. Database	differs	differs	not needed

Table 4.1: Overview of the matching hierarchy. The characteristics of corresponding feature-points and the necessity for prior knowledge on the homography are compared.

the original model-image space. This remapping step requires careful implementation (see Subsections 4.3 and 4.4) to avoid error accumulation and likely drift artifacts.

A value is assigned to each feature-point from the current camera-image that states the current knowledge about its correspondence situation: (0) The feature-point is considered to be an outlier, there is no corresponding point on the target; (1) the point is an inlier and the corresponding point on the target is known; (2) a candidate for this point was found on the target, but geometrical verification is still needed; (3) we did not yet find a match for this point. After extracting the feature-points from a new frame, each point is assigned the value 3. The values are updated while the feature-points are handed down through the matching hierarchy in anticipation of finding a valid correspondence. As soon as a correspondence for a feature-point is found on one level, this point is no longer considered in the following matching levels. This makes sense if we assume that the earlier levels give more reliable results than the later ones.

We now explain the tracking pipeline, i.e. the processing of each frame (compare Figure 4.1). We start by extracting feature-points and computing their descriptors from the current camera-image. These will serve as input to the target detection routine as explained in chapter 3. The tracking system is initialized, as soon as the detector has identified a target. For each detected target, homography estimation is performed on the database matches provided by the target detection routine. Finally, the estimated homography is used to initialize Kalman filtering and to draw the graphical overlay. Then, we proceed to the next frame: For each tracked target, we only consider feature-points lying in the area that is predicted by the Kalman filter. Now, the matching hierarchy comes into play: if the target detector did not identify a different target than in the previous frame, we perform inter-frame matching followed by warped-target matching. Matching against the database is always performed as a last step. Notice that object detection must not necessarily yield a solution: if no object is detected it is assumed that matching against the database failed but inter-frame matching and warped-target matching can still yield meaningful results. Those feature-points

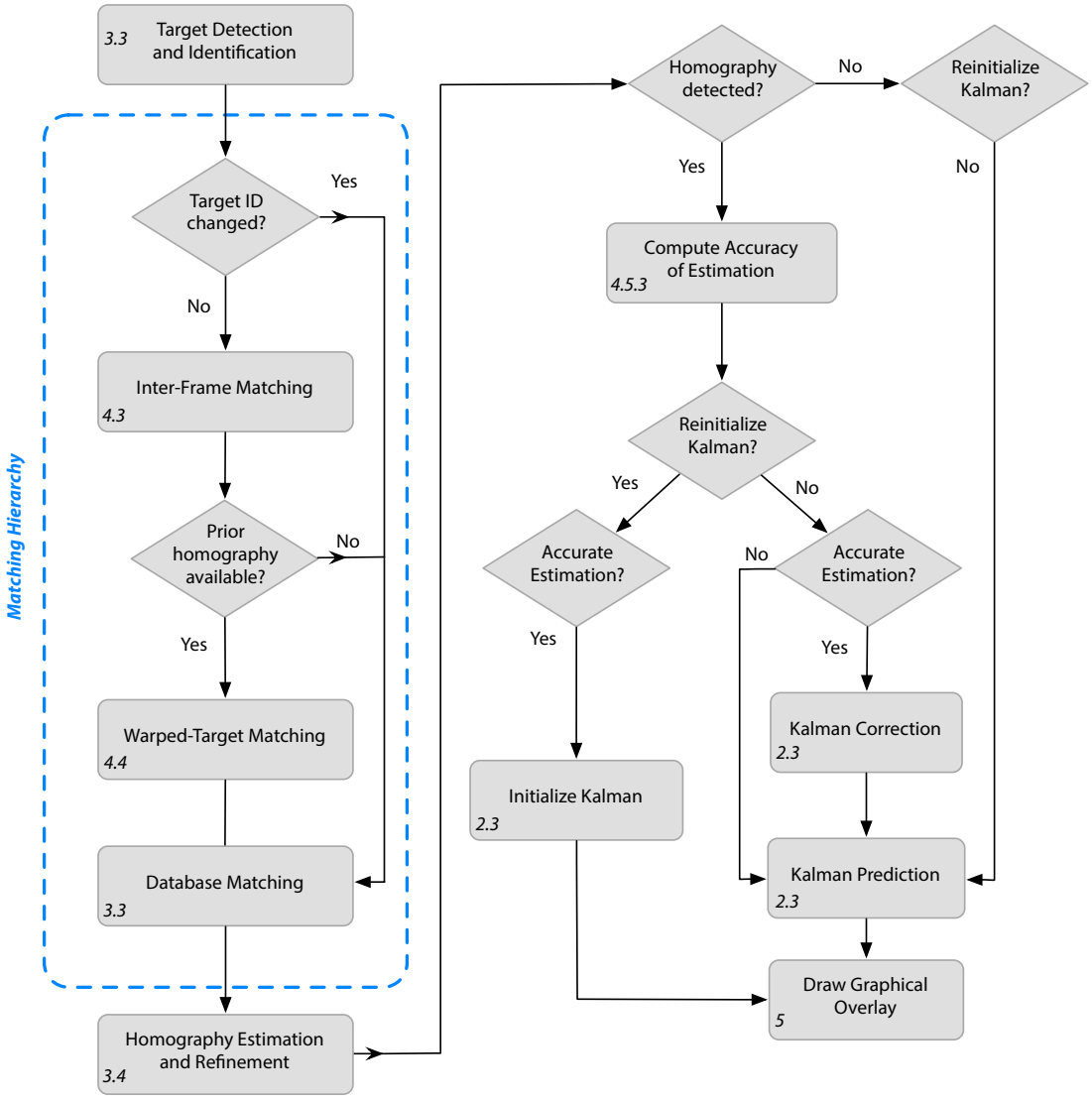


Figure 4.1: Overview of the tracking pipeline for one target.

that were not yet considered for matching, i.e. those that are spatially not close to a previously detected target, are matched against the database in order to detect new targets, which become visible to the camera.

After establishing the correspondences, we iterate the following two steps: (1) a homography is estimated using DLT based on the found matches and optimized with LM; (2) the inliers of the estimated homography are recomputed using the symmetric transfer error (eq. 3.2). These two steps are repeated until the number of inliers stays constant. A convexity test (see Subsection 3.3.2) is used to check if the resulting homography is a valid rigid-body transformation. In the next step, the accuracy of the estimated homography is computed. In particular, we are interested in how accurately the corners of the target are localized in the camera-image. Section 4.5.3 explains how this is done. Knowing the accuracy of the detected target points is not only a valuable input for the Kalman filter, it also helps to decide whether it is safe to use the estimated homography for mapping feature-points from the camera-image to the model-image during inter-frame matching in the following frame. This aspect is explained in more detail in Section 4.3.

4.2 Grid Matching

To exploit spatial coherencies between two images, we introduce grid matching. For inter-frame matching and warped-target matching, we assume that corresponding image points are spatially close together. Thus, it is sufficient to perform matching only on nearby feature-points within a certain search radius r. For efficiency, the image space is discretized using a 2D grid. The side length of each square cell in the grid is r. Assume we want to match a feature-point $\mathbf{x}'$ from the current camera-image against feature-points $\mathbf{x}_i$ from the previous camera-image. First, we place identifiers for each point $\mathbf{x}_i$ in its 3x3 cell neighborhood. Finding a match for $\mathbf{x}'$ is then performed by considering only those feature-points whose identifies are located in the same cell as $\mathbf{x}'$. The obtained candidates are further refined: They have to be within distance r of $\mathbf{x}'$ and they have to originate from the same scale octave as $\mathbf{x}'$ (we assume the target is visible at the same scaling in both images, also compare Section 3.2). The final match is the nearest neighbor according to the Hamming distance between their descriptors. As each feature-point is matched against few, spatially-close candidates, the likelihood of a false match is reduced, assuming the correct candidate is in the set of candidates. The procedure is visualized in Figure 4.2.

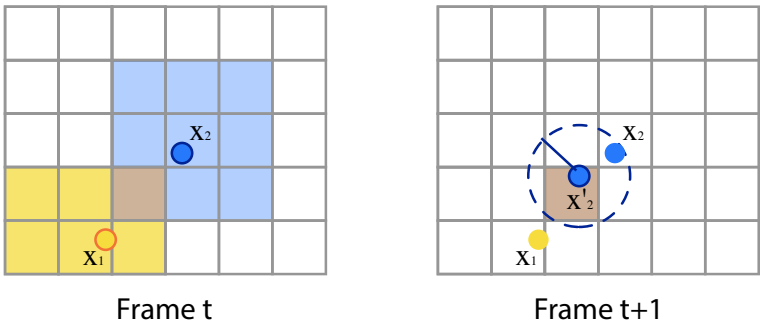


Figure 4.2: Illustration of grid matching. *Left:* the identifiers of previous feature-points $\mathbf{x}_1$ and $\mathbf{x}_2$ are placed in their 3x3 neighborhood. *Right:* the identifiers of the potential candidates for a match are contained in the same cell as $\mathbf{x}'_2$. The dashed circle indicates the search radius r.

4.3 Inter-Frame Matching

To our findings, the most reliable matches are obtained between two consecutive input video frames. The position and appearance of corresponding feature-points vary inherently little, which greatly simplifies the task of matching. This approach can also be understood as a frame-to-frame feature-point tracker for features that origin from the same target as detected in the previous frame. Thus, inter-frame matching is only performed if no different target than the previous one is detected. The correspondences $\mathbf{x}_t \leftrightarrow \mathbf{x}_{t-1}$ between two consecutive frames are established using grid matching. Actually, we are not interested in inter-frame correspondences, but in correspondences between the current camera-image and the model-image: assuming the existence of a match $\mathbf{x}_{t-1} \leftrightarrow \mathbf{x}'$ with $\mathbf{x}'$ being a point in the model-image, we can simply transfer the match to the current feature-point $\mathbf{x}_t$, which gives us the new correspondence $\mathbf{x}_t \leftrightarrow \mathbf{x}'$.

Since the model-image and the target in the current camera-image can have dramatically different appearances, it is by no means guaranteed that every matched point $\mathbf{x}_{t-1}$ also has a corresponding point in the model-image. In fact the number of feature-points that have no corresponding match in the model-image largely outnumbers those that do. Thus, even though we might have a large number of correspondences between two frames, we cannot necessarily make use of them. Unless the homography $\mathtt{H}_{t-1}$ from the previous frame is known with sufficient accuracy, then we can use the inverse of $\mathtt{H}_{t-1}$ to map the matched points $\mathbf{x}_{t-1}$ back into the model-image. The parameter τ_{HU} defines the threshold that decides when a homography estimation is considered too uncertain. (see Section 4.5.3). The threshold should be low enough to avoid drifting effects. If the uncertainty of the previous homography is too important, we cannot find a match and continue with the next level in the hierarchy.

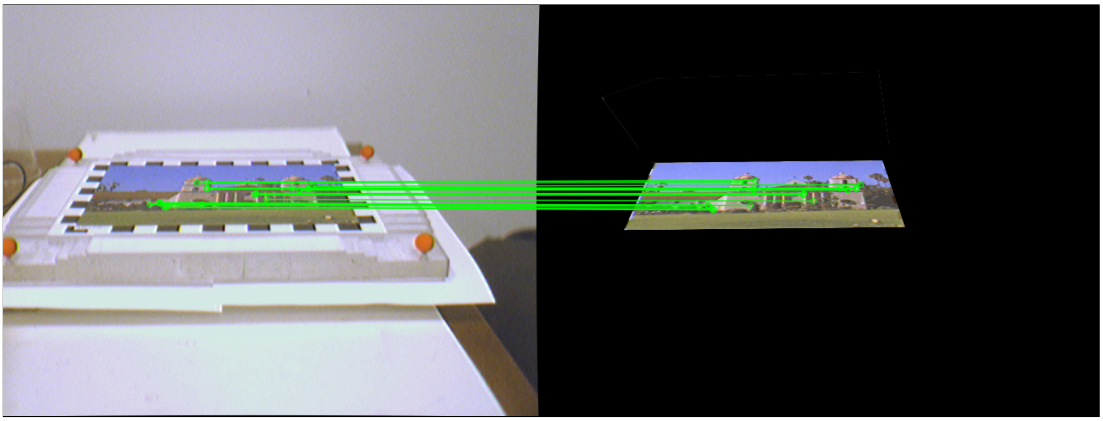


Figure 4.3: Illustration of warped target matching under perspective distortion. *Left:* camera image from UCSB [GHT11] video sequence. *Right:* warped model-image using previously estimated homography.

4.4 Warped-Target Matching

The idea behind warped-target matching is to become robust against significant perspective distortion. As most descriptors, ORB is not invariant to perspective distortion. By warping the model-image into a similar position as the target in the camera-image, this limitation is successfully bypassed. A similar position is obtained by using an estimated homography from a recent previous frame. A more recent homography will better correspond to the perspective transformation of the target in the current frame. We extract the feature-points and compute the descriptors that are then matched against the descriptors from the current frame. If the last detected homography is not older than a few frames, the warped target should be geometrically close to the target in the current camera-image. Thus, we can again employ the grid matching approach. If this is not the case, greedy matching will be used instead. When a new match is found for a feature-point that was already successfully matched in the previous frame, it is only updated if the Hamming-distance of the new match is smaller than the one of the previously found match. A visualization for warped target matching is given in Figure 4.3. Furthermore, model-points that are not visible in the camera-image will also not be visible in the warped-target image, limiting the number of potential matching candidates. Unlike inter-frame matching, this method does not need to cope with drifting problems: we use a homography $\mathtt{H}$ to warp the model-image and establish matches between the camera-image and the warped model-image. If a feature-point in warped model-image was matched, we use $\mathtt{H}^{-1}$ to map the feature-point back to the model-image. This procedure introduces no errors that are due to erroneous homography estimation.

4.5 Kalman Filter Tracking

Our next and final tracking component is a Kalman filter. The contribution of the Kalman filter is two fold: First it helps to minimize jitter from the detected tracks due to its smoothing characteristics. Second, it allows to bridge over single frames for which no homography could be detected. To our findings, this happens frequently when strong motion blur makes it impossible to extract feature-points. When designing a Kalman filter, one needs to decide over what parameters to apply the filter. The entries of the homography seem to be an obvious choice, however this choice turns out to be non-optimal as the entries of the homography do not allow an imminent geometric interpretation. This makes it difficult to design a dynamic model for these parameters. Since a homography is fully defined by four points we choose to apply the Kalman filter directly to the four corner-points of the target. In particular, we use four Kalman filters independently, one for each corner. This alternative allows a simple formulation of the underlying dynamic model but ignores the geometric correlation between the corner points.

4.5.1 State and Observation Vectors

The state vector $\mathbf{X}$ and the observation vector $\mathbf{Y}$ are defined as follows:

$$\mathbf{X} = (p_x, p_y, v_x, v_y)^\top \tag{4.1}$$

$$\mathbf{Y} = (p_x, p_y)^\top \tag{4.2}$$

The state $\mathbf{X}$ contains the position $\mathbf{p} = (p_x, p_y)^\top$ and the velocity $\mathbf{v} = (v_x, v_y)^\top$ of the corner point. In a single frame, we can only observe the position of a corner, not the velocity. The measurements $\mathbf{c}^m$ of a corner position $\mathbf{c}$ is obtained by first estimating a homography $\mathtt{H}$ as described in Section 3.4 and then compute $\mathbf{c}^m = \mathtt{H} \cdot \mathbf{c} = (c_x^m, c_y^m, c_w^m)^\top$, which gives $(p_x, p_y)^\top = (c_x^m / c_w^m, c_y^m / c_w^m)^\top$.

4.5.2 Linear Dynamic Model

We approximate the movement of the target with a constant velocity model. The dynamic model $\mathtt{D}$ is given by the following matrix:

$$\mathtt{D} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

where dt is the time difference between two consecutive frames. We assume dt to be constant as we apply our system to video sequences with constant frame rates. In a real-time system, dt has to be recomputed for each new frame.

The measurement model describes which parameters of the state vector can directly be observed or measured. In our application, we can only measure the position of the target corners, not the velocities. This is reflected in the measurement model matrix $\mathtt{M}$:

$$\mathtt{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.4}$$

The accuracy depends on a lot of factors, including the number of points, the correctness of the matches as well as the geometric distribution of the points. For example, if all the points lie on one line, then the estimated homography will most probably be degenerated. In the next section, we will see how to express this uncertainty.

4.5.3 Measurement Uncertainty

The goal of this section is to calculate the uncertainty $\Sigma_{\mathtt{M}_t}$ associated with a point transformed under a given homography $\mathtt{H}$. This uncertainty will serve as value for $\Sigma_{\mathtt{M}_t}$ in the Kalman filter. Specifically, we want to compute a covariance matrix for $\mathbf{x}'_i$ where $\mathbf{x}'_i = \mathtt{H} \cdot \mathbf{x}_i$. We use the approach from [HZ00] as follows:

1. Estimate a homography $\mathtt{H}$ given point correspondences $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$.
2. Compute the Jacobian matrix $\mathtt{J} = \partial \mathbf{X}' / \partial \mathbf{h}$.
3. Compute the covariance matrix of $\mathbf{h}$ given by $\Sigma_{\mathbf{h}} = (\mathtt{J}^\top \Sigma_{\mathbf{X}'}^{-1} \mathtt{J})^+$.
4. Finally, the covariance matrix for the point $\mathbf{x}'$ is given by $\Sigma_{\mathbf{x}'} = \mathtt{J}_{\mathbf{h}} \cdot \Sigma_{\mathbf{h}} \cdot \mathtt{J}_{\mathbf{h}}^\top$

We shortly introduce the above used notation: $\mathbf{X}'$ is the set of all points $\mathbf{x}'_i$. The covariance matrix $\Sigma_{\mathbf{X}'}$ corresponds to the measurement uncertainty of the points $\mathbf{x}_i$. The pseudo inverse $\mathtt{X}^+$ is defined as:

$$\mathtt{X}^+ = \mathtt{X}^{+\mathtt{A}} = \mathtt{A} \cdot (\mathtt{A}^\top \cdot \mathtt{X} \cdot \mathtt{A})^{-1} \cdot \mathtt{A}^\top \tag{4.5}$$

For our specific case, we can assume that the points $\mathbf{x}_i$ are known exactly, as they correspond to the corners of the target image. This allows us to use the simpler case of error in the scene image only. Further, we assume that points $\mathbf{x}'_i$ have one pixel standard deviation. This means the covariance matrix $\Sigma_{\mathbf{X}'}$ is the identity matrix.

We now present the procedure to actually perform the aforementioned steps. First, we estimate $\mathtt{H}$ as explained in Section 3.4 and normalize it so that $\|\mathtt{H}\|^2 = 3$. Then, we compute the Jacobian matrix $\mathtt{J} = \partial \mathbf{X}' / \partial \mathbf{h}$ where the vector $\mathbf{h} \in \mathbb{R}^9$ contains the entries of $\mathtt{H}$. A formula for $\partial \mathbf{x}'_i / \partial \mathbf{h}$ is given by:

$$\mathtt{J}_i = \partial \mathbf{x}'_i / \partial \mathbf{h} = \frac{1}{w'_i} \begin{bmatrix} \mathbf{x}_i^\top & \mathbf{0}^\top & -x'_i \cdot \mathbf{x}_i^\top \\ \mathbf{0}^\top & \mathbf{x}_i^\top & -y'_i \cdot \mathbf{x}_i^\top \end{bmatrix} \tag{4.6}$$

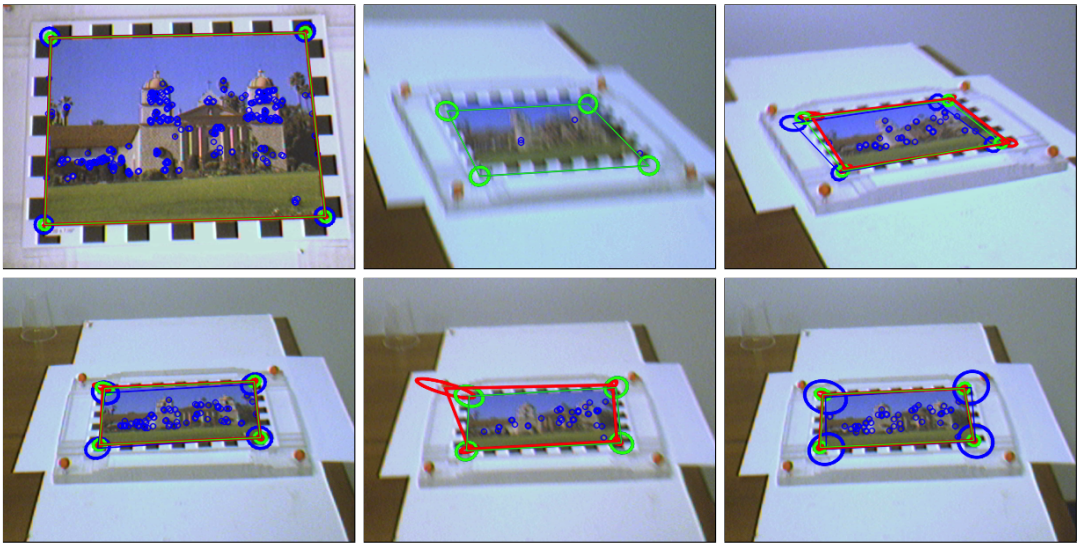


Figure 4.4: Illustrations of the Kalman filter in action. The red polygon shows the current measurement, the blue one is the prediction from the Kalman filter and the green one corresponds to the corrected state. The covariances $\Sigma_{\mathbf{x}'} = \Sigma_{\mathtt{M}_t}$, $\Sigma^{-}_{\mathbf{x}'_t}$ and $\Sigma^{+}_{\mathbf{x}'_t}$ are visualized with red, blue and green ellipses respectively. The major diameter of the ellipse is directly proportional to the first eigenvalue of the corresponding covariance matrix. To make the covariances easily visible, we choose a scaling factor of 6. *First row, from left to right:* (1) Good measurement, prediction and correction. (2) No homography was detected, the corrected state corresponds to the predicted one. (3) The corrected state is interpolated between the predicted one and the current measurement. *Second row:* sequence of three consecutive frames. In the middle frame, the estimation of the homography, i.e. measurement, is inaccurate. The Kalman filter successfully compensates the inaccuracy.

Note that we use homogeneous coordinates, in particular we have $\mathbf{x}_i = (x_i, y_i, 1)^\top$ and $\mathbf{x}' = (x'_i, y'_i, w'_i)^\top$. Then, the complete Jacobian $\mathtt{J}$ is obtained by stacking the matrices for all points $\mathbf{x}_i$ on top of each other so that $\mathtt{J} = (\mathtt{J}_1^\top, ..., \mathtt{J}_i^\top, ..., \mathtt{J}_n^\top)^\top$. As motivated in [HZ00], we can now compute the covariance matrix $\Sigma_{\mathbf{h}}$ using a Householder matrix $\mathtt{A}$ corresponding to the vector $\mathbf{h}$. As we assumed $\Sigma_{\mathbf{X}'}$ to be the identity matrix, the covariance $\Sigma_{\mathbf{h}}$ is given by:

$$\Sigma_{\mathbf{h}} = (\mathtt{J}^\top \mathtt{J})^{+\mathtt{A}_1} = \mathtt{A}_1(\mathtt{A}_1^\top(\mathtt{J}^\top \mathtt{J})\mathtt{A}_1)^{-1}\mathtt{A}_1^\top \tag{4.7}$$

where $\mathtt{A}_1$ denotes the first 8 columns of the Householder matrix $\mathtt{A}$ given by

$$\mathtt{A} = \mathtt{I} - 2 \cdot \frac{\mathbf{v}\mathbf{v}^\top}{\mathbf{v}^\top \mathbf{v}} \tag{4.8}$$

where $\mathtt{I} \in \mathbb{R}^{9\times 9}$ is the identity and $\mathbf{v} = \mathbf{h} + \|\mathbf{h}\|\mathbf{e}_9$ with $\mathbf{e}_9 = (0, ..., 0, 1)^\top \in \mathbb{R}^9$.

Finally, we can compute the covariance matrix for the points $\mathbf{x}'$ given by the formula:

$$\Sigma_{\mathbf{x}'} = \mathrm{J}_{\mathbf{h}} \Sigma_{\mathbf{h}} \mathrm{J}_{\mathbf{h}}^{\top} \tag{4.9}$$

Figure 4.4 visualizes a few examples. We can now directly use this covariance matrix to decide if the estimated homography is accurate or not: For each corner point, we test if the first eigenvalue of its covariance matrix exceeds a threshold τ_{HU}. If one corner fails the test, the estimated homography is classified as too inaccurate. The test is performed corner wise to guarantee that all corner points are mapped accurately. Figure 4.4 (bottom, middle) shows an example of an inaccurate homography for which one corner failed the test. This information is used during inter-frame matching to decide if the previously estimated homography can be used to back-project camera-image points into the model-image.

Chapter 5

Graphical Overlay

This chapter describes how to overlay the scene image with additional information, such as 3D objects (See Figure 5.1). This corresponds to the *augmentation* step in the pipeline of our augmented reality system. The goal of this chapter is to compute the pose of the target inside the scene image relative to the camera. By pose we denote a 6DoF vector composed of the rotation and translation of the target, which can be used as input by a subsequent OpenGL routine. The overall approach consists in decomposing a given homography `H` into a rotation matrix `R` and a translation vector $\mathbf{t}$. We will also need the intrinsic camera parameters `K`, which are determined in an offline camera-calibration step.

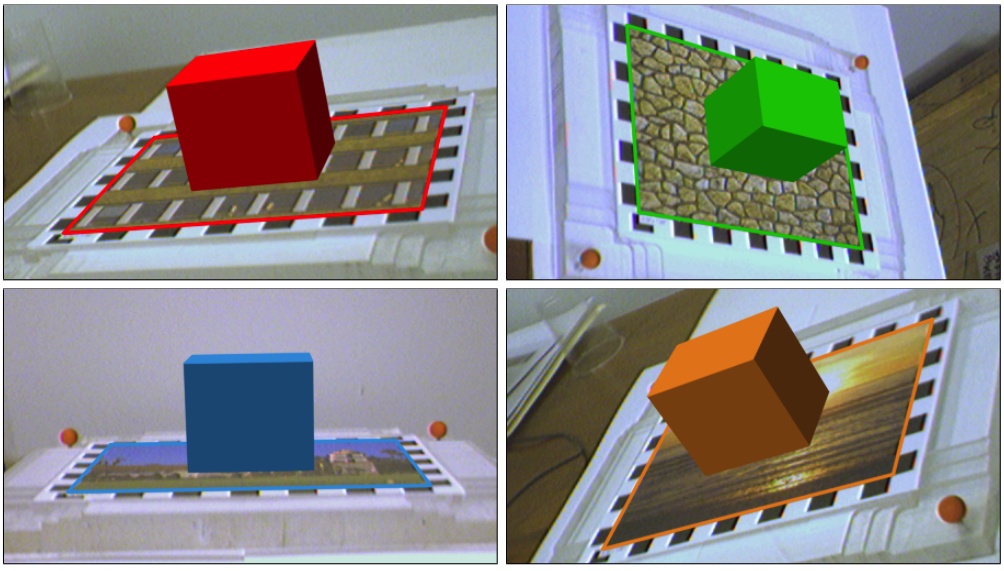


Figure 5.1: Illustrations of the graphical overlay. In our case, we draw a 3D cuboid on the middle of the detected target object. The pose computations are based on the homography represented by the rectangle. The colors denote different targets.

5.1 Overview

A homography $\mathtt{H}$ describes a *mapping* of points from a 2D image plane onto another 2D image plane. Homography estimation has been covered to a great extend in Section 2.1 and 3.4 . However, at this point, we are more interested in a *projection* that projects 3D model points $\mathbf{X}_i$ onto a 2D image plane. Since we are working with planar targets we cannot directly apply the DLT algorithm as we did for homography estimation. This is due to numerical instabilities described in [HZ00] (pp. 179-180). Thus our approach relies on decomposing the already estimated homography $\mathtt{H}$. The resulting closed-form solution can again serve as a starting point for an iterative algorithm such as Levenberg-Marquardt.

5.2 From Homography to Pose

We now describe our approach to homography decomposition in order to recover the target pose. We first introduce the notation used in this section. A pose $\mathtt{P}_i = [\mathtt{R}_i|\mathbf{t}_i] \in \mathbb{R}^{3\times4}$ consists of a rotation matrix $\mathtt{R} \in \mathbb{R}^{3\times3}$ and a translation $\mathbf{t} \in \mathbb{R}^3_i$. Together with the corresponding camera-matrix $\mathtt{K}_i$ the pose $\mathtt{P}_i$ projects a 3D point $\mathbf{X} = (X, Y, Z, 1)^\top$ to its 2D counterpart $\mathbf{x} = (x, y, w)^\top$. This relation is formalized by:

$$\mathbf{x} = \mathtt{K}_i \cdot \mathtt{P}_i \cdot \mathbf{X} \tag{5.1}$$

Note that $\mathbf{x}$ and $\mathbf{X}$ are given in homogeneous coordinates. We now assume that $\mathbf{X}$ lies on the surface of a target. Then the model-image points $\mathbf{x}$ and scene-image points $\mathbf{x}'$ are defined as:

$$\mathbf{x} = \mathtt{K}_0 \cdot [\mathtt{R}_0|\mathbf{t}_0] \cdot \mathbf{X} \tag{5.2}$$

$$\mathbf{x}' = \mathtt{K}_1 \cdot [\mathtt{R}_1|\mathbf{t}_1] \cdot \mathbf{X} \tag{5.3}$$

where $\mathtt{K}_0$ and $\mathtt{K}_1$ correspond to the cameras recording the model-image and the scene-image respectively. Similarly, $[\mathtt{R}_0|\mathbf{t}_0]$ and $[\mathtt{R}_1|\mathbf{t}_1]$ describe the rotation and translation of the cameras relative to the target. We recall that a homography $\mathtt{H}$ (defined up to a scale factor λ) maps a point $\mathbf{x}$ onto $\mathbf{x}'$.

$$\mathbf{x}' = \lambda \cdot \mathtt{H} \cdot \mathbf{x} \tag{5.4}$$

The goal now is to find a relation between the known homography $\mathtt{H}$ and the pose $\mathtt{P}_1 = [\mathtt{R}_1|\mathbf{t}_1]$ in order to express $\mathtt{R}_1$ and $\mathbf{t}_1$ in terms of the entries of $\mathtt{H}$. The model images are recorded under controlled conditions, hence the pose $\mathtt{P}_0$ is known. In particular, we used an orthogonal projection $\mathtt{K}_0$ along the Z-dimension as well as $[\mathtt{R}_0|\mathbf{t}_0] = [\mathtt{I}|\mathbf{0}]$, which allows us to express X in terms of x:

$$\begin{aligned}
\mathbf{x} &= \mathtt{K}_0 \cdot [\mathtt{I}|\mathbf{0}] \cdot \mathbf{X} \\
\begin{bmatrix} x \\ y \\ w \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}
\end{aligned} \tag{5.5}$$

Without loss of generality, we further assume the model plane is on $Z = 0$ in the world coordinate system. We denote the i^{th} column of $\mathtt{R}_1$ by $\mathbf{r}_i$. We find:

$$\begin{aligned}
\mathbf{x}' &= \mathtt{K}_1 \cdot [\mathtt{R}_1|\mathbf{t}_1] \cdot \mathbf{X} \\
&= \mathtt{K}_1 \cdot \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & | & \mathbf{t}_1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \\
&= \mathtt{K}_1 \cdot \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & | & \mathbf{t}_1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\
&\overset{(5.5)}{=} \mathtt{K}_1 \cdot \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & | & \mathbf{t}_1 \end{bmatrix} \cdot \mathbf{x} \\
&\overset{(5.4)}{=} \lambda \cdot \mathtt{H} \cdot \mathbf{x}
\end{aligned} \tag{5.6}$$

We denote $\mathtt{H}$ by $\begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$, then we have:

$$\lambda \cdot \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \mathtt{K}_1 \cdot \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & | & \mathbf{t}_1 \end{bmatrix} \tag{5.7}$$

Since the camera matrix $\mathtt{K}_1$ is known from an offline calibration step, the final pose $\mathtt{P} = [\mathtt{R}|\mathbf{t}] = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & | & \mathbf{t}_1 \end{bmatrix}$ can readily be computed:

$$\begin{aligned}
\mathbf{r}_1 &= \lambda^{-1} \cdot \mathtt{K}_1^{-1} \cdot \mathbf{h}_1 \\
\mathbf{r}_2 &= \lambda^{-1} \cdot \mathtt{K}_1^{-1} \cdot \mathbf{h}_2 \\
\mathbf{r}_3 &= r_1 \times r_2 \\
\mathbf{t} &= \lambda^{-1} \cdot \mathtt{K}_1^{-1} \cdot \mathbf{h}_3
\end{aligned} \tag{5.8}$$

The computation for $\mathbf{r}_3$ is motivated by [Zha00] and [LF05] as follows: As $\mathtt{R}$ is a rotation matrix, its columns must be orthonormal. Hence, we can compute r_3 as the cross product of $\mathbf{r}_1$ and $\mathbf{r}_2$. Finally, λ is chosen to normalize the Euclidian length of the columns $\mathbf{r}_i$ to 1.

5.2.1 Dealing With Noise

Of course, because of noise, the estimated homography $\mathtt{H}$ is not an exact solution. Hence, the pose estimation is also affected by noise. The direct effect is that the estimated rotation matrix $\mathtt{R}$ is not orthonormal and thus it does not have the properties of a true rotation matrix.

In a first step, we consider the normality of the columns $\mathbf{r}_i$. Ideally, each column $\mathbf{r}_i$ should validate $\|r_i\| = 1$. A naive method would be to normalize each column individually. This is a non-optimal as it results in shearing effects and corrupts the scaling. A better approach is given by equation (5.8). As a homography is only defined up to a scale, we can make use of the scaling factor λ for our task. In particular, λ should be the same for $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{t}$ to avoid the aforementioned effects. Ideally, one would choose $\lambda = \|\mathtt{K}_1^{-1} \cdot \mathbf{h}_1\| = \|\mathtt{K}_1^{-1} \cdot \mathbf{h}_2\|$. As this equality generally does not hold, we compute λ as the average of both norms:

$$\lambda = \frac{\|\mathtt{K}_1^{-1} \cdot \mathbf{h}_1\| + \|\mathtt{K}_1^{-1} \cdot \mathbf{h}_2\|}{2} \tag{5.9}$$

In a second step, we consider the orthogonality of the columns $\mathbf{r}_i$. [Zha00] proposes a method to find the best rotation matrix $\mathtt{R}_c$ so that $\mathtt{R}_c \cdot \mathtt{R}_c^{-1} = \mathtt{I}$. "Best" is interpreted as the smallest Frobenious norm of the difference $\mathtt{R}_c - \mathtt{R}$. The method works as follows. First, we compute the SVD decomposition of $\mathtt{R} = \mathtt{U} \cdot \mathtt{S} \cdot \mathtt{V}^\top$. Then $\mathtt{R}_c$ is given by $\mathtt{R}_c = \mathtt{U} \cdot \mathtt{V}^\top$. The resulting pose $\mathtt{P}_c$ differs from the original pose $\mathtt{P}$ as visualized in Figure 5.2. A qualitative evaluation is given in Chapter 6.

5.2.2 Orientation Preserving

We now consider the question of orientation preservation. The target is oriented towards the camera in both the model-image as well as the scene-image. The camera is always on the same side of the planar target. This constellation should be preserved during pose estimation. This is guaranteed if the homography determinant is *positive* [HZ00] (pp. 522-523). Hence, before decomposing the homograph matrix into rotation and translation, we have to make sure the determinant of the homography $\mathtt{H}$ is positive. Specifically, we normalize the homography so that its determinant equals 1. The normalized homography $\mathtt{H}_n$ is obtained by

$$\mathtt{H}_n = \sqrt[3]{det(\mathtt{H})}^{-1} \cdot \mathtt{H} \tag{5.10}$$

As a homography $\mathtt{H}$ is never singular, we always have $det(\mathtt{H}) \neq 0$. It is also easily verified that the sign of the determinant of a 3×3 matrix does not change if the matrix is multiplied by a positive scalar. Thus we do not need to worry about the normalization step in equation (5.9). The effect of orientation preservation is visualized in Figure 5.3.

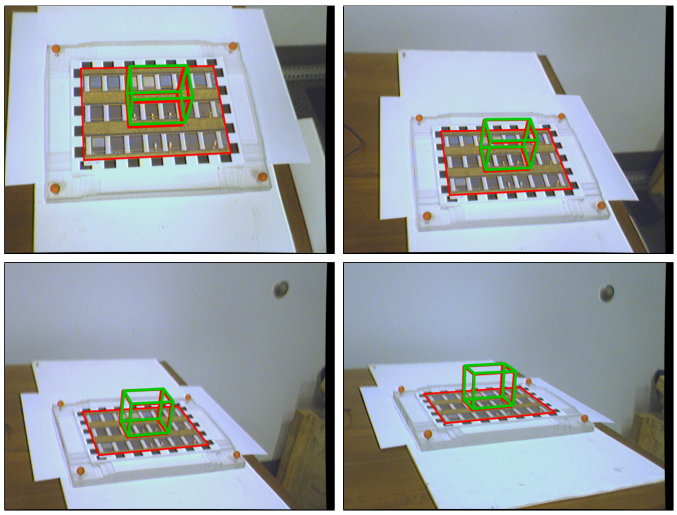


Figure 5.2: Illustrations of rotation correction as described in Section 5.2.2. For a better visualization, only the wireframe of the cube is drawn. The red cube corresponds to the original rotation matrix $\mathtt{R}$, the green one to the corrected rotation matrix $\mathtt{R}_c$.

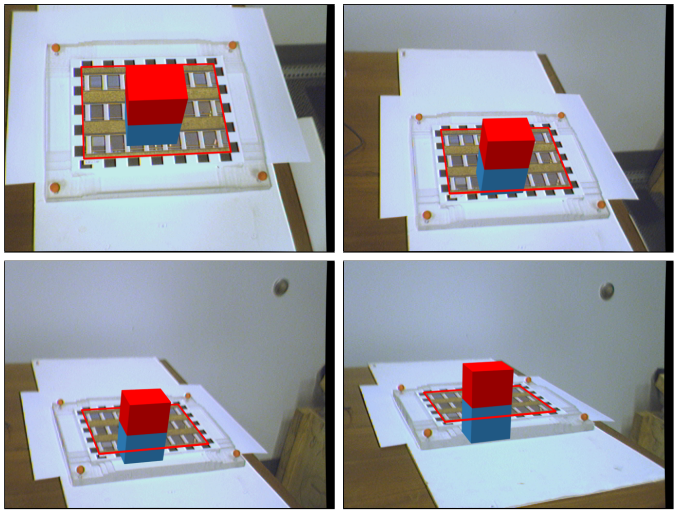


Figure 5.3: Illustrations of the same homography with opposite sign of its determinant. The red cube corresponds to a positive determinant, it is oriented towards the camera. The blue cube corresponds to a negative determinant, it is oriented away from the camera.

Chapter 6

Evaluation

We have used publicly available datasets to evaluate the performance of our AR system. This chapter starts by introducing these datasets and explains the used evaluation methodology. The remaining of the chapter consists of several evaluations which motivate various design choices and compare different input-parameter values.

6.1 Datasets

To create comparable results for different tracking approaches and parameter values, we rely on two annotated datasets. For each frame, the groundtruth information, i.e. the correct homography transformation of the target, is available. The first is the *Visual Tracking Dataset* from the Four Eyes Lab of U.C. Santa Barbara [GHT11]. The second is provided by *Metaio*, an AR company [LBMN09]. We will refer to the sets as the UCSB dataset and the METAIO dataset. Both datasets consist of a number of targets (see Figures 6.1 and 6.2) and video sequences which show these targets under various dynamic behaviors and appearances (see Tables 6.2 and 6.1). The video-sequences contain images from real cameras, not synthetically generated content, which makes the datasets similar to real world applications.

Although conceptually the two datasets are very similar regarding the content of the sequences, there are a few practical differences to consider: The video frames from the METAIO dataset are free of camera distortion, unlike the frames from the UCSB dataset, which first have to be undistorted. The calibration parameters required for undistorting are provided with the dataset. In the METAIO dataset, the background (i.e. everything that is not target) is replaced with solid white color to remove the visual markers that were necessary to extract the groundtruth. In the UCSB dataset, the background and a chessboard pattern, used for groundtruth extraction, are still visible. The UCSB website suggests to

Figure 6.1: The UCSB dataset targets [GHT11]. We will refer to them as target 1 to 6 in the same order as they appear here, from left to right.

ID	Title	Content
1	Dynamic Lighting	Continuous illumination changes.
2	Static Lighting	Discrete illumination changes.
3	Perspective Distortion	Target rotates out of plane.
4	Panning	Target moves sideways with motion blur.
5	Rotation	Camera rotates around its optical axes.
6	Free Movement	Unconstrained, free movement of hand-held camera.
7	Zoom	Camera moves perpendicularly away from target.

Table 6.1: Video sequences from UCSB dataset.

remove the background as a preprocessing step before applying a tracking algorithm. We used the provided groundtruth homography to extract only feature-points that lie on the surface of the target. On one side, this is clearly a simplification for a tracking algorithm as feature-points from a cluttered background create outliers. On the other side, a frame-to-frame tracker would immensely benefit from the easily detectable markers. Finally, the groundtruth for the UCSB dataset is publicly available for download which makes it convenient for rapid evaluations while the METAIO groundtruth is not publicly available. However, METAIO offers to evaluate obtained tracking results and to send back the corresponding evaluation results.

6.2 Methodology

For the evaluation of our tracking system, we are interested in the ratio of successfully tracked frames per sequence and in the accuracy of the estimated homographies per frame. As accuracy measure, we compute the RMS distance between the estimated target-corners $\mathbf{x}'_i$ and the groundtruth target-corners $\bar{\mathbf{x}}_i$ obtained through the datasets. The estimated target corners $\mathbf{x}'_i$ are computed by mapping the corners of the model-image into the camera-image using the estimated homography. The RMS distance is then computed by:

$$RMS = \sqrt{\frac{1}{4}\sum_i \|\bar{\mathbf{x}}_i - \mathbf{x}'_i\|^2} \tag{6.1}$$

Figure 6.2: The METAIO dataset targets [LBMN09].

ID	Title	Content
1	Angle	Perspective distortions.
2	Range	Different range of scalings.
3	Fast Far	Fast movement far from camera.
4	Fast Close	Fast movement close to camera.
5	Illumination	Illumination changes.

Table 6.2: Video sequences from METAIO dataset.

Based on the accuracy, we compute the ratio of successfully tracked frames. Whenever the RMS error is higher than 10 px we assume the tracker lost the target and mark the frame as not successfully tracked. Finally, we analyze the statistical distribution of the error for successfully tracked frames. This approach is consistent with the methodology proposed by [LBMN09] which allows a meaningful comparison of both datasets.

The overall procedure for our evaluation consists in first analyzing the different components of the detection and tracking part separately. We do this to compare different implementation approaches and motivate our final choice. Finally, we also give an extensive evaluation of the complete detection and tracking system. We start by evaluating the components of the detection part.

6.3 RANSAC Sample Selection

In this section, we analyze the performance of RANSAC. Specifically, we compare RANSAC using guided-sample-selection (as described in Section 3.3.2) to naive random-sample-selection as described in Section 2.2. We will refer to them as *guided*-RANSAC and *random*-RANSAC respectively. In particular, we investigate if the guided-sample-selection yields enhancements over random-sample-selection in terms of robustness, amount of inliers, accuracy and speed.

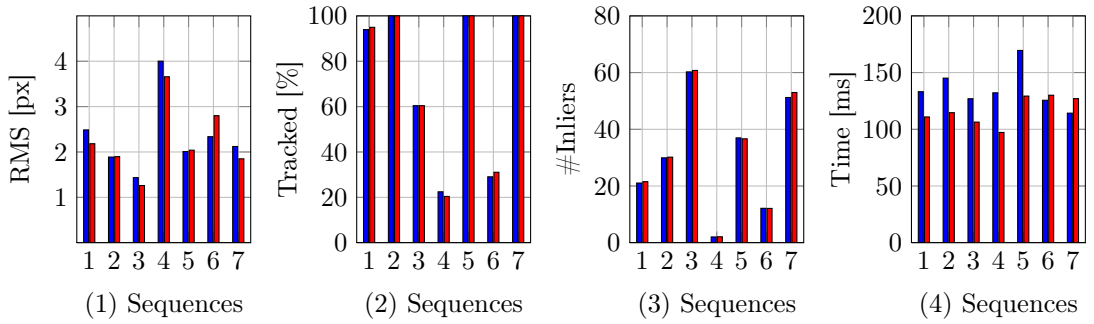


Figure 6.3: Comparing random-RANSAC *(red)* and guided-RANSAC *(blue)* on target 1 for all sequences. *From left to right:* (1) Average RMS error over all correctly tracked frames. (2) Ratio of correctly tracked frames. (3) Average number of inliers per frame. (4) Average time to process a frame.

	RMS	**tracked**	**#inliers**	**time**
random	3.270 px	66.80 %	53.02	333.1 ms
guided	3.271 px	68.54 %	53.61	182.5 ms

Table 6.3: Comparing random-RANSAC and guided-RANSAC. Average values over all targets and all frames of all sequences.

6.3.1 Setup

The evaluation is performed on the UCSB dataset. The database contains the descriptors from the six dataset targets. Per frame, 800 feature-points are extracted. The corresponding descriptors are matched independently against the database using nearest-neighbor-search with Hamming distance and a distance threshold of 55bit (ORB descriptor length is 256bits). The resulting matches are filtered using ratio-test with factor 0.8. The remaining correspondences serve as input to RANSAC. A correspondence is classified as outlier if the symmetric transfer error (eq. 3.2) exceeds 6 px. Based on the inliers provided by RANSAC, the homography is computed using normalized DLT with LM optimization. For both approaches, we compare the average RMS error over all correctly tracked frames, the ratio of correctly tracked frames, the average number of inliers per frame and the average computation time per frame.

6.3.2 Results and Discussion

The average results over all the sequences and all the targets are given in Table 6.3. Figure 6.3 exemplifies these results for target 1. All the qualitative results are to be found in Appendix A, Figure A.2. The overall performance of guided-RANSAC and random-RANSAC are very similar; except for the runtime-performance: guided-RANSAC is almost twice as fast as random-RANSAC. The

better runtime-performance of guided-RANSAC was not expected. We explain this by the careful choosing of matches during the RANSAC iterations which results in a faster convergence of the RANSAC algorithm. Additionally, the convexity-constraint in guided-RANSAC guarantees that only non-degenerated homographies are returned. As a consequence, we will keep the constrained RANSAC as part of our final system.

6.4 Normalized DLT and LM

In this section, we compare the accuracy of the DLT algorithm and its normalized version, the nDLT algorithm. Both algorithms will be applied to the same set of inlier correspondences. Additionally, we inspect the gain in accuracy acquired through optimizing the solutions in a non-linear fashion using LM as described in Section 3.4.2.

6.4.1 Setup

The setup for the experiment is similar to the one used for RANSAC evaluation. However, this time the experiments are performed with various numbers of extracted feature-points and we only rely on guided-RANSAC to eliminate outliers. Based on the inliers, we estimate a homography using DLT, the normalized DLT (nDLT), the DLT with LM optimization (DLT+LM) and the nDLT with LM optimization (nDLT+LM). The evaluation is performed on all the targets and all the sequences from the UCSB dataset. For each approach, we compare the accuracy of the estimated homography. The average RMS error for each approach is computed on the same set of frames, namely those for which RMS(nDLT) < 10 px.

6.4.2 Results and Discussion

Figure 6.4 visualizes the average RMS error exemplarily for target 5 with 400 extracted feature-points and the average RMS error over all the targets and sequences using different numbers of extracted feature-points. Similar results are obtained for the other targets. Figure A.3 in Appendix A shows the results for all the targets and for different amounts of feature-points. The numerical values of the right figure are indicated in Table 6.4. We can observe a considerable improvement by applying LM to DLT. However, very similar results are achieved if nDLT is directly applied. We could not observe a similar improvement when applying LM to the results of nDLT. Non-surprisingly, the accuracy of the homography estimation increases when using more feature-points. The number of feature-points seems to have no significant influence on the discrepancy between

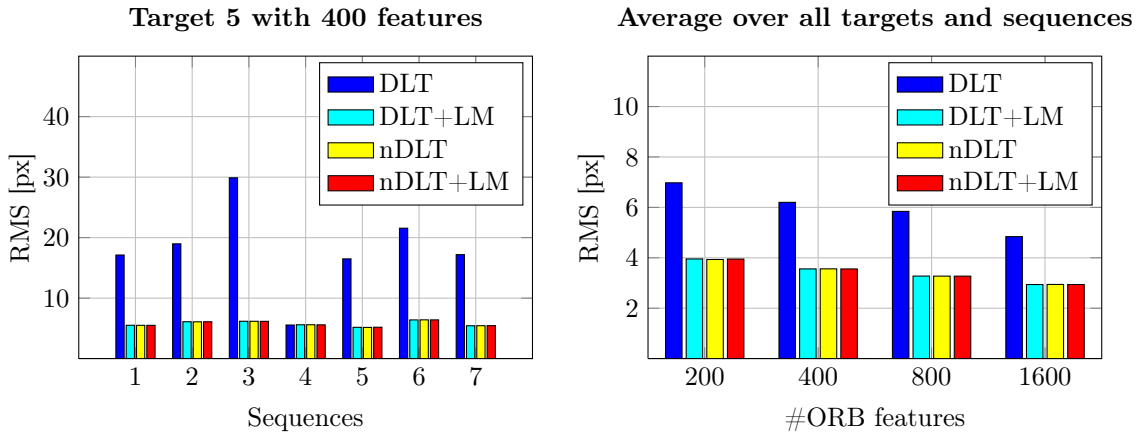


Figure 6.4: Comparing DLT, normalized DLT and influence of optimization with LM. *Left:* average RMS error over all frames for target 5 and 400 feature-points on all sequences. *Right:* average RMS error over all targets and sequences with different number of extracted feature-points.

#features	DLT	DLT+LM	nDLT	nDLT+LM
200	6.975 px	3.954 px	3.963 px	3.950 px
400	6.200 px	3.559 px	3.562 px	3.558 px
800	5.841 px	3.275 px	3.273 px	3.272 px
1600	4.841 px	2.934 px	2.943 px	2.940 px

Table 6.4: Results from homography estimation evaluation: average RMS error for different estimation methods and different amount of feature-points. The same results are visualized in the right of Figure 6.4.

the different approaches. For the final detection pipeline we keep nDLT with LM optimization as it gives the most accurate results.

6.5 Detection Parameter Optimization

Currently our detection system has four parameters: the threshold value t_{HD} for nearest-neighbor Hamming-distance, the factor f_{RT} for the ratio-test, the maximum number of extracted feature-points n and the symmetric transfer error threshold $\tau_{d_\perp}$. In Section 3.2, we empirically found that $t_{HD} = 55$ limits the number of false correspondences and [Low04] proposes the value $f_{RT} = 0.8$. Thus, we are left with two remaining parameters: the number of feature-points n and the symmetric transfer error threshold $\tau_{d_\perp}$.

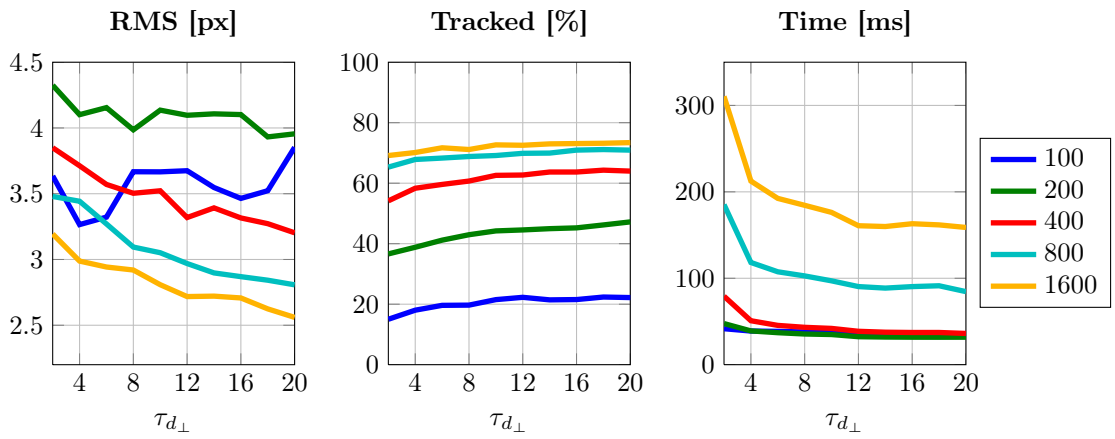


Figure 6.5: Visualizing the influence of threshold $\tau_{d_\perp}$ and number of feature-points n. *Left:* RMS error for correctly tracked frames. *Middle:* number of correctly tracked frames. *Right:* average time in ms per frame. The different colors correspond to different values for n.

		# features n				
		100	**200**	**400**	**800**	**1600**
$\tau_{d_\perp}$	**4**	3.265 px	4.101 px	3.712 px	3.442 px	2.987 px
	8	3.667 px	3.985 px	3.503 px	3.094 px	2.919 px
	12	3.674 px	4.096 px	3.318 px	2.969 px	2.718 px
	16	3.464 px	4.101 px	3.315 px	2.869 px	2.707 px
	18	3.852 px	3.955 px	3.202 px	2.807 px	2.559 px

Table 6.5: RMS for different values of n and $\tau_{d_\perp}$. The values are the average over all targets and sequences from the UCSB dataset.

6.5.1 Setup

The evaluation setup did not change, only nDLT+LM is used to estimate the homography. To analyze the influence of the parameters, the evaluation is performed with $n \in \{100, 200, 400, 800, 1600\}$ and $\tau_{d_\perp} \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$.

6.5.2 Results and Discussion

The average results over all targets and all sequences are visualized in Figure 6.5. The RMS plot for $n = 100$ is not representative as only $\pm 20\%$ of the frames were correctly tracked, thus the untypical behavior in the left plot. The corresponding numerical results are given in Table 6.5 and Table 6.6. Quite surprisingly, the RMS error and the number of correctly tracked frames do not increase with the threshold $\tau_{d_\perp}$, on the contrary, they even decrease. This is maybe due to the

$\tau_{d_\perp}$	# features n = 100	200	400	800	1600
4	18.02 %	38.82 %	58.33 %	67.84 %	70.14 %
8	19.69 %	42.96 %	60.70 %	68.82 %	71.13 %
12	22.27 %	44.55 %	62.70 %	69.87 %	72.53 %
16	21.53 %	45.23 %	63.69 %	70.94 %	73.10 %
18	22.18 %	47.20 %	63.99 %	70.93 %	73.41 %

Table 6.6: Percentage of successfully tracked frames for different values of n and $\tau_{d_\perp}$. The values are the average over all targets and sequences from the UCSB dataset.

sample selection strategy in the RANSAC routine and, perhaps, the additional number of correspondences marked as inliers which outperform the fewer but more accurate correspondences. As expected, the overall accuracy improves with more feature-points. Additionally, with increasing $t_\perp$ the average processing time per frame decreases. This makes sense as more correspondences are marked as inliers during a RANSAC iteration and thus the number of inliers increases faster.

6.6 Overall Detection Performance

We now inspect the overall performance of the detection system. As performance measure, we look at the number of correctly tracked frames and the accuracy of the correctly tracked frames. The evaluation is performed on the UCSB- and the METAIO-dataset. As an added benefit, the detection evaluation generates reference values which are later used to compare it to the results of tracking evaluation.

6.6.1 Setup

Detection makes use of guided RANSAC for outlier removal and normalized DLT with LM optimization for homography estimation. The database contains the descriptors of all the targets of the respective datasets. We set the parameters to $n = 5000$ and $\tau_{d_\perp} = 6$.

6.6.2 Results and Discussion

The ratios of successfully detected frames are shown in Table 6.7 and 6.8. The Figures 6.7 and 6.9 show the distribution of RMS error for each sequence. At a first glance, we observe that the METAIO dataset is generally more difficult then the UCSB dataset in terms of detection rate and accuracy. This is mostly due to more challenging dynamic behavior of the targets. However, we observe

Target	Dyn. Light	Static Light	Persp. Dist.	Panning	Rotation	Free Move	Zoom
1	100 %	100 %	64.6 %	95.9 %	100 %	38.5 %	100 %
2	97.0 %	100 %	69.4 %	87.8 %	100 %	67.5 %	100 %
3	97.0 %	98.7 %	71.4 %	100 %	100 %	57.1 %	100 %
4	100 %	100 %	75.5 %	95.9 %	100 %	75.4 %	100 %
5	25.3 %	41.8 %	48.9 %	12.2 %	51.0 %	12.2 %	59.2 %
6	45.5 %	55.7 %	57.1 %	0.00 %	57.1 %	15.0 %	49.0 %

Table 6.7: Ratio of successfully tracked frames from the UCSB dataset.

Target	Angle	Range	Fast Far	Fast Close	Illumination
1	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %
2	7.00 %	16.9 %	3.20 %	6.20 %	12.6 %
3	12.8 %	35.8 %	5.30 %	29.1 %	46.1 %
4	32.7 %	48.8 %	6.30 %	46.2 %	69.3 %
5	13.1 %	16.9 %	3.10 %	13.5 %	30.7 %
6	23.0 %	35.6 %	7.20 %	24.9 %	56.0 %
7	5.40 %	8.4 0 %	3.30 %	2.50 %	1.90 %
8	35.4 %	31.3 %	5.30 %	15.8 %	54.4 %

Table 6.8: Ratio of successfully tracked frames from the METAIO datatsets.

similar results for comparable sequences: our detection system is rather robust against lighting changes (*Dynamic Lighting, Static Lighting, Illumination*), in-plane rotation (*Rotation*) and target scaling (*Zoom, Range*). This robustness originates from the invariance characteristics of the ORB descriptor. Furthermore, as we employ geometric validation with RANSAC, the system correctly handles repetitive feature-points (METAIO: target 3 and 4, UCSB: target 2). As the system is based on feature-point extraction, it is inherently failure-prone to fast moving target. Fast motion results in a blurred image which makes it very challenging to extract corner-based features (*Fast Far, Fast Close*). The same is true for low-textured targets which naturally yield very little usable features (UCSB Target 5 and 6, METAIO Target 1 and 7). For target 1 in the METAIO dataset, no usable detection results were obtained. There is however potential in improving the performance for targets facing perspective distortion. (*Perspective Distortion, Free Move, Angle, Range*)

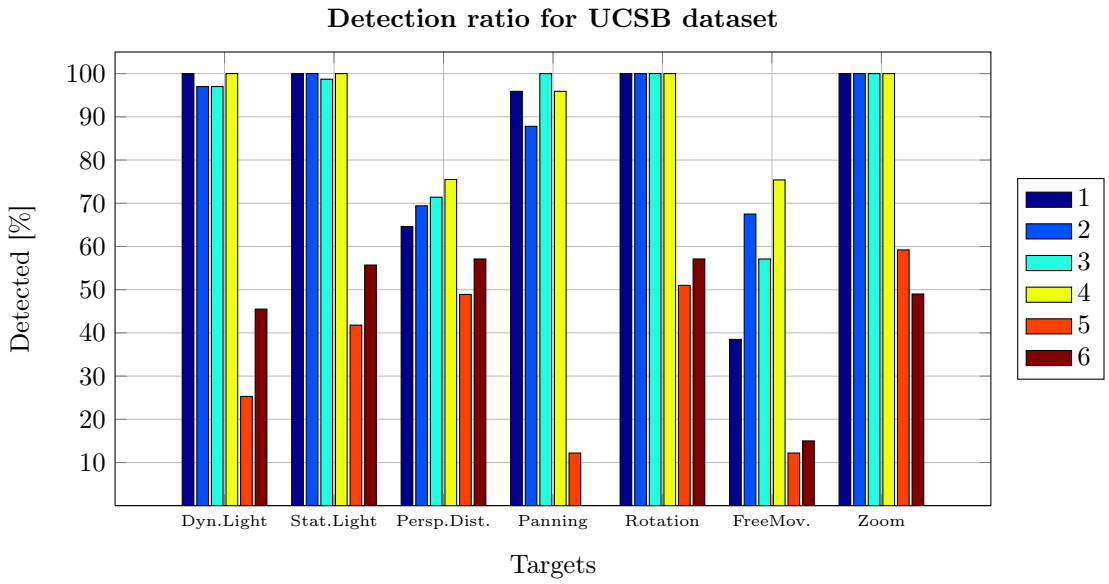


Figure 6.6: Ratio of successfully tracked frames for the UCSB dataset, grouped by sequences. The colors correspond to the different dataset targets.

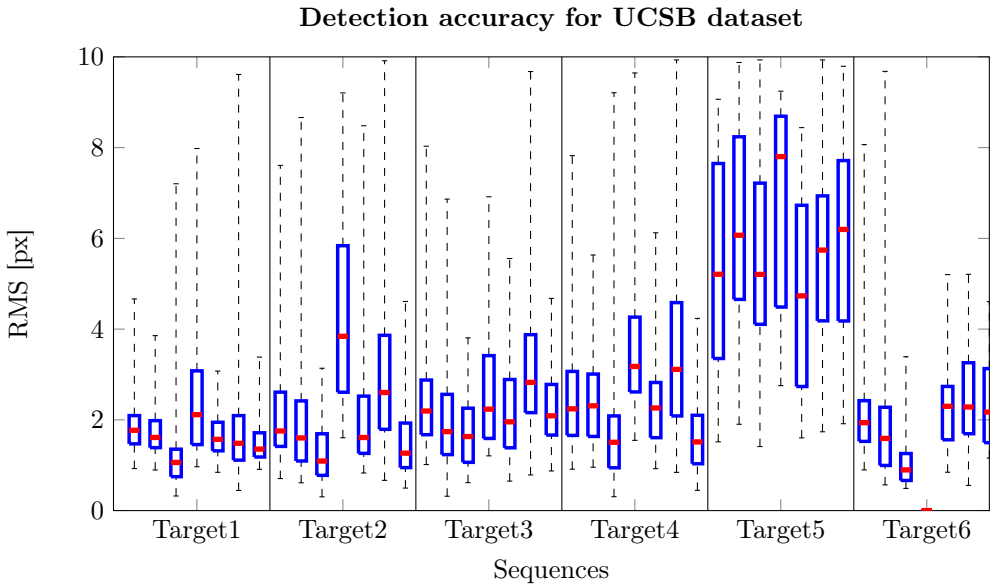


Figure 6.7: Distribution of RMS error for each sequence of the UCSB dataset, grouped by targets. Only successfully tracked frames are taken into account. The whiskers denote minimum and maximum, the box spans from first to third quartile, the red segment shows the mean RMS error.

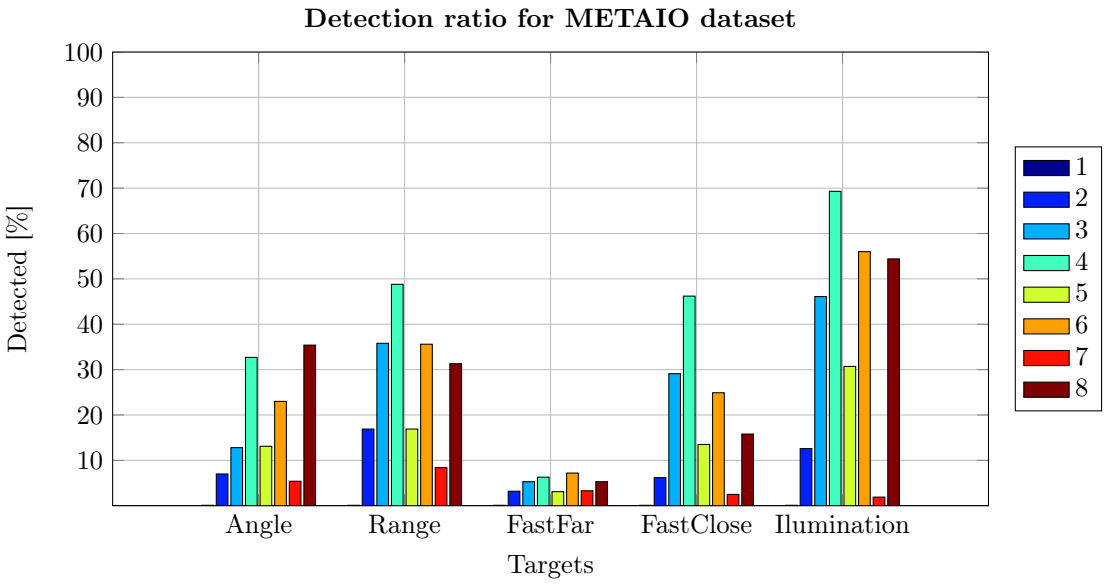


Figure 6.8: Ratio of successfully detected frames for the METAIO dataset, grouped by sequences. The colors correspond to the different dataset targets.

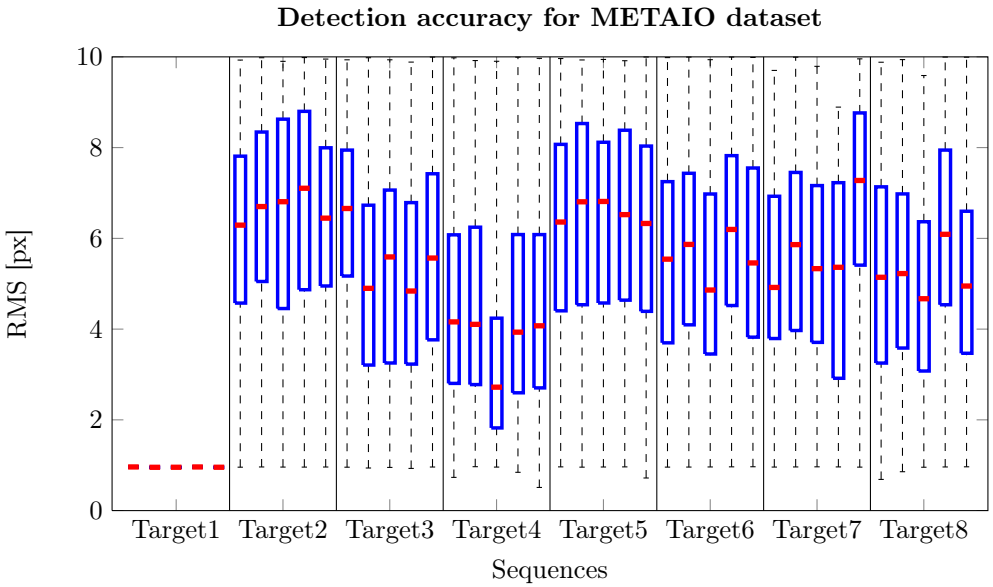


Figure 6.9: Distribution of RMS error for each sequence of the METAIO dataset, grouped by targets. Only successfully tracked frames are taken into account. The whiskers denote minimum and maximum, the box spans from first to third quartile, the red segment shows the mean RMS error.

Figure 6.10: Comparing different warped matching approaches. *Left:* the model-image is warped. *Right:* the camera-image is warped. The left part of each image shows the (warped) model-image, the right part shows the (warped) camera-image. For a less cluttered visualization, only the strongest matches (green) are plotted.

6.7 Warped Matching

In the following sections, we will start the evaluation of the tracking part. In this section, we motivate our warped-target matching approach presented in Section 4.4. The main idea behind warping an image for matching is to become robust against perspective distortion. The goal is to perform matching on a pair of images under the same perspective transformation. For that, we need to apply a perspective transformation on one image from the pair. We can either choose to warp the model-image or the camera-image. In order to decide which one to pick for the final implementation, we count the number of matches returned by each approach. Figure 6.10 shows both matching approaches.

6.7.1 Setup

For the evaluation, we used the third video sequence of the UCSB dataset, which shows the target object at first roughly perpendicular from top, then as the camera goes down, under in increasing perspective distortion. The angle between the normal of the target and the optical axis of the camera varies between 0° and approximately 90°. In each frame, we extracted at most 1200 feature-points originating from the target surface. Both perspective transformations are computed with the same (inverted) homography estimated in the previous frame. The matches are computed using grid matching with a search radius of 5 px. Then we compare the number of found matches for each frame.

6.7.2 Results and Discussion

The number of matches for each frame and both approaches are plotted in Figure 6.11. We exemplarily show the results for target 3 and the average over all targets. The individual results for all the targets are given in appendix A, Figure A.4. On the average, we observe that both approaches give similar results. For small angles, warping the scene-image yields more matches. As the angle increases,

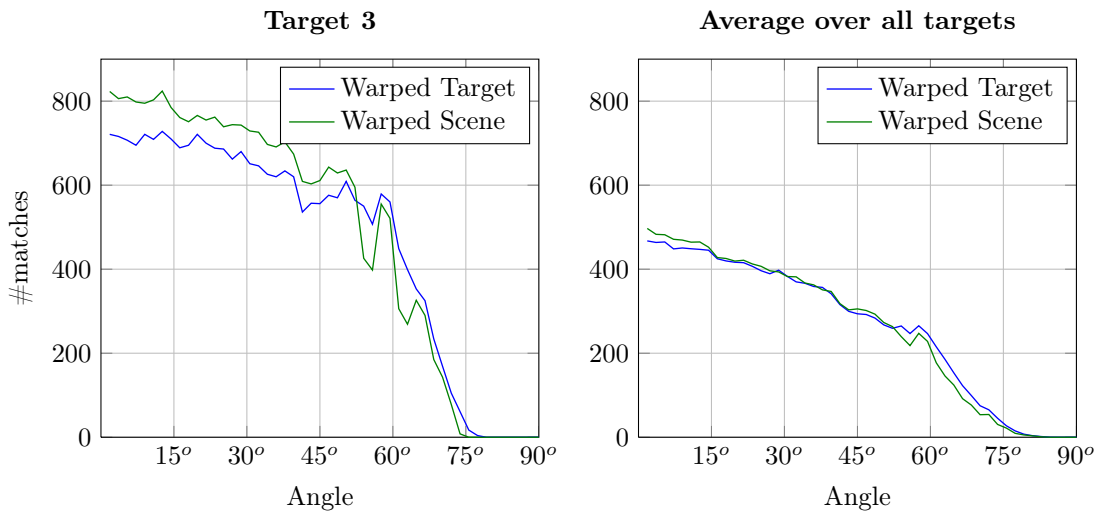


Figure 6.11: Comparing the number of matches for different warped matching approaches: warping the model-image *(blue)* and the camera-image *(green)*. The angle is defined by the target surface normal and the optical axe of the camera. *Left:* number of matches for target 3. *Right:* average number of matches over all targets.

the warped-target approach gives increasingly better results. Above ±50°, the latter slightly outperforms the warped-scene approach. As the goal of the warped matching approach is to increase the number of matches for large angles, and thus become more robust against perspective distortion, we chose the warped-target approach for our final system.

6.8 Tracking Parameter Optimization

In this section we analyze the influence of the tracking parameters. Besides the parameters from the detection part, we have two new additional parameters: in grid matching, each cell has a certain size s, in inter-frame matching we decide based an on uncertainty value τ_{HU} whether an estimated homography is accurate enough for back-projection (see Section 4.3). The goal now is to find a good combination of values for all these parameters in the sense of smallest RMS and highest tracking ratio.

6.8.1 Setup

The evaluation is performed for the parameters and values shown in Table 6.9. As the number of parameters and parameter values is quite numerous, in order to save time, the evaluation is performed only on sequence 6 (*Free Move*) for all the

Parameter	Symbol	Values
Max. number of extracted features	n	400, 800, 1600
Symmetric transfer error threshold	$\tau_{d_\perp}$	2^1,2^2,2^3,2^4,2^5,2^6,2^7,2^8,2^9,2^{10}
Grid cell side length	s	5, 10, 20, 40, 80
Homography uncertainty threshold	τ_{HU}	5, 20, 80, 320

Table 6.9: Tracking parameters and evaluated values.

targets. It is the longest and most challenging sequence. We again considered different values for $\tau_{d_\top}$ to further investigate the results obtained during the detection evaluation in Section 6.5.

6.8.2 Results and Discussion

Figure 6.12 shows the ratio of successfully tracked and their accuracy for s and τ_{HU} for different values of n. The results for different values of $\tau_{d_\perp}$ are shown in Figure 6.13. The best results (in terms of accuracy and ratio of correctly tracked frames) are achieved with cell side length $s = 40$ px. The figures show that larger values for τ_{HU} increase the number of tracked frames, but reduce the overall accuracy. We also notice that for $\tau_{\mathrm{HU}} > 80$ the number of tracked frames barely seems to increase. To choose a value for τ_{HU} requires finding good compromise between these two aspects. We decided on $\tau_{\mathrm{HU}} = 40$. Again, without surprise, an increasing number of feature-points gives increasingly better results. As for the value of $\tau_{d_\perp}$, we observe a local minima around $\tau_{d_\perp} = 16$ in terms of RMS error and detection ratio. After this point, the number of correctly tracked frames stays more or less constant and the RMS slowly increase. This corresponds better to our understanding of the parameter as the results found in Section 6.5.

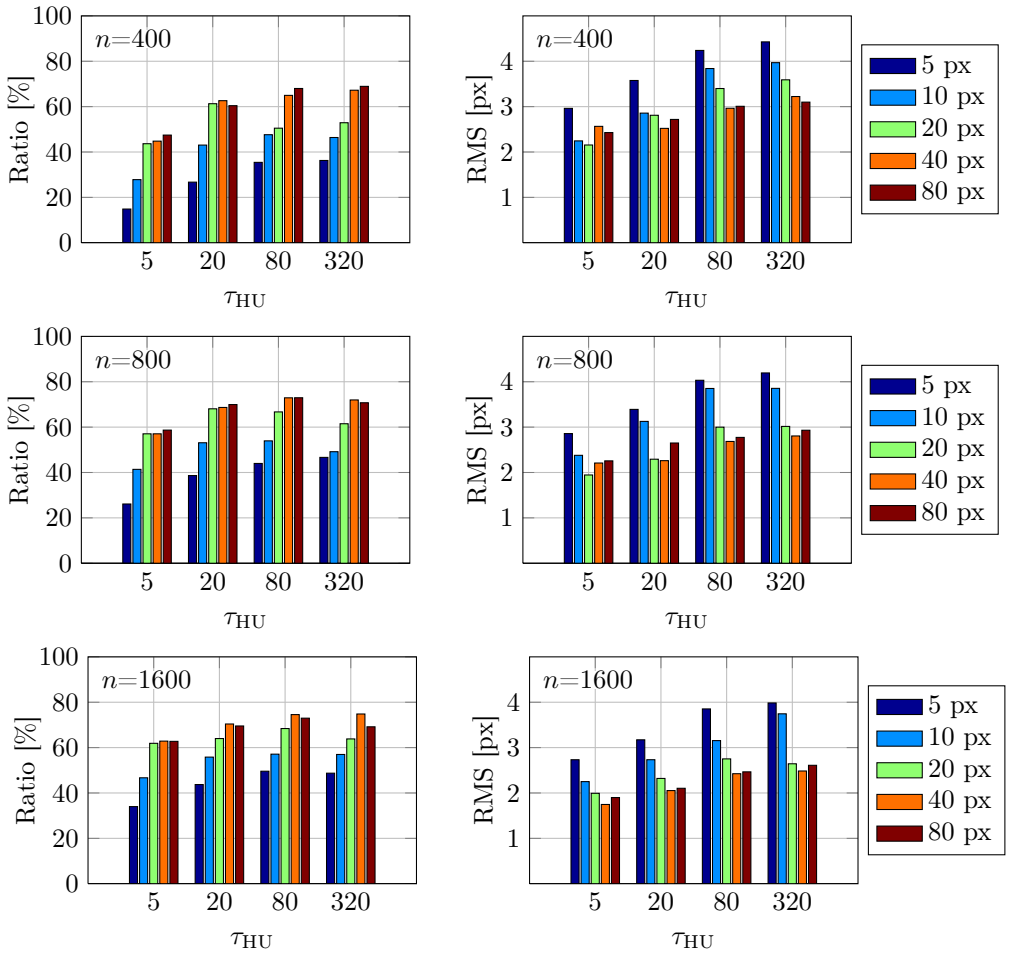


Figure 6.12: Tracking parameter analysis for n, s and τ_{HU} with $\tau_{d_\perp} = 6$. Average values over all targets in sequence 6. The colors correspond to different values of s. *Left column:* ratio of successfully tracked frames. *Right column:* accuracy of the successfully tracked frames.

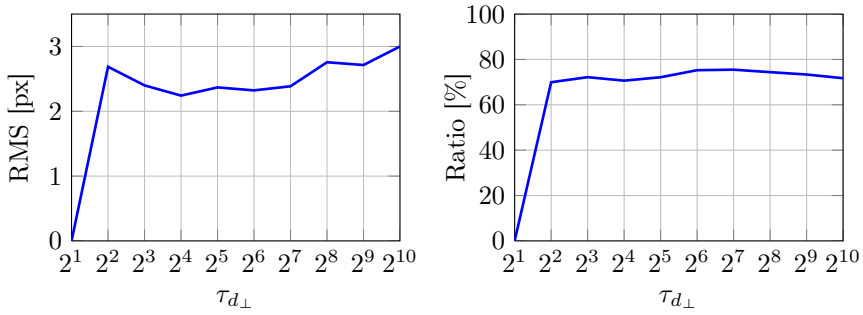


Figure 6.13: Tracking parameter analyses for $\tau_{d_\perp}$ with $n = 800$, $s = 40$ px and $\tau_{\text{HU}} = 40$. Average values over all targets in sequence 6.

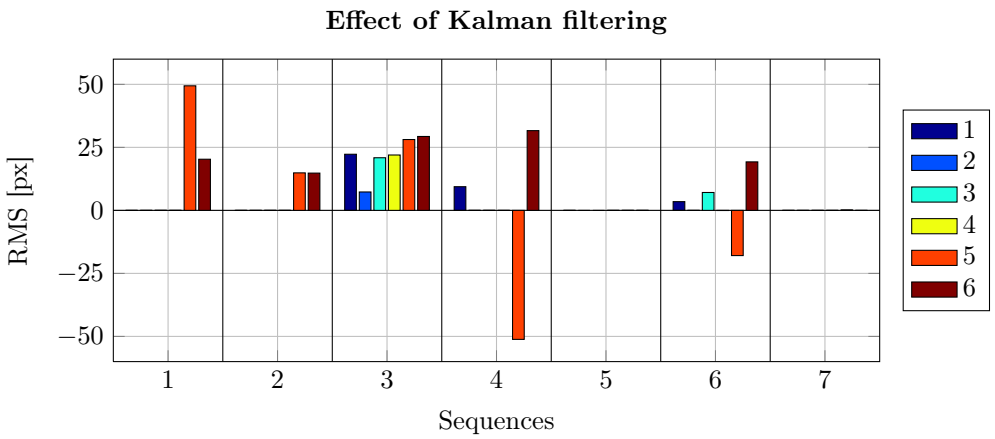


Figure 6.14: Difference between the RMS error of the current measured homography and the corrected homography. The colors indicate the different targets. A bar with positive magnitude signifies that the corrected homography is more accurate than the measured one.

6.9 Kalman Filter

In this section, we analyze the accuracy of the Kalman filtered homography by comparing them to the per frame estimated homography. We show that Kalman filtering improves the overall accuracy of the homography estimation when no measurement is available for several frames and that it helps to smooth fluctuating estimations.

6.9.1 Setup

We compare the difference between the accuracy of the measured homography and the Kalman filtered homography. Again, we consider all the targets and all sequences of the METAIO dataset. In order to get objective results, we consider the RMS of all frames, not just the correctly tracked ones as in the previous evaluations.

6.9.2 Results and Discussion

Figure 6.14 shows the improvements due to Kalman filtering with regard to the average RMS error. Exemplarily, we show in Figure 6.15 the RMS error for each frame of sequence 1 showing target 3. We compare the RMS error of the estimated (measured) homography to the filtered (corrected) homography. The effect of the Kalman filter is twofold: First, it increases the number of successfully tracked frames and secondly it returns more jitter-free homographies. Looking at Figure 6.14, we see that the difference in accuracy is most significant for se-

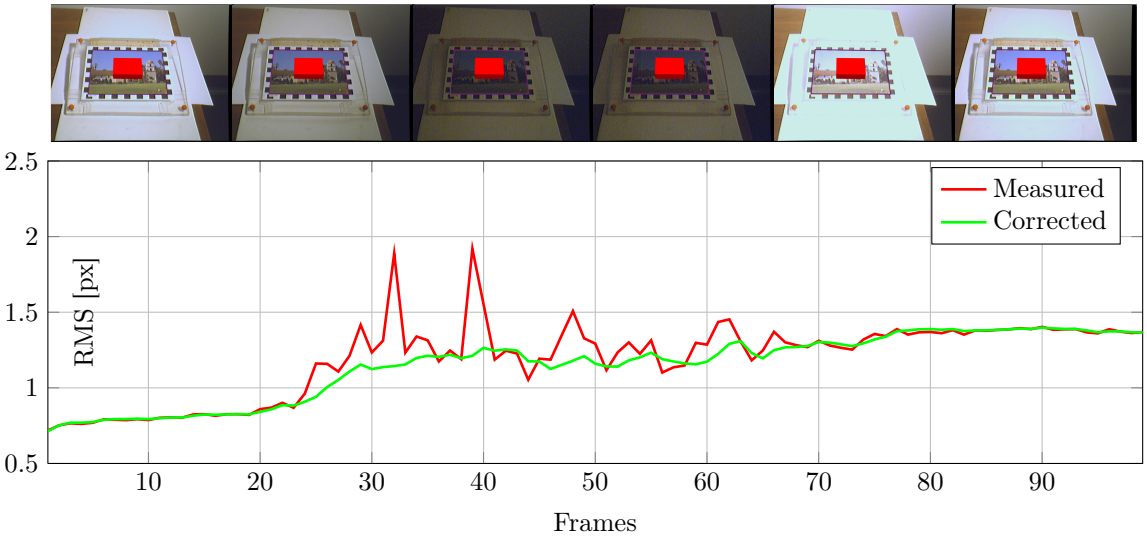


Figure 6.15: Effect of Kalman filtering: As the lighting decreases, the measurements become less accurate and we observe a fluctuating behavior. The Kalman filter successfully smoothes the error. The camera-images above the plot illustrate the content of the respective frames.

quences (3, 6) and targets (4,5) that are more prone to tracking failure (compare Figure 6.16). The last frames of sequence 3 are affected by significant perspective distortion which yields very inaccurate measurements. The motion prediction of the Kalman filter successfully provides homography estimation for these frames.

Figure 6.15 shows the smoothing effect of the Kalman filter. The contrast in the camera-image decreases with the lighting which results in a smaller number of extractable feature-points and thus also less correspondences. Fewer matches results in less reliable homography estimations and also more fluctuation between neighboring frames. Of course, the Kalman filtered solution is not always better as illustrated by target 5. Upon closer analyzation of the sequence, we found that the average measurements were not very reliable and thus the Kalman filter put more trust into the dynamic model which unfortunately was even more wrong than the measurements.

In conclusion, we can say that the Kalman filter improves the overall results. The main benefit of the filter is the over bridging of single frames which did not yield accurate measurements or no measurements at all. However, this is only the case if the overall measurements are already quite reliable.

Target	Dyn. Light	Static Light	Persp. Dist.	Panning	Rotation	Free Move	Zoom
1	100 %	100 %	100 %	100 %	100 %	78.6 %	100 %
2	100 %	100 %	100 %	100 %	100 %	100 %	100 %
3	100 %	100 %	87.8 %	100 %	100 %	89.6 %	100 %
4	100 %	100 %	95.9 %	100 %	100 %	99.2 %	100 %
5	68.7 %	79.7 %	91.1 %	28.6 %	100 %	34.7 %	100 %
6	58.6 %	78.5 %	83.7 %	14.3 %	100 %	32.9 %	100 %

Table 6.10: Ratio of successfully tracked frames for the UCSB dataset.

Target	Angle	Range	Fast Far	Fast Close	Illumination
1	1.10 %	1.80 %	0.70 %	1.80 %	1.60 %
2	68.7 %	63.6 %	12.2 %	16.8 %	50.2 %
3	67.4 %	51.1 %	10.2 %	20.3 %	84.7 %
4	96.8 %	56.2 %	11.9 %	43.8 %	96.6 %
5	73.5 %	53.6 %	6.10 %	35.2 %	77.9 %
6	72.0 %	66.8 %	13.1 %	15.0 %	97.4 %
7	8.50 %	13.2 %	5.10 %	4.20 %	4.00 %
8	98.0 %	65.7 %	9.60 %	17.9 %	81.0 %

Table 6.11: Ratio of successfully tracked frames for the METAIO dataset.

6.10 Overall Tracking Performance

In this section, we evaluate the overall tracking performance of the tracking system and compare it to the results obtained for the detection system in Section 6.6. Again, we measure the performance using the ratio of correctly tracked frames and the accuracy of the correctly tracked frames.

6.10.1 Setup

For evaluation, we used the parameter-values found in the previous sections. In particular, we have $\tau_{d_\perp} = 8$, $s = 40$ px, $\tau_{\mathrm{HU}} = 40$ and $n = 5000$ for maximum performance. We consider all the targets and all the sequences for the UCSB- and the METAIO-dataset.

6.10.2 Results and Discussion

The ratios of successfully tracked frames for the UCSB and METAIO datasets are shown in Table 6.7 and Table 6.8 respectively. For better visual comparison, the ratios are also visualized in Figure 6.16 and Figure 6.18. The accuracy of the correctly tracked frames is expressed using the RMS error and is shown in Figure 6.17 and Figure 6.19.

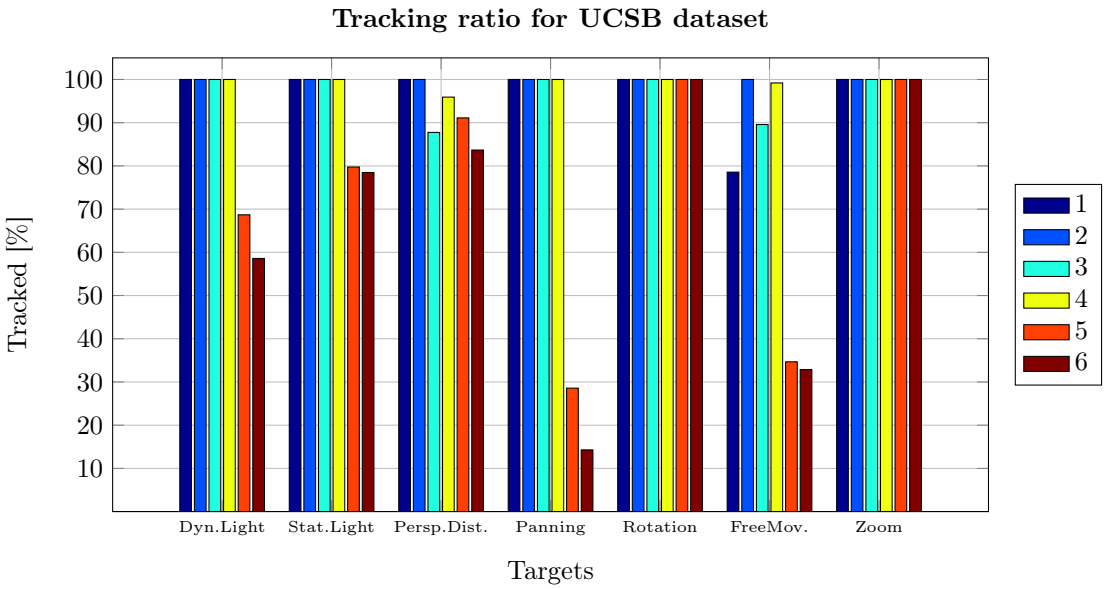


Figure 6.16: Ratio of successfully tracked frames for the UCSB dataset, grouped by sequences. The colors correspond to the different dataset targets.

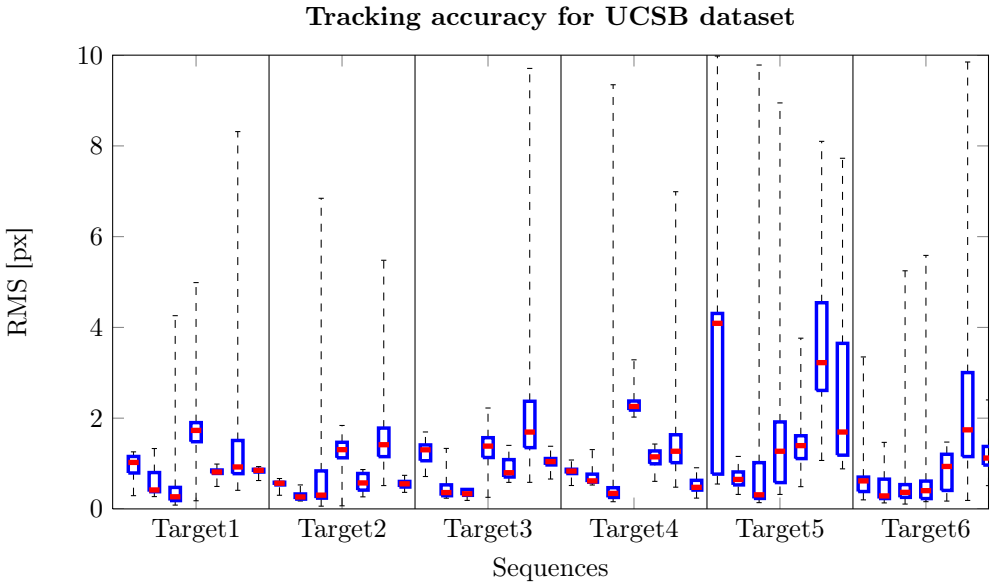


Figure 6.17: Distribution of RMS error for each sequence of the UCSB dataset, only successfully tracked frames are taken into account. The whiskers denote minimum and maximum, the box spans from first to third quartile, the red segment shows the mean RMS error.

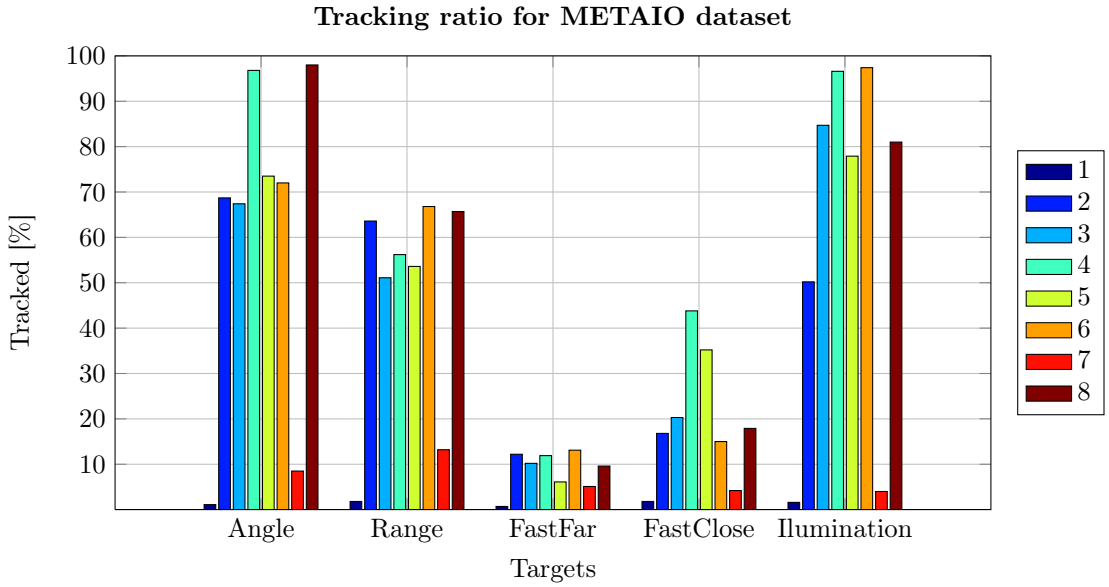


Figure 6.18: Ratio of successfully tracked frames for the METAIO dataset, grouped by sequences. The colors correspond to the different dataset targets.

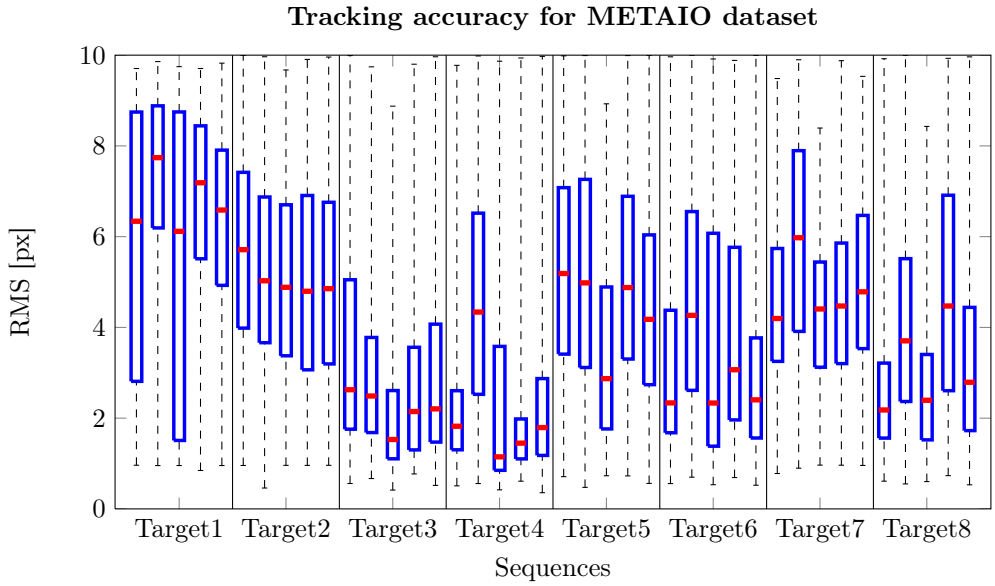


Figure 6.19: Distribution of RMS error for each sequence of the METAIO dataset, only successfully tracked frames are taken into account. The whiskers denote minimum and maximum, the box spans from first to third quartile, the red segment shows the mean RMS error.

At first sight, tracking immensely improves the performance of our AR system. The overall detection ratio was increased while at the same time reducing the average RMS error. Well textured targets (1,2,3,4) are successfully tracked throughout almost all the sequences. Targets with little texture (5,6) are considerably more difficult to track. The most considerable improvements over the detection results are achieved for sequences that show large perspective distortions. This improvement is due to warped-target matching and inter-frame matching.

6.11 Graphical Overlay Performance

We now analyze the accuracy of the 6DoF pose estimation described in Chapter 5. We compare the pose $\mathtt{P}$ and its orthogonalized counterpart $\mathtt{P}_c$.

6.11.1 Setup

To evaluate the accuracy of the pose estimation we measure the RMS error between the target corner-points $\mathbf{x}'_i$ in the scene image and the projected target corner-points $\mathbf{x}_i^{\mathtt{P}} = \mathtt{K} \cdot \mathtt{P} \cdot \mathbf{x}_i$. The experiment is performed for both $\mathtt{P}$ and $\mathtt{P}_c$. The pose estimation is based on an input homography. We perform the experiment using the groundtruth homography $\mathtt{H}_{GT}$ provided by the UCSB dataset and the homography $\mathtt{H}$ estimated by our system. The evaluation is performed on a video sequence of 500 frames.

6.11.2 Results and Discussion

The qualitative results are given in Figure 6.20. The top figure shows the results for pose based on the groundtruth homography. The two bottom plots show the results for the pose based on the estimated homography. The middle plot shows the error $RMS(\mathtt{H}\mathbf{x}_i, \mathbf{x}_i^{\mathtt{P}})$, the bottom plot shows $RMS(\mathbf{x}', \mathbf{x}_i^{\mathtt{P}})$. The additional error introduced by the estimated pose $\mathtt{P}$ is close to zero, while the error introduced by $\mathtt{P}_c$ is unequally larger. On the one hand $\mathtt{P}_c$ fulfills the properties of a rotation matrix but on the other hand $\mathtt{P}$ gives more accurate results which is more important to us. To conclude, we can say that our approach gives very satisfying visual results, under the assumption of a well-estimated input homography. Figure A.7 in appendix A shows visual results for pose estimation based on groundtruth homographies.

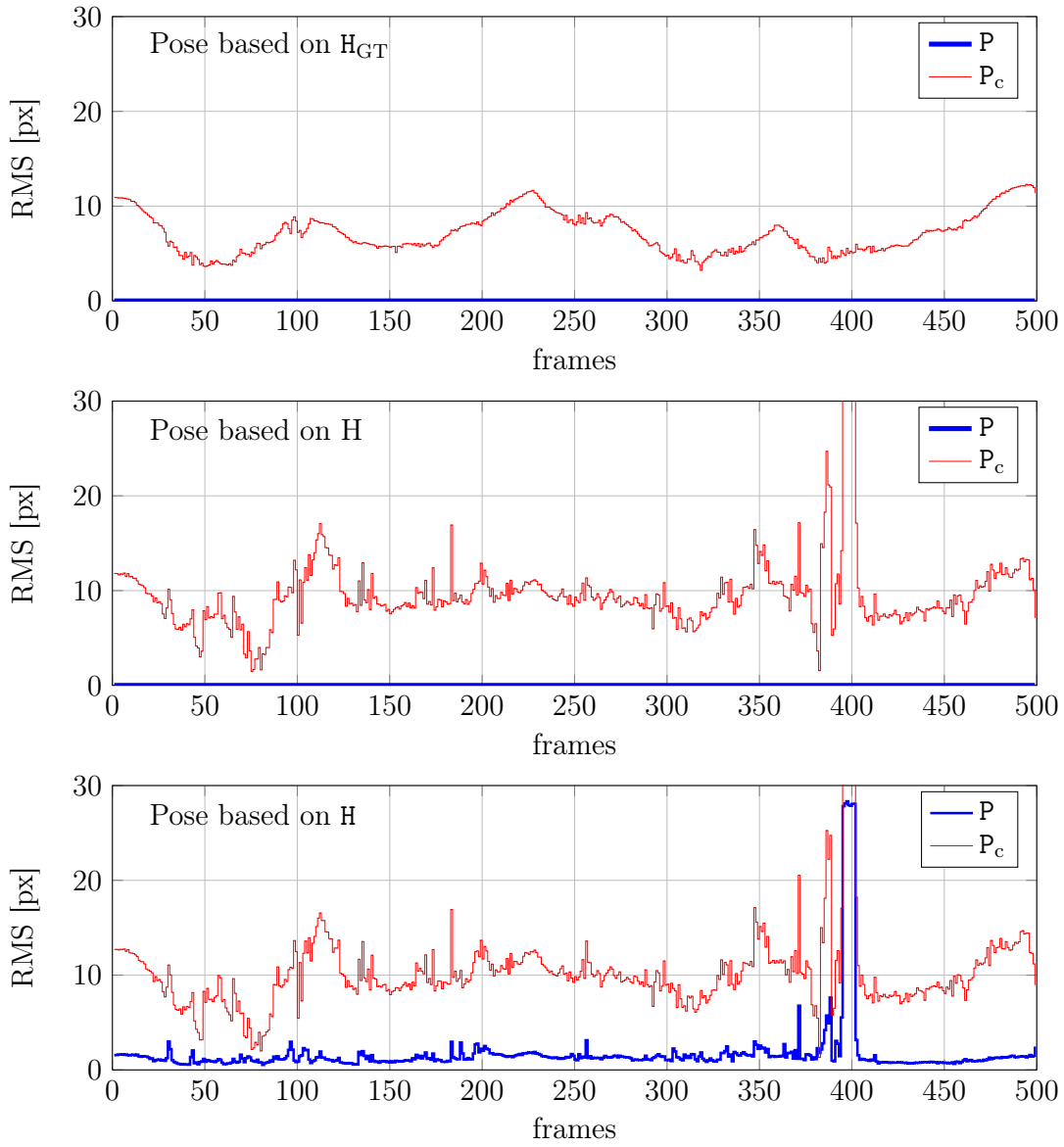


Figure 6.20: The figures show the reprojection error of the target corner-points. *Top:* results for pose based on the groundtruth homography. *Middle:* error $RMS(\mathtt{H}\mathbf{x}_i, \mathbf{x}_i^{\mathtt{P}})$ based on the estimated homography. *Bottom:* error $RMS(\mathbf{x}', \mathbf{x}_i^{\mathtt{P}})$ based on the estimated homography.

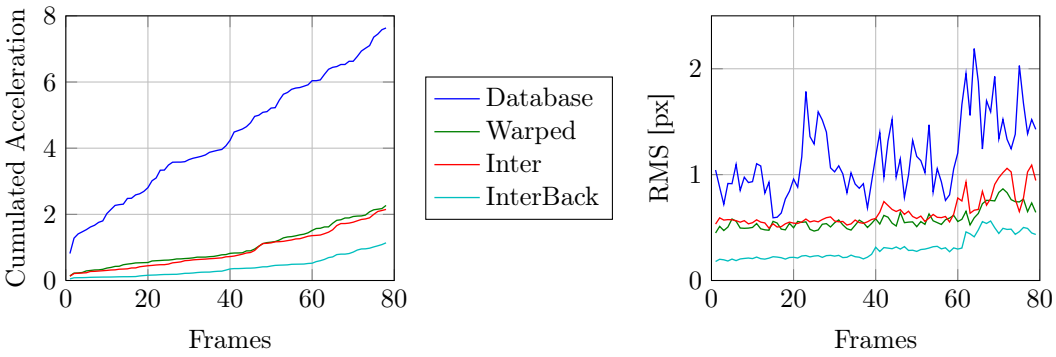


Figure 6.21: Accumulated acceleration and per frame RMS error over sequence 2 for target 2. *Database* is the result for homography estimation based only on database matches. *Warped* additionally uses warped-matching, *Inter* adds the inter-frame matches and *InterBack* also adds back-projected inter-frame matches.

6.12 Visual Jitter and Perspective Distortion

By visual jitter we understand rapid shaking of the rendered objects due to estimation errors. Jitter reduces the quality of an authentic visual impression and should therefore be minimized. The following presents an evaluation methodology for jitter and shows how our hierarchical matching approach reduces jitter while also improving the robustness against perspective distortion.

6.12.1 Setup and Evaluation Methodology

Looking at the RMS error alone is not an option for evaluating jitter as it can be arbitrarily high even if no jitter is observable. Jitter is most notable in static scenes, as it is very easy for the eye to notice a moving object in a static background. In this case, we think the acceleration is a meaningful measure as it directly represents what the human eye perceives as jitter. Together with the RMS we obtain a quite good measure for jitter. For our evaluation, we consider a static scene and a static camera as in sequences 1 or 2 of the UCSB dataset. If the estimation is jitter-free, the mapped target-corners should be static. This means that their acceleration is constant and zero. To evaluate the robustness to perspective distortion, we compare the RMS error against α defined by the angle between the surface normal and the camera's optical axes. Sequence 3 is ideal for this scenario, it starts with $\alpha = 0°$ and ends with $\alpha = 90°$.

6.12.2 Results and Discussion

The curves in Figure 6.21 show the results for four different homographies based on an increasing number of matches: (Database) Only the greedy matches between the camera-image and the model-image are considered. (Warped) Addi-

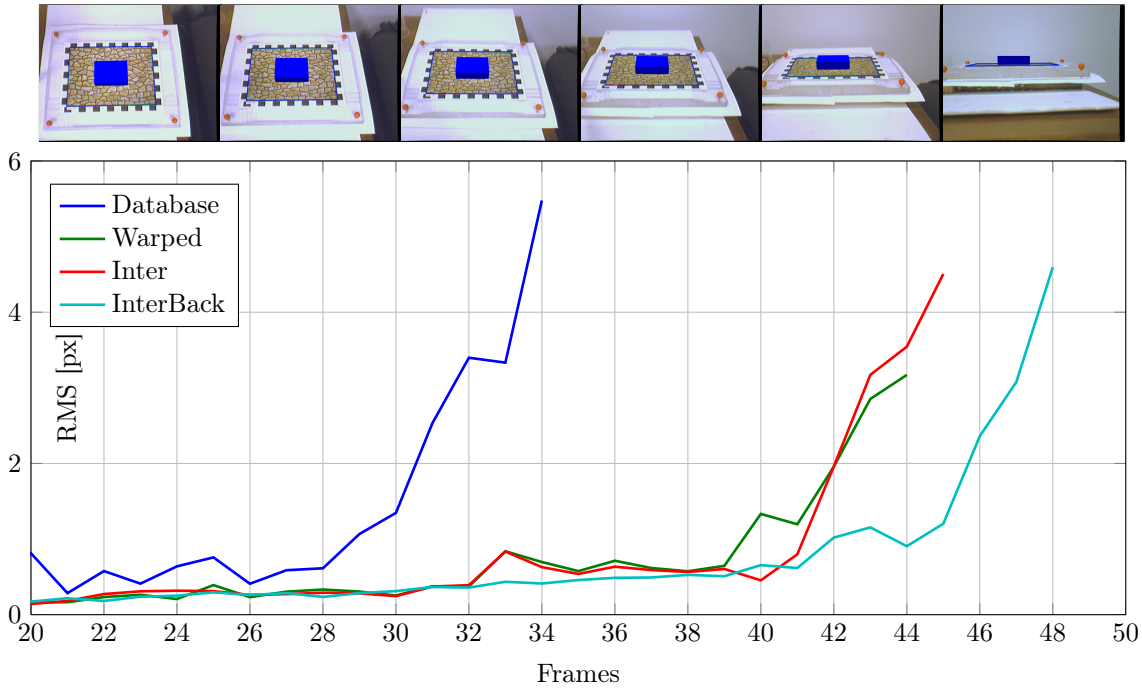


Figure 6.22: Robustness of the framework to increasing perspective distortion. The plot shows per frame RMS error over sequence 3 for target 2. The camera-images above the plot illustrate the content of the respective frames.

tionally to the database matches, also the warped-target matches are used. (Inter) More matches are added from the inter-frame matches (InterBack). Finally, also the back-projected matches from inter-frame matching are added. Figure 6.21 exemplarily shows the results of the jitter evaluation. Figure 6.22 illustrates the results for the perspective distortion evaluation. Both evaluations show clearly that our hierarchical matching approach very successfully avoids jitter and is able to handle significant perspective distortion (Almost 90°). It is also apparent that inter-frame matching without back-projecting unmatched feature merely increases to performance compared to warped-target matching. However, assuming back-projecting can be performed, it gives significantly better results regarding jitter and perspective distortion.

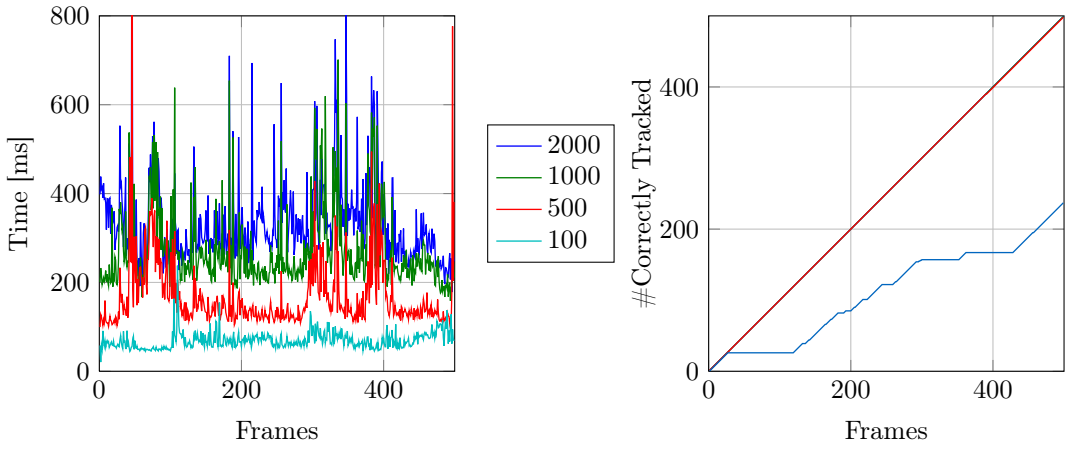


Figure 6.23: The color indicates different values for the number of feature-points n. *Left:* Computation time per frame in milliseconds. *Right:* Accumulated number of correctly tracked frames.

6.13 Runtime Performance

Although optimization for speed was not one of our goals, we still provide the numbers for comparison with possible future optimizations.

6.13.1 Setup

We perform the evaluation on sequence 6 for target 2 of the UCSB dataset. The database contains the 6 targets from the UCSB dataset. The resolution of the video sequence is 640×480 px. Our PC's CPU is an Intel Core i7 running at 3.4Ghz. Although the CPU has multiple cores, our tracker runs only single-threaded. We use different numbers of feature-points n. For each n, we record the frame-rate and the number of correctly tracked frames.

6.13.2 Results and Discussion

Figure 6.23 shows the effect of different numbers of feature-points on the computation time and amount of successfully tracked frames. The left plot shows that the computation-time increases with the number of feature-points. The right plot shows the accumulated number of correctly tracked frames over the complete sequence. For $n = 500$, $n = 1000$ and $n = 2000$ the results are identical and ideal. For $n = 100$, only half of the frames are correctly tracked. Again, we consider a frame to be correctly tracked if the RMS error is smaller then 10 px. Without surprise, the number of feature-points has an influence on the runtime performance and the accuracy of the tracking results. For 100 feature-points we obtain ± 16 fps but the tracking performance is not useable for practical applications. For

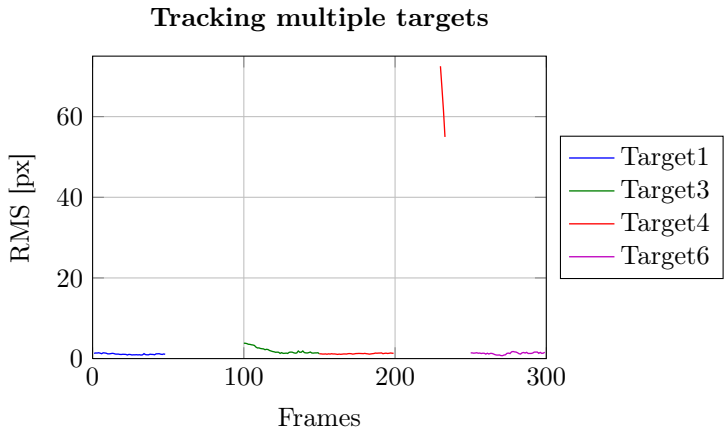


Figure 6.24: Detecting multiple different targets during one sequence. The color indicates the currently detected target. Frames 230-233: Erroneous detection.

500 feature-points, we obtain a frame-rate of ±6 fps, while successfully tracking all frames. Still better (jitter-free) results are obtained for n=2000. However, the frame-rate drops to ±3 fps, which is far from real-time performance.

6.14 Robust Multiple Target Tracking

In this last evaluation, we show that our system can successfully differentiate between multiple targets. The goal is to proof that the system robustly detects targets from the database and returns no detections when no learned target is visible. We also show that our system can handle situations during which the target instantly changes.

6.14.1 Setup

For this evaluation, we traine the system with target 1, 3, 4 and 6 from the UCSB dataset, all the targets from the METAIO dataset and two additional targets. In total, the database contains 14 targets. We create a video sequence showing different targets by concatenating all the *Rotation* sequences of the UCSB dataset. By not training the system with target 2 and 5, we simulate the case when no learned target is visible. We limit the number of feature-points to 2000. This means that the database contains at maximum 28000 descriptors. For each frame and detection, we compute the RMS error.

6.14.2 Results and Discussion

Figure 6.24 shows the RMS error for each frame of the sequence. The different colors correspond to the id's of the detected targets. Between frames 0-50, target

1 was successfully detected. The frames 51-100 contained no target from the database and no detection was observed. Frames 101-150 show target 3, frames 151-200 target 4; note the different behavior of the RMS error before frame 150 and after frame 150 which is where the id of the visible target changed. Between frame 251 and 300, target 6 is detected. One wrong detection was made at frame 230. After a new target was detected, the initial RMS error decreased. This shows the effect of the hierarchical matching used during tracking. In general, we observed that the system copes well with varying targets. Whenever a target from the database became visible, it was correctly tracked. As soon as a different target became visible, it was instantly detected. Upon analyzing on the erroneous detection at frame 230, we found that the fluctuation is due to wrong matches between the camera-image and the previously detected warped model-image. The resulting matches yielded a valid homography, which was then used to initialize the Kalman filter. Since the erroneous detection occurred only in one frame, the track stopped after 3 frames as the Kalman filter received no new measurements.

Chapter 7

Conclusion

In this work, we explored the task of multiple target tracking for marker-less augmented reality applications. We implemented a tracker for multiple planar targets, based on the ORB [RRKB11] feature-point descriptor. The implementation is written in C++ and uses the ORB implementation provided by the OpenCV library [Bra00]. We put the focus on accurate camera-pose estimation rather than real-time capabilities. Although, by limiting the number of extracted feature-points, we could achieve a frame-rate of up to 7fps on a Intel Core i7 CPU @ 3.4GHz, while maintaining acceptable tracking results.

The main components of the framework are a detector, a tracker and a graphical overlay. The detector returns a homography that maps the model-image onto the target in the camera-image. The homography is estimated from a set of feature-point correspondences using the normalized DLT algorithm and Levenberg-Marquardt to refine the solution. We observed that data normalization increases the accuracy of the DLT considerable, up to the point that one might consider skipping the refine-step in favor of speed. The outliers in feature-point correspondences are removed using RANSAC, which we enhanced with a sanity check to make sure only rigid-body transformations are returned. We showed that this approach not only improves the robustness of our system, but also increases the speed. The tracker is based on the Kalman filter [FP02], which applies a consistent dynamic movement on the target. We showed that the system detects slightly more frames with the filter. However, we think that a more elaborated filter, could even further improve the results. We introduced a hierarchical matching scheme and experimentally proved its benefits over database-matching. We extracted additional matches from consecutive frames and perspectively transformed model-images, which yields more accurate and jitter-free homography estimations. The graphical overlay computes the 6DoF pose from the estimated homography which we used to render an artificial object on the tracked target. We presented a direct solution for the pose estimation problem, by decomposing a given homography. In the evaluation part, we analyzed the performance of our

system by looking at the accuracy of the estimated homography and the ratio of correctly tracked frames. For the evaluation we used publicly available datasets, namely UCSB [GHT11] and METAIO [LBMN09]. We evaluated most components of our framework under different lighting conditions and target movements. In particular, we proved that our framework is robust against considerable perspective distortion and showed how the hierarchical matching scheme minimizes jitter and improves accuracy.

7.1 Future Work

We now present possible improvements and additions from which the AR system could benefit. Besides the obvious optimization for speed and real-time capability, we mention a few potential points:

Database Matching. In the current implementation, we use a naive greedy approach for establishing correspondences between the descriptors from the camera-image and the descriptors from the database. This leaves plenty of room for improvement, especially regarding speed. Possible alternatives are tree-based matching, hashing-based algorithms or more advanced approaches like visual vocabularies as described in [GL11].

Dynamic Tracking Model. The current implementation uses the traditional Kalman filter with a constant velocity movement model. Clearly, this model is a severe simplification of the actual, complex dynamic-behavior of a hand-held camera. More elaborated approaches, such as Extended Kalman filter (EKF), which is very popular in robots [TBF+05], or particle filters [FP02] could be interesting alternatives. Additionally, it would definitely make sense to directly apply the filter to the pose of the tracked target.

3D-Targets. Supporting also non-planar targets comes as a natural next step. Relatively quick results could be obtained by assigning a 3D position to the feature-points in the model-image and then compute the pose as described in [HZ00] p.181. This approach is similar to [PLW08].

RANSAC Frame-to-frame feature tracking yields a number of correspondences that are inliers with high probability. This knowledge can be used to improve RANSACs conversion speed.

We finish with a list of a few smaller enhancements: Different parameterization for the homography could be considered, such as [HZ00] (p.111). The estimated 6DoF pose could be refined in an additional step using Levenberg-Marquardt together with M-estimators and weighting factors [LF05]. As we have shown that

the accuracy of the estimated homography increases with the number of correspondences, one could try to iteratively establish new matches for feature-point pairs that were first identified as outliers. A similar approach is presented in [SÖL+10].

Appendix A

Figures

The appendix contains additional evaluation figures and shows visual results of our AR framework.

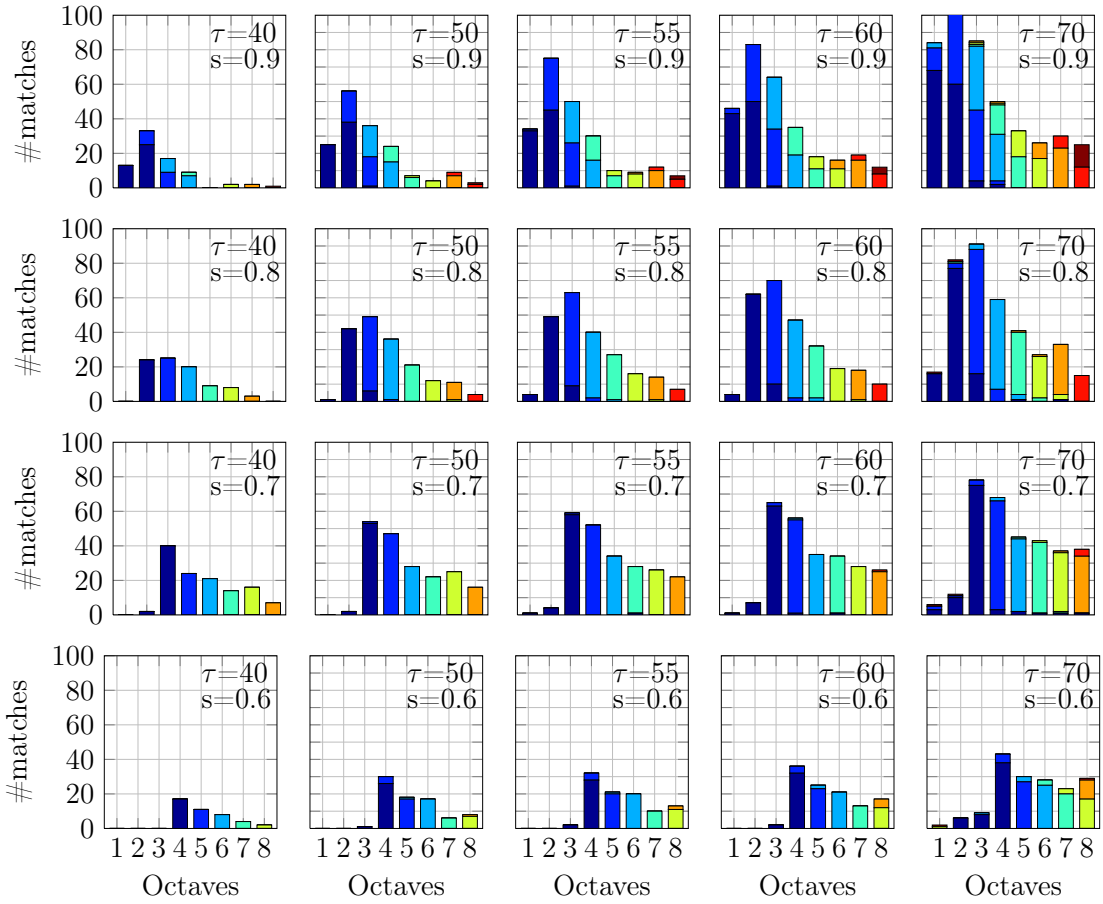


Figure A.1: Figures from setup described in section 3.2 showing various thresholds τ and scalings s.

Figure A.2: Comparing guided (red) and random (blue) RANSAC. The shown values are the average results over all input frames. See section 6.3.

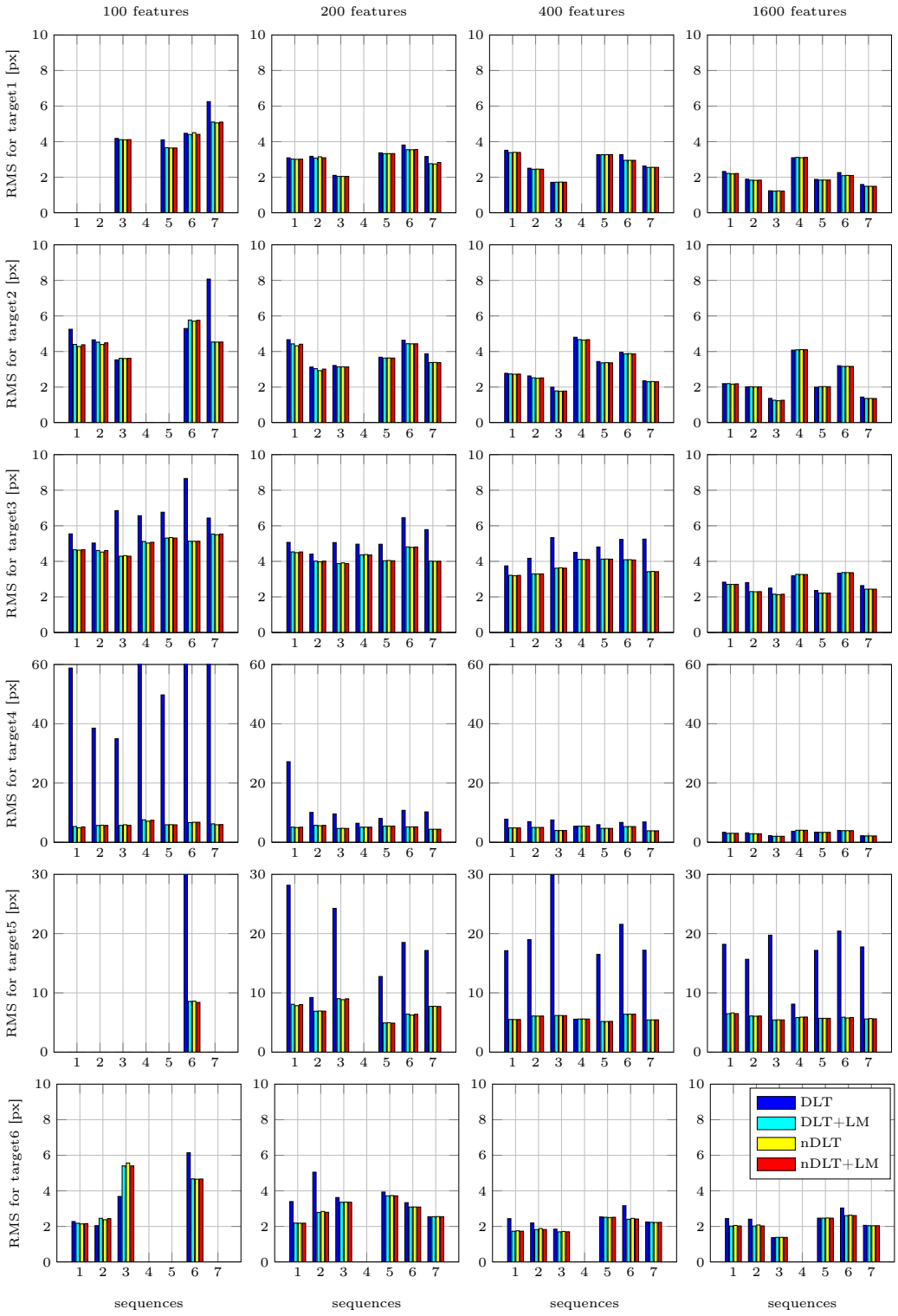


Figure A.3: Comparing DLT, normalized DLT and influence of optimization with LM. See section 6.4.

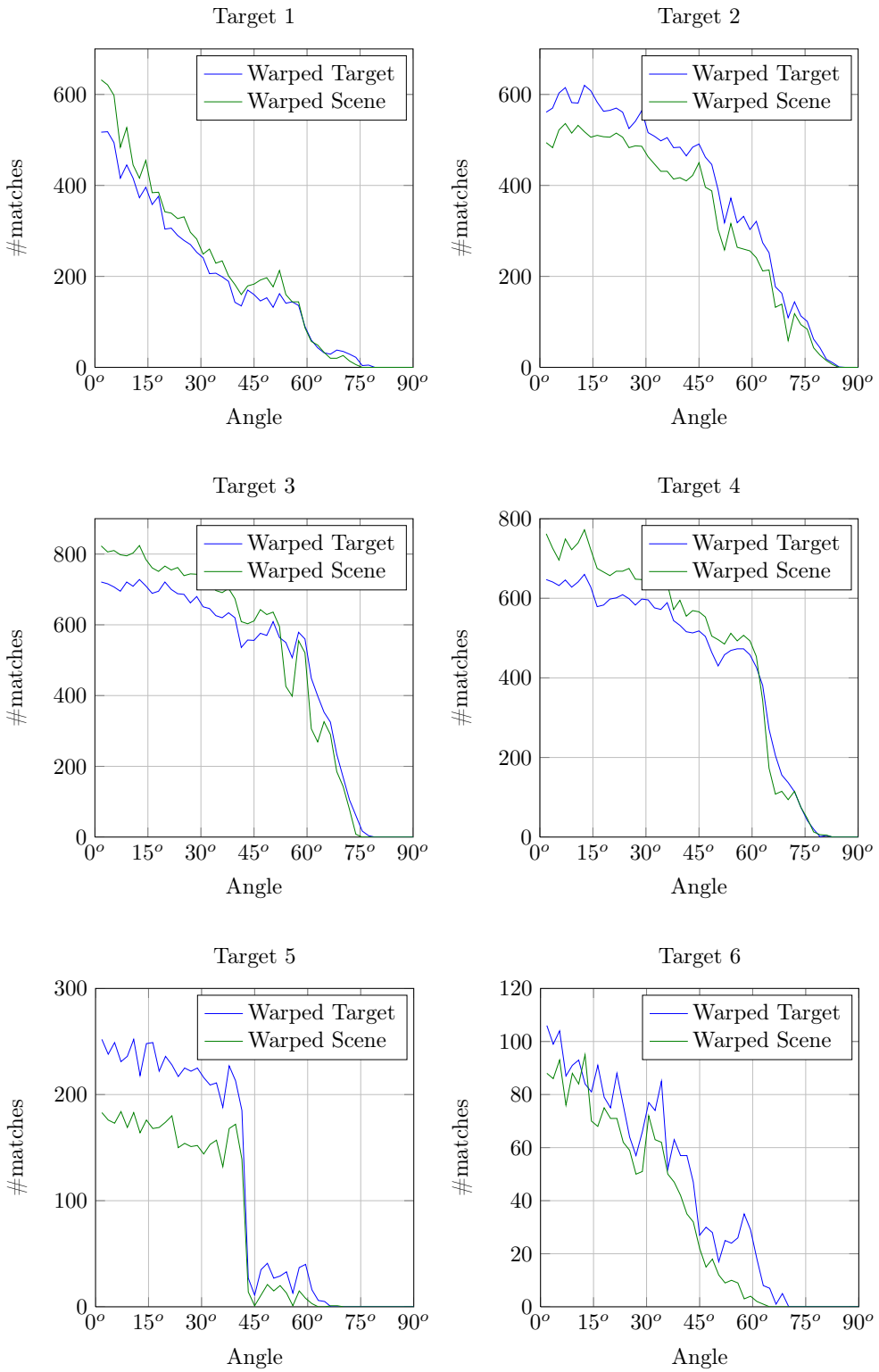


Figure A.4: Comparing different warping approaches for matching as explained in section 6.7.

Figure A.5: Figures from setup described in section 6.11 showing the robustness of the pose estimation algorithm based on groundtruth homography.

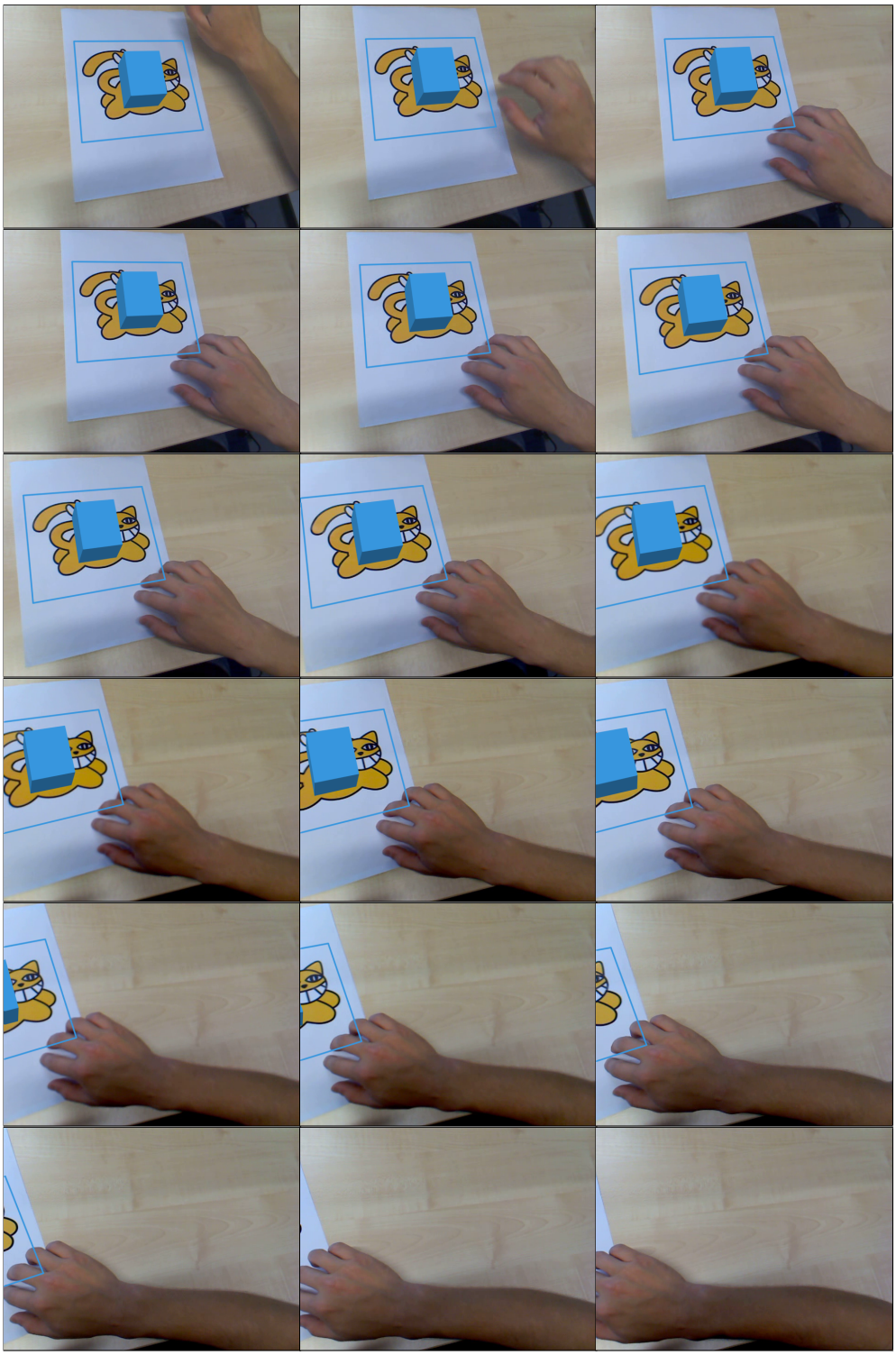


Figure A.6: Illustration of target leaving the field of view of the camera. The target is still detected in the third last shown frame.

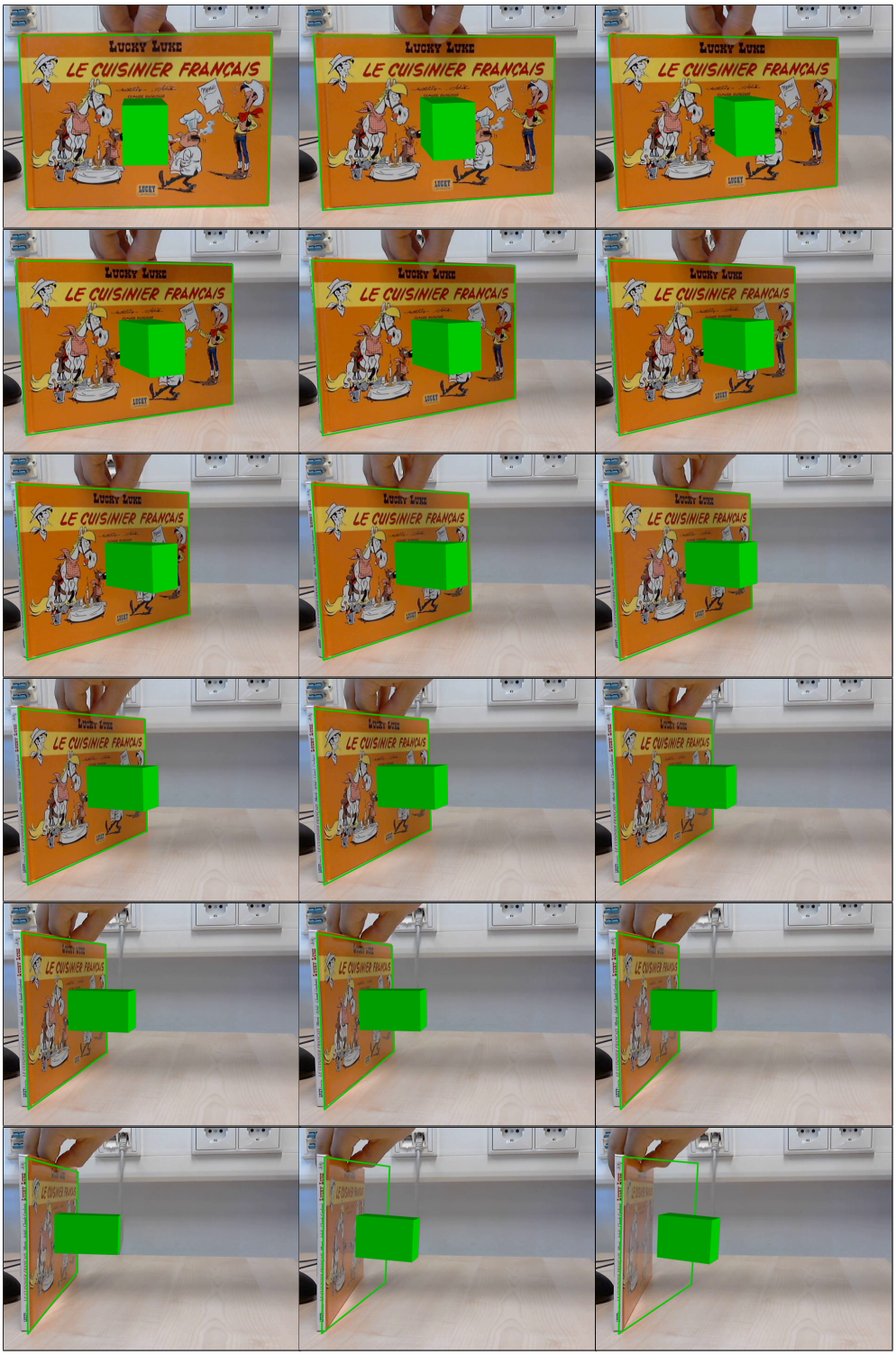


Figure A.7: Illustration of target with increasing perspective distortion. The tracking breaks only close to 90°.

Bibliography

[AWK+09] Clemens Arth, Daniel Wagner, Manfred Klopschitz, Arnold Irschara, and Dieter Schmalstieg. Wide area localization on mobile phones. In *ISMAR*, pages 73–82, 2009.

[BN06] Christopher Bishop and Nasser Nasrabadi. *Pattern recognition and machine learning*. Springer, 2006.

[Bol13] Csaba Bolyós. Scarlet–real time mobile augmented reality library, 2013.

[BR05] Oliver Bimber and Ramesh Raskar. *Spatial augmented reality: merging real and virtual worlds*. A K Peters Limited, 2005.

[Bra00] Gary Bradski. Open computer vision library. *Dr. Dobb's Journal of Software Tools*, 2000.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417. 2006.

[CG67] Harold S.M. Coxeter and Samuel L. Greitzer. *Geometry Revisited*. Mathematical Association of America, 1967.

[CLO+12] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *TPAMI*, 34(7):1281–1298, 2012.

[CM05] Ondrej Chum and Jiri Matas. Matching with prosac-progressive sample consensus. In *CVPR*, pages 220–226, 2005.

[CMFO07] Fernando Caballero, Luis Merino, Joaquin Ferruz, and Anibal Ollero. Homography based kalman filter for mosaic building applications to uav position estimation. In *ICRA*, pages 2004–2009, 2007.

[Dic10] Ernst D. Dickmanns. *Dynamic Vision for Perception and Control of Motion*. Springer, 2010.

[EB08] Michael Eckmann and Terrance E. Boult. Spatio-temporal consistency and distributivity as qualities of features. In *CVPRW*, 2008.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, 1981.

[FL88] Olivier D. Faugeras and Francis Lustman. Motion and structure from motion in a piecewise planar environment. *IJPRAI*, 2(03):485–508, 1988.

[FP02] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[GHT11] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *IJCV*, 94(3):335–360, 2011.

[GJ+10] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[GL11] Kristen Grauman and Bastian Leibe. *Visual object recognition*. Morgan & Claypool Publishers, 2011.

[GZWS09] Lukas Gruber, Stefanie Zollmann, Daniel Wagner, and Dieter Schmalstieg. Evaluating the trackability of natural feature-point sets. In *ISMAR*, pages 189–190, 2009.

[Hor86] Berthold Horn. *Robot Vision*. MIT Press, 1986.

[HZ00] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[JSM10] Farrokh Janabi-Sharifi and Mohammed Marey. A kalman-filter-based method for pose estimation in visual servoing. *T-RO*, 26(5):939–947, 2010.

[KPS+07] Johannes Koehler, Alain Pagani, Didier Stricker, Michael Felsberg, Michael Zöllner, Yulian Pastarmov, Harald Wuest, Mario Becker, Gabriele Bleser, and Pedro Santos. Real-time camera pose estimation using correspondences with high outlier ratios. *JSID*, 15(9):679–689, 2007.

[KSF07] Jens Klappstein, Fridtjof Stein, and Uwe Franke. Applying kalman filtering to road homography estimation. In *ICRA*, 2007.

[LBMN09] Sebastian Lieberknecht, Selim Benhimane, Peter Meier, and Nassir Navab. A dataset and evaluation methodology for template-based tracking algorithms. In *ISMAR*, pages 145–151, 2009.

[LF05] Vincent Lepetit and Pascal Fua. *Monocular model-based 3d tracking of rigid objects: A survey*. Now Publishers Inc, 2005.

[LLF05] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, pages 775–781, 2005.

[LMNF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *IJCV*, 81(2):155–166, 2009.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[LSM+03] Yong Liu, Maux Storring, Thomas Moeslund, Claus Madsen, and Erik Granum. Computer vision based head tracking from reconfigurable 2d markers for ar. In *ISMAR*, pages 264–267, 2003.

[MBSS10] David Monnin, Etienne Bieber, Gwenaél Schmitt, and Armin Schneider. An effective rigidity constraint for improving ransac in homography estimation. In *Advanced Concepts for Intelligent Vision Systems*, pages 203–214. 2010.

[MN95] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-d objects from appearance. *IJCV*, 14(1):5–24, 1995.

[MV+07] Ezio Malis, Manuel Vargas, et al. Deeper understanding of the homography decomposition for vision-based control, 2007.

[NPS10] Tobias Nöll, Alain Pagani, and Didier Stricker. Markerless camera pose estimation-an overview. In *VLUDS*, pages 45–54, 2010.

[OFL07] Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit. Fast keypoint recognition in ten lines of code. In *CVPR*, 2007.

[PLW08] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. In *ISMAR*, pages 117–120, 2008.

[RD05] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *ICCV*, pages 1508–1515, 2005.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV*, pages 430–443. 2006.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *ICCV*, pages 2564–2571, 2011.

[SLK09] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Scramsac: Improving ransac's efficiency with a spatial consistency filter. In *ICCV*, pages 2090–2097, 2009.

[SÖL+10] Eduard Serradell, Mustafa Özuysal, Vincent Lepetit, Pascal Fua, and Francesc Moreno-Noguer. Combining geometric and appearance priors for robust homography estimation. In *ECCV*, pages 58–72. 2010.

[Stu00] Peter Sturm. Algorithms for plane-based pose estimation. In *CVPR*, volume 1, pages 706–711, 2000.

[SWR+09] Gerhard Schall, Daniel Wagner, Gerhard Reitmayr, Elise Taichmann, Manfred Wieser, Dieter Schmalstieg, and Bernhard Hofmann-Wellenhof. Global pose estimation using multi-sensor fusion for outdoor augmented reality. In *ISMAR*, pages 153–162, 2009.

[TBF+05] Sebastian Thrun, Wolfram Burgard, Dieter Fox, et al. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2005.

[TCT+10] Gabriel Takacs, Vijay Chandrasekhar, Sam Tsai, David Chen, Radek Grzeszczuk, and Bernd Girod. Unified real-time tracking and recognition with rotation-invariant fast features. In *CVPR*, pages 934–941, 2010.

[TCT+12] Gabriel Takacs, Vijay Chandrasekhar, Sam Tsai, David Chen, Radek Grzeszczuk, and Bernd Girod. Rotation invariant fast features for large-scale recognition. In *SPIE Optical Engineering+ Applications*, pages 84991D–84991D. International Society for Optics and Photonics, 2012.

[TK91] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ., 1991.

[UM12] Hideaki Uchiyama and Eric Marchand. Object detection and pose tracking for augmented reality: Recent approaches. In *FCV*, 2012.

[WRM+08] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR*, pages 125–134, 2008.

[WRM+10] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *TVCG*, 16(3):355–368, 2010.

[WSB09a] Daniel Wagner, Dieter Schmalstieg, and Horst Bischof. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *ISMAR*, pages 57–64, 2009.

[WSB09b] Daniel Wagner, Dieter Schmalstieg, and Horst Bischof. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *ISMAR*, pages 57–64, 2009.

[YJS06] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *CSUR*, 38(4):13, 2006.

[Yua06] Chunrong Yuan. Markerless pose tracking for augmented reality. In *Advances in Visual Computing*, pages 721–730. 2006.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *TPAMI*, 22(11):1330–1334, 2000.