UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA TEÓRICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUSTAVO FÜHR

3D Tracking for an Augmented Reality System

Trabalho de Graduação

Claudio Jung
Orientador

Porto Alegre, 19 de Setembro de 2011

CONTENTS

LIST OF FIGURES **5**

LIST OF TABLES **6**

ABSTRACT **7**

RESUMO **8**

1 INTRODUCTION **9**

1.1 Project Details **10**

2 POSE CAMERA ESTIMATION PROBLEM IN AUGMENTED REALITY **11**

3 STATE OF THE ART METHODS **14**

3.1 Tracking-by-detection Methods **14**
3.1.1 SIFT Features 15
3.2 Recursive Tracking Methods **15**
3.2.1 Kanade-Lucas-Tomasi Feature Tracker 16
3.2.2 Bayesian Tracking and Particle Filtering 16
3.2.3 Template Matching 18

4 EXPERIMENTS WITH STATE OF THE ART METHODS . . . **25**

4.1 Dataset Used in the Experiments **25**
4.2 Experiments **26**
4.2.1 SIFT Features 26
4.2.2 Kanade-Lucas-Tomasi Feature Tracker 28

4.2.3 Bayesian Tracking 29
4.2.4 Template Matching 30
4.3 Discussion **32**

5 PROPOSED SYSTEM AND IMPLEMENTATION **34**

5.1 Proposed System **34**
5.2 Implementation Details **35**
5.3 Results **37**

6 CONCLUSION **41**

REFERENCES **43**

LIST OF FIGURES

2.1 Example of an AR application. To "augment" the virtual teapot, the camera pose must be obtained 11
2.2 Image formation geometry in the pinhole camera model 12

3.1 An example of detection of SIFT points in a texture (431 points were detected) 15

4.1 The texture present in the target object used in the dataset . . . 25
4.2 Examples of frames in the dataset used in the experiments 26
4.3 Tracking by detection using SIFT features 27
4.4 SIFT-based method performance in the test dataset 27
4.5 Lost of track using a KLT tracker 28
4.6 Performance of KLT in the test dataset 28
4.7 Particle filter performance in the test dataset 29
4.8 IC and ESM methods performances in the test dataset. 30
4.9 IC loses track due to fast motions 31
4.10 Number of iterations to converge: due to the ESM second-order characteristic, the algorithm takes much less iterations to converge as oppose to first-order algorithms such as IC 31
4.11 Example frames of the dataset showing tracking results for several of the studied methods 33

5.1 Times taken per iteration for several ESM implementations 36
5.2 Flow diagram of the ESM implementation on GPU 37
5.3 Hybrid system performance in the test dataset 38
5.4 Pecentage of good localization for several methods. Here, good localization is defined as the number of frames with a NCC value greater than the threshold ϵ. 39
5.5 Screenshots of the final system with augmented objects 40

LIST OF TABLES

4.1 Comparison of the studied tracking methods for planar objects . . 32

5.1 Times (in ms) taken per iteration of different ESM implementations 38
5.2 NCC average (and median value) computed for the whole sequence of the dataset. 39

ABSTRACT

One of the requirements when working with augmented objects in an Augmented Reality system is to identify the pose of the target object with respect to the camera with high accuracy and in real-time. The problem is of great importance in the field, thus it has been widely studied and there is a variety of methods proposed.

The objective of the final year project described in this report was to study these techniques in search for a robust solution to the problem of real-time tracking of a textured plane for Augmented Reality purposes. We present the studies of several state-of-the-art methods, separating them into two categories: tracking-by-detection methods and recursive methods. We also show the results of experiments evaluating these methods under conditions commonly encountered in practical applications.

Based on the experiments results, a hybrid system is proposed to combine the complementary qualities of two methods, achieving real-time performance and robustness to the challenges present in the 3D tracking of a textured plane. Finally, a GPU implementation of the system is described which improves the performance in more than three times compared to the CPU version – the final system runs at 30 frames per second.

Keywords: Augmented reality, 3D tracking, feature points, template-based tracking.

Rastreamento 3D para um Sistema de Realidade Aumentada

RESUMO

Um dos requisitos para se trabalhar com objetos aumentados em um sistema de Realidade Aumentada é a identificação (precisa e em tempo real) da posição e orientação do objeto alvo em relação à camera que captura a cena. Este problema é de grande importancia na área, consequentemente ele é amplamente estudado e diversos métodos já foram propostos.

Este trabalho de conclusão de curso tem por objetivo descrever o estudo realizado de várias técnicas do estado da arte na procura de uma solução robusta para o problema de rastreamento em tempo real de um objeto planar texturizado. Os diferentes métodos encontrados na literatura são apresentados, separados em duas categorias: rastreamento por detecção (*tracking by detection*) e métodos recursivos. Também são mostrados resultados de experimentos para avaliação das técnicas em condições comumente encontradas em aplicações reais de Realidade Aumentada.

Baseado nos experimentos realizados, um sistema hibrido é proposto para combinar as qualidades complementares de dois métodos, melhorando a robustez aos desafios presentes no rastreamento 3D. Adicionalmente, uma implementação do sistema em GPU é descrita, o que aumenta a performance do sistema em mais de três vezes – o sistema final processa 30 imagens por segundo.

Palavras-chave: Realidade Aumentada, rastreamento 3D, pontos de interesse, rastreamento baseado em *template*.

1 INTRODUCTION

In order to insert augmented objects in a real scene, a meaningful correspondence between the virtual objects and the real ones is required. To achieve this objective, an estimate of the camera pose in relation with the world must be obtained to place the virtual object in a "realistic" way. Besides that, in most AR applications, this estimation technique must perform in real-time.

Tracking an object in a video sequence consists in retrieving its localization continuously, despite motions of camera or motions of the object itself. As 3D tracking is a very useful tool in many different fields such as Computer Vision and Augmented Reality, it is a constant subject of research (Feng et al., 2008) and several methods were developed. A comprehensive review of many of these techniques can be found in (Lepetit and Fua, 2005).

The characteristics of the target object are very important to simplify the tracking. Fiducials, also called landmarks or markers, are easy to track in real-time and, for this reason, have been used for a long time in Augmented Reality applications. However, their utilization requires engineering the environment, which is not always desired and sometimes is even impossible (e.g. outdoor environments).

Therefore, the use of the object's natural features is certainly better although it makes the problem of tracking more challenging. To simplify the task, a 3D model of the object is usually employed for tracking. In this work, a textured plane is the target object. This kind of object is frequently used in AR systems to further reduce complexity, obtaining higher frame rates.

Different approaches were developed to solve the problem of camera estimation, depending on the type of object, the degrees of freedom of the object and camera, and the application itself. All the methods described here can be divided in two categories. The methods of the first category, called tracking-by-detection methods, try to find the object in the current frame matching detected features with a pre-computed database of the object features — in these techniques, each frame is processed independently. By contrast, recursive methods use the previous information to search the current object position and orientation around previous locations.

The objective of the final year project described in this report was to study the state of the art techniques in search for a robust solution to the problem of real-time tracking of a textured plane for Augmented Reality purposes. The only information required is the texture of the object. Also, contrarily to SLAM (Simultaneous Localization and Mapping) (Davison et al., 2007) and SfM (Structure from Motion) methods, we are not interested in the mapping of a scene but rather the tracking of a priori known model. We proposed a new method that combines the complementary qualities of two of the studied tecnhiques. Finally, an efficent implementation of this

hybrid approach on GPU is shown, which allows the system to run in real-time.

1.1 Project Details

This final year project took place at the Vision Research Department of THALES Training & Simulation and is focused on vision based camera registration for AR applications.

As there is a considerable number of tracking methods, the literature study of the problem was a constant task in the period of the internship. For the same reason, there was a phase of experimentation involving several methods; these experiments were performed using MATLAB due to the possibility of fast development in this programming language.

First, experiments using SIFT Matching (see Section 4.2.1) were performed. Then, several recursive methods were researched to couple with the SIFT solution and experiments were carried out to evaluate their tracking robustness. The first two methods, KLT (shown in Section 3.2.1) and Particle Filtering (Section 3.2.2), did not improve tracking.

Next, we studied the template-based methods (described in Section 3.2.3) that showed good results for tracking planar objects. In this report we describe all the methods that were studied in this project, emphasizing the Efficient Second-Order Minimization (Section 3.2.3.3). This method, in our proposed method, was combined with the SIFT Matching to achieve a higher tracking robustness.

After the state-of-the-art evaluation, we implemented this hybrid method on C++ using the library OpenCV. Also, to increase performance we use the library SIFT-GPU that implements the SIFT's detection and matching on GPU. For the ESM implementation, a GPU-based implementation was carried out to increase the frame rate using the NVidia's CUDA architecture.

Finally, the insertion of augmented objects using the OpenSceneGraph library was done, resulting in a complete real-time tracking system for Augmented Reality.

2 POSE CAMERA ESTIMATION PROBLEM IN AUGMENTED REALITY

Augmented Reality (AR) is the term applied when an application mixes a vision of the real world with superimposed virtual objects, in opposition to virtual reality systems in which all the objects are virtual ones. Usually, the real world information is provided through a camera and the virtual objects are placed in a manner that is consistent to the scene that is observed. The Figure 2.1 shows an example of an AR application, consisting of an "augmented" virtual teapot at the top of a book's cover.

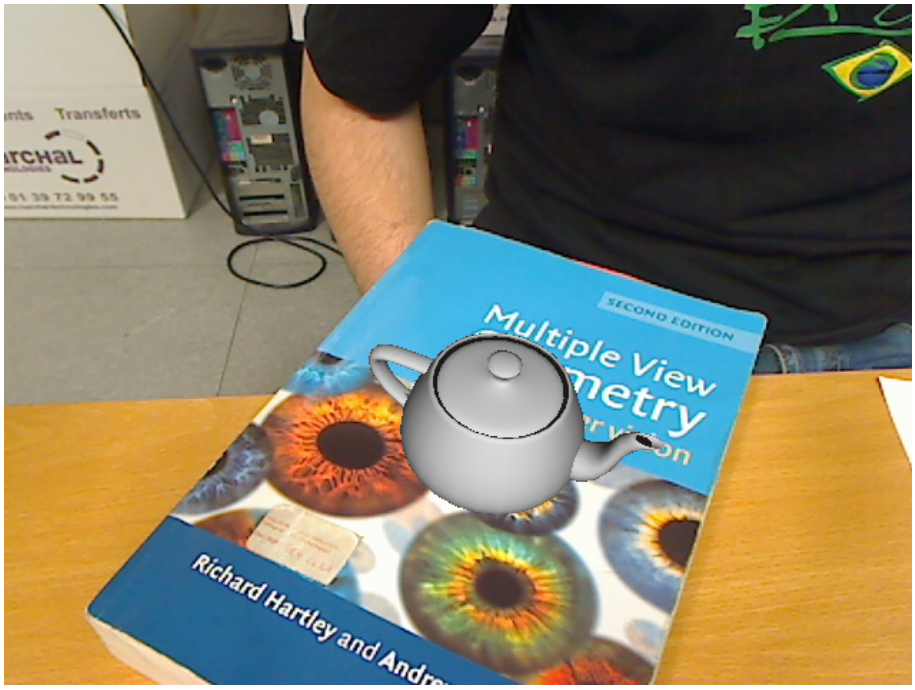


Figure 2.1: Example of an AR application. To "augment" the virtual teapot, the camera pose must be obtained

It is obvious that, in order to insert virtual objects in respect with the real scene, the camera pose in relation to the real world must be obtained; this problem is known for *Pose Camera Estimation*. This problem is inherently related with the geometry present in the image formation of the camera. In this work, we will be dealing with the common pinhole camera model that uses a perspective projection as shown by Figure 2.2.

The camera pose in relation with the object coordinate system can be retrieved

by finding a correspondence between the 3D object points $\mathbf{M} = [X, Y, Z]^\mathsf{T}$ and its 2D projections $\mathbf{m} = [X, Y]^\mathsf{T}$ in the image plane.

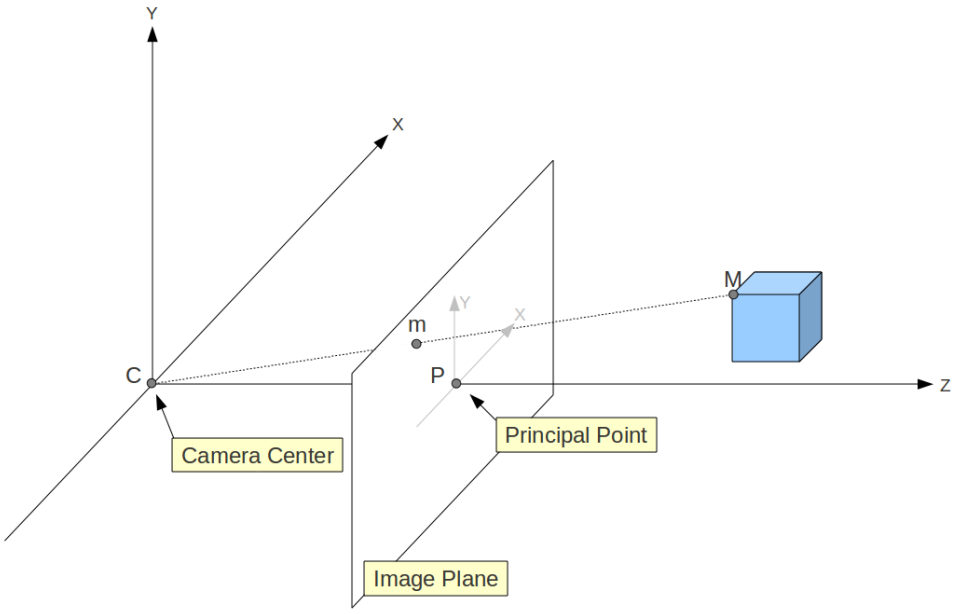


Figure 2.2: Image formation geometry in the pinhole camera model

Described with matrices, the pinhole camera model is represented in equation (2.1).

$$s\mathbf{m}' = P\mathbf{M}', \tag{2.1}$$

where s is a scale factor, $\mathbf{m}'$ and $\mathbf{M}'$ are the homogeneous coordinates of $\mathbf{m}$ and $\mathbf{M}$, respectively. Thus, the objective of tracking is to estimate the projection matrix P along the frame sequence. The matrix P can be decomposed as:

$$P = K[R|\mathbf{t}]$$

where K is intrinsic calibration matrix composed of camera internal parameters, such as:

$$K = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

The values of K are such that:

- α_x and α_y are the scale factors in the image x and y axes — these values are proportional to the focal length of the camera.
- u_0 and v_0 are the coordinates of the principal point P that is the intersection of the principal axis and the image plane, as indicated in Figure 2.2.

- s is a parameter regarding the skewness of the two images axes. Its value is nonzero only if the axes are not perpendicular.

The matrix K is constant for the video sequence, assuming that the camera zoom parameters do not change. Moreover, its values can be estimated in an offline procedure, as described in (Zhang, 2000), for instance. In the context of this report *K is assumed to be known.*

The interest of the methods described in the following sections is to find the extrinsic parameters in equation (2.1), i.e. the 3×3 matrix rotation R and the translation vector $\mathbf{t}$. Formally, the matrix $[R|\mathbf{t}]$, called external parameters matrix, is the Euclidean transformation from the world coordinate system to the camera coordinate system. And, for tracking applications, R and $\mathbf{t}$ represent the target object's position and orientation with respect to the camera.

The kind of object that we are interested in this work are textured planar objects, which simplifies the task of estimating the camera pose. If we consider 3D points lying on a plane with $Z = 0$, the projection matrix H that maps a point $\mathbf{M} = [X, Y, 0]^\mathsf{T}$ can be recovered by writing

$$\begin{aligned}
\mathbf{m}' &= P\mathbf{M}' \\
&= K \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{r_3} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \\
&= K \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\
&= H\mathbf{M}'
\end{aligned} \tag{2.3}$$

where $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_3}$ respectively are the first, second and third column of the rotation matrix R. Notice that, by abuse of notation, we use again $\mathbf{M}'$ denoting a point laying on the object, but in this case $\mathbf{M}' = [X, Y, 1]^\mathsf{T}$, i.e. it contains the homogeneous coordinates of a 2D-point since the Z coordinate is always equal to zero.

Several tracking methods focus on estimating the homography H that is a 3×3 matrix, to later recover the rotation and translation. The problem of decomposing an homography in R and $\mathbf{t}$ is known as Homography Decomposition and it is well addressed in (Malis and Vargas, 2007). The method used in our work is described in (Zhang, 2000).

3 STATE OF THE ART METHODS

The presentation of state-of-the-art techniques for 3D tracking in this report is separated in two main categories: tracking-by-detection methods and recursive methods. The main difference between them is that methods in the first category track each frame separately, i.e., no previous knowledge of the object trajectory is used.

The methods in the latter category use the poses obtained in previous time steps to estimate the object position and orientation in the current frame. Such an approach can greatly simplify the tracking, but it has its problems. Notably, if something goes wrong between two consecutive frames, e.g., due to a complete occlusion or a large displacement of the target object, the system might lose the tracking object. Moreover, manual initialization is often required in these techniques.

The popularity of the ARToolKit (Kato et al., 2000) comes from the fact that the system uses a tracking-by-detection approach to increase robustness. However, the toolkit requires the engineering of the environment with fiducial markers. One of the current challenges in Augmented Reality is to achieve the same performance level of the ARToolkit relying only on the object's natural features.

In the next two sections, both families of methods are presented. In Section 4, experiments carried out to evaluate their performances are shown.

3.1 Tracking-by-detection Methods

Tracking-by-detection approaches estimate the camera pose by detecting a known texture (or features) in each frame individually, without prior knowledge of the pose. In the studied methods, the natural features searched in a frame are called *interest points* which are points lying on the object and stable under image changes, i.e., it is possible to recover their positions in each frame to estimate the object pose.

Most of the methods of this class can be briefly described in two main steps:

- First, in an offline step, a database of the object's interest points is built from stored images (in some cases, only one image is enough). Several methods for detection and description of these natural features have been proposed (Mikolajczyk and Schmid, 2005; Lowe, 1999). A good comparison of different descriptors can be found in the work of Mikolajczyk and Schmid (Mikolajczyk and Schmid, 2005).

- Next, for each frame, interest points are detected in the whole image and then, matched against the database. From these correspondences between 2D

points, an estimation of the homography is obtained — for this, at least four correspondences must be found.

Usually, a minimization of the back-projection errors is done to compute the homography and a robust estimator, such as RANSAC (Fischler and Bolles, 1981), can be used to eliminate spurious matches.

In addition to the steps described above, to reduce jitter and enforce temporal consistency, a motion model can be included to impose temporal continuity constraints across frames.

Since the methods are very similar in their basis, the method for detecting and describing interest points is directly related with the tracker performance. In our experiments, which are described in section 4.2.1, we use the Scale Invariant Feature Transform (SIFT) (Lowe, 1999) for detect and represent interest points and no motion model was incorporated.

3.1.1 SIFT Features

Lowe proposed a feature point detection and descriptor called SIFT (Scale Invariant Feature Transform) (Lowe, 1999) that has been shown to be very efficient (Mikolajczyk and Schmid, 2005) and has been successfully applied to 3D tracking (Skrypnyk and Lowe, 2004).

The method to detect and describe SIFT features consists in several steps:

- SIFT keypoints are the identified feature points locations in the scale space of the image that are maxima or minima of the difference-of-Gaussian (DoG) function.
- A filter that excludes points within a region of low contrast is applied to the points obtained in the previous step.
- Finally, the image gradients in the local region of each keypoint are encoded using histograms of oriented gradients (HOGs) to provide rotation invariance.

An example of detection of SIFT points is shown in Figure 3.1.

Figure 3.1: An example of detection of SIFT points in a texture (431 points were detected)

3.2 Recursive Tracking Methods

Instead of trying to detect the object in each frame, traditional tracking methods search for the object around their previous positions. This approach greatly simplifies

the task of tracking, even if an initialization of the camera pose must be provided to start the system. The methods presented in this section compute recursively the camera trajectory using previous computed information such as homographies, positions of the object points or camera poses.

3.2.1 Kanade-Lucas-Tomasi Feature Tracker

The KLT (Kanade-Lucas-Tomasi) (Lucas and Kanade, 1981; Shi and Tomasi, 1993) tracker detects interest points in the first frame and defines a small window around each point. Then, in the subsequent frames, these windows are searched using affine transformations or only translations. The inter-frame movement is assumed to be small and the image brightness is assumed to be constant.

The translation of a window between the time t and $t+\tau$ defined by ξ and η must satisfies the following.

$$I(x, y, t+\tau) = I(x-\xi, y-\eta, t). \tag{3.1}$$

To compute this movement, generally, the displacement $\mathbf{d} = [\xi, \eta]$ is found by minimizing the residual error in a given window of pixels W:

$$\int_W (I(\mathbf{x}, t+\tau) - I(\mathbf{x}-\mathbf{d}, t))wd\mathbf{x}, \tag{3.2}$$

where w is a weighting function that can be, for instance, set to 1 for all pixels or a Gaussian-like function that emphasizes the central region of the window. Using the Taylor expansion of $I(\mathbf{x}-\mathbf{d}, t)$, we can solve equation (3.2) for $\mathbf{d}$:

$$\left[\sum_W \mathbf{G}^\mathsf{T}\mathbf{G}\right]\mathbf{d} = \sum_W \mathbf{G}^\mathsf{T} I(\mathbf{x}, \tau) \tag{3.3}$$

where $\mathbf{G} = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]$. This solution is based on the Lucas-Kanade algorithm, which is described in more details in section 3.2.3.

To detect feature points, Shi et al. propose in (Shi and Tomasi, 1993) that good features are those that can be tracked well, within a mathematical definition. Thus, the extraction of features depends on the tracking method that is used.

In the KLT method, features are selected by finding image points in which the eigenvalues (λ_1 and λ_2) of $\mathbf{G}^\mathsf{T}\mathbf{G}$, computed in their windows, is above a threshold ϵ_k.

$$\min(\lambda_1, \lambda_2) > \epsilon_k \tag{3.4}$$

3.2.2 Bayesian Tracking and Particle Filtering

Most of the Bayesian tracking methods aim to estimate the probability density function of states in the space of possible camera poses. Formulated in this fashion, they follow a top-down approach — not tracking points, as in other methods, but directly estimating and evaluating the camera pose.

Kalman filters (Welch and Bishop, 1995) have been used to estimate the localization with good results (Kragic and Kyrki, 2006), but they can work only with Gaussian distributions. Consequently, Particle Filtering is gaining interest in recent years for 3D tracking because it allows the use of more general distributions.

Particle filtering has been used to estimate the camera pose (Pupilli and Calway, 2006) with robustness and in real-time with or without the object 3D model —

we analyzed the former category. The problem in the second category consists in tracking the camera pose and simultaneously mapping the environment (Davison et al., 2007), known as SLAM (Simultaneous Localization and Mapping).

The purpose of a particle filter is to approximate a posterior density function with samples, called particles. To solve the problem of tracking, the posterior density function can be expressed in the general form of the recursive Bayes filter:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}, Z) = \frac{p(\mathbf{y}_k|\mathbf{x}_k, Z)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})} \tag{3.5}$$

where $\mathbf{x}_k$ is the state vector (camera motion) at time k, $\mathbf{y}_{1:k}$ are the observations $\mathbf{y}$ from the time $t = 1$ until $t = k$ and Z is a set of 3D points that represents the structure of the 3D model to be track. The reader interested in more details in Bayesian filters can find them in (Haug, 2005). In practice, each particle p_i corresponds to a possible motion of the camera pose and its weight is associated with the trustiness that the particle corresponds to the real camera motion.

The set of N particles p_i^k at the time k, denoted P_k, gives an estimate for the camera motion $\hat{\mathbf{X}}_k$ as follows:

$$\hat{\mathbf{X}}_k = E(X_k|Y_{1:k}) = \sum_{i=1}^{N} w_i^k \mathbf{x}_i^k,$$

assuming that the weights are normalized, such that $\sum_{i=1}^{N} w_i = 1$.

3.2.2.1 State Model

A particle p_i^k is composed of the motion estimates of the rotation R_i^k and translation $\mathbf{t}_i^k$. Rotations are modeled using unit quaternions (Altmann, 1986) for performance improvement and simplicity of manipulation. Therefore, the state model vector $\mathbf{x}$ is represented by seven values:

$$\mathbf{x} = [q_w, q_x, q_v, q_z, t_x, t_y, t_z]\,,$$

where t_x, t_y, t_z represent a translation and q_w, q_x, q_v, q_z are the quaternion values.

3.2.2.2 Motion Model

The motion model describes the changes in time of the particles states. It is difficult to assume any kind of movement for the camera pose a priori, so the main approach is to "move" the particles according to a random walk (Tao et al., 2009; Pupilli and Calway, 2005). Thus, the evolution of a particle is modeled by a uniform distribution.

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = U(\mathbf{x}_{k-1} - \mathbf{v}, \mathbf{x}_{k-1} + \mathbf{v}),$$

where $\mathbf{v}$ represents the uncertainty about the incremental motion. The value of the parameter $\mathbf{v}$ is very important for the system because, if too large, it may cause the divergence of particles (meaning poor estimate) and, if too small, the filter might not respond well in the presence of rapid movements.

To cope with these problems, in (Marimon and Ebrahimi, 2007) and (Tao et al., 2009), the authors proposed the introduction of an adaptive state transition model that adjust the value of $\mathbf{v}$ dynamically, in proportion to the difference of the last two estimates.

3.2.2.3 Measurement Model

The most important component of a particle filter is the measurement model that must be able to evaluate the particles in a consistent manner, distributing high weights for particles with good poses and low values for particles with bad ones.

Several likelihood functions have been proposed (Tao et al., 2009; Pupilli and Calway, 2005; Ababsa and Mallem, 2011; Marimon and Ebrahimi, 2007). These approaches only differ in their sets of observation points used to compute the likelihood function $p(\mathbf{y}_k|\mathbf{x}_k, Z)$.

In (Pupilli and Calway, 2005) and (Marimon and Ebrahimi, 2007), the authors propose correlation between pre-computed templates and the current frame as the set of observations. The procedure runs as follows. In an offline stage, a template is associated with each detected feature point and, for each frame, a field of cross correlation with the current image is computed. The set of observations is defined as the points in this field that are above a threshold.

Another approach, presented in (Tao et al., 2009), is to use the matching of natural features, using a reduced version of the Scale Invariant Feature Transform (SIFT (Lowe, 1999)). Thus, the set of observations are feature points that were matched with a reference frame (usually this is a picture of the object or a region in the first frame).

All the techniques described above evaluate the likelihood function in the same fashion. For each particle, the projection of a 3D feature point $\mathbf{z}_i$, denoted $C(\mathbf{z}_i, \mathbf{x}_k)$, is computed according to the particle's estimate of rotation and translation. Then, the likelihood of a particle is proportional to the numbers of feature points $\mathbf{z}_i$ whose projections are close enough to at least one observation point, i.e.

$$p(\mathbf{y}_k|\mathbf{x}_k, Z) \propto \exp\left(\sum_{i=1}^{M} \prod_{\mathbf{u}\in y_{ki}} d(\mathbf{u}, \mathbf{z}_i, \mathbf{x}_k)\right) \tag{3.6}$$

where M is the number of feature points and $d(\mathbf{u}, \mathbf{z}_i, \mathbf{x}_k)$ is a function that defines if the projection $C(\mathbf{z}_i, \mathbf{x}_k)$ is an inlier with respect to the observation point $\mathbf{u}$:

$$d(\mathbf{u}, \mathbf{z}_i, \mathbf{x}_k) = \begin{cases} 1 & \text{if } \| \mathbf{u} - C(\mathbf{z}_i, \mathbf{x}_k) \| < \varepsilon_d \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

3.2.3 Template Matching

Image alignment is used to register a 2D image template to an image using a deformation. When used to track objects, rather than tracking feature points, template-based techniques track the whole object, minimizing the sum of squared distances between the template and the current frame intensities at several $\mathbf{x}$ reference locations.

If the target object is planar, each frame showing the object can be deformed using a homography, in such a way that the object, in the back-warped image, is viewed with the same pose as in the reference frame. To approximate this optimal homography, template matching methods iteratively search, using the current homography estimate, the parameters of a homography that minimizes the error between the template and the current frame warped-back.

Since the seminal work of Lucas and Kanade (Lucas and Kanade, 1981) that presented an iterative image registration technique, several modifications to the

original algorithm have been proposed (Baker and Matthews, 2004; Malis, 2004). Although the original form of the algorithm was very computationally expensive, recent formulations made possible its use for real-time tracking (Benhimane and Malis, 2007; Jurie and Dhome, 2001).

In the following section, the original Lucas-Kanade algorithm is presented. Next, sections 3.2.3.2 and 3.2.3.3 show two alternative formulations for the template matching that are more efficient and section 4.2.4 presents some experiments involving these two techniques.

3.2.3.1 Lucas-Kanade Algorithm

The objective of the Lucas-Kanade algorithm (Lucas and Kanade, 1981) is to find the parameters $\mathbf{p} = [p_1, p_2, \dots p_n]^\mathsf{T}$ of a function W that warps the image I onto the coordinate frame of a template T, minimizing the following equation.

$$O(\mathbf{p}) = \sum_{\mathbf{x}} \left[I(W(\mathbf{x}; \mathbf{p})) - T(\mathbf{x}) \right]^2, \tag{3.8}$$

where $\mathbf{x}$ are pixel locations in the template. Note that warping back the image I to compute $I(W(\mathbf{x}; \mathbf{p}))$ requires interpolating the image I at the sub-pixels locations $W(\mathbf{x}; \mathbf{p})$.

In order to simplify the computation, the algorithm assumes that a current estimate for $\mathbf{p}$ is known, then it iteratively minimizes the equation (3.10) with respect to an increment $\Delta\mathbf{p}$ in the parameters. The update of $\mathbf{p}$ in the Lucas-Kanade algorithm is additive:

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}. \tag{3.9}$$

The equation (3.8) is rewritten to comport this alteration:

$$\sum_{\mathbf{x}} \left[I(W(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x}) \right]^2. \tag{3.10}$$

The algorithm iterates until the norm of the increments of $\mathbf{p}$ are below a threshold; i.e. $\|\Delta\mathbf{p}\| \leq \epsilon_c$.

The solution of the equation (3.10) is a nonlinear optimization task since normally, the image I is nonlinear with respect to $\mathbf{x}$. Hence, the equation (3.10) is linearized by performing a first-order Taylor expansion that, for a generic function $f(x)$, can be described as follows.

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x \tag{3.11}$$

In equation (3.10), the term $I(W(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p}))$ is linearized resulting in the expression:

$$\sum_{\mathbf{x}} \left[I(W(\mathbf{x}; \mathbf{p})) + \nabla I \frac{\partial W}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right]^2, \tag{3.12}$$

where ∇I represents the gradient $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ of the image I evaluated at $W(\mathbf{x}; \mathbf{p})$ — i.e. ∇I is computed in the coordinate frame of I and then warped back using the current parameters $\mathbf{p}$. The term $\frac{\partial W}{\partial \mathbf{p}}$ is called the Jacobian of the warp $W(\mathbf{x}; \mathbf{p}) = [W_x(\mathbf{x}; \mathbf{p}), W_y(\mathbf{x}; \mathbf{p})]^\mathsf{T}$ defined as follows:

$$\frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{bmatrix}. \tag{3.13}$$

This Jacobian can be constant for simple warps, such as affine transformations and translations. For other warps, the parametrization of W is very important in order to simplify the computation task.

To minimize the expression (3.12), we take its partial derivative with respect to $\Delta\mathbf{p}$ and set it equal to zero. Then, solving this equation for $\Delta\mathbf{p}$, results in the following expression:

$$\Delta\mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial W}{\partial \mathbf{p}} \right]^{\mathsf{T}} \left[T(\mathbf{x}) - I(W(\mathbf{x};\mathbf{p})) \right], \tag{3.14}$$

where H is the Hessian matrix approximate by the Gaussian-Newton method:

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial W}{\partial \mathbf{p}} \right]^{\mathsf{T}} \left[\nabla I \frac{\partial W}{\partial \mathbf{p}} \right]. \tag{3.15}$$

The Lucas-Kanade algorithm iterates using equations (3.14) and (3.9) to approximate the minimization of equation (3.10). The derivation described above is not the only possible solution to approximate the minimization equation (3.8), and several methods have been proposed; a good comparison can be found in (Baker and Matthews, 2004). The next section will present one of these approaches, that it was proven to be empirically equivalent (in terms of convergence) to the Lucas-Kanade algorithm.

3.2.3.2 Inverse Compositional

In the Lucas-Kanade algorithm, the Jacobian $\frac{\partial W}{\partial p}$ and the gradient ∇I generally must be computed at each iteration, because they depend on the current estimate of $\mathbf{p}$; in the case of $\frac{\partial W}{\partial p}$, as mentioned, it can be constant for simple deformations. The Inverse Compositional algorithm (IC) (Baker and Matthews, 2001) tries to reformulate the original minimization function with the intention of precomputing some of the values to achieve more computational efficiency.

The first difference of the Inverse Compositional algorithm to the original definition is the compositional aspect of the parameters update. Rather than an additive update, like in equation (3.9), the update is defined by:

$$W(\mathbf{x};\mathbf{p}) \leftarrow W(\mathbf{x};\mathbf{p}) \circ W(\mathbf{x};\Delta\mathbf{p})^{-1}. \tag{3.16}$$

Thus, together with the parametrization of the warp W, it is necessary to provide to the algorithm a rule for the function composition.

The inversion of the warp $W(\mathbf{x};\Delta\mathbf{p})$ is due to the inverse characteristic of the algorithm, that switches the role of the image and the template. Then, the minimization function (3.10) can be rewritten.

$$O(\Delta\mathbf{p}) = \sum_{\mathbf{x}} \left[T(W(\mathbf{x};\Delta\mathbf{p})) - I(W(\mathbf{x};\mathbf{p})) \right]^2 \tag{3.17}$$

With these two modifications in the original proposition, it is possible to pre-compute the Hessian H matrix and the Jacobian $\frac{\partial W}{\partial p}$ as the following derivations show.

First, we can apply a first-order Taylor expansion on equation (3.17) to obtain:

$$\sum_{\mathbf{x}} \left[T(W(\mathbf{x};\mathbf{0})) + \nabla T \frac{\partial W}{\partial \mathbf{p}} \Delta\mathbf{p} - I(W(\mathbf{x};\mathbf{p})) \right]^2 . \tag{3.18}$$

Assuming that the warp $W(\mathbf{x};\mathbf{0})$ is the identity warp, the solution for $\Delta\mathbf{p}$ is:

$$\Delta\mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[\nabla T \frac{\partial W}{\partial \mathbf{p}} \right]^{\mathsf{T}} [I(W(\mathbf{x};\mathbf{p})) - T(\mathbf{x})] \tag{3.19}$$

where H is the matrix defined as follows:

$$H = \sum_{\mathbf{x}} \left[\nabla T \frac{\partial W}{\partial \mathbf{p}} \right]^{\mathsf{T}} \left[\nabla T \frac{\partial W}{\partial \mathbf{p}} \right] . \tag{3.20}$$

This matrix can be pre-computed and used across iterations since there is nothing in the matrix that depends on $\mathbf{p}$; also, the Jacobian $\frac{\partial W}{\partial \mathbf{p}}$ is evaluated at $(\mathbf{x};\mathbf{0})$ and ∇T is the gradient of the original template, thus they can be precomputed too.

Compared with the Lucas-Kanade algorithm, the IC is more computationally efficient and can be use for real-time tracking, but it still relies on a first-order approximation of the minimization function. The objective of the algorithm described in the next section is to perform a second-order approximation while maintaining a high computational efficiency.

3.2.3.3 Efficient Second-Order Minimization

The Efficient Second-Order Minimization algorithm (ESM) (Malis, 2004) performs a second-order approximation of the minimization function that reduces the number of iterations to converge and, given a suitable parameterization, conserves the same computation complexity as standard first-order approaches.

With the intent to compare the ESM method with the previous ones, we reformulate the minimization problem. For an image I with q pixels, we can define a $(q \times 1)$ vector $\mathbf{y}(\Delta\mathbf{p})$ of the image differences, such as:

$$\mathbf{y}(\Delta\mathbf{p}) = \begin{bmatrix} y_1(\Delta\mathbf{p}) & y_2(\Delta\mathbf{p}) & \dots & y_q(\Delta\mathbf{p}) \end{bmatrix}^{\mathsf{T}}, \tag{3.21}$$

where

$$y_i(\Delta\mathbf{p}) = I(W(W(\mathbf{x}_i;\Delta\mathbf{p});\mathbf{p})) - T(\mathbf{x}_i).$$

Note that $\mathbf{p}$ are the assumed known approximations for the W parameters. Next, to linearize the vector $\mathbf{y}(\Delta\mathbf{p})$, a second order Taylor approximation on $\mathbf{y}(\Delta\mathbf{p})$ is performed:

$$\mathbf{y}(\Delta\mathbf{p}) \approx \mathbf{y}(\mathbf{0}) + J(\mathbf{0})\Delta\mathbf{p} + \frac{1}{2} M(\mathbf{0},\Delta\mathbf{p})\Delta\mathbf{p}. \tag{3.22}$$

where the $(q \times n)$ Jacobian matrix $J(\mathbf{p})$ and the $(q \times n)$ matrix $M(\mathbf{p_a},\mathbf{p_b})$ are such as:

$$J(\mathbf{p}) = \nabla_{\mathbf{p}}\mathbf{y}(\mathbf{p}) = \begin{bmatrix} \frac{\partial y_1(\mathbf{p})}{\partial p_1} & \frac{\partial y_1(\mathbf{p})}{\partial p_2} & \cdots & \frac{\partial y_1(\mathbf{p})}{\partial p_n} \\ \frac{\partial y_2(\mathbf{p})}{\partial p_1} & \frac{\partial y_2(\mathbf{p})}{\partial p_2} & \cdots & \frac{\partial y_2(\mathbf{p})}{\partial p_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y_q(\mathbf{p})}{\partial p_1} & \frac{\partial y_q(\mathbf{p})}{\partial p_2} & \cdots & \frac{\partial y_q(\mathbf{p})}{\partial p_n} \end{bmatrix} \tag{3.23}$$

and

$$M(\mathbf{p_a}, \mathbf{p_b}) = \nabla_{\mathbf{p_a}}(J(\mathbf{p_a})\mathbf{p_b}) = \left[\frac{\partial^2 y_1(\mathbf{p_a})}{\partial {\mathbf{p_a}}^2}\mathbf{p_b} \quad \frac{\partial^2 y_2(\mathbf{p_a})}{\partial {\mathbf{p_a}}^2}\mathbf{p_b} \quad \cdots \quad \frac{\partial^2 y_q(\mathbf{p_a})}{\partial {\mathbf{p_a}}^2}\mathbf{p_b} \right].$$

In order to compute $M(\mathbf{p_a}, \mathbf{p_b})$ it is necessary to obtain the (8×8) Hessian matrices, one for each pixel. As in the previous methods, we use a sum of squared distances to minimize the error vector.

$$f(\Delta\mathbf{p}) = \frac{1}{2}\|\mathbf{y}(\mathbf{p}) + J(\mathbf{p})\Delta\mathbf{p} + \frac{1}{2}M(\mathbf{0}, \Delta\mathbf{p})\Delta\mathbf{p}\|^2 \tag{3.24}$$

A necessary condition for finding the global or the local minimum is that the f derivate in relation with $\Delta\mathbf{p}$ must be equal to zero. This operation is performed in the equation (3.24); then, the value of $\Delta\mathbf{p}$ can be estimate iteratively (using the standard Newton minimization):

$$\Delta\mathbf{p} = -S^{-1}J(\mathbf{p})^{\mathsf{T}}\mathbf{y}(\mathbf{p}),$$

where the matrix S depends on the Hessian matrices and must be invertible:

$$S = J(\mathbf{0})^{\mathsf{T}}J(\mathbf{0}) + \sum_{i=0}^{q} \frac{\partial^2 y_i(\mathbf{p})}{\partial \mathbf{p}^2} y_i(\mathbf{p}).$$

The two previous methods for template matching presented can be written in the same form above for comparison. For the Lucas-Kanade method, the error vector $\mathbf{y}$ can be formulated as the matrix of the individuals errors y_i such that $\forall i \in \{1, 2, \ldots, q\}$:

$$y_i(\Delta\mathbf{p}) = I(W(\mathbf{x_i}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x_i}).$$

Then, the Jacobian J, using the definition in (3.23) is:

$$J(\mathbf{p}) = \nabla_{\mathbf{p}}\mathbf{y}(\mathbf{p}) = \nabla I \frac{\partial W}{\partial \mathbf{p}}.$$

It can be seen that the Lucas-Kanade algorithm compute a first-order approximation of the matrix S:

$$S \approx J(\mathbf{p})^{\mathsf{T}}J(\mathbf{p}).$$

In the Inverse Compositional method, the error function y_i is extracted from equation (3.17):

$$y_i(\Delta\mathbf{p}) = I(W(\mathbf{x_i}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x_i}).$$

And the Jacobian $J(\mathbf{p})$ is defined as:

$$J(\mathbf{p}) = \nabla_{\mathbf{p}}\mathbf{y}(\mathbf{p}) = \nabla T \frac{\partial W}{\partial \mathbf{p}}.$$

As shown, the term $\nabla T \frac{\partial W}{\partial \mathbf{p}}$ could be precomputed, so the approximation of S in this case is constant:

$$S \approx \widehat{J}$$

First-order approximations of S simplify the computation task of the minimization, because does not involve the process of the Hessian matrices. The main idea of the ESM method is to perform a second-order approximation without computing the Hessian matrices. This is done by approximating the matrix $M(\mathbf{p_a}, \mathbf{p_b})$ in equation (3.22) using an first-order Taylor series approximation of $J(\Delta\mathbf{p})$ about $\mathbf{0}$:

$$J(\Delta\mathbf{p}) \approx J(\mathbf{0}) + M(\mathbf{0}, \Delta\mathbf{p})$$

$$M(0, \Delta\mathbf{p}) \approx J(\Delta\mathbf{p}) - J(\mathbf{p}).$$

Then, it becomes possible link the above equation with the error vector equation (3.22):

$$\mathbf{y}(\Delta\mathbf{p}) \approx \mathbf{y}(\mathbf{0}) + J(\mathbf{0})\Delta\mathbf{p} + \frac{1}{2}(J(\Delta\mathbf{p}) - J(\mathbf{0}))\Delta\mathbf{p}$$

$$\mathbf{y}(\Delta\mathbf{p}) \approx \mathbf{y}(\mathbf{0}) + \frac{1}{2}(J(\mathbf{0}) + J(\Delta\mathbf{p}))\Delta\mathbf{p}. \tag{3.25}$$

In (Benhimane and Malis, 2007), the authors demonstrate that the sum of Jacobians $J(\mathbf{0}) + J(\Delta\mathbf{p})$, if the deformation is a homography, can be rewritten as follows.

$$J_{esm} = J(\mathbf{0}) + J(\Delta\mathbf{p}) = (J_I + J_T)J_W(J_H(\mathbf{0}) + J_H(\Delta\mathbf{p})), \tag{3.26}$$

where J_I depends on the gradient of the warped image, J_T depends on the gradient of the template, J_W is computed using the pixels points in the template and J_H depends on the homography parametrization.

In the original application of ESM to homography computation (Benhimane and Malis, 2007), it was shown that if we parametrize the computation using Lie Algebra we can state that $J_H(\mathbf{0}) = J_H(\Delta\mathbf{p})$. This is very important for real-time applications, because in this formulation, the only Jacobian in expression (3.26) that must be computed at each iteration is J_I. Formulate with Lie Algebra, the homography update is done using the matrix exponential function:

$$H \leftarrow He^{A(\Delta(p))},$$

where A is a basis of the Lie Algebra. More information about Lie Algebra applied to homography parametrization can be found in (Benhimane and Malis, 2007).

Using the expression (3.26), the cost function (3.24) can be rewritten.

$$f(\Delta\mathbf{p}) = \frac{1}{2}\|\mathbf{y}(\mathbf{0}) + J_{esm}(\Delta\mathbf{p})\|^2$$

Solving for $\Delta\mathbf{p}$ results in:

$$\Delta\mathbf{p} = J_{esm}^{+}\mathbf{y}(\mathbf{0}).$$

where J_{esm}^{+} is the Moore-Penrose pseudo-inverse of J_{esm}. Since the number of rows is greater than the number of columns in this case, the pseudo-inverse is defined as:

$$J_{esm}^{+} = (J_{esm}^{\mathsf{T}} J_{esm})^{-1} J_{esm}^{\mathsf{T}}$$

The complete ESM algorithm is shown in Algorithm 1 in pseudo code format.

Algorithm 1 ESM tracking method

Require: T : template image; threshold ϵ_c; k_{max} : maximum number of iterations; I : current image; $\hat{H}$: initial estimate of homography;

Pre-compute J_W and J_H as in (Benhimane and Malis, 2007)
Calculate the gradient J_T using a Sobel filter
$done \leftarrow$ false
$H \leftarrow \hat{H}$
$k \leftarrow 0$
while ($done$ = false) and ($k < k_{max}$) **do**
    $k \leftarrow k + 1$
    Calculate I_W warping the current image I with H
    Calculate the gradient J_I using the image I_W (Sobel filter)
    $J_{esm} \leftarrow 2(J_I + J_T)J_W J_H$
    $\mathbf{y} \leftarrow I_T - I_W$ [y is the current error]
    $\Delta\mathbf{p} \leftarrow (J_{esm}^{\mathsf{T}} J_{esm})^{-1} J_{esm}^{\mathsf{T}}\mathbf{y}$
    $H \leftarrow He^{A(\Delta\mathbf{p})}$
    if $\|\Delta\mathbf{p}\| < \epsilon_c$ **then**
        $done \leftarrow$ true
    end if
end while
return H

Since its formulation, the ESM algorithm has also been adapted to work in more complex tracking situations such as the presence of large illumination changes (Silveira and Malis, 2007), blur (Park et al., 2009) and the tracking of deformable objects (Silveira and Malis, 2010).

4 EXPERIMENTS WITH STATE OF THE ART METHODS

4.1 Dataset Used in the Experiments

To compare the different methods, we performed experiments with the same dataset, consisting of a video with around 900 frames of the target object — a planar object with the texture of Figure 4.1. The frame sequence displays several challenges for the tracking, commonly encountered in a real life scenario, such as: rotations, partial occlusions, large displacements and severe changes in viewing angles and scale. Examples of these can be seen in Figure 4.2.

Figure 4.1: The texture present in the target object used in the dataset

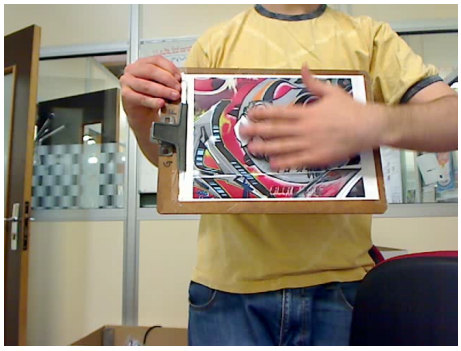
As the ground truth of the object pose is not known, to estimate the error of localization, we first warp the image using the inverse homography or transformation matrix provided by the tracking method and second, compare this image with the original template employing a Normalized Cross Correlation (NCC):

$$NCC(T, I_w) = \frac{\sum_{\mathbf{x}}(T(\mathbf{x}) - \mu_t)(I_w(\mathbf{x}) - \mu_w)}{q^2 \sigma_t \sigma_w}, \tag{4.1}$$

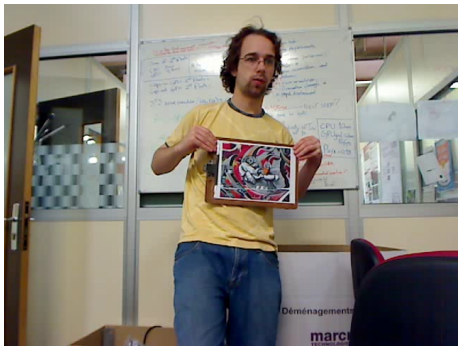
where μ_t and μ_w are the mean pixels intensities of the template T and back-warped image I_w and σ_t and σ_w their standard deviations; q is the total number of pixels.

Notice that, when the object is seen in a challenging condition, such as those when an occlusion happens or a frame has too much blur, the NCC coefficient cannot be very high even if the tracking is very accurate. Nonetheless, the NCC can provide important information about the accuracy of the pose and has been used in tracking experiments (Park et al., 2009).

(a) Frame #256 — Oblique viewing angle



(b) Frame #362 — Partial occlusion



(c) Frame #676 — Scale change

(d) Frame #799 — Beginning of a large displacement

Figure 4.2: Examples of frames in the dataset used in the experiments

4.2 Experiments

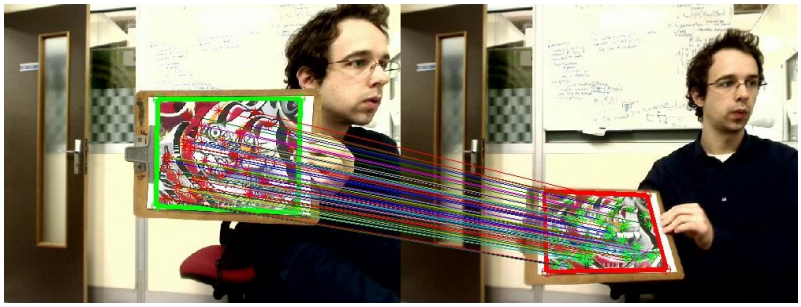
4.2.1 SIFT Features

In the SIFT-based system, the points detected in the template image (as shown in Figure 3.1) are stored and for each frame, SIFT points are detected and matched, using a Euclidean distance between two descriptors. Given these correspondences, an homography is estimated using RANSAC estimator to remove spurious matches.

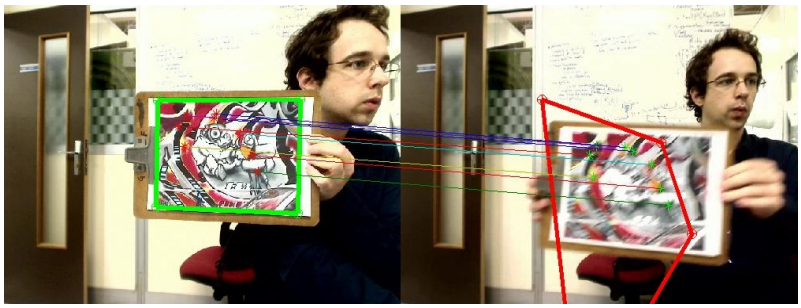
We ran some tests to evaluate the tracking performance of this method; two examples are shown in the Figure 4.3. To better evaluate the performance of the tracking system we also tested the method using our dataset — the results are shown in Figure 4.4.

The method can produce good poses (in terms of a NCC-based metric) under several difficult circumstances, as it can be seen in Figure 4.3a and in a large part of the dataset experiment shown in Figure 4.4. However, in the presence of blur due to fast movements the number of matches can be reduced, resulting a poor homography estimate, as shown in Figure 4.3b. Furthermore, other problems can be noticed if we analyze the NCC coefficients in Figure 4.4: the tracking system fails with frames that present severely oblique viewing angles and when the object is too far from the camera — in the latter case, the number of correspondences is reduced, causing jitter. Another drawback of this system, for Augmented Reality applications, is the high computational cost. However, a GPU optimization of the SIFT's detection and matching is proposed in (Sinha et al., 2006) which enables real-time performance.

Despite these problems, tracking-by-detection techniques have the capacity to



(a) Good homography estimate



(b) Poor homography estimate due to blur

Figure 4.3: Tracking by detection using SIFT features

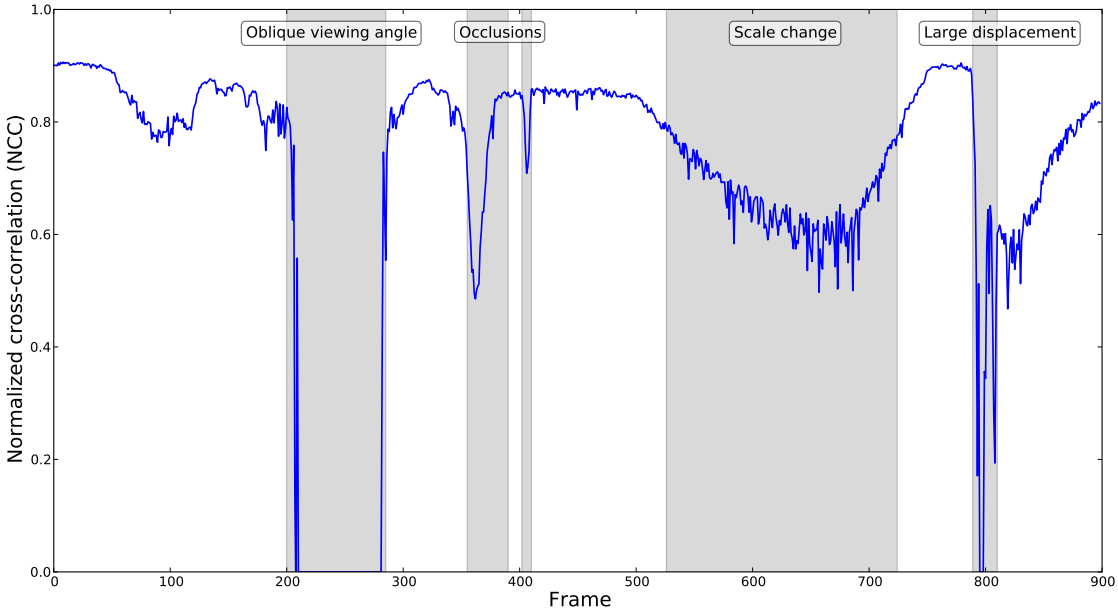


Figure 4.4: SIFT-based method performance in the test dataset

recover after they lost the object's position, which is exemplified in the Figure 4.4 that shows the system recovery after a large displacement.

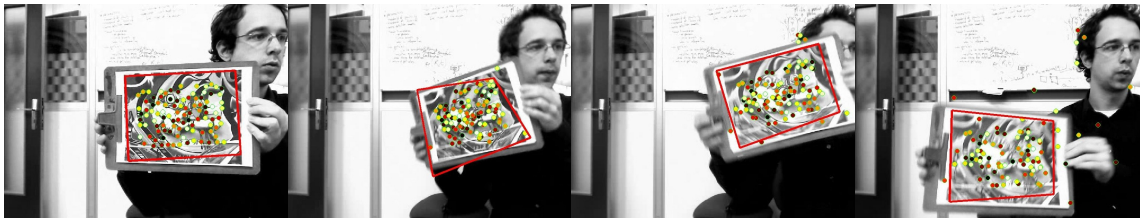
There are several approaches that combine tracking-by-detection methods and recursive tracking (e.g. (Ladikos et al., 2007)) to increase performance and/or estimation accuracy. Another improvement is to add a motion model in order to reduce jitter.

4.2.2 Kanade-Lucas-Tomasi Feature Tracker

To analyze the performance of a KLT tracker, we performed a test by tracking the detected points laying on the object in the first frame. The pixels inside a given window are weighted as proposed in (Schreiber, 2007) which has the effect of excluding pixels that show resulted in large residual errors in previous frames.

To compute the homography of the plane, RANSAC was used to remove outliers and no reinitialization of the interest points was made, with the intention to examine the error accumulation.

As we can see in Figure 4.5, the method progressively loses track of several features points due to fast motion and error accumulation. The experiment using the test dataset shows the same results, as it can be seen in Figure 4.6.



(a) Frame #20 (b) Frame #40 (c) Frame #60 (d) Frame #80

Figure 4.5: Lost of track using a KLT tracker

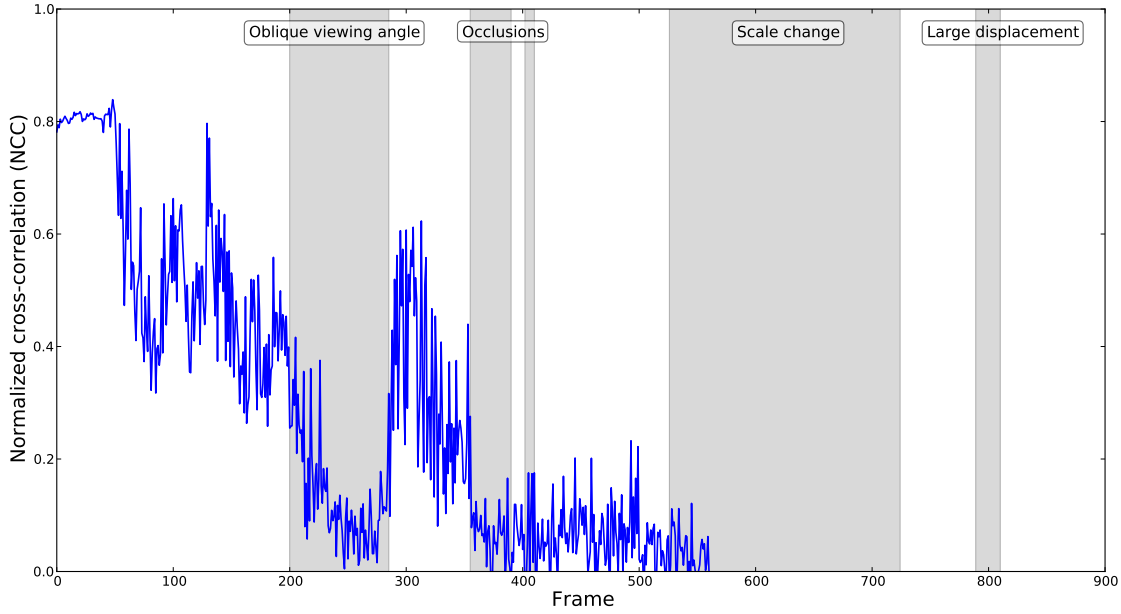


Figure 4.6: Performance of KLT in the test dataset

Despite the fact that the KLT tracker loses track easily when fast movements or occlusions occur, if the displacement between two frames remains small, the KLT tracker can be used in real-time. Due to its simplicity and good performance, hybrid approaches has been proposed (e.g. (Choi et al., 2008)) to improve tracking results.

4.2.3 Bayesian Tracking

After the study of the different types of measurement, SIFT matches were used as the set of observation points responsible for evaluating the pose hypotheses. The experiment of the particle filter in the dataset is shown in Figure 4.7. For performance reasons, the number of particles used was 300. Because of the random characteristic of the algorithm, the test was repeated 5 times and the plot shows the average for these executions.

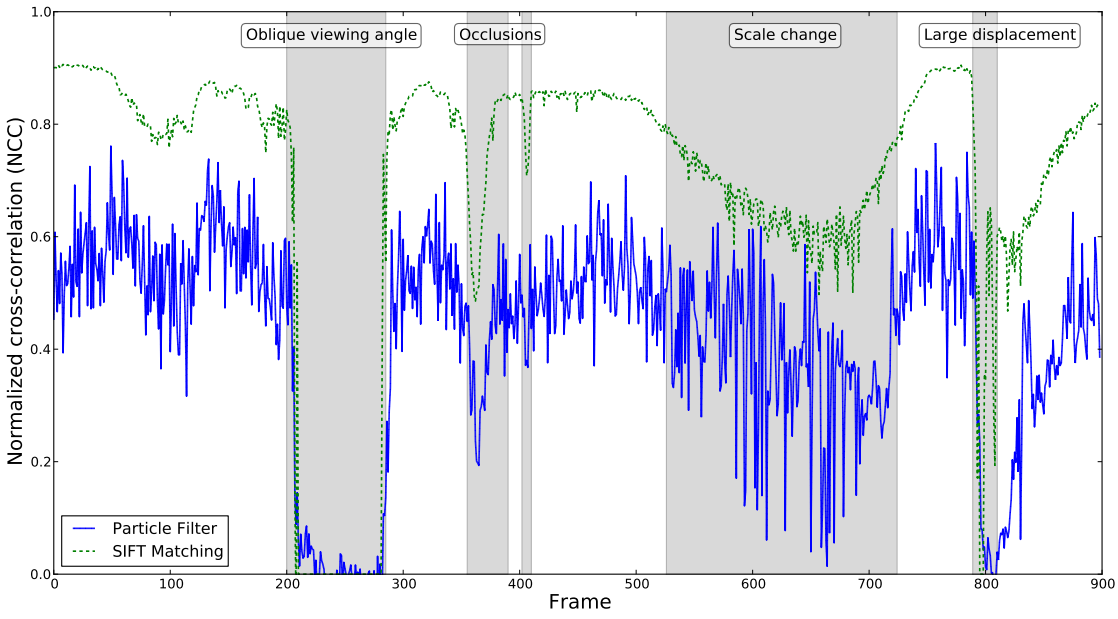


Figure 4.7: Particle filter performance in the test dataset

The results using the particle filter present the same pattern as the tracking using SIFT matches. Indeed, this correlation was already expected because the filter uses SIFT matches to evaluate particles. It is also clear from this experiment that a particle filter using only this kind of features cannot account for situations when few matches are found (e.g. when the object is observed from a oblique angle). Moreover, particle filters introduce a great deal of jitter due to the random nature of the algorithm which is further increased by the reduced number of particles used in the experiment.

To achieve better results, complementary methods can be included in the measurement model, so that, when one information is not available, other method can provide useful information to evaluate the particles. Despite that, it is hard to achieve real-time performance with particle filters without compromising the method accuracy. This is a prohibitive aspect of this method for our application because, in order to cope with fast movements, a large number of particles is required. This means that, when using particle filters, a choice between accuracy and real-time performance must be made. Because of that, particle filters are more popular in Computer Vision in problems that can be solved offline, such as Human Pose Estimation (Deutscher and Reid, 2005).

4.2.4 Template Matching

Experiments to evaluate two template-based methods, IC and ESM, were carried out using the same test dataset as for the other methods. The version of the ESM algorithm implemented for the tests was the ESM-Blur (Park et al., 2009), that modifies the original proposition by multiplying the jacobian of the template J_T, in equation (3.26), by a factor (we used 0.5). As the name suggests, this small modification makes the method more robust in the presence of blurry images.

Typically, template matching algorithms have two parameters: the threshold ϵ_c that defines when the method converged, i.e, if the norm in the parameters update is less or equal to ϵ_c, the computation for the frame is finished; additionally, a value for the maximum number of iterations allowed is fixed to determine when the solution did not converged. For the experiments performed, the initialization was done using SIFT (Lowe, 1999) matches to obtain an initial homography, $\epsilon_c = 0.01$ and the maximum number of iterations was fixed to 50. The results are presented in Figure 4.8.

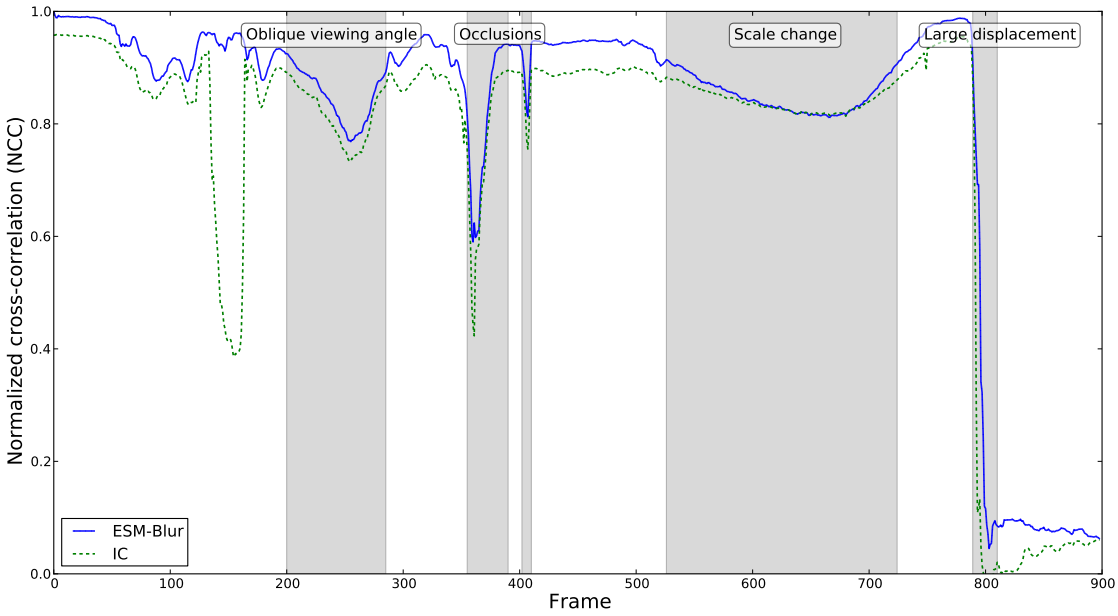


Figure 4.8: IC and ESM methods performances in the test dataset.

The results show us that the ESM method is more accurate for almost the whole sequence. These results arise from the fact that, as discussed, the ESM approximates the minimization function using a second-order approximation instead of using first-order approximations (as in the case of IC). It is possible to observe from the Figure 4.8 that around the frame #150 the IC method almost loses track of the object while the ESM maintain a good performace. After this, the IC method recovers a good localization, but this is not always the case. Another experiment, using a smaller dataset (consisting mainly of rapid motions) is shown in Figure 4.9 where the IC is unable to recover a good localization after loosing track around the 60^{th} frame.

The number of iterations necessary for convergence was also measured for both methods. The results for the test dataset are shown in Figure 4.10. The ESM

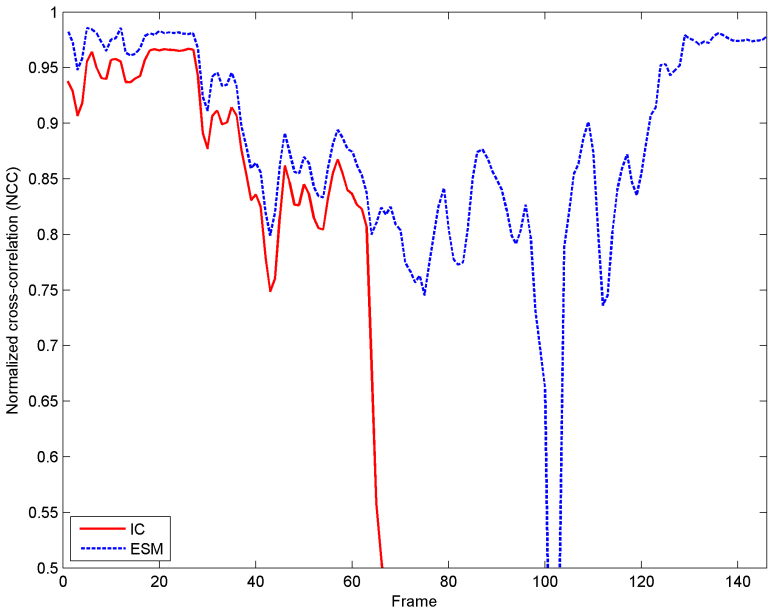


Figure 4.9: IC loses track due to fast motions

converges in less iterations than first-order algorithms, even though a single iteration can be more computationally expensive.

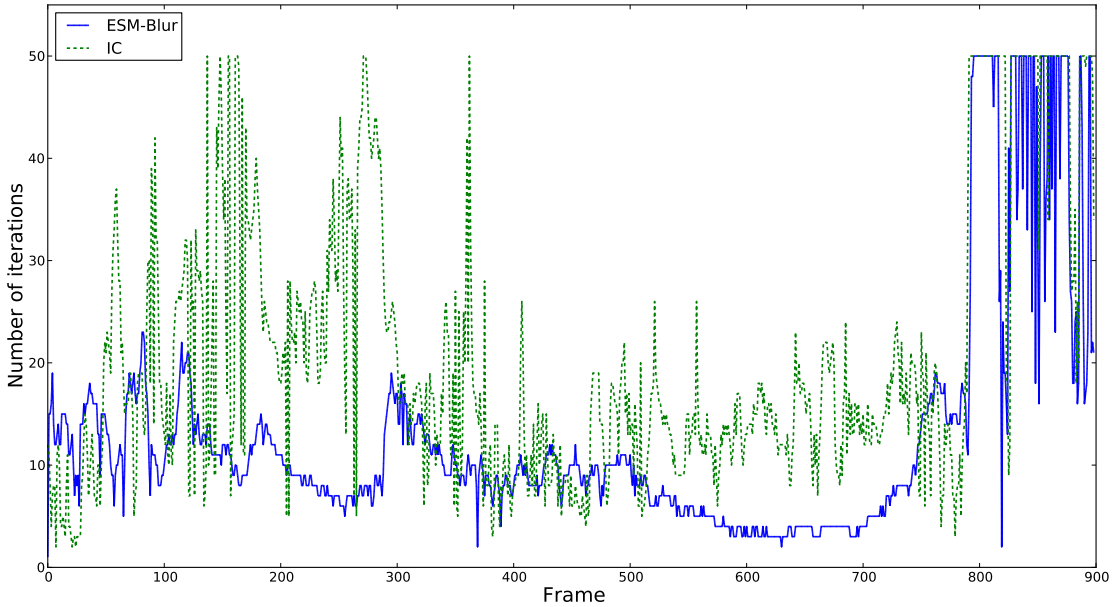


Figure 4.10: Number of iterations to converge: due to the ESM second-order characteristic, the algorithm takes much less iterations to converge as oppose to first-order algorithms such as IC

Despite the good results, template-based methods are unable to recover the tracking after a large displacement, as it can be seen in Figure 4.8. This suggests

that, in order to obtain a robust system, a reinitialization scheme must be put in place to recover a good pose when the tracking is lost.

4.3 Discussion

Tracking is one of the most important tasks in many fields as Augmented Reality, Visual Servoing, Surveillance, etc. Therefore, several 3D tracking methods can be found in the literature and the subject continues to be researched. However, the problem of robustly tracking an object using only its natural features is still an open problem.

To summarize the study of the several methods presented, Table 4.1 shows the performances of each method in the presence of constraints observed in real life applications.

Method × Constraint	Real-time	Scale	Oblique viewing	Large displacements	Blur	Occlusion
SIFT Matching	+	+	–	+	–	+
KLT	+	=	–	–	–	–
Particle Filter (SIFT)	–	+	=	=	=	+
IC (Template-based)	+	=	=	–	=	=
ESM (Template-based)	+	+	+	–	+	=

Table 4.1: Comparison of the studied tracking methods for planar objects

Table 4.1 shows that, on one hand, all the recursive methods that were tested cannot work well with large displacements, as discussed before. On the other hand, the tracking-by-detection based on SIFT matches is not robust under all constraints, giving poor estimates when the object is viewed with severely oblique angle, frames present blur or the camera is too far from the object. For a qualitative evaluation, Figure 4.11 shows several frames of the test dataset with the homography given by the different methods.

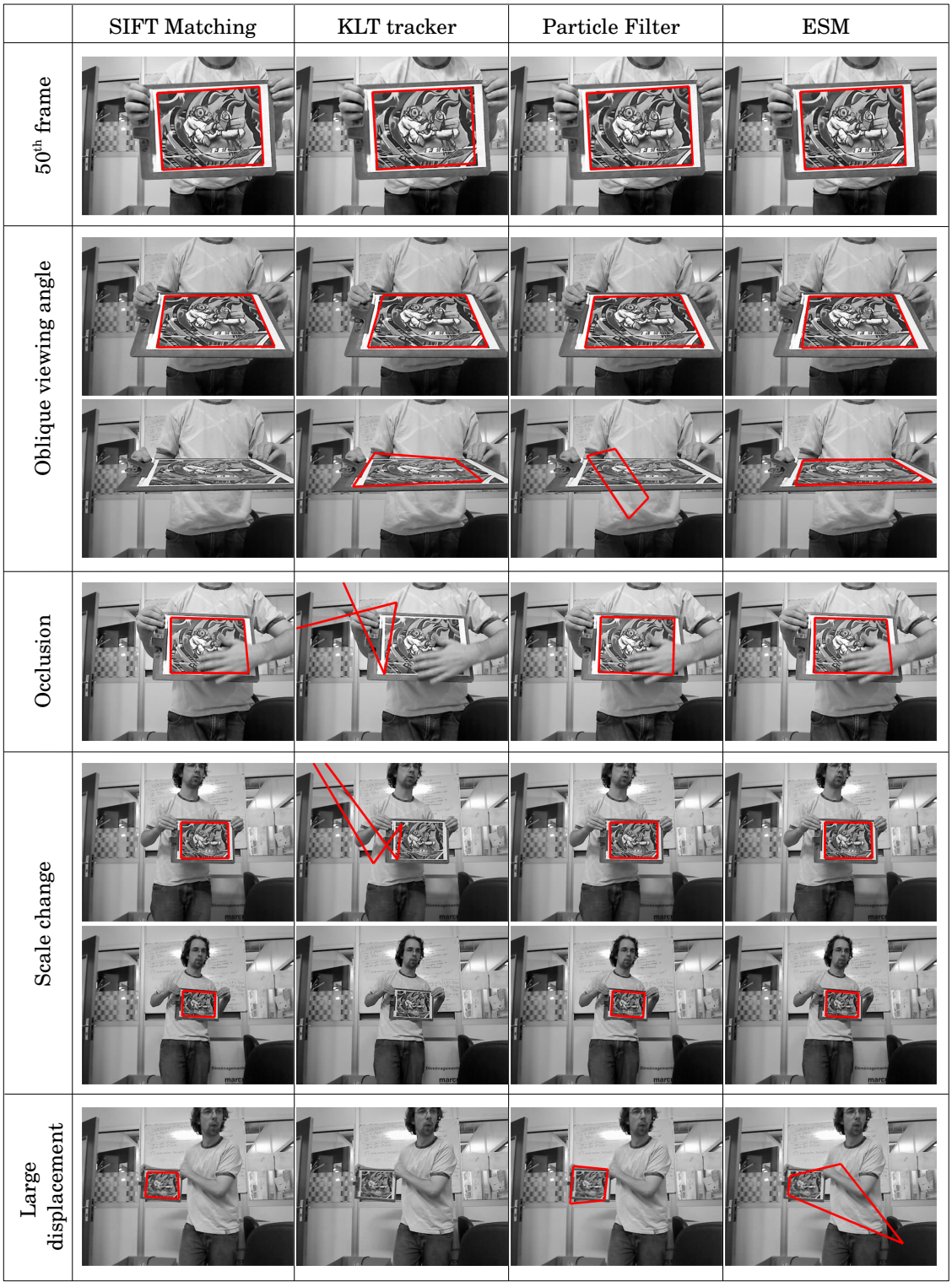


Figure 4.11: Example frames of the dataset showing tracking results for several of the studied methods

5 PROPOSED SYSTEM AND IMPLEMENTATION

Analyzing again the Table 4.1, it can be noticed that the SIFT Matching is the only technique that can cope in the presence of large displacements that commonly happen in real world scenarios. This comes with no surprise given that the technique is the only tracking-by-detection method in the comparison: the independence of the process for each frame is the key to deal with large displacements.

Recursive methods lose track of the object in the presence of large displacements because they assume that displacements will remain small. Moreover, template-based methods work minimizing a function that can result a local minimum instead of the global one.

These results indicate that a robust solution for the tracking might be a hybrid approach, using the complementary qualities of two methods. In this section, we propose a hybrid system using SIFT matching and ESM tracking.

5.1 Proposed System

To achieve a good performance, hybrid approaches were proposed (Ladikos et al., 2007) to combine the complementary qualities of two (or more) families of tracking methods. Our approach is similar to (Ladikos et al., 2007) and the sense that it also proposes the union of a recursive method with a tracking-by-detection method. However, our choice of methods is different from their choice. We combine two of the studied methods: ESM-Blur (Park et al., 2009) (recursive) and the SIFT Matching (Skrypnyk and Lowe, 2004) (tracking by detection).

The proposed system can be described with two modes:

- Tracking-by-detection mode: SIFT features are extracted and matched against the database of the template features. If the procedure finds a minimum of ϵ_s correspondences, an homography H is computed using RANSAC. Then, the system goes to the recursive tracking mode.

- Recursive tracking mode: With the H provided in the previous frame, the current frame is processed using a template-based technique (ESM-blur (Park et al., 2009)). After the convergence of the minimization (or the maximum allowed number of iterations exceeded), the frame is warped using the inverse of the computed homography. To evaluate the current estimate of H, this back-warped image is compared to the model using a normalized cross correlation (NCC) (see equation (4.1)).

If the NCC coefficient is inferior to a threshold ϵ_n, the system changes to the previous mode because it is assumed that the template-based tracker lost the object. The value of ϵ_n was defined empirically — a value around 0.6 shows good results.

5.2 Implementation Details

For the tracking-by-detection mode we used the library SIFT-GPU[1] that implements the SIFT detection and matching on GPU (Sinha et al., 2006). The size of frames has a great impact on the performance because the detection of SIFT features is performed in the whole frame — the complete tracking-by-detection mode, using a NVIDIA Quadro FX 3800 and 640×480 frames, leads to the performance around 12fps.

For the recursive tracking mode, we started with a CPU implementation using the OpenCV library, which provides optimized functions to perform the algebraic operations that the ESM needs. This first implementation did not result a real-time performance.

As the most costly step in the ESM algorithm is the computation of J_{esm}, a second implementation was carried out using the fact that it is *not necessary to compute J_{esm} explicitly*. In Algorithm 1, it can be seen that J_{esm} is used to compute the update of the homography parameters $\Delta\mathbf{p}$. These values are computed using two multiplications involving J_{esm}: $J_{esm}^{\mathsf{T}} J_{esm}$ and $J_{esm}^{\mathsf{T}}\mathbf{y}$.

Given that J_{esm} is a q×8 matrix and writing the values of its i^{th} line as $[a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i]$, the product $J_{esm}^{\mathsf{T}} J_{esm}$ can be written

$$J_{esm}^{\mathsf{T}} J_{esm} = \begin{bmatrix} a_1 & a_2 & \dots & a_q \\ b_1 & b_2 & \dots & b_q \\ c_1 & c_2 & \dots & c_q \\ d_1 & d_2 & \dots & d_q \\ e_1 & e_2 & \dots & e_q \\ f_1 & f_2 & \dots & f_q \\ g_1 & g_2 & \dots & g_q \\ h_1 & h_2 & \dots & h_q \end{bmatrix} \begin{bmatrix} a_1 & b_1 & c_1 & d_1 & e_1 & f_1 & g_1 & h_1 \\ a_2 & b_2 & c_2 & d_2 & e_2 & f_2 & g_2 & h_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_q & b_q & c_q & d_q & e_q & f_q & g_q & h_q \end{bmatrix}$$

$$J_{esm}^{\mathsf{T}} J_{esm} = \sum_{i=1}^{q} \begin{bmatrix} a_i^2 & a_i b_i & a_i c_i & a_i d_i & a_i e_i & a_i f_i & a_i g_i & a_i h_i \\ a_i b_i & b_i^2 & b_i c_i & b_i d_i & b_i e_i & b_i f_i & b_i g_i & b_i h_i \\ a_i c_i & b_i c_i & c_i^2 & c_i d_i & c_i e_i & c_i f_i & c_i g_i & c_i h_i \\ a_i d_i & b_i d_i & c_i d_i & d_i^2 & d_i e_i & d_i f_i & d_i g_i & d_i h_i \\ a_i e_i & b_i e_i & c_i e_i & d_i e_i & e_i^2 & e_i f_i & e_i g_i & e_i h_i \\ a_i f_i & b_i f_i & c_i f_i & d_i f_i & e_i f_i & f_i^2 & f_i g_i & f_i h_i \\ a_i g_i & b_i g_i & c_i g_i & d_i g_i & e_i g_i & f_i g_i & g_i^2 & g_i h_i \\ a_i h_i & b_i h_i & c_i h_i & d_i h_i & e_i h_i & f_i h_i & g_i h_i & h_i^2 \end{bmatrix}. \quad (5.1)$$

Notice that the matrix $J_{esm}^{\mathsf{T}} J_{esm}$ is symmetric, thus for each pixel only 36 values are required to find the matrix (and 8 values for $J_{esm}^{\mathsf{T}}\mathbf{y}$). It can be seen in equation (5.1) that each pixel can be processed independently, thus only one passage in the template pixels is necessary to compute $J_{esm}^{\mathsf{T}} J_{esm}$ and $J_{esm}^{\mathsf{T}}\mathbf{y}$.

[1]The library is currently available for free at http://www.cs.unc.edu/c̃cwu/siftgpu/

This can increase the algorithm speed, because, if the values of the two matrix multiplications are computed directly, calculate $\Delta\mathbf{p}$ consists only of the inversion of an 8×8 matrix and a multiplication by a 8×1 vector.

The times taken per iteration are shown in Figure 5.1. It can be seen that the computation times in the implementation that does not calculate the J_{esm} explicitly (in the picture, this version is labeled CPU - version 2) are smaller than the times in the first CPU implementation.

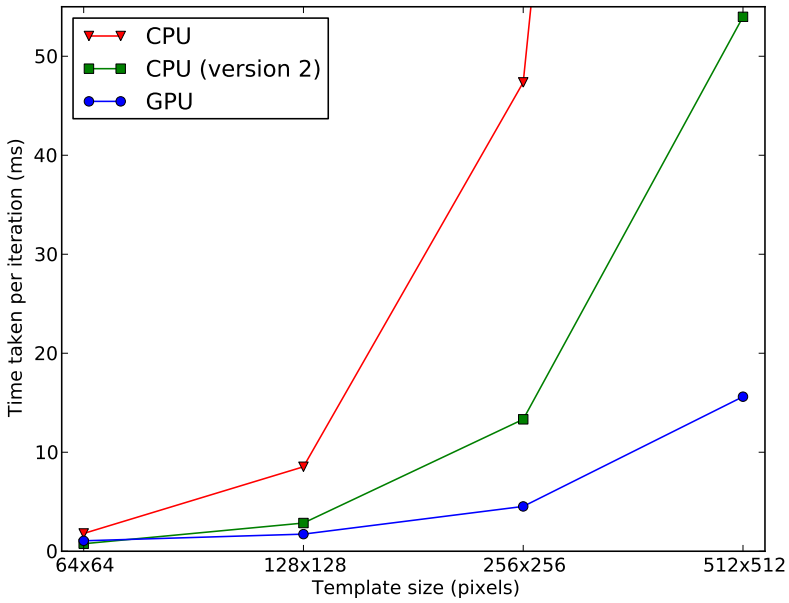


Figure 5.1: Times taken per iteration for several ESM implementations

To further increase the performance, we developed a GPU version of ESM, also not computing J_{esm} explicitly, as the above method. The application was coded using the NVIDIA's CUDA framework that greatly simplifies general purpose programming on GPU. GPUs are designed to process data in parallel, which is suitable for image processing applications and therefore tracking applications.

Two of the most important aspects in general-purpose computing on graphics processing units (GPGPU) is to try to achieve high parallelism between threads and to minimize the GPU-CPU transfers, which are slow. Our implementation, shown in the diagram of Figure 5.2, follows these characteristics: all of the procedures that are executed on the GPU are of high parallelism. In all the cases, for each pixel coordinate of the template a thread on the GPU is created.

An important issue faced was how to perform the sums of values needed to build the matrices $J_{esm}^{\mathsf{T}}J_{esm}$ and $J_{esm}^{\mathsf{T}}\mathbf{y}$ on GPU. The naive approach of using the same GPU memory area is not optimized and incorrect because writing in the GPU memory is not an atomic operation. If the sum is fixed to be atomic, the performance will drop because each thread must wait its time to read/write the memory. The implemented solution consists of each thread writing its own matrix with a parallel reduction procedure taking place afterwards. At each time-step, the number of threads and matrices is reduced by a factor of 2. After $\log q$ steps we have only one matrix.

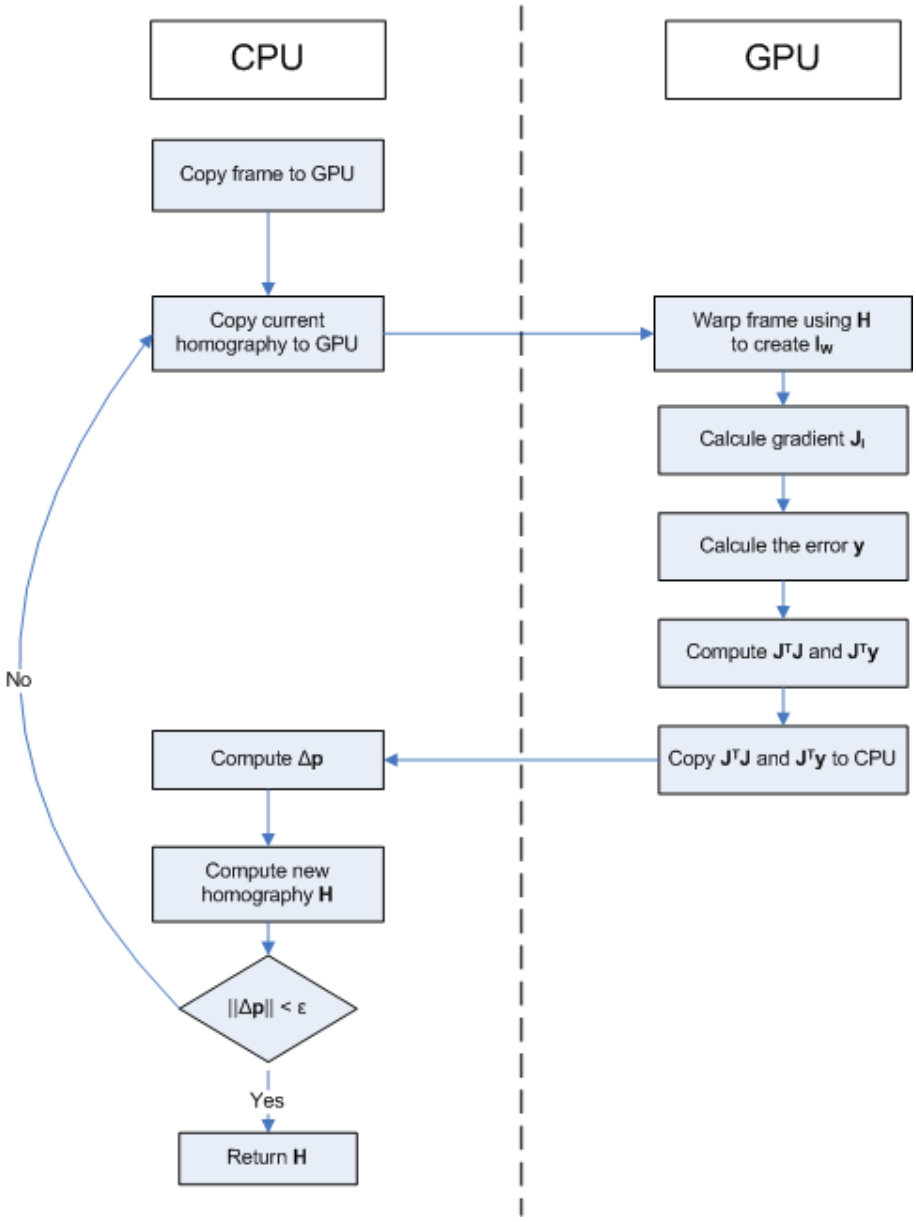


Figure 5.2: Flow diagram of the ESM implementation on GPU

After the computation of the final $J_{esm}^{\mathsf{T}} J_{esm}$ and $J_{esm}^{\mathsf{T}} \mathbf{y}$, the matrices are copied to the CPU to compute $\Delta\mathbf{p}$. The calculation of $\Delta\mathbf{p}$ is done on CPU, because it cannot achieve a large optimization on GPU, as it involves only a few values.

Our implementation of ESM in GPU greatly improves the performance in comparison with CPU implementation, as the Figure 5.1 shows. For templates of size equal or larger than 128×128, the GPU implementation is faster. Table 5.1 shows additional information about the times taken per iteration of each implementation.

5.3 Results

The tracking performance with the dataset used along this report is shown in Figure 5.3. It can be seen that the system retains a good localization in almost every frame and recovers itself after a large displacement. These good results are mainly due to two important aspects. First, the characteristic of tracking-by-detection methods

Implementation × Template size	64×64	128×128	256×256	512×512
CPU	1.79	8.53	47.35	266.02
CPU (version 2)	**0.75**	2.84	13.32	53.98
GPU	1.05	**1.72**	**4.52**	**15.61**

Table 5.1: Times (in ms) taken per iteration of different ESM implementations

that treat each frame individually and therefore are able to recover a good pose when something goes "wrong" between two consecutive frames (a large displacement in our dataset). Second, the robustness of template-based techniques that are able to track the target object in real-time when the inter-frame displacement remains small.

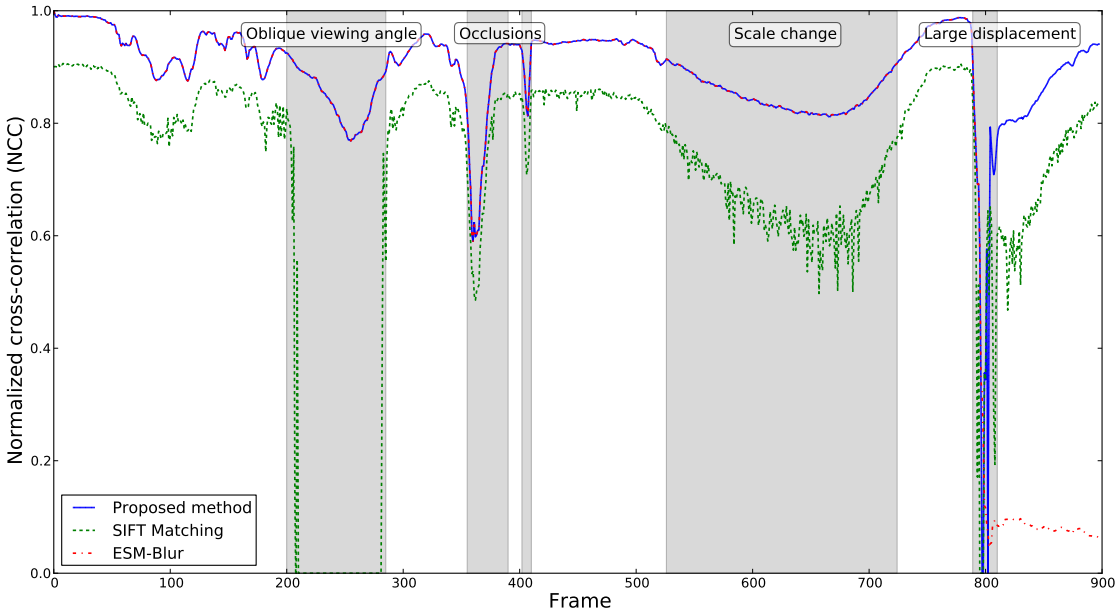


Figure 5.3: Hybrid system performance in the test dataset

Table 5.2 compares the means and medians of the NCC value for the whole sequence in our dataset. It is clear that our proposed method improves significantly the accuracy of tracking for the dataset tested. The difference between our hybrid approach and other methods could be even bigger if the sequence had more large displacements and oblique viewing, the problems that ESM and SIFT Matching (respectively) cannot handle very well.

However, both the mean and the median values are corrupted by outliers: the frames where the tracker is lost. To further evaluate the results, the percentage of good poses was computed for all the methods. As before, the definition is that a good pose is obtained when the computed NCC value is above a given threshold. In Figure 5.4, this metric is applied for several methods and for different thresholds.

The robustness of our method makes it suitable for real world applications. The method was used to place virtual objects in the target object in Augmented Reality application. Screenshots of final system are shown in Figure 5.5. The performance of the system achieves (in average) 30 frames per second.

Method	Mean NCC value	Median
SIFT Matching	0.70	0.79
KLT Tracker	0.17	0.05
Particle Filter (SIFT)	0.42	0.46
IC	0.75	0.86
ESM-Blur	0.80	0.90
Proposed Method	**0.89**	**0.90**

Table 5.2: NCC average (and median value) computed for the whole sequence of the dataset.

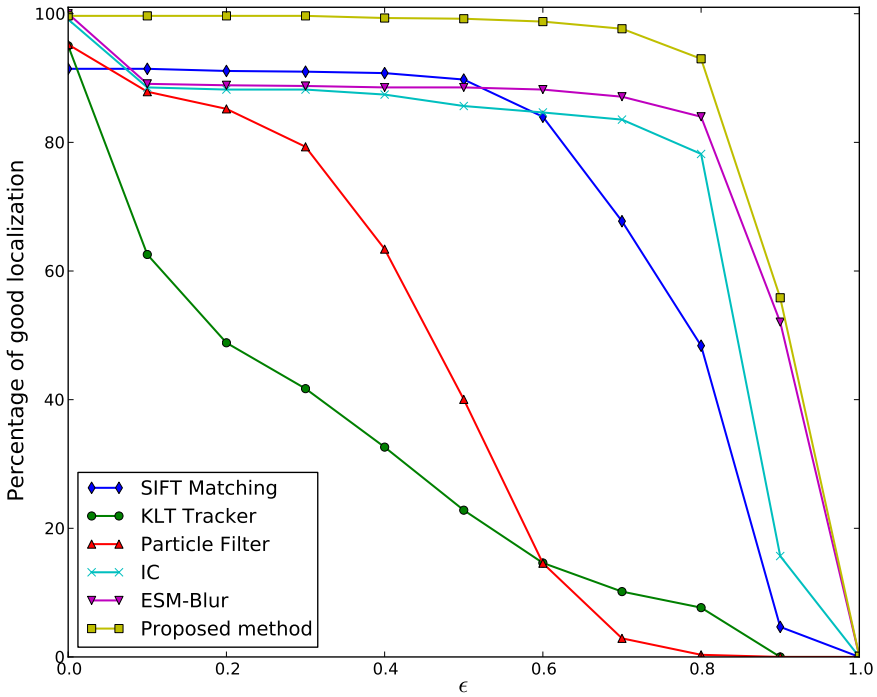


Figure 5.4: Pecentage of good localization for several methods. Here, good localization is defined as the number of frames with a NCC value greater than the threshold ϵ.

(a) Teapot object (b) A 3D avatar of the author

Figure 5.5: Screenshots of the final system with augmented objects

6 CONCLUSION

The 3D tracking literature is massive because of the many applications that use tracking and the variety of solutions to the problem. All these approaches can be separated into two categories, namely, tracking-by-detection and recursive tracking, based on the independence of the current frame estimation.

Tracking-by-detection methods search in the whole frame for an object's pattern that was learned in an offline stage. Each frame is processed individually, such that, the information of the previous camera poses is not used. This aspect leads to a good recovery when the system is lost, due to a large movement, severe blur or a complete occlusion.

The tracking-by-detection method using SIFT features shows good results, dealing well with partial occlusions and large displacements, but when frames present blur or the target object is too far from the camera, the matching of natural features is compromised. The lack of 3D-2D correpondences leads to a low quality pose estimation and can cause jitter.

Traditional tracking methods, estimate recursively the object pose, searching nearby previous poses for the current object's position and orientation. This characteristic can simplify the tracking and increase temporal consistence, but comes with a price. In such systems, a manual initialization is often required and if the system loses the target object pose, e.g. when a large movement occurs, it cannot recover the tracking and a reinitialization must be done. In our work, three recursive methods were studied: KLT, Particle Filtering and Template-based.

The use of a KLT-like tracker can result good localization when the camera movement is smooth, but fails otherwise, due to the assumption in such systems that the inter-frame displacement will remain small. Moreover, the approach assumes that the image brightness is constant along the frames and that, in most formulations, the interest point deformation is an affine transformation, which can be false for 3D tracking.

Particle filters provide a framework for camera pose estimation using different types of measurements. With the types of measurement studied, even when partial occlusions occurred, the system gives good estimates, but it shows jitter. Also, in frames which the measure is weak or not available, the technique cannot well evaluate the particles, leading to poor estimations. Moreover, if the displacement between two frames is expected to be large, as in our application, the particles must be spread in a larger part of the state space of possible camera's poses and therefore, a high number of particles is required to improve accuracy; the higher number of particles creates a cost in the computation, such that, in many particle filtering systems the real-time performance cannot be achieved.

The latter recursive method shows the best results. Template-based methods can work well in the presence of blur (Park et al., 2009) or large illumination changes (Silveira and Malis, 2007). Additionally, our set of experiments shows that when the object is viewed in oblique angles, the tracking accuracy remains high, specially when using the Efficient Second-order Minimization (ESM) method. Even so, occlusions are difficult to handle and these techniques suffer from the same problems of recursive methods discussed.

The good performance of the ESM method and its inability to recover after the lost of the object pose, suggests that the method, to be a complete solution, must be coupled with a tracking-by-detection technique that performs a re-initialization of the ESM algorithm whenever the tracking is lost. The hybrid approach proposed in our work, show great improvements in the tracking performance. Additionally, optimizations on GPU were performed in its implementation to achieve real-time performance.

REFERENCES

Ababsa, F. and Mallem, M. (2011). Robust camera pose tracking for augmented reality using particle filtering framework. *Machine Vision and Applications*, 22:181–195.

Altmann, S. L. (1986). *Rotations, quaternions, and double groups*. Clarendon Press.

Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1090–1097.

Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255.

Benhimane, S. and Malis, E. (2007). Homography-based 2d visual tracking and servoing. *International Journal of Robotics Research*, 26(7):661–676.

Choi, C., Baek, S., and Lee, S. (2008). Real-time 3D object pose estimation and tracking for natural landmark based visual servo. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 22–26.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1052–1067.

Deutscher, J. and Reid, I. (2005). Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61:185–205.

Feng, Z., Duh, H. B.-L., and Billinghurst, M. (2008). Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 193–202.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Haug, A. J. (2005). A tutorial on bayesian estimation and tracking techniques applicable to nonlinear and non-gaussian processes. Mitre technical report, The MITRE Corporation.

Jurie, F. and Dhome, M. (2001). A simple and efficient template matching algorithm. In *International Conference in Computer Vision*, pages 544–549.

Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K., and Tachibana, K. (2000). Virtual object manipulation on a table-top ar environment. *IEEE/ACM International Symposium on Augmented Reality*, pages 111–119.

Kragic, D. and Kyrki, V. (2006). Initialization and system modeling in 3d pose tracking. In *International Conference on Pattern Recognition*, pages 643–646.

Ladikos, E., Benhimane, S., and Navab, N. (2007). A realtime tracking system combining template-based and feature-based approaches. In *International Conference on Computer Vision Theory and Applications*, volume 2, pages 325–332.

Lepetit, V. and Fua, P. (2005). Monocular model-based 3D tracking of rigid objects: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 1:1–89.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157. IEEE Computer Society.

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679.

Malis, E. (2004). Improving vision-based control using efficient second-order minimization techniques. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1843–1848.

Malis, E. and Vargas, M. (2007). Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA.

Marimon, D. and Ebrahimi, T. (2007). Combination of video-based camera trackers using a dynamically adapted particle filter. In *International Conference on Computer Vision Theory and Applications*, pages 363–370.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630.

Park, Y., Lepetit, V., and Woo, W. (2009). ESM-Blur: Handling & rendering blur in 3D tracking and augmentation. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 163–166.

Pupilli, M. and Calway, A. (2005). Real-time camera tracking using a particle filter. In *Procedings British Machine Vision Conference*, pages 519–528.

Pupilli, M. and Calway, A. (2006). Particle filtering for robust single camera localisation. In *First International Workshop on Mobile Vision*, pages 1–14.

Schreiber, D. (2007). Robust template tracking with drift correction. *Pattern Recognition Letters*, 28(12):1483–1491.

Shi, J. and Tomasi, C. (1993). Good features to track. Technical report.

Silveira, G. and Malis, E. (2007). Real-time visual tracking under arbitrary illumination changes. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–6.

Silveira, G. and Malis, E. (2010). Unified direct visual tracking of rigid and deformable surfaces under generic illumination changes in grayscale and color images. *International Journal of Computer Vision*, 89(1):84–105.

Sinha, S. N., Frahm, J., Pollefeys, M., and Genc, Y. (2006). GPU-based video feature tracking and matching. Technical report, In Workshop on Edge Computing Using New Commodity Architectures.

Skrypnyk, I. and Lowe, D. G. (2004). Scene modelling, recognition and tracking with invariant image features. In *IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 110–119.

Tao, G., Lijun, L., Wei, L., and Cheng, W. (2009). Registration using adaptive particle filter and natural features matching techniques for augmented reality systems. *Assembly Automation*, 29:75–84.

Welch, G. and Bishop, G. (1995). An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, Department of Computer Science.

Zhang, Z. (2000). A flexible new technique for camera calibration. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 1330–1334.