Federal University of Pernambuco
Computer Science Center

Post-graduation in Computer Science

ANALYSIS AND EVALUATION OF OPTIMIZATION TECHNIQUES FOR TRACKING IN AUGMENTED REALITY APPLICATIONS

João Marcelo Xavier Natário Teixeira

PHD THESIS

Recife
March, 2013

Federal University of Pernambuco
Computer Science Center

João Marcelo Xavier Natário Teixeira

ANALYSIS AND EVALUATION OF OPTIMIZATION TECHNIQUES FOR TRACKING IN AUGMENTED REALITY APPLICATIONS

Thesis presented to the Computer Science Post-Graduation Program of the Federal University of Pernambuco in partial fulfillment of the requirements for the Degree of PhD in Computer Science.

Advisor: Judith Kelner
Co-advisor: Veronica Teichrieb

Recife
March, 2013

Catalogação na fonte

Bibliotecária Jane Souto Maior, CRB4-571

Teixeira, João Marcelo Xavier Natário
Analysis and evaluation of optimization techniques for tracking in augmented reality applications / João Marcelo Xavier Natário Teixeira. - Recife: O Autor, 2013.
93 f.: il., fig., tab.

Orientador: Judith Kelner.
Tese (doutorado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui bibliografia.

1. Ciência da computação. 2. Visão computacional. I. Processamento de imagem. I. Kelner, Judith (orientadora). II. Título.

004 CDD (23. ed.) MEI2013 – 089

Tese de Doutorado apresentada por **João Marcelo Xavier Natário Teixeira** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**ANALYSIS AND EVALUATION OF OPTIMIZATION TECHNIQUES FOR TRACKING IN AUGMENTED REALITY APPLICATIONS**" orientada pela Profa. Judith Kelner e aprovada pela Banca Examinadora formada pelos professores:

___

Prof. Rafael Dueire Lins
Centro de Informática / UFPE

___

Prof. Silvio de Barros Melo
Centro de Informática / UFPE

___

Prof. Breno Ramos Sampaio
Departamento de Economia / UFPE

___

Prof. Eduardo Simões de Albuquerque
Instituto de Informática / UFG

___

Prof. Luciano Pereira Soares
Departamento de Informática / PUC/RJ

Visto e permitida a impressão.
Recife, 1 de março de 2013.

___

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco

RESUMO

Aplicações de visão computacional em tempo real executadas em campo fazem uso frequente de computadores vestíveis, os quais apresentam uma restrição crítica na quantidade de processamento que podem suportar, uma vez que a maior parte da aplicação (se não sua totalidade) deverá executar na plataforma vestível. É fundamental um esquema de balanceamento de carga capaz de permitir que a aplicação utilize mais poder de processamento quando os cenários de entrada apresentam mais restrições visuais, por exemplo, referentes ao objeto a ser rastreado, e diminua tal processamento com o objetivo de economizar bateria e tempo de CPU em aplicações quando o vídeo capturado é mais controlado (mais acessível).

O fato de aplicações de visão computacional executarem em uma variedade de plataformas justifica se definir um modelo que ajuste automaticamente o rastreador em uso em aplicações com restrições de recursos computacionais. A degradação de desempenho em plataformas vestíveis pode ser maior do que a esperada, uma vez que plataformas desktop e móvel apresentam diferentes níveis de configurações de hardware, e consequentemente, diferentes restrições de desempenho.

Esta tese de doutorado soluciona o problema do rastreamento de objetos usando um modelo de decisão, objetivando utilizar o algoritmo menos custoso sempre que possível. Como objetivos específicos têm-se: investigar e implementar diferentes técnicas de rastreamento, para escolher/definir uma métrica de referência que possa ser usada para detectar interferência na imagem (oclusão, ruído, etc.), propor um modelo de decisão que permita chaveamento automático entre diferentes rastreadores visando balancear o desempenho da aplicação baseado na métrica escolhida, e diminuir a quantidade de processamento requerida pela aplicação sem comprometer a qualidade do rastreamento envolvido.

A eficiência do sistema será verificada através de estudos de caso sintéticos que compreendem diferentes classes de objetos que podem ser rastreados, focando em aplicações de realidade aumentada que executam em plataformas vestíveis. Diferentes algoritmos de rastreamento farão parte do modelo de decisão e através do chaveamento entre eles, será demonstrado que é possível atingir uma melhoria no desempenho de até três vezes, mantendo uma qualidade mínima definida como erro de reprojeção de até 10 pixels quando comparado à utilização apenas do algoritmo que gera a melhor qualidade de rastreamento, independente do seu custo computacional. O impacto desse trabalho implicará em uma melhor qualidade de aplicações com restrições de quantidade de memória, carga de baterias, entre outras.

Palavras-chave: Realidade Aumentada, Visão Computacional, Alto Desempenho.

ABSTRACT

Real-time computer vision applications that run on the field and make frequent use of wearable computers have a critical restriction on the amount of processing they can perform, because of the fact that most (if not all) of the application runs on the wearable platform. A balancing scheme capable of allowing the application to use more processing power is fundamental both when input scenarios present more visual restrictions regarding, for example, the object to be tracked, and also to reduce processing in order to save battery and CPU time for other applications when the captured video is better controlled (more accessible).

The fact that computer vision applications may run on a variety of platforms justifies the need for defining a model that automatically adjusts the tracker being used in applications with hard performance constraints. Performance degradation in wearable platforms can be greater than expected, as desktop and mobile platforms present different levels of hardware capabilities, and consequently, different performance restrictions.

This doctoral thesis addresses the object tracking problem using a decision model, in such a way that prioritizes using the least computationally intensive algorithm whenever possible. It has the following specific objectives: to investigate and implement different tracking techniques, to choose/define a reference metric that can be used to detect image interference (occlusion, image noise, etc.), to propose a decision model that allows automatic switching of different trackers in order to balance the application's performance, and to reduce the application's workload without compromising tracking quality.

The effectiveness of the system will be verified by synthetic case studies that comprise different object classes that can be tracked, focusing on augmented reality applications that can run on wearable platforms. Different tracking algorithms will be part of the proposed decision model. It will be shown that by switching among these algorithms, it is possible to reach a performance improvement of a factor of three, while keeping a minimum quality defined by a reprojection error of 10 pixels when compared to the use of only the best algorithm independent of its computational cost. This work results in better performance of applications with memory and battery restrictions.

Keywords: Augmented Reality, Computer Vision, High Performance.

ACKNOWLEDGEMENTS

To God, thanks for all the blessings that allowed me to be a better person every day.

To my parents and family, thanks for all the support.

To my grandparents, thanks for still teaching me important life lessons.

To Alline, thanks for all patience, love and support over all these years.

To JK, thanks for being responsible for what I am today regarding my academic life. You know you can always count on me.

To VT, thanks for making me keep focus on the important things. Also, thanks for being a model to me, even in distance. I wish one day I can the be half of what you are.

To Javier, thanks for all the support while I was in Chile. Also, thanks to the friends from UChile, specially Patricio, Marcelo, Jaime, Francisco and Wilma.

Thanks to my colleagues from GRVM, specially the “old school” friends Bernardo, Daliton, Guilherme, Saulo, Gabriel, Crystian and Mouse.

Thanks to CAPES and CNPq for partially funding this research.

CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF TABLES

CHAPTER 1 INTRODUCTION

The term “vision” does not simply refer to the ability of capturing visual information, but also refers to the possibility of understanding content being viewed. It is not enough to see the environment; one must understand it. The information that comes from vision can be used to perform, among others, the acts of positioning and detection. Vision is only relevant when those actions are achieved. For example, one can imagine a scenario containing a corridor with a transparent glass wall that blocks the way of a person. The glass, depending on the point of view, may not be noticed due to its level of transparency. In this case, vision fails, since it was not possible to generate enough information about the environment that would allow for the viewer to change position. Another interesting example is a mirror maze - the excess of visual information results in a loss of spatial reference, and, accordingly, mobility becomes a challenge.

Computer vision seeks to understand the environment, with the analogous objective of using the acquired information to guide user actions[1]. One way of understanding the environment is to analyze the objects inserted in it and how they move. Through observation of objects, their location and their behavior, it is possible to know what happens and which action to take at a specific moment. This is achieved by the use of object trackers, which are capable of following the movement of an object even when it is not entirely visible to the capture sensor[2].

There is a large number of object tracking algorithms. The basic differences between them resides in their precision, performance, and applicability. Some trackers have particularities and are intended to be used in specific controlled environments with low noise rate, uniform illumination, etc. Other tracking algorithms are more precise and more robust to occlusions and extra interferences on the tracking process, therefore demanding more computational effort.

1.1 STATEMENT OF THE PROBLEM

Despite the large number of tracking techniques found in the literature, it is not clear that a tracker exists that provides the best performance ratio in many different scenarios while still maintaining a pre-defined tracking quality. Trackers already exist that are better applied to specific cases[3][4][5]. Depending on the tracker chosen, the application (the system as a whole) can perform poorly under certain conditions. Applications that show strong computational power limitations work with fixed restrictions on tracking precision and processing speed[96]. At present, to the best knowledge of the author, there is no real-time balancing scheme of instantaneous application needs, and no further analysis of the control factor for tracking precision and performance ratio of the application.

Real-time computer vision applications that run on the field, frequently making use of wearable computers, have a critical restriction on the amount of processing they can perform. This is generally due to the fact that most parts of the application run on the wearable platform[97][98]. It is fundamental a balancing scheme capable of allowing the application to use more processing power when input scenarios present more visual restrictions regarding the object to be tracked. By diminishing such processing will allow the solution to save battery and CPU time for other applications when the captured video is more controlled (more accessible).

1.2 MOTIVATION

The nature of this thesis is directed to planar object tracking algorithms and how they can be applied to systems with critical processing power restrictions. To "track an object" means to follow its movement, or in other words, to be able to identify its position and orientation in a specific image frame. This objective may be reached using one of two approaches: either by following the object from the starting frame to the current one or by searching the entire current frame for the object. The first case represents pure object tracking. The second case uses object detection to perform object tracking. Pure object tracking restricts the search area, minimizing the effort needed for finding the object over the image. Pure object tracking cannot support wide-baseline displacements, which can be necessary in certain scenarios. Object detection does not present such limitation, but presents the onus of having to search a broader area of the image and using other image matching techniques for finding the target object[1].

It is said that texture-based tracking has good quality when it is capable of mapping the template image to the one given as input in a coherent way, in other words, both are superimposed accordingly. This means that the tracking algorithm can find a transformation that performs a satisfactory geometric alignment between the template used as reference and the object from the input image[6].

This research considers improving the tracking process by using a decision model in a way that the goal is to minimize the computational load while maintaining the tracking quality. The creation of a decision model involves the selection of both input parameters that will impact the decision and also a utility function capable of measuring how well it performs[7]. This work intends to analyze different planar object and image characteristics in order to extract the parameters to be used in the proposed model and to develop different error measures that can be used as a utility function. We also intend to find methods to diminish or balance the workload in computer vision applications running over wearable platforms.

Therefore, this thesis has the following specific objectives:

- To investigate and implement different techniques applied to texture-based tracking of planar objects;
- To choose/define a reference metric that can be used to detect image interference: occlusion, image noise, and others;
- To create a decision model that allows automatic switching of different planar object trackers in order to balance the performance of the application based on the chosen metric;
- To diminish the workload of the application without compromising tracking quality.

1.3 RELEVANCY

The fact that computer vision applications run on a variety of platforms justifies the need for defining a model that automatically adjusts the tracker to use with applications with hard performance constraints. Performance degradation in wearable platforms can be greater than expected. For instance, desktop and mobile platforms present different levels of hardware capabilities, and consequently, different performance restrictions.

Whenever possible, the decision model should choose the simplest tracker given that the quality (with the threshold specified by the developer) is not harmed. It will be shown that every time quality cannot be assured, the decision model selects the algorithm with highest probability of tracking the object (usually the most complex one).

The tracking phase is one of the most performance demanding phases in computer vision applications. Because of that, a solution that saves processing time, guaranteeing quality of application (through the preservation of the tracking quality), can allow the execution of computer vision applications in restricted platforms, such as wearable computers.

1.4 RESEARCH DESIGN

AR applications work through the understanding of the environment and later addition of content located coherently in space. Such understanding is done through the use of image processing and computer vision techniques to recognize the template information used as reference to position the virtual content. This processing phase can be divided into two separate steps: detection and tracking. Once the object is detected, a tracking algorithm can be applied in order to follow the object throughout the scene, without having to detect it again. Both detection and tracking algorithms usually demand a high computational cost, and should be chosen and implemented carefully in order to take advantage of all available computational resources.

The problem of lack of resources is magnified when the goal is to develop AR applications that have mobility as a requirement. In this case, the user must use the application on the field through some mobile platform, such as a wearable computer, which has hard energy consumption and computational power restrictions. A great challenge in this area is how to maximize the available resources in a way that allows for a better experience with the application.

A solution to this problem is the modification or construction of a "scheduling" scheme for the tracking algorithms. Without the need to modify existing tracking algorithms, the idea is to define comparison criteria in order to enable their switching according to the input scenario in which the application is inserted. This way, simply by adequately choosing which tracking algorithm to use in real time, it is possible to obtain better application efficiency.

The proposed methodology used in this work is comprised of the following phases:

- Studying of the existing different object tracking approaches (advantages, limitations, scenarios);
- Definition of comparison criteria in order to enable tracking algorithms switching according to the input scenario;
- Analysis of different warp functions;
- Definition of the test scenarios (occlusion and scene luminance variations, camera movement);
- Selection of different object trackers;
- Definition of a tracker-switching decision model;
- Validation of the decision model with training and test scenarios.

Chapter 2 defines the context in which this work is inserted into. Chapter 3 consists of studying the existing different object tracking approaches. Chapter 4 refers to comparison criteria that enable tracking switching according to the input scenario and also examines how speedup and error measurements are calculated in this work.

After the study of tracking techniques, the problem of choosing the best suited tracking algorithm was defined as problem of optimization, mainly based on two components: warp function, which maps to the object movement over the image (it includes its rotation, scale and deformation), and visibility aspects of the object, which includes luminance variations and occlusions by the scene the object is inserted into. Different warp functions were analyzed in chapter 5 so that it was possible to verify what information each component provided to the application and its adequacy for generic AR applications.

The application scenarios for the proposed approach include image sequences commonly found in AR applications, i.e., images that have some type of object susceptible to being tracked as content and the accompanying problems regarding their correct tracking. A study of the problems that eventually hinder tracking performance showed that the two most important issues are occlusion and scene luminance variations. Based on this information, the test scenarios were defined in chapter 6. The selection of input data for the tests should parallel characteristics commonly found on real-world scenarios, which are: occlusion between objects, luminance variance, and camera movement. Different image sequences varying the levels of occlusion and luminance will also be tested, due to their influence on the change of tracker to be used. The data used in both training and test phases will come from public image datasets widely available and used by the scientific community, which allows the repeatability of this work.

In chapter 7, different combinations of feature detectors and extractors are selected, leading to the selection of tracking options to be used while the application is executed. The switching between different trackers should be defined by a metric capable of pointing the most adequate tracker given the information possessed up to that moment.

Chapter 8 takes into consideration the trackers and the metrics defined for comparing them and defines a tracker-switching decision model for automatically choosing the most adequate tracker at a given moment.

Chapter 9 validates the proposed work by applying the designed model to both training and test scenarios, and finally, chapter 10 lists the contributions and directions for future work.

CHAPTER 2 STATE OF THE ART AND RESEARCH DESIGN

This chapter describes how computer vision algorithms are applied to augmented reality applications and the different tracking approaches used to perform visual tracking. In sequence, a brief overview is given about a technique used for determining the best algorithm for feature selection, focusing on tracking quality. The major differences between this method and the technique proposed in this thesis will be highlighted. This chapter also provides the reader with an introduction to Decision Theory, a concept widely used on the conception of the proposed model.

2.1 VISUAL TRACKING

Tracking an object of interest in a video sequence means continuously identifying its position and orientation despite movement of either the camera or the object [8][10]. A variety of approaches can be used for this task depending on the type of object being tracked and the degrees of freedom inherent to the object and camera [8]. Figure 1 illustrates the accurate tracking of a face despite its level of occlusion by another object that is not of interest for the tracking.

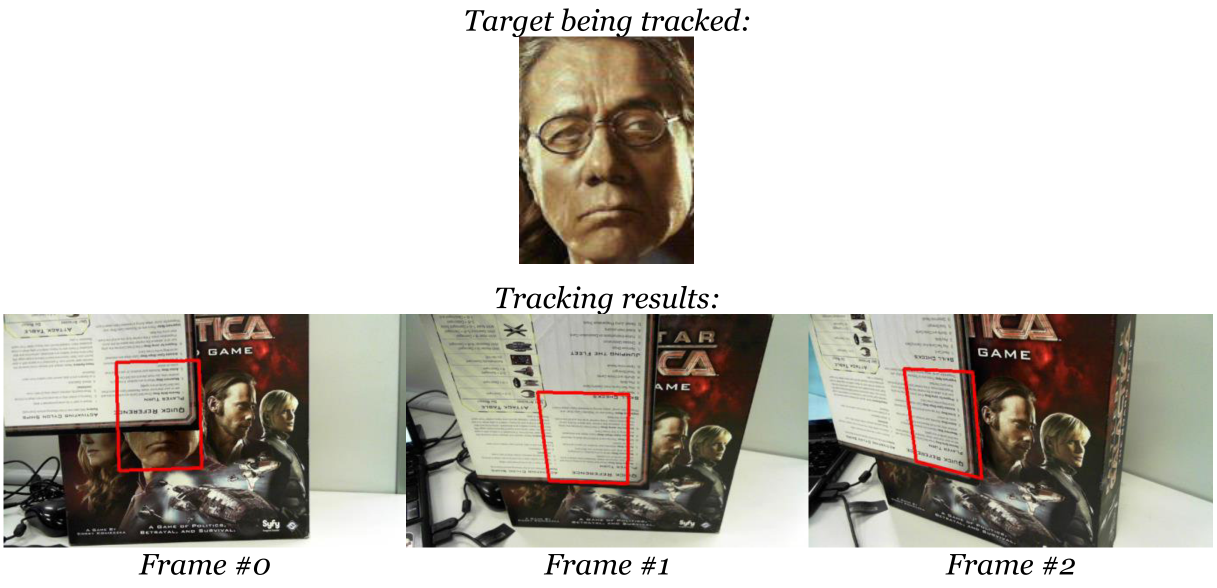


Figure 1. Example of an accurate object tracking robust to occlusion [11].

When 2D tracking is used, the goal is to retrieve a 2D transformation from the object projected on the captured image that best represents the motion that occurred [9]. Many models can be applied in order to handle appearance changes due to perspective effects or deformations. The homography is one of the most used transformations regarding 2D tracking of planar objects, since it is generic enough to deal with all possible perspective effects [12]. This thesis focuses on tracking of planar objects since it serves as basis for more complex tracking approaches (for example, 3D tracking). As it will be shown in section 5.2, almost every object can be considered locally planar. Alternatively to planar objects, deformable templates are one approach that can be used in order to track deformable meshes [13], as shown in Figure 2.

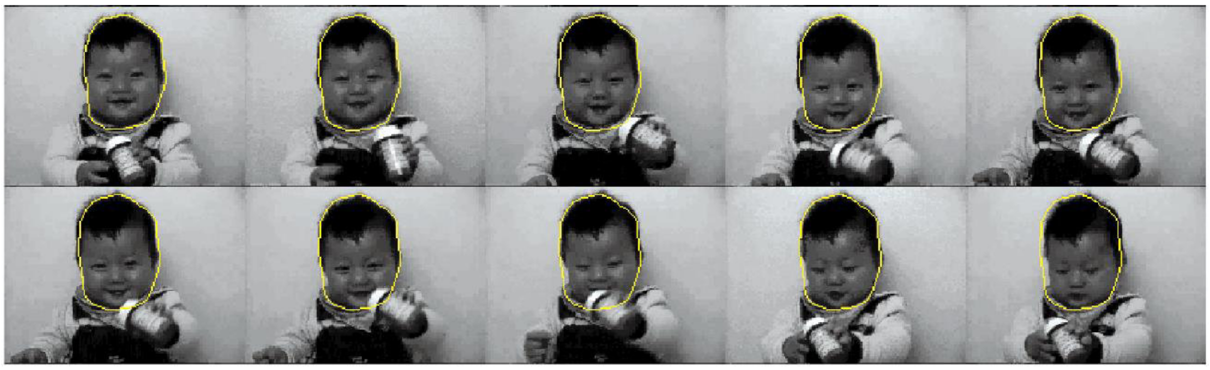


Figure 2. Face tracking using a deformable template [13].

In computer vision, general video analysis can be broken down into three different steps: detection of interesting moving objects, frame to frame tracking of those objects and analysis of their tracks to recognize possible behaviors [2]. Because of such use in computer vision, object tracking is important in the following major areas:

- 3D reconstruction [99];
- motion-based recognition [14];
- automated surveillance [15];
- video indexing [16];
- human-computer interaction [17];
- traffic monitoring [18];
- vehicle navigation [19].

Figure 3 illustrates examples of computer vision applications in each of the previously described areas. Besides these areas, object tracking algorithms can also be used in Augmented Reality systems that require real-time registration of the object to be augmented.

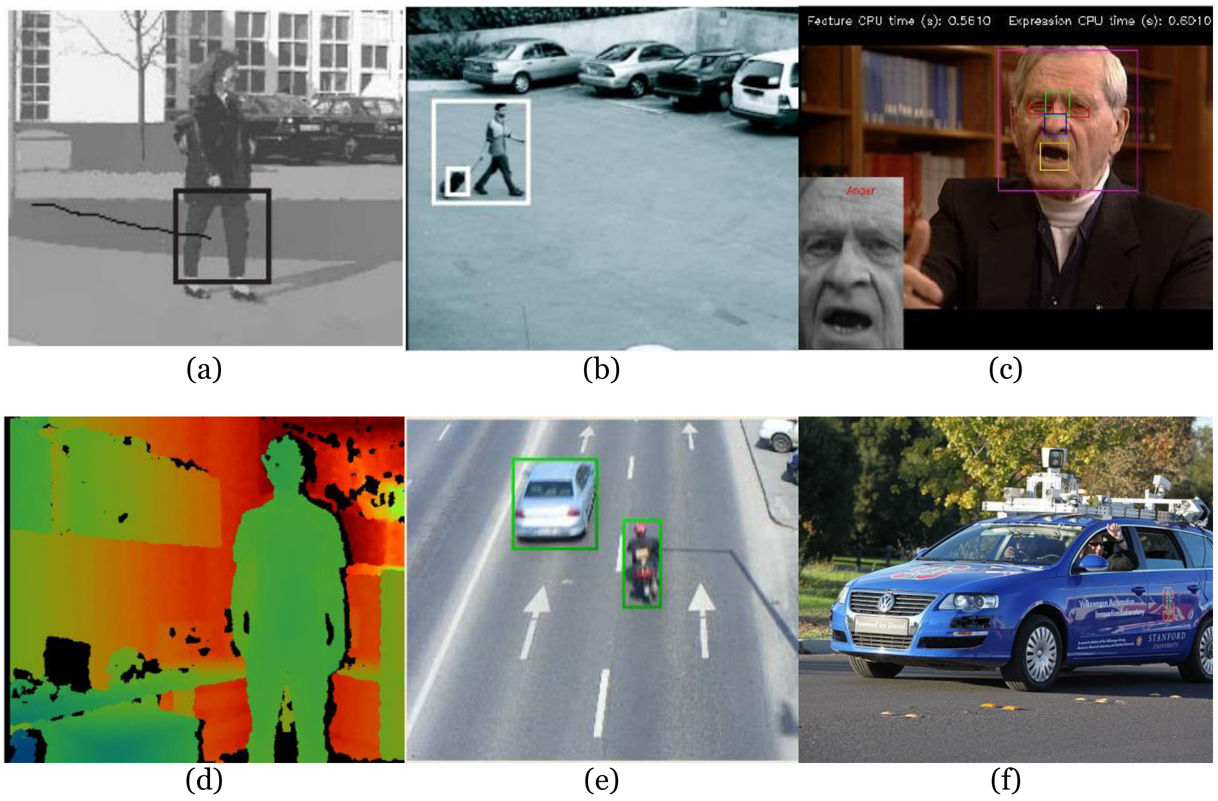

Feature CPU time (s): 0.5610 Expression CPU time (s): 0.6010
Anger
(a)
(b)
(c)
(d)
(e)
(f)

Figure 3. Computer vision applications: a) motion-based recognition [20]; b) automated surveillance [15]; c) video indexing [21]; d) human-computer interaction [22]; e) traffic monitoring [23]; f) vehicle navigation [24].

2.1.1 TRACKING IN AUGMENTED REALITY APPLICATIONS

Augmented Reality (AR) applications involve superimposing computer-generated content on real scenes in real-time [25]. Tracking is a critical component of most AR applications, since the objects in both real and virtual worlds must be properly aligned with respect to each other in order to preserve the idea of the two worlds coexisting.

Medical visualization [26], maintenance and repair [27][100], annotation [28], entertainment [29], navigation and targeting [30] and collaboration [101] are some examples of areas that were already explored by AR applications.

Traditionally used in AR, vision-based tracking described as two main steps: extract information from the input video using image processing algorithms and perform the pose estimation itself, or, in other words, find the transformation that best maps the object model from one frame to the next one.

The information that comes from the input image is basically composed of image features that are easy to extract from the scene. Both Marker-based AR and Markerless AR [33] are based on this principle, with the difference that the first one adds some templates that do not originally belong in the scene in order to enable the tracking. These templates are often called fiducials and they constitute image features easy to extract as well as reliable measurements for the pose estimation. In the case of Markerless AR, the template to be used is the object to be tracked itself, using its natural features. Because of that, not every object can be tracked easily, due to the fact that its texture should have a certain level of differentiability compared to the rest of the scene [32].

Other tracking approaches can be used as well, such as approaches that rely on edge information [33], feature information [34] and those that do not consider any object model *a priori*[8][35]. This thesis will provide more detail regarding the template-based approaches.

The template-based tracking techniques do not necessarily rely on local features such as edges or other features, but rather on global region tracking through the use of the whole pattern of the object to be tracked. These methods can be useful in handling more complex objects that are difficult to model using local features due to their repeatability, for example. Such scenarios can be computationally expensive, but in some cases effectively formulated [2].

2.2 DECISION THEORY

Making decisions in face of uncertainty is part of our lives. In economics, psychology, philosophy, mathematics, statistics, among other areas, one must act without knowing the consequences from the action taken. In order to deal with these problems on a rational basis, a theoretical structure for decision making that includes uncertainty must be used. Decision theory provides such structure, in the form of a framework in which all available information is used to deduce which of the decision alternatives is best according to the decision maker's preferences [7]. It distinguishes between decisions and outcomes. Not every time a good decision results in a good outcome. The procedure that comes from decision theory will allow us to obtain the best possible logical solution. In other words, it will minimize the consequences of getting an unfavorable outcome.

Common sense can be formalized by decision theory. It can be represented in an unambiguous way when a mathematical language is used. Two streams of thought serve as the foundations of decision theory: utility theory, which focuses on values, and probability theory, which focuses on information. This work will deal with the decision making problem as an instance of the first one, utility theory [92].

The first step for structuring a decision making process is to define numerical values to the possible outcomes, which falls within the area covered by modern utility theory. The basic assumption to be made is that two possible outcomes can be compared. This way, preference over outcomes can be stated and one of the possible paths to the solution can be elected. A common tool from utility theory, the utility function [93], provides a means of consistently describing the decision maker’s preferences through a scale of real numbers. It is no more than a means to logical deduction based on the preferences regarding the problem.

CHAPTER 3 STUDY OF EXISTING TRACKING APPROACHES

Tracking is a difficult computer vision task due to the potential variability of the images of the object that is to be tracked over time. Such variability arises from three main causes: variation in target pose or deformations, illumination changes and target occlusion[102]. Thus, the two main challenges for visual tracking are to develop accurate models of image variability and also to design computationally efficient algorithms that use those models [44]. This chapter focuses on studying some of the existing tracking approaches used in AR applications.

3.1 DIFFERENT TRACKING APPROACHES

Numerous approaches to object tracking have been proposed over the years. They primarily differ from each other based on object representation, image features or motion model [2]. This section will briefly introduce different major tracking techniques for a variety of scenarios.

3.1.1 MEAN SHIFT

Comaniciu et al. [36] introduced an efficient tracking scheme for nonrigid objects. The target model is represented by an ellipsoidal region in the image. The influence of different target dimensions is eliminated by normalizing targets to a unit circle. Since the target model comprehends a probability distribution function in the feature space, an isotropic kernel is used in order to assign different weights to the pixels, being the smallest one the farthest away from the center. This way, the robustness of the density estimation is increased, as the peripheral pixels are least reliable due to often being affected by occlusions or interference from the background.

The Mean Shift tracker maximizes the appearance similarity iteratively, which is based on the Bhattacharyya coefficient[42], by comparing the histograms of the object and the window around the hypothesized object location. For each iteration, a Mean Shift vector is computed such that the histogram similarity is increased. The process is increased until convergence is achieved.

An obvious advantage of the Mean Shift tracker over a standard template matching approach is the elimination of a brute force search and the computation of the translation of the object path (plus scale) in a small number of iterations. The main disadvantage is that it cannot compute the entire object pose – only a simple motion model (translation and scale) is supported.

3.1.2 Camshift

The Camshift (Continuously Adaptive Mean Shift Algorithm) [37] appeared as an improvement to the original Mean Shift algorithm. Since the Mean Shift operates on probability distributions, color histograms are used. The problem with the original approach is that color distributions derived from video image sequences change over time, so the algorithm must be modified to adapt dynamically to the probability distribution it is tracking.

Instead of a set or externally adapted window size, Camshift relies on the zeroth moment information, which is the distribution area found under the search window. Thus, window radius is set to a function of the zeroth moment found during search. The problem with a fixed window size is that a value that works at one distribution scale is not suitable for another scale as the object moves closer to or farther from the camera. While small fixed-size windows may get lost entirely for large object translation on the scene, large fixed-size windows may include distractors (objects that should not be tracked) and/or too much noise.

One of the disadvantages of Camshift is that it relies on color distributions alone, so that errors in color (colored lighting, extremes in illumination) could cause errors in tracking. A solution to this problem could be to adopt other approaches such as feature tracking and motion analysis, which would add more complexity to the algorithm.

3.1.3 Kalman Filter

The Kalman Filter is a predictor-corrector type estimator that minimizes the error covariance of the system [38][39]. It has been used extensively for tracking in interactive computer graphics, and has two major characteristics: it maintains both the mean vector for the state and also the covariance matrix of the uncertainty state.

The process is estimated using a form of feedback control: the filter estimates the process state at a certain time and then gathers feedback in the form of measurements. Kalman Filter equations are divided into two groups: time update and measure update equations. The first group of equations is responsible for projecting forward both the current state and error covariance while the second is responsible for the feedback.

The discrete Kalman Filter time update equations are described as follows:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{1}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2}$$

Where $\hat{x}_k^-$ represents the a priori state estimate, $\hat{x}_{k-1}$is the a posteriori state estimate from previous state and u_k the control input state. A and B are the matrices that refer to the state and control input state from a previous time step $k-1$ to the current time step, respectively.

P_k^- and P_{k-1} are the a priori estimate error covariance for the current state and the a posteriori estimate error covariance for the previous state, respectively. Q represents the process noise covariance.

In sequence, the discrete Kalman Filter measurement update equations are also listed:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (3)$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H \hat{x}_k^-) \quad (4)$$

$$P_k = (I - K_k H) P_k^- \quad (5)$$

where $K_k$is the Kalman gain or blending factor that minimizes the a posteriori error covariance equation. H is the measurement equation that relates the state to the measurement z_k. R is the measurement noise covariance matrix. I is the identity matrix of corresponding size.

One of the drawbacks of the Kalman Filter is that it performs the probability density propagation for problems in which the system can be described through a linear model and in which both system and measurement noises are white and Gaussian. Most 2D tracking applications involve a nonlinear motion model, which means that the original Kalman Filter implementation cannot be used. In order to solve this problem, an extension of the Kalman Filter (called Extended Kalman Filter, or EKF) can be adopted [40]. The nonlinear functions are linearized using Taylor series expansion. As with the original Kalman Filter, the extended version also assumes that the state is distributed by a Gaussian.

3.1.4 PARTICLE FILTER

The most noticeable limitation from Kalman filtering is the assumption that the state variables follow a Normal (Gaussian) distribution. Consequently, the Kalman filter provides poor estimations for state variables that do not follow a Gaussian distribution [2]. This limitation can be solved by the use of particle filters. With particle filters, the conditional state density is represented at each time t by a set of samples (particles) with different weights (sampling probability), that map the importance of the sample [41].

Particle filtering's most common sampling scheme is the importance sampling method. It is composed of three distinct phases:

- Selection: N random samples are selected;
- Prediction: for each selected sample, a new sample is generated using a zero-mean Gaussian error and a non-negative function;
- Correction: the weights for the new samples are computed using the

measurements found. The probability can be modeled as a Gaussian density.

3.1.5 *HARMONY FILTER*

The Harmony Filter uses the Harmony Search approach in order to optimize the tracking function based on the Bhattacharyya coefficient as distance measure[42][43].

The Harmony Filter uses color histograms to model the targets. As a result, it is possible to define a fitness function by comparing the histogram of the target with other histograms generated from different regions in the image. The histogram with best fitness function value represents the optimal point in the search space, corresponding to the most likely location of the target.

The Harmony Filter uses a two dimensional histogram based on the H (hue) and S (saturation) of the HSV color space. The V (value) channel is discarded. By ignoring the third channel, the filter becomes more robust to changing lighting conditions between frames. It also reduces the overall complexity of the system and consequently increases its performance.

Every time the tracking is required, the harmony memory is initialized using the current frame information along with the target's previous location. The previous location information is used to predict the five-dimensional state vector containing position, velocity, and scale data. A simple motion model is adopted and a new state vector is created by using randomly generated acceleration in the x and y directions. The Harmony Search is then applied until the solution converges or a maximum number of iterations are reached. Trials have demonstrated that better results are obtained using the Harmony Filter in comparison with standard Kalman Filter and Particle Filter approaches [43].

3.1.6 *OPTICAL FLOW*

Optical flow algorithms consider a set of points pertaining to object to be tracked and assume that they move coherently in space. This way, it is possible to estimate both motion and illumination parameters through observations of the image variations. Region-based methods make use of all available image intensity information, eliminating the need to identify and model a special set of features to track. Region-based methods can be considered an effective complement to local feature-based algorithms.

Optical flow can be estimated by different means: phase correlation, block-based methods, differential methods, and discrete optimization methods, among others. A well-known example of differential method for estimating optical flow, based on partial derivatives of the image is the KLT (Kanade Lucas Tomasi) [45], which uses image patches and an affine model for the flow field.

The optical flow computational load is very high and, consequently, a number of precautions should be taken in order to improve performance. One such precaution, for example, is to perform the tracking at multiple levels of resolution. A coarse-to-fine tracking decreases the search area and therefore speeds the tracking stage.

3.1.7 ESM

Many minimization algorithms could be used to estimate the transformation parameters involved in the tracking process. The Newton method has the highest local convergence rate as it is based on a second-order Taylor series of the SSD (Sum of Squared Differences) with the disadvantage of being time consuming. Another disadvantage is that if the Hessian is not positive definite, convergence problems can occur.

The efficient second-order minimization tracker (ESM) [46] solves both speed and convergence problems [47]. It has a high convergence rate compared to the Newton method without the need to compute the Hessian.

The ESM can be applied in tracking a planar object using, for instance, a homography H. A total of 8 parameters are needed to define this warp function up to a scale factor. The homography parameterization is done by choosing its determinant to be equal to one. An additional constraint is that H is inserted in the Special Linear group of dimension 3, which is the group of 3×3 matrices that have the determinant equal to 1. This parameterization simplifies the computation of the Jacobian, improving ESM's performance.

Another work has extended the ESM tracker to handle motion blur in 3D object tracking [48]. They have introduced an image formation model that considers the possibility of blur and applies such a model in a generalization of the original ESM algorithm. This allows a faster convergence and a more accurate/robust tracking even under a large amount of blur.

3.1.8 MUTUAL INFORMATION BASED TRACKER

Mutual information is an alignment function that was first introduced in the context of information theory [50]. It represents the quantity of information shared between two random variables. If the two variables/images are aligned then their mutual information is maximal. It can be calculated by using the following formulas:

$$MutualInformation(A,B) = \sum_{i,j} p_{AB}(i,j) \times \log\left(\frac{p_{AB}(i,j)}{p_A(i) \times p_B(j)}\right), \\ for\ i,j \in [0, number\ of\ histogram\ bins], \tag{6}$$

with

$$p_I(i) = \frac{1}{N_x}\sum_x \emptyset(i - I(x)) \tag{7}$$

and

$$p_{AB}(i,j) = \frac{1}{N_x}\sum_x \emptyset(i - A(x)) \times \emptyset(j - B(x)), \tag{8}$$

where N_x is the number of pixels in the images.

In [50], a mutual information based tracker is proposed, in which the formulas are applied to grayscale values, such that the color information from the three channels has to be converted to a single gray one. The $\emptyset$ function used is a $\boldsymbol{4th}$-order $\boldsymbol{b}$-spline, in order to smooth the images and get a better alignment.

3.2 Best Algorithm Selection

The MuFeSaC algorithm (Multi-Feature Sample Consensus) works as an adaptive and automatic procedure to choose a reliable feature detector when there is more than one available[49]. Since its goal is to be used with self-localizing mobile platforms, it focuses on finding the best features (feature detector selection) over a video sequence. Moreover, it only considers improving the quality of the detected features, not considering the complexity of the algorithms involved.

MuFeSaC's algorithm pipeline can be divided into 5 different phases. The first phase consists on the individual analysis of the feature detection methods (FD). For each $FD_i = 1, 2, 3, \ldots N$ (being N the number of feature detectors available), this phase starts by extracting features from two successive frames. In sequence, the putative matches are found using proximity and cross correlation. By performing RANSAC and iterating to a convergence, the d-estimated parameters S of model M fitted during the iterations of RANSAC are collected. Then, the d-variate probability distribution B_i based on n ($n > 30$) iterations of parameter estimates collected ($S_1 \ldots S_n$) are estimated.

The second phase consists in calculating the score single feature outlier consensus ($SFOC_i$) using the model selection criterion ($ICOMP$) described in sequence.

$$\begin{aligned} ICOMP &= Lack\ of\ fit\ + Profusion\ of\ uncertainty \\ &= -2\log(Likelihood\ of\ \mu_i) + 2C_1(F^{-1}(\textstyle\sum_i)) \end{aligned} \tag{9}$$

F^{-1} is the inverse Fisher information matrix, μ_i and $\sum_i$ are the maximum likelihood estimates of the mean and covariance computed as the first two moments of B_i. The C_1 measure and the F^{-1} are computed using the two following equations:

$$C_1(F^{-1}(\Sigma_i)) = \frac{s}{2}\log\left[\frac{tr(F^{-1}(\Sigma_i))}{s}\right] - \frac{1}{2}\log|F^{-1}(\Sigma_i)| \tag{10}$$

and

$$F^{-1}(\Sigma_i) = \begin{bmatrix} \Sigma_i & 0 \\ 0 & D^{+}p(\Sigma_i \otimes \Sigma_i)D^{+}p' \end{bmatrix}, \tag{11}$$

with s being the rank of F^{-1}, $|.|$ refers to the determinant and tr refers to the trace of the matrix. $D^{+}p$ is the Moore-Penrose inverse of vectorized Σ_i and $\otimes$ represents the Kronecker product. The C_1 measure for penalizing uncertainty is obtained by maximizing mutual information in d-dimensions.

The third phase is responsible for computing the competing feature consensus score ($CFCS_i$) by evaluating competing distributions B_i for different hypotheses and then choosing the optimal consensus combinatorial cluster among the competing feature detectors. The Akaike information criterion (AIC) is used to score the different hypotheses as follows:

$$\begin{aligned} AIC(\mu_i, \Sigma_i, k) \quad &= -(Likelihood\ of\ feature\ cluster) \\ &\quad + Parameter\ parsimony\ after\ clustering \end{aligned} \tag{12}$$

In the fourth phase, the optimal feature detector is chosen to be the one with minimum$SFOC_i + CFCS_i$. After this phase, the entire process is repeated every k frames, where k can be adapted based on scene complexity (typically a k value of 300 is used).

It can be said that the work proposed in this thesis is an evolution of the MuFeSaC algorithm, in the sense that it also takes information from the scene to automatically decide which tracking algorithm to use. In addition, it guarantees that the chosen algorithm is the least complex one given a certain error threshold.

3.3 DISCUSSION

Some considerations must be made about the trackers listed above. Both Mean Shift and Camshift use a blob of color information to aid in keeping track of the object. It relies on the use of 1D histograms, which lack spatial information. Their use is indicated when the object to be tracked will change drastically over time, such as rotating or deformable objects. These trackers are considered robust against some occlusion but are not as precise in comparison to other approaches, such as Kalman and Particle Filters, for determining specific point positions.

Kalman and Particle Filters provide a precise estimate for the points and objects being tracked. The drawback resides in their recursive nature, which is responsible for causing a problem of error accumulation over time. In order to get better results with the particle filter, in comparison to the Kalman model, it is possible to increase the number of particles used. This results in a higher computational load, however, since the computer must process more hypotheses in every iteration.

The Harmony Filter can converge more accurately and more quickly than the Particle Filter does as it relies heavily on the most current observational information to improve the initial improvisations, in contrast to the Particle Filter that only uses it to score particles without changing their initial position in search space. Its performance directly depends on the objective function complexity, since it makes use of a direct search method (Harmony Search), and can be applied to any optimization problem (linear or nonlinear). The parameters may be tuned in order to speed up the convergence process.

A tracking approach using optical flow information relies on image variations from frame to frame. The variations must be small in order to guarantee that the tracker works well. Depending on how erratic and fast the movement of the target object is, ambiguity can arise. This problem can occur when using textures with high repeatability. Motion cannot be estimated well in textures with areas of poor gradient. Out of all the tracking approaches cited before, the ESM tracker is the one that presents the best results in terms of balance between accuracy and speed.

Significant progress in object tracking has been made in recent years [2]. Different robust trackers were developed and can track objects in real-time for simple and well-defined scenarios. By "well-defined scenarios" we mean that some assumptions are made, such as approximations of the centroid of the object, smoothness of motion, few occluded areas and quasi-constant illumination. Some of the presented trackers fail when used in realistic scenarios, such as automated surveillance, human computer interaction or vehicle navigation, for example, due to either their lack of precision or their computational load. New solutions are continuously being proposed to deal with these problems.

The next great challenge in object tracking development is to create algorithms for tracking objects in unconstrained videos: noisy, compressed, unstructured, and acquired from moving cameras or from multiple views. The use of mutual information is one interesting approach that tackles these problems, since it is capable of finding a relation between multimodal data (data acquired from different sensors, for instance, relating a drawing with a picture or a map with an aerial view of certain location) [50]. This advantage comes with the drawback of high computational cost. The problem now becomes how to balance the use of the most adequate tracking techniques depending on the input scenarios and the platforms' computational power.

CHAPTER 4 DEFINITION OF COMPARISON CRITERIA FOR TRACKING ALGORITHMS SWITCHING

Given the goal of the proposed model of maintaining tracking quality while minimizing processing demand, the following question arises: how to measure tracking quality? This chapter is dedicated to such topic.

4.1 MEASURING TRACKING QUALITY

The simplest approach is to perform a direct comparison of specific points of the template being tracked and the tracked result, given that ground-truth data is available. This can only be used during the model training phase, because at the time of execution there is no ground-truth data to be based on. The first error measure (namely reprojection error) used to evaluate the tracking quality in this work was calculated using the difference between the four template corners projected using the homography found by the algorithm, as follows:

$$Error = \sqrt{\tfrac{1}{4}\textstyle\sum_{j=1}^{4}\left\|x_j - x_j{}^{*}\right\|^2}. \tag{13}$$

In the previous equation, x_j and $x_j{}^{*}$represent the reference point and the imaged reference point, respectively.

This approach only considers the homography, in other words, the object motion. A second approach free from ground-truth information relies solely on the texture information resulting from the tracking process. Different approaches were analyzed, such as ZNCC[88], MI [89], and SSIM [90]. All of them are very robust to luminance variations. Of the three approaches listed, the SSIM showed to be more adequate in measuring the tracking quality, due to the fact that it is based on three image components: luminance, contrast and structure. The next equations show how it is calculated:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x{}^2 + \mu_y{}^2 + C_1} \tag{14}$$

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{15}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x{}^2 + \sigma_y{}^2 + C_2} \tag{16}$$

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}} \tag{17}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3} \tag{18}$$

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y) \tag{19}$$

$$SSIM(x,y) = l(x,y) * c(x,y) * s(x,y) \tag{20}$$

In the previous equations, x and y represent two non negative image signals, x_i and y_i represent pixel intensities, N corresponds to the number of pixels in the images and C_1, C_2 and C_3 are constants placed in order to avoid instability.

Both reprojection error and SSIM are directly related. The difference between them is that while the first varies from 0 to infinity, the second is bounded by the [0,1] interval. The relationship between them will be further explored in the analysis chapter (chapter 9). Since the SSIM value will be used as the utility function of the decision model proposed, all comparisons regarding tracking quality will refer to this metric.

4.2 SPEEDUP AND ERROR DEFINITION

As previously stated, the main goal of the proposed decision model is to identify when it is possible to change the employed tracking algorithm to a computationally less demanding one without significantly harming tracking quality. In order to guarantee the minimum amount of error, one should use the most complex tracking algorithm. While this diminishes the rate of error, it demands more processing power. That being said, the speedup of the proposed model is calculated based on the processing time of the most complex algorithm, which, out of the tests performed, was the SIFT-SIFT combination. In the worst case scenario, whenever the model cannot find a suitable algorithm for the current frame characteristics, the algorithm to be used will be the most complex one.

The processing time in milliseconds for each algorithm combination used is shown in Table 1. It describes the time spent for calculating a single feature, considering detection, extraction and matching phases.

Table 1. Mean execution times per feature for each tracking phase.

	SIFT-SIFT	SURF-SURF	FAST-SIFT	FAST-SURF	ORB-ORB
Detection	0.1199	0.1048	0.0003	0.0003	0.0121
Extraction	0.1883	0.1539	0.1025	0.0367	0.0118
Match	0.0308	0.0269	0.0224	0.0230	0.0125
Overall	**0.3390**	**0.2857**	**0.1253**	**0.0602**	**0.0364**

For a sequence composed of 5000 frames (1000 for each of the five parameters), the speedup is calculated according to the following equation:

$$WorstTime = WorstAlgorithmTime * NumOfFrames \tag{21}$$

$$ProposedTime = \sum_{i=1}^{NumAlgorithms} AlgorithmTime_i * NumOfFrames_i \tag{22}$$

$$Speedup = \frac{WorstTime}{ProposedTime} \tag{23}$$

$WorstAlgorithmTime$ refers to the total time required (considering detection, description and matching) of the most complex algorithm. As shown in Table 1, this points to the SIFT-SIFT combination. $NumOfFrames$ is the total number of frames used on the sequence, which in our case is 5000. $AlgorithmTime_i$ is the total time (considering detection, description and matching) of a certain algorithm. Since five different combinations are being considered, $NumAlgorithms$ is 5. At last, $NumOfFrames_i$ corresponds to the number of frames chosen by the decision model for a specific algorithm.

According to Lieberknecht et al.[81], an image is considered to be successfully tracked when the mean of the sum of the reprojection errors of its four corners is not higher than 10 pixels. As with the speedup calculation, the original amount of frames not detected by the SIFT-SIFT combination is calculated, to be used as reference error. The SIFT algorithm was chosen to be the reference due to its broad utilization in visual tracking applications, despite the SURF-SURF combination showing a lower general SSIM result for the tested sequences. After that, for each of the four models generated, the number of not detected frames is also calculated. Both base error and proposed error are calculated as follows:

$$BaseError = \frac{NumOfDetectedFrames}{NumOfFrames} \tag{24}$$

$$ProposedError = \frac{\sum_{i=1}^{NumAlgorithms} NumOfDetectedFrames_i}{NumOfFrames} \quad (25)$$

It is important to note that the reprojection error can only be used when ground truth data is available. Since this is not the case when executing tracking applications in real time, SSIM must be used instead. Using the same training data captured earlier, it is possible to establish a relationship between reprojection error and SSIM. As shown in Figure 4, the frames can be considered "tracked successfully" when the reprojection error is no higher than 10 pixels. This leads to a confidence level of 99.04% when the SSIM value is higher than 0.267.

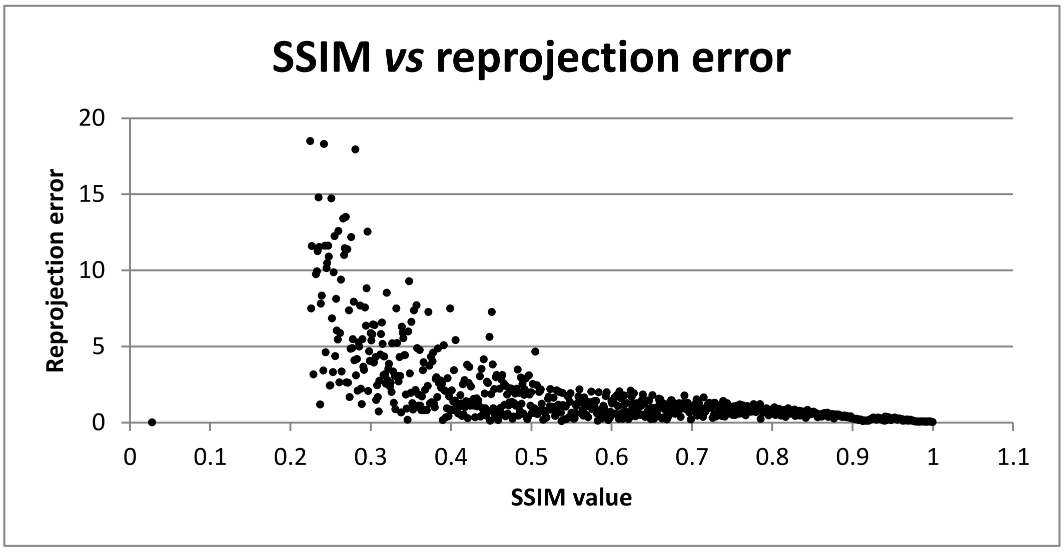


Figure 4. Correlation between SSIM and reprojection error. Points above the x axis map to sucessfully detected frames (reprojection error threshold equals 10 pixels).

CHAPTER 5 ANALYSIS OF DIFFERENT WARP FUNCTIONS

This chapter focuses on a classification of target objects according to their texture information and on how the motion of an object can be mathematically represented. In this work, the tracking process is performed based on the texture data acquired from the target object[103].

5.1 OBJECT CLASSES

Tracking algorithms work by analyzing the visual features of images and then computing their relative motion along frames. In [81], the authors propose a dataset and evaluation methodology for template-based tracking algorithms. Their motivation in creating such a dataset was to create a fair comparison between a wide range of template-based algorithms. Some use corners, edges and even whole regions of the image. The dataset originates from planar objects, which comprise one of the most fundamental ways of finding relative position and orientation of a camera with respect to different objects in real time. They consider that most tracking algorithms assume objects as at least locally planar. In order to represent all possible planar targets, they proposed four different object classes, namely “Low Texture”, “High Texture”, “Normal Texture” and “Repetitive Texture”. Each class is represented by two targets, as shown in Figure 5.

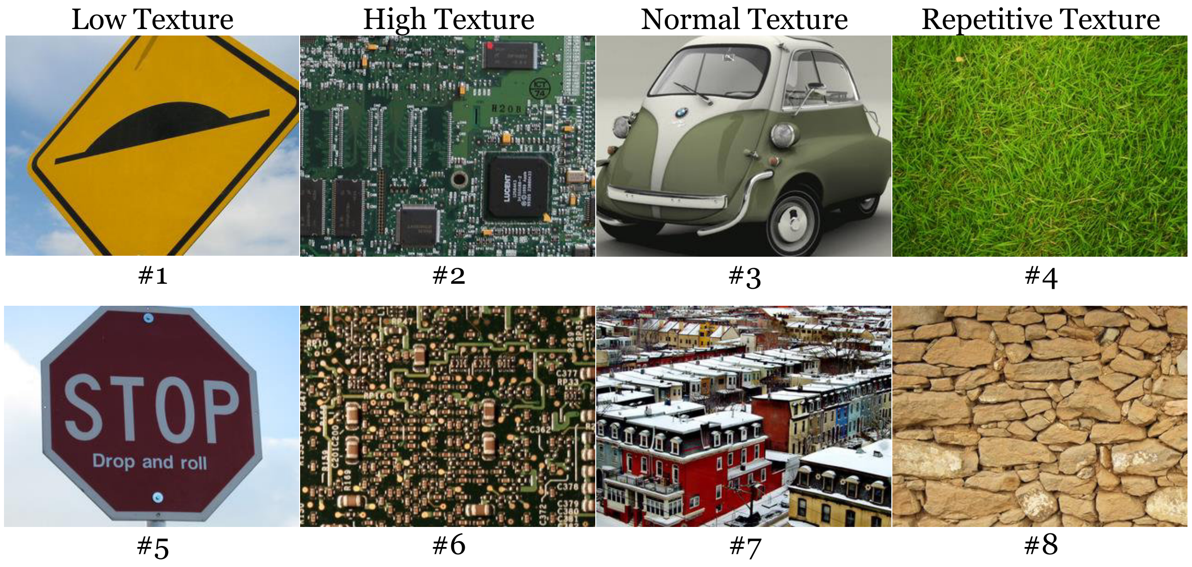


Figure 5. Four different object classes: The “Low Texture” group consists of images of road signs which are composed of two distinct colors and large uniform areas, thus large edges are visible. In the “High Texture” group there are images of electronic boards, one image with mainly large components, and another with mainly small ones. Images of a car and of a cityscape are in the “Normal Texture” object class. Finally, the group of “Repetitive Texture” based images is composed of an image of a wall made of

different sized stones and of an image with English lawn which features many extremely small structures everywhere [81].

In this work, four different models were generated, based on each available planar object class. Since each class has two targets, the first object of each class was used for training the model, while the second one was used to evaluate the results obtained. The template size used for all models was 320×240, while the full image resolution to be tracked was 640×480.

5.2 INPUT PARAMETERS

According to the previous section, a key characteristic was used to classify the objects into different classes: its texture. In order for the model to be as generic as possible, it would be wise not to depend exclusively on object texture itself. Thus, other parameters had to be used as inputs for the model. After some research, we decided to extract parameters from both object movement and scene influence over the tracked object. A total of five different parameters were selected, three of them pertaining to the tracking warp function (transformation from one frame of the sequence to another) and the other two related to the object illumination and occlusion.

The object motion generally occurs in the form of a parametric motion [82]. Depending on the object itself and its motion in 3D space, different warp functions can be defined. A warp function's general notation can be

$$w: P^2 \times R^n \rightarrow P^2, \tag{26}$$

in which w is the warp function that takes a 2D point and takes it to another 2D point. n is the number of parameters of the considered motion model that corresponds to the number of DOF that defines the image transformation.

Four different warp functions commonly used in object tracking are: Translation, Similitude, Affine and Homography, respectively with 2, 4, 6 and 8 parameters [82]. Table 2 illustrates the main differences between these four parametric motion models.

Table 2. Parametric motion models commonly used in planar object tracking.

Model	**Parameters**	**Formula**
Translation	$p = (\boldsymbol{t})$ (2)	$w(x, p) = x + \boldsymbol{t}$
Similitude	$p = (s, \varphi, \boldsymbol{t})$ (4)	$w(x, p) = (1 + s)R(\varphi)x + \boldsymbol{t}$
Affine	$p = (a_{11}, a_{12}, a_{21}, a_{22}, \boldsymbol{t})$ (6)	$w(x, p) = Ax + \boldsymbol{t}$
Homography	$p = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32})$ (8)	$w(x, p) = Hx$

In the previous table, t represents a translation in x and y directions, p is the set of parameters used by the warp function, s maps to the scale and φ to the rotation angle used. A and H are affine and homography transforms, respectively.

As the homography motion model is less restrictive, it was chosen to represent the motions in this work. The first three input parameters, namely Rotation, Scale and Distortion, are extracted directly from the homography. Figure 6 illustrates the homography in matrix form and highlights which part is used in each parameter estimation.

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Figure 6. Homography sections: the section highlighted in red contains rotation and scale information; the green section stores translation information; the orange section provides the values used in the projective distortion computation The white part is typically equal to 1.

According to Hartley and Zisserman[82], every projective transformation can be decomposed into the product of three matrices, such as

$$H = H_S H_A H_P = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ V^T & v \end{bmatrix} = \begin{bmatrix} A & t \\ V^T & v \end{bmatrix}, \tag{27}$$

in which A represents the affine part of the transform, K relates to the anisotropic scale and V^T to the distortion parameters. I is a 2×2 identity matrix and v is a scalar that can be used to make the homography be represented by 8 parameters up to a scaling factor, for $v \neq 0$.

The first two input parameters, rotation and scale, are extracted from the affine part of the homography. Both rotation and scale can be retrieved using SVD (Singular Value Decomposition), as follows:

$$A = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} = R(\theta)R(-\varphi)DR(\varphi), \tag{28}$$

$$\text{with } D = \begin{bmatrix} \rho_1 & 0 \\ 0 & \rho_2 \end{bmatrix}, \tag{29}$$

$$\text{and } UDV^T = (UV^T)(VDV^T) = R(\theta)(R(-\varphi)DR(\varphi)). \tag{30}$$

Some algorithms are more invariant to rotation than others. Thus, rotation can be considered as an important parameter to compare tracking algorithms. Scaling can be dealt with by the use of image pyramids, so that scaling's influence over image tracking is not as significant as rotation. The important scenarios related to scaling occur when the object is so small or so big that features are lost, since the image sensor is not capable of capturing the entire object information.

The third parameter, distortion, represents the module of the vector formed by the h_{31} and h_{32} values, as shown in equation (31). The higher this value, the more biased the object, meaning it is more difficult to track it. This way, more robust trackers should fail less with the increase of the distortion value than simpler ones.

$$Distortion = \sqrt{{h_{31}}^2 + {h_{32}}^2} \tag{31}$$

Illuminance variation can occur for different reasons. Its main consequence is that it can modify the appearance of either the whole scene or perhaps just isolated parts of it. In the first case, this is usually known as global illumination variation, while in the second case, local illumination variation. The main difference between these two varying scenarios is that in the global case, the pixels of the whole image are affected coherently. That does not happen in the local variation case, when only parts of the image are affected by the light. As shown in[83], local lighting variations are more complex to deal with and can generate more tracking failures than global variations.

Assuming that the object was tracked previously and the result of this tracking process is an image with the same dimensions of the object image being tracked, the luminance variation is calculated according to equation (32):

$$Luminance = \frac{\sum_{i=0}^{numPixels}\left(\frac{{I^*}_i}{I_i}\right)}{numPixels}, \tag{32}$$

in which $numPixels$ corresponds to the number of pixels pertaining to the reference image, ${I^*}_i$ is a pixel value (luminance) from the input image and I_i is a pixel value (luminance) from the reference image.

The input parameter representing occlusion is not so simple to calculate[104]. One may think that a simple color comparison is enough, meanwhile other authors state that it is safer to use both color and gradient information [84][85][86]. We make use of LBP (Local Binary Pattern) to find which parts of the image are really occluded [87]. This makes the occlusion detection process more robust to luminance variations. The occlusion parameter is then calculated as follows:

$$LBP_i = \sum_{n=1}^{8} s(x_{i,n} - x_{i,0})2^n, \qquad s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{33}$$

$$Occlusion = \frac{\sum_{i=0}^{numPixels} cmp(LBP_i^{Ref}, LBP_i^{Frame})}{numPixels}, \tag{34}$$

$$cmp(\alpha,\beta) = \begin{cases} 1, \left(\sum_{j=0}^{7} ((\alpha\ AND\ 2^j)\ XOR\ (\beta\ AND\ 2^j)) \gg j \right) > 4 \\ 0, otherwise \end{cases} \tag{35}$$

In the previous equations, $x_{i,n}$ represents the n^{th} pixel of an equally spaced circular neighborhood of pixel i ($x_{i,0}$ coresponds to the center pixel), $numPixels$ maps to the number of pixels on the reference image, and LBP_i^{Ref} and LBP_i^{Frame} are the LBP value of a pixel in the reference image and input image, respectively. Both AND and XOR are bitwise operators and $\gg$ is the shift-right operator. The calculation of the occlusion rate involves comparing the LBP values of all pixels; when the similarity between them is lower than four, the pixel is considered occluded. In the end, the amount of occluded pixels is divided by the total amount of image pixels.

That being said, the key point of the proposed model is to find the cases where each algorithm performs well, and switch to the one that demands less computational power whenever possible. "Performing well" means that an algorithm gives as output an acceptable result, respecting a previously defined error threshold.

CHAPTER 6 DEFINITION OF THE TEST SCENARIOS

This chapter describes how the training sequences were generated and performs a comparison regarding different background scenarios in order to justify the approach chosen.

6.1 TRAINING SEQUENCES

The training phase of each object class model starts with the generation of control image sequences. Each sequence corresponds to the variation of a single parameter, so that the error measurements can be computed and later combined by the model. 20 sequences were generated, which corresponds to five parameters against four objects classes. A total of 1000 frames were generated for each parameter, with the values linearly distributed through a minimum and maximum value. The ranges used for each of the parameters are shown in Table 3.

All four image classes were used in the generation of the control image sequences. In order to illustrate how the parameters varied, Figure 7 to 11show the parameters applied to some of the dataset images.

Table 3. Parameter ranges used (rotation values are in radians).

	Rotation	Scale	Distortion	Luminance	Occlusion
MIN_VALUE	0	0.25	0	0.092057	0
MAX_VALUE	2π	5	0.005768	3.246554	0.338867

The rotation sequence was comprised of a full 360° rotation over the object plane, as shown in Figure 7. Since the other parameters should remain unaltered, the center of mass of the object stays on the same place during the entire sequence.

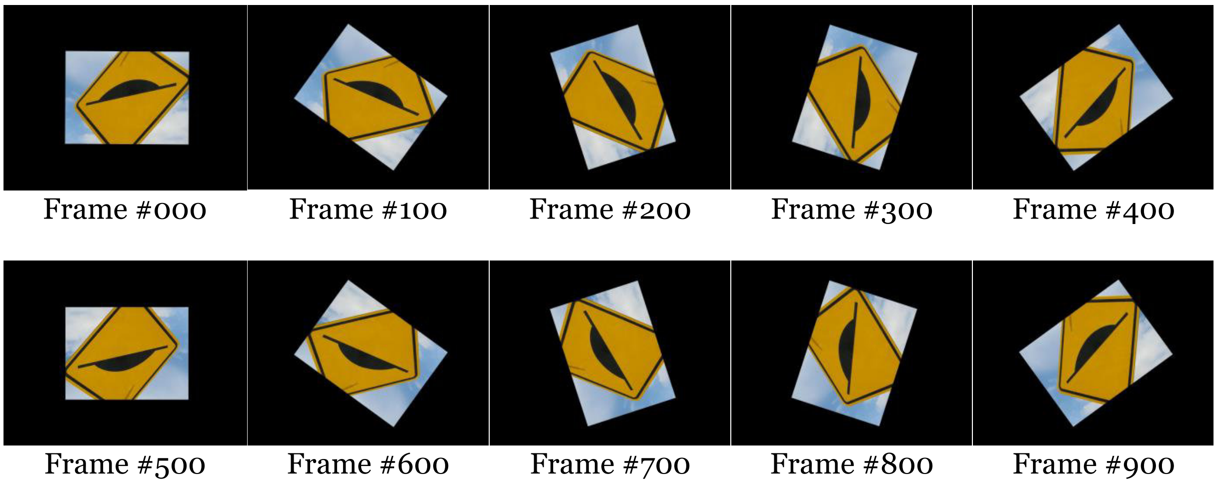


Figure 7. Samples from the rotation sequence.

The scale sequence started representing 25% of the original object size and increases until reaching 500%. Despite the fact that only the scale varies, with the increase of this parameter, some parts of the object start to fall out of the camera's view, representing an effect similar to an occlusion. Figure 8 shows samples from the scale sequence.

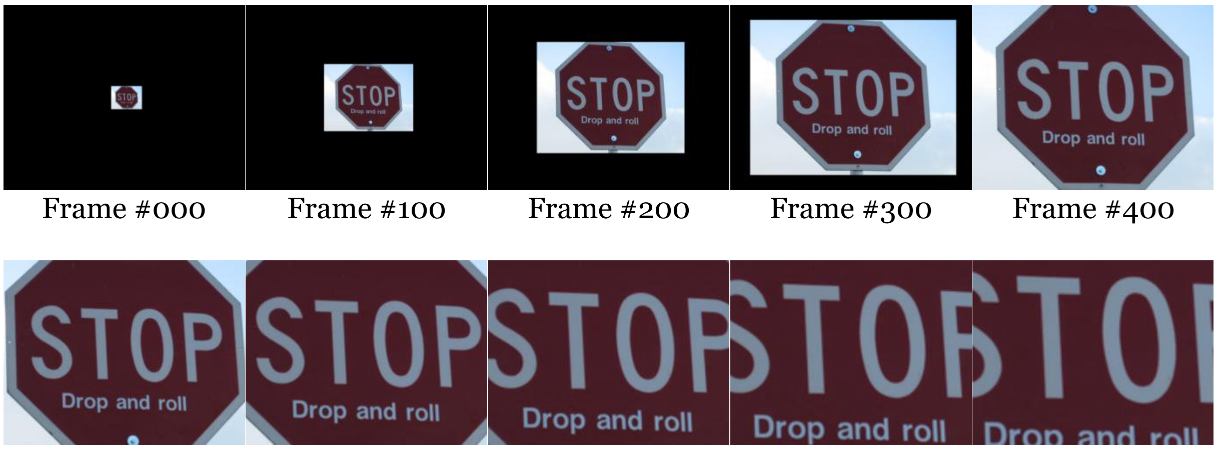


Figure 8. Samples from the scale sequence.

The distortion sequence was divided in two parts. The first included a distortion resulting from a horizontal rotating movement, while the second part was based in a vertical rotation. In both cases, the object was rotated from 0º to approximately 80º, when the planar object was almost not visible. Figure 9 presents some samples from the distortion sequence.

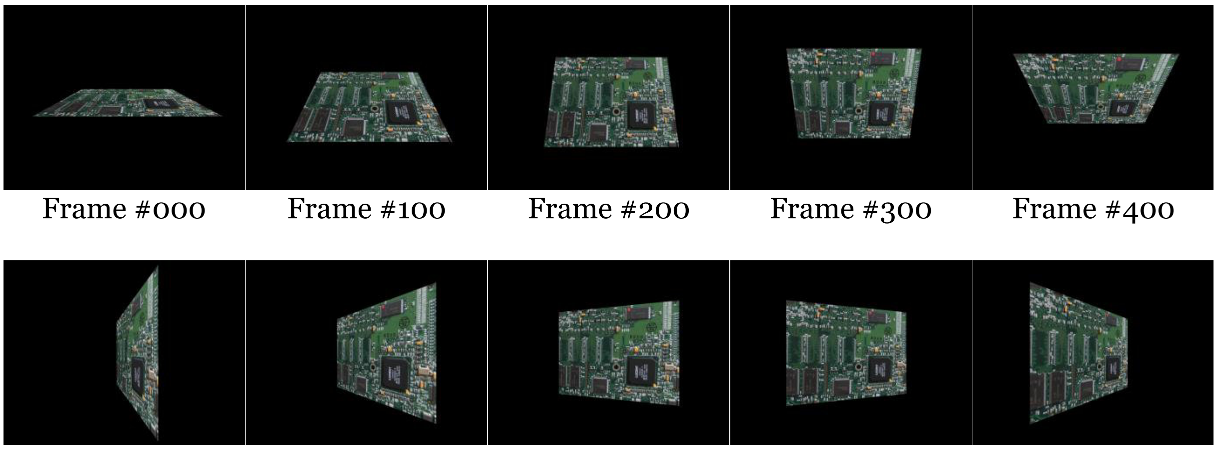


Figure 9. Samples from the distortion sequence.

The illumination sequence alters the object from a 9.2% illumination compared to the original object to 324.65%. The structural information of the object starts to change when some of its pixels reach the maximum possible white value. Figure 10 illustrates some examples from the luminance sequence.

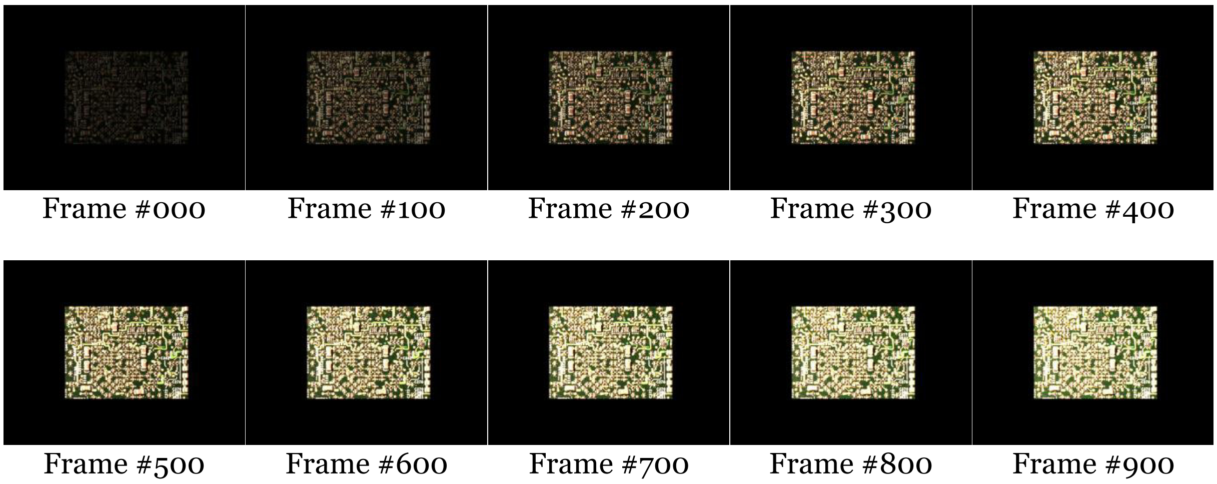


Figure 10. Samples from the luminance sequence.

At last, the occlusion sequence starts consuming the object from a state of complete visibility to an occlusion rate of 0.3388. It is important to note that this value does not directly correspond to a 33% visual occlusion, because it is calculated using LBP correspondences. Since the LBP threshold used was 50%, fewer different pixels are considered as occluded pixels. According to the tests performed, the 0.3388 occlusion rate maps to an 80% visual occlusion approximately. Figure 11 shows some examples from the occlusion sequence.

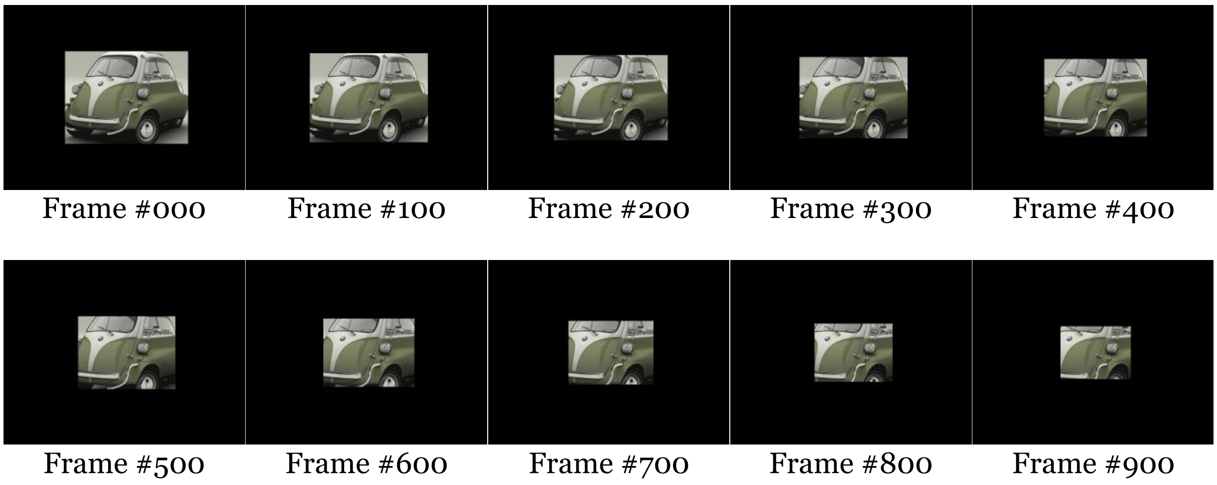
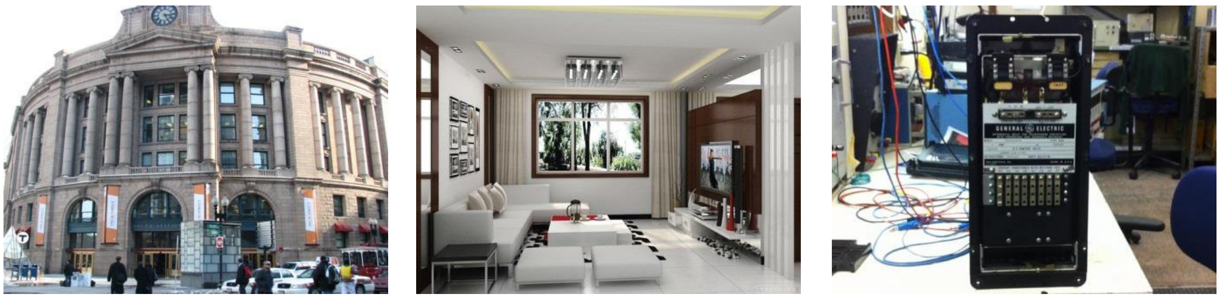


Figure 11. Samples from the occlusion sequence.

Note that the training sequences are composed by the template placed in a black background, varying according to the parameter being processed. Due to the nature of the trackers used, which are keypoint-based trackers, this specific test configuration should present similar results in the case of a different background color. Different backgrounds could generate false matches, but the number of outliers is not sufficient enough to make the results differ from the original approach. In order to analyze such a proposition, we have calculated the SSIM value using the SIFT algorithm for three different backgrounds, shown in Figure 12. They were chosen because they correspond to common scenarios of application of AR systems. The first one represents an outdoor environment, in which the user is free to move in the open space while observing his/her surroundings; the second background refers to an indoor scenario, located inside a house; the last scenario is in the context of AR supporting maintenance of electrical equipment, and represents a maintenance laboratory of a local Electricity production company. Samples of the generated sequences are illustrated in Figure 13, Figure 14 and Figure 15.



(a) (b) (c)

Figure 12. Three different backgrounds used in AR applications: a) outdoor environment; b) indoor environment; c) laboratory for maintenance of electrical equipments.

Table 4 lists the correlation values of the SSIM regarding each of the three backgrounds and the original (black) one. It is possible to notice that, for all parameters, the correlation among the values found was higher than 0.934028. Statistically, this fact (correlation higher than 0.9) is enough to say that the tests present similar behaviour and one background can be used in the place of the others [95].

Table 4. Correlation values for the SSIM values using different backgrounds.

	Black *vs* background #1	**Black *vs* background #2**	**Black *vs* background #3**
Rotation	0.961186	0.934028	0.975594
Scale	0.974302	0.976893	0.971979
Distortion	0.963291	0.959513	0.962330
Lumminance	0.970602	0.968853	0.973633
Occlusion	0.993852	0.993632	0.994141

Another statistical test was performed based on the P-value of the T-test. Table 7 lists the values found for the second analysis. The T-test analyzes the concentration of the mean value for the sequences, stating that the more the sequences are alike the more the p-value is closer to 0.5 [95].

Table 5. P-value of the T-test regarding SSIM values using different backgrounds.

	Black *vs* background #1	**Black *vs* background #2**	**Black *vs* background #3**
Rotation	0.506973	0.450660	0.504166
Scale	0.508148	0.508981	0.508311
Distortion	0.506125	0.520940	0.506277
Lumminance	0.520480	0.518283	0.518876
Occlusion	0.594415	0.618615	0.570997

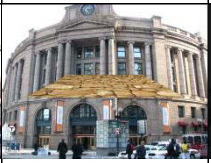
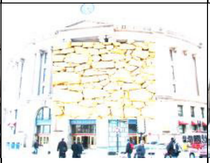
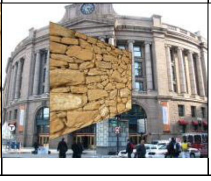
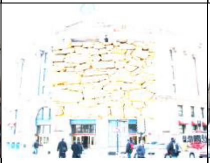
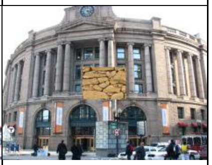
	Rotation	Scale	Distortion	Lumminance	Occlusion
Frame #0000					
Frame #0100					
Frame #0200					
Frame #0300					
Frame #0400					
Frame #0500					
Frame #0600					
Frame #0700					
Frame #0800					
Frame #0900					

Figure 13. Image sequences corresponding to the five parameter variations generated by mixing the template to be tracked with outdoor background.

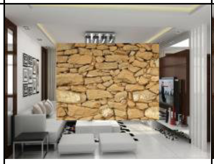
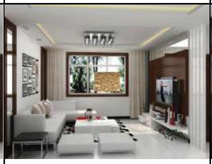
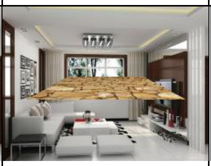
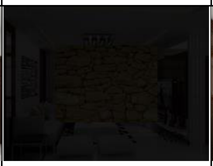
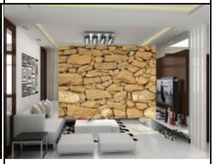
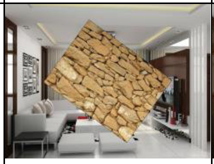
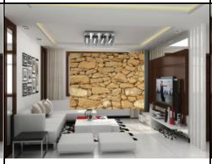
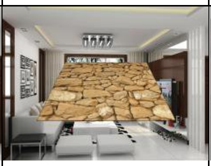
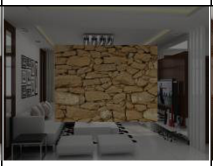
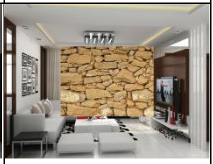
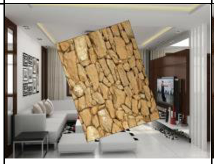
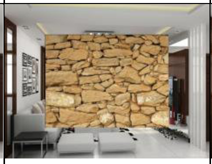
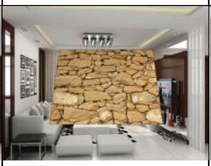
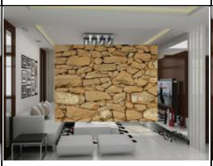
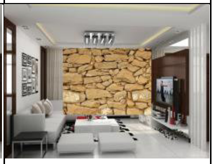
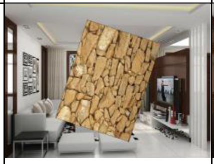
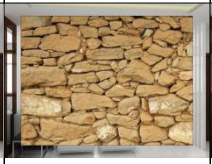
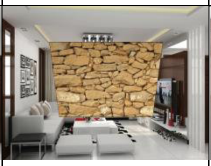
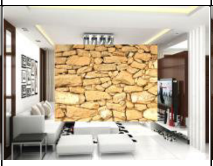
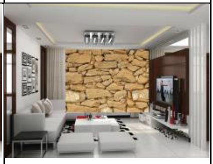
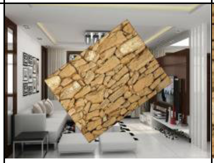
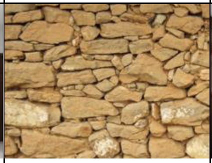
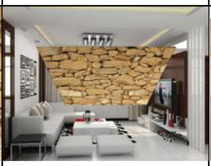
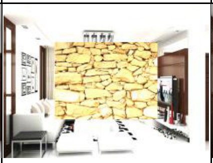
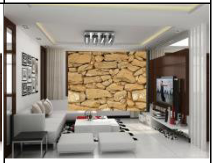
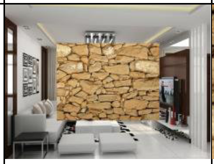
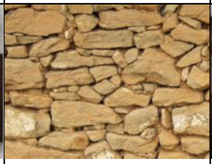
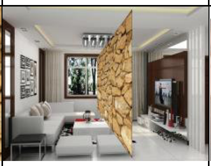
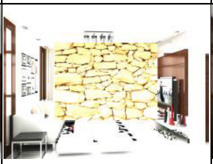
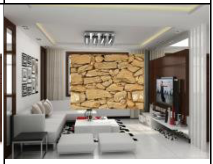
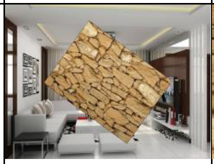
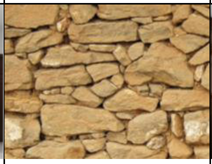
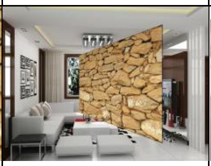
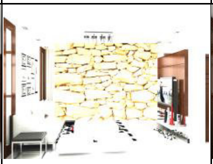
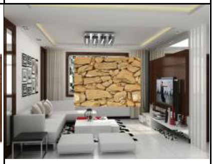
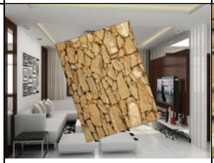
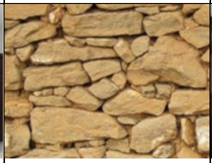
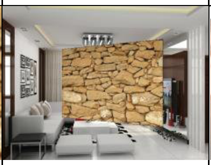
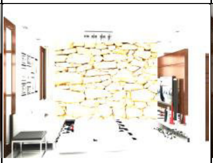
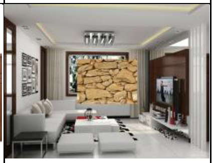
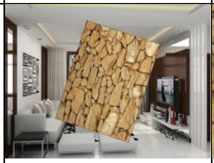
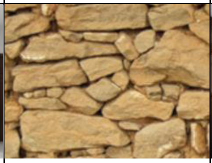
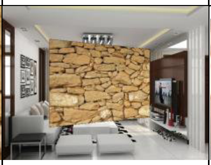
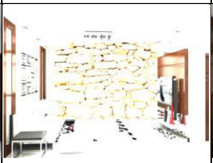
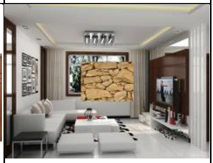
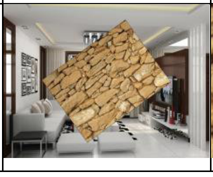
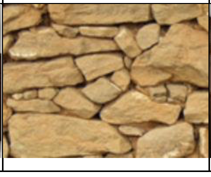
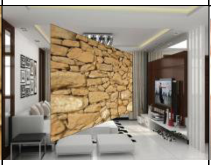
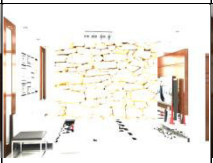
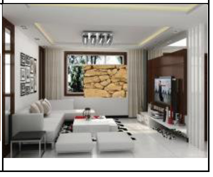
	Rotation	Scale	Distortion	Lumminance	Occlusion
Frame #0000					
Frame #0100					
Frame #0200					
Frame #0300					
Frame #0400					
Frame #0500					
Frame #0600					
Frame #0700					
Frame #0800					
Frame #0900					

Figure 14. Image sequences corresponding to the five parameter variations generated by mixing the template to be tracked with indoor background.

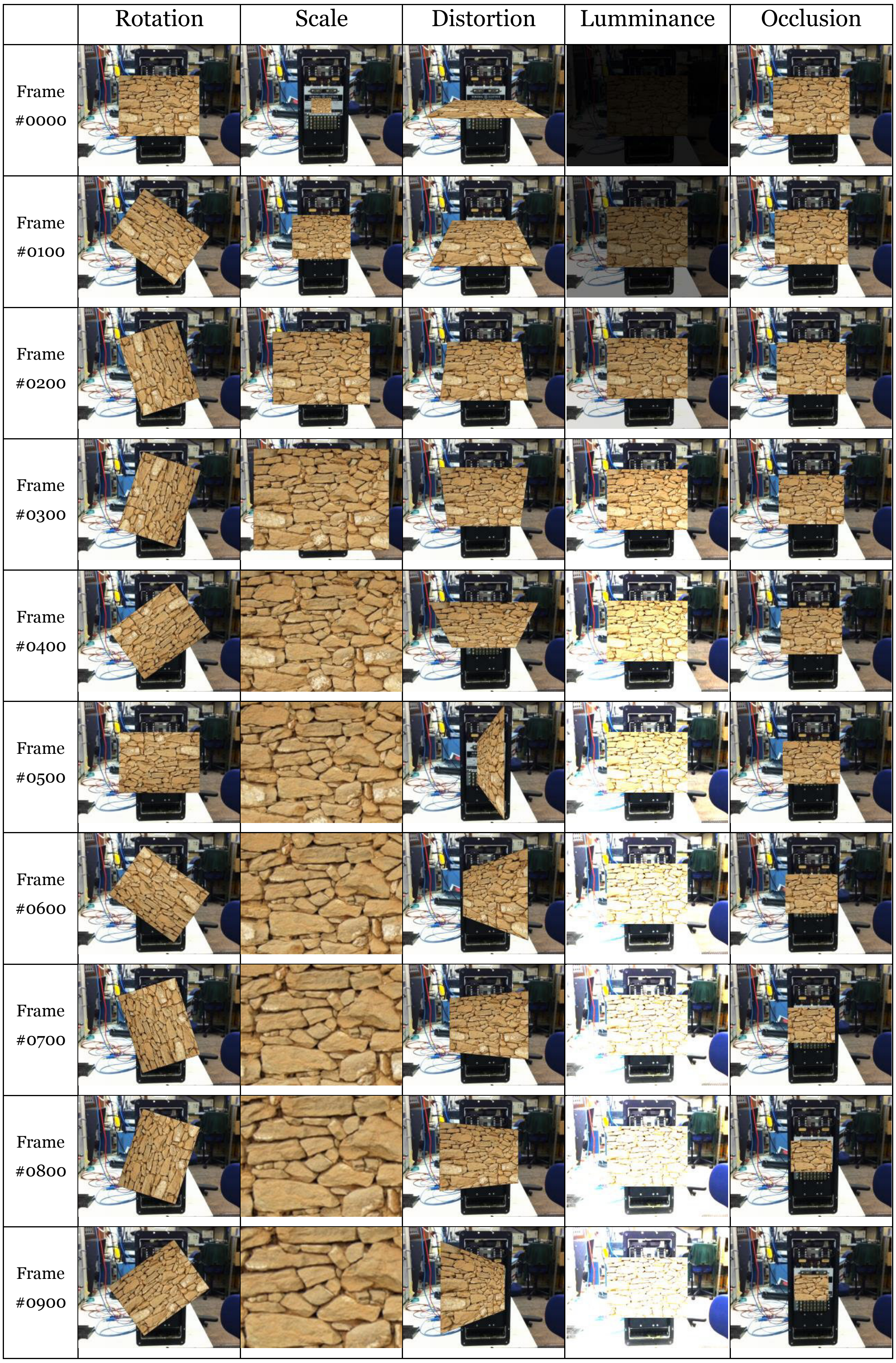


Figure 15. Image sequences corresponding to the five parameter variations generated by mixing the template to be tracked with a maintenance laboratory context.

CHAPTER 7 SELECTION OF DIFFERENT OBJECT TRACKERS

Pattern recognition methods can be classified in two different areas, according to [6]: decision-theoretic and structural. The first category deals with patterns described using quantitative descriptors, such as length, area and texture. The second category identifies patterns best described by qualitative descriptors, such as shape and boundary.

An image feature can be defined as an “interesting” part of an image and serves as starting point for computer vision applications. In pattern recognition literature, the name feature is often used to denote a descriptor [6]. The two most common operations regarding image features are feature selection and feature extraction. The former selects the most interesting parts of a certain image, while the latter describes these parts by filling its corresponding descriptor array.

Several criteria must be taken into consideration when selecting proper detector-descriptor combinations. Two of them are the repeatability rate and information content. The first one evaluates how the geometric stability behaves under different transformations, while the second one measures the distinctiveness of the features [51].

Since this work makes use of different combinations of feature detectors and extractors, this chapter will give a brief description about those used in this thesis. The selected algorithms are the ones with best results (best tracking quality) found in literature.

7.1 FEATURE DETECTORS

The most widely used interest point detector is likely the Harris corner detector [52], proposed back in 1988, based on the eigenvalues of the second-moment matrix. However, Harris corners are not scale-invariant. Lindeberg introduced the concept of automatic scale selection [53]. This allows for detecting interest points in an image, each with their own characteristic scale. He experimented with both the determinant of the Hessian matrix as well as the Laplacian (which corresponds to the trace of the Hessian matrix) to detect blob-like structures. Mikolajczyk and Schmid refined this method, creating robust and scale-invariant feature detectors with high repeatability[55], which they coined Harris-Laplace and Hessian-Laplace [54]. They used a (scale-adapted) Harris measure or the determinant of the Hessian matrix to select the location, and the Laplacian to select the scale. Focusing on speed, Lowe [56] approximated the Laplacian of Gaussian (LoG) by a Difference of Gaussians (DoG) filter.

Beyond the Harris model and it’s variants, several other scale-invariant interest point detectors have been proposed. Examples are the salient region detector proposed by Kadir and Brady [57], which maximizes the entropy within the region, and the edge-based region detector proposed by Jurie *et al.* [58].

Research on existing detectors [59][60] concludes that Hessian-based detectors are more stable and repeatable than their Harris-based counterparts. Using the determinant of the Hessian matrix rather than its trace (the Laplacian) seems advantageous, as it fires lesson elongated, ill-localized structures. Also, approximations like the DoG can bring speed at a low cost in terms of lost accuracy.

Four different feature detectors are used in this work: FAST, ORB, SURF and SIFT. They were selected mainly because of their code availability and their performance. It is important to notice that these four feature detectors are those most widely used by the scientific community.

7.1.1 FAST

The FAST algorithm (Features from Accelerated Segment Test[61]) takes into consideration the local neighborhood information of pixels and makes use of a 16 pixel ring around the interest point being analyzed, as shown in Figure 16.

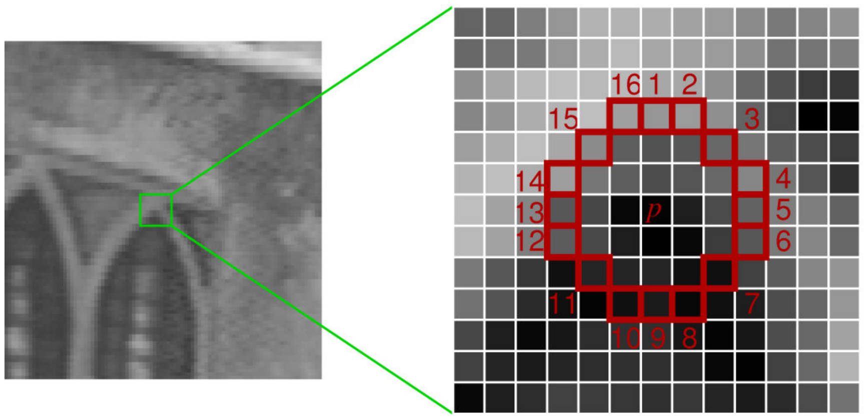


Figure 16. Neighborhood information used by the FAST algorithm.

The FAST algorithm works as follows. The central point is considered a corner if there is a sequence of n pixels on the ring in which all of them are brighter than the central pixel plus a threshold. By taking n equal to 12, it is possible to exclude a high number of non-corner pixels, since the test only needs to verify at most four pixels (1, 3, 5 and 9), corresponding to the compass orientations (north, south, east and west, respectively).

If the central point, which is the point being analyzed, is indeed a corner, at least three of the four tested pixels should be brighter than the central pixel plus a threshold or darker than the central pixel minus a threshold. After applying this initial filtering process to all pixels, the complete test can be applied to the remaining ones.

Despite presenting high performance due to this test optimization, the algorithm described shows a series of weaknesses:

- The optimized non-corners filtering does not generalize well for $n < 12$;
- The choice for the specific order for testing the surrounding pixels requires existing knowledge about the image features (an optimal order and rotation should be different);
- The knowledge acquired from the initial four tests is not used on the complete test after the first filtering processing;
- The process performed detects multiple near features.

Through the use of machine learning, it is possible to significantly improve the performance of the FAST feature detector. The complete process can be divided into two phases. At first, in order to build a feature detector for a specific n value, all the corners contained in a trainning image set are detected (preferably, these images pertain to the same application domain in which the application will be used). This corner detection for the entire set uses the complete test (checking the values of all 16 surrounding pixels). For each of the 16 positions, the analyzed pixel can have one of the following states:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \leq I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t \quad \text{(similar)} \\ b, & I_p + t \leq I_{p\to x} \quad \text{(brighter).} \end{cases} \tag{36}$$

In the previous formula, $I_{p\to x}$ corresponds to the pixel intensity on neighborhood position x, I_p is the central pixel being analyzed and t is the threshold used.

By choosing a specific x value and computing $S_{p\to x}$ for every p pertaining to the training images, it is possible to partition the central pixels into the three distinct sets P_d, P_s and P_b, according to the result of the $S_{p\to x}$test.

Let K_p be a boolean variable that is true whenever p is a corner and false otherwise. The algorithm's second phase consists of using the algorithm described in ID3 (Induction of Decision Trees) [62] and starts by selecting the location of the neighborhood that detains the most relevant information about the decision if a pixel is really a corner, measured through the K_p entropy.

The K_p entropy level is calculated as follows:

$$H(P) = (c + \bar{c}) \log_2(c + \bar{c}) - c \log_2 c - \bar{c} \log_2 \bar{c}, \tag{37}$$

where

$$c = \left|\{p|K_p is\ true\}\right| \text{ (number of corners)}$$

and

$$\bar{c} = \left|\{p|K_p is\ false\}\right| \text{ (number of non corners).}$$

The choice of x then yields the information:

$$H(P) - H(P_d) - H(P_s) - H(P_b). \quad (38)$$

After determining which neighborhood position represents the highest amount of information, the process is recursively applied in all three subsets and a new x is chosen for each of them, with the goal of maximizing the information obtained from all three. The entire process reaches an end when the entropy of the entire set reaches zero. This means that all analyzed central pixels have the same K_p value, i.e., they are all corners or non-corners. Given that the process was finalized, there is a decision tree capable of correctly classifying all the pixels of the training set, according to the test criteria of the chosen FAST configuration.

Because of the fact that the segment test is not based on a corner response function, it is not possible to directly apply a non-maximum suppression scheme to the selected features. It is necessary to compute a score value V for each detected corner, and use such value to eliminate neighbor corners with a lower V score.

The V value is calculated based on the sum of absolute differences between the pixels in the contiguous arch, as follows:

$$V = max\left(\sum_{x \in S_{bright}} |I_{p \to x} - I_p| - t, \sum_{x \in S_{dark}} |I_p - I_{p \to x}| - t\right). \quad (39)$$

7.1.2 ORB

The ORB algorithm [63] is built based on the well-known FAST keypoint detector [61] and the BRIEF descriptor [64]. For this reason, it is called ORB (Oriented FAST and Rotated BRIEF). Figure 17 shows a typical matching result using ORB for real-world images with viewpoint change. Green lines are valid matches; red circles indicate unmatched points.

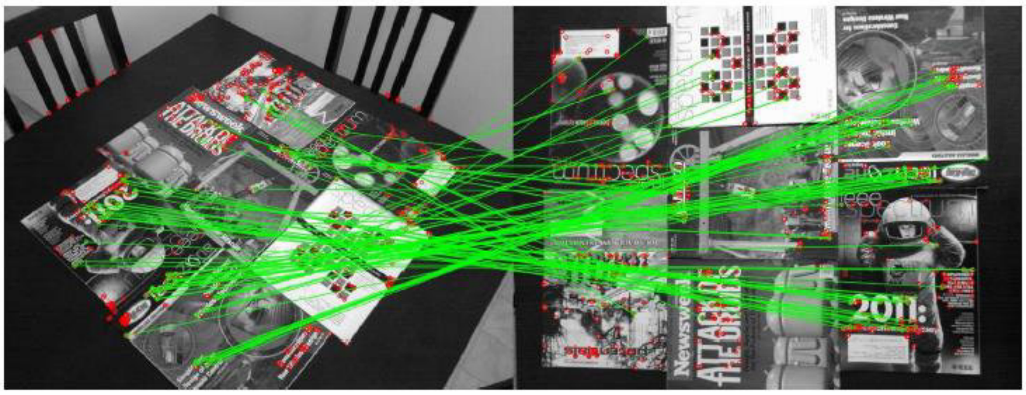


Figure 17. ORB detector in action.

The advantages of ORB are that it:

- Adds a fast and accurate orientation component to FAST;
- Develops an efficient computation of oriented BRIEF features;
- Analyzes the variance and correlation of oriented BRIEF features;
- Makes use of a learning method for de-correlating BRIEF features under rotational invariance, leading to better performance in nearest-neighbor applications.

The ORB detector makes use of the FAST-9 variation (circular radius of 9), due to its proven good performance record [63]. As FAST does not produce a measure of cornerness, ORB employs a Harris corner measure [65] to order its keypoints. For a target number N of keypoints, ORB first sets the threshold low enough so that it will be possible to get more than N keypoints, then they are ordered according to the Harris measure and only the top N points are picked up. FAST also does not produce multi-scale features. ORB employs a scale pyramid of the image, thus producing FAST features (filtered by Harris) at each level in the pyramid.

ORB's approach for calculating the keypoint orientation uses a simple but effective measure of corner orientation, the intensity centroid [65]. It assumes that a corner's intensity is offset from its center, and this vector may be used to impute an orientation. The moments of a patch can be defined as follows:

$$m_{ab} = \sum_{x,y} x^a y^b I(x,y). \tag{40}$$

And by using these moments, the centroid is calculated as:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right). \tag{41}$$

Therefore, a vector can be constructed from the corner's center, O, to the centroid, OC. The orientation of the patch is calculated as

$$\theta = atan2(m_{01}, m_{10}). \tag{42}$$

7.1.3 SURF

The SURF detector (Speeded Up Robust Features) [66] is based on the Hessian matrix, but uses a very basic approximation, just as DoG [67] is a very basic Laplacian-based detector. It relies on integral images to reduce the computation time and therefore it is also known as the "Fast-Hessian" detector.

Given a point$x = (x, y)$ in an image I, the Hessian matrix $H(x, \sigma)$ at scale σ is defined as follows:

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}, \tag{43}$$

where $L_{xx}(x, \sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in point x, and similarly for $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$.

The 9×9 box filters in Figure 18 are approximations for the Gaussian second order derivatives with $\sigma = 1.2$ previously described and represent the lowest scale possible (i.e. highest spatial resolution). The approximations are denoted by D_{xx}, D_{xy} and D_{yy}. The weights applied to the rectangular regions are kept simple for computational efficiency, but it is needed to further balance the relative weights in the expression for the Hessian's determinant with $\frac{|L_{xy}(1.2)|F|D_{xx}(9)|F}{|L_{xx}(1.2)|F|D_{xy}(9)|F} = 0.912 \cong 0.9$, where $|x|F$ is the Frobenius norm. This yields:

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2. \tag{44}$$

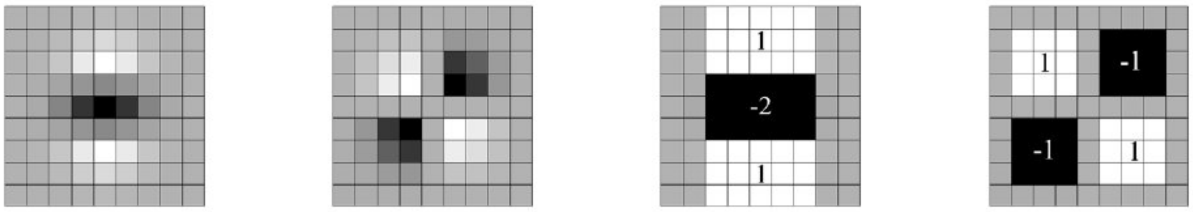


Figure 18. Left to right: the (discretized and cropped) Gaussian second order partial derivatives in y-direction and xy-direction, and their approximations using box filters. Gray regions are equal to zero.

The keypoint selection for the SURF algorithm is described as follows. At first, an integral image is built based on the original image. Such an integral image is used for speeding up the process of computing filters over the input image. The scale-space analysis starts with the SURF algorithm using image pyramids of filters (not images) to approximate the Laplacian of Gaussian (LoG), and, for this reason, it runs faster than SIFT, which uses DoG for approximation. The filters used by SURF span several octaves with a fixed number of scales in each, similar to SIFT. One of the advantages of using integral images is that it helps to keep the running speed constant as it is insensitive to increasing filter sizes. SURF uses box filters to calculate the Hessian determinant. The size of the box filter used is a multiple of its current scale. All filters passes occur on the original image, instead of iteratively like SIFT, which allows for parallel execution. The feature points are maxima of the determinants in the adjacent scale and points ($3 \times 3 \times 3$), similar to SIFT.

After selecting the feature points, orientation assignment is the next step. At first, the responses from the 2 first-order Haar wavelet filters (1,-1), in both dx and dy orientations, are collected on each feature point. These responses are put on a 2D plane as vectors $[dx, dy]$. Once more, the integral image is used with this box filter. An orientation window of 60 degrees is slid around the origin at the xy plane. Each angle will have a corresponding sum of magnitudes for every vector inside that window. The dominant orientation would be the angle of the largest sum. The results for the SURF detector can be seen in Figure 19.

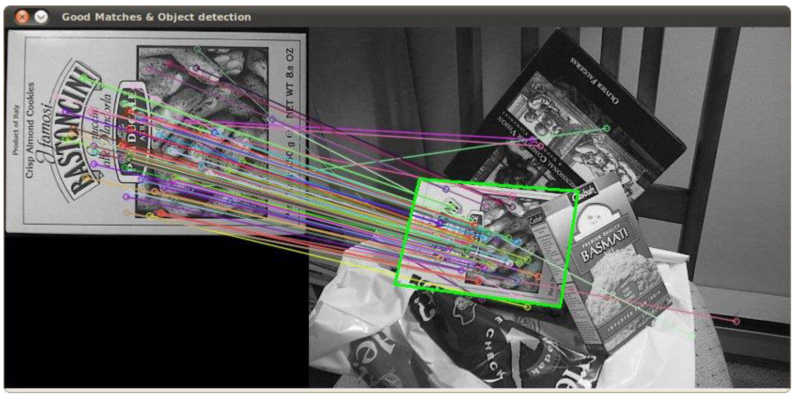


Figure 19. SURF detector in action.

7.1.4 SIFT

The SIFT algorithm (Scale Invariant Feature Transform) is the oldest of all four object detectors described in this chapter. It works in a way that the features detected by it are invariant to scale, rotation, translation, and partially invariant to illumination changes and affine projections. In sequence, a brief description will give information about how SIFT is able to support each of its main capabilities.

The scale invariance capability is supported through the use of successive applications of Gaussian filters and their derivatives, since this is the only kernel that allows scale space analysis, according to Lindeberg [68].

The rotation invariance is obtained by selecting points of maximum and minimum based on the DoG applied over scale space. Such point selection can be efficiently computed through the construction of image pyramids with resampling between each pyramid level. This process finds keypoints in regions and scale with high variation, which makes such regions stable and effectively representative of the image being described.

The smoothing operations are performed using $\sigma = \sqrt{2}$ and two 1D kernel passes (due to the fact that the 2D Gaussian is a separable filter), which allows the application of the filter using only 7 points for each pass. The successive pyramid levels are generated using bilinear interpolation with pixel spacing of 1.5 in each direction. This 1.5 spacing means that each new pixel is generated based on the combination of four adjacent pixels. Whenever a point of maximum or minimum is found in a certain pyramid level, the location corresponding to the pixel on the next level is calculated. In case it continues to be a pixel of maximum or minimum, regarding its eight neighbors, the pixel follows as a feature candidate. Otherwise, the pixel is discarded.

Each pyramid level is processed in order to extract both the gradients and the orientations. For each pixel A_{ij}, the gradient magnitude M_{ij} and the orientation R_{ij} are calculated using the following differences:

$$M_{ij} = \sqrt{\left(A_{ij} - A_{i+1,j}\right)^2 + \left(A_{ij} - A_{i,j+1}\right)^2} \tag{45}$$

$$R_{ij} = atan2(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{ij}) \tag{46}$$

Each point successfully classified as feature has a main orientation, which enables the rotation invariance aspect of the algorithm. This orientation is determined by the peak found on the local image gradient orientations histogram. The histogram is created using a weighted Gaussian window with $\sigma = 3$ times the values used during the pyramid generation. The weights are multiplied by the gradient threshold and then accumulated on the histogram in positions according to the R_{ij} orientation value. The histogram possesses 36 bins, covering 360 degrees of rotation (using intervals of 10 degrees each).

Figure 20 illustrates some features detected using the SIFT algorithm. The image at the top corresponds to the original image, without any modifications. The second image was generated from the first by the application of rotation, scale and stretching transformations, brightness and contrast changes and noise addition. Despite those modifications, 78% of the features found in the second image show a strong correspondence to the features found in the first image. In both images, each detected feature is represented by a square region and its orientation through a line that has its origin in the feature center and goes to the region's border.

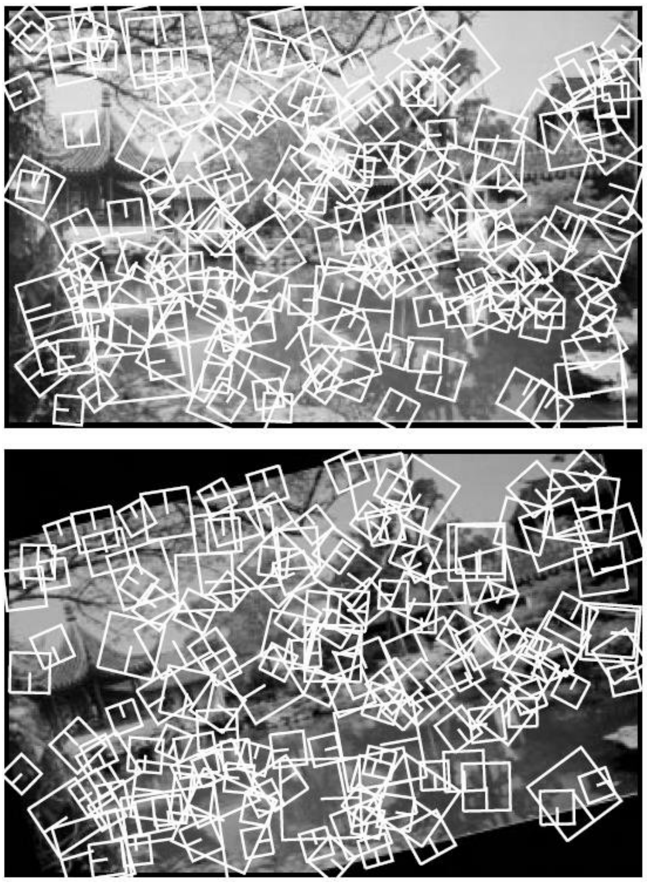


Figure 20. Features detected using the SIFT algorithm in two different images.

7.2 FEATURE EXTRACTORS

There is an even larger number of feature descriptors that have been proposed, including Gaussian derivatives [68], moment invariants [69], complex features [70][71], steerable filters [72], phase-based local features [73], and descriptors representing the distribution of smaller-scale features within the interest point neighborhood. The latter, introduced by Lowe [56], has been shown to outperform the others [75]. This can be explained by the fact that they capture a substantial amount of information about the spatial intensity patterns, while at the same time are resistant to small deformations or localization errors. The descriptor in [56], called SIFT for short, computes a histogram of local oriented gradients around the interest point and stores the bins in a 128-dimensional vector (8orientation bins for each of the 4 × 4 location bins).

Various refinements on this basic scheme have been proposed. Ke and Sukthankar[76] applied PCA (Principal Component Analysis) on the gradient image. This PCA-SIFT yields a 36-dimensional descriptor which is fast for matching, but has proved to be less distinctive than SIFT in a second comparative study by Mikolajczyk et al. [60]. Additionally, a slower feature computation reduces the effect of fast matching. In the same paper [60], the authors have proposed a variant of SIFT, called GLOH, which was proven to be even more distinctive with the same number of dimensions. However, GLOH is computationally more expensive.

Three different descriptor extractors are used in this work: ORB, SURF and SIFT. They were selected because of their code availability and high quality of performance. It is important to note that these feature extractors are the ones most widely used by the scientific community. FAST use is specific for feature detection, since it does not define a way for representing the neighborhood of the feature. Because of that, FAST is commonly used in conjunction with a feature extractor (such as ORB, SURF or SIFT, for example).

7.2.1 ORB

As described above, the ORB algorithm bases its descriptor extraction phase on the BRIEF descriptor [64]. This descriptor is composed of a binary string that represents an image patch constructed from a set of binary intensity tests. Considering a smoothed image patch p, a binary test τ is defined by:

$$\tau(p; x, y) = \begin{cases} 1 : p(x) < p(y) \\ 0 : otherwise \end{cases}, \tag{47}$$

where $p(x)$ is the intensity of p at a point x. The feature is then defined as a vector of n binary tests:

$$f_n(p) := \sum_{1 \le i \le n} 2^{i-1}\, \tau(p; x_i, y_i). \tag{48}$$

Different test distributions were considered by the ORB algorithm. One of the distributions that showed good performance occurred with a Gaussian distribution around the center of the patch. The vector length was chosen to be $n = 256$.

In order to make BRIEF invariant to in-plane rotation, ORB steers the BRIEF descriptor according to the orientation of keypoints. For any feature set of n binary tests at location (x_i, y_i), define the $2 \times n$ matrix

$$S = \begin{pmatrix} x_1, \dots, x_n \\ y_1, \dots, y_n \end{pmatrix}. \tag{49}$$

Using the patch orientation θ and the corresponding rotation matrix R_θ, it is possible to construct a "steered" version S_θ of S:

$$S_\theta = R_\theta S. \tag{50}$$

Consequently, the steered BRIEF operator becomes:

$$g_n(p, \theta) := f_n(p) | (x_i, y_i) \in S_\theta. \tag{51}$$

The angles are discretized to increments of $\frac{2\pi}{30}$ (12 degrees), and a lookup table of precomputed BRIEF patterns is constructed. As long as the keypoint orientation is consistent across views, the correct set of points S_θ will be used to compute its descriptor.

7.2.2 SURF

One of the challenges for the SURF algorithm is to try to reduce its descriptor's dimension and complexity, while keeping it sufficiently distinctive. By mixing crudely localized information and the distribution of gradient related features yields good distinctive power while it fends off the effects of localization errors in terms of scale or space. The use of relative strengths and orientations of gradients reduces the effect of photometric changes. The first step in the construction of a SURF descriptor is to fix a reproducible orientation based on information from a circular region around the interest point. Then, a square region is constructed in alignment with the selected orientation and the SURF descriptor is extracted from it.

The Haar-wavelet responses in x and y directions are calculated for a circular neighborhood of radius $6s$ around the interest point, with s being the scale at which the interest point was detected. The sampling step is scale dependent and chosen to be s. Since integral images are used for computing the wavelet responses, only six operations are needed, regardless of the wavelet size. Once the wavelet responses are calculated and weighted with a Gaussian ($\sigma = 2.5s$) centered at the interest point, the responses are represented as vectors. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window covering an angle of $\frac{\pi}{3}$.radians The horizontal and vertical responses with the window are summed and then yield a new vector. The longest vector found lends its orientation to the interest point.

The construction of the square region for capturing the descriptor components starts by splitting up the window (with size equal to $20s$) into smaller 4×4 square sub-regions. For each sub-region, a few simple features at 5×5 regularly spaced sample points are computed. The wavelet responses are summed up over each sub-region and form a first set of entries to the feature vector. This way, each sub-region has a four-dimensional descriptor vector v as follows:

$$v = (\sum dx, \sum dy, \sum |dx|, \sum |dy|), \tag{52}$$

which corresponds to the summation of horizontal and vertical responses and the summation of the absolute horizontal and vertical responses. The absolute responses are added to the vector in order to bring information about the polarity of the intensity changes. This results in a descriptor vector for all 4×4 sub-regions of length 64.

7.2.3 SIFT

Once there is information available about the location, scale and orientation for each feature, the SIFT descriptor can describe the local image region in a manner invariant to these transformations. In addition, such representation must be robust against small shifts in local geometry, similar to the ones that occur from affine of 3D projections. This robustness to local geometric distortions can be obtained by representing the local image region using multiple images, each representing a number of orientations (also known as orientation planes). Each orientation plane contains only the gradients corresponding to that orientation, with linear interpolation used whenever there are intermediate orientations. Each plane is blurred and resampled in order to support larger shifts in positions of the gradients.

In order to efficiently implement the previous approach, the same precomputed gradients and orientations for the pyramid levels (used before for orientation selection) can be used again. For each detected keypoint, the pixel sampling from the pyramid level in which the feature was detected is taken into consideration. Pixels in a radius of 8 pixels around the keypoint are inserted in the orientation planes.

The global orientation is always measured relative to the one of the keypoint by subtracting the keypoint original orientation. Usually SIFT's default configuration adopts eight orientation planes, each sampled over a 4×4 grid of locations, with a sample spacing equal to the pixel spacing used for gradient detection. The blurring process is achieved by allocating the gradient of each pixel among its eight closest neighbors in the sample grid, using linear interpolation in orientation and the two spatial dimensions.

For sampling the image at a larger scale, the same process is repeated for a second level of the pyramid one octave higher. This time, a 2×2 rather than a 4×4 sample region is used. Using this value, approximately the same image region will be examined at both scales, so that any nearby occlusions will not affect one scale more than the other. Therefore, the total number of samples in the SIFT key vector, from both scales, is $8 \times 4 \times 4 + 8 \times 2 \times 2$ or 160 elements, which gives enough measurements for high specificity.

The matching of SIFT features can be performed by using nearest neighbors search [77]. To speed up the search, a modified version of the k-d tree algorithm, called the best bin first search method, can be applied [78]. The efficiency of the search can also be improved by giving the samples generated at a larger scale twice the weight of those at the smaller scale. This means that the larger scale is in effect able to filter the most likely neighbors for checking at the smaller scale. It also improves recognition performance by giving more weight to the least-noisy scale. Additionally, an efficient way to cluster reliable model hypotheses is to use the Hough transform [79][80]to search for keys that agree to a particular model pose.

7.3 DISCUSSION

Some considerations must be made regarding the feature detectors and descriptors listed before. In both sections, the algorithms are listed according to their complexity, from the simplest to the most complex one. Regardless of its age, the SIFT algorithm is still the one that presents the best repeatability and distinctiveness results, due to the information it takes into consideration for selecting the features and the size of its descriptor. In comparison to the SURF algorithm, SIFT shows less approximations and therefore is more robust to rotations than SURF, as demonstrated in chapter 5. While ORB constructs its descriptor using a string of binary information, the SIFT algorithm stores an array of floats that describe the complete neighborhood of the feature.

In order to take advantage of the particular benefit of each feature detector and descriptor described, this thesis works with different combinations of them. Such detector-extractor pairs will be detailed in the following sections.

7.4 TRACKING ALGORITHMS

This work used five different tracking algorithms so that the proposed model could vary between them. All of them are tracking-by-detection algorithms, which means that the tracking task is based on detection, which enables large displacements of the target object among sequential frames. The tracking-by-detection algorithms were composed of both a feature detector and a feature extractor. Thus, five pairs of algorithms were analyzed (the first algorithm of the pair corresponds to the detector, the second to the extractor): SIFT-SIFT, SURF-SURF, FAST-SIFT, FAST-SURF, and ORB-ORB. The SIFT-SIFT, SURF-SURF and ORB-ORB are well known combinations, while the other two related to the FAST feature detector were chosen so that their performance stayed in the middle of the others. This way, it was possible to have a set of different tracking algorithms distributed based on their performance.

By observing each training sequence separately for each algorithm pair, it was possible to notice the cases where one should be prioritized in place of the others. Figures 17 to 21 show the comparison of SSIM results for the five training sequences generated from the third model (first image of the High Texture object class), given that each of the five algorithms was tested. The SSIM was chosen to be illustrated on the figures, instead of the reprojection error, simply due to the fact that its value is restricted to the [0,1] interval, as described before. This way, the SSIM is easier to visualize. A total of 25 thousand frames were used to generate these images. Table 6 lists the mean SSIM values for each parameter tested.

The way in which SSIM relates to the reprojection error will be detailed in section 6.1.

Table 6. Mean SSIM values for the five parameters tested.

	SIFT-SIFT	SURF-SURF	FAST-SIFT	FAST-SURF	ORB-ORB
Rotation	0.682	0.848	0.237	0.443	0.833
Scale	0.524	0.467	0.285	0.219	0.374
Distortion	0.651	0.572	0.607	0.403	0.514
Luminance	0.583	0.565	0.613	0.557	0.393
Occlusion	0.472	0.49	0.482	0.484	0.461
Overall Mean	**0.583**	**0.588**	**0.445**	**0.421**	**0.515**

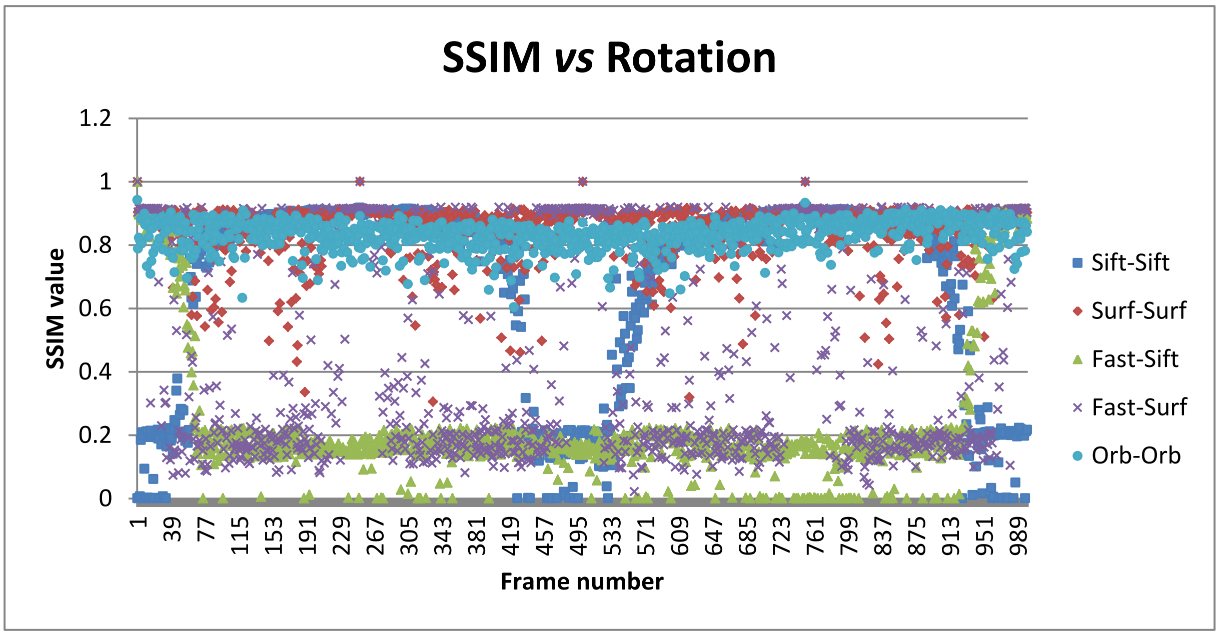


Figure 21. SSIM results for all algorithms, applied on the first image of the High Texture object class and varying the rotation parameter.

From Figure 21, it is noted that the ORB-ORB combination is very robust against object rotations, second only to the SURF-SURF combination, which is far more computationally complex. In this scenario, because of the fact that only rotations are applied to the object being tracked (there are no luminance, occlusion, distortion or scale variations), its texture does not change during the entire sequence. Thus, in this case, one could raise the SSIM threshold and consider as good tracking results the situations in which the SSIM value is higher than 0.6. Chapter 9 shows that when considering an error threshold of 10 pixels, a much lower SSIM threshold needs to be applied.

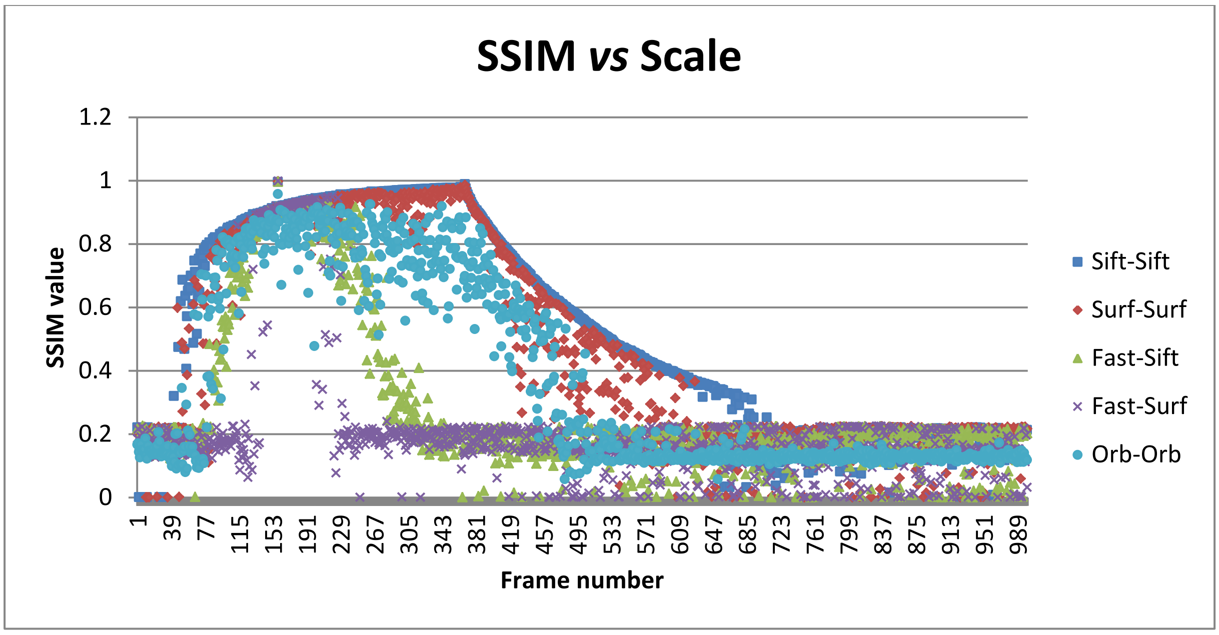


Figure 22. SSIM results for all algorithms, applied on the first image of the High Texture object class and varying the scale parameter.

Regarding the scale scenario, the SIFT-SIFT combination obtained the best results, followed by SURF-SURF and ORB-ORB. As the scale value gets closer to one (100% of the original size) the SSIM value increases. The tolerance for scale variation of both FAST-SIFT and FAST-SURF is very low. As shown in Figure 22, they achieve their best results only for the interval between frames #106 and #246 where the scale variation was between 75 and 141% of the original object size.

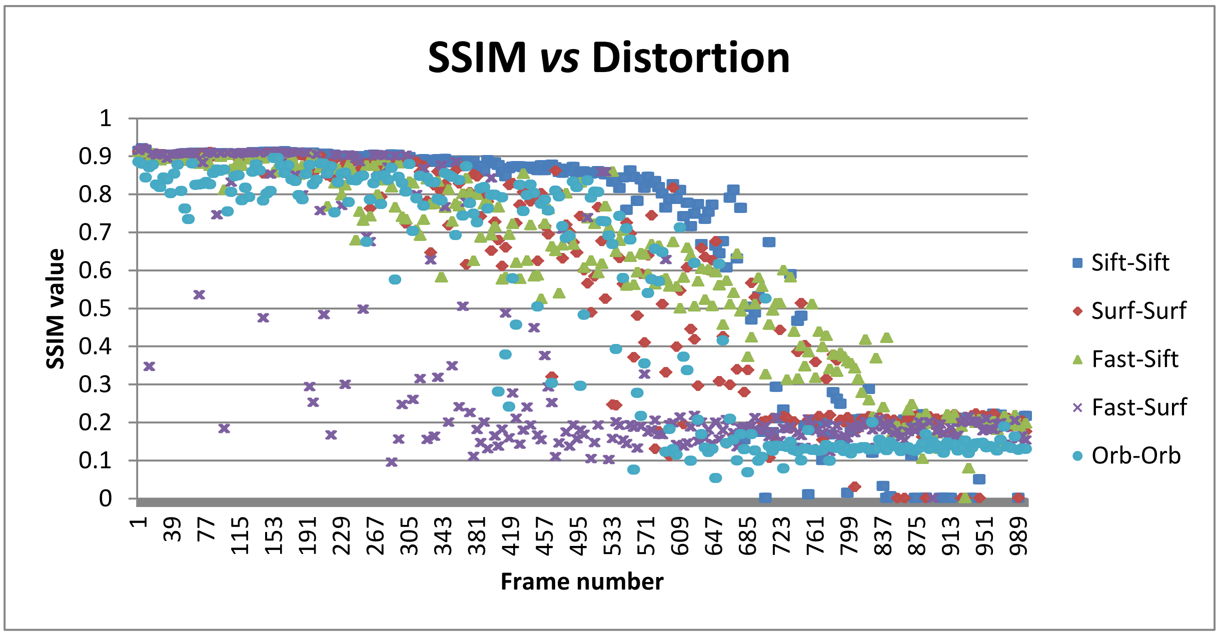


Figure 23. SSIM results for all algorithms, applied on the first image of the High Texture object class and varying the distortion parameter.

According to Table 6, the SIFT extractor seems to be the most robust against image distortions. From Figure 23, it is clear that the best results were generated by the SIFT-SIFT and FAST-SIFT combinations.

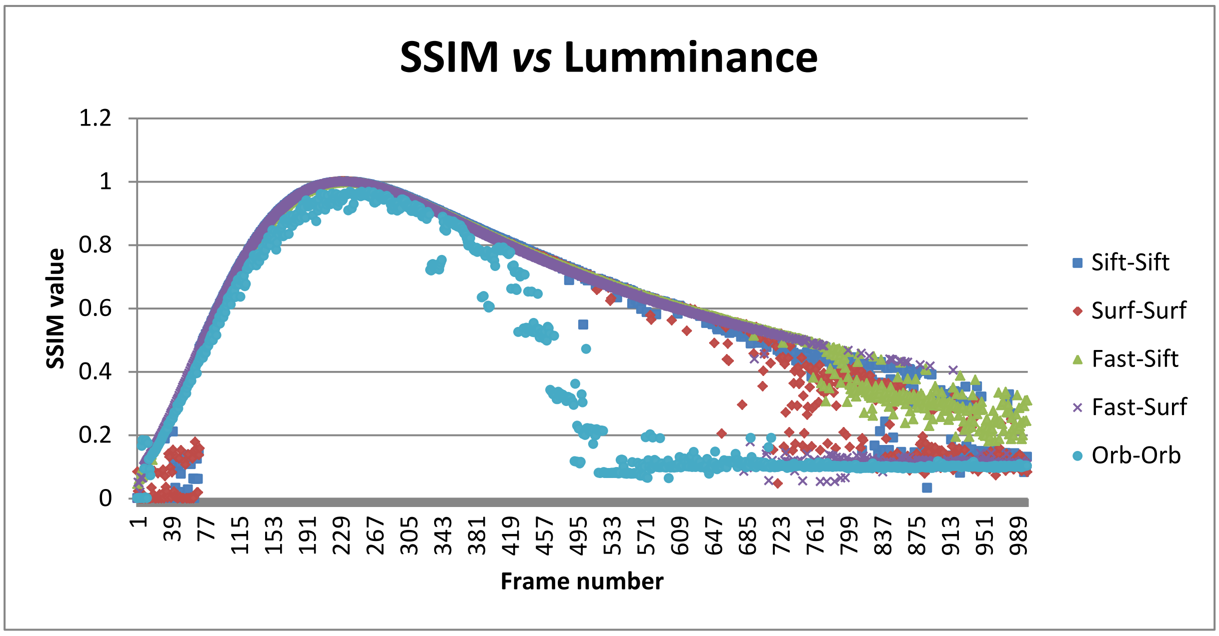


Figure 24. SSIM results for all algorithms, applied on the first image of the High Texture object class and varying the luminance parameter.

Similarly to the scale scenario, Figure 24 shows that all five algorithms tend to show better results when they are closer to the original image luminance. The best results were obtained by the FAST-SIFT combination, followed by SIFT-SIFT and SURF-SURF. The decay in tracking quality regarding the ORB-ORB combination is perceptible in Figure 24, when its SSIM values start to go down earlier than the other algorithms, from frame #316 to #561, where the luminance variation increases from 108 to 186% of the original image brightness.

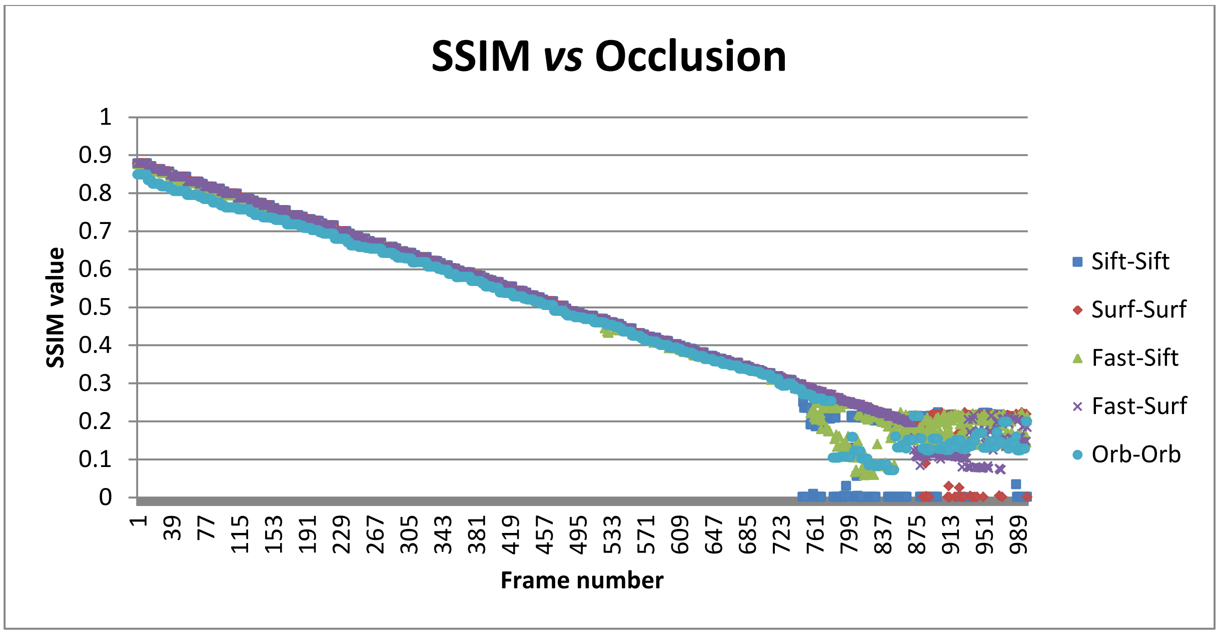


Figure 25. SSIM results for all algorithms, applied on the first image of the High Texture object class and varying the occlusion parameter.

The occlusion scenario represented by Figure 25 showed similar results for all algorithms. Since there is no significant difference between them, this parameter could be easily removed from the model without prejudicing the overall model decision by classification. One important aspect to observe is the impact its removal could have in both speed and quality of tracking results.

CHAPTER 8 DEFINITION OF A TRACKER-SWITCHING DECISION MODEL

Tracking objects using wearable computing platforms can be considered a typical example of trade-off in the computing area. As with many situations, one aspect is lost so that something is gained in return. In our case, the trade-off occurs between computational power (which relates to the speed of the algorithm being executed) and tracking accuracy (which relates to the amount of error after registering the object against the template being search).

In order to create the decision process that controls such a trade-off, there must be a full comprehension of all possible consequences of a particular choice. To perform this process automatically, so that the computer can select autonomously the best algorithm for a specific situation, the computer must be able to decide when to switch between the available algorithms. In the following sections, we describe a model that acts like a decision maker with the goal to preserve quality and improve the processing speed of object tracking applications.

8.1 THE OTS DECISION MODEL

The proposed model can be defined as a "decision model" because it maps the relationships between elements of the decision and the forecasted results in order to better understand or control the tracking problem. In our case, the "elements of the decision" can be considered the model inputs, while the results serve as the model's corresponding outputs.

We propose a general solution to the problem of optimizing the planar object tracking task using texture information without applying significant changes to the computer vision application code. This means that the tracking algorithms themselves are not modified. Instead, different options of algorithms are selected on-the-fly according to the input scene. The approach described here is generic enough as its results depend mainly on the algorithms available. Any set of algorithms can be used, provided that the model information is generated following the same sequence of steps listed in the methodology previously described in section 1.4. The proposed decision model is called OTS, since it works as an Object Tracking Switcher that changes the algorithm in use according to external input that comes from the image frames.

"La fille aux yeux d'or" (the girl with golden eyes) or simply the best reachable model should be able to provide good results when applied to any type of object being tracked. In this thesis, the term "good results" means that quality is maintained according to a threshold and processing speed is improved against the proposed decision model overhead. Unfortunately, object appearance may vary in a way that a simple and concise model would not be simple enough if it has to support every existing object. Because of that fact, there is some effort by the scientific community in trying to classify objects according to their visual features, so that simple models can be created focusing each category. These so called object categories or object classes will be discussed further in sequence.

8.2 MODEL WORKFLOW

The training phase of the model was responsible for acquiring information regarding how each algorithm is supposed to work when facing a specific input. After such a phase, there was available data representing an error approximation for the 5 input parameters analyzed independently. For each parameter, a range between MIN_VALUE and MAX_VALUE was specified. The parameter values were evenly distributed from MIN_VALUE to MAX_VALUE, in a way that their growth was linear. A total of 1000 samples for each pair Parameter/Algorithm were stored for each parameter, containing the corresponding error values. For the tests performed in this work, 25 arrays of 1000 elements each were stored, in order to be later used in the decision process.

It is important to save data separately because this allows us to choose which parameters to take into consideration while performing the algorithm switch decision. In order to simplify the decision model construction, each parameter was trained separately from the others, allowing them to be treated independently during the decision process. As it will be explained, the decision process functions like a cascade of tests.

The test cascade means that each set of parameters extracted in real time has to pass through five different evaluations (one for each parameter), so that the algorithm is considered a valid option. At first, all five algorithms are inserted on a queue according to their processing demand, from the simplest to the most complex one (e.g., from ORB-ORB to SIFT-SIFT). The time required for each algorithm to process a single image feature was shown in chapter 4. The order of the test cascade is also influenced by the training phase. It is ordered from the most restrictive to the less one. The restrictive grade of an algorithm is calculated based on the amount of frames that do not pass the test. It was possible to perceive the following order from the training phase:

Light > Scale > Distortion > Occlusion > Rotation

By applying the most restrictive tests first it is possible to optimize the entire testing phase, since more algorithms are eliminated and fewer tests have to be performed. After each test, the algorithms that show an error higher than the specified error threshold are eliminated from the algorithm queue. In the end, the first algorithm that remains on that queue is chosen by the decision model. In case there is no remaining algorithm on the queue, the model outputs the most complex one. This happens due to the fact that the most complex algorithm has a higher probability of finding the object missing on the frame sequence.

Figure 26 illustrates the OTS workflow from input image acquisition to algorithm response by the model. The process starts by extracting the model parameters used as input to the decision model. The results from the last successful frame tracked are used in tandem with the template information to obtain both movement (scale, distortion and rotation) and texture-related (occlusion, luminance) parameters. After that, the obtained values are passed to a cascade of tests, each of them corresponding to a specific parameter.

For every test, a set of available tracking algorithms is passed as input, along with the tracking threshold to be used. The model accesses the table of values generated by the training phase and gives as output all the algorithms that satisfy the given threshold. In the example of Figure 26, the first test, regarding luminance, receives the five available algorithms and gives as output only the four algorithms that satisfied the threshold. In this case, the FAST-SURF failed to reach the desired quality and will not be used as input to the following tests. Since the FAST-SIFT algorithm did not pass the Scale test, it was also removed from the set of possible algorithm choices.

The following test, for distortion, removed the SURF-SURF algorithm, leaving only two possible options available: SIFT-SIFT and ORB-ORB. Both remaining algorithms passed Occlusion and Rotation tests. In the end, these two algorithms were able to satisfy the quality threshold for all tests. Since the ORB-ORB combination corresponds to the less complex algorithm that satisfies the desired constraints, it is given as output of the decision model.

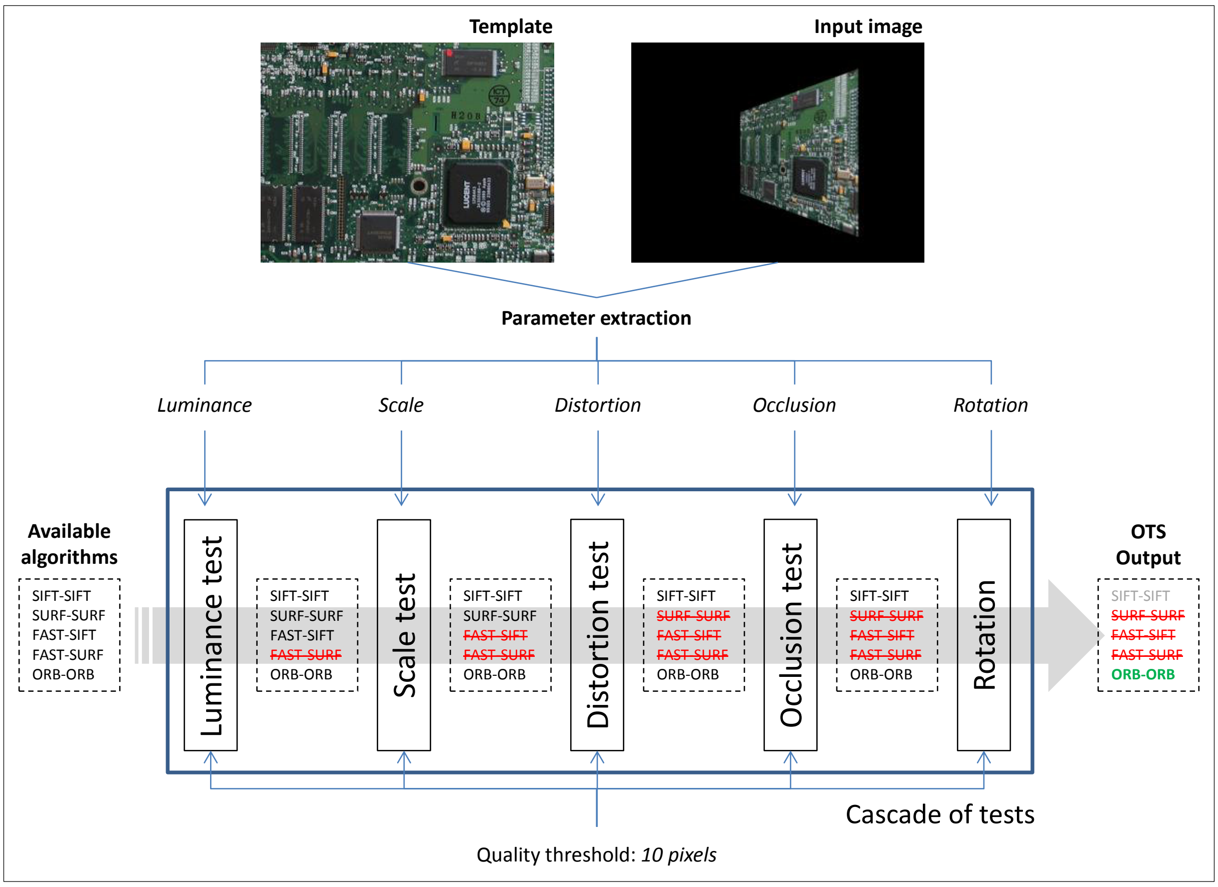


Figure 26. Diagram for the OTS workflow.

The OTS will be evaluated and the results obtained with the proposed approach will be discussed in the next chapter.

CHAPTER 9 VALIDATION OF THE DECISION MODEL

The execution platform for both generating the data for the training phase and validating the proposed scheme was an Intel Core i7 – 3720QM CPU (@2.6GHz), with 16 GB of RAM memory and running Windows 8 Professional 64 bits. Despite the fact this CPU has 4 physical processors (8 logical ones), a single one was used due to the original implementation of the algorithms that comes from the single-threaded OpenCV library [94].

The OpenCV implementation of the algorithms (both feature detectors and extractors), follows the FeatureDetector and DescriptorExtractor interfaces, which facilitates the testing process. It is important to note that some implementations provided by OpenCV are not fully optimized versions of the algorithms.

This chapter performs an analysis regarding the speedup obtained with the proposed model and the effect it has over the tracking quality. It shows how each of the four generated models, corresponding to the four planar object classes discussed in section 5.2, behaves when applied to different classes of objects, in respect to both speedup and error. After performing a more complete time analysis of the entire process, including the parameter extraction phase, a selective scheme of the model is also proposed, in order to further diminish its computational load.

9.1 SPEEDUP AND PRECISION RESULTS

The created models were tested against both training and evaluation sequences, in order to assess the speedup obtained and their behaviour when applied to objects from other classes. Four different models were constructed, one for each texture class. Each of the four models was based on the first images of each object class, Low Texture, High Texture, Normal Texture and Repetitive Texture, representing models #1, #3, #5 and #7, respectively. Despite being only four models, they were named so it is easier to reference their original image sequence.

It is also relevant that, due to the nature of the algorithms, which are mainly based on texture processing, objects with low texture will not have the same tracking quality as the other object classes and should behave as outliers.

The speedup results were calculated as described in section 6.1 and they are shown in Table 7. In each column, the result with the best speedup is highlighted. The second model created, based on the first image of the High Texture object class, showed the greatest results for most of the cases, as listed in the last column of the table. The only scenarios in which the top model is not the best performer were those in which it was applied to both images of the Low Texture object class, but only by a very small difference. Such a result means that the second model is more general to support all other image scenarios and should be used when the application being developed focus no particular image template.

Table 7. Speedup obtained with the proposed approach. The four different models generated were applied to each of the eight available scenarios.

	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**Mean**
#1 (low texture)	1.18	1.28	1.23	1.23	1.20	1.17	1.22	1.16	1.21
#3 (high texture)	3.31	3.26	2.91	2.99	3.25	3.16	2.94	3.15	3.12
#5 (normal texture)	3.33	3.27	2.53	2.59	3.12	2.67	2.53	2.44	2.81
#7 (repetitive texture)	3.00	2.92	2.75	2.76	2.88	2.86	2.77	2.96	2.86

The error rates obtained are shown in Table 8. They result from applying an error threshold of 10 pixels. As expected, the fewest error situations happened when the specific models where applied to their respective sequences, as can be observed for image sequences number #1, #3, #5 and #7.

Table 8. Error rate obtained with the proposed approach. The four different models generated were applied to each of the eight available scenarios. The error threshold used was 10 pixels. The first line shows the original error values, without using the proposed approach (we consider the original error values the ones from the SIFT-SIFT combination).

	82.86%	14.24%	20.06%	12.40%	26.24%	12.82%	22.44%	22.34%
	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**
#1	66.18%	14.20%	18.16%	12.66%	25.60%	12.92%	13.34%	21.80%
#3	74.60%	20.14%	15.82%	14.90%	21.78%	19.64%	19.82%	25.52%
#5	74.38%	15.42%	19.26%	14.84%	15.04%	15.80%	13.98%	27.34%
#7	74.30%	15.22%	17.50%	13.98%	17.92%	16.76%	11.68%	26.06%

9.2 VARYING THE ERROR THRESHOLD

One of the advantages of the proposed decision model is that it can be used, even after constructed, in tandem with different threshold values. For example, the model based on image sequence number #3, which presented the best speedup results, was used to show how the speedup varies according to the error threshold change. The evolution of speedup is shown in Figure 27. As expected, it increases as the threshold value increases, as less complex algorithms are used for a longer period of time during tracking.

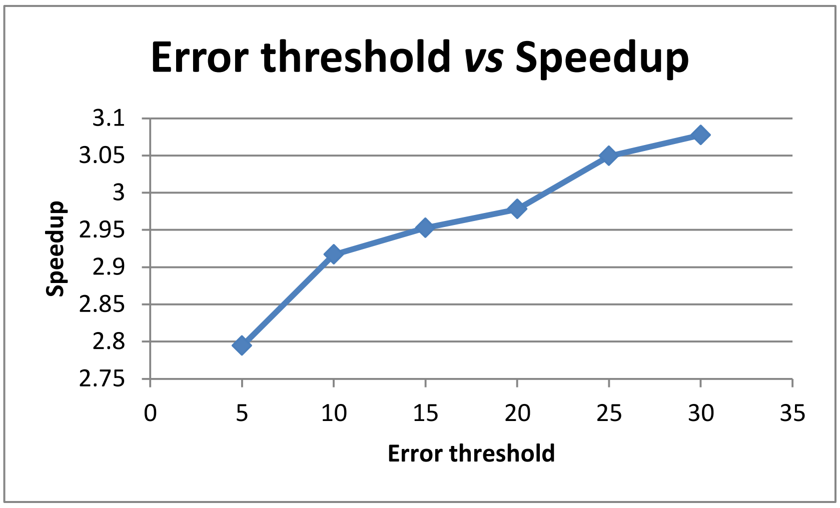


Figure 27. Analysis of speedup evolution according to error threshold increase. The highest relative increase in speedup occurs between error interval [5, 10].

It is important to notice that the speedup increase is not linear if compared to the threshold error increase. This means that at some point it is not advisable anymore to keep increasing the threshold, since the losses with error will be higher than the gains in speed. It is suggested that error threshold is kept at a maximum of 10 pixels, since this value shows the higher variation compared to the previous threshold value.

9.3 TIME ANALYSIS

The proposed model works based on parameters that come from the input image frames. While the mean execution time of the algorithms is known, the time necessary to extract such parameters must be known as well. With this information, it will be possible to identify when it is advantageous to use the proposed approach. The only additional overhead in processing that comes from the scheme proposed regards the parameter extraction phase, error evaluation (SSIM computation) and the decision cost itself. The time involved in the execution of the decision is not significant, since for each frame, just a few memory accesses are needed (e.g., for the tested scenario when five algorithms and five parameters are used, there were only 25 memory accesses).

The mean time for each frame to extract the parameters is shown in Table 11. The rotation and scale parameters extraction time were grouped together since they are part of the same process.

Table 9. Mean time spent per frame in each processing step.

	Execution times (ms)
Rotation + Scale	0.0107640
Distortion	0.000202
Luminance	0.1623920
Occlusion	4.0899950
SSIM	3.1674500

It can be observed from Table 9 that the computation times for both occlusion parameter and error evaluation (SSIM) are orders of magnitude higher than the other parameters (rotation + scale, distortion and luminance). While some of them just require a few arithmetic calculations, those two parameters involve processing the entire object image, making comparisons against the reference image and using additional algorithms (as happens to occlusion calculation, using LBP calculations). This time analysis is relevant because even if the decision model processing can be executed in a parallel thread, the occlusion parameter calculation demands about 55.04% of the entire model processing time, while the SSIM calculation takes 42.62% of it. Because SSIM is used in the error evaluation, it cannot be removed. The only alternative, when necessary, is to make a selective decision, by using only selected parameters, instead of always employing all parameters.

9.4 SELECTIVE SPEEDUP AND PRECISION RESULTS

A different way of adopting the proposed model is to specifically select which parameters will be used to evaluate the algorithm decision. As shown in the previous section, the occlusion parameter calculation is expensive if compared to the others, so it can be necessary to remove it from the decision process. This means that it does not need to be extracted in real time, because it is not being considered anymore. This change will surely have an impact on both speedup and error results compared to the previous attempts. Since the model is now less restrictive (i.e., performs less tests in the decision process), it should show an increase in speedup in detriment of a higher error. Table 10 and Table 11 show the new speedup and error values obtained by using the model without considering the occlusion parameter evaluation.

Table 10. Speedup obtained with the selective approach (not considering the occlusion parameter). The four different models generated were applied to each of the eight available scenarios.

	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**Mean**
#1	1.82	1.82	1.81	1.81	1.81	1.81	1.81	1.82	1.81
#3	3.43	3.38	3.24	3.26	3.37	3.37	3.35	3.43	3.35
#5	3.70	3.55	3.40	3.41	3.49	3.49	3.42	3.70	3.52
#7	3.16	3.06	2.95	2.95	3.03	3.03	2.97	3.16	3.04

Table 11. Error rate obtained with the selective approach. The four different models generated were applied to each of the eight available scenarios. The error threshold used was 10 pixels. The first line shows the original error values, without using the proposed approach (only the SIFT-SIFT combination is used).

	82.86%	14.24%	20.06%	12.40%	26.24%	12.82%	22.44%	22.34%
	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**
#1	61.84%	14.40%	18.06%	12.48%	18.20%	12.90%	12.74%	14.10%
#3	74.52%	19.88%	18.82%	15.38%	22.40%	21.32%	20.80%	27.52%
#5	74.00%	16.46%	20.46%	18.04%	17.82%	17.42%	16.10%	29.10%
#7	74.34%	14.62%	17.62%	15.68%	18.08%	18.14%	13.42%	26.46%

After the modification, the model generated by image sequence number #5 obtained the new best speedups. Besides that, by comparing both original and new speedup results and doing the same regarding error, it is possible to verify a mean increase in speedup of 17.21% and a mean error increase of just 0.83%. For this reason, the removal of the occlusion test showed to be very promising, since it enhances more the speedup than it harms tracking quality.

This chapter presented the results obtained with the decision model proposed, using the methodology described in chapter 4. We showed that both speedup and error values vary according to the model choice. In general, it is possible to obtain a speedup of a factor of 3, while keeping the error below a specific threshold (10 pixels). We also showed that the execution time of the model decision process can be optimized by not considering all parameters while searching for the best suitable tracking algorithm. By eliminating the most expensive parameter calculation (occlusion), we were able to improve the speedup while still maintaining the error bellow the initial threshold.

CHAPTER 10 CONCLUSION

This thesis had as its main objective to propose a decision model capable of optimizing the tracking quality vs. computational demand by choosing the most suitable algorithm according to the object being tracked. To accomplish this, different tracking algorithms, image features, similarity measures and image datasets had to be carefully analyzed.

The problem of performance optimization in AR applications operating in constrained mobile platforms was addressed. Sometimes, due to the lack of computational resources, the application execution is prohibitive because the computational load of the algorithm does not match the available processing power, making real-time processing difficult.

The proposed decision model was created based on the correlation between five different algorithms. Whenever the scenario also involves memory consumption, the number of algorithms supported by the model can be diminished. For example, one may use the proposed decision model to select when is advisable to switch from SIFT-SIFT (the most expensive) combination to ORB-ORB (the least expensive), and then when to come back to the initial setup. Fewer algorithms supported by the model imply fewer decision options and reduced memory need. What happens in this case is that the computational load changes abruptly from these two combinations. When more algorithms are available, the change is more subtle, and a higher speedup can be achieved.

10.1 CONTRIBUTIONS

This thesis gave birth to several different contributions, listed in the following sections.

10.1.1 COMPARISON AMONG DIFFERENT SIMILARITY MEASURES

Image registration is the process of overlaying two or more images of the same scene taken at different times, from different viewpoints, and/or by different sensors [91]. The goal with registration is to geometrically align two images (the reference and the captured images). In this work, in order to select a utility function good enough to correctly judge the decision of the model, a broad research project was undertaken. It started with a particular family of image registration methods, namely those that are area-based. These methods deal with the images without attempting to detect salient objects, trying to establish a correspondence by considering the image information in its entirety. Twelve different distance measures were analyzed, the majority of them using histogram comparisons. From those twelve, the mutual information was the only one robust to illumination changes. Due to its complexity and instability when dealing with structural distortions on the image, it was also discarded. In the end, the SSIM value was used as utility function, due to its robustness when comparing structural, brightness and intensity information from the image.

10.1.2 COMPARISON AMONG DIFFERENT TRACKING ALGORITHMS

Chapter 2 and 3 discussed several algorithms that can be used in object tracking applications. It is possible to use this information as a guide for deciding which algorithm to use according to the desired application scenario. Chapter 3 focused on algorithms that can be used in texture-based tracking approaches and the methods currently showing the best results were described and analyzed. A deeper analysis regarding their performance while varying the model parameters can be found in Chapters 5 and 6. The study regarding tracking algorithms in this thesis gave birth to an improvement of an existing tracking algorithm, known as FERNS[105]. Usually, the processing power of the GPU is explored in order to improve tracking algorithms whenever such resources are available [106][107].

10.1.3 ANALYSIS OF THE RELATIONSHIP AMONG ERROR MEASURES

In order to use SSIM as utility function for the proposed model, it was necessary to find a clear relationship between both SSIM and the reprojection error. While the meaning of a SSIM is only clear when it is 1, which represents two completely equal images being compared, it is not clear how different the images are for values different than one. While the reprojection error is an amount measured in pixels, the SSIM value does not have a unit and varies on the interval [0,1]. In order to comprehend how the SSIM values worked, a direct relationship between these two error measurements was studied.

Since one must know ground truth data before utilizing the reprojection error, this was not possible to do with unknown data. For that reason, all 25k frames from the training phase of the model were used in order to acquire error measurements information, SSIM and the reprojection error. It is obvious that when the reprojection decreases, the SSIM value tends to get closer to 1. The main problem here is how to represent a SSIM value that corresponds to a specific reprojection error of, for example, 10 pixels. The SSIM value was varied from 0 to 1 using intervals of 0.001, in a way that 1000 different SSIM values were generated. After that, a spatial hash-like structure was computed so that it was possible to know the mean reprojection error for each of the SSIM values. After that, a function was implemented so that it received as input the reprojection error and provides as output the corresponding SSIM value along with its confidence interval. This function is used by the model and is also available on the OTS library.

10.1.4 THE PROPOSED DECISION MODEL

The proposed model was generated and tested with success against synthetic data. It showed a significant improvement over the original execution environment without modifying the tracking algorithms, just by switching them; it was possible to obtain a mean improvement in performance of about 3 times.

The results were satisfactory for the four object classes tested and the selective capability of the model allows it to be completely configurable by the user, deciding which tests will contribute to the algorithm decision. Since the model is completely modular, new parameters can easily be added to it. This work does not require the user to correctly classify the object he/she intends to track into one of the four object classes. Indeed, he/she can test the four models generated against the available input data to see which gives the best results. In fact, this could be used as an automatic classification process for determining the class of the object.

10.1.5 OBJECT TRACKING SWITCHER LIBRARY

The proposed decision model was fully implemented using the C++ language and is available as an open-source library at the link *http://sourceforge.net/p/otslibrary*. It can be customized according to the application needs and also modified to support different algorithms that do not comply with OpenCV object tracking interfaces.

10.1.6 PUBLICATIONS DIRECTLY RELATED TO THIS THESIS

This thesis produced some contributions in the form of academic publications that are directly related to the object of study. They are listed as follows:

- Teixeira, João Marcelo Xavier Natário; Kelner, Judith; Teichrieb, Veronica. A planar object tracking approach robust to total occlusion. In: Workshop de Realidade Virtual e Aumentada, 2012. Anais do WRVA 2012, 2012.
- Emerenciano, T.; Tenório, L.; Moura, Guilherme; Teixeira, João Marcelo; Almeida, Gabriel; Kelner, Judith. Um Protocolo para Auxílios Visuais em Aplicações de Realidade Aumentada Colaborativa. In: 14th Symposium on Virtual and Augmented Reality, 2012, Niterói. Proceedings of the 14th Symposium on Virtual and Augmented Reality, 2012.
- Santos, R.; Silva, Daliton; Teixeira, João Marcelo; Kelner, Judith. Rastreamento de Objetos Usando Template Matching e Variação de Iluminação. In: 14th Symposium on Virtual and Augmented Reality, 2012, Niterói. Proceedings of the 14th Symposium on Virtual and Augmented Reality, 2012.
- Vasconcelos, L.; Breyer, Felipe; Teixeira, João Marcelo Xavier Natário; Kelner,

Judith; Teichrieb, Veronica. Desenvolvimento de uma Plataforma Móvel para Inspeção Termal de Equipamentos Baseada em Realidade Aumentada. In: ERGODESIGN USIHC, 2011, Manaus. Proceedings of the USIHC 2011, 2011.

- Teixeira, Joao Marcelo; Teichrieb, Veronica; Kelner, Judith. The influence of partial occlusion on object tracking 2011. In V Latin American Summer School on Robotics, Santiago, Chile (Poster).
- Leão, C. W. M.; Reis, B.; Teixeira, João Marcelo; Kelner, Judith. Mivamain: A Platform for Assisting Maintenance with Augmented Reality 2011. In V Latin American Summer School on Robotics, Santiago, Chile (Poster).
- Teixeira, João Marcelo Xavier Natário; Reis, Bernardo; Teichrieb, Veronica; Kelner, Judith. An optimized label-broadcast parallel algorithm for connected components labeling. In Programmable Logic Conference (SPL), 2010, Ipojuca. p. 99-104.
- Reis, Bernardo; Oliveira Filho, P. S. B.; Vasconcelos, L.; Teixeira, João Marcelo Xavier Natário; Teichrieb, Veronica; Kelner, Judith. MarkerMatch: an Embedded Augmented Reality case study. In: Symposium on Virtual and Augmented Reality, 2010, Natal. Proceedings of the SVR 2010, 2010.
- Reis, Bernardo; Teixeira, João Marcelo Xavier Natário; Albuquerque, E.; Teichrieb, Veronica; Kelner, Judith. Análise de técnicas de limiarização adaptativa para realidade aumentada embarcada. In: Workshop de Realidade Virtual e Aumentada, 2010, São Paulo. Anais do WRVA 2010, 2010.
- Leão, Crystian Wendel M.; Teixeira, João Marcelo Xavier Natário; Albuquerque, E.; Teichrieb, Veronica; Kelner, Judith. Melhorando o desempenho do rastreamento de pontos de interesse em imagens através do paralelismo em GPU. In: Workshop de Realidade Virtual e Aumentada, 2010, São Paulo. Anais do WRVA 2010, 2010.
- Teixeira, João Marcelo; Roberto, R. A.; Simões, F. P. M.; Teichrieb, Veronica; Kelner, Judith. Reconstrução 3D usando luzes estruturadas. Tendências e Técnicas. In Symposium on Virtual Reality, 2010.
- Farias, Thiago; Teixeira, João Marcelo; Leite, Pedro; Almeida, Gabriel; Almeida, Mozart Williams S.; Teichrieb, Veronica; Kelner, Judith. High Performance Computing: CUDA as a Supporting Technology for Next Generation Augmented Reality Applications. Revista de Informática Teórica e Aplicada, v. XVI, p. 63-88, 2009.
- Farias, Thiago; Teixeira, João Marcelo Xavier Natário; Almeida, Gabriel Fernandes de; Leite, Pedro; Teichrieb, Veronica; Kelner, Judith. A CUDA-enabled KLT Tracker for High Definition Images. In: Symposium on Virtual and Augmented Reality, 2009, Porto Alegre. Anais do SVR2009, 2009.
- Teixeira, João Marcelo Xavier Natário; Teichrieb, Veronica; Kelner, Judith. Finding

an adequate escape pod to real time Augmented Reality applications. In: Brazilian Symposium on Computer Graphics and Image Processing, 2009, Rio de Janeiro. Anais do SIBGRAPI 2009, 2009.

- Reis, Bernardo; Teixeira, João Marcelo Xavier Natário; Teichrieb, Veronica; Kelner, Judith. Detecção de Marcadores para Realidade Aumentada em FPGA. In: Brazilian Symposium on Computer Graphics and Image Processing, 2009, Rio de Janeiro. Anais do SIBGRAPI 2009, 2009.
- Reis, Bernardo; Teixeira, João Marcelo Xavier Natário; Teichrieb, Veronica; Kelner, Judith. Detecção de Marcadores para Realidade Aumentada em FPGA. In: Simpósio em Sistemas Computacionais - Workshop de Iniciação Científica, 2009, São Paulo. Anais do WSCAD-WIC 2009, 2009.

10.1.7 Publications Indirectly Related to this Thesis

This thesis produced some contributions in the form of academic publications that are indirectly related to the object of study. They are listed as follows:

- Leite, Pedro; Teixeira, João Marcelo; Farias, Thiago; Reis, Bernardo; TEICHRIEB, Veronica; Kelner, Judith. Nearest Neighbor Searches on the GPU. International Journal of Parallel Programming, v. 40, p. 313-330, 2012; ISSN/ISBN: 08857458;
- Santos, Artur; Teixeira, João Marcelo; Farias, Thiago; Teichrieb, Veronica; Kelner, Judith. Understanding the Efficiency of kD-tree Ray-Traversal Techniques over a GPGPU Architecture. International Journal of Parallel Programming, v. 40, p. 331-352, 2012; ISSN/ISBN: 08857458.
- Reis, Bernardo; Teixeira, João Marcelo; Breyer, Felipe; Vasconcelos, Luis Arthur; Cavalcanti, Aline; Ferreira, André; Kelner, Judith. Increasing kinect application development productivity by an enhanced hardware abstraction. In: The 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2012, Copenhagen. Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '12. New York: ACM Press, 2012. p. 5-14.
- Teixeira, João Marcelo; Reis, Bernardo; Macedo, Samuel; KELNER, Judith. Open/Closed Hand Classification Using Kinect Data. In: 14th Symposium on Virtual and Augmented Reality (SVR), 2012, Rio de Janiero. Proceedings of the 14th Symposium on Virtual and Augmented Reality, 2012. p. 18-25.
- Reis, Bernardo; Teixeira, J. M. X. N.; Kelner, Judith. An open-source tool for distributed viewing of kinect data on the web. In: Workshop de Realidade Virtual e Aumentada, 2011, Uberaba. Anais do Workshop de Realidade Virtual e Aumentada, 2011.

- Teixeira, João Marcelo Xavier Natário; Albuquerque, E.; Santos, Artur Lira dos; Teichrieb, Veronica; Kelner, Judith. Improving ray tracing anti-aliasing performance through image gradient analysis. In: Simpósio em Sistemas Computacionais, 2010, Petrópolis. Anais do WSCAD-SSC 2010, 2010.
- Souza, Eduardo Antonio; Vasconcelos, L.; Teixeira, João Marcelo Xavier Natário; Albuquerque, E.; Kelner, Judith; Teichrieb, Veronica. Evaluating the CapCam: a device for thermal inspection of electrical equipment. In: Workshop de Realidade Virtual e Aumentada, 2010, São Paulo. Anais do WRVA 2010, 2010.
- Santos, Artur Lira dos; Teixeira, João Marcelo Xavier Natário; Farias, Thiago; Teichrieb, Veronica; Kelner, Judith. kD-Tree Traversal Implementations for Ray Tracing on Massive Multiprocessors: a Comparative Study. In: International Symposium on Computer Architecture and High Performance Computing, 2009, São Paulo. Anais do SBAC-PAD2009. New York: IEEE Computer Society, 2009. p. 41-48.
- Leite, Pedro; Teixeira, João Marcelo Xavier Natário; Farias, Thiago; Teichrieb, Veronica; Kelner, Judith. Massively Parallel Nearest Neighbor Queries for Dynamic Point Clouds on the GPU. In: International Symposium on Computer Architecture and High Performance Computing, 2009, São Paulo. Anais do SBAC-PAD2009. New York: IEEE Computer Society, 2009. p. 19-25.
- Figueiredo, Lucas Silva; Teixeira, João Marcelo Xavier Natário; Teichrieb, Veronica; Kelner, Judith. An Open-Source Framework for Air Guitar Games. In: Brazilian Symposium on Games and Digital Entertainment, 2009, Rio de Janeiro. Anais do SBGames 2009, 2009. p. 74-82.
- Carvalho Jr., A.; Farias, Thiago; Teixeira, Joao Marcelo; Teichrieb, Veronica; Kelner, Judith. Aplicando Model-Driven Development à Plataforma GPGPU. In: Simpósio em Sistemas Computacionais, 2009, São Paulo. Anais do WSCAD-SSC 2009, 2009.

10.2 FUTURE WORKS

This research has raised many possibilities of use for the proposed decision model. Some alternatives to enhance the work produced by this thesis are also explained bellow.

- To test the model with real data and evaluate its performance in a real-time application will be important in proving the validity of the concepts developed during this work. Testing is already being carried out using two sponsored projects that make use of mobile platforms for executing AR applications.
- A detailed time analysis regarding the impact of the model utilization over applications is also planned. This will identify cases in which the proposed decision model should run in a separate thread, so that the application

execution will not be harmed.

- The definition of a tracking interface generic enough to represent both model-based and texture-based trackers would allow the inclusion of new algorithms to the model in an easy way.

Finally, we plan to extend the proposed model to track 3D objects. This topic involves a series of new studies since different motion parameters can be taken into consideration. An inverse approach of the parameter estimation can be used, by considering camera pose parameters instead of only object parameters.

REFERENCES

[1] Richard Szeliski. 2010. Computer Vision: Algorithms and Applications (1st ed.). Springer-Verlag New York, Inc., New York, NY, USA.

[2] Yilmaz, A., Javed, O., and Shah, M. Object tracking: A survey. ACM Comput. Surv., 38, December 2006.

[3] Czyzewski, A., Dalka, P. "Examining Kalman Filters Applied to Tracking Objects in Motion", Image Analysis for Multimedia Interactive Services, 2008. WIAMIS '08. Ninth International Workshop on , vol., no., pp.175-178, 7-9 May 2008.

[4] Huang, C., Wu, B., and Nevatia, R. Robust Object Tracking by Hierarchical Association of Detection Responses. In Proceedings of the 10th European Conference on Computer Vision: Part II (ECCV '08), David Forsyth, Philip Torr, and Andrew Zisserman (Eds.). Springer-Verlag, Berlin, Heidelberg, 788-801.

[5] Zhou, Q., Aggarwal, J.K. "Object tracking in an outdoor environment using fusion of features and cameras". Image Vision Comput. 24(11): 1244-1255 (2006).

[6] Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[7] North, D.W. "A Tutorial Introduction to Decision Theory," Systems Science and Cybernetics, IEEE Transactions on , vol.4, no.3, pp.200-210, Sept. 1968.

[8] Lepetit, V. and Fua, P. "Monocular model-based 3d tracking of rigid objects: A survey". In Foundations and Trends in Computer Graphics and Vision, pages 1-89, 2005.

[9] Wang, T., Gu, I.Y.H., Shi, P. Object Tracking using Incremental 2D-PCA Learning and ML Estimation. Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on , vol.1, no., pp.I-933-I-936, 15-20 April 2007

[10]Wang, Y., Van Dyck, R. E., and Doherty, J. F. Tracking moving objects in video sequences. Conference on Information Sciences and Systems. Vol. 2. 2000.

[11]Teixeira, J. M. X. N., Kelner, J., Teichrieb, V. A planar object tracking approach robust to total occlusion. In: Workshop de Realidade Virtual e Aumentada. Anais do WRVA 2012, Paranavaí, Paraná.

[12]Mei, C., Benhimane, S., Malis, E., and Rives, P. "Efficient homography based tracking and 3-d reconstruction for single-viewpoint sensors". IEEE Transactions on Robotics, 24(6):1352-1364, Dec. 2008.

[13]Zhong, Y. and Jain, A. K. "Object tracking using deformable templates". IEEE Transactions on Pattern Analysis and Machine Intelligence, 22:544-549, 2000.

[14]Cedras, C. and Shah, M. "Motion-Based Recognition: A Survey". IVC, 13(2):129-155, March 1995.

[15]Javed, O. and Shah, M. Tracking and object classification for automated surveillance. In Proceedings of the 7th European Conference on Computer Vision-Part IV, ECCV '02, pages 343-357, London, UK, UK, 2002. Springer-Verlag.

[16]Visser, R., Sebe, N., and Bakker, E. Object recognition for video retrieval. In Proceedings of the International Conference on Image and Video Retrieval, pages 262-270, 2002.

[17]Ren, J., Vlachos, T., and Argyriou, V. Immersive and perceptual human-computer interaction using computer vision techniques, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010,pages 66-72. 2010.

[18]Gloyer, B., Aghajan, H. K., Siu, K., and Kailath, T. "Vehicle detection and tracking for freeway traffic monitoring". In Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on, volume 2, pages 970 -974 vol.2, oct-2 nov 1994.

[19]Jia, Z., Balasuriya, A., and Challa, S. "Vision based target tracking for autonomous land vehicle navigation: A brief survey". Recent Patents on Computer Science, 2(1):32-42, 2009.

[20]Heisele, B., Woehler, C. Motion-based recognition of pedestrians. Proceedings of the Fourteenth International Conference on Pattern Recognition, vol.2, no., pp.1325-1330 vol.2, 16-20 Aug 1998.

[21]Gagnon, L., Foucher,S., Laliberte, F., Lalonde, M., Beaulieu, M. Toward an Application of Content-Based Video Indexing to Computer- Assisted Descriptive Video, Computer and Robot Vision, 2006. The 3rd Canadian Conference on , vol., no., pp.8, 0-0 0.

[22]Reis, B., Teixeira, J. M. X. N., Breyer, F., Vasconcelos, L. A., Cavalcanti, A., Ferreira, A., Kelner, J. Increasing kinect application development productivity by an enhanced hardware abstraction. In: The 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2012, Copenhagen. Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '12. New York: ACM Press, 2012. p. 5-14.

[23]Kun, A.J., Vamossy, Z. Traffic monitoring with computer vision, 7th International Symposium on Applied Machine Intelligence and Informatics, 2009. SAMI 2009. pp.131-134, 30-31 Jan. 2009.

[24]Pinto, C. "How Autonomous Vehicle Policy in California and Nevada Addresses Technological and Non-Technological Liabilities". Intersect: The Stanford Journal of Science, Technology and Society; Vol 5 (2012).

[25]Azuma, R. T. "A survey of augmented reality". Presence: Teleoperators and Virtual

Environments, 6(4):355-385, August 1997.

[26]Zhang, X., Gunther, M., and Bongers, A. Real-time organ tracking in ultrasound imaging using active contours and conditional density propagation. In Proceedings of the 5th international conference on Medical imaging and augmented reality, MIAR'10, pages 286-294, Berlin, Heidelberg, 2010. Springer-Verlag.

[27]Henderson, S. J. and Feiner, S. K. Augmented reality in the psychomotor phase of a procedural task. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11, pages 191-200, Washington, DC, USA, 2011. IEEE Computer Society.

[28]Wither, J., DiVerdi, S., and Hollerer, T. "Annotation in outdoor augmented reality". Computers & Graphics, 33(6):679-689, December 2009.

[29]Lyu, M. R., King, I., Wong, T. T., Yau, E., and Chan, P. W. Arcade: Augmented reality computing arena for digital entertainment. 2005 IEEE Aerospace Conference, pages 1-9, 2005.

[30]Narzt, W., Pomberger, G., Ferscha, A., Kolb, D., Muller, R., Wieghardt, J., Hortner, H. and Lindinger, C. "Augmented reality navigation systems". Univers. Access Inf. Soc., 4:177-187, February 2006.

[31]Teichrieb, V., Lima, J. P. S. M., Apolinario, E. L., Farias, T. S. M. C., Bueno, M. A. S., Kelner, J. and Santos, I. H. F. "A Survey of Online Monocular Markerless Augmented Reality". International Journal of Modeling and Simulation for the Petroleum Industry, vol. 1, no. 1, August 2007.

[32]Lin, L., Wang, Y., Liu, Y., Xiong, C., and Zeng, K. "Marker-less registration based on template tracking for augmented reality". Multimedia Tools and Applications, 41(2):235-252, 2009.

[33]Lima, J. P., Teichrieb, V., Kelner, J., and Lindeman, R. W. Standalone edge-based markerless tracking of fully 3-dimensional objects for handheld augmented reality. In Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, VRST '09, pages 139-142, New York, NY, USA, 2009. ACM.

[34]Wendt, F. L., Bres, S., Tellez, B. and Laurini, R. Markerless outdoor localisation based on sift descriptors for mobile applications. In Proceedings of the 3rd international conference on Image and Signal Processing, ICISP '08, pages 439-446, Berlin, Heidelberg, 2008. Springer-Verlag.

[35]Schweighofer, G., Segvic, S. and Pinz, A. Online/realtime structure and motion for general camera models. In Proceedings of the 2008 IEEE Workshop on Applications of Computer Vision, pages 1-6, Washington, DC, USA, 2008. IEEE Computer Society.

[36]Comaniciu, D., Ramesh, V., and Meer, P. "Kernel-based object tracking". IEEE

Trans. Pattern Anal. Mach. Intell., 25:564-575, May 2003.

[37]Bradski, G. R., Santa Clara, and Intel Corporation. "Computer vision face tracking for use in a perceptual user interface". Interface, 2(2):1221, 1998.

[38]Welch, G. and Bishop, G. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[39]Beymer, D. and Konolige, K. Real-time tracking of multiple people using continuous detection. In Proc. ICCV Frame-rate Workshop, 1999.

[40]Davison, A. J., Civera, J. and Montiel, J. M. M. "Structure from Motion using the Extended Kalman Filter". Springer Tracts in Advanced Robotics. Springer, 2012.

[41]Kitagawa, G. "Non-gaussian state-space modeling of nonstationary time series". Journal of the American Statistical Association, 82(400):1032, 1987.

[42]Kailath, T. "The Divergence and Bhattacharyya Distance Measures in Signal Selection". IEEE Trans. On Comm. Technology 15(1), 52-60 (1967).

[43]Geem, Z. W. "Recent Advances In Harmony Search Algorithm", volume 270 of Studies in Computational Intelligence. Springer, 2010.

[44]Fourie, J., Mills, S. and Green, R. "Harmony filter: A robust visual tracking system using the improved harmony search algorithm". Image Vision Comput., 28:1702-1716, December 2010.

[45]Hager, G. D. and Belhumeur, P. N. "Efficient region tracking with parametric models of geometry and illumination". Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20(10):1025-1039, 1998.

[46]Tomasi, C. and Kanade, T. Detection and Tracking of Point Features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[47]Benhimane, S. and Malis, E. Real-time image-based tracking of planes using efficient second-order minimization. 2004 IEEERSJ International Conference on Intelligent Robots and Systems IROS IEEE, 1(x):943-948, 2004.

[48]Benhimane, S. and Malis, E. "Homography-based 2d visual tracking and servoing". Int. J. Rob. Res., 26:661-676, July 2007.

[49]Park, Y., Lepetit, V., and Woo, W. Esm-blur: Handling & rendering blur in 3d tracking and augmentation. In Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '09, pages 163-166, Washington, DC, USA, 2009. IEEE Computer Society.

[50]Sukumar, S. R., Page, D. L., Bozdogan, H., Koschan, A. F., Abidi, M. A. "MuFeSaC: Learning When to Use Which Feature Detector," Image Processing, 2007. ICIP 2007. IEEE International Conference on , vol.6, no., pp.VI-149-VI-152, Sept. 16 2007-Oct. 19 2007.

[51]Dame, A. and Marchand, E. Accurate real-time tracking using mutual information.

In ISMAR, pages 47-56, 2010.

[52]Schmid, C., Mohr, R., and Bauckhage, C. "Evaluation of Interest Point Detectors". Int. J. Comput. Vision 37, 2 (June 2000), 151-172.

[53]Harris, C., Stephens, M. A combined corner and edge detector. In: Proceedings of the Alvey Vision Conference. (1988) 147 - 151.

[54]Lindeberg, T. "Feature detection with automatic scale selection". IJCV 30(2) (1998) 79 - 116.

[55]Mikolajczyk, K. and Schmid, C. Indexing based on scale invariant interest points. In: ICCV. Volume 1. (2001) 525 - 531.

[56]Lowe, D. Object recognition from local scale-invariant features. In: ICCV. (1999).

[57]Kadir, T. and Brady, M. "Scale, saliency and image description". IJCV 45(2) (2001) 83 - 105.

[58]Jurie, F., Schmid, C. Scale-invariant shape features for recognition of object categories. In: CVPR. Volume II. (2004) 90 - 96.

[59]Mikolajczyk, K. and Schmid, C. "Scale and affine invariant interest point detectors". IJCV 60 (2004) 63 - 86.

[60]Mikolajczyk, K. and Schmid, C. A performance evaluation of local descriptors. PAMI 27 (2005) 1615-1630.

[61]Rosten, E. and Drummond, T. Machine learning for highspeed corner detection. In European Conference on Computer Vision, volume 1, 2006.

[62]Quinlan, J. R. "Induction of decision trees". Machine Learning 1 (1986) 81-106.

[63]Rublee, E., Rabaud, V., Konolige, K., Bradski, G. ORB: An efficient alternative to SIFT or SURF, Computer Vision (ICCV), 2011 IEEE International Conference on , vol., no., pp.2564-2571, 6-13 Nov. 2011.

[64]Calonder, M., Lepetit, V., Strecha, C., and Fua, P. Brief: Binary robust independent elementary features. In European Conference on Computer Vision, 2010.

[65]Rosin, P. L. "Measuring corner properties". Computer Vision and Image Understanding, 73(2):291 - 307, 1999.

[66]Bay, H., Tuytelaars, T. and Van Gool, L. Surf: Speeded up robust features. In European Conference on Computer Vision, May 2006.

[67]Lowe, D. "Distinctive image features from scale-invariant keypoints, cascade filtering approach". IJCV 60 (2004) 91 - 110.

[68]Lindeberg, T. "Scale-space theory: A basic tool for analysing structures at different scales", Journal of Applied Statistics, 21, 2 (1994), pp. 224-270.

[69]Brown, M., Lowe, D. G. Invariant features from interest point groups. In: 13th British Machine Vision Conference, Cardiff, British Machine Vision Assosciation (2002) 656-665.

[70]Schaffalitzky, F., Zisserman, A. Multi-view matching for unordered image sets, or How do I organise my holiday snaps? In: 7th Euproean Conference on Computer Vision, Springer (2002) 414-431.

[71]Rutkowski, W. S., Rosenfeld, A. A comparison of corner detection techniques for chain coded curves. Technical Report 623, Maryland University (1978).

[72]Langridge, D. J. "Curve encoding and detection of discontinuities". Computer Vision, Graphics and Image Processing 20 (1987) 58-71.

[73]Medioni, G., Yasumoto, Y. "Corner detection and curve representation using cubic b-splines". Computer Vision, Graphics and Image Processing 39 (1987) 279-290.

[74]Mokhtarian, F., Suomela, R. "Robust image corner detection through curvature scale space". IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 1376-1381.

[75]Shi, J., Tomasi, C. Good features to track. In: 9th IEEE Conference on Computer Vision and Pattern Recognition, Springer (1994).

[76]Ke, Y., Sukthankar, R. PCA-SIFT: A more distinctive representation for local image descriptors. In: CVPR (2). (2004) 506 - 513.

[77]Leite, P., Teixeira, J. M., Farias, T., Reis, B., Teichrieb, V., Kelner, J. "Nearest Neighbor Searches on the GPU". International Journal of Parallel Programming, v. 1, p. 1-18, 2011.

[78]Beis, J., and Lowe, D. G. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. Conference on Computer Vision and Pattern Recognition, Puerto Rico (1997), pp. 1000-1006.

[79]P. V. C. Hough, Method and Means for Recognising Complex Patterns, U.S. Patent No. 3069654, 1962.

[80]Loncomilla, P. and Del Solar, J. Improving SIFT-Based Object Recognition for Robot Applications, pages 1084-1092. Springer Berlin / Heidelberg, 2005.

[81]Lieberknecht, S., Benhimane, S., Meier, P. and Navab, N. A dataset and evaluation methodology for template-based tracking algorithms, 8th IEEE International Symposium on Mixed and Augmented Reality, 2009. ISMAR 2009, pp.145-151, 19-22 Oct. 2009.

[82]Richard Hartley and Andrew Zisserman. 2003. Multiple View Geometry in Computer Vision (2 ed.). Cambridge University Press, New York, NY, USA.

[83]Dame, A. A unified direct approach for visual servoing and visual tracking using mutual information. PhD thesis, Université de Rennes 1, Mention informatique, December 2010.

[84]Steger, C. "Occlusion, clutter, and illumination invariant object recognition." International Archives of Photogrammetry Remote Sensing and Spatial Information

Sciences 34.3/A (2002): 345-350.

[85]Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. "Gradient Response Maps for Real-Time Detection of Textureless Objects". IEEE Trans. Pattern Anal. Mach. Intell. 34, 5 (May 2012), 876-888.

[86]Hsiao, E. Occlusion reasoning for object detection under arbitrary viewpoint. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (CVPR '12). IEEE Computer Society, Washington, DC, USA, 3146-3153.

[87]Ojala, T., Pietikainen, M. and Harwood, D. "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on , vol.1, no., pp.582-585 vol.1, 9-13 Oct 1994.

[88]Di Stefano, L., Mattoccia, S., and Tombari, F. "Zncc-based template matching using bounded partial correlation". Pattern Recogn. Lett., 26:2129-2134, October 2005.

[89]Dowson, N. and Bowden, R. A unifying framework for mutual information methods for use in non-linear optimisation. In Proceedings of the 9th European conference on Computer Vision - Volume Part I, ECCV'06, pages 365-378, Berlin, Heidelberg, 2006. Springer-Verlag.

[90]Wang, Z., Bovik, A. C., Sheikh, H. R. and Simoncelli, E. P. "Image quality assessment: from error visibility to structural similarity," Image Processing, IEEE Transactions on , vol.13, no.4, pp.600-612, April 2004.

[91]Zitov, B. and Flusser, J. "Image registration methods: a survey". Image and Vision Computing, 21:977-1000, 2003.

[92]R. D. Luce and H. Raiffa, Games and Decisions: Introduction and Critical Survey. New York: Wiley, 1957.

[93]Walsh, W. E., Tesauro, G., Kephart, J. O. and Das, R. Utility functions in autonomic systems, Autonomic Computing, 2004. Proceedings. International Conference on , vol., no., pp. 70- 77, 17-18 May 2004.

[94]Gary Rost Bradski and Adrian Kaehler. 2008. Learning Opencv, 1st Edition (First ed.). O'Reilly Media, Inc.

[95]Bussab, W. O., & Morettin, P. A. (2010). *Estatística Básica* (6 ed.) Saraiva.

[96]Teixeira, João Marcelo Xavier Natário; Teichrieb, Veronica; Kelner, Judith. Finding an adequate escape pod to real time Augmented Reality applications. In: Brazilian Symposium on Computer Graphics and Image Processing, 2009, Rio de Janeiro. Anais do SIBGRAPI 2009, 2009.

[97]Reis, Bernardo; Oliveira Filho, P. S. B.; Vasconcelos, L.; Teixeira, João Marcelo

Xavier Natário; Teichrieb, Veronica; Kelner, Judith. MarkerMatch: an Embedded Augmented Reality case study. In: Symposium on Virtual and Augmented Reality, 2010, Natal. Proceedings of the SVR 2010, 2010.

[98]Vasconcelos, L.; Breyer, Felipe; Teixeira, João Marcelo Xavier Natário; Kelner, Judith; Teichrieb, Veronica. Desenvolvimento de uma Plataforma Móvel para Inspeção Termal de Equipamentos Baseada em Realidade Aumentada. In: ERGODESIGN USIHC, 2011, Manaus. Proceedings of the USIHC 2011, 2011.

[99]Teixeira, João Marcelo; Roberto, R. A.; Simoes, F. P. M.; Teichrieb, Veronica; Kelner, Judith. Reconstrução 3D usando luzes estruturadas. Tendências e Técnicas. In Symposium on Virtual Reality, 2010.

[100]Leão, C. W. M.; Reis, B.; Teixeira, João Marcelo; Kelner, Judith. Mivamain: A Platform for Assisting Maintenance with Augmented Reality 2011. In V Latin American Summer School on Robotics, Santiago, Chile (Poster).

[101]Emerenciano, T.; Tenorio, L.; Moura, Guilherme; Teixeira, João Marcelo; Almeida, Gabriel; KELNER, Judith. Um Protocolo para Auxílios Visuais em Aplicações de Realidade Aumentada Colaborativa. In: 14th Symposium on Virtual and Augmented Reality, 2012, Niterói. Proceedings of the 14th Symposium on Virtual and Augmented Reality, 2012.

[102]Reis, Bernardo; Teixeira, João Marcelo Xavier Natário; Albuquerque, E.; Teichrieb, Veronica; Kelner, Judith. Análise de técnicas de limiarização adaptativa para realidade aumentada embarcada. In: Workshop de Realidade Virtual e Aumentada, 2010, São Paulo. Anais do WRVA 2010, 2010.

[103]Santos, R.; Silva, Daliton; Teixeira, João Marcelo; Kelner, Judith. Rastreamento de Objetos Usando Template Matching e Variação de Iluminação. In: 14th Symposium on Virtual and Augmented Reality, 2012, Niterói. Proceedings of the 14th Symposium on Virtual and Augmented Reality, 2012.

[104]Teixeira, Joao Marcelo; Teichrieb, Veronica; Kelner, Judith. The influence of partial occlusion on object tracking 2011. In V Latin American Summer School on Robotics, Santiago, Chile (Poster).

[105]Leão, Crystian Wendel M.; Teixeira, João Marcelo Xavier Natário; Albuquerque, E.; Teichrieb, Veronica; Kelner, Judith. Melhorando o desempenho do rastreamento de pontos de interesse em imagens através do paralelismo em GPU. In: Workshop de Realidade Virtual e Aumentada, 2010, São Paulo. Anais do WRVA 2010, 2010.

[106]Farias, Thiago; Teixeira, João Marcelo; Leite, Pedro; Almeida, Gabriel; Almeida, Mozart Williams S.; Teichrieb, Veronica; Kelner, Judith. High Performance Computing: CUDA as a Supporting Technology for Next Generation Augmented Reality Applications. Revista de Informática Teórica e Aplicada, v. XVI, p. 63-88,

2009.

[107]Farias, Thiago; Teixeira, João Marcelo Xavier Natário; Almeida, Gabriel Fernandes de; Leite, Pedro; Teichrieb, Veronica; Kelner, Judith. A CUDA-enabled KLT Tracker for High Definition Images. In: Symposium on Virtual and Augmented Reality, 2009, Porto Alegre. Anais do SVR2009, 2009.

LIST OF ABBREVIATIONS

AR	Augmented Reality
AIC	Akaike information criterion
BRIEF	Binary robust independent elementary features
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CPU	Central processing unit
CV	Computer Vision
EKF	Extended Kalman filter
ESM	Efficient Second-order Minimization
FAST	Features from Accelerated Segment Test
GLOH	Gradient Location and Orientation Histogram
GPU	Graphics Processing Unit
GRVM	Virtual Reality and Multimedia Research Group
HSV	Hue, saturation, and value color model
ID3	Iterative Dichotomiser 3
KLT	Kanade–Lucas–Tomasi feature tracker
LBP	Local Binary Patterns
MI	Mutual Information
ORB	Oriented FAST and Rotated BRIEF
OTS	Object Tracking Switcher
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus
SIFT	Scale-Invariant Feature Transform
SSD	Sum of Squared Differences
SSIM	Structural Similarity index
SURF	Speeded Up Robust Features
SVD	Singular Value Decomposition
ZNCC	Zero Normalized Cross-Correlation