AALBORG UNIVERSITY
INSTITUTE OF ELECTRONIC SYSTEMS

Fredrik Bajersvej 7 ▪ DK-9220 Aalborg East Phone 96 35 80 80

TITLE:	Object Tracking With A Head Mounted Camera
THEME:	Image Analysis and Computer Graphics
PROJECT PERIOD:	8th Semester. From February 2nd 2001 to June 6th 2001
PROJECT GROUP:	822

PARTICIPANTS:
Jon Frydensbjerg
Tonny Gregersen
Lars Chr. Hausmann
Peter Korsgaard
Lars J. Kristensen

SUPERVISOR:
Claus B. Madsen

Abstract

This report deals with the development of a computer vision system for object tracking of everyday objects on a table using a head mounted camera. The problem at hand is derived from the EU funded project ARTHUR. A system must be developed which tracks objects with a frame rate of at least 10 frames per second. The system can in general be divided into 3 core parts; segmentation, stabilization, and tracking.
The stabilization, which estimates the orientation of the camera relative to the surroundings, utilizes window based correlation to estimate the disparity between images and a 2D affine method to convert the disparity to angles in world-space.
The segmentation of objects is solved by using a bipartite approach. Initially pixels belonging to the table are found using global thresholding in the hue channel of the HSV color model, and subsequently pixels belonging to objects are found using the convex hull.
The tracking part, is based on feature extraction of among others invariant moments and histograms. Mahalanobis distance is used for the feature matching, and feature prediction is based on simple linear extrapolation. Furthermore, a general window based object recognition method for handling occlusions is proposed which works on histograms in the HSV color space.
The system is designed using Matlab, and a prototype is implemented in C++ based on the Intel Corporation's Open Computer Vision Library (OpenCV). The prototype contains the segmentation and tracking component of the system. Occlusions are, however, not handled currently.
The test of the prototype showed that the system is capable of tracking separate objects and is robust with respect to scaling and rotations. Unique objects are tracked more robustly than objects with similar color. The demand regarding a frame rate of 10 frames per second has been tested and confirmed, as long as there are 6 or fewer objects in the camera's field of view.
It is concluded that the system with the chosen delimitations works as specified, but a number of future improvements is suggested. Specifically stabilization and the method for handling occlusions must be added to the current prototype in order to achieve better results.

PUBLICATIONS:	8
NUMBER OF PAGES:	150
FINISHED:	June 6th 2001

AALBORG UNIVERSITET
INSTITUT FOR ELEKTRONISKE SYSTEMER

Fredrik Bajersvej 7 ▪ DK-9220 Aalborg Øst Telefon 96 35 80 80

TITEL:	Object Tracking ved Anvendelse af Hovedmonteret Kamera
TEMA:	Billedanalyse og Computer Grafik
PROJEKTPERIODE:	8. semester. Fra d. 2. februar 2001 til d. 6. juni 2001
PROJEKTGRUPPE:	822

DELTAGERE:
Jon Frydensbjerg
Tonny Gregersen
Lars Chr. Hausmann
Peter Korsgaard
Lars J. Kristensen

VEJLEDER:
Claus B. Madsen

Synopsis

Denne rapport omhandler udviklingen af et computer vision system til object tracking af hverdagsobjekter på et bord ved anvendelse af et hovedmonteret kamera. Problemstillingen er afledt af EU-projektet ARTHUR. Der ønskes udviklet et system som kan tracke objekter med en opdateringsfrekvens på mindst 10 billeder per sekund.
Systemet kan overordnet opdeles i 3 kernedele; segmentering, stabilisering og tracking.
Stabiliseringen, der estimerer kameraets orientering i forhold til omverdenen, benytter vinduebaseret korrelation til at estimere billedforskydning og en 2D affin metode til omsætning af vinkler i verdensrummet.
Segmenteringen af objekter er løst ved anvendelse af en todelt tilgang. Først findes pixels, som tilhører bordet via *global thresholdning* i HSV farvemodellen's hue-kanal, og efterfølgende findes pixels, som tilhører objekterne på bordet via *convex hull*.
Tracking-delen baserer sig på feature extraction af bl.a. invariante momenter og histogrammer. Til feature matching bruges Mahalanobis afstand, og feature-prædiktion baserer sig på simpel, lineær ekstrapolation. Endvidere er en generel vinduesbaseret objektgenkendelsesmetode til håndtering af okklusioner foreslået, der anvender histogrammer i HSV-farverummet.
Systemet er designet vha. Matlab, og der er implementeret en prototype i C++, som benytter Intel Corporation's Open Computer Vision Library (OpenCV). Prototypen indeholder segmentering og tracking-delen af systemet. Dog håndteres okklusioner ikke i den nuværende C++ prototype.
Testen af prototypen viste, at systemet kan tracke separable objekter, og er robust overfor skalering og rotationer. Unikke objekter trackes dog mere robust end objekter med ens farve. Kravet om en opdateringsfrekvens på 10 billeder i sekundet var testet og er overholdt, så længe der er 6 eller færre objekter indenfor kameraets synsfelt.
Det kan konkluderes, at systemet overordnet virker som ønsket med de valgte begrænsninger, men flere fremtidige forbedringer er foreslået. Specielt skal stabilisering, og metoden til håndtering af okklusioner tilføjes til den nuværende prototype, for at bedre resultater kan opnås.

OPLAGSTAL:	8
SIDEANTAL:	150
AFSLUTTET:	6. juni 2001

Preface

This report is the result of a 8th semester Computer Vision and Graphics project, developed during the period from February 2nd to June 6th, 2001.

Figures, tables and equations used in the report are referred to as figure/table/equation X.Y, where X denotes the chapter number and Y represents the figure/table/equation number in the chapter.

References are in the form (⟨author⟩, ⟨year⟩), i.e. (Surname(s), 1999). All references are listed in the bibliography located last in this report.

Aalborg, June 6th, 2001.

Jon Frydensbjerg

Tonny Gregersen

Lars Chr. Hausmann

Peter Korsgaard

Lars Jochumsen Kristensen

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Recently there has been a considerable interest in combining live video from a camera with computer-synthesized graphical objects registered in some user's 3D environment. Possible applications of this visualization technique include guiding trainees through complex 3D manipulation and maintenance tasks, overlaying clinical 3D data during surgical planning, as well as developing 3D user interfaces. The resulting "augmented reality" (AR) systems allow "virtual" objects to be embedded into a user's environment as shown in figure 1.1. This raises issues that are unique to augmented reality (Kutulakos and Vallino, 1998, p. 1):

Establishing geometric relationships between physical and virtual objects: The virtual 3D locations of objects must be initialized in the user's environment before user interaction can take place.

Rendering virtual objects: Realistic augmentation of a 3D environment can only be achieved if objects are continuously rendered in a manner consistent with their assigned location in 3D space and the camera's viewpoint.

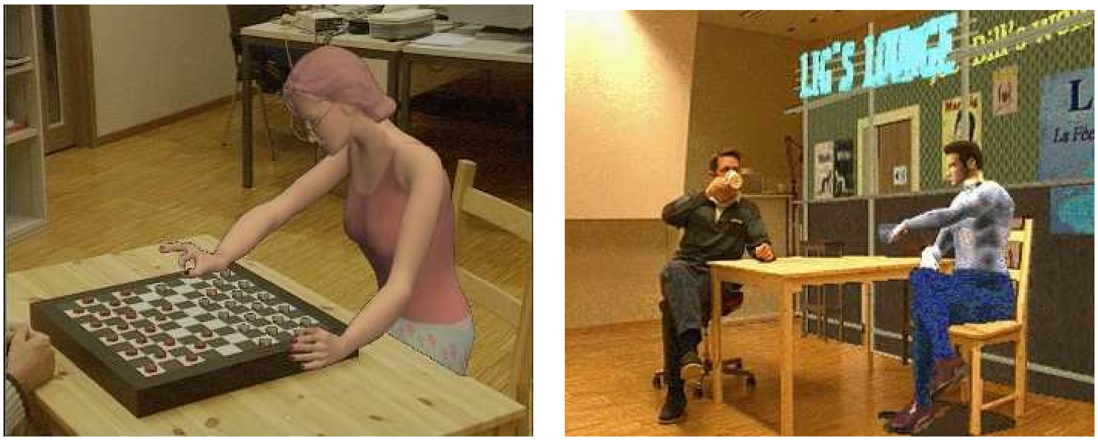


Figure 1.1: The above images represent two scenarios from (Torre et al., 2000). An autonomous virtual human is playing checkers against a real opponent in real-time, and a real and a virtual human are having tea together.

The subsequent sections describe the motivation for the project, the initial problem, and the report outline, respectively.

1.1 The ARTHUR Project

The ARTHUR project is made in collaboration between several European companies and universities with the purpose of developing an AR system which can be used for round table meetings. The following is primarily based on (IST, 2001).

The AR system will enable users to view virtual objects embedded in the real world through a pair of see-through glasses. This will be done by tracking a number of placeholder items using image analysis on pictures acquired by head-mounted cameras. The placeholder items used will be everyday items like cups, mobile phones, pens etc. The AR system track the placeholder items and draws a specific virtual object on top of each of them on each of the users see-through glasses. This is done in real-time and only if the objects are in-sight.

The equipment used will be a pair of see-through glasses, a laptop computer, and some sort of wireless network device for each user. Furthermore a number of everyday items will be used as placeholder objects. The augmented round table setting can be seen in figure 1.2.

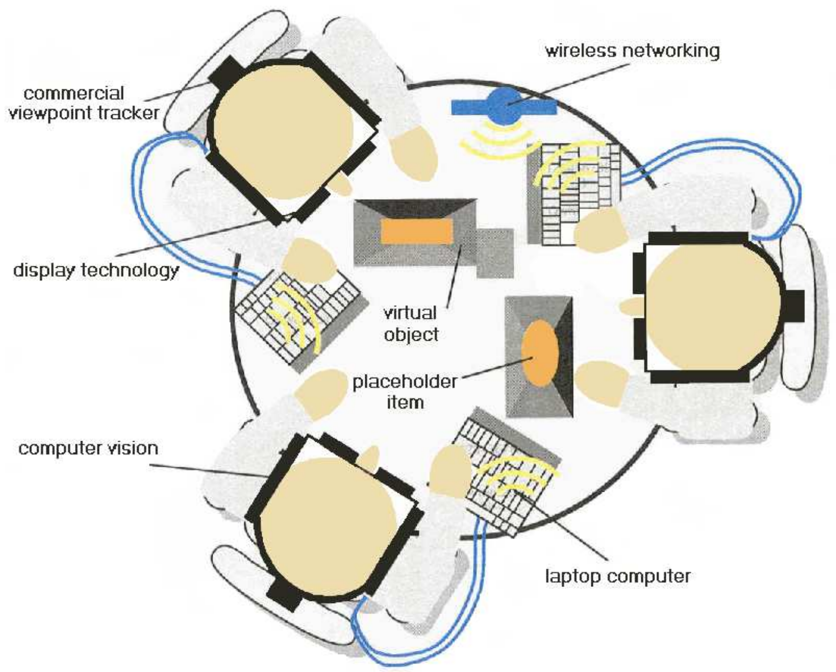


Figure 1.2: The round table setting in ARTHUR (from (IST, 2001, p. 27)).

The full title of the ARTHUR project is "Augmented Round Table for Architecture and Urban Planning", hence the acronym ARTHUR.

1.1.1 The ARTHUR Environment

The ARTHUR environment is to consist of four fundamental components which are described briefly in the following:

Augmented Reality Display Technology: A set of low weight, well-balanced, and wireless projection glasses is to be developed. These glasses will be used to enhance the user's natural environment, the real world, to a virtual world by combining the vision of the real world with artificial 3D objects.

Augmented Reality User Interface: A set of interface mechanisms will be developed in order to enable the user to interface with the AR system. The interface might use 3D menus, speech input, and gestures.

Computer Vision: The computer vision part is used to detect placeholder items using image analysis on the pictures acquired by the cameras. Image analysis will then be used to determine the virtual objects, their position and orientation. All this has to be done in real-time providing at least 10 frames per second (fps) (IST, 2001, p. 38).

Multi-User Augmented Reality System: The multi-user augmented reality system consists of a stereo visualisation system, an external device interface, an external application interface, and a network synchronisation module. The stereo visualisation system is to provide two different pictures of the virtual 3D objects, one for each LCD-screen in the see-through glasses, hereby making the virtual 3D objects appear at the appropriate distance. The external device interface will make ARTHUR able to communicate with external hardware like display and sensor technologies (e.g. a viewpoint tracker). The external application interface is to provide a general interface to the core of ARTHUR. This is done in order to provide an interface which makes it easy to use the core of ARTHUR within other applications. The network synchronisation module is made to enable real-time information sharing of data obtained by each of the participants equipment while keeping the virtual scene in a consistent state.

Finally the following application specific component is placed on top of the core of ARTHUR:

Architectural Design and Urban Planning Application: The Architectural Design and Urban Planning Application component contains the application specific parts of the program. Since the application area is architectural design and urban planning it will contain information which enables it to compute patterns of pedestrian movements, wind velocities, lighting analysis and so forth.

The five components are depicted in figure 1.3.

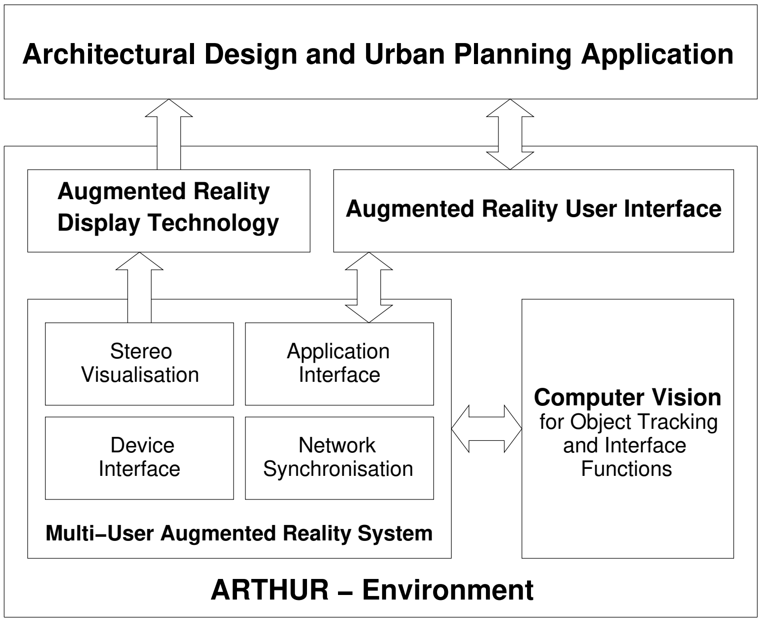


Figure 1.3: The architecture of ARTHUR from IST (2001, p. 22).

1.1.2 The Computer Vision Component

Aalborg University has been assigned the task of developing the computer vision component for ARTHUR. The component can be decomposed into three real-time (i.e. fast in terms of computation time) processing stages:

1. **Object tracking**. Placeholder items must be tracked between subsequent images by use of digital images acquired from a camera.
2. **Pose estimation** (also known as registration or exterior calibration). Determine the 3D position and orientation of tracked objects using computer vision analysis techniques.
3. **3D visualization**. Virtual objects are rendered by utilizing position/orientation data obtained from pose estimation of the physical objects.

This report describes a possible design of the object tracking stage as a standalone system (i.e., the initial stage). The pose estimation, 3D visualization, as well as interfaces between successive stages, are not part of this project.

1.1.3 Tracking

Before the problem of object tracking is further analyzed, it is defined:

- **Object tracking** is the process of consistently labeling the item(s) of interest between consecutive images.

Tracking is normally categorized as having either *active* or *passive* sensors (Birk, Moeslund, and Sørensen, 1996, p. 3). Active tracking in a computer vision application is when the tracking system is able to change sensor (i.e., camera) positions, e.g., track an object that moves away from the field of view using the analyzed movements of the objects to control the sensors. Passive tracking is conversely when the sensors are completely static. This report describes a passive tracking system, according to the definition above, which is unable to change sensor or to control the sensor position of the sensor, although the user may independently change sensor orientation.

1.2 Initial Problem

Based on section 1.1 the initial problem for this project can thus be formulated as:

> How can a robust, real-time system be constructed, which by use of a head-mounted camera, tracks placeholder items on a table?

It is important to note that this project is more concerned with the study and evaluation of methods used to construct such a tracking system than with necessarily specifying and constructing a working solution to the ARTHUR problem. This is due to the high complexity implied by requiring the use of a single head-mounted camera and arbitrary items as placeholder items, and the lack of known, robust solutions to tracking objects with these requirements.

1.3 Outline

The rest of this report is structured as shown in figur 1.4.

Preanalysis: Addresses relevant techniques and technologies related to the application domain and potential problems with these.

System specification: The system to be developed is specified based upon the preanalysis and external demands.

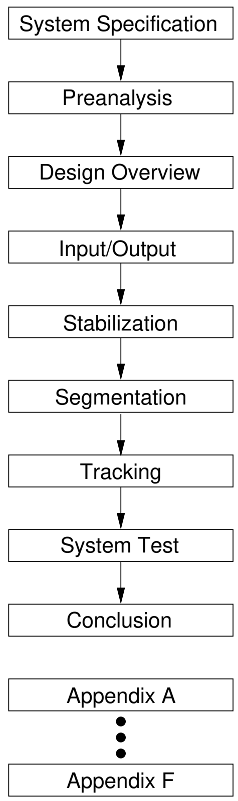


Figure 1.4: Report structure.

Design overview: Outlines the general design of the system consisting of the four main components.

Input/Output: Description of the component which concerns input and output of the system.

Stabilization: Description of the component which concerns camera stabilization.

Segmentation: Description of the component which concerns the segmentation made in the system.

Tracking: Description of the component which concern the tracking part of the system.

System test: Describes the test of the entire system and its results.

Conclusion: The conclusion and discussion of the results and the project.

Appendix A to F: Contains in depth information on subjects dealt with in the report.

CHAPTER 2

Preanalysis

This chapter deals with various problems and issues imposed by the problem and application domain. These considerations have been divided into six more or less related sections. The first two sections deals with general considerations regarding the environment and user interaction with a tracking system such as ARTHUR. Hereafter, specific delimitations and preliminary demands to this project are listed. The last three sections deals with the description of a mathematical model of the environment, problems with real cameras (opposed to the pinhole model), and issues with a head-mounted, movable camera, respectively.

To summarize; in this chapter the following will be discussed:

- The environment, in which the system is to be used, is described and analyzed in order to identify potential problems.
- A user/system interaction analysis is performed to understand how users can interact with the system.
- Preliminary demands and delimitations are presented.
- A simplified computer graphics based mathematical model defining the relationship between the camera, world and objects is presented.
- The video camera is analyzed to identify problems with lenses, grayscale vs. color, and shutter speed.
- The relative movement of the head-mounted camera and placeholder objects are analyzed in order to identify and describe motion related issues.

This chapter constitutes the basis for the system specification in chapter 3.

2.1 Environment

The environment in which the system operates is important for the robustness of the system and the rate of images to be processed. In a controlled environment, the algorithm used for locating and matching the placeholder items might be simplified. An increase in noise sources results in an increase of complexity in the system.

The environment consists of a number of different parts. These parts are listed below:

- Users

- Table
- Placeholder items
- Light

Each of these parts might cause problems; they are described in the following subsections.

2.1.1 Users

The users of the system have a shared table on which the placeholder items are placed. The users are seated around the table with different angles toward the table and the placeholder items. Figure 2.1 illustrates an example of the environment without the placeholder items.

Every user of the system has a head mounted camera which points in the direction of the user's point of view. The field of view of the camera is then captured as a sequence of images which are used for tracking the placeholder items.

As the system might service multiple users, it has to deal with multiple image sequences captured from different cameras. Using multiple cameras requires an initialization of the coordinate system in order to match the images retrieved from the different cameras.

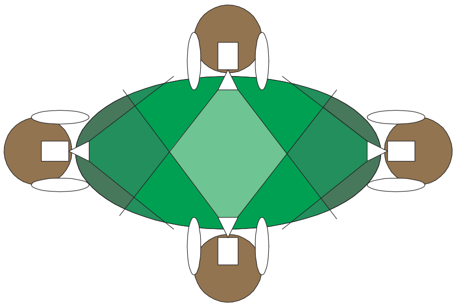


Figure 2.1: A user environment consisting of 4 users seated around a table. The different levels of green illustrates the number of cameras viewing the different areas of the table.

2.1.2 Table

The table is the surface on which the placeholder items are placed. The edges of the table mark the boundaries of where to track the placeholder items. All placeholder items placed outside the table are of no interest to the system.

As the table defines the area in which the system has to identify placeholder items and that there is the possibility of multiple users, it is important to match the coordinates of the table to the systems world coordinate system.

This issues a problem, since different tables might have different shapes, sizes, surfaces, and colors.

The initialization of the system with respect to the identification of the table might take different approaches depending on the level of automation. Suggestions for the initialization of the system with respect to the table are listed below sorted by their level of automation:

1. The system identifies the table without the help of the user. It makes use of the assumption, that the user mainly focus on the table and that human skin tissue (a user points at the table) is registered on the table. Figure 2.2 (a) illustrates an image presented to the system. As the figure shows the hand of the user is within the boundaries of the table (points at the table).

2. The system makes use of the information given from all users connected to the system. The texture and color of the surface, which is in the field of view of the different users represents the table. Figure 2.1 illustrated how a part of the table is in the field of view of all the users (the area with the brightest level of green).
3. The user interacts with the system in order help the system determine the table. The user of the system places a, to the system known, ring on the table. This ring might be physical or just visualized for the user. The system locates the area inside the ring and registers the texture and color. The area inside the ring provides a representation of the table. Figure 2.2 (b) shows an image which illustrates a table with the controlled environment in form of a virtual ring.
4. The system is told the boundaries of the table. This is done by placing known symbols on each of the corners and along the sides of the table. Figure 2.2 (c) gives an example of an image with marked boundaries.

(a) (b) (c)

Figure 2.2: Figure (a) illustrates how human tissue is used to indicate the location of the table. Figure (b) illustrates how a ring generates a controlled environment for detecting the color and texture of the table. Figure (c) illustrates the boundaries of the table marked with known templates.

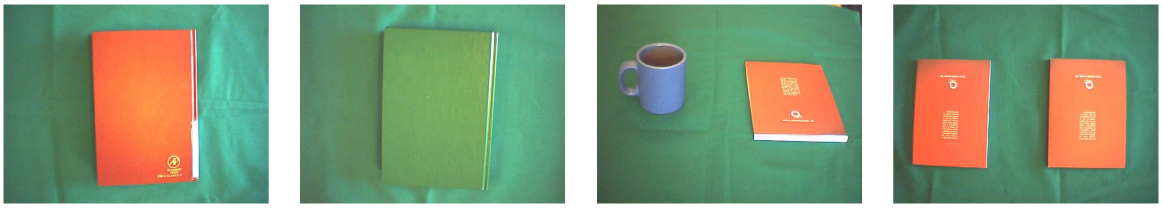
The system is initialized with regard to the table when the table texture and color is known.

2.1.3 Placeholder Items

The placeholder items used with the system are crucial to the robustness of the system. In order to make the segmentation of the placeholder item as simple and fast as possible, it is important to have some issues in mind when selecting the placeholder items. Some of these issues are as follows:

- The placeholder items must have an apperance which differs significantly from the table. This is important in order to detect a distinct edge between the table and the placeholder items. Figure 2.3 (a) and (b) shows an example on the issues concerning the difference in apperance between the placeholder items and the table.
- The shape of the placeholder items is also important. The more unique the shape of the placeholder items are, the easier it is to separate the placeholder items from each other. Figure 2.3 (c) and (d) shows an example of issues concerning the shape of the placeholder items.
- The color of the placeholder items can be used for tracking, since the sequence of images is captured in color. Using placeholder items with different colors make the placeholder items more unique. Figure 2.3 (c) and (d) give an example on issues concerning the color of the placeholder items.

The choice of the placeholder items leads to considerations regarding shape, color, and contrast to the table. In order to decrease the complexity of the segmentation and tracking, these three



(a) (b) (c) (d)

Figure 2.3: Figure (a) illustrates a placeholder item with large contrast to the table. Figure (b) illustrates a placeholder item with small contrast to the table. Figure (c) illustrates two placeholder items with unique shape and color. Figure (d) illustrates two items with similar shape and color.

issues has to be considered. The optimal placeholder item has a large contrast to the table, a unique shape, and a unique color compared to the other placeholder items.

2.1.4 Light

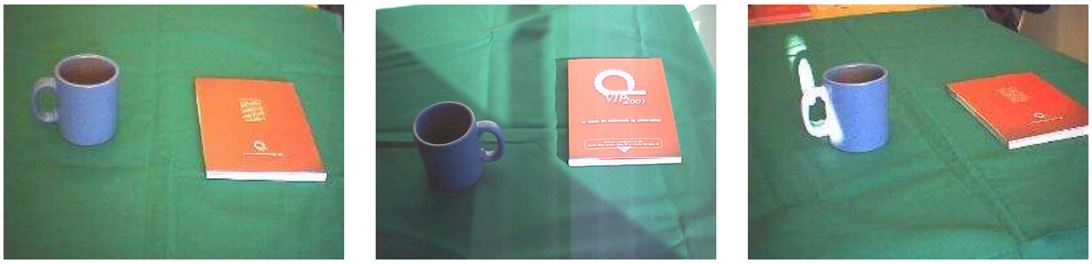
The light on the table and the placeholder item comes from different light sources. As the system might be used under different circumstances; the system has to be robust regarding a combination of different light sources. Different light sources and problems generated from these are listed below.

- Light bulbs as light sources gives some considerations due to the way bulbs produces light. This item is based on (Jähne, Haussecker, and Geissler, 1999, chapter 3). Light from light bulbs consist of a narrow spectrum of light waves compared to light from the sun which include all light waves within infrared, visual, and ultraviolet light.

  The temperature of the light produced by the light bulb has a direct influence on the spectrum of light waves generated by the bulb.

  This type of light combined with the functionality of the camera during capture, might separate areas of the image into the three color channels (RGB) differently with different types of light bulbs.
- The light contribution from the sun or light bulbs might cause shadows and different light intensity on the surface of the placeholder items. The low angle and the intensity of the sun compared to the other light sources increases the complexity of the problem of separating the placeholder items from the table, and other placeholder item. Because of the low angle of the sun compared to the light coming from the ceiling; the sun shining on a placeholder item casts long shadows on the table and perhaps on other placeholder items. This is illustrated in figure 2.4 (c).
- The light contribution might change in intensity if a cloud covers the sun. This changes the main light source from being the sun to being the lamps in the room and back. This is illustrated in figure 2.4 (b) using figure 2.4 (a) as reference.
- The light contribution changes locally if a reflection is sweeping through the room. This resolves in a high intensity area that changes relatively fast. This is illustrates in figure 2.4 (c).
- The light intensity varies from one side of the image to the other if shadows is cast on the table. This changes the apparency of the placeholder item whenever the light intensity is altered one way or the other; making it more difficult to match the placeholder item with the images. This is illustrated in figure 2.4 (c) using figure 2.4 (a) as reference.

The light sources influences the complexity and robustness of the tracking and segmentation algorithms. In order to decrease the complexity, and to increase the robustness of the algorithms;



(a) (b) (c)

Figure 2.4: Figure (a) illustrates images captured with the one of the cameras using controlled light. This is a reference image to figure (b) and (c). Figure (b) illustrates the impact of sunlights ability to cast shadows on the table and the other placeholder items. Figure (c) illustrates the ability of a reflection to alter the intensity in a local area of the image.

the light settings applied to the table and the placeholder items has to be uniform. This is possible in a controlled environment.

2.2 Interaction

The user of the system needs to interact with the system in order to get a common understanding of the environment. As the interaction module is only a part of the complete system; the user interface is more like a test environment interface, than an ordinary user interface.

The input to this module is an image sequence, which is captured from a head mounted camera. The sequence depicts the placeholder items on the table. The user controls the camera and thereby the field of view. This way of interaction gives the user an easy and natural way of controlling the system input image sequence.

There are three different areas of interaction between the user and the system. They are listed below:

- Initialization of the system.
- Presentation of the placeholder items to the system.
- Representation of the tracked placeholder items to the user.

It is important to initialize the system in order to get a useable output as described in section 2.1. The initialization may take different approaches depending on the level of user interaction. Several approaches are presented in subsection 2.2.1 for making an uniform system under different environments.

Another aspect concerning the user interaction with the system is marking the placeholder item for tracking. There are several approaches to this as described in subsection 2.2.2.

The result is the tracked placeholder items. When a new placeholder item is introduced to the system, it is labeled, so that it can be matched in later appearances.

2.2.1 Initialization of the System

The system must be initialized in order for all the users of the complete system to have an uniform understanding of the "world". In the following three approaches to initialization the system are listed:

- The user initializes the system by sweeping around the edges of the table. The system measures the number of pixel along each edge and translates it to a coordinate system. The light condition is registered during the sweep as well.

Pros: It is easy for the user to take the system into service. The level of user interactions is at a minimum as well as a controlled environment is generated during initialization.

Cons: The translation between the image sequence and the coordinate system is complex. As the complete system allows multiple users it is difficult to generate a shared understanding of the coordinates. As it is based on the location of the users around the table and the sweeps point of origin might be different from user to user.

- The user creates a controlled environment for the initialization of the system. This is done by setting up some rules of initialization. The sweep starts from the same point of origin, this applies to all the users of the complete system. The system combines the sweeps from all the users in order to generate the coordinate system. The light condition is registered as well.

  Pros: It is easy for the user to take the system into service because there is only a small number of rules to obey.

  Cons: The translation is still complex, but the increased amount of information makes the translation between the image sequence and the coordinate system more robust.

- A priori knowledge of the table is entered in the system from the users. This information might be the size and shape of the table. To support these information unique marks (templates) are placed along the edges of the table and a sweep around the edges of the table is performed. The light condition is registered during the sweep as well.

  Pros: Supplying the system with a priori knowledge decreases the complexity of the algorithm used for calculating the translation. The robustness of the algorithm increases with the amount of a priori knowledge given to the system.

  Cons: The users interaction with the system during the initialization increases along with the amount of information given to the system.

2.2.2 Presenting Placeholder Items to the System

This section deals with how it is possible to present new placeholder items to the system. The user might take more or less part in selecting the placeholder items, which the system is supposed to track. A number of approaches is described below:

- Every new placeholder item recognized is labeled for later tracking.

  Pros: This approaches have little to none interaction with the user.

  Cons: There are no restriction on the placeholder items labeled. This will resolve in a large amount of placeholder items that needs to be tracked, which increases the complexity of the system.

- Each time a new placeholder item is identified by the system, the user is asked if the placeholder item should be labeled.

  Pros: Not all of the found placeholder items is labeled and tracked. Only placeholder items approved by the users are tracked.

  Cons: It resolves in a large amount of user interaction due to the number of placeholder items found in the field of view. This is based on the fact that noise is present i.e., in form of artifacts either on the table, or placeholder items are found outside the table. Every time a not-initialized placeholder item enters the field of view it resolves in an interaction with the user. This is due to the fact that not-initialized placeholder items is not labeled and might be recognized again in later appearances.

- The user introduces the placeholder items for the system before using them. The introduction takes form in a controlled environment, such as the placeholder item in question is placed on the table inside a marked area.

  Pros: The user is in total control of which placeholder items the system is tracking. The interaction with the user is down to a minimum during the use of the system; only in the initialization phase is there interaction with the user when introducing placeholder items. Due to the limited amount of placeholder items to be tracked, the complexity is decreased.

  Cons: The user has to create the controlled environment in order to mark the placeholder item in question. It is more difficult to add a new placeholder item to the list placeholder items while the system is running.

2.2.3 Representation of the Tracked Placeholder Items to the User

Placeholder item representation deals with how a tracked placeholder item is presented to the user of the system. This test environment may use a graphical or textual output. Various techniques for representating approved placeholder items can be used, but this has not top priority due to the configuration as a test environment.

2.3 System Overview

This project is used as a prestudy of which techniques may be efficient to use in the Computer Vision component of ARTHUR. It is reasonable to alter the environment in this prestudy, hereby simplifying it without effecting the number of methods which can be used for the Computer Vision component.

In the simplified environment a single user is sitting at a table which is used as the workspace (see figure 2.5). On the table a number of rigid everyday items like cups, pens, and cellular phones are placed. These items are used as the placeholder items which are to be recognized and tracked using image analysis. The pictures are taken by a single head-mounted camera. The user may move and rotate the placeholder items around within the workspace. Both the camera and placeholder items thus have 6 degrees of freedom (DOF).

It is the task of the Computer Vision component to recognize the placeholder items and track their movement.

Preliminary Demands

At this point the demands to the system are a combination of demands made to the ARTHUR project and demands from the project proposal (Madsen, 2001):

- Use of a single head-mounted camera.
- Recognition of everyday items.
- Performance of at least 10 fps.
- Everyday items may be moved and rotated on a planar surface (the tabletop).

These demands are the fundamental demands. Additional demands/restrictions are added in the system specification (chapter 3) in order to make the system perform the task described above.

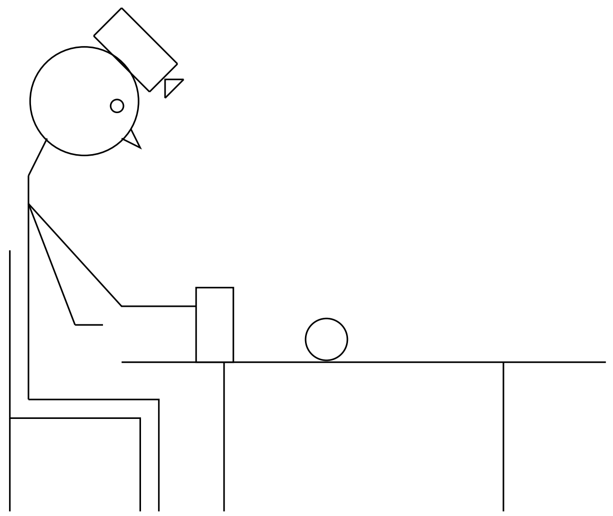


Figure 2.5: The simplified round table setting.

2.4 Mathematical Camera Model

The system makes use of a camera to depict the world as presented to the user of the system. In order to use the information presented to the user in form of a sequence of images, it is important to known the transformation between the world in which the placeholder items is placed and their representation in the image plane of the camera. This section elaborates on this topic in a basic mathematical way. This section is based on (Sonka, 2000).

2.4.1 Camera Model

The camera is placed in a 3D environment. As camera is only records a 2D environment a transformation from 3D to 2D is performed.

Figure 2.6 illustrates the 3D environment in which the camera is placed.

The model described in this section is a pinhole camera model which performs a perspective projection. This is illustrated by the scene point X (a point on the placeholder item) which is projected through the focal point C onto the image plane π as the point U.

The following topics are involved in the camera model illustated in figure 2.6.

Focal point is the point (optical center) where all the rays (light beams) are projected through.

Focal length is the distance from the image plane π to the focal point. Focal length is measured in pixels (f).

3D coordinate system describes a point in three dimensions. The model illustrated in figure 2.6 consist of five different coordinate systems. They are as follows:

The world Euclidean coordinate system (subscript $_w$) has origin at the point O_w. This coordinate system is used in the transformation between different coordinate systems, e.g. placeholder item and camera coordinate system. The degrees of freedom (DOFs) equals 6, rotation and translation along the X_w-, Y_w-, and Z_w-axis.

The camera Euclidean coordinate system (subscribe $_c$) has origin at the point O_c, the focal point. The coordinate axis Z_c is aligned with the optical axis and points away from the image plane π. There is a unique relation between the world and the camera coordinate systems, this relation is an Euclidean transformation consisting of a translation t and a rotation R. DOF equals 6, rotation and translation around the

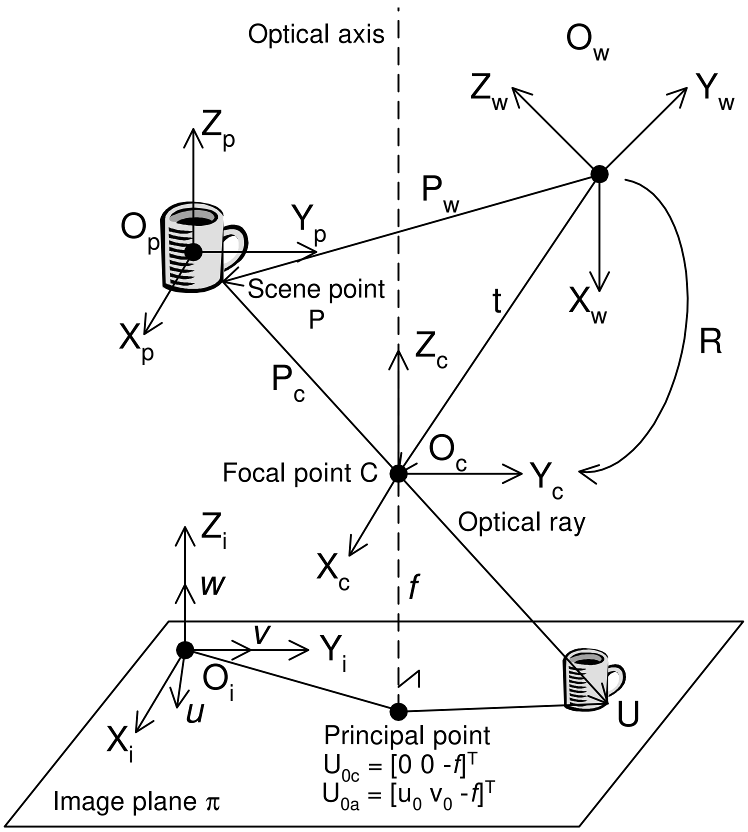


Figure 2.6: The geometry of a linear perspective camera.

X_c-, Y_c-, and Z_c-axis.

The image Euclidean coordinate system (subscript $_i$) has axes aligned with the camera coordinate system, with X_i, Y_i lying in the image plane π. The degree of freedom equals 0 as it is fixed to the camera Euclidean coordinate system.

The image affine coordinate system (subscript $_a$) is a linear coordinate transformation that includes the elementary transformations translation, rotation, scaling, stretching, and shearing. The origin O_i coincide with the origin of the image Euclidean coordinate system. The axis w, v are aligned with the axis Z_i, X_i, but the axis u may have a different orientation to the axis Y_i. DOF equals 0, as it is fixed to the camera Euclidean coordinate system.

The placeholder item Euclidean coordinate system (subscript $_p$) describes the orientation and position of the placeholder item within the world Euclidean coordinate system. DOF equals 6, rotation and translation around the axis X_p-, Y_p-, and Z_p-axis.

Transformation between coordinate systems

The transformation from the 3D location in the world Euclidean coordinate system to the 2D representation in the image affine coordinate system is made using the following three steps:

1. Perspective transform $\mathbf{T}_{persp}$
2. Translation $\mathbf{T}_{trans}$
3. Rotation $\mathbf{T}_{rot}$

The transformation is calculated on each point in the world Euclidean coordinate system using homogeneous coordinates. Homogeneous coordinates includes an extra dimension, e.g.

$$P_h' = [X\ Y\ Z\ 1]$$

The transformation matrix is a 4×4 matrix that transforms a homogeneous point in the world Euclidean coordinate system to same homogeneous point in the camera Euclidean coordinate system. The transformation matrix is a combination of three listed steps:

$$\mathbf{T} = \mathbf{T}_{persp} \cdot \mathbf{T}_{trans} \cdot \mathbf{T}_{rot}$$

Using a pinhole model it is possible to project the point in the camera Euclidean coordinate system onto the image plane π, as homogeneous coordinates in the image Euclidean coordinate system, or the image affine coordinate system.

Now that the mathematical considerations have been explained in some detail some considerations regarding the camera will be described.

2.5 Camera

The following considerations about the camera has been made:

- Deviations from the camera model
- Color versus black and white
- Progressive scan versus interlaced
- Camera settings

2.5.1 Deviations from Camera Model

The differences between the camera model described in section 2.4 and a real camera is outlined in the following.

A camera is schematically build as shown in figure 2.7. Basically it consists of one or several lenses which focuses the light on to the image plane. On the image plane a light detector is placed. The light detector is normally a *Charge Coupled Device* (CCD) chip which contains a grid of light sensitive points. An image is recorded by measuring the amount of light hitting each point in the grid during a short time (one divided by the shutter speed).

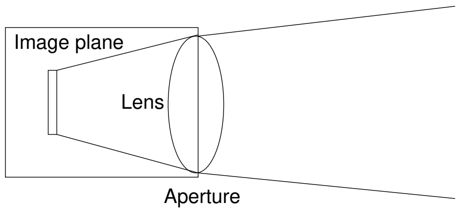


Figure 2.7: Camera schematics showing the aperture, lens and image plane.

In the pinhole camera model the camera has an infinitesimal opening (the so called *aperture*), but a real camera uses an opening of a certain size to allow enough light to enter the camera (Jain, Kasturi, and Schunck, 1995, p. 249). A camera uses a lens to gather even more light, but lenses have several drawbacks:

- Limited depth of field
- Distortions caused by imperfections in the lens system

The bigger the aperture size is the more light hits the image plane, but unfortunately also the depth of field is reduced. This can be seen from the lens equation:

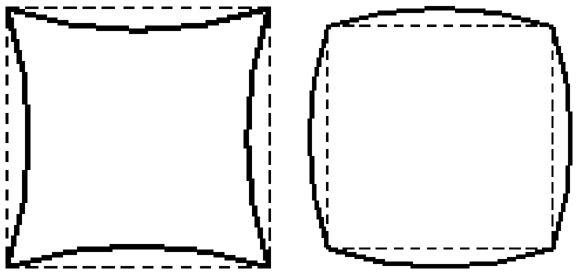
$$\frac{1}{z\prime} - \frac{1}{z} = \frac{1}{f} \tag{2.1}$$

where z is the distance to the object in the scene, $z\prime$ is the optimal distance from the center of the lens to the image plane, and f is the focal length of the lens.

When the distance (z) to the object goes towards infinity $z\prime$ goes towards f, and at smaller distances $z\prime$ becomes smaller than f.

If the object moves away from the distance z it will go out of focus and become blurred. Based on the the spatial resolution of the image plan, the diameter of the aperture, the focal length, and a maximum amount of blurring allowed the depth of field can be calculated.

In addition to problems with focus. Imperfections in camera lenses also causes distortion of the recorded image. Often lenses have a level of radial distortion caused by varying magnification levels in different radius of the center. This either leads to what is called the pillow or the barrel effect as seen in figure 2.8.



(a) (b)

Figure 2.8: Figure (a) and (b) illustrates the two kinds of radial distortion, the pillow effect and the barrel effect, respectively.

2.5.2 Color versus Black and White

Cameras used for computer vision normally record in either black and white (grayscale) or in color producing a level or red, green, and blue for each pixel. Often color information is a powerfull descriptor in object identification and segmentation (Gonzalez and Woods, 1993, p. 221). Figure 2.9 shows a situation where color information is needed to differentiate between two objects.

(a) (b)

Figure 2.9: Figure (a) illustrates how color can be used to differentiate objects which would be very hard to differentiate using a black & white camera as illustrated in figure (b).

2.5.3 Progressive Scan versus Interlaced

A video camera can either record in progressive scan mode or in interlaced mode. In interlaced mode each frame is divided into two fields, one containing the even lines and one containing the odd lines. Instead of recording the entire frame at once the even lines are recorded first and half a frame period later the odd lines are recorded. If the frame contains moving objects this offset in time between even and odd lines introduces artifacts in the picture as shown in figure 2.10.

Figure 2.10: Artifacts caused by interlacing. Notice the jagged lines.

2.5.4 Camera Settings

To ensure the best image quality the camera should allow the following settings to be set manually:

- White balance
- Gain
- Shutter speed

Most cameras contains a circuit called an automatic white balance, which automatically adjusts the image to the current color temperature of the surroundings. These circuits does not adapt instantaneously and therefore a large change in color can happen in situations with varying color temperature. An example of such a situation is a cloud temporarily occluding the light from the sun. In computer vision systems it is often more interesting wether objects keep the same color through the session instead of having the right shade. If the white balance can be set at a fixed value a lower change in shade between successive frames is experienced during a session with varying color temperature.

Furthermore most cameras has an automatic gain circuit (AGC) which adjusts the amplification level of the signal from the CCD to give a better performance in situations with very little light. Like the white balance circuit this system does not adapth instantly, and a minimum of change in color is experienced if the AGC is turned off.

Finally it is often possible to set the integration time, the so-called shutter speed. The higher the integration time and hence the lower shutter speed, the more light sensitive is the camera, because more light hits the CCD for each frame. Unfortunately if anything is moving in the field of view of the camera an effect called *motion blur* can be experienced. Figure 2.11 shows an example of this effect. As the level of motion blur increases when the shutter speed is lowered a compromise between the two must be found, which ensures both that the camera will be able to operate under the lighting conditions given and that motion blur will be small enough not to cause problems for the used computer vision algorithms.

Furthermore the camera should preferably have settings for frame rate, compression level (if used) and resolution.

Figure 2.11: Artifacts caused by motion blur. Notice that the steady background isn't blurred.

In the next section the various types of movement which the system will be able to handle will be analyzed.

2.6 Movement Estimation

Several types of movement occur in a system with a moving camera and moving objects:

- The placeholder items moving within the field of interest (the table).
- The pitch of the camera.
- The yaw of the camera.
- The instability of the camera due to the camera being headmounted.

The analysis of the listed autonomous motions is grouped into three categories; moving the placeholder item, moving the camera, and instability of the camera.

2.6.1 Moving the Placeholder Items

There are two kinds of placeholder items. The ones approved by the users, and other detected items which are of no interest. The placeholder items approved by the user are labeled for tracking. It is the movement of these placeholder items that is in focus. The movement or displacement of these placeholder items is done by the user of the system. It is assumed that the displacement of the placeholder item is done manually by hand. The velocity of this displacement is the same as for the human hand.

The Configuration of the Experiment to Estimate the Movement of the Placeholder Items

The estimation of maximum velocity is based on experiments done by the members of the group. Figure 2.12 illustrates the environment of the experiment:

Estimation of maximum velocity is based on a statistical evaluation of 30 observations. The observations are based on the following sequence.

1. Move the placeholder item to the point P1
2. Move the placeholder item in a curve to the point P2
3. Move the placeholder item in a curve to the point P1

Estimation of Maximum Velocity

The three enumerated steps listed above is executed 15 times, which equals 30 sweeps from point P1 to point P2 across the table. The experiment is executed six times by the five group members.

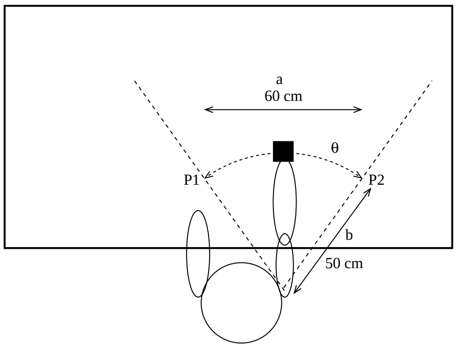


Figure 2.12: The environment of the experiment conducted to measure the maximum velocity of the placeholder item sweeping across the table.

The result of the experiment is an average of 24 seconds for 30 sweeps, with a standard deviation of 1,8 second for a sequence of 30 sweep across the table with the placeholder item.

Equation 2.2 calculate the angle of the sweep.

$$\angle\theta = sin^{-1}(\frac{\frac{a}{2}}{b}) = \arcsin(\frac{30}{50}) = 36,87° \tag{2.2}$$

The velocity of the moving placeholder item v_{plac}, measured in degrees per seconds, is calculated in equation 2.3.

$$v_{plac} = \frac{2 \cdot \angle\theta}{sweep\ time} = \frac{73,74}{\frac{24}{30}} = 92°/s \tag{2.3}$$

The average velocity v_{plac} is 92 $°/s$. Considering variability of the measured times: v_{plac} is in the interval of 86 - 100 $°/s$.

Using the hand for moving the placeholder item causes occlusions. This means that during the movement of a placeholder item it is partly covered by the moving hand. This increases the complexity of matching and tracking the placeholder item, which already experiences an increased complexity due to the displacement of the placeholder item.

2.6.2 Moving the Camera

The field of interest is normally equal to the surface of the table. As the size and shape of the table variate, it is not given that the field of view of the camera covers the entire table. This causes movement, either horizontal (yaw) or vertical (pitch), of the head mounted camera. The velocity of this movement is crucial to the functionality of the algorithms used for tracking the placeholder items. If the maximum velocity of these movements is determined, it is possible to narrow down the algorithms in question.

The Configuration of the Experiment to Estimate the Head Movement

Estimation of maximum velocity is based on experiments done by the members of the group. Figure 2.13 illustrates the environment of the experiment.

Estimation of maximum velocity is based on a statistical evaluation of 30 observations. The observations are based on the following sequence.

1. Face the upper left corner of the table and make the light mounted on the head hit the corner.
2. Turn the head and the light toward the upper right corner.
3. Stop turning the head when the light hits the upper right corner.

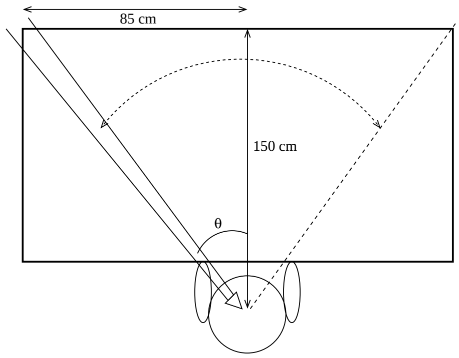


Figure 2.13: The environment of the experiment conducted to measure the maximum velocity of the moving head.

4. Turn the head and the light back toward the upper left corner.
5. Stop turning the head when the light hits the upper left corner.

The equipment and settings used for this experiment is listed below:

The light used in this experiment is coming from a light source which produces a narrow light beam. The light source is a head mounted flashlight, at the same location as the head mounted camera.

The table used in the experiment is of the dimensions 170x120 cm.

Estimation of Maximum Velocity

The five enumerated steps listed above is executed 15 times, which equals 30 sweeps along the edge of the table. The experiment is executed six times by the five group members. The result of the experiment is an average of 14 seconds for the 30 seconds, and a standard deviation of 1 second for a sequence of sweeps of the light beam from the upper left corner to the upper right corner.

The equation 2.4 calculate the angle of the sweep.

$$\angle\theta = \arctan(\frac{a}{b}) = \arctan(\frac{85}{150}) = 29,53^\circ \tag{2.4}$$

The velocity of the turning head v_{yaw} and v_{pitch}, measured in degrees per seconds, is calculated in equation 2.5.

$$v_{yaw} = v_{pitch} = \frac{2 \cdot \angle\theta}{sweep\ time} = \frac{59,06}{\frac{14}{30}} = 127^\circ/s \tag{2.5}$$

The average velocity v_{yaw} and v_{pitch} is 127 $^\circ/s$. Considering variability of the measured times: v_{yaw} and v_{pitch} is in the interval of 118 - 136 $^\circ/s$.

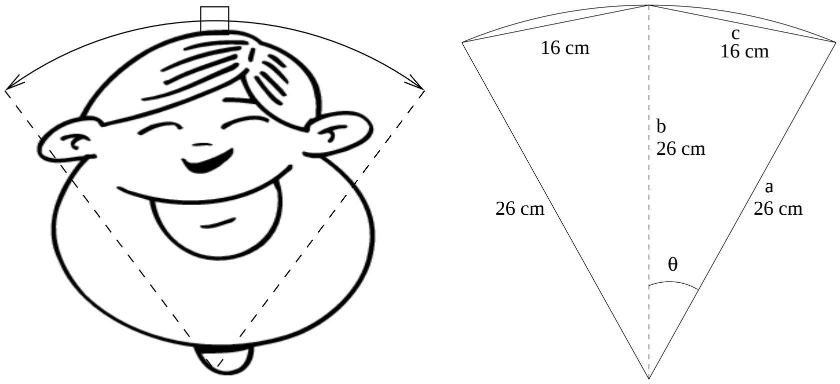
2.6.3 Instability of the Camera

The use of a headmounted camera cause unwanted movements in form of small yaws and pitches. These movements are in large scale dealt with in section 2.6.2. The last of the three degrees of freedom of the head is the rotation around the optical axis (moving the ear toward the shoulder and back).

The velocity of this rotation is estimated using an experiment conducted by the members of the group.

The configuration of the experiment to estimate the head rotation

Figure 2.14 illustrates the environment of the experiment.



(a) (b)

Figure 2.14: Figure (a) illustrates the environment of the experiment conducted to measure the maximum velocity of the rotating head. Figure (b) illustrates the measures of the experiment.

Estimation of maximum velocity is based on a statistical evaluation of 30 observations. The observations are based on the following sequence.

1. Rotate the head to the left (moving the left ear toward the left shoulder).
2. Stop rotating when experiencing natural resistance.
3. Rotate the head to the right (moving the right ear toward the right shoulder).
4. Stop rotating when experiencing natural resistance.
5. Rotate the head to the left (moving the left ear toward the left shoulder).
6. Stop rotating when experiencing natural resistance.

Estimation of maximum velocity

The six enumerated steps listed above is executed 15 times, which equals 30 rotations from shoulder to shoulder. The experiment is executed six times by the five group members. The result of the experiment is an average of 25 seconds for the 30 rotations, and a standard deviation of 1.4 seconds for the sequence of 30 rotations from the left shoulder to the right shoulder.

The equation 2.6 calculates the angle of the rotation.

$$\angle\theta = \arccos(\frac{a^2 + b^2 - c^2}{2 \cdot a \cdot b}) = \arccos(\frac{26^2 + 26^2 - 16^2}{2 \cdot 26 \cdot 26}) = 35,84^\circ \tag{2.6}$$

The velocity of the rotation of head v_{rot}, measured in degrees per seconds, is calculated in equation 2.7.

$$v_{rot} = \frac{2 \cdot \angle\theta}{sweep\ time} = \frac{71.68}{\frac{25}{30}} = 86^\circ/s \tag{2.7}$$

The average velocity v_{rot} is 86 $^\circ/s$. Considering variability of the measured times: v_{rot} is in the interval of 81 - 91 $^\circ/s$.

The results found in this section is summarized in table 2.1

Variable	Minimum [$°/s$]	Maximum [$°/s$]	Average[$°/s$]
v_{plac}	86	100	92
v_{yaw}	118	136	127
v_{pitch}	118	136	127
v_{rot}	81	91	86

Table 2.1: Results of movements experiments

2.7 Summary

The main purpose of this section is to summarize the main aspects of this chapter. Especially those which are relevant to the system specification, which is described in the next chapter.

The main focus in the system specification is build around demands to the system. In this chapter several demands has been mentioned, and the most important are as follows:

- Run in an uncontrolled environment.
- The placeholder objects should be rigid everyday objects (see section 2.1.3).
- The system should run in real time ie. with 10 frames per second (see section 2.3).

In the next chapter a system specification will be given.

CHAPTER 3

System Specification

In this chapter the problem, requirements, preconditions, and delimitation for the project are specified. First the specific problem for the project is specified in detail, primarily based on the preliminary demands and the analysis in chapter 2. Next requirements regarding the accuracy of the developed system are specified, followed by preconditions related to the hardware and equipment used for this project. Finally a delimitation is made based on the time and complexity of the task.

3.1 Specification

The system to be developed should be able to recognize, track and label everyday placeholder objects on a table (the virtual workspace) between subsequent frames using digital image analysis. The tracking must be performed using images acquiring from a single head-mounted camera.

A database containing representations of the placeholder items is necessary for the tracking process. The database has dynamic properties since changes in the real world (e.g. from moving around an item) must also be reflected in the database. It is important that the tracking system is able to recognize objects although they have left the field of view and returned.

Both the camera and the objects may move and rotate independently, and occlusions must be handled in order to obtain acceptable tracking results when objects are overlapping in the image plane. Additionally, the system should allow new objects to be added to the workspace at runtime without requiring complete reinitialization.

The primary objective of the system is to ensure that object pixels are consistently assigned the correct label. If for example a 2D workspace consisted of a circle, a square, and a triangle, these objects should at any time be assigned the labels 1, 2, and 3, respectively. Failure to achieve this, results in obvious augmentation problems if the system is extended with pose estimation (by use of the segmented pixel sets) and visualization capabilities.

The specific problem is depicted in figure 3.1.

As the problem described above is complex, it has proven neccesary to define certain constraints. A delimitation for this project is made in section 3.4.

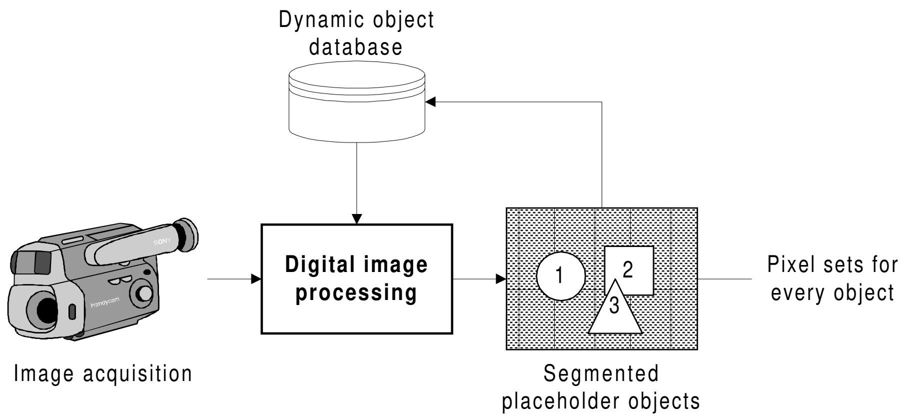


Figure 3.1: The specific problem of the project.

3.2 Requirements

Augmented reality systems are expected to run in real-time so that a user will be able to move about freely within the scene and see a properly rendered augmented image synchronized to his movements. This places two performance criterias to the object tracking system:

- Minimum update rate for tracking objects.
- Accuracy of the object registration/recognition.

Visually the real-time constraint is based on the user viewing augmented images without any visible "jumps" in the overlayed graphics. A rule of thumb is that the graphic system must be able to render the virtual scene at least 10 times per second in order to avoid "jumps". The ARTHUR project follows the rule of thumb as the specific requirement in the ARTHUR specification is 10 frames per second(IST, 2001, p. 38). However, a frame rate between 20 and 25 fps is desirable in order to minimize aliasing and thus enhance the visual experience, but 10 fps is set as the requirement for the minimum update rate.

Failures in ensuring reliable and accurate object registration have two possible causes:

Noise: Any noise/errors in the object tracking impacts the pose estimation stage of the AR system, and eventually the registration of the virtual image with the image of the real scene. The human visual system is very sensitive to visual errors which in this case would be the perception that the virtual object is not stationary in the real scene. Misregistrations of even a pixel can be detected under the right conditions (Vallino, 2000).

Time delays: As mentioned above, a minimum cycle time of 0.1 seconds is needed for acceptable real-time performance. If there are delays in tracking objects between subsequent frames then the augmented objects will tend to lag behind motions in the real scene. The system design should minimize the delays to keep overall delay within the requirements for real-time performance.

No quantitative requirement is defined as the desired accuracy of the object registration, as it has been decided to primarily focus on obtaining the 10 fps update rate (and nothing is specified with regard to accuracy in IST (2001)).

3.3 Hardware/Software Preconditions

The system will be implemented on a PC system with the following specifications:

CPU: 1 GHz Intel P3

OS: Linux 2.2.16

RAM: 256 MB

Frame grabber: ARVOO Picasso PCI-2SQ

Camera: ELMO QP49, lens: QT254 with focal length 4 mm.

It has been decided to use Intel's Open Vision Library (OVL) (Intel Corporation, 2001). OVL is a C/C++ API mainly aimed at real-time computer vision, running on top of an image processing library that provides the low-level functions. The primary advantage of OVL is that it contains highly optimized assembler and SIMD (single instruction multiple data) routines, which makes it well suited for a computationally demanding project.

The used camera is mounted on a pair of sunglasses by use of duct tape. The camera is a ELMO QP49H using a QT254 lens, which has a focal length of 4 mm (ELMO - Electricity Light Machine Organization, 2001), equal to a field of view of 60∘. Furthermore, it has a 1/4 inch IT-CCD. The camera is connected to a ELMO CC491 Control Unit, which is capable of adjusting white balance, shutter speed etc. The output from the control unit is a PAL TV-signal.

The frame grabber is a ARVOO Picasso PCI-2SQ, which is able to grab in PAL in up to 768x576 (EMJ Embedded Systems, 2001). The framegrabber is able to deliver pictures in RGB and YUV.

A set of test sequences have been recorded using a Sony DCR-VX2000 Digital Video Camcorder (SONY electronics, 2001). The pictures from the camera is saved tiff-format with a resolution of 640x480. The different test sequences contains images of a table with a number of everyday items in which either the camera has been rotated or objects have been moved in different ways. The used everyday items are a black and blue cellular phone, a white ball pen, a blue mug, a brown pencil case, a black plastic case, and a color calibration card.

3.4 Delimitation

Due to limited time it has been decided to delimit the project by prioritizing the design criteria specificied in section 3.2:

1. The update rate of the system.
2. Accuracy of the algorithms.

This tradeoff between speed and accuracy in computer systems is a common problem. It should be noted that by primarily focusing on the "real-time update rate" requirement might reduce the number of applicable techniques for image analysis, because they are too computationally slow using available technology.

Moreover the following preliminary contraints has been decided, which relates to the camera, environment, and placeholder objects, respectively. These constraints are primarily derived from the preanalysis in chapter 2, and have been decided based on a desire to make the system less error prone.

Camera Constraints

- A high shutter speed is used to diminish the effects of motion blurring.
- The white balance is calibrated before any image acquisition is performed.
- No auto gain is applied thereby eliminating a hardware source of error in the acquired images.

Environment Contraints

- The user must not move position relative to the table, but may rotate the head and/or lean either forward, backward, left or right.
- A green cloth is used on the table to provide an uniform, matte surface that is easily distinguishable and merely a diffuse reflector.
- Solely indoor illuminants are applied for lighting in order to avoid image intensity fluctuations.

Placeholder Item Contraints

- The used objects are colored with high contrast relative to the green cloth.
- Only rigid objects are used, as recognizing arbitrarily shaped (non-rigid) objects poses serious difficulties.
- New placeholder items are explicitly presented to the system, hereby removing the difficulty of automatically identifying new objects.
- The maximum number of objects in the field of view simultaneously has been set to 10 to avoid flooding the workspace.
- The required minimum number of pixels an object subtends in the image plane must be found. Very small objects would be virtually impossible to recognize robustly, so some tolerance value is desirable. The specific value must be experimentally found during the design, as it depends on many factors (e.g. the spatial resolution used for the image analysis).

The Specific Problem

In summary, and by omitting some insignificant delimitations and constraints, the specific problem addressed in this project can be stated as:

> How can a robust system be constructed, which by the use of a single head-mounted camera and solely indoor lighting, tracks rigid placeholder items on a table with a uniformly colored cloth, while obtaining update rates of at least 10 frames/second?

This concludes the system specification. The following chapters describes the design of the system.

CHAPTER 4

Design Overview

4.1 System Description

The system to be developed is a computer vision based object tracking system. Computer vision is the extraction of information from an image that usually is a two-dimensional projection of a real three-dimensional scene. Figure 4.1 illustrates the tasks of tracking an object in a scene and returning a representation of the object for further 3D manipulation.

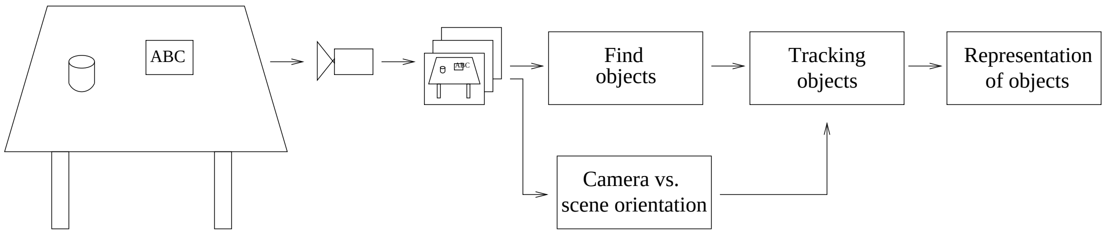


Figure 4.1: The tasks of object tracking

The classical steps in making a computer vision system are described in the following:

Image Acquisition Image acquisition is the first task in the process and is concerned with acquiring a digital image of the problem domain (the scene). This requires an imaging sensor (e.g. a video camera) and the ability to digitize the signal produced by the sensor. The sensor could produce different forms of images, e.g. monochrome or color images. The rate of the acquisition of images (frame rate) changes depending on the application.

Preprocessing After accuring the image comes the preprocessing, which prepares the images for the later tasks of the system. One way of preparing an images is by changing the color map, and down-scaling for real-time opportunities.

Segmentation After preprocessing, interesting regions of the images has to be found. This process is called segmentation. The regions of the segmented image represent objects that might be tracked.

Recognition and Interpretation The recognition and interpretation process match the regions with the initialized objects. Based upon the match it is possible to track the objects over time (a sequence of images).

Tracking As the regions in the image has become objects to the system the actual tracking is performed. This task is called tracking.

Stabilization In order to make the tracking more robust, and establish the relationship between the scene and the image, an extra task is introduced. This task is called stabilization. The purpose of this task is to supply the system with information of the orientation of the camera, as it is head-mounted and have the ability to rotate in all three directions. Based upon the orientation of the camera it is then possible to distinct the motion of the camera from the motion of the tracked objects.

Considering the previous described steps in making a computer vision system it is possible to decompose the system into the following four major tasks:

- Input/Output
- Segmentation
- Tracking
- Stabilization

These tasks and there relationship are illustrated in figure 4.2.

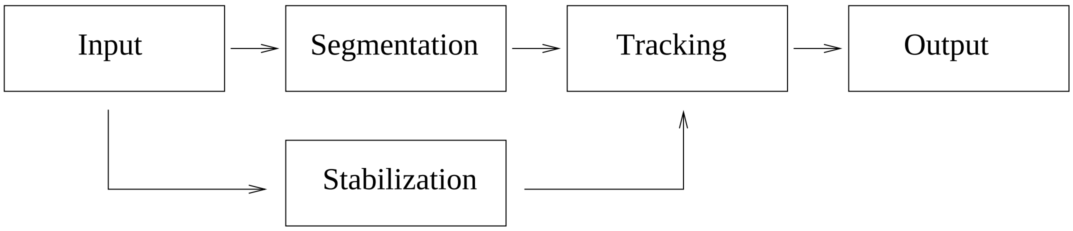


Figure 4.2: The block diagram of the system

The blocks depicted in figure 4.2 are briefly introduced in the following:

Input handles the acquisition of the image sequence from the single head-mounted camera. The preprocessing of the images is considered an input task. A detailed description of the input task is located in chapter 5

Stabilization handles the change in orientation of the camera. To do this, it uses the a priori knowledge of the scene composition, e.g. the table is static in the scene. The scene composition and the path of some non moving points in a sequence of images makes it possible to calculate the change in orientation of the camera, which are rotations around the axis X, Y, and Z. A detailed description of the stabilization task is located in chapter 6.

Segmentation handles the segmentation of the regions in the image, which are on the table. A detailed description of the segmentation task is located in chapter 7.

Tracking handles the tracking of the selected objects. Several features are measured for each object, and classification methods are used to compare objects from the database with the segmented objects. A detailed description of the tracking task is located in chapter 8.

Output handles the presentation of the tracked objects, and the orientation of the camera. A detailed description of the output task is located in chapter 5

This project only handles the two-dimensional projection of a real scene, which results in a representation relative to the origin of the camera image plane.

CHAPTER 5

Input/Output

This chapter describes the input and output components of the system. The location of the input and output components in the overall system can be seen in figure 5.1.

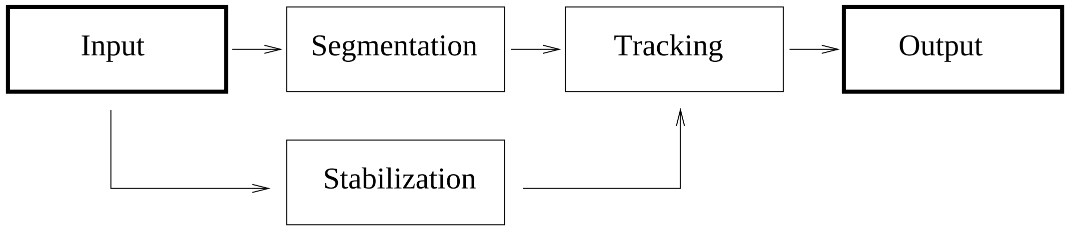


Figure 5.1: Location of the input and output components.

5.1 Purpose

This section will explain the purpose of the input and the output components.

5.1.1 Input Component

The purpose of the input component is twofold (see section 4.1). Firstly it has to acquire the images that are to be processed later on in the system. Secondly the input component has to handle the preprocessing part of the system. The preprocessing is used to increase the chances of success for the remaining parts of the system. Preprocessing normally deals with noise reduction, contrast enhancement and so forth.

The input component should be able to deliver images which are taken both from a framegrabber and from images stored on a storage medium in a specified format. The input component should also function as a preprocessor in that it formats the image handed to the rest of the system according to a choosen color model.

5.1.2 Output Component

The purpose of the output component is to display the processed image to the user on the screen. Furthermore the output component should provide functionality to communicate with the user of the system.

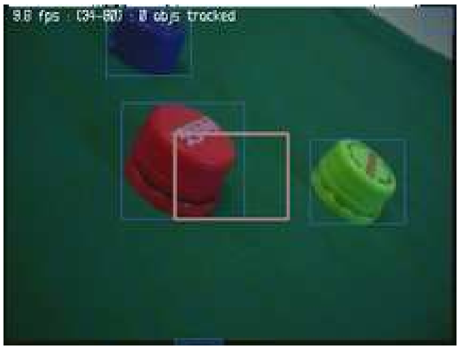


Figure 5.2: A possible output to the user from the system.

The image displayed to the user could look something like the one shown in figure 5.2.

5.2 I/O Specification

In this section the input/output specifications for the input and output component will be specified.

5.2.1 Input Component

Input: Signal from a camera or tiff images from a storage medium
Output: Processed image.

5.2.2 Output Component

Input: Processed image.
Output: Image shown on screen.

5.3 Domain

5.3.1 Input Component

The purpose of the input component is as stated above to provide the system with images to extract information from. There is generally two different scenarios:

- Online
- Offline

Online Mode

When the system is functioning in online mode it will be taking input directly from a camera which sends the signal through a framegrabber. Since the system should be able to run in real time the system has to provide images from the camera and framegrabber with a framerate of at least 10 frames pr. second (see section 3.2).

To get image from the framegrabber Video4Linux (see (Wolfgang Scherr, 2001)) is being used.

Offline Mode

When the system is functioning in offline mode it will be reading Tiff images from a storage medium. The main reason for the offline mode is to be able to test the system with controlled input.

To read the images from the storage medium libTiff (see (Frank Warmerdam and Mike Welles, 2001)) is being used.

5.3.2 Output Component

The output component will provide the user of the system with a visual image of the data that the system has processed and calculated.

It has been decided that the following user interaction is to be supported:

Initialize the table. In order to initialize the table. The user presses the `enter` key, when there is a clear view of the table in the middle box as shown in figure 5.2.

Begin to track object. After the table has been initialized the user can initialize an object by centering it in the middle box and pressing the `space bar`.

When the system is tracking objects it should clearly mark to the user which objects it is currently tracking. This could be done by putting a colored box (for instances as shown in figure 5.2) around the object when it is in the camera's field of view.

In order to show the processed image to the user and to handle interaction with the user libSDL (see (Sam Lantinga, 2001)) is being used.

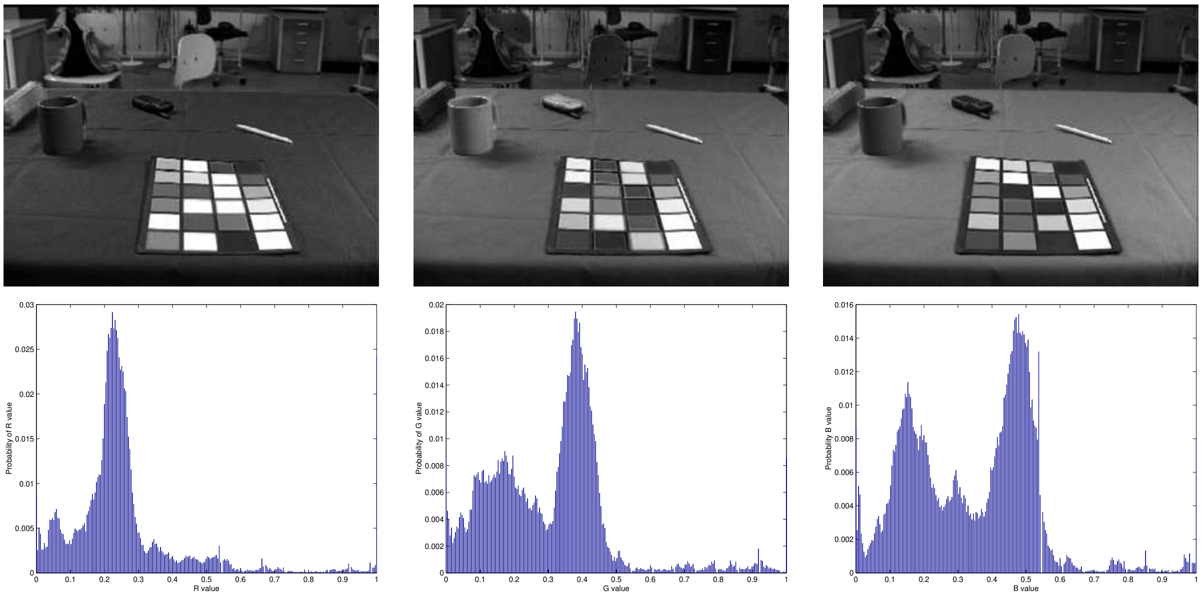
5.4 Overview

5.4.1 Input Component

As it was mentioned in section 5.1.1 one of the purpose of the input component was doing the preprocessing. One of the step normally associated with preprocessing is to choose the color model.

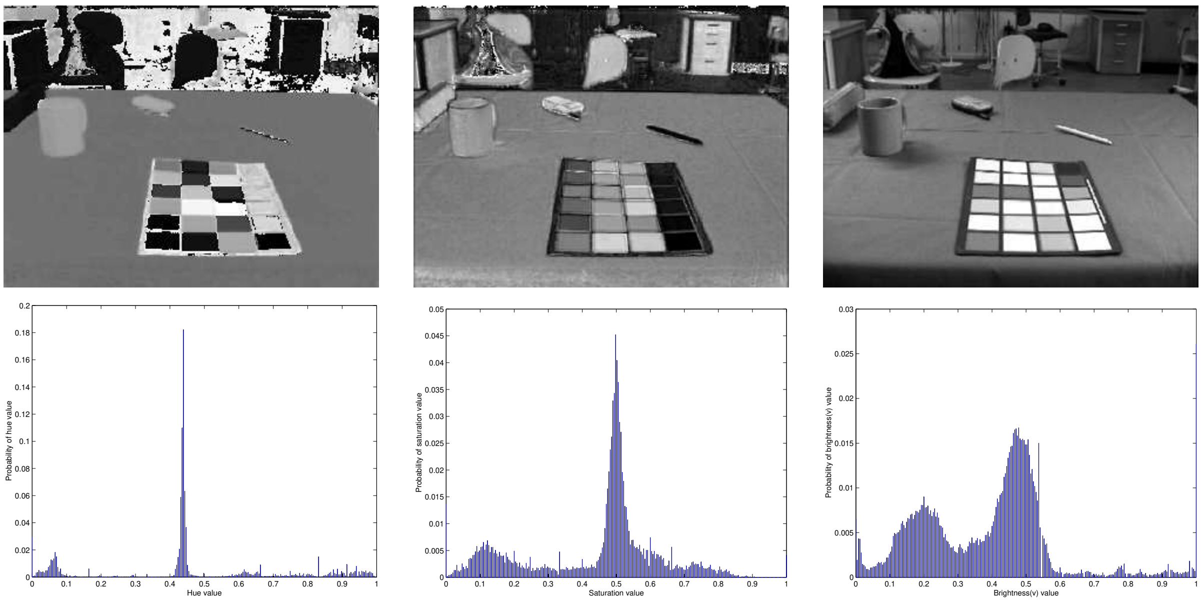
Figure 5.3: Image showing table with objects

In order to determine which of the three color models described in appendix B that is suitable to use, RGB, HSV, and rgb has been examined on images that are common to the system. The image that is being looked at can be seen in figure 5.3. To examine the color spaces each of the channels in the color spaces and the channels histogram is shown in figure 5.4, 5.5, and 5.6.



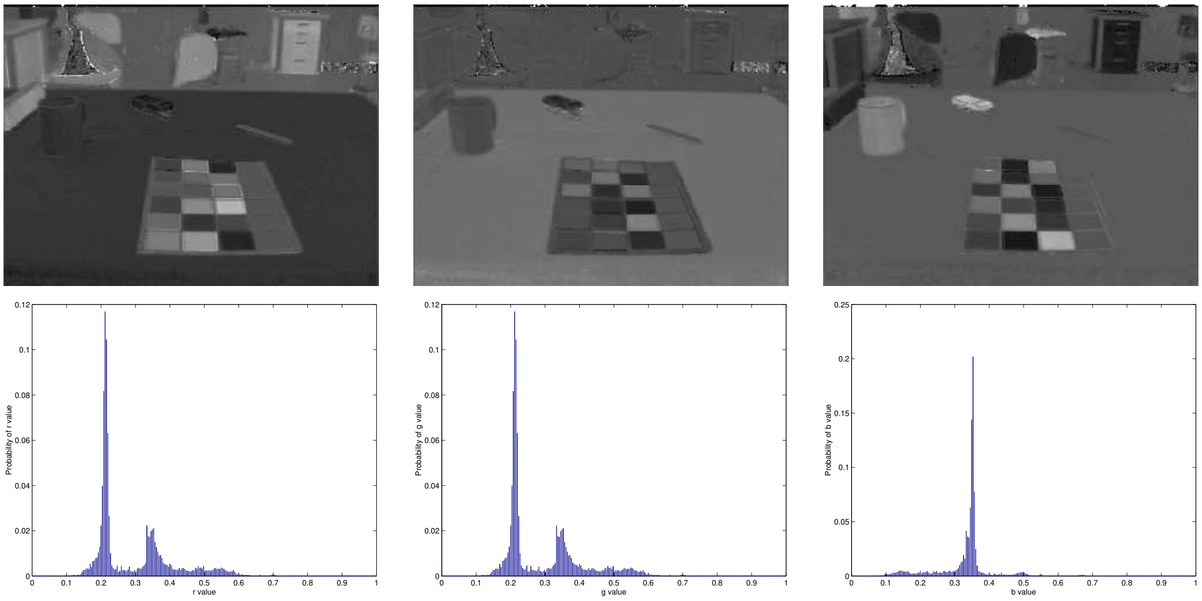
R channel G channel B channel

Figure 5.4: RGB channels of the image in figure 5.3



H channel S channel V channel

Figure 5.5: HSV channels of the image in figure 5.3



r channel g channel b channel

Figure 5.6: rgb channels of the image in figure 5.3

As it can be seen from the three color models as shown in figure 5.4, 5.5, and 5.6 and the histograms it would seem that the color model where the most distinct peak is the HSV color model's H channel. The peak caused by the green cloth on the table effectively marking the table. The distinct peak is important since it will make it easier to detect the table later on in the system. It can also be seen that another possible choice is the rgb color space since there is also a very distinct peak in the b channel. However when comparing the H channel with the b channel images it is easier to determine what is the table, objects, and person in the H channel than it is in b channel. This indicates that it will be easier to segment the table out when using the H channel.

CHAPTER 6

Stabilization

This chapter describes the stabilization component of the system. The location of the component in the overall system can be seen in figure 6.1.

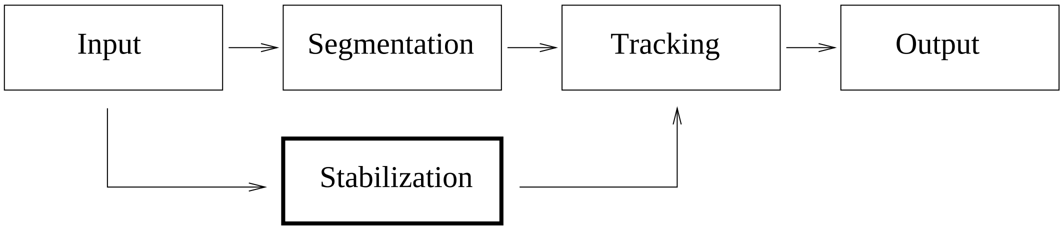


Figure 6.1: Location of the stabilization component.

6.1 Purpose

The purpose of the stabilization component is to determine the orientation of the camera in world-space using motion cues located in the image stream. The stabilization process is performed to ease the process of object recognition and object tracking by removing the effect caused by the motion of the camera from the image sequence, the so-called *ego-motion*.

6.1.1 Definitions

In order to describe the camera position and direction in world-space, a world coordinate system has to be defined.

As described in section 3.4 the user is not allowed to move around, only move his upper body and rotate his head around the three degrees of freedom. Because of this the primary ego-motion of the camera is rotation, and a sensible solution for a world coordinate system would be a spherical coordinate system with origo at the center of rotation of the camera. This is not the focal point of the camera as the camera is mounted on a set of glasses worn by the user as shown in figure 6.2. In a spherical coordinate system every point is uniquely determined by a radius, a rotation around the X axis called *pitch*, and a rotation around the Y axis called *yaw*. Using a spherical coordinate system is effectively equal to ignoring the translational part of the camera ego-motion.

The image plane in the world coordinate system can be seen in figure 6.3. The distance from

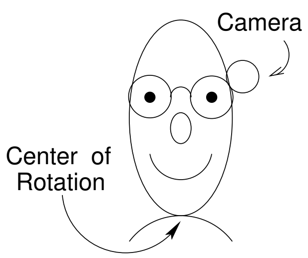


Figure 6.2: Because the camera is head-mounted, the center of rotation is located at the neck of the user, not at the focal point of the camera.

origo to the center of the image plane is the focal length f, its center is determined by pitch and yaw, and *roll* determines the rotation of the image plane around the focal axis.

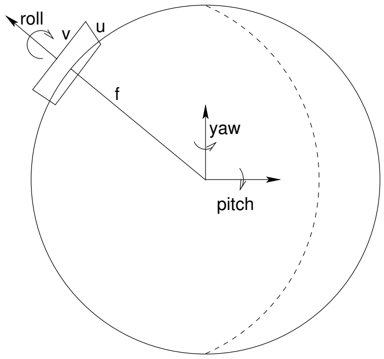


Figure 6.3: The spherical world coordinate system and the image plane.

Instead of defining an absolute orientation, the very first image is defined as having zero pitch, yaw, and roll. The orientation of all other images are relative to this image.

6.1.2 Input/Output Specification

Input: A sequence of images from the input component.
Output: The current orientation of the camera in world-space relative to the first image, described using the three Euler angles: pitch, yaw, and roll.

6.2 Domain

This section describes the domain in which the stabilization process is to operate. The purpose of describing the domain is to understand the problems caused by the environment and make it easier to select the best method for solving the stabilization problem.

The domain analysis is divided into three parts:

- Environment Composition
- Placeholder Item Motion
- Camera Motion

6.2.1 Environment Composition

The environment is divided into two main parts:

1. The table which is static

2. The part outside the table, which is not guaranteed to be static.

There are no constraints about motion outside the table.

Because of camera ego-motion the images of the image sequence does not contain the same amount of the two parts of the environment. Three cases describe the presence of the table inside the image. Figure 6.4 illustrates the three cases.

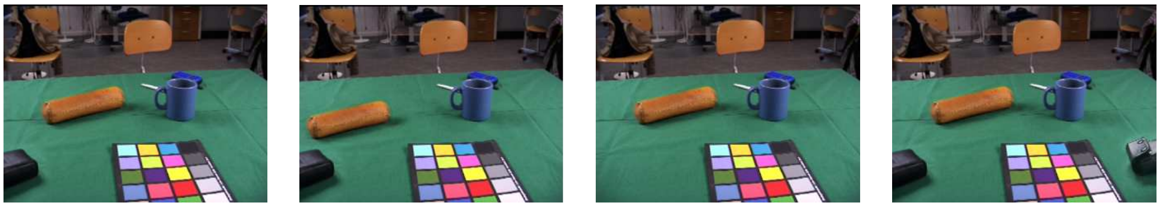
(a) (b) (c)

Figure 6.4: Figure (a) illustrates the case where the table covers the entire image. Figure (b) illustrates the case where the table covers part of the image. Figure (c) illustrates the case where the table is not present inside the image.

6.2.2 Placeholder Item Motion

In addition to the previous mentioned constraints concerning the table being static, movement of placeholder items placed on top of the table has to be taken into consideration. This is commonly referred to as *independent motion*. If a placeholder item moves, all pixels representing the placeholder item cannot be used to determine the ego-motion of the camera.

Independent movement of placeholder items can cause various motion in the image plane. Figure 6.4 illustrates three cases.



(a) (b) (c) (d)

Figure 6.5: Figure (a) illustrates an image captured during use. This will be used as a reference figure against figure (b), (c), and (d). Figure (b) illustrates a movement of a placeholder item (the brown pencil case) from one place in the image to another. Figure (c) illustrates the case where one of the placeholder items (the black box) disappear from the field of view, and figure (d) illustrates the case where a placeholder item enters the field of view.

6.2.3 Camera Motion

As described in section 3.4 the user is not allowed to move around, only move his upper body and rotate his head around the three degrees of freedom. Therefore, the primary ego-motion is rotation.

Figure 6.6 illustrates examples of rotation around all three axes (pitch, yaw, and roll).

(a) (b) (c) (d)

Figure 6.6: Figure (a) illustrates an image captured during use, and is used as a reference figure against figure (b), (c), and (d). Figure (b) illustrates a pure pitch, figure (c) illustrates a pure yaw, and figure (d) illustrates a pure roll.

6.3 Overview

The stabilization process can be subdivided into three subprocesses:

1. Calculation of the correspondence between images in the image sequence
2. Accuracy improvement by postprocessing of the correspondence data.
3. Conversion of the motion in the image plane to Euler angles in world-space

This division can be seen in figure 6.7. Part 1 deals solely with the difference between consecutive images, part 2 connects the two other parts, whereas part 3 deals with the conversion from difference in the image plane to the ego-motion of the camera in world-space.

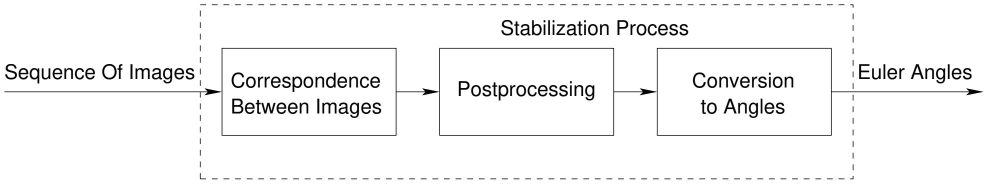


Figure 6.7: The three subprocesses of the stabilization process.

6.4 Correspondence Between Images

The correspondence part deals with the problem of calculating the displacement of the image elements between consecutive images, the so-called *disparity*.

The output of the correspondence part is a list of displacements, a so-called *disparity field*. Some methods only calculate disparity at a few points in the image, and therefore generate a sparse disparity field, whereas others generate dense disparity fields. Figure 6.8 shows a disparity field caused by pure roll.

Algorithms for solving the correspondence problem is typically divided into two categories:

- Feature-based methods
- Pixel-based methods

The difference between feature-based techniques and pixel-based techniques is that feature-based techniques explicitly utilize a priori knowledge of the contents of the images, whereas pixel-based techniques do not.

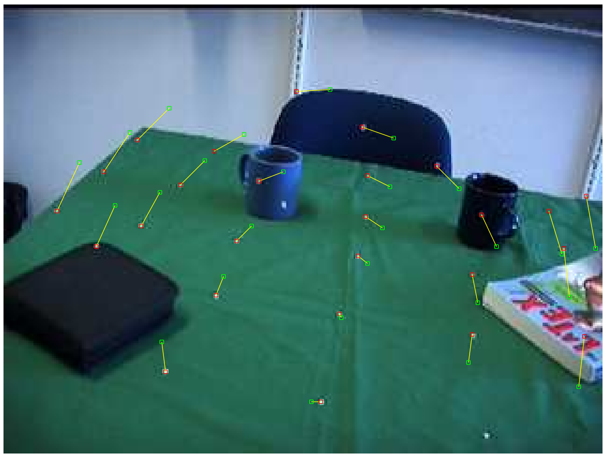


Figure 6.8: Disparity field from pure roll superimposed on image. The red dots show the positions in image i, and the green dots show the positions in image $i+1$

6.4.1 Applicable Methods

In the following a short description of a series of techniques for solving the correspondence problem will be described. For more information about the techniques see appendix D.

The described techniques are:

- Feature Matching
- Window Based Correlation
- Gradient Based Optical Flow

Feature Matching

Feature matching is a feature-based method, which works by extracting some kind of high-level features from the images of the sequence. The type of features depends on the application, but an example could be corners or lines. First features are extracted from each image independently, and then the features in image i and $i+1$ are matched with each other using a distance measure in feature-space. The best match is found, and the disparity is recorded. As the disparity is only recorded for each feature this method generates a sparse disparity field.

Pros: Robust against rotations and change of lighting depending on the chosen features. Fast.

Cons: Requires a priori knowledge of the image content. Requires that (a number of) features exists in all images.

Window Based Correlation

The Window Based Correlation method is a pixel-based method, and thus requires no a priori knowledge of the content of the images. The basic idea of the algorithm is to select image windows in image i using some kind of measure of their 'goodness'. A good image window contains structural information which makes it easy to uniquely locate it in the next image. After a series of image windows have been selected they are searched for in image $i+1$ using a similarity measure between two windows. An often used similarity measure is one divided by the sum of squared difference (SSD).

Pros: Does not require any a priori knowledge of the content of the images. Simple and relatively fast.

Cons: Not robust against change of lighting. Can generate a lot of outliers depending on the content of the images.

Gradient Based Optical Flow Estimation

The gradient method is a method for determining the optical flow between two consecutive images. When determining optical flow a motion vector is found for each pixel in the image for the motion from image i to image $i+1$.

The gradient based method uses a differential equation to express the image brightness function $I(x, y, t)$, describing the intensity at point (x, y) in the image at the discrete time t. Using the assumption that the intensity does not change during small motion the method produces an equation for the velocity vector v for each point (x, y) at the discrete time t. Unfortunately this vector has two unknowns in the equation. However, by assuming that the velocity field varies smoothly over an image the unknown variables can be found and the value for the displacement in the x- and y-direction can be found using iterative equations.

Pros: Precise at small motion.

Cons: Does not work with large motion. Requires texture in the images. Not robust against change of lighting. Slow.

6.4.2 Method Comparison

Table 6.1 compares the three methods based on which constraints they impose on the environment, their execution speed, density of the disparity fields, and robustness against noise in the images.

	Feature Matching	Correlation	Optical Flow
Constraints:	Features	Texture	Texture, small motion
Speed:	Fast	Medium	Slow
Density:	Sparse	Medium	Dense
Robustness:	Good	Fair	Poor

Table 6.1: Comparison of the three methods.

As described in section 6.2 the environment consists of two parts: the table and the background, which fill varying degrees of the images as the camera rotates. No a priori knowledge of the content and motion of the background is known, and it is therefore hard to select features which are guaranteed to work well when the table is outside the FOV. Therefore a pixel-based approach seems better suited for the problem.

The requirements (section 3.2) specify that the system must run at a framerate of at least 10 fps. In section 6.6 it is stated that a framerate of 10 fps leads to a maximum change in rotation of 13° of pitch and yaw between two consecutive frames. With this in mind the optical flow method does not seem to be a sensible solution.

6.4.3 Method Selection

Based on these considerations the Window Based Correlation method has been selected.

As described in appendix D.1, the algorithm for window based correlation contains several parameters, which can be tweaked to the particular problem at hand.

These parameters are:

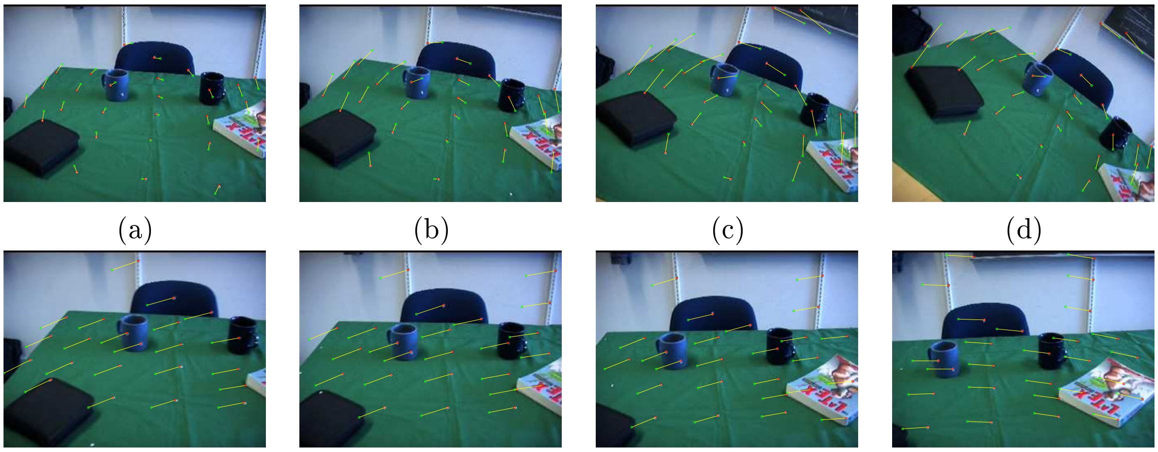
- Method for selecting start windows and number of start windows to use
- Representation of windows (histograms or raw data)
- Size of each window
- Distance measure
- Image resolution
- Color space

Of these parameters the choice of start windows, window representation, and distance measure can readily be selected. A fixed grid of 8 times 6 start windows will be used. Because the camera is allowed to roll, histograms has been selected as window representation, and the sum of squared difference on the histogram data has been selected as a distance measure.

Test

The choice of the other parameters are not as simple to determine. Instead a test must be conducted to search for the optimal parameter settings. In order to determine the optimal set of parameters a measurement of the quality of a disparity field must be defined. A possible measurement is the similarity to a *ground truth* disparity field.

In order to calculate *ground truth*, two test-sequences of 5 images each have been recorded. Sequence 1 contains pure roll, and sequence 2 contains a combination of pitch and yaw. In both sequences a series of small white markers have been placed in order to make it possible to calculate a disparity field between each consecutive frame by hand. As the markers are very small (around 2x2 pixels) their affect on the performance of the correlation algorithm is considered negligible. Figure 6.9 shows both sequences with the disparity fields superimposed.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6.9: The two hand-stabilized sequences. Figure (a), (b), (c), and (d) is from a sequence with pure roll, and figure (e), (f), (g), and (h) is from a sequence with a combination of pitch and yaw.

The similarity measure has been defined as the average sum of squared difference between the calculated displacement vectors and the nearest reference vectors:

$$D_{avg} = \frac{\sum_{i=1}^{n}(cal[i] - ref[\hat{i}])^2}{n} \tag{6.1}$$

Where n is the number of displacement vectors in the calculated disparity field, $cal[i]$ is the ith displacement vector, and $ref[\hat{i}]$ is the reference vector nearest $cal[i]$.

By this definition disparity fields with a smaller D_{avg}-value is of higher quality.

The task at hand now is to calculate a disparity field for each of the 8 pairs of two consecutive images using each of the possible configurations of the algorithm and select the combination with the smallest average D_{avg}-value. This would be the parameter combination which would generate disparity fields of the highest quality. As the correspondence algorithm is to be used in a real-time system, the effect on quality of the disparity field from various measures to decrease the computational load required by the stabilization process should be determined as well.

The parameter space to search has been selected as:

Histogram size

A histogram size of 16, 32, 64, and 128 buckets will be tested.

Window Size

A window size of 8x8, 16x16, and 32x32 pixels will be tested.

Image Resolution

Full (320x240) and half (160x120) resolution will be tested.

Color Space

As no a priori knowledge of the placeholder items are known it is necessary to use all three color components of RGB. From figure 5.5 it can be seen that the saturation channel of the HSV color space is able to differentiate between the table and the placeholder items fairly well. As the saturation channel furthermore is very robust against change of lighting, it is tested as well.

Threshold of Start Windows

The effect of using a threshold on the standard deviation of the start windows should be tested. The threshold setting has been imperially determined to be 15.

Results

The correlation algorithm has been executed on each of the images of the two sequences using each of the 96 parameter combinations (4 histogram sizes, 3 window sizes, 2 resolutions, 2 color spaces, and with/without threshold). The average of the disparity error for each sequence has been calculated, and the parameter combination has been sorted according to average disparity error. The 5 best combinations of window size and number of histogram buckets for each of the four combinations of color space and resolution is listed in table 6.2 and 6.3 for the case of no threshold, and table 6.4 and 6.5 for the case of thresholding. The fist number in each cell is the window size, the second number is the number of histogram buckets, and the number in parenthesis is the average disparity error.

	RGB full	RGB half	HSV full	HSV half
1.	32/64 (150.86)	32/16 (139.91)	32/64 (183.34)	32/128 (193.99)
2.	32/128 (152.02)	16/64 (158.78)	32/128 (191.91)	32/64 (230.01)
3.	16/128 (153.34)	32/64 (172.26)	32/32 (198.94)	32/32 (237.43)
4.	16/32 (156.45)	16/128 (179.66)	32/16 (260.56)	32/16 (285.82)
5.	16/64 (159.11)	32/32 (181.57)	16/128 (296.22)	16/64 (366.14)

Table 6.2: Sequence A with no threshold.

The first observation from looking at the four tables is that the quality of the disparity fields calculated from sequence B is much worse than the ones calculated from sequence A. The most obvious explanation is the fact that sequence B contains more motion than sequence A, because the mean disparity length is 18.5 pixels for sequence A and 32.1 pixels for sequence B calculated form the disparity fields calculated by hand.

Thresholding: The second observation is that thresholding improves the quality of the disparity

	RGB full	RGB half	HSV full	HSV half
1.	32/16 (763.07)	32/16 (693.08)	32/32 (685.45)	32/32 (739.17)
2.	16/32 (793.20)	16/16 (807.90)	32/128 (700.33)	32/16 (750.88)
3.	16/16 (800.14)	32/32 (847.13)	32/64 (705.27)	32/64 (759.92)
4.	32/32 (816.19)	32/64 (857.42)	16/32 (733.45)	32/128 (762.79)
5.	8/16 (828.70)	8/16 (878.17)	16/128 (740.72)	16/32 (1006.29)

Table 6.3: Sequence B with no threshold.

	RGB full	RGB half	HSV full	HSV half
1.	8/128 (47.62)	8/64 (69.75)	16/64 (122.39)	16/64 (120.74)
2.	16/32 (50.73)	8/32 (72.19)	16/128 (122.77)	32/64 (137.29)
3.	8/64 (55.55)	8/128 (83.67)	32/32 (133.57)	16/16 (138.08)
4.	16/64 (57.95)	16/64 (107.51)	32/128 (136.93)	32/128 (138.45)
5.	8/32 (65.16)	16/16 (108.05)	32/64 (139.21)	32/32 (162.75)

Table 6.4: Sequence A with a threshold of 15.

field. From the data in the four tables an improvement between 18 - 68 % can be seen from using thresholding.

Resolution: In order to determine the effect on the quality of the disparity fields of lowering the resolution, the difference in percentage between the quality of the best combination using full resolution and the best using half resolution has been calculated. The results can be seen in table 6.6.

Except for the two cases with a difference of 46.5% and 21.1%, all differences are less than 10% and to both sides. Therefore it is concluded that there is not a significant improvement from using full resolution.

Window Size: Looking at the 4 tables it is evident that a lot of parameter combinations have a disparity error very close to the best one, and it is therefore not enough to only look at 1st place in the 8 cases. Instead the dominant window size between the parameter combinations with an error less than 5% bigger than the best one, is selected for the each of the 8 cases. The results can be seen in figure 6.7 and 6.8.

From the two tables it can be seen that a 32x32 window is optimal in the case of no threshold, and in the case of thresholding 8x8 windows works best for RGB and 16x16 for saturation.

Histogram Size: Using the same approach as with the window size, the dominant histogram size in the 8 cases has been determined. The results can be seen in table 6.9 and 6.10.

As it can be seen from the two tables, no histogram size is significant better than the rest, therefore it is concluded that the right choice of histogram size is not very important. As 64 buckets are the most dominant size in 5 out of 16 cases it has been selected.

	RGB full	RGB half	HSV full	HSV half
1.	8/16 (532.71)	8/128 (569.63)	16/64 (455.66)	16/32 (359.61)
2.	16/32 (604.98)	16/16 (648.37)	16/128 (522.12)	32/16 (401.05)
3.	16/32 (625.63)	8/16 (670.64)	16/16 (530.83)	32/128 (460.82)
4.	8/128 (700.61)	16/32 (676.24)	32/128 (614.43)	32/32 (471.63)
5.	16/128 (716.35)	32/16 (678.18)	32/32 (616.02)	32/64 (473.26)

Table 6.5: Sequence B with a threshold of 15.

Seq. A RGB	Seq. A HSV	Seq. B RGB	Seq. B HSV
-7.3%	$+5.8\%$	-9.2%	$+7.8\%$
$+46.5\%$	-1.3%	$+6.9\%$	-21.1%

Table 6.6: The difference in percentage between full and half resolution. A negative value means that the lower resolution performs better. Top line is from no threshold, and bottom line is with.

RGB full	RGB half	HSV full	HSV half
32	32	32	32
16	32	32	32

Table 6.7: The most dominant window size for the 4 different cases without threshold. The top row is from sequence A and the bottom row is from sequence B.

RGB full	RGB half	HSV full	HSV half
8	8	16	16
8	8	16	16

Table 6.8: The most dominant window size for the 4 different cases with threshold. The top row is from sequence A and the bottom row is from sequence B.

RGB full	RGB half	HSV full	HSV half
128	16	64	128
16	16	32	32

Table 6.9: The most dominant histogram size for the 4 different cases without threshold. The top row is from sequence A and the bottom row is from sequence B.

RGB full	RGB half	HSV full	HSV half
128	64	16	16
8	8	16	16

Table 6.10: The most dominant histogram size for the 4 different cases with threshold. The top row is from sequence A and the bottom row is from sequence B.

Color space: The difference in disparity error in percentage of using saturation versus using RGB can be seen in table 6.11.

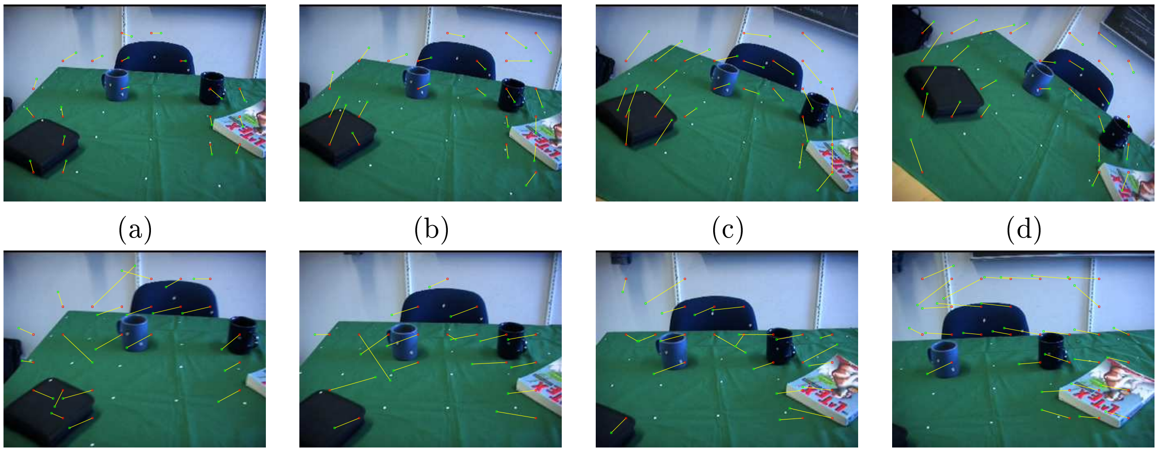
Seq. A full	Seq. A half	Seq. B full	Seq. B half
+21.5%	+38.7%	−10.2%	+6.7%
+157.1%	+73.1%	−14.5%	−36.9%

Table 6.11: The difference in percentage between RGB and Saturation resolution. A negative value means that saturation performs better than RGB. Top line is from no threshold, and bottom line is with.

From the table it can be seen that sequence A performs a lot worse with saturation than with RGB, whereas sequence B performs better. As the disparity fields of sequence B are much worse than the ones from sequence A, combined with the fact that the computational load from using saturation instead of RGB is approximately one third, saturation has been selected.

6.4.4 Parameters

The final selection of parameters are: A threshold of 15, half resolution, saturation, 16x16 windows and 64 buckets in the histograms. Figure 6.10 shows the calculated disparity fields using the selected parameters. As it can be seen from the figure the disparity fields from the selected approach contains false matches. Furthermore it is clear that sequence A is handled better than sequence B.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6.10: The calculated disparity fields using the selected parameters. Figure (a), (b), (c), and (d) is from sequence A, and figure (e), (f), (g), and (h) is from sequence B.

6.5 Postprocessing

Camera lenses and methods for calculating correspondence are not perfect, and therefore post-processing of the disparity field can seriously improve accuracy of the methods for converting the disparity into ego-motion of the camera in world-space.

6.5.1 Lens Distortion

According to (Trucco and Verri, 1998, p. 36) lens distortion can be modeled as radial distortion, described by:

$$x = x_d(1 + k_1r^2 + k_2r^4) \tag{6.2}$$
$$y = y_d(1 + k_1r^2 + k_2r^4) \tag{6.3}$$

Where (x_d, y_d) are the distorted coordinates, (x, y) the undistorted coordinates, r distance from the image center to (ax_d, y_d) and k_1, k_2 intrinsic parameters of the camera. From equation 6.2 and 6.3 it can be seen that the distortion is zero at the image center, and increases with the distance r from the image center.

If the intrinsic parameters are known, equation 6.2 and 6.3 can directly be used to unwarp the correspondence points.

6.5.2 Errors in Correspondence Calculation

Correspondence algorithms sometimes generate false matches, that is the pixel position found in image $i+1$ does not correspond to the same point in world-space as the one in image i. Such false matches are called *outliers*. Another source of errors are independent motion of the placeholder items.

Several techniques exists for removing outliers from the disparity field. An example of such an algorithm is the Local Continuity Method. This method is based on the fact that a disparity field of a static environment is smooth, that is displacement vectors have similar orientation and length as nearby vectors. The algorithm calculates a measure of the difference in orientation and size relative to a neighborhood of vectors, and outliers are rejected by simple thresholding.

6.6 Motion Conversion to Euler Angles

The third part of the stabilization process converts the disparity field into the three rotations (pitch, yaw, and roll) of the camera. In order to convert disparity to angles, each method considers the environment to be static between two consecutive images, and hence the displacement is strictly caused by the ego-motion (rotation of the camera). In a static environment with purely rotational ego-motion the environment can be viewed as an rigid object.

As the camera is head-mounted, it is possible to determine certain constraints on the rotations performed. In section 2.6 the maximum rotations using a table of dimensions 170x200 cm was estimated to be around $+/-30°$ of pitch and yaw, and around $+/-36°$ of roll. Furthermore the maximum angle velocity is determined to be around $130°/s$ for pitch and yaw, and around $90°/s$ for roll. As the table size is not guaranteed to be of the same dimensions as the one used for the measurements in section 2.6 a maximum angle of $+/-45°$ is selected for pitch and yaw.

Using a framerate of 10 fps the following constraints on the rotations can be determined:

Constraint	Pitch	Yaw	Roll
R_{max}	$45°$	$45°$	$36°$
V_{max}	$13°/f$	$13°/f$	$9°/f$

Table 6.12: The ego-motion constraints. R_{max} is the maximum angle $(+/-)$ which the camera can be rotated around the three axes, and V_{max} is the maximum difference in angle between two consecutive frames.

6.6.1 Applicable Methods

In the following a short description of a series of techniques for solving the problem of converting the disparity to angles will be described. For more information about the techniques see appendix

D. The described techniques are:

- 2D Affine Model
- Hyperbolic Projection Model
- Tensor Model
- Mosaic

2D Affine Model

The 2D affine model is based on the observation that a small camera rotation in world-space of a static environment will cause a 2D affine transformation of the rigid body described by the starting points of each displacement vector into the rigid body described by the end points of the displacements. In this 2D affine transformation the rotational component describes the roll, and the translation part describes the pitch and yaw. The components of this 2D transformation can be directly found using the least-squares solution to the linear equation $\mathbf{A}\overline{X} = \overline{B}$, where $\mathbf{A}$ is the parameters of the transformation, and $\overline{X}$ and $\overline{B}$ are the coordinates of the corresponding points in frame i and $i+1$.

Pros: Fast and simple.

Cons: Only accurate if the field of view of the camera is small. Because of the least squares approximation, it is sensitive to outliers.

Hyperbolic Projection Model

The Hyperbolic Projection Method builds upon the observation that the projection of a point in world-space under pure pitch or yaw follows a hyperbolic track in the image plane, and that a movement in the image plane caused by a combination of pitch and yaw can be divided into a combination of the hyberbolic tracks caused by the pitch and yaw.

Pros: Relative fast. Handles large rotations. Only requires one correspondence.

Cons: Does not handle roll.

Trilinear Tensor

The trilinear tensor is an extension to the fundamental matrix, the point, and line geometry of three images. It describes the spatial relation of three cameras, or three consecutive images. The trilinear tensor is an array of 27 entries, and denoted as α_i^{jk}, for $i, j, k = 1, 2, 3$. The trilinear tensor estimates the rotation around all three axis (pitch, yaw, and roll) simultaneously, with the constraints of small rotation and a minimum of seven corresponding points in the three consecutive images.

Pros: It estimates the three rotations simultaneously, and there is no 3D constraints to the scene (environment).

Cons: Normally based on more than the minimum seven points, an increase in the number of points resolve in a more accurate trilinear tensor. The 27 entries to the array is calculated using a SVD, and it does not work on large rotations.

Mosaic

Mosaic is normally used for reconstructing a scene from adjacent sub images. In order to map two sub images together, it requires the knowledge of the translation, rotation, and scale between the sub images. It is this knowledge that is used for stabilization. The translation is estimated using Fourier Transformation, while the rotation and scale is estimated using Zernike moments. The translation part describes the pitch and yaw, and the rotation part describes the roll. The mosaic works on corresponding points in the two sub images previously found.

Pros: It only requires a little overlap between the two sub images (10 - 20 %), in case of small rotation, and translations it is proven to be accurate.

Cons: It makes use of Fourier Transformation which requires a lot of computation, and it does not work on larger rotations.

6.6.2 Method Comparison

Table 6.13 compares the four methods on which constraints they impose on the ego-motion, their execution speed, and robustness against outliers in the disparity fields.

	2D Affine	Hyperbolic	Tensor	Mosaic
Constraints:	Small rotations, small FOV of camera	No roll	Small rotations	Small rotations
Speed:	Fast	Medium	Medium	Slow
Robustness:	Poor	Fair	Fair	Good

Table 6.13: Comparison of the four methods.

As the hyberbolic method does not handle roll, it cannot be used. All three remaining methods are only precise at small rotations. The tensor model and mosaicing both are robust against outliers in the disparity field, whereas the 2D affine model is not. Unfortunately they are both complex and computationally heavy compared to the 2D model.

6.6.3 Method Selection

Based on these considerations the 2D affine method has been selected.

The angles found by the algorithm are the difference in angles between two consecutive images. To get the absolute (relative to the very first image) rotation, the angles are simply summed up.

6.6.4 Conversion from Pixels to Euler Angles

because of ego-motion of the camera, it is necessary for the tracking component to track objects in world-space, not in the local coordinate system of the camera. As objects may be positioned at arbitrary positions in the image plane, and the stabilization process only delivers the world coordinates of the center of the image plane it is necessary to be able to calculate the world coordinates of an arbitrary point in the image plane.

The task at hand is to convert a pixel position in the image plane to a description (pitch, yaw) in world-space given the three angles from the stabilization process and the camera parameters, in particular the focal length.

The very first step is to convert the pixel positions (x, y) to image plane coordinates (u, v):

$$u = \frac{x - xres/2}{xres/2} \tag{6.4}$$

$$v = \frac{y - yres/2}{yres/2} \tag{6.5}$$

As (pitch, yaw) fully describes a point on the surface of the sphere (plus the implicit radius), we need to counter the roll of the image plane. This is done by rotating the point in image space by the opposite angle, $-roll$.

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} cos(-roll) & -sin(-roll) \\ sin(-roll) & cos(-roll) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{6.6}$$

As the image plane is now coplanar with the world coordinates, the angles of (u, v) can be calculated using simple geometry.

The two angles can be calculated directly from the displacement d from the image center using right-angled triangles.

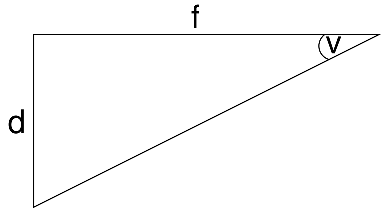


Figure 6.11: Conversion between displacement d and angle v given the focallength f

$$v = tan^{-1}(d/f) \tag{6.7}$$

And hence the final conversion algorithm is:

$$\begin{bmatrix} P_{pitch} \\ P_{yaw} \end{bmatrix} = tan^{-1}(\frac{1}{f} \begin{bmatrix} cos(-roll) & -sin(-roll) \\ sin(-roll) & cos(-roll) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}) \tag{6.8}$$

6.7 Summary

A stabilization system for determining the ego-motion of a purely rotating camera has been developed. The stabilization system consists of three parts: Determining the correspondence between images, postprocessing of the disparity field, and conversion of the disparity into ego-motion.

The window based correlation technique has been selected for determining the correspondence between images, and a test has been performed in order to determine the optimal parameters of the method. The selected parameters were half resolution, the saturation color channel, a threshold on the standard deviation of the start windows, windows of 16x16 pixels, and histograms with 64 buckets.

The correspondence algorithm could be improved by taking advantage of the temporal continuity of the disparity, and using a predictor on the size and position of the windows on interest. This would both improve accuracy and possible lower the computational load, as the windows of interest can be dynamically shrunken.

Accuracy could be improved with outlier rejection. The model with outlier rejection as postprocessing is perhaps too simple. Algorithms as the RANSAC algorithm (Trucco and Verri, 1998, p. 302) for rejection of outliers works as a feedback loop on the entire stabilization process.

The 2D affine model was selected for the task of converting the disparity field into ego-motion of the camera. This was primarily selected because of its simplicity.

The algorithm for conversion to angles does not contain any method for synchronization the angles with the environment, and will therefore accumulate errors as the system is running. This will cause the calculated angles to drift away from the actual angles of the ego-motion.

Furthermore the constrains on ego-motion described in section 6.6 are not used. As the 2D affine model is very sensitive to outliers, the constrains could be used to detect if the set of correspondences uses to calculate the angles contains excessive outliers. If this happens a new subset of the available correspondences can be selected to be used to calculate the angles.

CHAPTER 7

Segmentation

This chapter describes the segmentation component of the system. The location of the component in the overall system can be seen in figure 7.1.

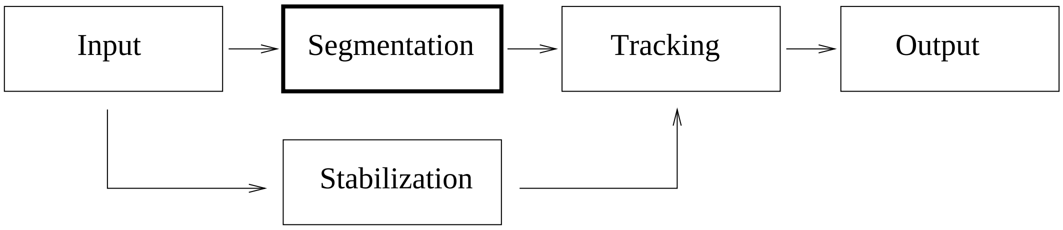


Figure 7.1: Location of the segmentation component.

7.1 Purpose

One of most the important stages in digital image processing is segmentation, because this process removes the unwanted and unimportant information. The specific purpose of the segmentation component in this project is to identify and separate placeholder items from the table in the acquired image. The set of pixels that contains the segmented objects are often called **B**inary **L**arge **OB**jects (BLOBs).

The segmented objects are then used by the tracking component (described in chapter 8) for feature extraction, matching, etc. Fast and accurate segmentation is in general important for any computer vision system to be successful.

It is important to note that the purpose and output of the segmentation stage is not perfect object separation. Segmentation of arbitrary (shape, texture, color, etc.), occluding objects in the image plane is a hard problem to solve using known methods, and it becomes even more difficult in real-time systems. Therefore, occluding objects are regarded and segmented as one BLOB, although this is incorrect. Chapter 8 discusses an alternative strategy for dealing with occluding objects, or tracked objects which have not been correctly identified.

7.1.1 Definitions

The term "training image" in the following refers to a reference image which is analyzed, and from which parameters are extracted for later use for classification purposes. Furthermore, instead of specifically expecting 8-bit integer color channels, normalized, real values between 0 and 1 are used in the images and color models described below.

7.1.2 I/O Specification

Input: An image from the input component.
Output: Segmented objects (BLOBs) placed on the table found inside the image.

7.2 Domain

The decisions made in section 3.4 with respect to the environment and placeholder items are important during the segmentation process. It is essential for the method described in this chapter that:

1. It was decided to cover the table with a green cloth, which is "... an uniform, matte surface that is easily distinguishable and merely a diffuse reflector".
2. The placeholder items are rigid and at the same time colored with high contrast relative to the green cloth.

Obviously, the philosophy behind these constraints are that they enable the use of fast, well-known segmentation methods.

Figure 7.2 shows a typical frame from one of the acquired sequences. Although there are five different objects on the table, the segmentation output in this case is three BLOBs due what was previously stated (the pen, cup and cellular phone counting as one).

Example image

Figure 7.2: Example image showing the environment, table and various placeholder items.

7.3 Overview

It is apparent that the segmentation process can be conveniently decomposed into two subprocesses, where the second subprocess depends on the output from the first. These subprocesses are:

Table pixels segmentation: Prior to the actual object segmentation the green cloth must be properly determined.

Object segmentation: Using information from the previous subprocess, this is the actual segmentation of object regions inside the table.

Figure 7.3 depicts this subdivision.

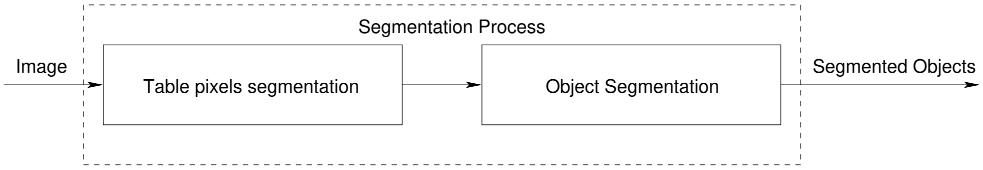


Figure 7.3: Segmentation subprocesses.

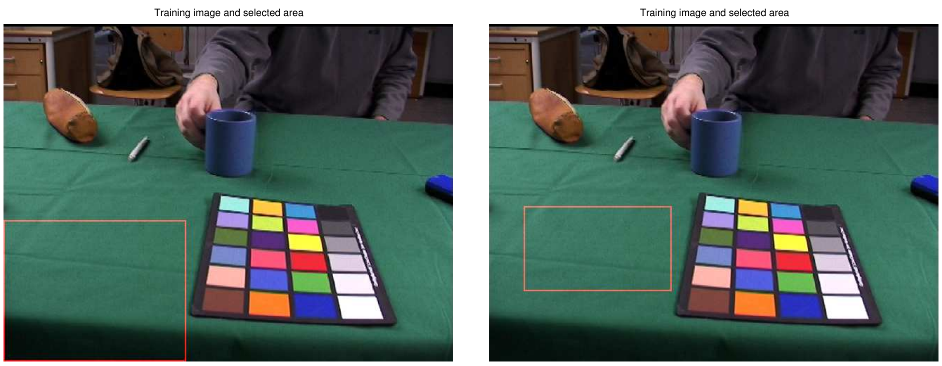
The design of these subprocesses are found in section 7.4 and 7.5, respectively.

7.4 Table Pixels Segmentation

Before the table segmentation can be performed the color model and channel(s) of interest must be determined. Appendix B describes some well-known color models. Of these the possibility was either the "rgb" or "HSV" color model due to their ability of separating chromaticity and illumination. This property is important for reliable segmentation where illumination may vary. In most cases brightness change due to shadows casted by the objects, but also to some extent from creases in the green cloth.

It is only necessary to examine two of the three color channels in the rgb color model, as no additional information is contained in the third channel (e.g., $b = 1 - r - g$). Merely the hue channel is examined for the HSV model, as it contains just the color information.

A training image was randomly picked from one of the acquired sequences, and used for examining the color models. The significance of shadow pixels was examined by selecting a rectangular window region with both a normally lit area and pixels in shadow (see figure 7.4(a)). The color model that shows the best separation ability was then tested on a window consisting of just well lit pixels (see figure 7.4(b)) for comparison.



(a) (b)

Figure 7.4: Figure (a) Training image with a region containing pixels directly luminated and in shadow. Figure (b) Same image as (a) except that a smaller region with no pixels in shadow is defined.

The result of plotting the window from figure 7.4(a) in rgb color space can be seen in figure 7.5. Just the r- and g-channels are shown.

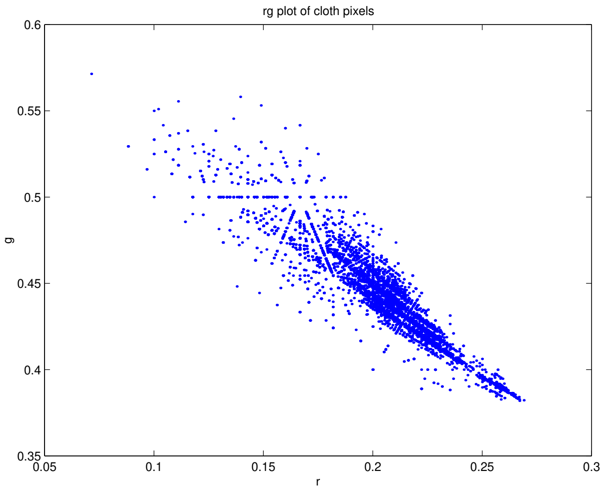
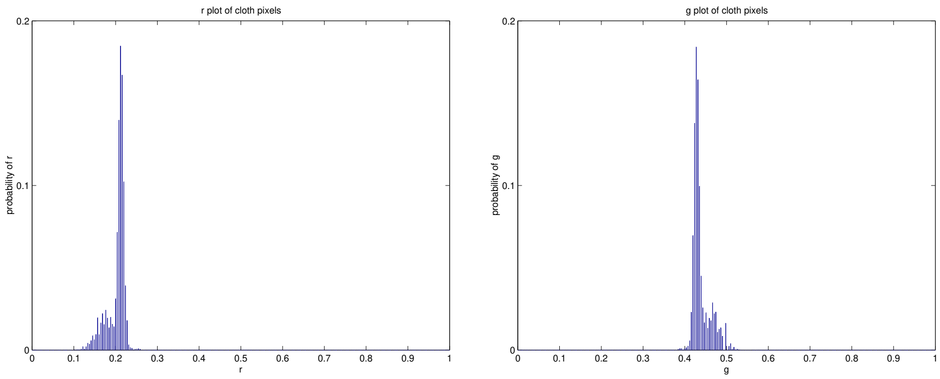


Figure 7.5: (r,g)-chromatic plot of cloth pixels from the rectangular region in figure 7.4(a) ($\mu = (0.2047, 0.4403), \sigma = (0.0213, 0.0208)$).

Generating histograms as the probability distribution of the r- and g-channel results in figure 7.6. It can be observed that none of the channels exhibit normal distribution. This is due to the rgb color model's inability to separate saturation from chromaticity (Bradski, 1998).



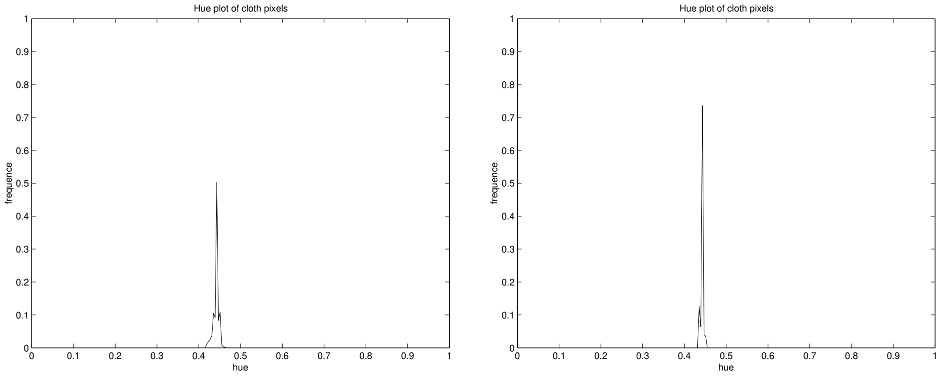
(a) (b)

Figure 7.6: Figure (a) Histogram of the r-chromatic channel (region in figure 7.4(a)) showing a wide, definite non-Gaussian distribution. Figure (b) Histogram of the g-chromatic channel calculated from the same region as (a). The same remarks apply for the g channel.

The result of plotting the window from figure 7.4(a) in HSV color space can be seen in figure 7.7(a) as a probability distribution, where just the hue channel is shown. It is observed that the hue channel has an approximately symmetrical, normal shape, and a very distinct peak.

By comparing the rgb and HSV color models it must be concluded that although more computationally demanding (for conversion), the hue channel from the HSV model is the better at finding a desirable, narrow distribution which can be used for proper segmentation. The hue channel is thus used for segmentation as described below.

The result of plotting the hue channel as a histogram of the window in figure 7.4(b) is shown in 7.7(b). By comparing 7.7(a) and 7.7(b) it should be noted that not even the HSV color model is capable of separating chromaticity ideally when illumination (and/or saturation) changes. This is partly due to numerical and quantization problems, but also caused by the problems of transforming color between its spectral components and a (more practical) trichromatic color model used for computing. Refer to section B.2 for more details concerning color image formation.



(a) (b)

Figure 7.7: Figure (a) Hue channel histogram from the rectangular region in figure 7.4(a) with pixels directly lit and shadow ($\mu = 0.44, \sigma = 0.0065$). Figure (b) Case showing a very distinct peak in the hue channel histogram, when choosing the region with well-illuminated pixels from figure 7.4(b) ($\mu = 0.4415, \sigma = 0.0037$).

7.4.1 Applicable Methods

Figure 7.8 shows the hue channel of figure 7.4(a), as an example from which table pixels must be extracted.

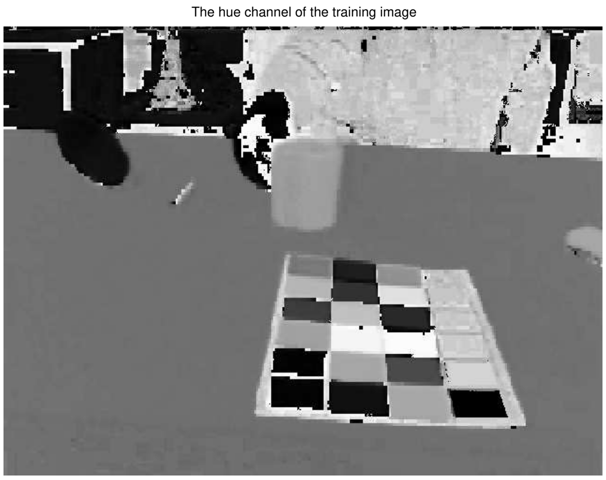


Figure 7.8: The hue channel of figure 7.4(a) as a grayscale image.

In the following a selection of applicable segmentation techniques are briefly described for segmenting the hue channel. Appendix C contains more detailed descriptions of the segmentation methods. The described techniques are:

- Edge detection
- Global thresholding
- Uniform quantization
- Pixel aggregation
- Region splitting and merging

Edge Detection

Segmentation based upon edge detection relies on discontinuities between homogeneous regions. By calculating local derivatives in the image, edges can be detected. The Sobel operator is a popular operator for finding these derivaties. For the segmentation to be useful in the context of this project, edges must subsequently be linked to obtain closed boundaries by examining either magnitude or angle. Additional processing is necessary, however, for finding specific modalities like the table.

Pros: Useful for finding discontinuities like the edge between background and table.

Cons: Multiple image passes necessary.

Global Thresholding

Global thresholding is based on the principle of simply comparing each pixel value to determine whether it is smaller or greater than some predefined threshold. This threshold is typically determined by some histogram-based, probabilistic method. As global thresholding uses static thresholds the success of the method relies on these bound(s) being chosen carefully.

Pros: Simple and very computationally efficient solution.

Cons: Static thresholds means no local adaption.

Uniform Quantization

Actually quantization can be understood as a form of global thresholding with multiple bounds. The number of discrete values is reduced in a many-to-one relation by exploiting similarity, e.g. achieved by division followed by rounding or efficiently by arithmetic bit-shifting. Hereafter there may be several unconnected regions defined by each value. However, for finding table regions this method requires postprocessing.

Pros: Simple and efficient.

Cons: As for global thresholding (static thresholds), but requires additional processing for table region determination.

Pixel Aggregation

Region growing is a popular segmentation approach (e.g., (Birk et al., 1996)) that repeatedly groups pixels recursively until some "growing predicate" is not satisfied. A special case is "pixel aggregation", which is a simple region growing method where pixels are grown from a "seed" point and appended to the segmented region if they satisfy some similarity criterion (texture, color, etc.). The algorithm can be improved by determining similariy in a local region (e.g., 3×3 or 5×5 pixels). If the "seed point" can be determined in advance pixel aggregation is a robust segmentation technique.

Pros: Fairly robust.

Cons: Algorithm requires a "seed point".

Region Splitting and Merging

By initially subdividing the image into a set of arbitrary, disjointed regions followed by a series of merges and splits, very robust segmentation can be achieved (e.g., (Leppajarvi, 1999)). The method is identical to the pixel aggration method in that the algorithm proceeds as long as some predicate is fulfilled. With a priori knowledge of the features of the table this algorithm would allow robust segmentation.

Pros: Robust segmentation.

Cons: Some variants are complex to implement and relatively slow.

7.4.2 Method Comparison

Table 7.1 compares the five methods on which constraints they impose on the environment, their execution speed, and robustness without prior image processing.

	Edge detection	Global thresholding	Uniform quantization	Pixel aggregation	Region splitting and merging
Constraints:	Discontinuities	Static	Static	"Seed" point	Table features
Speed:	Medium	Very fast	Fast	Medium	Slow
Robustness:	Fair	Fair	Poor	Good	Good

Table 7.1: Comparison of the five methods.

Three conclusions can be made from the table:

1. *Region splitting and merging* is the optimal solution for offline applications.
2. *Global thresholding* is the fastest and simplest to implement.
3. *Pixel aggregation* is probably the best trade-off between speed and robustness.

7.4.3 Method Selection

Global thresholding has been selected as the preferred method for segmentation primarily due to its simplicity and reasonable robustness, although proper threshold values must be found at runtime. As long as lighting does not change characteristic drastically at runtime, global thresholding is assessed to achieve good segmentation results, which, if necessary, may be enhanced by additional postprocessing.

7.4.4 Specific Design and Implementation Issues

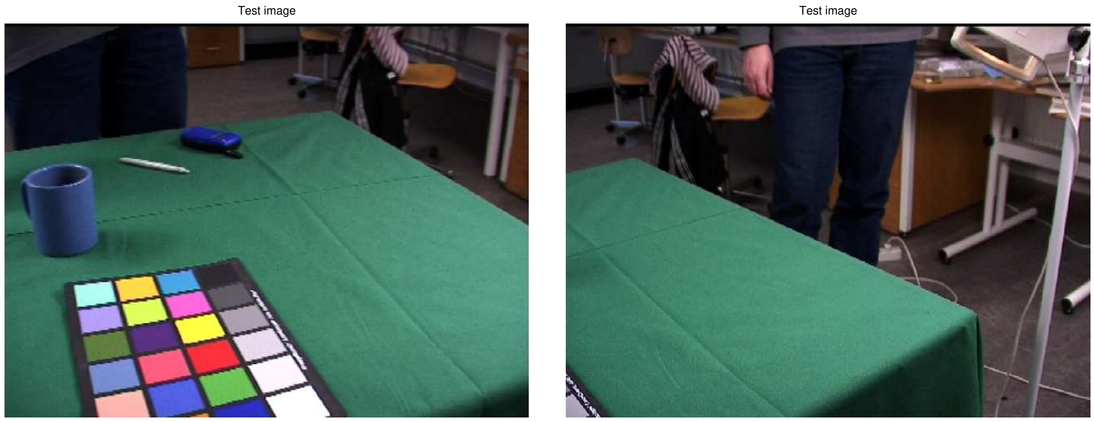
The specific design of the segmentation component is described in the following. Prior to actually applying global thresholding, a method for determining suitable thresholding bounds is presented. It is expected that a HSV converted version of the input is available for color segmentation. Figure 7.9 illustrates the images used for demonstrating the effect and possible problems of the methods and techniques in the segmentation procedure.

Threshold Bounds Determination

Before applying the global thresholding for segmenting table pixels, the lower and upper bound, T_{low} and T_{high} respectively, must be determined. As it is assumed that lighting conditions are somewhat static these bounds can be found during system initialization and then remain constant.

Figure 7.10 shows the previously discussed probability distribution histogram of a selected rectangular window of table pixels from the training image. Bounds for the thresholding can be found by analyzing the histogram. This assumes that the histogram was calculated from a representative sample of table pixels.

Initially the location of the peak value, hue_{peak}, is determined by the position of the highest probability. T_{low} and T_{high} are then initialized to the value of hue_{peak}. Hereafter the expected



(a) (b)

Figure 7.9: Figure (a) Segmentation design test image used as example in the following. Figure (b) Another segmentation test image.

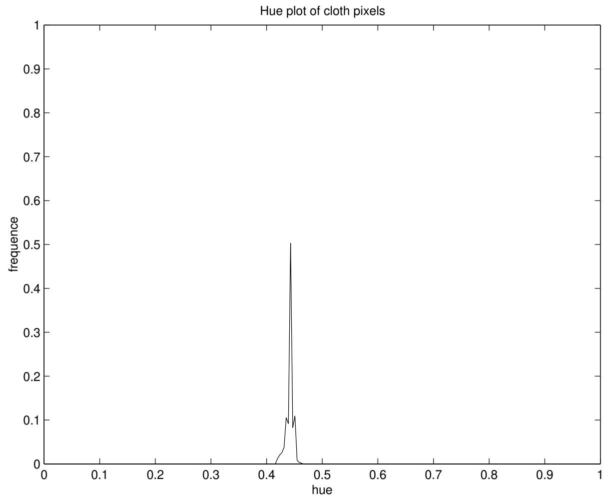


Figure 7.10: Hue channel histogram from the rectangular region in figure 7.4(a) ($\mu = 0.44, \sigma = 0.0065$). Identitical to figure 7.7(a).

symmetrical shape is exploited by incremental decreasing and increasing of T_{low} and T_{high} until some predefined percentage of the histogram mass, P_{mass}, is covered.

$P_{mass} = 0.99$ is used in the current implementation. $P_{mass} = 1.0$ would not make a sensible choice, since just a single noise pixel could then result in erroneous thresholding bounds and results.

To increase robustness, T_{low} and T_{high} are finally expanded relative to hue_{peak} by a small factor, T_{factor}, and a constant, $T_{constant}$, using equation 7.1. Tests indicated that conservative choices for these variables are $T_{factor} = 1.3$ and $T_{constant} = 0.04$ (or 10 levels for 8-bit color channels), which ensures reliable segmentation, even for pixels in shadow.

$$\begin{aligned} T_{low} &= hue_{peak} - T_{constant} - (hue_{peak} - T_{low})T_{factor} \\ T_{high} &= hue_{peak} + T_{constant} + (T_{high} - hue_{peak})T_{factor} \end{aligned} \tag{7.1}$$

Figure 7.11 shows the histogram with thresholds included.

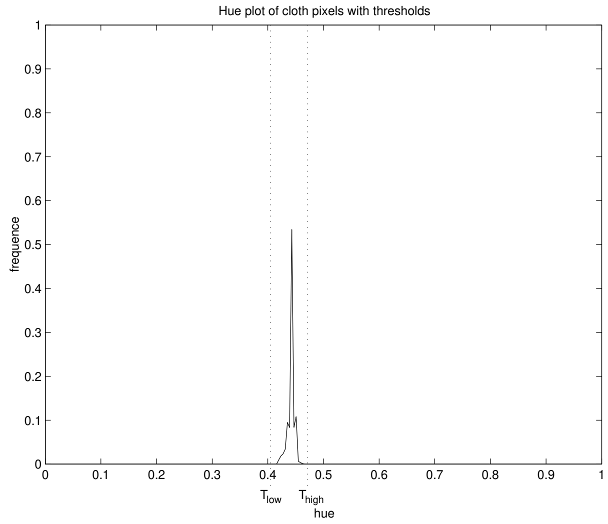


Figure 7.11: Figure 7.10 with global thresholding bounds, T_{low} and T_{high}.

Global Thresholding

When the thresholding bounds have been properly determined, simple global thresholding can be applied. A binary mask image at the same size as the input image is the output/result of the global thresholding operation. By traversing the entire input, corresponding pixels in the binary mask image are then either set or cleared.

To compensate for desaturated colors being erroneously classified as pixels belonging to the table, two comparisons is made for each pixels. Both of the following conditions must be satisfied for a mask pixel to be set:

1. $T_{low} < hue(x, y) < T_{high}$, where $hue(x, y)$ is the pixel from the hue channel of the input image to be tested.
2. $S_{low} < saturation(x, y)$, where $saturation(x, y)$ is the corresponding pixel from the saturation channel.

$S_{low} = 0.4$ (or 100 for 8-bit channels) is used in the current prototype. The choice of S_{low} depends on the signal-to-noise ratio (SNR) of the input image, where a lower SNR yields a higher S_{low} value.

Figure 7.12 shows the result of global thresholding. Global thresholding is clearly successful at segmenting the table pixels.

Binary Morphology

The image in figure 7.12 contains artifacts, for instance near some of the table edges. Binary morphology is a post processing method for removing such unwanted artifacts, although the image mask is slightly disorted from the structuring element.

The *opening* operator (Gonzalez and Woods, 1993, p. 524) (erosion followed by dilation) is applied to the image mask using a square 5×5 pixel structuring element. A smaller 3×3 structuring element did generally not remove all artifacts.

Figure 7.13 shows the result of applying the *opening* operator on the table pixel mask from the figure 7.12, and it is observed that the table contour is generally smoothed as expected.

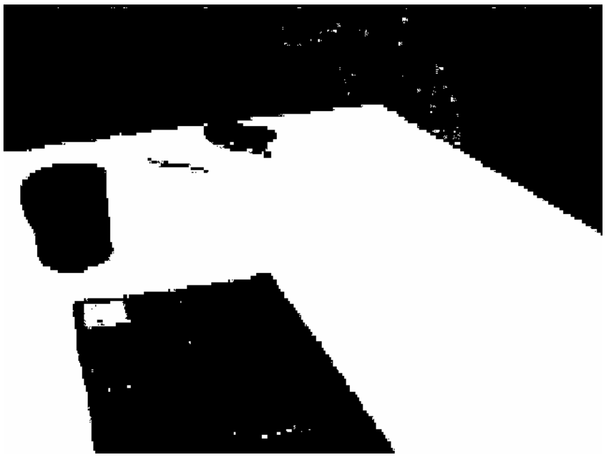


Figure 7.12: Result after global thresholding figure 7.9(a).

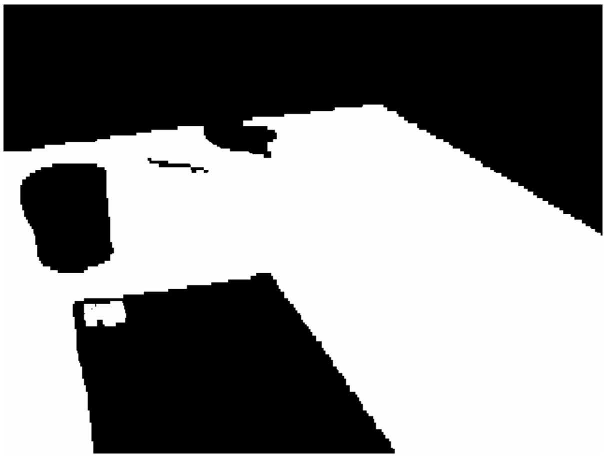


Figure 7.13: Result after performing binary morphology on figure 7.12.

Pin Pointing

From figure 7.13 it can be seen that the image mask still contains a closed region from the color checker object. However, output of the table segmentation procedure must not include any objects. Therefore regions with smaller area than some predefined constant must be removed from the table mask, with the consequence of only leaving the table region.

This procedure is often referred to as "pin pointing". All external contours (represented by chain codes, see appendix A) and their area are determined, and regions smaller than A_{min} pixels is erased.

$A_{min} = 10\%$ of all pixels in the image plane is currently used as threshold.

Figure 7.14 shows the result of pin pointing the table region in figure 7.13.

Concluding Remarks

Of the parameters mentioned previously the most important is the thresholding bounds, T_{low} and T_{high}. If the range covered by these variables are too narrow, table segmentation fails to generate a correct, uniform mask. Figure 7.15 shows an example of this. Conversely, if the range is too wide more non-table pixels are thresholded and possibly included as the table mask.

Both situations could pose a potential problem in the object segmentation stage, which is described in the next section.

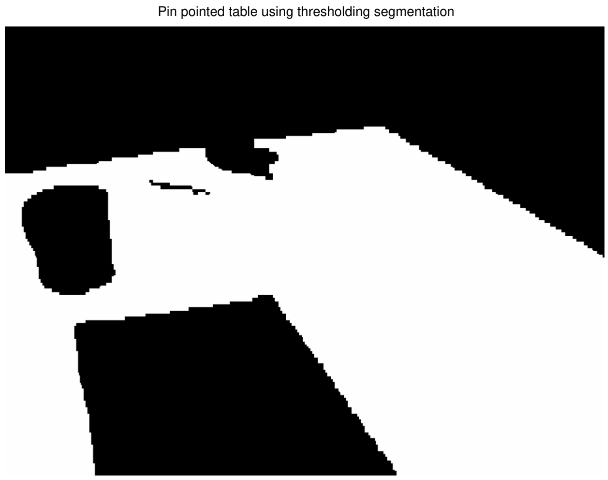


Figure 7.14: Result after pin pointing table regions from figure 7.13.

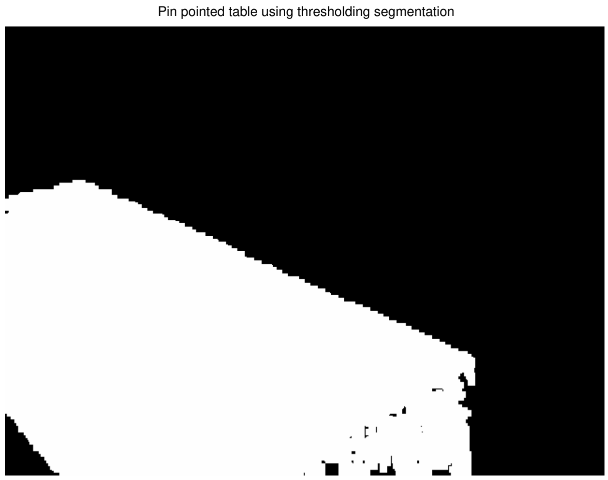


Figure 7.15: Result after performing the table pixels segmentation procedure on figure 7.9(b), illustrating the importance of proper thresholds.

7.5 Object Segmentation

The table segmentation procedure was described in the previous section, which produced a binary mask as output with just the table pixels using a color segmentation technique.

7.5.1 Specific Design and Implementation Issues

Before finishing segmentation the BLOBs on the table must be found. This can be achieved by continued processing on the binary table mask.

Applying the Convex Hull

The mask from figure 7.14 contains some closed regions, which could easily be extracted. However, objects near the image border or outer edges of the table such as the cellular phone can not be extracted using this method. For extracting those objects positioned near the extremes a mask of the table without holes is necessary. The convex hull (see appendix A) is a general method for finding the optimal mask.

Initially the contour of the external table region is calculated as an accurate polygon. Hereafter, the convex hull is determined and the convex hull is rendered in a new image, as a separate mask.

Figure 7.16 shows the result after the convex hull mask has been calculated and rendered for figure 7.14.

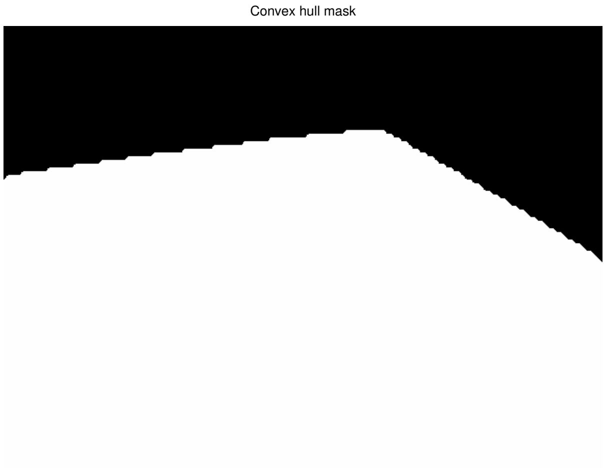


Figure 7.16: Result after calculating the convex hull mask of 7.14.

Figure 7.17 indicates that without a priori knowledge about the environment, the algorithm is unable is unable to detect that the color checker plate in the lower left corner is placed on the table. However, the convex hull is regarded as the optimal, unconstrained solution to the problem.

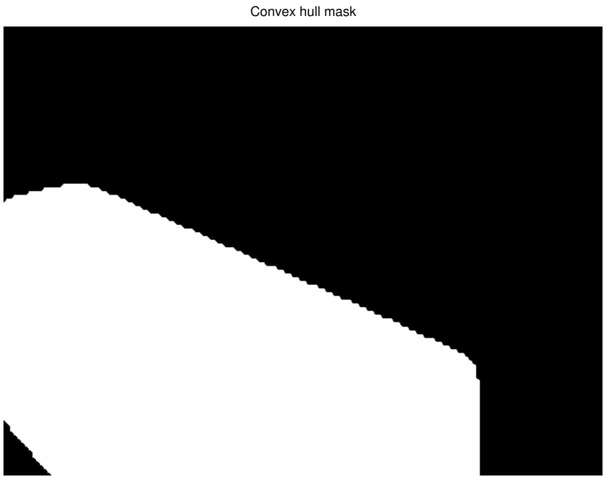


Figure 7.17: Result after performing the table pixels segmentation and calculating the convex hull mask of 7.9(b), illustrating problems with the table region not being complete.

Combining the Masks - Finding the BLOBs

By traversing each pixel in both the convex hull mask and original table mask, and calculating the logical exclusive-or (XOR) operation between the two images, a final combined mask can be created where the objects are separated, and thus can be segmented directly.

By performing this operation, table pixels set in both images are cleared, closed objects in the original image remain, and objects at the outer area of the mask become BLOBs whose mass inside the image plane can be segmented.

To avoid segmenting very small objects that are the outcome of lacking precision between the digitized convex hull mask and the table image mask, the pixel area of the objects in the combined mask must be greater than O_{min}. Objects satisfying this condition are then the output/result of the object segmentation step, and both their mask and segmented region from the original image is saved for later use in the feature extraction stage of the tracking component (chapter 8).

$O_{min} = 0.2\%$ of all pixels in the image plane is currently used as the minimum object size threshold.

Figure 7.18 illustrates the segmented object visualized with independent colors.

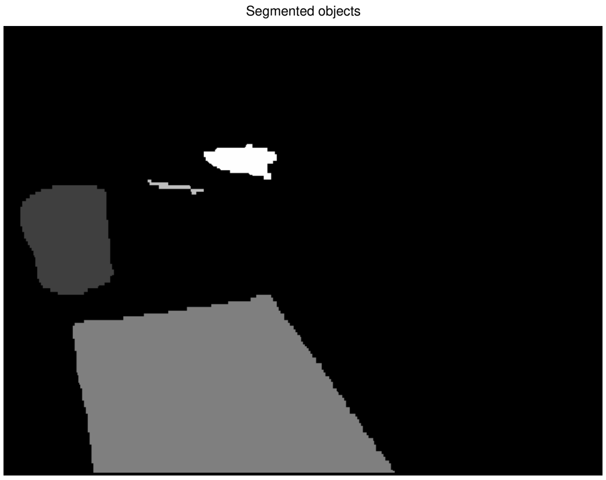


Figure 7.18: Final result after combining figure 7.14 and 7.16 and coloring each BLOB.

Concluding Remarks

The preceding procedure assumes the table (and cloth) are convex. If this assumption is not satisfied the algorithm will find false objects at concavities. However, provided the working environment is not too unusual this will normally not pose a problem.

7.6 Summary

To summarize, the segmentation component was subvided into two subprocesses, each which was previously described in detail. Thresholding was used for segmentation in the hue channel of a HSV image, and objects found were segmented using the convex hull.

A test of the subprocesses has been carried out as part of the system test in chapter 9.

Figure 7.19 shows a block diagram of the proposed solution.

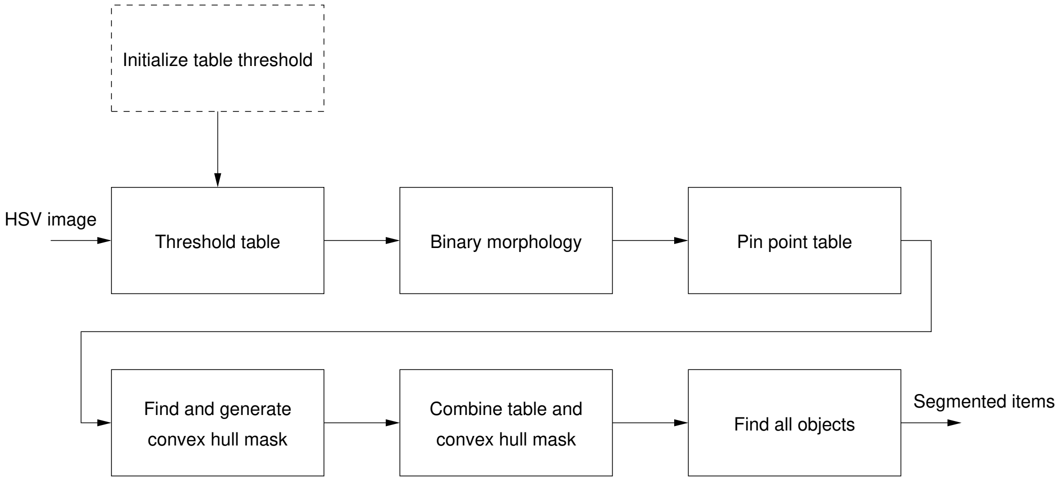


Figure 7.19: Block diagram for the segmentation component.

CHAPTER 8

Tracking

This chapter describes the tracking component of the system. The location of the component in the overall system can be seen in figure 5.1.

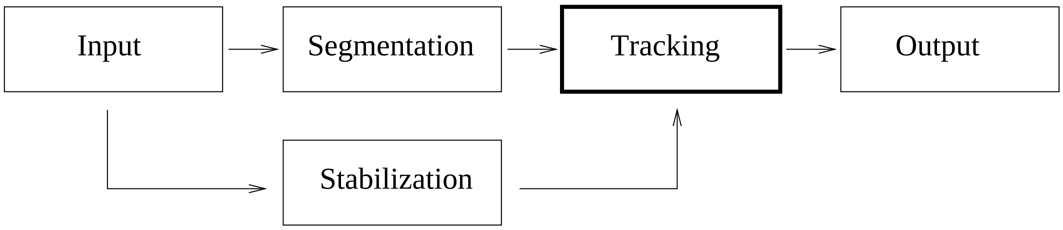


Figure 8.1: Location of the tracking component.

8.1 Purpose

Based on the output of the segmentation and stabilization component, which is segmented objects and the world-space orientation of the camera, the tracking component must locate and recognize the placeholder items present in the acquired image. These placeholder items are stored in a database, and are explicitly added upon a user-triggered event (in the current prototype this event is a keyboard stroke).

8.1.1 Definitions

Tracking objects is dependent on the ability to predict the trend of e.g. a movement (or some other feature) between consecutive frames. This is only possible if the system is able to register the time passed between events with reasonable granularity. A granularity of 1 ms was selected, which is adequately precise, as an update frequency of 10 Hz corresponds to 100 ms. Usage of the term "system time" in the following refers to a reading of the system clock containing the present time in seconds and milliseconds.

A database containing the description of placeholder items is used for representing the state of the system, referred to as the "item database" in the following. The item database is basically a primitive container for which items can be added, removed, and processed sequentially.

Placeholder items are computationally defined as an structure composed of several simpler entities/elements. Reasons for having specific elements are discussed in the design below. Regarding the "feature vector", its contents of presently used features are found in section 8.4.

A placeholder item currently consists of the following entities:

Object ID: An integer representing the object identification number as an unique, global quantity.

Binary mask: The binary mask produced by the segmentation procedure as an image (see figure 8.2(b)).

HSV image data: A copy of the region from the HSV-image used during segmentation, corresponding to the binary mask (see figure 8.2(c)).

Position: The position of the BLOB's upper-left corner in image plane.

Size: The spatial extent (a, b) of the BLOB in the image plane.

Reference features: The original feature vector obtained when the object was initialized.

Predicted features: The predicted next state feature vector.

Current features: The current state of the feature vector, containing the latest measurements.

Old features: The previous set of feature measurements.

Frame last recognized: A integer (in a global sense) specifying at what frame the object was last identified.

Time last recognized: The system time at which the object was last identified.

(a) (b) (c)

Figure 8.2: Figure (a) A rectangular window containing a BLOB. Figure (b) The segmented BLOB. Figure (c) Image data created by combining figure (a) with the mask in figure (b) (sometimes referred to as "cookie-cutting").

Moreover, a global frame counter exists, which is incremented for each iteration/frame.

8.1.2 I/O Specification

Input: Segmented objects (BLOBs) placed on the table determined by the segmentation component, and the current orientation of the camera in Euler angles obtained from the stabilization component.
Output: BLOBs classified as placeholder items from the system's item database.

8.2 Domain

Explicit object initialization is currently implemented by predefining a 2D rectangle in the image plane where segmented objects can be added to the item database. The first BLOB found inside this "viewfinder" is then added to the database, after features has been extracted (see section 8.4) and stored in the **Reference features** variable of the new placeholder structure. Obviously, it is important that the object is properly positioned inside the viewfinder when initialized to obtain good tracking results. Otherwise the risk of obtaining undescriptive features becomes inherent and erroneous tracking is unavoidable.

Initially the contents of **Reference features** is replicated into **Predicted features**, **Current features** and **Old features**. **Frame last recognized** and **Time last recognized** is set to the current frame and system time respectively. The remaining variables are all derived of the BLOB found, and their initial values should therefore be self explanatory.

For every segmented object found by the segmentation component, features are extracted in order to allow feature matching. A temporary database with the segmented objects are created and destroyed for each frame, and the extracted features are stored in the **Reference features** vector of the structure corresponding to a temporary object.

The existence of object occlusions in the image plane are important. Two objects can be said to be either *separable*, *partially* occluding or *fully occluding*. Figure 8.3 show the three cases of separability and occlusion.

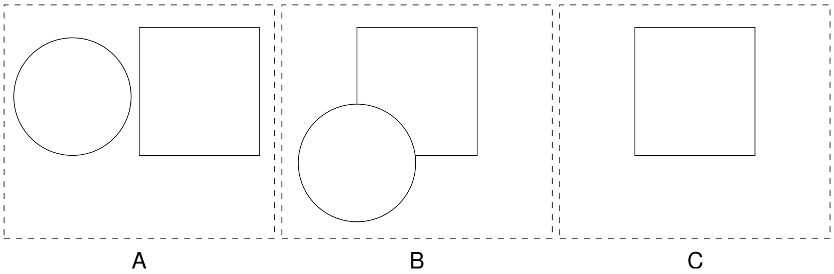


Figure 8.3: Three examples with: (A) separate BLOBs, (B) partial occlusion, and (C) complete occlusion (the cicular object is occluded by the rectangular).

The user is allowed to move and rotate placeholder items on the table at runtime. However, the presence of the user's hands and arms inside the workspace/images and the resulting complete or partial occlusions is a problem which is not dealt with directly in this design. Hands/arms are treated like any other object.

8.3 Overview

The tracking component has been conveniently subdivided into a number of subprocesses. One approach for solving this problem, used for this project, is initially to match all incoming segmented BLOBs with the objects from the item database using distance matching on an extracted feature vector (i.e., measurement). The feature vector represents the state of the object, and matching is done between the extracted and the predicted next state. If objects from the item database are not matched, a generic matching method is applied for the segmented objects that are not matched. Finally the item database is updated with the latest measurements.

Figure 8.4 depicts this subdivision, which is partly derived from (Birk et al., 1996).

The subprocesses are described in the following order:

Feature extraction on the segmented BLOBs is described in section 8.4.

Prediction of the present state of individual placeholder items based on previous states (memory) is described in section 8.5.

Feature matching between the extracted features and predicted features for objects in the item database is described in section 8.6.

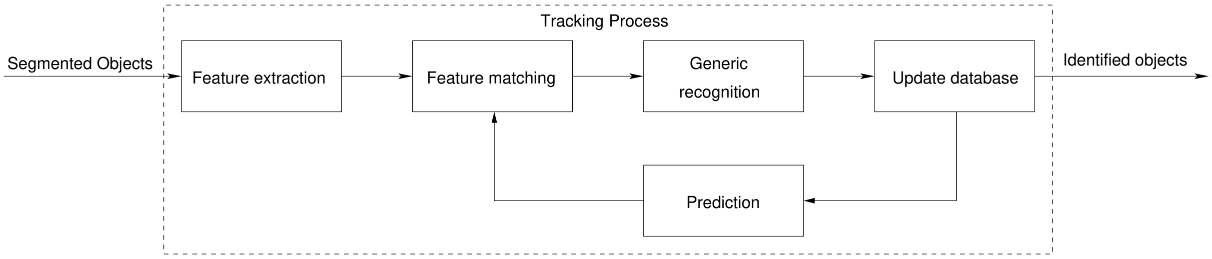


Figure 8.4: Tracking subprocesses.

Generic recognition for handling partial occlusion problems on BLOBs, which were not cleanly matched is described in section 8.7.

Update database as the final procedure before the tracking are ready for the next estimation as described in 8.8.

The approach just described is referred to as a "High level token"-matching system, where a token is a set of features representing the state of an object.

Some notes about the developed real-time prototype with respect to some of the subprocesses are described in this chapter:

1. The output of the stabilization is currently unused in the feature extraction stage. Image plane coordinates (x, y) are used instead as compensation.
2. The generic recognition ("subregion matching") technique described has been designed and tested separately. A thorough integration of that block in the tracking component is thus not described.

8.4 Feature Extraction

In order to separate objects in feature space, features containing characteristics of each BLOB must be extracted. It was decided to extract both short-term, "variant" features (e.g., spatial position, size, etc.) suited for tracking and "invariant" features (e.g., invariant moments, histograms, etc.) that by definition are insensitive to translation, scaling and rotation. Invariant features are conversely useful for recognition, when the variant features are unreliable.

8.4.1 Specific Design and Implementation Issues

Table 8.1 shows a list of the 61 features, which are extracted for each BLOB. Some remarks about the features:

- The optical center of mass (x', y') is found using zeroth and first order spatial moments (Gonzalez and Woods, 1993, p. 514).
- The invariant "Hu moments", which are based on the second and third central moments (Gonzalez and Woods, 1993, p. 516), are calculated based on the pixel data from the hue channel in order to capture color features.
- The probabilistic distributions are sampled using 16 bins.

Although some of the features in table 8.1 by definition are invariant, some fluctuation should be expected when used in practice. Scaling, for instance, have a significantly impact on the amount of discrete data available for the calculations. Feature calculations for objects subtending a small area in the image plane is thus less precise.

Index	Persistence	Description
0-1	Variant	Center position in world-space (P_{pitch}, P_{yaw}) from equation 6.8
2-3	Variant	Spatial size of the BLOB
4-5	Invariant	Optical center of mass (centroid) of the BLOB
6-12	Invariant	Hu moments
13-28	Invariant	Probabilistic distribution (histogram) of the hue channel
29-44	Invariant	Probabilistic distribution (histogram) of the saturation channel
45-60	Invariant	Probabilistic distribution (histogram) of the value channel

Table 8.1: The 61 variant and invariant features extracted.

8.5 Prediction

Prediction (or estimation) is the problem of predicting the next state of an object from its previous feature vector measurements. Prior to feature matching, the state objects from the present item database must be predicted to ensure correct classification.

Appendix F describes various predictors. Some are based on purely data driven regression/extrapolation and others, implementing more advanced mathematical methods, include a system state model.

8.5.1 Specific Design and Implementation Issues

Consider the position of a moving object. If the movement is sufficiently smooth, the system should have no problem determining the next state by simple linear extrapolation. The change of object appearance is another case where prediction is useful, e.g., for a rotating object that exhibits a specific textural characteristic depending on orientation. This will influence on even the invariant moments.

The prediction method applied is based on just the two previous sets of measurements, and is described in appendix F as the "minimal memory, linear extrapolation" (a special case of polynomial fitting and extrapolation). This method is very fast, as it does not require any matrix calculations, but is more sensitive to noise than e.g. Kalman filtering.

For each object identified in the previous frame, each element k in the **Predicted features** vector, $\overline{p}$, of the respective object is calculated as:

$$\overline{p}_k = \frac{t_p - t_1}{t_2 - t_1} * (\overline{c}_k - \overline{o}_k) + \overline{o}_k \tag{8.1}$$

where the remaining variables specifies:

- t_p, present system time.
- t_1, system time at the previous estimation.
- t_2, system time at the last but one estimation.
- $\overline{c}_k$, value of feature k at the previous estimation from the **Current features** vector.
- $\overline{o}_k$, value of feature k at the last but one estimation from the **Old features** vector.

In the prototype, variant and invariant features are all predicted indentically using equation 8.1.

8.6 Feature Matching

The purpose of the feature matching block is to find the probability that an extracted object match a predicted object from the item database, i.e., find a match between two feature vectors that expresses the similarity between these two.

The similarity between two objects can be calculated as the distance between them in feature space. Appendix E describes various distance metrics.

8.6.1 Specific Design and Implementation Issues

The distance, c_{ij}, between a predicted and extracted object in feature space is placed in a matrix, where lower values corresponds to better matches. The matrix is called the "C-matrix" (i.e., **C**onfidence matrix), when this matrix has been found the objects can be classified.

Feature Distance Determination

The C-matrix is defined as:

$$\mathbf{C} = \begin{array}{c c} & \begin{array}{ccc} \bar{b}_1 & \dots & \bar{b}_n \end{array} \\ \begin{array}{c} \bar{a}_1 \\ \cdot \\ \cdot \\ \cdot \\ \bar{a}_m \end{array} & \begin{bmatrix} \mathbf{C}_{11} & \dots & \mathbf{C}_{1n} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \mathbf{C}_{m1} & \dots & \mathbf{C}_{mn} \end{bmatrix} \end{array} \tag{8.2}$$

where the predicted feature vectors from the item database are named $\bar{a}_1$ to $\bar{a}_m$ and the extracted feature vectors $\bar{b}_1$ to $\bar{b}_n$.

Simple Euclidian distance as a metric for finding, c_{ij}, is not a feasible solution since it would optimally require all features to have the same range (e.g., $[0-1]$), otherwise the risk of some feature "overpowering" others numerically becomes inherent. For some features (e.g., the Hu moments) it is difficult to predict their value range, which makes normalization hard. A better solution for a distance measure is to use the Mahalanobis distance, which takes feature-covariance properly into account, but is actually nothing more than a weighted Euclidean distance. In its basic form Mahalanobis distance is calculated as:

$$c_{ij} = \sqrt{(\bar{a}_i - \bar{b}_j)\mathbf{V}^{-1}(\bar{a}_i - \bar{b}_j)^T} \tag{8.3}$$

where $\mathbf{V}^{-1}$ is the inverse of the feature-covariance matrix. However, the inverse covariance is processed as described in section 8.6.1 before using equation 8.3.

The calculation of the **V**-matrix is carried out each frame by calculating the covariance of a concatenated list of feature vectors from three sources:

1. The extracted feature vectors from the current frame (**Reference features**) from matrix **B**.
2. A history matrix containing the previous o extracted feature vectors as a bounded buffer that allows elements to be overwritten.
3. All objects from the item database. However, feature values are processed by equation 8.4 which finds a derived feature value, f', before insertion in the concatenated list of feature vectors. Moreover this modified feature value replaces the element in the a_i vector and is thus used when the Mahalanobis distance is calculated.

$$f' = \begin{cases} (k\ o) - (l\ p) & \text{for invariant features,} \\ p & \text{for variant features.} \end{cases} \tag{8.4}$$

where

- o is the feature value of interest from **Old features**, and
- p is the corresponding value from **Predicted features**.
- $k = (1 - O_{factor})\ s$
- $l = 1 - k$

O_{factor} is a factor between 0 and 1 which determines the fraction of the originally acquired data that is used for feature matching.
s is calculated from the following equation:

$$s = \begin{cases} 0 & \text{if } (t_p - t_l) > T_{limit}, \\ 1 - \frac{t_p - t_l}{T_{limit}} & \text{otherwise.} \end{cases} \tag{8.5}$$

where t_p is the current system time, t_l is the **Time last recognized** for the object of interest, and T_{limit} is the prediction timeout period, where the predicted features can no longer be trusted.
The essence of the above equations are that the currently measured value of the invariant features are "trusted" more when little time has passed since it was last identified. Otherwise the original reference values are used.
T_{limit} is currently defined as being 500 ms.

This yields a total of $m+n+o$ feature vectors, which together are then used to obtain an estimate of the covariances.

A good choice for o was emprically found to be 250 for the current prototype.

The inverse of $\mathbf{V}$, $\mathbf{V}^{-1}$, is determined using the singular value decomposition (SVD), as $\mathbf{V}$ is usually very ill-conditioned. The SVD can then be used for constructing a low-rank approximated solution to the pseudo inverse matrix. Any singular values less than a tolerance, S_{min}, are treated as zero.

The overall performance of the feature matching component depends much on the value of S_{min}. If the tolerance is set too low, noise in the measurements gets transfered into the covariance inverse, as well. If, conversely, the tolerance is too high the system is unable to capture subtle trends in the data that would be well-suited for discrimination.

S_{min} is currently set to 0.1. This value have experimentally been found to yield a more stable system then either 0.01 or 1.

Covariance Weighting

As it seems reasonable that some features are more reliable than others a feature weight scheme has been incorporated. A Hu moment should for instance, theoretically be a better descriptor than a single histogram bin.

Elements in the inverted covariance matrix, V^{-1}, are multiplied by factors related to their indices. Two multiplications are necessary as the covariance matrix is symmetrical across the diagonal.

Table 8.2 shows the currently used feature weights.

Before the C-matrix is determined, a special inverse covariance matrix is found for each object in the item database, based on the weighted matrix just described.

Variant features are scaled by the s factor from equation 8.5 (again taking symmetry into account). This reduces the significance of variant features as the time since the last match increases, thereby emphasizing the meaning of the invariant features.

Feature	Weight
Center position	2
Spatial size	2
Optical center of mass	3
Hu moments	4
Hue channel histogram	1
Saturation channel histogram	1
Value channel histogram	1

Table 8.2: Features weights used for scaling covariance elements.

Object Classification

When the C-matrix has been calculated, the objects can finally be classified if their confidence c_{ij} is less than a static threshold, C_{match}.

The matches are processed from the best to the worst, by applying the following algorithm step-by-step:

1. Traverse the C-matrix and find the lowest c_{ij} value (the best match), m.
2. End if m is greater than C_{match}.
3. Mark object i from the item database as being matched as the extracted object j.
4. Set all entries of the C-matrix in row i and column j to ∞. This effectively marks both the predicted and extracted as being used.
5. Goto 1.

$C_{match} = 1.2$ is used in the present implementation, as false matches typically results in values greater than 3.0 and positive matches are less than 1.0.

8.7 Generic Recognition

When feature matching fails, a supplementary, generic recognition method for handling occlusions is necessary for object recognition. An approach proposed in this section, called "subregion matching", is based on histogram matching of small regions using the HSV color model.

However, it must be noted that the technique is not implemented in the current prototype, and can not be described in complete detail, as there are still some issues left to consider.

8.7.1 Specific Design and Implementation Issues

Under the assumption that all movement in the image plane is primarily due to translations and rotations, and objects are somewhat homogeneous, small histograms can be used to solve the occlusion problem. This is a not completely true as some scaling does take place, but the method does seem to work quite well if the homogeneity criterion is fulfilled, and the appearance of the objects are not too similar.

For all extracted objects not found in the current image the algorithm starts by finding pixel subregions (or windows) of size $x \times y$ in the BLOB that contains more than z pixels. To increase robustness the histograms are calculated with a small overlap between subregions ("oversizing").

$z = 20$ pixels is used in the tests performed below, and the amount of overlap was set to 30%.

Histograms for every oversized subregion are then sampled and stored, using w bins independently for the hue-, saturation- and value-channel. Normalization to probability distributions are used for generality purposes.

The usefulness of the color information contained in hue-channel depends on how saturated the colors are. If colors are very desaturated and near greyscale, the use of the hue-channel for recognition becomes error-prone. The goodness, α, of the color information is estimated using the following sum:

$$\alpha = \sum_{i=0}^{w-1} h(i) f(i) \tag{8.6}$$

where

- $h(i)$ is bin i of the satuaration probability distribution.
- $f(i)$ can be any monotonously increasing function in the interval from 0 to 1. Currently $f(i) = \sqrt{\frac{i}{w-1}}$ is used as shown in figure 8.5.

The goodness of the color information of all subregions are then computed and stored for use in the matching step.

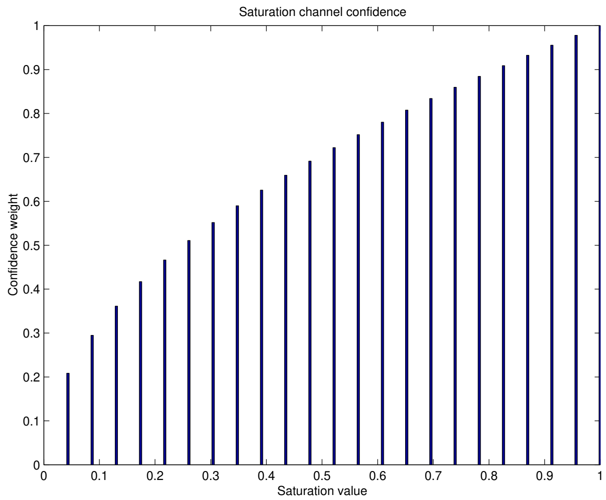


Figure 8.5: Saturation weighting defined by $f(x) = \sqrt{x}$, $x \in [0;\ 1]$.

It is assumed that histograms have been computed analogue for objects in the item database. This could appropriately be done when they were initialized.

All histograms from the item database and extracted objects can then be compared using the Euclidian distance metric as described below. Matching is then carried out using an approach identical to the one described in section 8.6.1.

The goodness of the color data from two subregions to be compared, with goodness α_1 and α_2 respectively, are then determined by calculating:

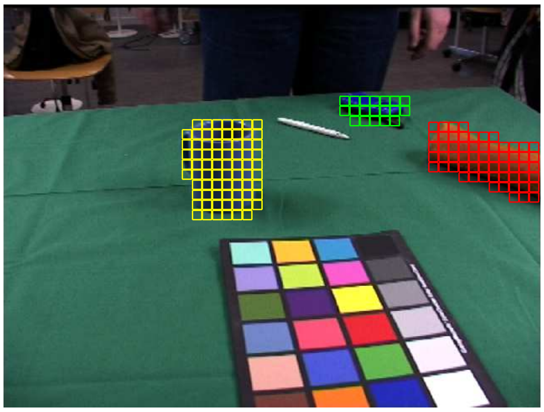
$$\begin{aligned} \beta &= min(\alpha_1, \alpha_2) \\ \gamma &= 1 - \beta \end{aligned} \tag{8.7}$$

The addition of the sum of squared difference (SSD) of the hue-channel scaled by β and the value-channel scaled by γ are then calculated, and signifies the similarity between the subregions.

8.7.2 Examples

Figure 8.6, 8.7, and 8.8 shows an applied example of the subregion matching method.

Original subregions.



Matched subregions (black regions are unknown).

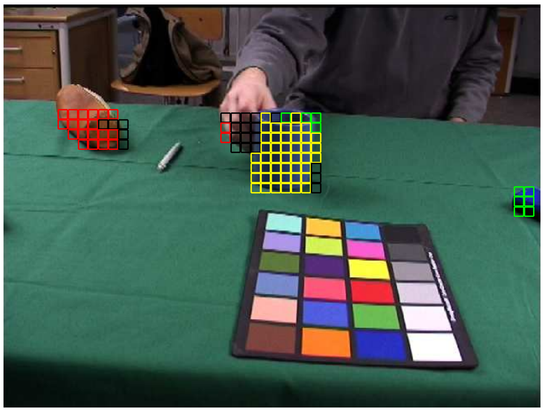


Figure 8.6: Subregion method using 16 bin histograms and $12 \times 12/16 \times 16$ pixel subregions.

In figure 8.9, 8.10, and 8.11 another example of the subregion matching method can be seen.

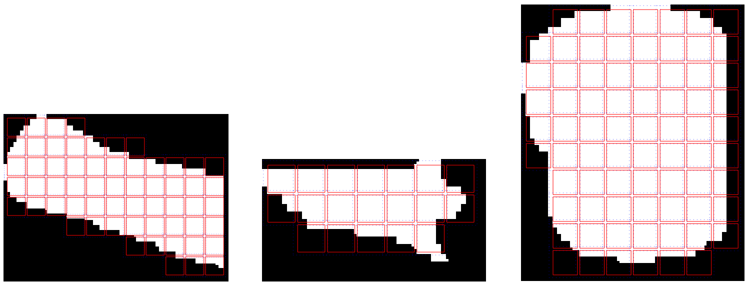
8.7.3 Concluding Remarks

The subregion matching method seems to have a potential to (at least partly) solve the problem of occlusion if the objects have some degree of similarity, although the examples show that the method is unable to match all regions correctly. Much better performance was achieved if the similarity between the objects from the item database and the extracted items was greater (for instance, between two consecutive frames). By maintaining a dynamic history buffer of histograms from a placeholder item, better results could be expected.

Tests showed that the choice of histogram bins, the size of the windows and the amount of overlap significantly impacts on the performance of the algorithm.

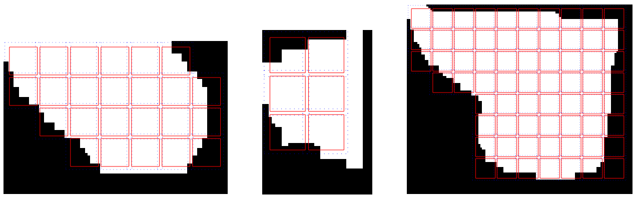
This method handles the problem with desaturated colors, and have great real-time potential. If the distances in color space are combined with knowledge of the position of the object in the previous frame, more robust matching is possible.

Clustering methods must be applied to reject "outliers" among the region matches. By proper clustering of subregions, it is possible to obtain regions containing the placeholder item.



(a) (b) (c)

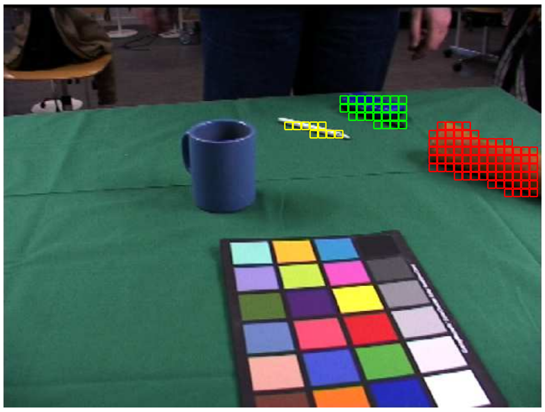
Figure 8.7: Figure (a-c) Reference BLOB masks from upper figure 8.6 ($12 \times 12/16 \times 16$ pixel subregions overlay).



(a) (b) (c)

Figure 8.8: Figure (a-c) Extracted BLOB masks from upper figure 8.6 ($12 \times 12/16 \times 16$ pixel subregions overlay).

Original subregions.



Matched subregions (black regions are unknown).

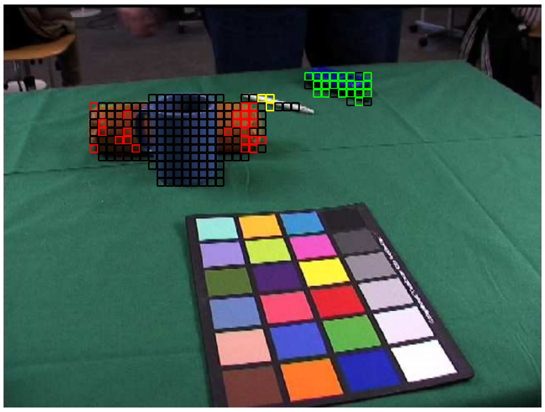
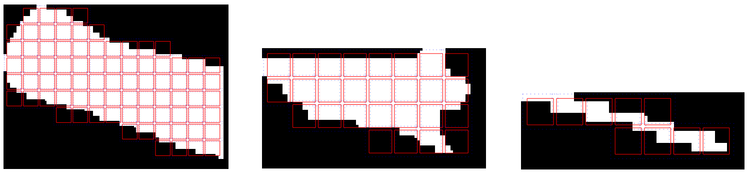
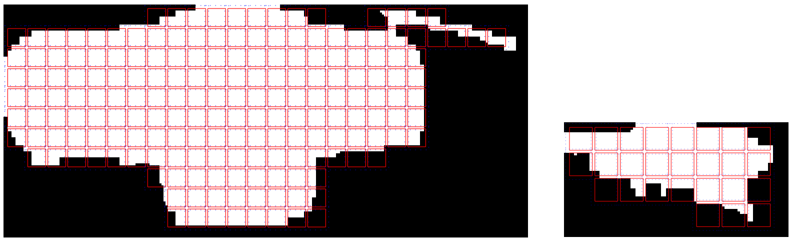


Figure 8.9: Subregion method using 16 bin histograms and $10 \times 10/13 \times 13$ pixel subregions.



(a) (b) (c)

Figure 8.10: Figure (a-c) Reference BLOB masks from lower figure 8.9 ($10 \times 10/13 \times 13$ pixel subregions overlay).



(a) (b)

Figure 8.11: Figure (a-c) Extracted BLOB masks from lower figure 8.6 ($10 \times 10/13 \times 13$ pixel subregions overlay).

This method should also be more thoroughly examined with respect to handling the scaling problem properly.

8.8 Update Database

As the last step in the tracking procedure, the item database must be updated to reflect the present state, so that e.g. predictions can be made for the subsequent frame.

8.8.1 Specific Design and Implementation Issues

Updating the item database is currently achieved by for each object identified at the current instance:

- Copying the contents of **Current features** to **Old features**. This is necessary as the prediction process requires the last but one features to exist.
- Copying the **Reference features** for the matching BLOB in the current frame to **Current features**, effectively updating the current state with the newest measurements.
- Setting **Frame last recognized** to the value of the global frame counter.
- Assigning **Time last recognized** the current system time.

8.9 Summary

To summerize, the tracking component was subdivided into a number of subprocesses. The feature extraction block extract feature elements from the segmented BLOBs using both variant and invariant features (primarily color-based). These features are then used at the feature matching stage to identify objects from the item database. As the matching method fails for partially occluding objects, a generic recognition approach called "subregion matching", is proposed as the supplementary recognition method. Finally the object database is updated with the latest measurements for the found objects. This is primarily done to ensure that the prediction of the next object state, represented by a dedicated feature vector, works properly.

The tracking component was to some extent simplified, and the appliance of more robust methods (e.g., Kalman filtering for prediction) will therefore improve tracking accuracy. However, depending on the choice of method this might as well increase execution time of the tracking procedure.

A test of the subprocesses has been carried out as part of the system test in chapter 9.

Figure 8.12 shows a block diagram of the proposed solution.

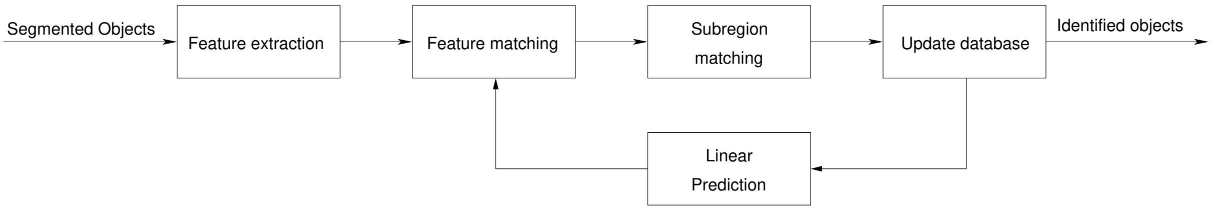


Figure 8.12: Block diagram for the tracking component.

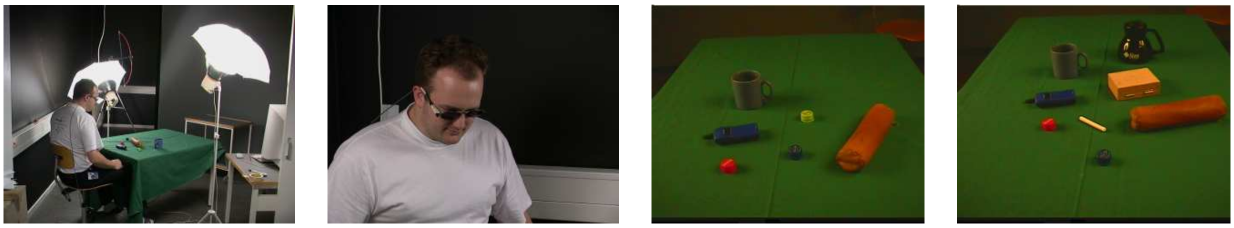
CHAPTER 9

System Test

The purpose of the system test is to check the system functions as described in the system specification in chapter 3. The complete system as described in chapter 3 is not fully implemented in C++ (or C++ based libraries like OpenCV, and SDL), and therefore not tested for all the functionalities. Section 9.2 describes what parts of the complete system that are implemented.

9.1 Description of the Test Environment

The system is conducted in a test environment that is identical to a typical use case. The test environment is shown in figure 9.1 (a).



(a) (b) (c) (d)

Figure 9.1: The system test environment. Figure (a) shows the entire environment. Figure (b) shows the head-mounted camera. Figure (c) shows the objects used in the test for correct recognition. Figure (d) shows the objects used in the test for time consumption.

Illumination is generated by 2 500 Watt lamps of the type *Photax 3200K head White interfit*. They are shown in figure 9.1 (a).

The table covered with a green cloth . The green cloth is shown in figure 9.1 (a).

The objects used during the different test setups is divided into two sets. The first set is used in the “Test for correct recognition”, see section 9.3. The second set is used in the "Test for time consumption", see section 9.4.

The first set is shown in figure 9.1 (c). The objects are: One blue cellular phone, one blue cup, one pencil case, one blue stopper, one green stopper, and one red stopper.

The second set is shown in figure 9.1 (d). The objects are: One blue cellular phone, one blue cup, one black cup, one box of wood, one pencil case, one white chalk, one blue stopper, and one red stopper.

It is not all objects that has a unique color, e.g. the blue cellular phone, the blue cup, and the blue stopper. The choice is made to measure the robustness of the tracker toward similar colored objects. But the blue objects has a unique size, and shape.

Camera and Camera control unit . The camera is from the company Elmo, and is of the type *ELMO QP49*. The optical used is of the type *ELMO QT254, with focal length 4 mm.* The camera is head-mounted as shown in figure 9.1 (b).

The Camera control unit is from the company Elmo, and is of the type *ELMO CC491*. The camera control unit has following switch setup:

- AGC = OFF
- W.B. = MANU

The camera control unit has following software setup:

- FILE = A
- SHUTTER = AUTO
- REDESTAL = 00
- SYNC = INT
- AREA = LINK1
- WB OFFSET = 00

Framegrabber is from the company ARV00, and is of the type *PICASSO PCI-2SQ*. The framegrabber has the following setup:

- Maximum fps = 12.5
- Mode = Progressive scan
- Resolution = 320x256
- Color mode = RGB 24bit

Computer is a Pentium III 1GHz with 256 MB ram running RedHat 7.0.

9.2 Description of the Implemented System

This section describes in what degree the different components are implemented in C++. As previous mentioned it is only the parts implemented in C++ that is tested during the system test, and the other parts that is implemented in Matlab is tested in their respective design chapters. Figure 9.2 shows the design overview of the system.

The following list contains the components of the system. A more detailed description of the degree of implementation in C++ follows.

- Input
- Segmentation
- Tracking
- Output
- Stabilization

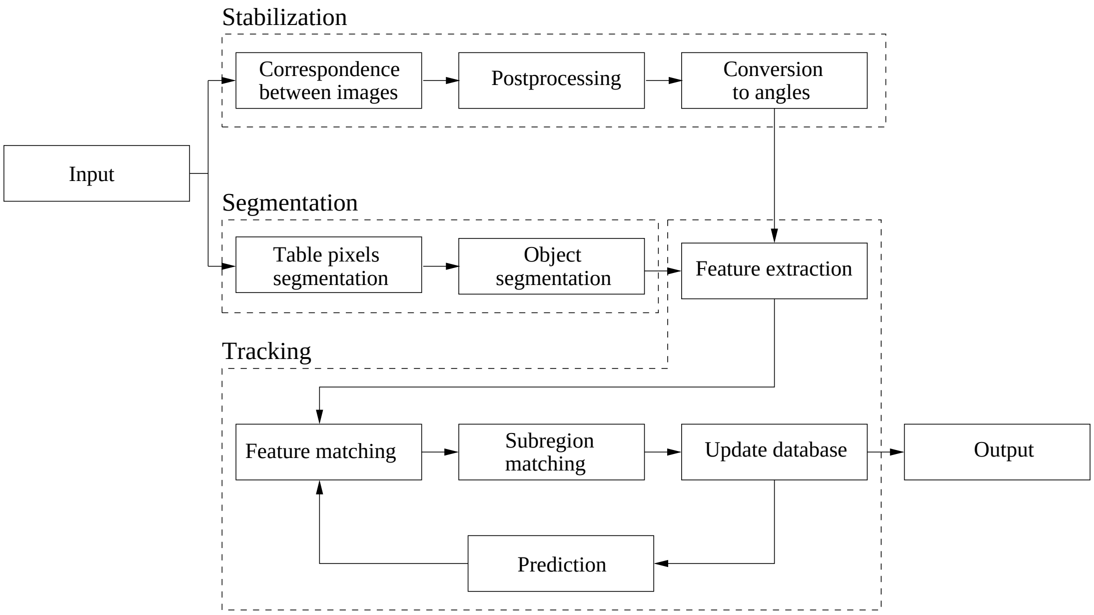


Figure 9.2: The design overview of the system. Not all of components are implemented in C++.

9.2.1 Input

The input component is fully implemented in C++, and provides the rest of the components with acquired and preprocessed images. The component is implemented in two versions depending on the source of acquisition.

The offline version acquires the images from storage media and was used during design. The online version acquires the images from the framegrabber and is used for the system test, and during normal use. There has be added some functionality to this component in order easy the interaction and test of the system. The added functionality is listed below:

Keyboard is used for interacting with the system, e.g. when an object is initialized. Note: when an object is initialized and added to the database of objects, it is not possible to remove it while running the system.

9.2.2 Segmentation

The segmentation component is fully implemented in C++, and provides the tracker component with regions (BLOBs) inside the table for possible tracking.

9.2.3 Tracking

The tracker component is not fully implemented in C++. The part of the tracker component that is implemented in C++ provides recognition and tracking of multiple non-occluded objects. The other part that handles the recognition and tracking of occluded objects is not implemented in C++, but implemented and tested in Matlab. In chapter 8 is a description of the part that handles occluded objects.

9.2.4 Output

The output component is fully implemented in C++. There has be added some functionality to this component in order easy the interaction and test of the system, see figure 9.3 (a). The added functionality is listed below:

Initialization window is an area in the center of the image with the size of 1/16 of the total image. The area is marked with a bounding box, and used for initialization of the table and objects.

Object boundary box is an rectangular box surrounding the object. The color of the box changes according to the degree of correctness in the recognition of an object compared to the object database. The color of the boundary box gradually changes from red to green depending on the degree of correctness, where green is the best match. In case of an unknown object inside the table the boundary box is colored blue.

Object identifier is a number attached to the boundary box indicating the number the object has in the item database.

9.2.5 Stabilization

The stabilization component is not implemented in C++, and therefore not tested during this system test. In chapter 6 is a description of the degree of implementation in Matlab, as well as a test of the implemented component.

9.3 Test for Correct Recognition

The implemented system is test for correct recognition. In order to do this a number of tests are conducted simulating different aspects of normal use. Henceforth is a description of the tests conducted to test the system ability recognize correctly, with certain limits.

9.3.1 Description of the Performed Tests

Test setups

List of the four test setups describing the numbers of objects, and how they are moving in the scene.

1. The table contains one object, where the object is movable. The setup is illustrated in figure 9.3 (a).
2. The table contains multiple objects, where only one single object is movable and the rest is considered static. The setup is illustrated in figure 9.3 (b).
3. The table contains two objects, where both objects are movable along independent paths. The setup is illustrated in figure 9.3 (c).
4. The table contains multiple objects, where two objects are movable along independent paths and the rest is considered static. The setup is illustrated in figure 9.3 (d).

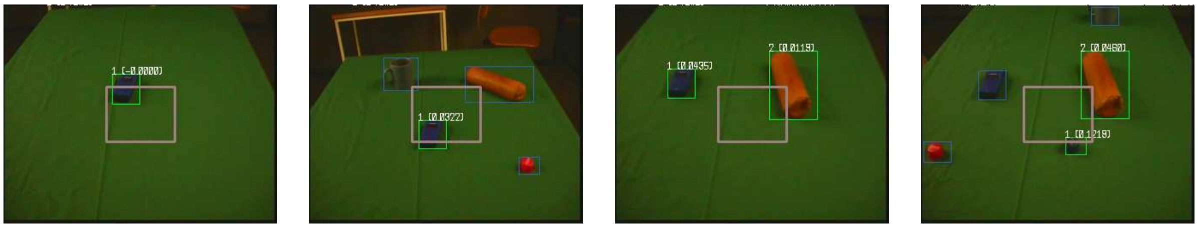
9.3.2 Tests Performed in the Different Test Setups

Measurement for the performed tests.

The six tests performed are measured differently, and therefore divided into two types of measurement.

Test 1 is measured visual. The result is true if the table is colored in cyan except for the objects inside the table, they are colored black. The result is false if not both criteria is fulfilled.

Test 2-6 is measured numerical based on a visual registration. Each test is measured on two things:



(a) (b) (c) (d)

Figure 9.3: The four test setups for the system test. Figure (a) illustrates the setup with one movable object. Figure (b) illustrates the setup with multiple object, the blue cellular phone is movable. Figure (c) illustrates the setup with two movable objects along independent paths. Figure (d) illustrates the setup with multiple objects, the blue cellular phone and the pencil case are movable along independent paths.

- The number of lost tracking.
- The number of misidentified objects. When the identification number attached to the boundary box of the object does not match the identification number the object has in the item database (the number in the initialization sequence).

List of performed tests

There are conducted a series of six tests for each test setup. Each of the tests is conducted for at time period of 1 minute. The reason for this limited time period is due to a problem in the GNU compiler delivered with RedHat 7.0. The compiler introduces a memory leak that resolves in a total memory usage within time period of 5 minutes. The memory leak is not present in other versions of the GNU compiler.

The six tests is described below.

1. **Initialization and recognition of the table with a separation of objects.**

   Purpose : To test the systems stability towards recognition of the table over time. It is important that the system recognizes the objects placed on the table and register them as objects (not as a part of the table). The test is conducted for 1 minute.

   Environment setup : The camera is head-mounted and the field of view of the camera is sweeped randomly over the table during the test. The velocity of the movement is delimited by the constraints described in section 2.6 *Movement Estimation*. The settings of the camera and framegrabber is described in section 9.1.

   The table is covered by a piece of green cloth, and is initialized before conducting the test.

   The numbers of objects depends on the test setup. The objects placed on the table is static at all time during the test.

   Expected result : A recognition of the table with a separation of the objects inside the table. The table is colored cyan, and the objects colored black.

2. **Stable recognition and tracking of static objects over time.**

   Purpose : To test the systems stability toward recognition of static objects over time. It is important that the system recognizes all objects inside the table, but only tracks the initialized objects. The test is conducted for 1 minute.

   Environment setup : The camera is head-mounted, and the field of view of the camera is sweeped randomly over the table with the objects inside the field of view at all time during the test. The velocity of the movement is delimited by the constraints described in section 2.6 *Movement Estimation*. The settings of the camera and framegrabber is described in section 9.1.

The table is covered by a piece of green cloth, and is initialized before conducting the test.

The number of objects depend on the test setup. The objects placed on the table is static at all time during the test. The objects of interest is initialized before conducting the test.

Expected result : A recognition of the table with separation of the objects inside the table. The boundary box of the objects of interest are marked with a color scaling from red to green. The rest of the objects are marked blue for unknown objects. In addition the objects of interest is enumerated with a number indicating the order it was initialized, i.e. the number in the item database.

3. **Stable recognition and tracking of moving objects over time.**

Purpose : To test the systems stability towards recognition of moving objects over time. It is important that the system recognizes all objects inside the table, but only tracks the initialized objects. The test is conducted for 1 minute.

Environment setup : The camera is head-mounted, and the field of view of the camera is sweeped randomly over the table with all objects inside the field of view at all time during the test. The velocity of the movement is delimited by the constraints described in section 2.6 *Movement Estimation*. The settings of the camera and framegrabber is described in section 9.1.

The table is covered by a piece of green cloth, and is initialized before conducting the test.

The number of object depends on the test setup. The objects of interest are moved around the table while the rest are static at all time during the test. The objects of interest are not allowed to move outside the field of view of the camera at any time during the test. The objects of interest is initialized before conducting the test.

Expected result : A recognition of the table with separation of the objects inside the table. The boundary box of the objects of interest are marked with a color scaling from red to green. The rest of the objects are marked blue for unknown objects. In addition the objects of interest is enumerated with a number indicating the order it was initialized, i.e. the number in the item database.

4. **Stable recognition and tracking of objects during scaling of the objects (zoom).**

Purpose : To test the systems stability towards recognition of scaled objects over time. It is important that the system locates all the object inside the table, but only recognizes and tracks the initialized objects. The test is conducted for 1 minute.

Environment setup : The camera is head-mounted. The camera is moved toward and away from the objects of interest during the test to simulate a scaling of the objects, e.g an object near the camera is bigger than the same object far away. The velocity of the movement is delimited by the constraints described in section 2.6 *Movement Estimation*. The settings of the camera and framegrabber is described in section 9.1.

The table is covered by a piece of green cloth, and is initialized before conducting the test.

The number of object depends on the test setup. The objects of interest as well as the other objects are placed on the table and are static at all time during the test. The objects of interest is initialized before conducting the test.

Expected result : A recognition of the table with separation of the objects inside the table. The boundary box of the objects of interest are marked with a color scaling from red to green. The rest of the objects are marked blue for unknown objects. In addition the objects of interest is enumerated with a number indicating the order it was initialized, i.e. the number in the item database.

5. **Stable recognition and tracking of objects after reentering the field of view.**

Purpose : To test the systems stability towards recognition of objects after reentering the field of view of the camera over time. It is important that the system locates all the objects inside the table, but only recognizes and tracks the initialized objects. The test is conducted for 1 minute.

Environment setup : The camera is head-mounted. The objects of interest is moved outside and then inside the field of view of the camera again. This is done repeatedly during the test. The velocity of the movement is delimited by the constraints described in section 2.6 *Movement Estimation*. The settings of the camera and framegrabber are described in section 9.1.

The table is covered by a piece of green cloth, and is initialized before conducting the test.

The number of object depends on the test setup. The objects of interest are moved in and out of the field of view of the camera, while the other objects are static at all time during the test. The objects of interest is initialized before conducting the test.

Expected result : A recognition of the table with separation of the objects inside the table. The boundary box of the objects of interest are marked with a color scaling from red to green. The rest of the objects are marked blue for unknown objects. In addition the objects of interest is enumerated with a number indicating the order it was initialized, i.e. the number in the item database. The expected results of the objects of interest are only valid, while the objects are inside the field of view of the camera.

6. **Stable recognition and tracking of objects during rotation (roll) of the camera.**

Purpose : To test the systems stability towards recognition of objects during roll of the camera over time. It is important that the system locates all the objects inside the table, but only recognizes and tracks the initialized objects. The test is conducted for 1 minute.

Environment setup : The camera is head-mounted, and is allow only roll during the test. The camera rolls from one side to the other, repeatedly. The velocity of the movement is delimited by the constraints described in section 2.6 *Movement Estimation*. The settings of the camera and framegrabber is described in section 9.1.

The table is covered by a piece of green cloth, and is initialized before conducting the test.

The number of object depends on the test setup. The objects of interest as well as the other objects are placed on the table and are static at all time during the test. The objects of interest is initialized before conducting the test.

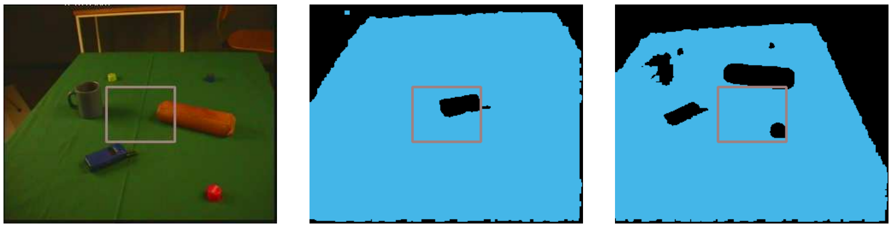
Expected result : A recognition of the table with separation of the objects inside the table. The boundary box of the objects of interest are marked with a color scaling from red to green. The rest of the objects are marked blue for unknown objects. In addition the objects of interest is enumerated with a number indicating the order it was initialized, i.e. the number in the item database.

9.3.3 Results of the Tests Performed in the Different Test Setups

Test 1: Initialization and Recognition of the Table with Objects Marked as Unknown

The purpose of this test was to test the systems stability towards recognition of the table over time. It is important that the system recognizes the objects placed on the table and registers them as objects (not part of the table). The test was conducted for 1 minute, and inspected visual.

Figure 9.4 shows some examples of correct recognition of the table. The objects inside the table is separated from the table as required to pass the test.



(a) (b) (c)

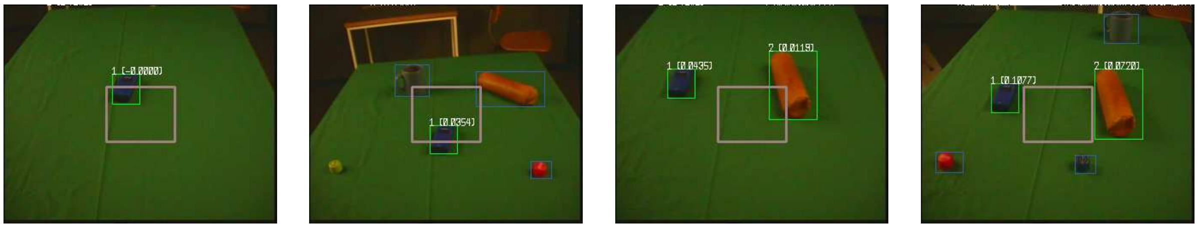
Figure 9.4: Results of test 1 in the test of correct recognition, where figure (a) shows a reference to figure (a) & figure (b). Figure (b) shows the test of one object inside the table. Figure (c) shows the test of multiple objects inside the table.

The evaluation of the test is based on the result of the test and the requirements to it. As seen in figure 9.4 the table is recognized perfect, and the objects inside the table are separated from the mask of the table. The blue cup in the back of the figure 9.4 (c) is the only one that is not perfectly separated from the table, this is because of the narrow handle and the low resolution (320x256).

Test 2: Stable Recognition and Tracking of Static Objects Over Time.

The purpose of this test was to test the systems stability towards recognition of static objects over time. It is important that the system recognizes all objects inside the table, but only tracks the initialized objects. The test was conducted for 1 minute, and based on a visual registration.

Figure 9.5 shows some examples of correct recognition of static objects. The objects of interest was tracked as required to pass the test, and table 9.1 shows in what degree.



(a) (b) (c) (d)

Figure 9.5: Results of test 2 in the test of correct recognition, where figure (a) shows test setup 1. Figure (b) shows test setup 2, figure (c) shows test setup 3, and figure (d) shows test setup 4

The evaluation of the test is based on the result of the test and the requirements to it. As seen in table 9.1 there is a problem in test setup 4, where the system looses track of the blue cellular phone. This problem is shown in figure9.6 (b) where the blue stopper is recognized as the blue cellular phone.

Test setup	Number of lost tracking	Number of mis-identifications
1	0	0
2	0	0

	Number of lost tracking		Number of mis-identifications	
Test setup	Cellular phone	Pencil case	Cellular phone	Pencil case
3	0	0	0	0
4	> 20	0	0	0

Table 9.1: Results of test 2 in the test of correct recognition in numbers. The number of lost tracking of the blue cellular phone in test setup 4 above 20 was not counted due to instability of the tracking

(a) (b)

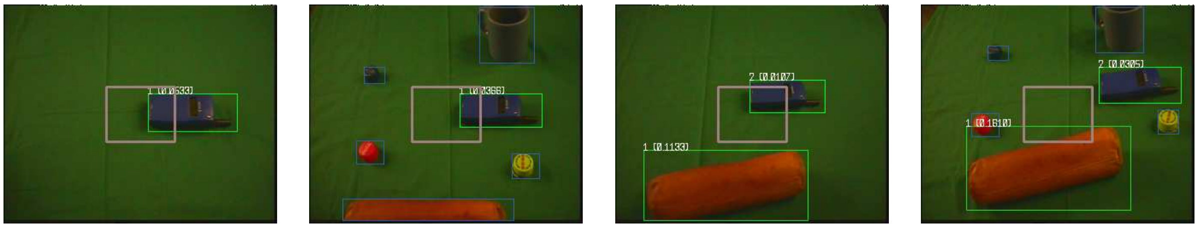
Figure 9.6: Figure (a) shows an example with correct tracking for reference. Figure (b) shows where the blue stopper is recognized as the blue cellular phone

The reason for the problem with the lost tracking of the blue cellular phone to the blue stopper is the choice of features. Out of the 61 features used 48 of them are color related in the color space HSV. As both the stopper and the cellular phone is blue, and the area of the objects are similar it is difficult to distinguish them from each other.

Test 3: Stable Recognition and Tracking of Moving Objects Over Time.

The purpose of this test was to test the systems stability towards recognition of moving objects over time. It is important that the system recognizes all objects inside the table, but only tracks the initialized objects. The test was conducted for 1 minute, and based on a visual registration.

Figure 9.7 shows some examples of correct recognition of moving objects. The objects of interest was tracked as required to pass the test, and table 9.2 shows in what degree.



(a) (b) (c) (d)

Figure 9.7: Results of test 3 in the test of correct recognition, where figure (a) shows test setup 1. Figure (b) shows test setup 2, figure (c) shows test setup 3, and figure (d) shows test setup 4

The evaluation of the test is based on the result of the test and the requirements to it. As seen in table 9.2 there is no lost track, and no misidentification.

The reason that this test do not experience the same problems with the blue cellular phone as the 2nd test is due to two factors. First is the phone closer to the camera and therefor evolving a larger area than the stopper. Secondly the camera is as static as possible with a head-mounted

Test setup	Number of lost tracking	Number of mis-identifications
1	0	0
2	0	0

	Number of lost tracking		Number of mis-identifications	
Test setup	Cellular phone	Pencil case	Cellular phone	Pencil case
3	0	0	0	0
4	0	0	0	0

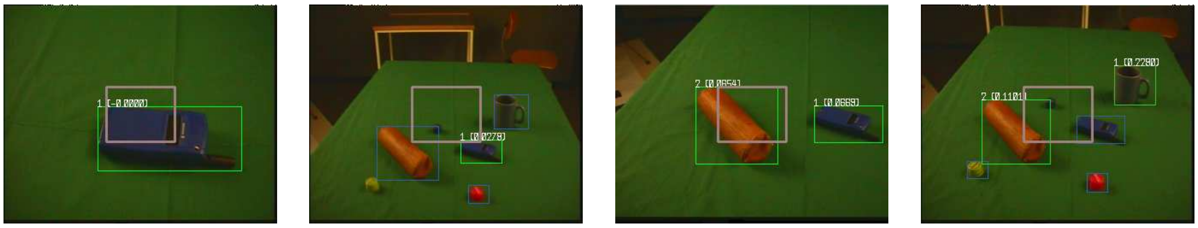
Table 9.2: Results of test 3 in the test of correct recognition in numbers. The number of lost tracking of the blue cellular phone in test setup 4 above 20 was not counted due to instability of the tracking

camera while the blue cellular phone is moving in the scene.

Test 4: Stable Recognition and Tracking of Objects During Scaling of the Objects (Zoom).

The purpose of this test was to test the systems stability towards recognition of scaled objects over time. It is important that the system recognizs all objects inside the table, but only tracks the initialized objects. The test was conducted for 1 minute, and was based on a visual registration.

Figure 9.8 shows some examples of correct recognition of scaled objects. The objects of interest was tracked as required to pass the test, and table 9.3 shows in what degree.



(a) (b) (c) (d)

Figure 9.8: Results of test 4 in the test of correct recognition, where figure (a) shows an object of interest in closeup in test setup 2, and figure (b) shows the same object of interest from a distance in test setup 2. Figure (c) shows two objects of interest in closeup in test setup 4, and figure (d) shows the same two objects of interest from a distance in test setup 4.

Test setup	Number of lost tracking	Number of mis-identifications
1	0	0
2	0	0

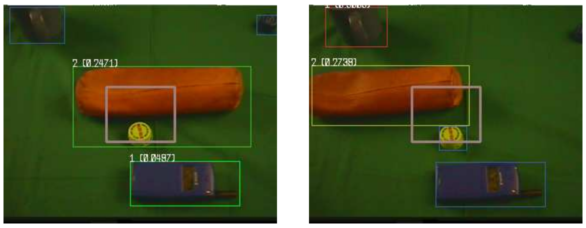
	Number of lost tracking		Number of mis-identifications	
Test setup	Cellular phone	Pencil case	Cellular phone	Pencil case
3	0	0	0	0
4	4	0	0	0

Table 9.3: Results of test 4 in the test of correct recognition in numbers.

The evaluation of the test is based on the result of the test and the requirements to it. As seen in table 9.3 there is a problem in test setup 4, where the system looses track of the blue cellular phone.

The problem in test setup 4 is shown in figure9.9 (b) where the blue cup is recognized as the blue cellular phone.

The reason for the problem with the lost tracking of the blue cellular phone to the blue cup is the same as for the 2nd test the choice of features. Out of the 61 features used 48 of them is



(a) (b)

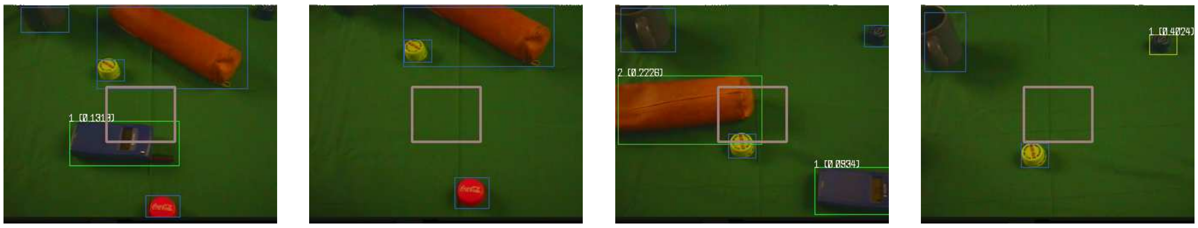
Figure 9.9: Figure (a) shows an example with correct tracking for reference. Figure (c) shows where the blue cup is recognized as the blue cellular phone

color related in the color space HSV. As both the stopper and the cellular phone is blue, and the area of the objects are similar because only a part of the blue cup is in the field of view of the camera. Then is difficult for the system to distinguish them from each other.

Test 5: Stable Recognition and Tracking of Objects after Reentering the Field of View.

The purpose of this test was to test the systems stability towards recognition of objects after reentering the field of view of the camera over time. It is important that the system recognizes all objects inside the table, but only tracked the initialized objects. The test was conducted for 1 minute, and was based on a visual registration.

Figure 9.10 shows some examples of correct recognition of objects after reentering the field of view of the camera. The objects of interest was tracked as required to pass the test, and table 9.4 shows in what degree.



(a) (b) (c) (d)

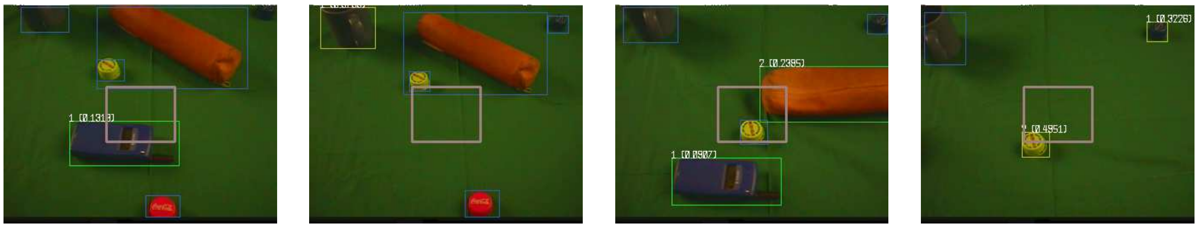
Figure 9.10: Results of test 5 in the test of correct recognition, where figure (a) shows an object of interest inside the field of view of the camera in test setup 2, and figure (b) shows the same object of interest outside the field of view of the camera in test setup 2. Figure (c) shows two objects of interest inside the field of view of the camera in test setup 4, and figure (d) shows the same two objects of interest outside the field of view of the camera in test setup 4.

Test setup	Number of lost tracking	Number of mis-identifications
1	0	0
2	8	0

	Number of lost tracking		Number of mis-identifications	
Test setup	Cellular phone	Pencil case	Cellular phone	Pencil case
3	0	0	0	0
4	7	0	3	0

Table 9.4: Results of test 5 in the test of correct recognition in numbers.

The evaluation of the test is based on the result of the test and the requirements to it. As seen in table 9.4 there is a problem in test setup 2 & 4, where the system looses track of the blue cellular phone in test setup 2, and both the blue cellular phone, and the pencil case in test setup 4.



(a) (b) (c) (d)

Figure 9.11: Figure (a) shows an example with correct tracking for reference. Figure (c) shows where the blue cup is recognized as the blue cellular phone. Figure (c) shows an example with correct tracking for reference. Figure (d) shows where the blue stopper is recognized as the blue cellular phone, and where the green stopper is recognized as the pencil case.

The reason for the problem with the lost tracking of the blue cellular phone and the pencil case is because they do not exist in the field of view of the camera, and the tracker tries to find the best match, which and the in the visual part of the scene is the two stoppers.

Test 6: Stable Recognition and Tracking of Objects During Rotation (Roll) of the Camera.

The purpose of this test was to test the systems stability towards recognition of objects during roll of the camera over time. It is important that the system recognized all objects inside the table, but only tracked the initialized objects. The test was conducted for 1 minute, and was based on a visual registration.

Figure 9.12 shows some examples of correct recognition of objects during roll of the camera. The objects of interest was tracked as required to pass the test, and table 9.5 shows in what degree.

(a) (b) (c) (d)

Figure 9.12: Results of test 6 in the test of correct recognition, where figure (a) shows an object during small roll of the camera in test setup 2, and figure (b) shows the same object during large roll of the camera in test setup 2. Figure (c) shows two objects of interest during small roll of the camera in test setup 4, and figure (d) shows the same two objects during large roll of the camera in test setup 4.

The evaluation of the test is based on the result of the test and the requirements to it. As seen in table 9.5 there is no lost track, and no misidentification.

The reason that this test do not experience the same problems with the blue cellular phone as some of the other test is due to two factors. The is used of invariant features, such as histogram and invariant moments. Secondly the camera is as static as possible in regarding pitch and yaw. The camera only exposed to rolls.

Test setup	Number of lost tracking	Number of mis-identifications
1	0	0
2	0	0

	Number of lost tracking		Number of mis-identifications	
Test setup	Cellular phone	Pencil case	Cellular phone	Pencil case
3	0	0	0	0
4	0	0	0	0

Table 9.5: Results of test 6 in the test of correct recognition in numbers.

9.4 Test for Time Consumption

The implemented system is test for time consumption. In order to do this a number of tests are conducted simulating different aspects of normal use. Below is a description of the tests conducted to measure the time consumption of the total system, and the main sub components.

9.4.1 Description of the Performed Tests

1. Function of fps compared to the number of objects on the table.
2. Average time consumption of the subcomponents of the system.
   - Input
   - Segmentation
   - Tracking
   - Predictor

Test setups

List of the three test setups describing the numbers of objects, and how they are moving in the scene.

1. The table contains an increasing number of objects in the range of zero to eight. For every test an object is added to the list of objects inside the table. All objects are static. The setup is illustrated in figure 9.13 (b).
2. The table contains eight objects in all test. For every test an object is added to the list of initialized objects inside the table. All objects are static during test. The setup is illustrated in figure 9.13 (c).
3. The table contains eight initialized objects in all test. For every test an object is added to field of view of the camera. All objects are static during test. The setup is illustrated in figure 9.13 (d).

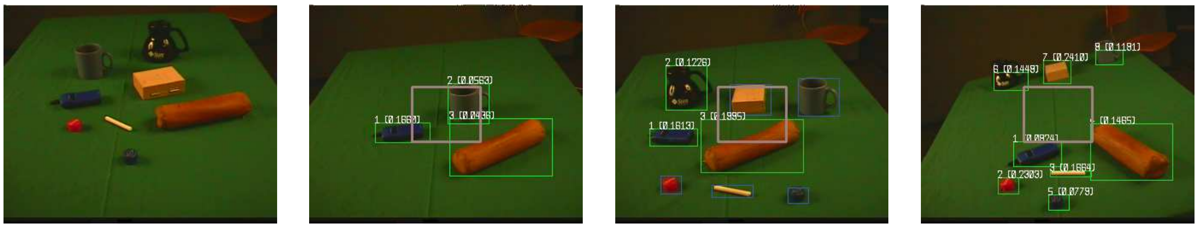
List of performed tests

There are conducted two tests for each test setup. Each of the tests is conducted for at time period of 1 minute. The reason for this limited time period is as previous mentioned due to a problem in the GNU compiler delivered with RedHat 7.0.

The two tests is described below.

1. **Function of fps compared to the number of object on the table**.

   Purpose : To measure the systems performance with an increasing number of objects inside the table. It is important that the system recognizes the objects placed on the



(a) (b) (c) (d)

Figure 9.13: The three test setups for the time consumption test. Figure (a) illustrates the reference for the three setups containing the objects used. Figure (b) illustrates the setup with an increasing number of initialized objects. Figure (c) illustrates the setup with an increasing number of initialized objects in the set of eight objects inside the table. Figure (d) illustrates the setup with an increasing number of objects in the field of view of the camera from a set of eight initialized objects. All eight objects are initialize, but only the lower 5 objects are in the field of view of the camera during test (test of 5 objects).

table and register them as objects (not part of the table). The test is conducted for 1 minute.

Environment setup : The camera is head-mounted and held as static as possible when using as head-mounted camera. The settings of the camera and framegrabber is described in section 9.1.

The table is covered by a piece of green cloth, and is initialized before conducting the test.

The numbers of objects ranges from zero to eight depending on the test setup. The objects placed on the table is static at all time during the test.

Expected result : A measurement in fps for the range of objects.

2. **Time consumption of subcomponents**.

Purpose : To measure the systems performance with multiple objects inside the table. The number of objects depend on the test setup. It is important that the system recognizes the objects placed on the table and register them as objects (not as a part of the table). The test is conducted for 1 minute.

Environment setup : The camera is head-mounted and held as static as possible when using a head-mounted camera. The settings of the camera and framegrabber is described in section 9.1.

The table is covered by a piece of green cloth, and is initialized before conducting the test.

The number of objects ranges from zero to eight depending on the test setup. The objects placed on the table is static at all time during the test.

Expected result : A measurement of the time consumption of the subcomponents. The subcomponents measured are; Input, segmentation, feature extraction, feature matching, prediction, and the SVD in feature matching.

Measurement for the performed tests.

The two tests performed are measured differently, and therefore divided into two types of measurement.

Test 1 is measured in fps. This is done by diving the number of frames captured by the time period in seconds.

Test 2 is measured in milliseconds (ms). The selection of milliseconds in stead of a more accurate time granulation is made to decrease the fluctuations of the measured time due to scheduled system calls, e.g. network communication.

9.4.2 Results of the Tests Performed in the Different Test Setups

Test 1: Function of fps Compared to the Number of Object on the Table

The purpose was to measure the systems performance with an increasing number of objects inside the table. It is important that the system recognizes the objects placed on the table and register them as objects (not part of the table). The test is conducted for 1 minute, and is measured in fps.

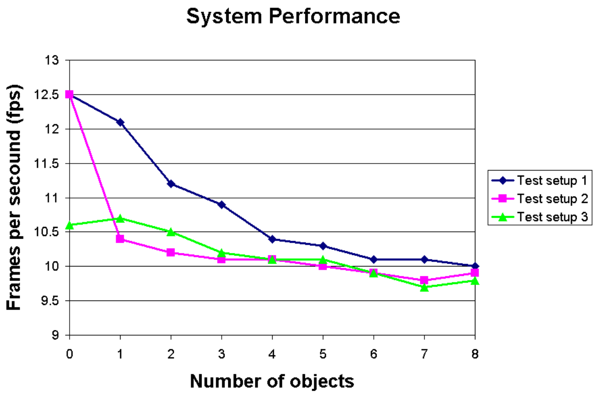


Figure 9.14: Result of test 1 in the test for time consumption, which shows the framerate of the system decreases as the number of objects increases. The maximum number of objects before exceeding the requirement of 10 fps are 6

The evaluation of the test is based on the result of the test and the requirements to it. As seen in figure 9.14 both test setup 1 & 2 with zero objects inside the table are running at maximum framerate, which is 12.5 for the framegrabber with current settings. This is due to the subcomponents that extracts and match features are not executed for zero objects. The time consumption of these subcomponents are evaluated in the test time consumption of sub components.

The trend of the three test setups are fitting each other with an increasing number of initialized and tracked objects in the field of view of the camera. The reason for that is due to the number of initialized objects inside the field of view of the camera is eight for all three test setup. The little variation in the framerate is because of the different time consumption in extraction the features from the objects. This is due to the location of the objects are not unique for the three test setups, which resolves in different numbers of pixels describing the objects, which again resolve in different calculation time for each feature, e.g. calculation of moments.

Test 2: Time Consumption of Subcomponents

.

The result of test setup 1

The purpose was to measure the systems performance with multiple objects inside the table. The number of objects is depends on the test setup. It is important that the system recognizes the objects placed on the table and register them as objects (not part of the table). The test is conducted for 1 minute, and the time consumption of the subcomponents is measured in milliseconds.

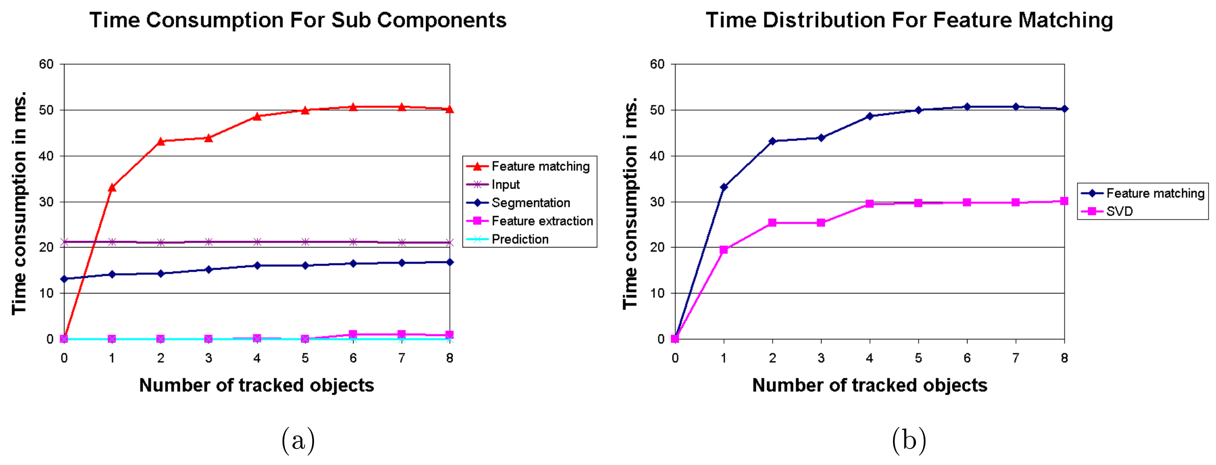


Figure 9.15: Results of test setup 1 in test 2 in the test for time consumption. Figure (a) shows the trend in the time consumption of the different subcomponents during an increase in the number of objects. Figure (b) shows the time consumption of the SVD compared to the total time used by the feature matching component.

The evaluation of the test is based on the result of the test and the requirements to it. As seen in figure 9.15 (a) is the feature matching component the main time consumer. As the feature matching component is used to match the objects, it resolves in no time consumption for zero objects initialized. The rest of the subcomponents uses nearly constant time with increasing number of objects.

Figure 9.15 (b) shows the time distribution for the feature matching component. As the SVD, see section 8.6 for more details, is the most calculation heavy algorithm in the feature matching component, the time consumption of this algorithm is investigated. The total time used by this component depends on the time consumption of the SVD algorithm.

The result of test setup 2

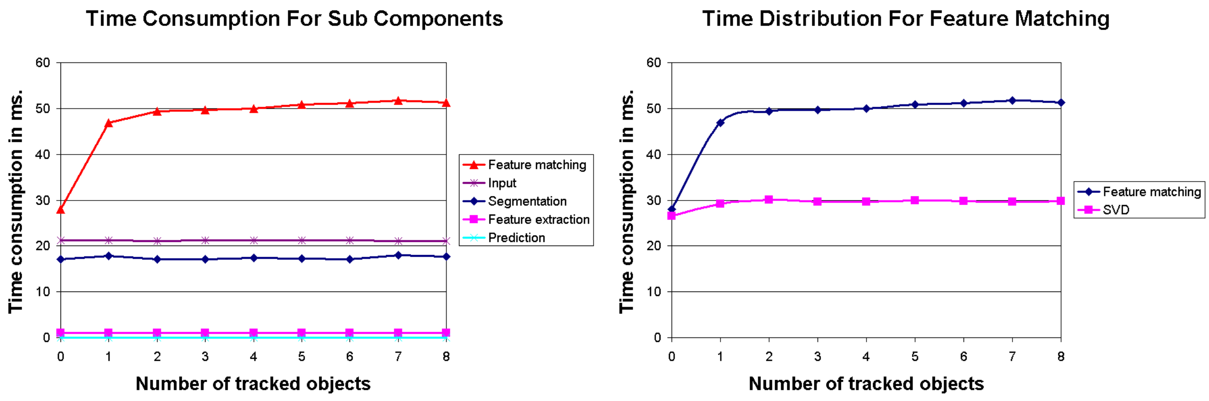
The purpose was to measure the systems performance with multiple objects inside the table. The number of objects is depends on the test setup. It is important that the system recognizes the objects placed on the table and register them as objects (not as a part of the table). The test is conducted for 1 minute, and the time consumption of the subcomponents is measured in milliseconds.

The evaluation of the test is based on the result of the test and the requirements to it. As seen in figure 9.16 (a) is the feature matching component the main time consumer. As the number of objects is eight during the entire test, and the number of objects the system tracks increases, it resolves in a minimum time consumption for the feature matching component, used for calculate the SVD, see section 8.6 for more details, on the eight objects. The rest of the subcomponents uses nearly constant time with increasing number of objects.

Figure 9.16 (b) shows the time distribution for the feature matching component. As the SVD is the most calculation heavy algorithm in the feature matching component, the time consumption of this algorithm is investigated. The total time used by this component depends on the time consumption of the SVD algorithm.

The result of test setup 3

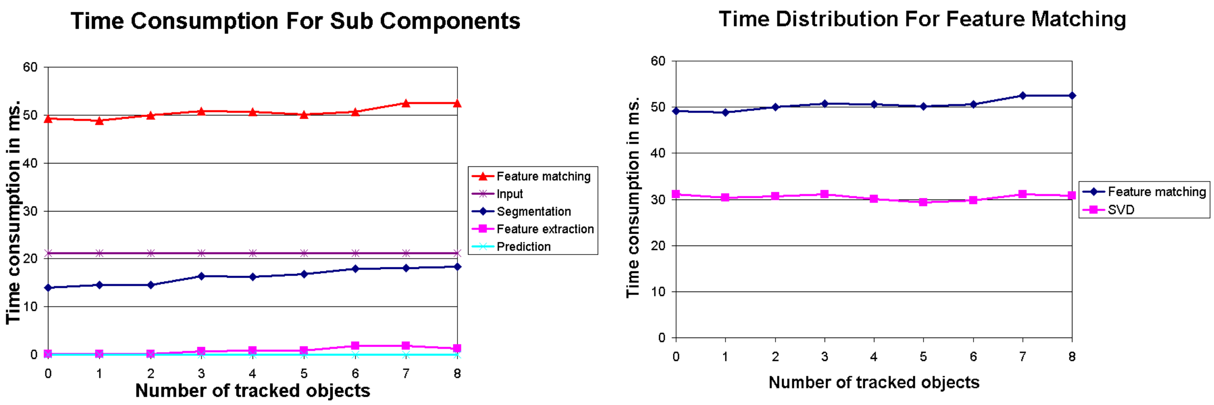
The purpose was to measure the systems performance with multiple objects inside the table. The number of objects is depends on the test setup. It is important that the system recognizes the objects placed on the table and register them as objects (not as a part of the table). The



(a) (b)

Figure 9.16: Results of test setup 2 in test 2 in the test for time consumption. Figure (a) shows the trend in the time consumption of the different subcomponents during an increase in the number of objects. Figure (b) shows the time consumption of the SVD compared to the total time used by the feature matching component.

test is conducted for 1 minute, and the time consumption of the subcomponents is measured in milliseconds.



(a) (b)

Figure 9.17: Results of test setup 3 in test 2 in the test for time consumption. Figure (a) shows the trend in the time consumption of the different subcomponents during an increase in the number of objects. Figure (b) shows the time consumption of the SVD compared to the total time used by the feature matching component.

The evaluation of the test is based on the results of the test and the requirements to it. As seen in figure 9.17 (a) is the feature matching component the main time consumer. As the number of initialized objects is eight during the entire test, and the number of objects visible to the system increases, it resolves in a near constant time consumption for the feature matching component. The rest of the subcomponents uses nearly constant time with increasing number of objects.

Figure 9.17 (b) shows the time distribution for the feature matching component. As the SVD, see section 8.6 for more details, is the most calculation heavy algorithm in the feature matching component, the time consumption of this algorithm is investigated. The total time used by this component depends on the time consumption of the SVD algorithm.

9.5 Summary

The system test was divided into two types of test. They are,

- Test for correct recognition
- Test for time consumption

The test for correct recognition showed that the implemented system is robust toward roll and scaled objects. The reason for this is the number of invariant features, e.g. moments. The system recognized objects with unique colors, because of the number of color features, but objects with similar color and size caused troubles for the system, due to the lack of variant features to support the invariant features. The predictor recognized objects even though the have left the field of view of the camera, this is due to the missing motion registration over time.

The test for time consumption showed that the system framerate decreases as the number of initialized objects increases. If there are more than 6 initialized objects the framerate exceeds the requirement of 10 fps. The main time consumer was the feature matching component, and especially the SVD algorithm. The rest of the components uses nearly a constant amount of time with a different number of initialized objects.

CHAPTER 10

Conclusion

This chapter contains the conclusion and discussion of the project.

10.1 Conclusion

The purpose of this project was to determine how a robust real-time system can be constructed, which by the use of a head-mounted camera, tracks placeholder items on a table. The motivation for constructing such a system was the EU-funded project ARTHUR (Augmented Round Table for Architecture and Urban Planning), which is an Augmented Reality system used for round table meetings. ARTHUR is based on a computer vision system for tracking the everyday items (placeholder items) for later graphical augmentation.

This project focuses on the tracking part of ARTHUR, and the aim was to determine whether or not it is possible to track everyday items in an everyday environment using purely computer vision based methods. This lead to the specific problem of the project.

> How can a robust system be constructed, which by the use of a single head-mounted camera and solely indoor lighting, tracks rigid placeholder items on a table with a uniformly colored cloth, while obtaining update rates of at least 10 frames per second?

10.1.1 System Design

The system designed is shown in figure 10.1. The design of the system was divided into three main components, illustrated by dashed lines. Each of these components represents a group of functionality. Together they form the computer vision based tracker of this project. The input and output components supplies the system with information in form of images and user interaction.

The three main components of the system was:

- Stabilization
- Segmentation

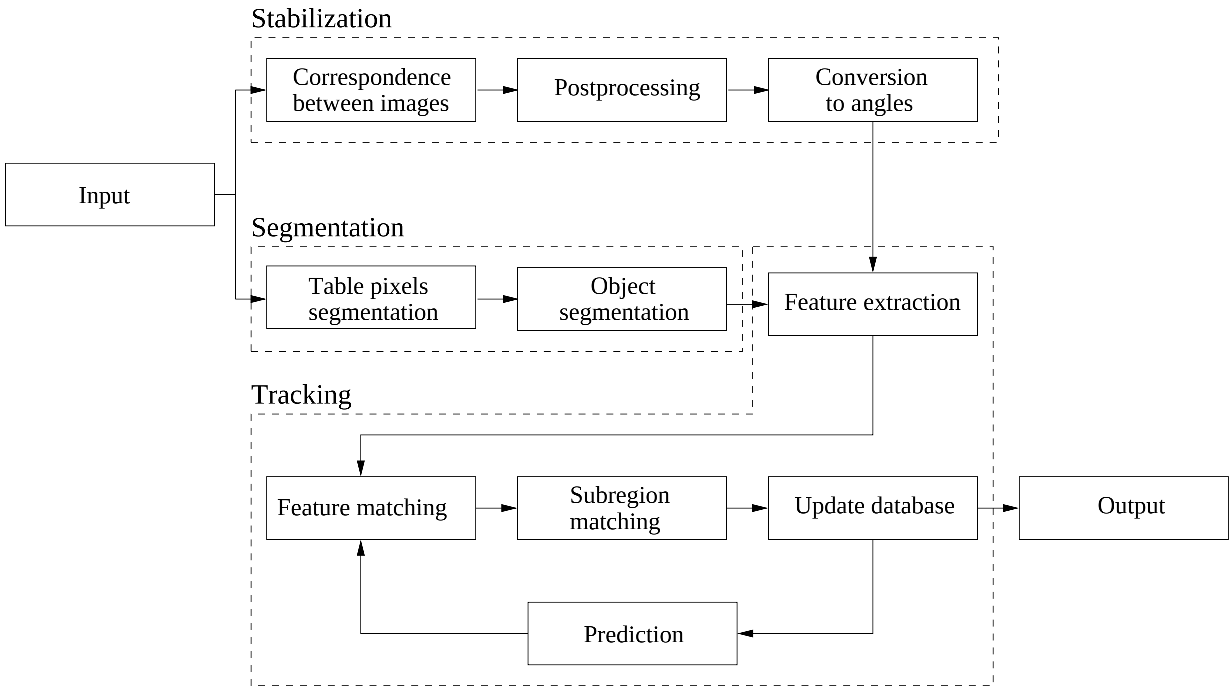


Figure 10.1: The design overview of the system.

- Tracking

The purpose of the stabilization component was to register the rotations of the head-mounted camera about the three axes (pitch, yaw, and roll), the so-called *ego-motion*. The stabilization component was divided into three sub components: determination of the correspondence between images, postprocessing of the disparity field, and conversion of the disparity into ego-motion.

The windows based correlation technique was selected for determining the correspondence between images. A test was made in order to determine the optimal parameters, which was found to be a resolution of 160×120 pixels, using histograms with 64 buckets from windows of 16×16 pixels. The histograms were calculated from the saturation channel of the HSV color space, and a threshold on the standard deviation of the windows was used in order to improve accuracy.

The 2D affine model was selected for conversion of the disparity field into ego-motion of the camera. This approach was primarily decided due to the simplicity of the method.

The purpose of the segmentation component was to remove unimportant information, in this case everything outside the table. This component was divided into two subcomponents: The table pixel segmentation which generates the mask of the table, and the object segmentation which finds all the objects inside the table.

The table pixels segmentation subcomponent was used to segment the table. This was performed in the hue channel of the HSV color space, and the threshold levels were based on an analysis of the table. Postprocessing on the thresholded image consists of binary morphology, and finally the table was pin pointed.

The object segmentation subcomponent was used to segment the objects inside the table. This was achieved by performing an exclusive-or operation between the original image and the convex hull binary mask of the table. From this the objects were found as BLOBs in the binary image.

The purpose of the tracking component was to recognize the segmented items as objects from the item database. Two types of objects are to be recognized: separable objects, and partially occluded objects. Initially all objects found are treated as separable objects.

Tracking of the separable objects were based on features extracted from the found objects followed by Mahalanobis distance-based feature matching between the extracted features and the predicted features for the objects in the item database. The item database was finally updated with information from the last presentation of the objects.

Tracking of the partially occluded objects was based on a subregion histogram matching technique, which divided objects into a number of smaller windows.

10.1.2 The Obtained Results

The results obtained are based on the conducted system test. The system test was conducted on a prototype implemented in C++, using the third party library OpenCV. The following components were fully implemented in the prototype: The input, segmentation, tracking, and the output component.

The results showed that the system was robust with respect to roll and scale of objects. Objects with unique colors were tracked with a higher degree of robustness than objects with similar colors. Objects with similar colors required a significant difference in size in order to obtain a high degree of robustness.

The requirement of a system update rate of 10 fps was tested and confirmed for the case of six or fewer objects in the camera's field of view. The most time consuming component of the system was the feature matching component, and especially the *singular value decomposition* (SVD) algorithm used in this component.

From the obtained results it can be concluded that it is possible to construct a computer vision based tracking system, which has the capability to track everyday items using a single head-mounted camera while obtaining an update rate of 10 fps. The desired update rate can be achieved for a maximum of six everyday items in the current prototype. The robustness of the system depends on the uniqueness (in the sense of color or size) of the placeholder items.

10.2 Discussion

In the following, possible improvements of the system are discussed.

10.2.1 Stabilization

During the development of the stabilization component several problems were found.

The two main problems were the following:

- Outliers in the disparity fields
- Drift of the calculated angles

Outliers are false matches in the disparity fields. Drift is the accumulated error due to errors in the estimation of ego-motion, e.g. numerical errors during the calculation of the rotations, and errors caused by outliers.

The following techniques can be applied in order to improve the accuracy of the stabilization component:

- Rejection of Outliers by Rotations (Adam, Rivlin, and Shimshoni, 2001)
- Random Sample Consensus (RANSAC)(Trucco and Verri, 1998, p. 326)
- Trilinear Tensor, see section D.5

- Mosaic, see section D.6

Furthermore, magnetic trackers, stereo vision, and calibration could be useful for improving the accuracy of the stabilization component. However, these techniques were not included in the proposal of the project.

10.2.2 Segmentation

A number of things could be done in order to improve the segmentation component.

In order to improve the segmentation of the table and objects inside the mask of the table, other segmentation algorithms could be applied. It is likely that the following algorithms could be used to improve the robustness of the segmentation:

- Region growing, see section C.4.2
- Region splitting and merging, see section C.4.3

10.2.3 Tracking

In the following a number of suggestions are made on how to improve the tracking.

Feature extraction is used for extraction of characteristic information describing the objects. The system test showed the need for additional features. These features could be:

- External characteristics. E.g. the ratio between height and width or moments based on the boundary of an object.
- Domain specific features. For example, how near an object is positioned to a border (image plane or table edge).

Feature matching is used in order to determine the probability that two objects match. In the current implementation the SVD is the most resource demanding part of the algorithm. One of the following methods could be used to improve the feature matching:

- Replace the Mahalanobis distance by the χ^2 (Chi-square) (see appendix E) distance metric. χ^2 compensates for feature scaling problems without the need for calculating the SVD.
- Implement the subregion matching method for handling of occlusions.
- Explicitly model the hands of the user to overcome interaction problems.

Prediction is used for predicting the state of the object. A simple linear extrapolation is currently used. The predictor can be optimized using:

- Regression of samples from e.g. the last 5-6 time steps.
- Either Kalman or Condensation filtering.

This concludes the suggestions for optimization of the components in the developed system.

Finally, it should be noted that the use of OpenCV has decreased the time spent on implementation. It is therefore highly recommended that future development of the system should use OpenCV or similar libraries.

Bibliography

The Matlab Image Toolbox. Mathworks Inc, 1999.

A. Adam, E. Rivlin, and I. Shimshoni. ROR: Rejection of Outliers by Rotations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):78–84, January 2001.

F. Badra, A. Qumsieh, and G. Dudek. Rotation and Zooming in Image Mosaicing. pages 50–55, October 1998. ISBN 0-8186-8606-5.

H. Birk, T. B. Moeslund, and J. S. Sørensen. Tracking Multiple Objects using Multiple Cues. Technical report, Laboratory of Image Analysis, Aalborg University, 1996.

G. R. Bradski. Computer Vision Face Tracking as a Component of a Perceptual User Interface. *IEEE Workshop on Applications of Computer Vision, Princeton*, pages 214–219, 1998.

W. Burger and B. Bhanu. Estimating 3D Egomotion from Perspective Image Sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11), November 1990.

T. H. Cormen, C. E. Lesiserson, and R. L. Rivest. *Introduction to Algorithms.* the MIT Press, 1998. ISBN 0-262-53091-0.

E. R. Davies. *Machine Vision.* Academic Press, 2nd edition, 1997. ISBN 0-12-206092-X.

ELMO - Electricity Light Machine Organization. QP-49 H / MP -49 H / CC-491.
WWW: http://www.elmo.de/microccd/qp49h.htm, 2001.

EMJ Embedded Systems. Specifications for Picasso PCI-RGB-2SQ.
WWW: http://www.emjembedded.com/products/multimedia/spec.pci1042sq.html, 2001.

Frank Warmerdam and Mike Welles. TIFF Software.
WWW: http://www.libtiff.org/, 2001.

R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 3rd edition, 1993. ISBN 0-201-50803-6.

A. Heckbert. Matrix Distance.
WWW: http://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/matrdist.htm, 2000.

R. Hosie. Description of the kalman filter.
WWW: http://www.computing.edu.au/∼robin/thesis/node17.html, 1995.

Intel Corporation. Open Source Computer Vision Library.
WWW: http://www.intel.com/research/mrl/research/opencv/, 2001.

M. Isard and A. Blake. CONDENSATION – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29(1):5–28, 1998.

IST. Augmented Round Table for Architechture and Urban Planning. Proposal IST-2000-28559 by Information Societies Technology (IST), 2001.

R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill International, 1995. ISBN 0-07-113407-7.

B. Jähne, H. Haussecker, and P. Geissler. *Handbook of computer vision and applications, volume 1, Sensors and Imaging*. ACADEMIC PRESS, 1999. ISBN 0-12-379771-3.

E. B. Koller-Meier and F. Ade. Tracking multiple objects using the Condensation algorithm. *Robotics and Autonomous Systems*, 34(2-3):93–106, 2001.

K. N. Kutulakos and J. R. Vallino. Calibration-Free Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 4(1), January-March 1998.

S. Leppajarvi. *Image Segmentation and Analysis for Automatic Color Correction*. PhD thesis, Lappeenranta University of Technology, 1999.

C. B. Madsen. ARTHUR. Computer Vision and Graphics project proposal, Aalborg University, 2001.

K. Prazdny. Determining the instantaneous direction of motion from optical flow generated by a curvilinear moving observer. *Computer Graphics Image Processing*, 17:238–248, 1981.

B. Rousso, S. Avidan, A. Shashua, and S. Peleg. Robust Recovery of Camera Rotation from Three Frames. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 796–802, June 1996. ISBN 0-8186-7258-7.

Sam Lantinga. Simpel DirectMedia Layer.
WWW: http://www.libsdl.org/, 2001.

K. S. Shanmugan and A. M. Breipohl. *Random Signals: Detection, Estimation and Data Analysis*. Wellesley-Cambridge Press, 1988. ISBN 0-471-81555-1.

A. Shashua. Albebraic Functions For Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, August 1995.

A. Shashua. Model-Based Brightness Contraints: On Direct Estimation of Structure and Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):992–1015, September 2000.

M. Sonka. 3D Vision (Part I): Geometry for 3D vision.
WWW: http://www.icaen.uiowa.edu/∼dip/LECTURE/3DVisionP1_2.html#basics, 2000.

SONY electronics. DCR-VX2000.
WWW: http://64.14.40.97:80/explore_products/productinfo.jsp, 2001.

G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2nd edition, 1998. ISBN 0-9614088-5-5.

M. Störring, H. J. Andersen, and E. Granum. Physics-based modelling of human skin colour under mixed illuminants. *Laboratory of Computer Vision and Media Technology, InterMedia-Aalborg Aalborg Universitet*, 2000.

R. Torre, S. Balcisoy, P. Fua, M. Ponder, and D. Thalmann. Augmented Reality for Real and Virtual Humans. **WWW:** http://ligwww.epfl.ch/∼fua/augm/, 2000.

E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998. ISBN 0-13-261108-2.

J. Vallino. Introduction to Augmented Reality.
WWW: http://www.cs.rit.edu/∼jrv/research/ar/introduction.html, 2000.

D. R. Wilson and T. R. Martinez. Improved Heterogeneous Distance Functions.
WWW: http://axon.cs.byu.edu/∼randy/jair/wilson.all.html, 1997.

Wolfgang Scherr. Video For Linux.
WWW: http://roadrunner.swansea.linux.org.uk/v4l.shtml, 2001.

APPENDIX A

Representation Schemes

This appendix describes some methods to represent the objects that has been segmented out of the image. This appendix is primarily based on (Gonzalez and Woods, 1993, chapter 8)

The image has been divided into objects as described in appendix C and a scheme to represent the objects is needed. This can be done in various ways. This appendix will try to describe some of the most standard methods. It should be noted that some approaches uses the raw pixels gotten from the segmentation but that is more the exception to the rule than the standard. The methods described in this appendix will be chain codes, polygon approximations, and convex hull.

A.1 Chain Codes

One of the most basic way to represent the objects found during the segmentation is chain codes. The principel in chain codes is to use straight line segments of a specific length and direction to represent the boundary of the object. Usually these representations are based upon 4 or 8 connectivity of the segments. The direction of the segments could be coded using a numbering scheme as shown in figure A.1

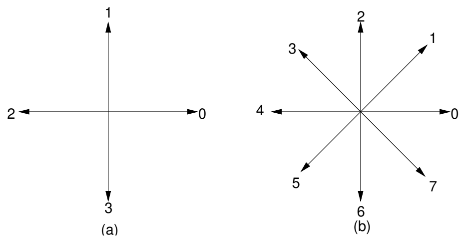


Figure A.1: Directions for (a) 4-connectivity (b) 8-connectivity chain codes.

When the image is processed and the segmentation is being done. The image can be said to be divided into grids with equal horizontal and vertical spacing. A possible way to represent the object by chain code was to generate a chain code going in one of the directions around the object following the boundary of the object; either clockwise or counterclockwise. This is generally not a good representation due to two things:

1. The chain code will usually be very long

2. Small disturbance due to noise or imperfect segmentation will generate changes in the chain code that may be releated to the shape of the object.

A frequent approach to solve the the problems mentioned above is to use a larger grid spacing. When traversing the boundary a boundary point is assigned to nearest grid value in the new grid. The new boundary can then be represented with one of the schemes presented in figure A.1. However it is obvious that the chain code of the boundray depends on the starting point of the chain code.

To solve that problem the chain code can be normalized by treating the chain code as a circular sequence of direction numbers and redefine the starting point so that the resulting chain code form an integer as small as possible. In order to normalize for the rotation of the boundary the *first difference* of the chain code can be used. The *first difference* is made by counterclockwise counting the number of directions that separate two adjacent elements of the code. Normalizing with regard to size can be done by altering the size of the resampling grid.

The aboved mentioned normalizations are only excact if the boundaries themself are invariant to both rotation and scale change which is rarely the case. This can be reduced by orienting the resampled grid along the principal axes of the object to be coded also known as shape codes. This is discused next.

A.2 Shape Codes

As was mentioned in section A.1 the first difference chain codes was depedent upon the starting postion on the chain. A way to overcome this is by using a shape number. A shape number is the first differrence of the chain code which has the smallest magnitude. This is examplified in figure A.2 all though just shown for a 4-connected boundary and the starting point is marked by the dot. Additionally the order of a shape code is defined as the number of digits in its representation.

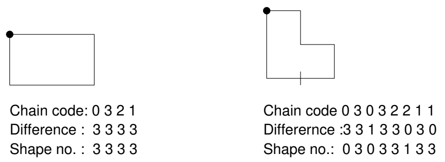


Figure A.2: Summary on chain code, first difference, and shape numbers.

Even though the first difference is invariant to rotation it is still dependent in general of the orientation of the grid. The grid orientation could be normalized by aligning the grid after the major and minor axis of the object.

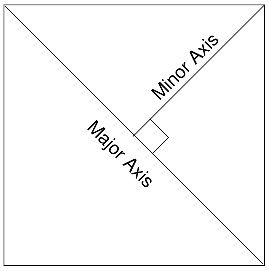


Figure A.3: Illustration of major and minor axis.

The major axis of a boundary is defined as being the straight-line segments joining the two points farthest from each other. Where as the minor axis is the axis perpendicular to the major axis and of such a length that a box could be formed that just encloses the object. The rectangle just described is known as the basis rectangle. By alignning the chain code grid with the basis rectangle will in most cases result in a unique shape number.

A.3 Polygon Approximation

Using polygon approximation to describe the boundary of an object gives the advantage that the boundary can be approximated with an arbitrary accuracy. It can be seen that the approximation will be excact if the number of segments is equal to the number of points in the boundary so that each adjacent point in the boundary defines a segment. It is however clear that in practise it the goal is to capture the essential shape with the fewest number of polygon segments.

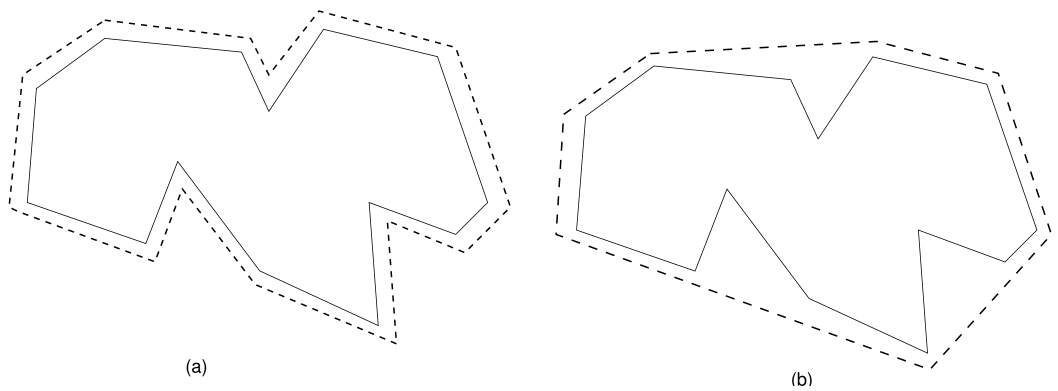


Figure A.4: Polygon approximation. Figure (a) shows the optimal approximation while (b) is the minimum perimeter polygon

Figure A.4(a) represents the more accurate approximation of the object. The optimal boundary can then be thought of as a rubber band which is shrunk as shown in figure A.4(b). This gives the boundary the minimum parameter and thus the optimal representation.

Gonzalez and Woods (1993) proposses another way to handle polygon approximation by using merging based on error or other criteria. An approach could be to merge points along a boundary until the least square error line fit of the points merged so far exceeds a threshold. When the threshold is exceed the paramaters for the line are stored and the error is reset whereafter the procedure is repeeated. When the procedure finishes the intersections of adjacents lines form the vertices of the polygon. It can seen that one of the problems with the above mentioned method is that vertices in general does not correspond to inflections such as corners in the boundary. This is a problem since when tracking a straight line with this method and the line turns a corner the method will overshoot since it will take a number of points before the threshold is exceeded. In order to handle this problem splitting along with merging can be applied.

Gonzalez and Woods (1993) proposses an approach to splitting by dividing a segment repeatedly into two parts until a given criterion is met. A method could be to use the maximum perpendicular distance from a boundary segment to a line joining its two end points not exceed a preset threshold. Naturally if the threshold is exceed the points becomes a vertex which will divide the initial segment into two smaller segments. It is clear that the described approach will seek the most prominent inflection points since those are most often the farthest apart.

A.4 Convex Hull

When the boundary has contains one or more significant concaveties that carries shape information it can be described robustly using convex hull.

The *convex hull* H of an arbitrary set of points Q is the smallest convex polygon containing Q as pictured in figure A.5. Intuitively the points Q can be thought of as nails sticking out from a board, and the convex hull is then the shape formed by putting a rubber band around the nails.

Due to digitization, noise, and variations in segments the digitial boundaries tend to be irregular. In order to overcome this problem it is standard practise to smooth the boundary before partitioning.

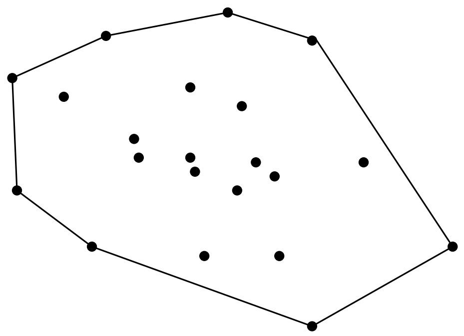


Figure A.5: A set of points Q and it's convex hull H.

There exists several methods to compute the convex hull. Some of the methods will be listed here but for a thorough treatment look to (Cormen, Lesiserson, and Rivest, 1998). Some methods used to compute convex hull is:

- Gramhan's Scan
- Jarvis's March
- Incremental Method
- Divide-and-Conquer Method
- Prune-and-Search Method

Most of the above mentioned methods can calculate the convex hull with a complexity corresponding to $O(n \log n)$.

APPENDIX B

Color Models

This appendix will start by describing the characteristics of color and the formation of color images. With that in place a description of the standard color models will be given.

B.1 Color Characteristics

Humans generally distinguish colors based on three characteristics (Gonzalez and Woods, 1993):

Brightness describes the achromatic intensity.

Hue describes the dominant color perceived by an observer that is red, orange, or yellow is an objects hue.

Saturation describes the purity of the color or the amount of white light mixed with a hue.

Pure colors, such as red, green, yellow or blue, are fully saturated, while pink (red and white) are less saturated. A number of color models or spaces are based on these characteristics. One of these is described later.

B.2 Color Image Formation

Sir Isaac Newton first revealed the spectral properties of light as he observed that sunlight passing through a glass prism is spread into a continuous spectrum of colors ranging from violet to red. This corresponds to the range of wavelength from approximately 380 nm to 750 nm, which is known as the visible spectrum of the electromagnetic waves. Gonzalez and Woods (1993) and Leppajarvi (1999)

Based on the observation of Sir Isaac Newton the formation of color images can be modeled by spectral integration given the following (Störring, Andersen, and Granum, 2000):

- λ is the wavelength.
- $E(\lambda)$ is the spectrum of the light source.
- $\rho(\lambda)$ is the reflectance of the object.
- $L(\lambda) = E(\lambda) \cdot \rho(\lambda)$ is the reflected light of the object.

- $f(\lambda)$ is the spectral sensitivity of the observer i.e. camera or scanner.

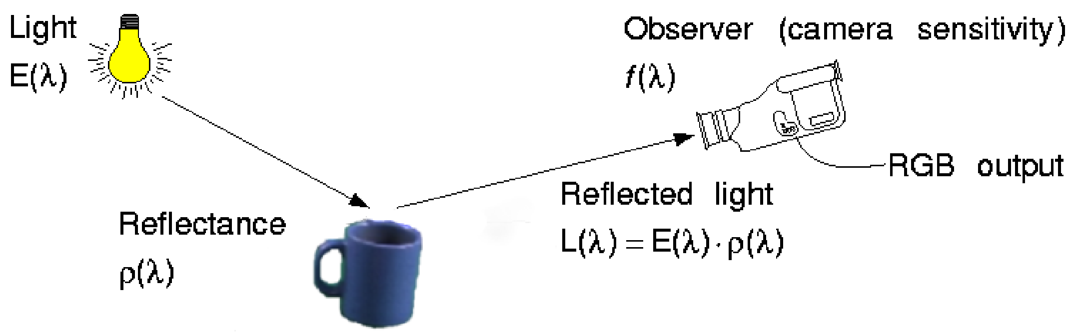


Figure B.1: A color image formation scenario.

Figure B.1 pictures the scenario for obtaining a color image with a camera. The formation of the three primary colors R (red), G (green) and B (blue), which is the basic color model for image processing since color cameras, displays, and scanners are often provided with RGB signal input or output, can be stated as:

$$R = \int_{\lambda} E(\lambda)\rho(\lambda)f_R(\lambda)d\lambda \tag{B.1}$$

$$G = \int_{\lambda} E(\lambda)\rho(\lambda)f_G(\lambda)d\lambda \tag{B.2}$$

$$B = \int_{\lambda} E(\lambda)\rho(\lambda)f_B(\lambda)d\lambda \tag{B.3}$$

where $f_R(\lambda)$, $f_G(\lambda)$ and $f_B(\lambda)$ are the spectral sensitivities of for example a camera. Figure B.2 gives an example of the spectrum of a light source, object and the three RGB color spectral sensitivities.

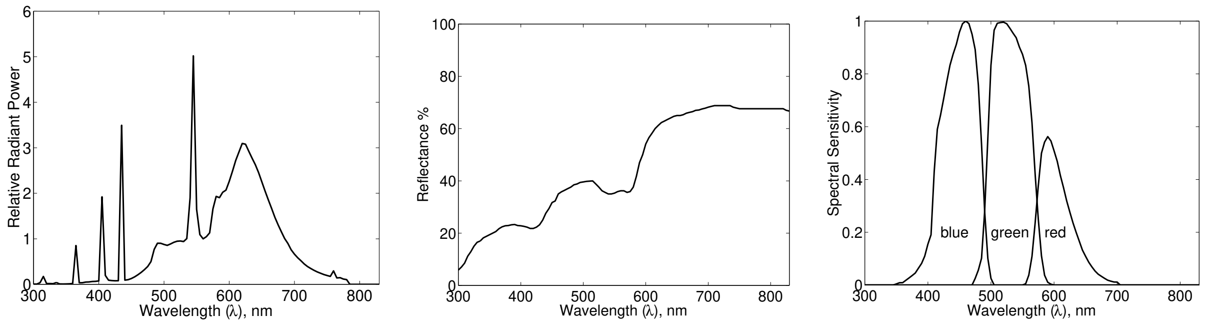


Figure B.2: Example of the spectrums of a light source, object and camera sensitivities. (Störring et al., 2000)

In the following the most widely used color models will be introduced. Given that color models or spaces are specifications of a three-dimensional system of color sensations, where colors are represented by points in these spaces.

B.3 RGB Color Model

As stated before the RGB model is an approximation of the color spectrum. That is, the mixture of the three primary colors can generate only a subspace of the visible color spectrum. The RGB model approximates the perception of color by the human eye, which has a trichromatic spectrum, like the one in figure B.2.

The RGB space forms a cube, where each color is described by its RGB components, that is, a vector from the origin in a Cartesian coordinate system and limited by the surface of the cube. The grey scale colors are placed on the diagonal of the cube from black (R=G=B=0) to white (R=G=B=1), with the R, G and B values normalized to the range $[0, 1]$. In practice the values of R, G, and B are each represented as 8 bit values in the range [0,255]. Figure B.3 pictures the RGB unit cube with some colors indicated.

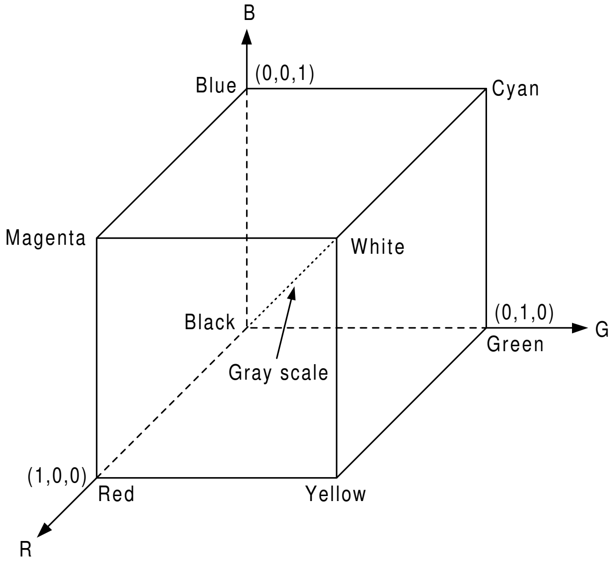


Figure B.3: The RGB color unit cube. Based upon (Gonzalez and Woods, 1993, p. 226)

The disadvantages of the RGB model are:

- High correlation between its components in natural images.
- Difficult to visualize a color given the RGB components.

B.4 The rgb Color Model

In order to eliminate the effect of illumination intensity, normalized colors, r, g, and b have been introduced (Störring et al., 2000):

$$r = \frac{R}{R+G+B} \tag{B.4}$$

$$g = \frac{G}{R+G+B} \tag{B.5}$$

$$b = \frac{B}{R+G+B} \tag{B.6}$$

$$\text{where } r+g+b = 1 \tag{B.7}$$

That is, the rgb plane or chromaticity plane is fully specified by two of its components for example r and g.

The rgb plane is perpendicular to the intensity line in the RGB cube and crosses the points for pure red (1,0,0), green (0,1,0) and blue (0,0,1). Figure B.4 depicts the rgb plane in the RGB unit cube and the chromaticity plane specified by r and g.

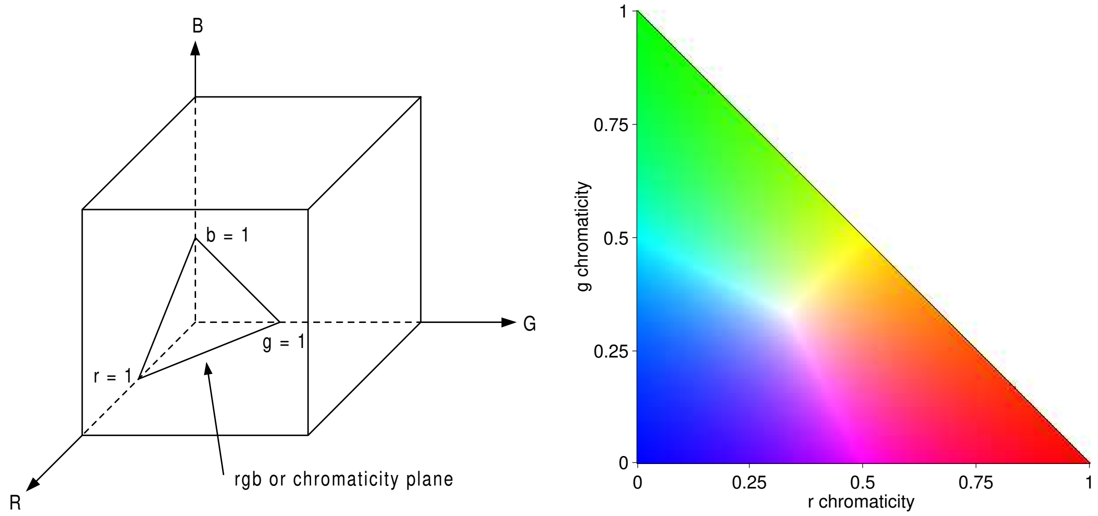


Figure B.4: The RGB color cube and the chromaticity plane. Left: The RGB color cube with the chromaticity plane in it. Right: The chromaticity plane specified by r and g. (Störring et al., 2000)

The rgb color model is especially useful for analyzing the color of the reflected light independent of the scale of intensity. It is however still difficult to visualize the color given by the r and g component.

B.5 HSV Color Model

As stated before human color perception depends on color attributes such as hue, saturation and brightness. That is the reason why several color spaces use attributes closely related to these. The HSV model is based on the attributes; hue, saturation, and value. Hue describes a pure color, saturation is the extent to which a pure color is mixed with white light, and the value expresses the brightness. Hue, saturation and value can be obtained from RGB colors with the following equations:

$$H = \cos^{-1}\left(\frac{0.5 \cdot [(R-G)+(R-B)]}{\sqrt{[(R-G)^2+(R-B)(G-B)]}}\right) \tag{B.8}$$

However if $\frac{B}{V} > \frac{G}{V}$ hue is calculated as $H = 360^o - H$. At last hue is normalized to the range $[0,1]$ by letting $H = \frac{H}{360^o}$. It is also important to note that hue is not defined when saturation $S = 0$ as this corresponds to $R = G = B$.

$$S = 1 - \frac{3}{R+G+B} \cdot min(R,G,B) \tag{B.9}$$

Similar to hue, saturation is not defined when $V = 0$, which is obvious from the equations.

$$V = max(R,G,B) \tag{B.10}$$

Figure B.5 depicts the HSV color cone or space. As it can be seen $H = 0^o$ corresponds to red, $H = 120^o$ corresponds to green and $H = 240^o$ corresponds to blue. Value starts at zero at the bottom of the cone, which corresponds to black regardless of hue or saturation. With $V = 1$ the color is white if $S = 0$. If saturation is not zero the color is defined as the mixture of H, S and V.

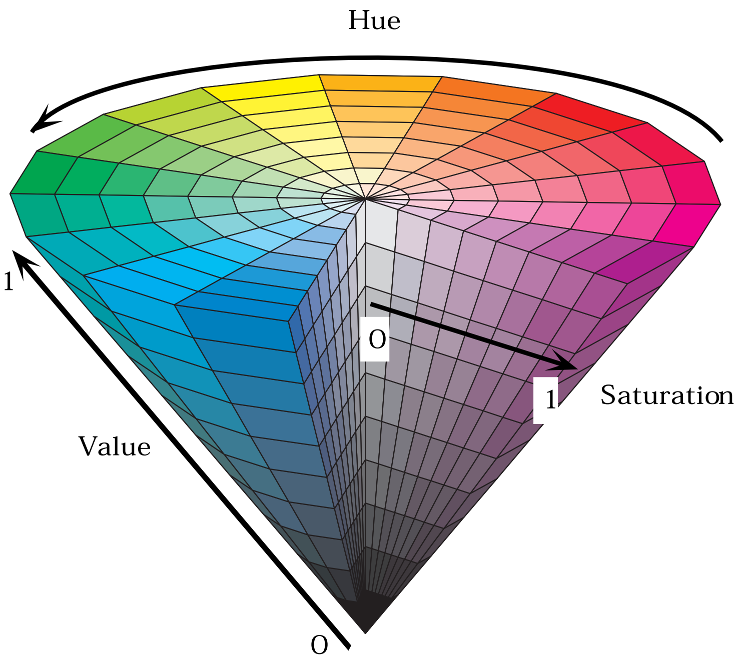


Figure B.5: The HSV color space. (mat, 1999)

The reason that the HSV color space is widely used, for example in image enhancement, is because the intensity or brightness is decoupled from the color information i.e. chromaticities (Gonzalez and Woods, 1993). Any monochrome or grey level enhancement technique can therefore be used on color images by applying these on the value channel.

The description of the HSV color space is based upon (Gonzalez and Woods, 1993) and the Matlab Image Toolbox (mat, 1999).

APPENDIX C

Segmentation Methods

This appendix will deal with various kinds of segmentation methods. Segmentation is the process where the image is subdivided into its objects. According to (Gonzalez and Woods, 1993) the segmentation process is one of the most difficult process in the image processing. Gonzalez and Woods (1993) claims that it is eventual the segmentation part of the system that determines the success or failure of the image analysis. The methods described in this appendix works on grey level images but the same methods applies to color images well. This appendix is based on (Gonzalez and Woods, 1993, chapter 7).

Segmentation is based upon two basic properties: discontinuity and similarity. The approaches for each of these properties are:

- Discontinuity
  - Point detection
  - Line detection
  - Edge detection
- Similarity
  - Thresholding
  - Quantization
  - Region-oriented
    - Region growing
    - Region splitting and merging

That is, segmentation based upon discontinuity detects isolated points, lines, or edges in an image and segmentation based upon similarity detects similar or equally valued regions evaluated upon some kind of criteria as for example the grey level.

C.1 Detection of Discontinuities

Detection of discontinuities is usually performed by running a 3×3 mask through the image. In the following subsections techniques for point, line and edge detection will be described.

C.1.1 Point Detection

Figure C.1 depicts a mask which can be used for detecting isolated points by stating that a point is detected if

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure C.1: Mask for detecting isolated points.

$$|R| > T \tag{C.1}$$

where T is a nonnegative threshold and R is given by

$$R = -1 \cdot z_1 - 1 \cdot z_2 - 1 \cdot z_3 - 1 \cdot z_4 + 8 \cdot z_5 - 1 \cdot z_6 - 1 \cdot z_7 - 1 \cdot z_8 - 1 \cdot z_9 \tag{C.2}$$

That is, R is the weighted sum of the point examined and its 8-connected neighbors.

C.1.2 Line Detection

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

+ 45º

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

2	-1	-1
-1	2	-1
-1	-1	2

- 45º

Figure C.2: Masks for detecting differently orientated lines.

Line detection can be utilized by the masks depicted in figure C.2. It can be easily verified that each mask responds strongly to one pixel wide lines in the direction of the 2's in the masks. Figure C.3 depicts an example of the result using the line detection masks.

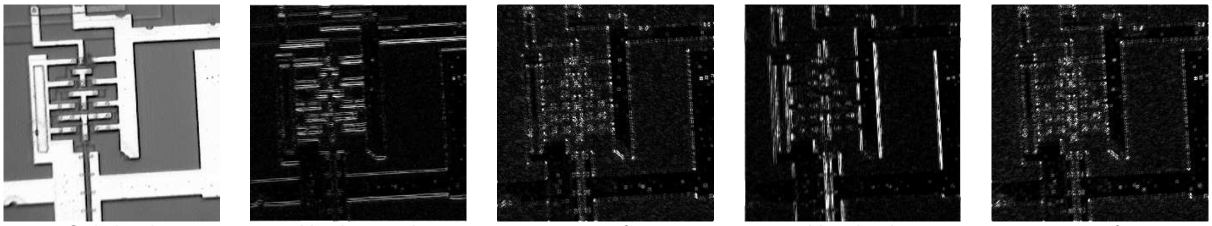


Figure C.3: Line detection masks applied to an image of an integrated circuit.

C.1.3 Edge Detection

An edge in an image is the boundary between two regions of distinct grey-level values. Segmentation based upon edge detection makes the assumption that regions are relatively homogeneous in their grey-level properties, so that edges indicate the presence of a boundary. Edges can be detected by calculating local derivatives in the image. One of these derivatives is the gradient.

The gradient of an image $f(x, y)$ at location (x, y) is the vector

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{C.3}$$

Edges can be found where the magnitude of the gradient is distinctively large since the magnitude indicates how sharp an edge is. The magnitude denoted ∇f, referred to simply as the gradient, is given by

$$\nabla f = \mathrm{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2} \tag{C.4}$$

To save time and resources this is commonly calculated as

$$\nabla f \approx |G_x| + |G_y| \tag{C.5}$$

Together with the magnitude the direction of the gradient is also an important indicator. The direction angle, denoted $\alpha(x, y)$, of the gradient at (x, y) is given by

$$\alpha\,(x, y) = \tan^{-1}\left(\frac{G_x}{G_y}\right) \tag{C.6}$$

The partial derivatives G_x and G_y can be calculated in digital form in several ways. One of these is the Sobel operators, which have the advantage of providing both a differencing and smoothing effect. The Sobel operator masks are depicted in figure C.4 and are calculated as

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{C.7}$$
$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{C.8}$$

where the gradient is calculated for the pixel at z_5.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

3 x 3 region

-1	-2	-1
0	0	0
1	2	1

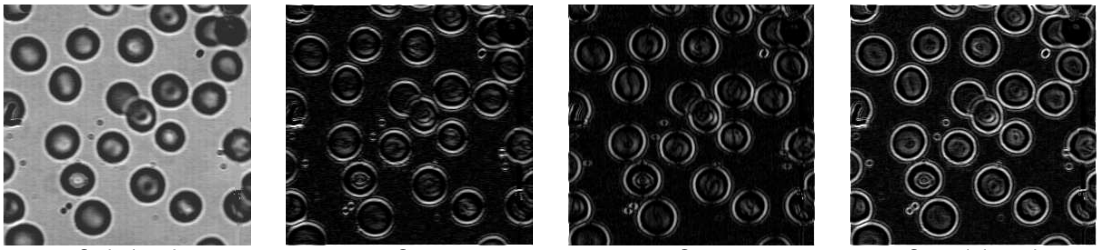
G_x

-1	0	1
-2	0	2
-1	0	1

G_y

Figure C.4: Masks for Sobel operators.

Figure C.5 depicts an example of an image where the gradients G_x and G_y have been computed and at last combined to form the gradient image. From this image the edges are clearly indicated.



Original G_x G_y Combined

Figure C.5: Sobel operators applied to an image of blood cells.

As it can be seen from figure C.5 the edges are clearly indicated, but do not seem connected. The following section shortly describes how edges can be linked.

C.1.4 Edge Linking

A simple method for linking edges together is to analyze the close neighborhood of all pixels in an image $f(x, y)$. If some points are similar they are linked to form a continuous boundary of pixels.

Similarity between pixels can be determined by examining either the magnitude or the angle. That is, for an edge pixel at (x', y') in the close neighborhood of (x, y) similarity in magnitude can be determined if

$$| \nabla f(x,y) - \nabla f(x',y') | \leq T \tag{C.9}$$

where T is a nonnegative threshold.

Similarity in angle can be determined if

$$| \alpha(x,y) - \alpha(x',y') | < A \tag{C.10}$$

where A is a nonnegative angle threshold.

A point is linked to another point if both criterias are satisfied. This process is repeated for every location in the image and a record of connected edges is kept to separate the individual edges in the image.

C.2 Thresholding

Thresholding is based upon the distribution of grey level values in an image, which consist of a background and objects. Images of this type usually have a grey-level histogram of the form shown in figure C.6, where the background is the dark mode and the objects are grouped in the bright mode. Given such a histogram a simple method to extract the objects is to select a threshold T_1 which separates the two modes. That is, any point in an image $f(x, y)$ is labeled an object if $f(x, y) > T_1$. This principle can be extended from a bimodal histogram to a multi-modal histogram using several thresholds to distinct the different modes. Figure C.6 depicts a tri-modal histogram with two thresholds T_1 and T_2. Multilevel thresholding is, however, less reliable and more complex and therefore seldom used.

A bimodal thresholding may be viewed as an operation that involves tests against a function T of the form

$$T = T\left[x, y, p(x,y), f(x,y)\right] \tag{C.11}$$

where $f(x, y)$ is the grey level of point (x, y), and $p(x, y)$ denotes some local property of this point. $p(x, y)$ could, for example, be the average grey level of the neighborhood around (x, y). A thresholded image $g(x, y)$ is defined as

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) \leq T \end{cases} \tag{C.12}$$

If the background is dark, pixels labeled 1 are objects and pixels labeled 0 is the background.

The threshold is called *global* if T only depends on $f(x, y)$ and *local* if T depends on $f(x, y)$ and $p(x, y)$. If T also depends on the spatial coordinates x and y, the threshold is called *dynamic*.

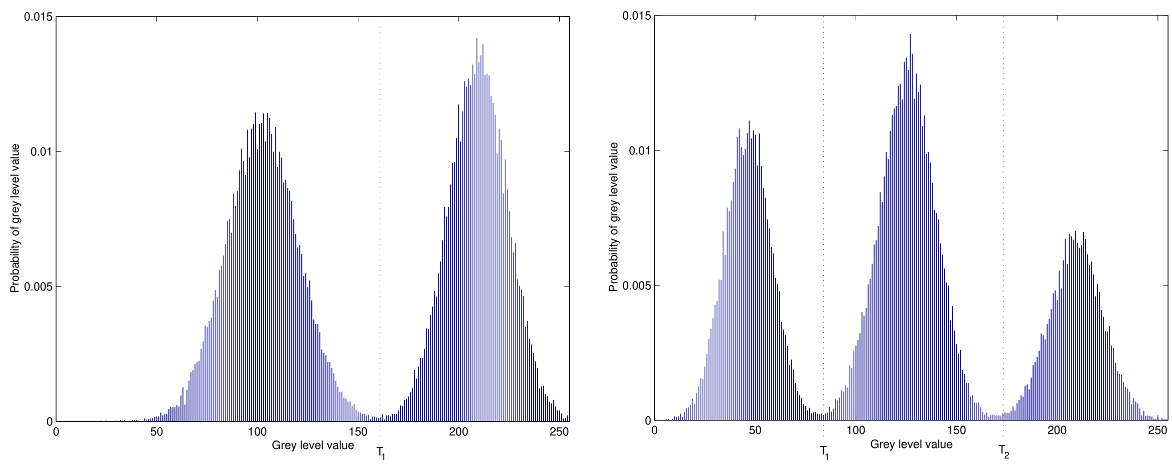


Figure C.6: Bimodal and tri-modal histograms of grey level images.

C.2.1 Simple Global Thresholding

Simple global thresholding uses a single global threshold T to partition the image. Figure C.7 depicts an image and its histogram. From the histogram a threshold $T = 110$ has been selected and applied to the image. Figure C.7 depicts the result.

Illumination plays a central role in the application of thresholding. Turn to (Gonzalez and Woods, 1993, p. 444-447) for further information on this subject.

C.3 Quantization

A technique which is actually a form of thresholding is quantization. Quantization is a many-to-one relation, which reduces the number of unique values in grey level in an image. For example, the grey level range [0..255] can be reduced to the range [0..31] by multiplying every grey level in the image by 31/255 and rounding the values to integers. This type of quantization is called *uniform quantization*.

As a segmentation technique each of the new values can be viewed upon as defining regions. That is, there may be several unconnected regions defined by each value. The technique corresponds to selecting several threshold equally spaced thereby defining different regions. It is, therefore, not based on the modal properties of the histogram. The technique is, however, not that different from multilevel thresholding.

Leppajarvi (1999) proposes a color quantization - merging algorithm for segmentation of color images, which uses a quantization method called optimal (minimum variance) quantization. After quantization the over-segmented image is merged to form more uniform and connected regions. The algorithm will not be described as this is out of the scope of this project.

C.4 Region-Oriented Segmentation

Region-oriented segmentation is based on finding regions directly. Two different techniques are commonly used for this purpose:

- Region growing
- Region splitting and merging

These two techniques will be described shortly after the definition of basic formulation.

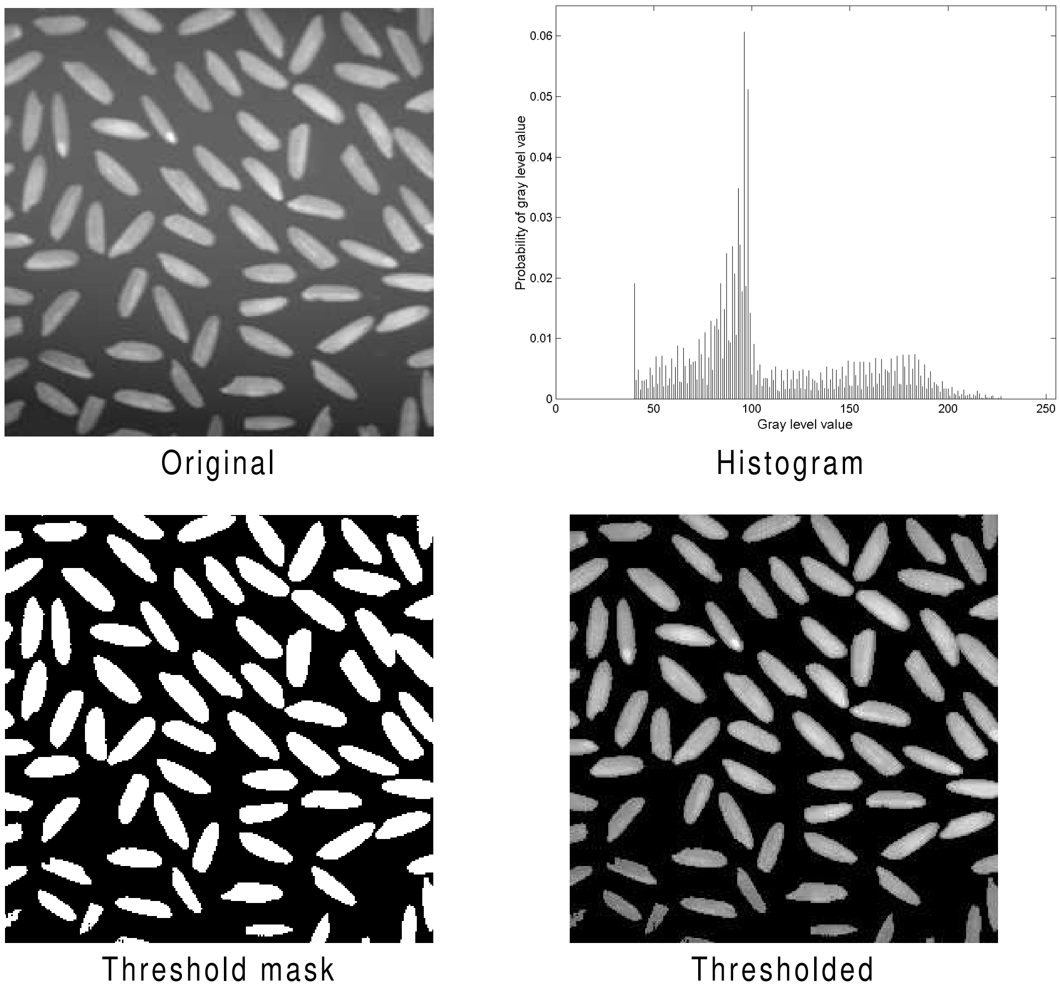


Figure C.7: Images of rice on a dark background segmented using thresholding.

C.4.1 Basic Formulation (Gonzalez and Woods, 1993, p. 458)

Let R represent the entire image region. Segmentation is then the process that partitions R into n subregions, R_1, R_2, ..., R_n, such that

1. $\bigcup_{i=1}^{n} R_i = R$,
2. R_i is a connected region, $i = 1, 2, ..., n$,
3. $R_i \cap R_j = \emptyset$ for all i and j, $i \neq j$,
4. $P(R_i) = \text{TRUE}$ for $i = 1, 2, ..., n$, and
5. $P(R_i \cup R_j) = \text{FALSE}$ for $i \neq j$,

where $P(R_i)$ is a logical predicate over the points in set R_i.

C.4.2 Region Growing

Region growing is a procedure that groups pixels recursively into larger regions until the growing predicate can no longer be satisfied. One simple method for region growing is pixel aggregation.

Pixel aggregation starts with selecting a set of "seed" points from which regions are grown by appending pixels that satisfy a predefined predicate. The predicate could be based on similarity in grey level, texture, color etc.

Figure C.8 depicts an example of pixel aggregation where the predicate for region growth is defined as pixels which only differ 10 percent in grey level with the seed point and which are 8-connected to at least one pixel previously included in the region. A single seed is selected (indicated as a black dot) and the growth of the procedure is depicted in consecutive frames.

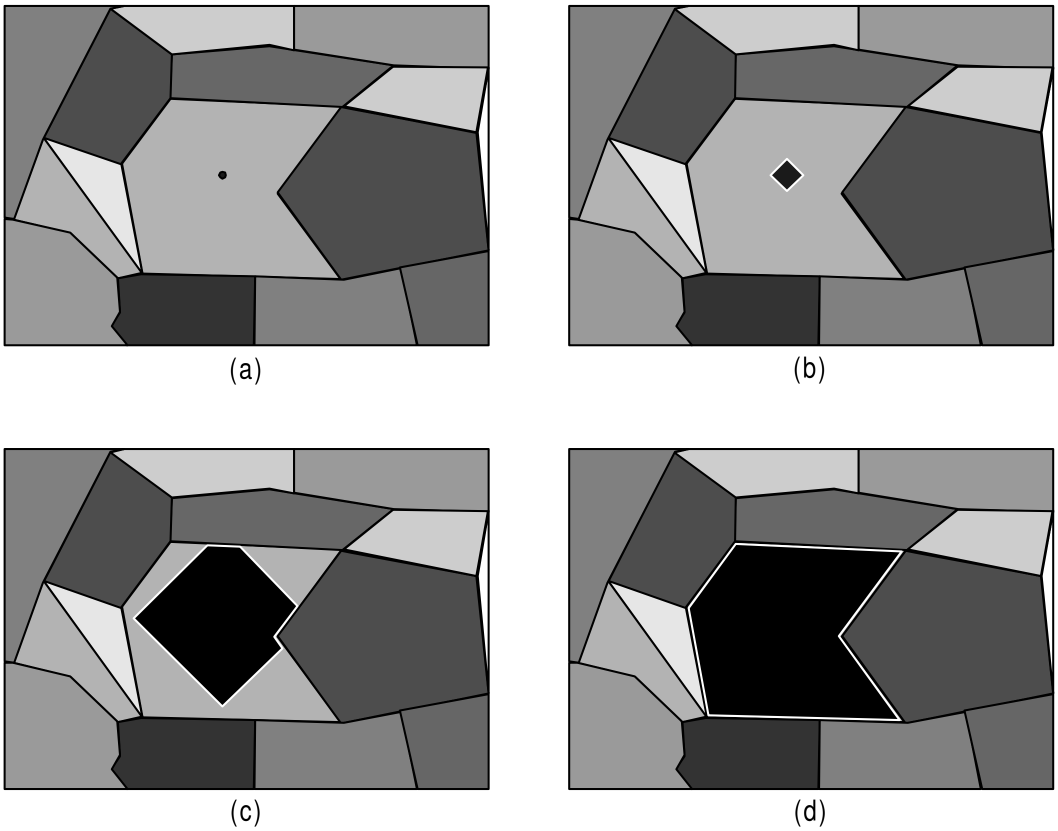


Figure C.8: Example of stages in region growing.

C.4.3 Region Splitting and Merging

Another commonly used technique is region splitting and merging, which initially subdivides an image into a set of arbitrary, disjointed regions and then merges and/or splits the regions in an attempt to satisfy the conditions stated in section C.4.1. One example of a split and merge algorithm for a square image is as follows:

1. split any region R_i into four quadrants where $P(R_i) = \text{FALSE}$.
2. merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k) = \text{TRUE}$.
3. stop when no further merging or splitting is possible.

Figure C.9 depicts an example using the described algorithm. The predicate $P(R_i)$ is TRUE if all pixels in R_i have the same grey level. Initially this is not the case and the region is split as shown in figure C.9(a). Now the top left region satisfies the predicate but the remaining three are split as shown in figure C.9(b). Several regions are now merged but two regions, containing the lower part of the object, do not satisfy the predicate and are split. Figure C.9(c) illustrates this. At last the regions are merged and two regions remain: the background and the object illustrated in figure C.9(d).

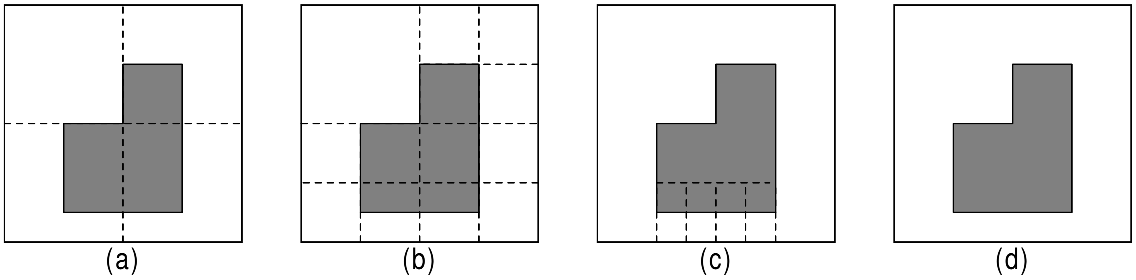


Figure C.9: Example of split-and-merge algorithm. (From (Gonzalez and Woods, 1993, p. 463).)

APPENDIX D

Stabilization Methods

This chapter deals with methods for stabilization. Stabilization is the task of determining ego-motion of the camera based on image data. Algorithms for solving the stabilization task can be subdivided into two subtasks:

- Locating correspondence between image points in consecutive images.
- Conversion of the correspondence into ego-motion of the camera.

Section D.1 and D.2 deals with methods for solving the correspondence problem, whereas section D.3 to D.6 deals with methods for conversion of the corresponding image points to ego-motion of the camera.

D.1 Window-based Correlation

The family of window-based correlation techniques belongs to the group of pixel based methods for solving the correspondence problem. Therefore it does not require any a priori information about the contents of the images.

D.1.1 Algorithm

The basic algorithm consists of two parts:

1. Select start image windows in frame i
2. For each window found search for the window in frame $i+1$ with the highest similarity.

The two parts of the algorithm will now be discussed.

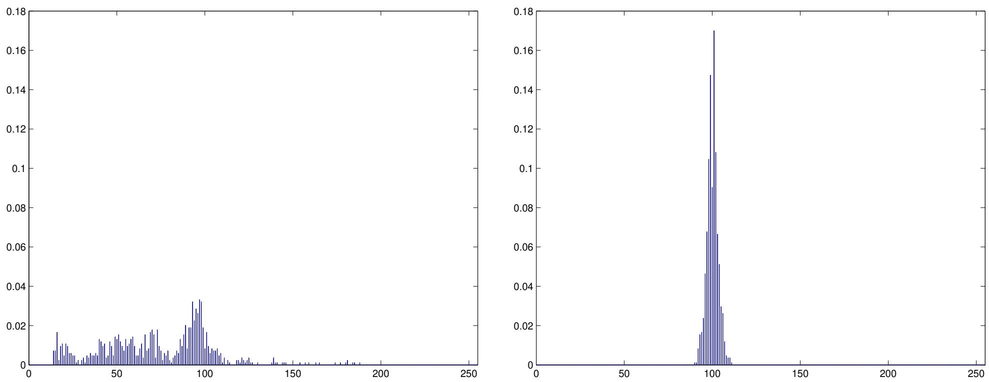
D.1.2 Selecting Start Windows

The performance of the window-based correlation algorithm highly depends on the technique used for selecting start windows. A 'good' start window must contain an unique texture in order for the algorithm to be able to locate the corresponding window in the next image. Figure D.1 shows an example of a good and a bad start window.

Figure D.1: An example of a good (green) and bad (red) start window

Several techniques exists for finding good windows. Examples of these are statistical methods and time derivative methods.

Statistical methods uses a statistical measure of the 'goodness' of a window. A possible choice of measure is the standard deviation (STD). Figure D.2 shows the histograms of the two windows in figure D.1. As it can be seen there is a significant difference in the distibution of the two windows. The good window has a STD of 31, whereas the STD of the bad window is only 3.



(a) (b)

Figure D.2: Histograms of the good (a) and bad (b) image window

Time derivative methods look at the difference in value of each pixel between two consecutive frames. The amount of change between a window in frame i to $i + 1$ is then used as a measure of the 'goodness' of the window.

Figure D.3 shows a difference picture with the two images windows. The average difference per pixel of the good window is 27.6, whereas it is 3.9 for the bad window.

Figure D.3: A difference picture with the good (green) and bad (red) start window

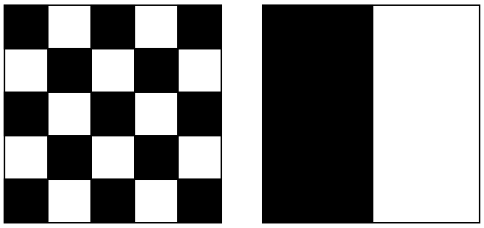
D.1.3 Searching for Best Match

After good start windows has been found, a method for searching for the best matching windows in the next frame is needed. This is done by searching through a region of interest in the next image, comparing the start window with the window of the current position using a similarity measure S. An often used similarity measure is one divided by the sum of squared difference (SSD) (Trucco and Verri, 1998, p. 147).

$$S = \frac{1}{\sum_{i=1}^{nelements}(winb[i] - wina[i])^2} \tag{D.1}$$

Where $wina$ and $winb$ are the windows to compare, and $nelements$ is the number of elements in both windows.

Depending on the application the similarity measure can either operate on the raw pixel data or on histograms of the windows. The advantage of using raw pixel data is that it can be used to differentiate between two windows with the same distribution of pixel values, but different structure as shown in figure D.4, whereas histograms cannot. On the other hand, a window of raw pixel data is not robust against noise and rotations of the image.



(a) (b)

Figure D.4: Checkerboard (a) and Half/half (b) image window with the same distribution of colors.

Another option is to use histograms of the image windows. This has the advantage that histograms are more robust against rotations than raw pixels. Furthermore it is computationally less expensive to calculate the similarity between histograms than raw pixels as the window size increases.

D.1.4 Summary

Window based correlation techniques selects windows of interest in the first image and searches for the best match in the second window using correlation. It does not require any a priori knowledge of the contents of the images. Several techniques exist to increase the accuracy and decrease the time complexity of the method.

D.2 Gradient Based Optical Flow Estimation

The gradient based method is an optical flow method, and therefore determines the motion (correspondence) for each pixel. Like other optical flow methods the gradient based method is most efficient if the displacement is small between consecutive frames. (Davies, 1997, p. 430)

The gradient based method uses the change in image brightness to estimate the motion between two consecutive images. It models the changes using partial differential equations.

A sequence of images sampled at a certain interval can be expressed as an image brightness function $I(x, y, t)$, which describes the intensity at point (x, y) in the picture at the discrete time t. The change in I over time can be expressed as

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} \tag{D.2}$$

assuming that I is continuous. It is assumed that the intensity does not change during small motion, which means that

$$\frac{dI}{dt} = 0 \tag{D.3}$$

Using this in D.2 we get the following:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{D.4}$$

This gives the velocity vector $\mathbf{v}$ for each point $\mathbf{x} = (x, y)$ in the image-plane at time t:

$$\mathbf{v} = \frac{d\mathbf{x}}{dt} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} \tag{D.5}$$

The equation has two unknowns, x and y, and only one equation, and is thus unsolvable. However, assuming that the velocity field varies smoothly over an image it is possible to iteratively determine the velocity vector using the following iterative equations:

$$\mathbf{x} = x_{average} - I_x\frac{P}{D} \tag{D.6}$$

$$\mathbf{y} = y_{average} - I_y\frac{P}{D} \tag{D.7}$$

where

$$P = I_x x_{average} + I_y y_{average} + I_t \tag{D.8}$$

and

$$D = \lambda^2 + I_x^2 + I_y^2 \tag{D.9}$$

where I_x, I_y, I_t, and λ is the spatial gradients in the x and y direction, the temporal gradient, and a constant multiplier, respectively.

In most cases the method works quite well. However, the assumption that there is a linear change in intensity cause a problem in some cases. Boundaries of objects commonly causes a change in depth and therefore invalidates the smoothness constraint. Furthermore, two objects moving towards each other will cause a problem, because the method is unable to handle the abrupt change in velocity field at the boundaries.

D.2.1 Summary

The gradient based method for estimation of optical flow determines the motion (correspondence) of each picture element in the image using the change in brightness of the image elements. The algorithm is only accurate in cases of small motion between images.

D.3 2D Affine Model

The 2D affine model is based on the observation that motion in the image plane caused by rotational ego-motion can be approximated by a 2D affine transformation of the rigid object described by the correspondence points. Figure D.5 shows the effect of rotational ego-motion in world space, and figure D.6 shows the corresponding 2D affine transformation of the correspondence points.

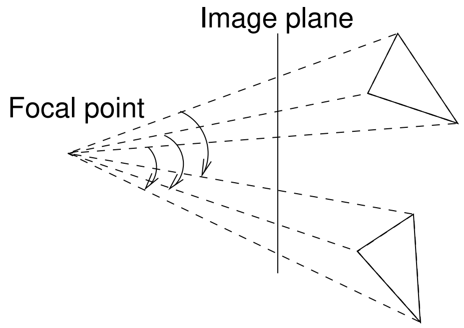


Figure D.5: Effect of rotational ego-motion in world-space.

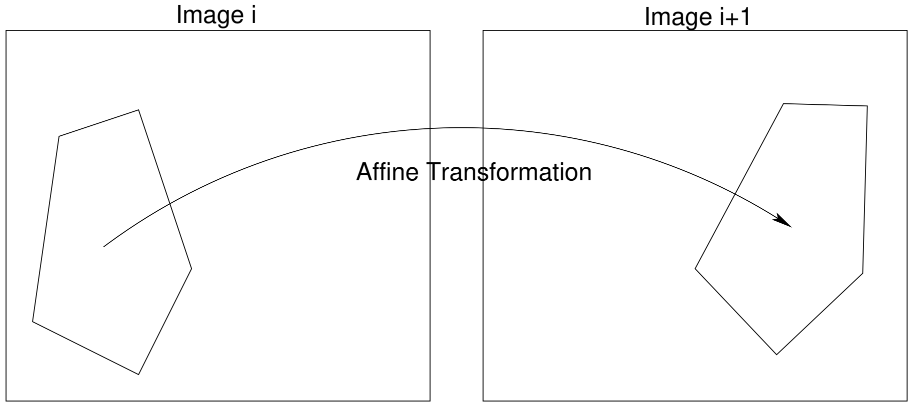


Figure D.6: The corresponding 2D affine transformation.

An affine transformation consists of a rotation, a translation and a scaling. The rotational part of the affine transformation corresponding to rotational ego-motion directly corresponds to the roll of the camera, whereas the translation in the x-direction corresponds to yaw, and the translation in the y-direction corresponds to pitch. The scaling does not correspond to any rotation, and is thus not considered.

There is a simple geometric relation between the translation and the pitch and yaw of the camera. The relation is:

$$angle = tan^{-1}(\frac{d}{f}) \tag{D.10}$$

Where d is the translation and f is the focal length.

D.3.1 Algorithm

The task at hand is to determine the 2D affine transformation $\mathbf{T}$, which describes the correspondence between image i and $i+1$. In homogeneous coordinates the transformation has the form:

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & T_x \\ sin(\theta) & cos(\theta) & T_y \\ 0 & 0 & 1 \end{bmatrix} \tag{D.11}$$

Applying the transformation, each corresponding point translates into two equations in 4 unknowns:

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & T_x \\ sin(\theta) & cos(\theta) & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{D.12}$$

Where (x, y) is the correspondence point in image i, (x', y') is the correspondence point in image $i + 1$, and $cos(\theta)$, $sin(\theta)$, T_x and T_y are the four unknowns.

As the two equations contains four unknowns, two correspondences are needed to determine the four unknowns. Ordering the four equations from two correspondences into a linear system of the form $\mathbf{A}\overline{X} = \overline{B}$ gives:

$$\begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ x_2 & -y_2 & 1 & 0 \\ x_2 & y_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) \\ sin(\theta) \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \end{bmatrix} \tag{D.13}$$

Where $\mathbf{A}$ is the points in image i, $\overline{X}$ is the four unknowns, and $\overline{B}$ is the points in image $i + 1$.

As the algorithms for solving the correspondence problem calculate more than two correspondences, the system is over-determined, and $\overline{X}$ can be found using least squares.

The correspondence coordinates used are relative to the center of rotation (roll).

D.3.2 Accuracy

The 2D affine model is only an approximation because the image plane is flat, and not a sphere section. Therefore the distance from the focal point to the center of the image plane is shorter than to the corner of the image plane. This has the effect that a point which is projected to the center of the image plane will have a smaller translation from a rotation of the camera θ degrees than a point which is projected to the corner of the image plane.

In order to calculate the error caused by this, an equation for the difference in translation near the center and near the corner must be derived. Figure D.7 and D.8 shows the two cases.

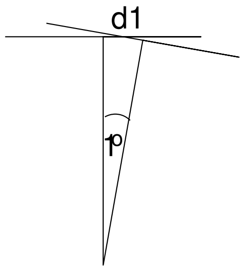


Figure D.7: Displacement of a point at the center of the image plane.

The translation of a point which is projected to the center of the image plane caused by a rotation of the camera 1° as a function of the FOV of the camera is:

$$d1 = f \cdot tan(1°) = \frac{tan(1°)}{tan(FOV)} \tag{D.14}$$

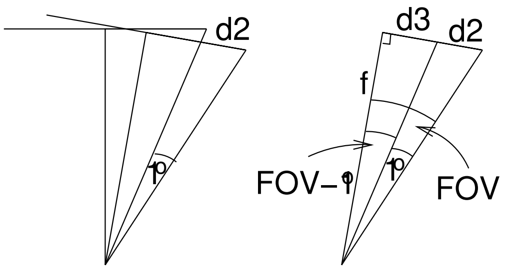


Figure D.8: Displacement of a point at the corner of the image plane.

Similar, the translation of a point which is projected to the corner of the image plane caused by a rotation of the camera 1° as a function of the FOV of the camera is:

$$d2 = 1 - d3 = f \cdot tan(FOV - 1°) = \frac{tan(FOV) - tan(FOV - 1°)}{tan(FOV)} \tag{D.15}$$

And finally the relation r between the two translations is:

$$r = \frac{tan(FOV) - tan(FOV - 1°)}{tan(1°)} \tag{D.16}$$

r is 1 when FOV is double of the rotation angle. r increases as FOV is increased. A common value for the FOV is 30°, which gives an relation r of 1.32.

D.3.3 Summary

The 2D affine model approximates rotational ego-motion by an affine transformation of the correspondence points in image i into the points in image $i + 1$. The 2D transformation can be found by solving an over-determined linear system of the form $\mathbf{A}\overline{X} = \overline{B}$ using least squares. Unfortunately least squares approximation is sensitive to outliers in the correspondences.

D.4 Hyperbolic Projection Method

Ego-motion consisting of pure pitch and yaw can be determined using an algorithm developed by (Burger and Bhanu, 1990). The description of the method follows the original description closely. Though, some issues which are not important in this context have been left out.

The method is based on mathematical considerations made using the camera model shown in figure D.9. In the model a 3D point $\mathbf{P} = (XYZ)^T$ is projected onto the image plane, where it is represented by the point $\mathbf{p} = (x, y, f)^T$, given by

$$x = \frac{fX}{Z} \tag{D.17}$$

and

$$y = \frac{fY}{Z} \tag{D.18}$$

where f is the focal length. By insertion we get the following representation of the point $\mathbf{p}$:

$$\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \cdot \begin{bmatrix} X \\ Y \end{bmatrix} \tag{D.19}$$

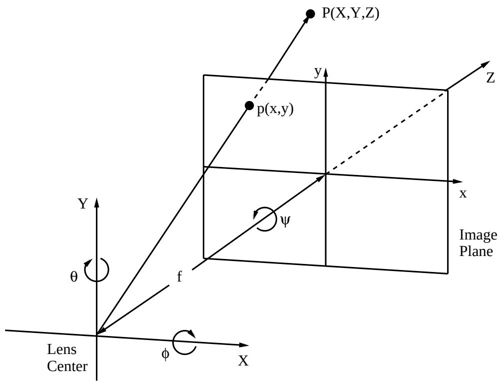


Figure D.9: Camera model showing the coordinate system, lens center, image plane, and angles of rotation. The origin of the coordinate system 0 is located at the lens center. The focal length is the distance between the lens center and the image plane (from (Burger and Bhanu, 1990))

A rotation R in 3D can be described by successive rotations about the three axis', where R_ϕ, R_θ, and R_ψ are the rotation about the X-, Y-, and Z-axis respectively:

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}, R_\theta = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}, R_\psi = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{D.20}$$

Any rotation of point **P** to point **P'** can be written as:

$$\mathbf{P'} = \mathbf{R} \cdot \mathbf{P} = R_\phi \cdot R_\theta \cdot R_\psi \cdot \mathbf{P} \tag{D.21}$$

The method by (Burger and Bhanu, 1990) only handles roll and pitch. Rotating about the X-axis and the Y-axis by $-\phi$ and $-\theta$, respectively, moves the 3D point from **P** to another 3D point **P'**:

$$\mathbf{P} \to \mathbf{P'} = R_\phi \cdot R_\theta \cdot P = \begin{bmatrix} cos\theta & 0 & sin\theta \\ sin\phi\ sin\theta & cos\theta & -sin\phi\ cos\theta \\ -cos\phi\ sin\theta & sin\phi & cos\phi\ cos\theta \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{D.22}$$

As the 3D point moves from **P** to **P'**, the point P on the image plane moves to P' given by:

$$\mathbf{p} \to \mathbf{p'} = \begin{bmatrix} x' \\ y' \end{bmatrix} = f \cdot \frac{\begin{bmatrix} X\ cos\theta + Z\ sin\theta \\ X\ sin\phi\ sin\theta + Y\ cos\phi - Z\ sin\phi\ cos\theta \end{bmatrix}}{-X\ cos\phi\ sin\theta + Y\ sin\phi + Z\ cos\phi\ cos\theta} \tag{D.23}$$

By inverting D.17 and D.18 and using them in equation D.22 a 2D rotation mapping $r_\phi r_\theta$ is made which moves image point $\mathbf{p} = (x, y)$ to the new image point $\mathbf{p'} = (x', y')$ during the 3D rotations $R_\phi R_\theta$:

$$R_\phi \cdot R_\theta(\mathbf{P}) : \mathbf{P} \to \mathbf{P'} \tag{D.24}$$

$$r_\phi r_\theta : x = (x, y) \to p' = (x', y') \tag{D.25}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = f \cdot \frac{\begin{bmatrix} x\ cos\theta + f\ sin\theta \\ x\ sin\phi\ sin\theta + y\ cos\phi - f\ sin\phi\ cos\theta \end{bmatrix}}{-x\ cos\phi\ sin\theta + y\ sin\phi + f\ cos\phi\ cos\theta} \tag{D.26}$$

Equation D.26 shows how to determine new pixel coordinates when the original image has been rotated by angles ϕ and θ.

The amount of rotation ϕ and θ which causes a point in the image plane to move from $\mathbf{x}_0$ to $\mathbf{x}_1$ can now be determined.

According to (Prazdny, 1981), horizontal rotation $\mathbf{R}_\phi$ and vertical rotation $\mathbf{R}_\theta$ will move along hyperbolic paths when applied separately. The horizontal rotation $\mathbf{R}_\phi$ moves a given image point $\mathbf{x}_0 = (x_0\ y_0)$ along a path described by the following equation:

$$r_\phi(x_0) : y^2 = y_0^2 \frac{f^2 + x^2}{f^2 + x_0^2} \tag{D.27}$$

The vertical rotation $\mathbf{R}_\theta$ moves the image point $\mathbf{x}_1 = (x_1\ y_1)$ along a path described by the following equation:

$$r_\theta(x_1) : x^2 = x_1^2 \frac{f^2 + x^2}{f^2 + y_1^2} \tag{D.28}$$

As well as the 3D rotation can be done uniquely in two ordered steps ($\mathbf{R}_\phi$ followed by $\mathbf{R}_\theta$), the rotation in the image plane can be separated into $\mathbf{r}_\phi$ followed by $\mathbf{r}_\theta$:

$$r_{\phi\theta} = r_\phi \cdot r_\theta \tag{D.29}$$

where

$$\mathbf{r}_\theta : \mathbf{x}_0 = (x_0 y_0) \rightarrow \mathbf{x}_c = (x_c y_c) \tag{D.30}$$

and

$$\mathbf{r}_\phi : \mathbf{x}_c = (x_c y_c) \rightarrow \mathbf{x}_1 = (x_1 y_1) \tag{D.31}$$

Figure D.10 shows how the hyperbolic paths going through $\mathbf{x}_0$ and $\mathbf{x}_1$ intersects in the point $\mathbf{x}_c$. The coordinates of this point can be calculated as:

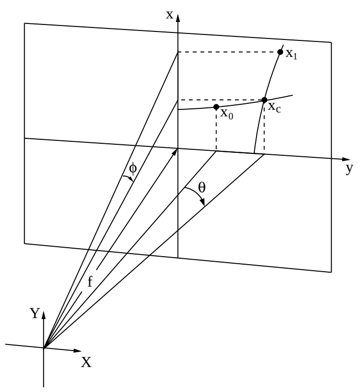


Figure D.10: Effects of camera rotation. Rotation about the Y-axis is applied first, which moves the arbitrary image point $\mathbf{x}_0$ along a hyperbolic path to $\mathbf{x}_c$. Subsequent rotation downwards about the $\mathbf{X}$-axis moves $\mathbf{x}_c$ to $\mathbf{x}_1$. The image location $\mathbf{x}_c$ is found by intersecting the two hyperbolae passing through $\mathbf{x}_0$ and $\mathbf{x}_1$ (from (Burger and Bhanu, 1990))

$$\begin{cases} x_c^2 = x_1^2 \frac{f^2+y_c^2}{f^2+y_1^2} \\ \\ y_c^2 = y_0^2 \frac{f^2+x_c^2}{f^2+x_0^2} \end{cases}$$
$$\Updownarrow$$
$$x_c^2 = x_1^2 \frac{f^2+y_0^2\frac{f^2+x_c^2}{f^2+x_0^2}}{f^2+y_1^2}$$
$$\Updownarrow$$
$$x_c^2 = x_1^2 \frac{f^2(f^2+x_0^2)+y_0^2(f^2+x_c^2)}{(f^2+x_0^2)(f^2+y_1^2)}$$
$$\Updownarrow$$
$$x_c^2 = x_1^2 \frac{f^4+f^2x_0^2+f^2y_0^2+x_c^2y_0^2}{(f^2+x_0^2)(f^2+y_1^2)}$$
$$\Updownarrow$$
$$x_c^2 = x_1^2 \frac{f^2(f^2+x_0^2+y_0^2)+x_c^2y_0^2}{(f^2+x_0^2)(f^2+y_1^2)}$$
$$\Updownarrow$$
$$x_c^2 = \frac{x_1^2f^2(f^2+x_0^2+y_0^2)}{(f^2+x_0^2)(f^2+y_1^2)} + x_c^2\frac{x_1^2y_0^2}{(f^2+x_0^2)(f^2+y_1^2)}$$
$$\Updownarrow$$
$$x_c^2\left(1-\frac{x_1^2y_0^2}{(f^2+x_0^2)(f^2+y_1^2)}\right) = x_1^2f^2\frac{f^2+x_0^2+y_0^2}{(f^2+x_0^2)(f^2+y_1^2)}$$
$$\Updownarrow$$
$$x_c^2\left(\frac{(f^2+x_0^2)(f^2+y_1^2)-x_1^2y_0^2}{(f^2+x_0^2)(f^2+y_1^2)}\right) = x_1^2f^2\frac{f^2+x_0^2+y_0^2}{(f^2+x_0^2)(f^2+y_1^2)}$$
$$\Updownarrow$$
$$x_c^2 = x_1^2f^2\frac{f^2+x_0^2+y_0^2}{(f^2+x_0^2)(f^2+y_1^2)-x_1^2y_0^2}$$

By taking the square root on both sides we get the expression for x_c:

$$x_c = f \cdot x_1 \cdot \sqrt{\frac{f^2+x_0^2+y_0^2}{(f^2+x_0^2)(f^2+y_1^2)-x_1^2y_0^2}} \tag{D.32}$$

The expression for y_c can be calculated using the same method, which give the following expression for y_c:

$$y_c = f \cdot y_0 \cdot \sqrt{\frac{f^2+x_1^2+y_1^2}{(f^2+x_0^2)(f^2+y_1^2)-x_1^2y_0^2}} \tag{D.33}$$

Using equations D.32 and D.33 it is possible to calculate rotation ϕ and θ as shown in figure D.10. Doing this leads to the following two equations for the desired rotations:

$$\phi = tan^{-1}\left(\frac{y_c}{f}\right) - tan^{-1}\left(\frac{y_1}{f}\right) \tag{D.34}$$

$$\theta = tan^{-1}\left(\frac{x_c}{f}\right) - tan^{-1}\left(\frac{x_0}{f}\right) \tag{D.35}$$

D.4.1 Summary

The hyperbolic projection model can be used to determine pitch and yaw provided that the egomotion does not contain any roll. The method is based on the fact that a rotation is projected to a hyperbolic path in the image plane.

D.5 Camera Rotation using Trilinear Tensor

The Trilinear Tensor Method is a technique for recovering rotational ego-motion. This appendix is based on the papers (Rousso, Avidan, Shashua, and Peleg, 1996), and (Shashua, 1995). Severals approaches to this problem exists. One is to compute the camera rotation after computing the translation of the camera (the epipole). The second approach assumes a specific 3D model. The third and the one describes in this appendix is called trilinear tensor. This method makes use of three images to make it more robust, and no 3D scene structure is assumed.

The Trilinear tensor is used for computing the three homography matrices needed to calculate the rotation of the camera. A homography matrix maps the image of a 3D plane in one frame into the image of the next frame. As the homographies do not correspond to any physical plane in the scene, there is no restriction to the scene composition.

This appendix is divided into three parts.

- The Homography Matrix
- Trilinear tensor
- Computing Small-angle Rotation

D.5.1 The Homography Matrix

A homography matrix maps a 3D plane between two images, and forms a linear space of rank 4. Figure D.11 illustrates the homography mapping of point p to p' using the 3D plane π.

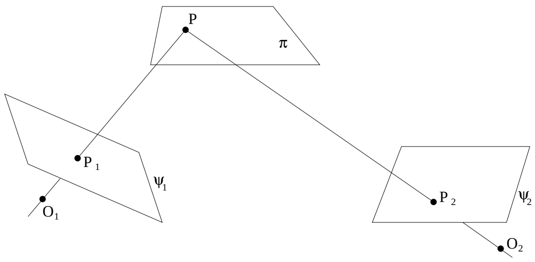


Figure D.11: The homography mapping of point $\overline{p}$ to $\overline{p}'$ using the 3D plane π on the images Ψ and Ψ', after (Rousso et al., 1996)

Let $\overline{P}$ be a point in the 3D-scene defined as $\overline{P} = (x, y, 1, 1/z)^T$, where x, y are the image coordinates in the first image, z is the depth of the point, and f is assumed to be 1.

The point P is projected on the first image using the transformation matrix M resolving in the image point $\overline{p} = (x, y, 1)$. The transformation matrix M is the intrinsic matrix of the camera.

$$\mathbf{M} = \begin{bmatrix} f_x & f_{xy} & x_o \\ 0 & f_y & y_o \\ 0 & 0 & 1 \end{bmatrix} \tag{D.36}$$

Point $\overline{P}$ in image 2 is defined as $\overline{p}' = (x', y', 1)$ and:

$$\overline{p}' \cong \mathbf{M}'\mathbf{R}\mathbf{M}^{-1}\overline{p} + \frac{1}{z}\mathbf{M}'\mathbf{T} \tag{D.37}$$

When the points $\overline{P}$ belongs to plane π, then $d_\pi = n_\pi^T(z\mathbf{M}^{-1}\overline{p})$, where $\overline{n}_\pi$ (normal vector) $d\pi$ (scalar) are the parameters of π in the first camera coordinate system. We obtain $\frac{1}{z} = \frac{1}{d\pi}\overline{n}_\pi^T(\mathbf{M}^{-1}\overline{p})$, and therefor,

$$\overline{p}' \cong \mathbf{M}'\mathbf{R}\mathbf{M}^{-1}\overline{p} + \frac{1}{d\pi}\mathbf{M}'T(\overline{n}_\pi^T\mathbf{M}^{-1}\overline{p}) \;=\; \mathbf{H}_\pi\overline{p} \tag{D.38}$$

The homography matrix associated with π is

$$\mathbf{H}_\pi \cong \mathbf{M}'(\mathbf{R} + \frac{\mathbf{T}\overline{n}^T}{d\pi})\mathbf{M}^{-1} \tag{D.39}$$

A homography matrix $\mathbf{H}_\pi$ is a transformation associated with Ψ and Ψ', and with the 3D plane π where Ψ and Ψ' is images. For any point in π, the homography matrix $\mathbf{H}_\pi$ maps $\overline{p}$ to $\overline{p}'$, see figure D.11.

D.5.2 Trilinear tensor

The trilinear tensor is an extension to the fundamental matrix of two images, an the point, and line geometry of three images. It describes the spatial relation of three cameras, or three consecutive images. The same point in three consecutive images is denoted as a triplet. Three consecutive camera matrices are denoted by $[\mathbf{I};\ \overline{0}], [\mathbf{A};\ \overline{v}'][\mathbf{B};\ \overline{v}'']$, where the left 3 x 3 minor is a homography matrix due to the (arbitrary) references plane and the fourth column is the epipole which is the projection of the center of projection of the first camera onto the second and third image planes, respectively.

Let $\overline{P}$ be a point in 3D projective space projecting onto $\overline{p}, \overline{p}', \overline{p}''$ in the two-dimensional projective space represented by the images Ψ, Ψ', Ψ''. The relationship between the 3D and 2D spaces is represented by the 3 x 4 camera matrices, i.e.,

$$\overline{p} = [\mathbf{I};\ \overline{0}]\overline{P} \;=\; (x, y, 1)^T \tag{D.40}$$

$$\overline{p}' \cong [\mathbf{A};\ \overline{v}']\overline{P} \;=\; (x', y', 1)^T \tag{D.41}$$

$$\overline{p}'' \cong [\mathbf{B};\ \overline{v}'']\overline{P} \;=\; (x'', y'', 1)^T \tag{D.42}$$

The trilinear tensor is an array of 27 entries:

$$\alpha_i^{jk} = \overline{v}'^k\overline{b}_i^j - \overline{v}''^j\overline{a}_i^k \qquad i, j, k = 1, 2, 3 \tag{D.43}$$

The covariant-contravariant summation convention is used as tensorial notation. A point is an object whose coordinate component is specified with superscripts, i.e., $\overline{p}^i = (p^1, p^2, ...)$. These are called contravariant vectors. An element in two-dimensional space (representing hyper-planes $\rightarrow$ lines in the 2D plane), is called a covariant vector and is represented by subscripts, i.e., $\overline{s}_j = (s_1, s_2, ...)$. Indices repeated in covariant and contravariant forms are summed over, i.e., $\overline{p}^i\overline{s}_i = \overline{p}^1\overline{s}_1 + \overline{p}^2\overline{s}_2 + ... + \overline{p}^n\overline{s}_n$. This is known as a contraction.

The tensor α_i^{jk} forms the set of coefficients of certain trilinear forms that vanish on any corresponding triplet $\overline{p}, \overline{p}', \overline{p}''$, i.e., functions of images that are invariant to object structure. These functions have the following form;

$$\mathbf{s}' \;=\; \begin{bmatrix} 1 & 0 & -x' \\ 0 & 1 & -y' \end{bmatrix} \tag{D.44}$$

and,

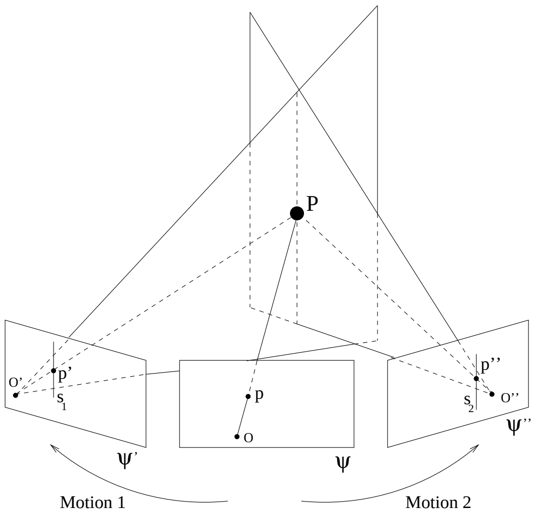


Figure D.12: The geometry of three views. Image Ψ, Ψ', and Ψ'' are obtained from cameras centered at $\overline{o}$, $\overline{o}'$, and $\overline{o}''$. The point $\overline{P}$ in the scene are projects to the image points $\overline{p}$, $\overline{p}'$, and $\overline{p}''$. $\overline{s}$ and $\overline{s}'$ are any image lines passing through the points $\overline{p}$, $\overline{p}'$, and $\overline{p}''$, respectively, after (Shashua, 2000)

$$\mathbf{s}'' = \begin{bmatrix} 1 & 0 & -x'' \\ 0 & 1 & -y'' \end{bmatrix} \tag{D.45}$$

Then the tensor equations are:

$$\mathbf{s}'_j \mathbf{s}''_k \overline{p}^i \alpha_i^{jk} = 0 \tag{D.46}$$

The equation D.46 in a more explicit form.

$$x''\alpha_i^{13}\overline{p}^i - x''x'\alpha_i^{33}\overline{p}^i + x'\alpha_i^{31}\overline{p}^i - \alpha_i^{11}\overline{p}^i = 0 \tag{D.47}$$

$$y''\alpha_i^{13}\overline{p}^i - y''x'\alpha_i^{33}\overline{p}^i + x'\alpha_i^{32}\overline{p}^i - \alpha_i^{12}\overline{p}^i = 0 \tag{D.48}$$

$$x''\alpha_i^{23}\overline{p}^i - x''y'\alpha_i^{33}\overline{p}^i + y'\alpha_i^{31}\overline{p}^i - \alpha_i^{21}\overline{p}^i = 0 \tag{D.49}$$

$$y''\alpha_i^{23}\overline{p}^i - y''y'\alpha_i^{33}\overline{p}^i + y'\alpha_i^{32}\overline{p}^i - \alpha_i^{22}\overline{p}^i = 0 \tag{D.50}$$

Every corresponding triplet contributes with four linear independent equations, then seven corresponding triplet is enough to determine the tensor α_i^{jk} (up to scale).

D.5.3 Computing Small-angle Rotation

According to (Rousso et al., 1996) is the skew-symmetric form of $\hat{R}$ unique. $\hat{R}$ is the approximated rotation matrix, and the approximated homography matrix of the plane at infinity. This can be expressed as

$$\hat{R} = \begin{bmatrix} 1 & \Omega_Z & -\Omega_Y \\ -\Omega_Z & 1 & \Omega_X \\ \Omega_Y & -\Omega_X & 1 \end{bmatrix} \tag{D.51}$$

As the small-angle rotation matrix is skew-symmetric, its elements can be spanned by three approximate homography matrices. In close form the equations for the three rotations are,

$$\Omega_X = det \begin{pmatrix} \alpha_2^{j3} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} / K \tag{D.52}$$

$$\Omega_Y = det \begin{pmatrix} -\alpha_1^{j3} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} / K \tag{D.53}$$

$$\Omega_Z = det \begin{pmatrix} \alpha_1^{j2} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} / K \tag{D.54}$$

$$K = det \begin{pmatrix} \alpha_2^{j2} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} / K \tag{D.55}$$

where α_2^{j2} stand for $(\alpha_2^{12}, \alpha_2^{22}, \alpha_2^{32})$, etc. This recovers directly small rotations $(\Omega_X, \Omega_Y, \Omega_Z)$ from trilinear tensor.

D.5.4 Summary

The trilinear Tensor Method is a technique for calculating all three rotations of the ego-motion using correspondences between three images. The method does not imply any constraints on the 3D structure of the environment. The method is only accurate in the case of small rotations.

D.6 Stabilization using Mosaic

Mosaicing can be used as an approach to estimation of the rotation and translation of the camera. This appendix is primary based on (Badra, Qumsieh, and Dudek, 1998). Normally mosaic is not used for this purpose, but used for reconstructing a scene from adjacent sub images, see figure D.13. A lot of effort has be put into the design of a robust algorithm for mosaicing with as few constraints as possible. Several of these other algorithms are mentioned in (Badra et al., 1998).

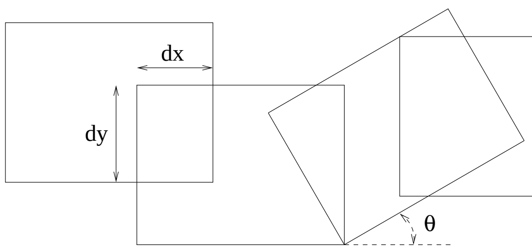


Figure D.13: The general problems of mosaicing. This figure illustrates lateral translation (left side), and rotation (right side). The problem concerning zooming is not illustrated, from (Badra et al., 1998)

The problem is to simultaneously solve lateral translation, rotation about any point, and zooming.

The following assumptions are made, according to (Badra et al., 1998), in order to propose a good solution.

- There is no warping of the image (no elastic deformation).
- The sub images do not exhibit a high level of repetitive similarity, for example, a sinusoidal or a checker board variation in intensity.
- Intensity in the image does not change arbitrarily, but there may be a change in overall contrast due to change in illumination or camera parameters.
- There in at least 10 % overlap between adjacent image.

The algorithm briefly described in this appendix is based on the paper (Badra et al., 1998). The algorithm makes use of Zernike moments, and offers a complete solution to the recovery of both rotational and scaling parameters without extensive computational correlation and search algorithms. The algorithm only requires a little overlap (10-20%) and handles arbitrary rotations.

The first assumption for a good solution (no warping of the image) is a problem for this algorithm, as well as for most of the other image mosaicing approaches. It would not work for higher order alignment models, such as affine or perspective.

Yet, the Zernike moments algorithm stands better than the other approaches mentioned in (Badra et al., 1998) toward the two cases.

- A camera mounted on a tripod. In this case, the camera is not allowed to change its position in space. It is only able to translate in any direction, and rotate about its vertical axis. Therefor, no warping could occur.
- A handheld camcorder in which the motion is slow and uniform. In this case, slight warping might occur, but can be safely neglected due to the large overlap between consecutive images.

D.6.1 Theory

The theory of this algorithm is briefly described. A more detailed description of the theory behind the algorithm can be found in (Badra et al., 1998).

Fourier Transform

Kuglin and Hines proposed a method for aligning two images shifted relative to each other. This method is called phase-correlation and operates in the Fourier domain. The method states that given two images f_1 and f_2 only exposed to a shift (dx, dy), their respective Fourier Transforms in polar coordinates can be found using;

$$F_1 = |F_1|e^{-i\phi_1} \tag{D.56}$$

$$F_2 = |F_2|e^{-i\phi_2} \tag{D.57}$$

Extracting the different of phase between the two images is done by multiplying F_1 by F_2^*, and dividing by the magnitude.

$$\triangle = \frac{|F_1||F_2|e^{-i(\phi_1-\phi_2)}}{|F_1||F_2|} \tag{D.58}$$

$$\triangle = e^{-i(\phi_1-\phi_2)} \tag{D.59}$$

Taking the inverse Fourier Transform of $\triangle$ results in a delta function whose peaks corresponds to the relative spatial shift between two images. There is still problems in the case of different scaling and large rotations. According to (Badra et al., 1998) it still gives good results in case of small rotations, when it is pure rotations. It has proven to be extremely accurate to within a single pixel.

Zernike Moments

Zernike moments belongs to a set of orthogonal polynomials, which allows independent moment invariance to be constructed easily. These moments have several advantages in terms of noise sensitivity, information redundancy, and image representation ability.

Zernike Polynomials constitute a complete set of complex polynomials that are orthogonal over the interior of the unit circle, $x^2 + y^2 = 1$. These polynomials are denoted as $V_{nm}(x, y)$, they can be expressed by a radial and an angular component.

$$V_{nm}(x,y) = V_{nm}(\rho,\theta) = R_{nm}(\rho)e^{im\theta} \tag{D.60}$$

where:

- n is a non-negative integer (the order of the Zernike moment).
- m is an integer obeying the constraints $n - |m|$ is even and $|m| \leq n$ (the repetition of the Zernike moment).
- ρ is the magnitude of the vector from the origin to the point $(x, y) = x^2 + y^2$.
- θ is the angle between the vector and the x-axis, in counterclockwise direction.

The Radial component $R_{nm}(\rho)$ is defined as

$$R_{nm}(\rho) = \sum_{s=0}^{\frac{n-|m|}{2}} \frac{(-1)^s (n-s)!}{s!\left(\frac{n+|m|}{2} - s\right)!\left(\frac{n-|m|}{2} - s\right)!}\rho^{n-2s} \tag{D.61}$$

Consider an image $f(x, y)$ in polar coordinates $f(\rho, \theta)$, and the rotated replica $f_r(\rho, \theta)$ rotated by an angle of α. The relation of two images are,

$$f_r(\rho,\theta) = f_r(\rho,\theta-\alpha) \tag{D.62}$$

Using the original images in polar coordinates, it is possible to express the Zernike moments as,

$$A_{nm} = \frac{n+1}{\pi}\int_0^{2\pi}\int_0^1 f(\rho,\theta)R_{nm}(\rho)e^{im\theta}\rho\ d\rho\ d\theta \tag{D.63}$$

The Zernike moments of the rotated image in polar coordinates are,

$$\begin{aligned} A^r_{nm} &= \frac{n+1}{\pi}\int_0^{2\pi}\int_0^1 f(\rho,\theta-\alpha)R_{nm}(\rho)e^{im\theta}\rho\ d\rho\ d\theta \\ &= A_{nm}e^{im\alpha} \end{aligned} \tag{D.64}$$

If the complex values of A_{nm} and A^r_{nm} could be computed, the relative angle of rotation α between the two images can be recovered by equation D.65

$$\angle\alpha = \frac{1}{m}\frac{A^r_{nm}}{A_{nm}} \tag{D.65}$$

By taking the ratio of the magnitude of A_{nm} and A^r_{nm}, the relative scale s of the two images can be approximated to within a reasonable degree of accuracy. s is approximated using equation D.66,

$$s = \frac{|A_{nm}|}{|A^r_{nm}|} \tag{D.66}$$

Algorithm for Estimation of Rotation and Translation

The approach described in this appendix is based on selecting a preliminary correspondence, and then refining it using the Zernike moment computation. Due to the recursive nature of the algorithm, Zernike moments are robust enough to give accurate results even when there a difference between the corresponding points in the two images.

The approach is summarized in the following pseudo-code, followed by comments.

1. Select potential correspondences using interest operator, define points of points p_1, and p'_1.
2. Calculate initial estimate $t_0 = (p_{1x}$ - p'_{1x} , p_{1y} - $p'_{1y})$.
3. Compute Zernike moments about p_1, and p'_1.
4. Estimate R_n, and S_n.
5. Align images.
6. Compute t_{n+1} using FFT.
7. If $|t_{n+1} - t_n| < \delta \ \rightarrow$ END, else return to step 3.

The algorithm is based on corresponding points in the two consecutive images. A disk of predefined radius is mapped around each of the points, and the Zenike moments within each disk are computed. Initial estimates, R_0 and S_0, of the relative rotation and scaling of the two consecutive images are then calculated. This allow one of the images to be brought into correspondence with the other by applying the suitable scaling and rotation. Given this correction for scale and rotation, it is possible to refine the translation estimation t_0. The process is repeated on the corrected images. This is done until the two images align perfectly, or until the alignment error is below a threshold δ. The number if iterations and the resolution of the images is a trade off for speed.

D.6.2 Summary

The Mosaic Method is a technique for recovering all three rotations of the ego-motion. The method does only require little overlap between the images, but it is computationally heavy.

APPENDIX E

Feature Matching Methods

Feature matching is concerned with classifying a set of unknown feature vectors. This is typically done by computing the minimum distance in feature space between all combinations of known vectors (often called "classes") and the unknown vectors. The computation results in a matrix with real, scalar elements representing the correspondance between two vectors.

Definition: given an $m \times p$ data matrix, $\mathbf{A}$, and an $n \times p$ data matrix $\mathbf{B}$, we compute a distance matrix $\mathbf{D}$. m and n equals the number of samples (row feature vectors) and p corresponds to the actual number of features. For row distances, the $\mathbf{D}_{ij}$ element of the distance matrix is the distance between vector $\mathbf{A}_i$ and $\mathbf{B}_j$, which results in a $m \times n$ $\mathbf{D}$ matrix.

E.1 Distance Functions

A selection of distance functions from (Wilson and Martinez, 1997) and (Heckbert, 2000) are defined below.

E.1.1 Minkowsky Distance

The Minkowsky distance is defined as

$$\mathbf{D}_{ij} = \left(\sum_{k=1}^{p} |(\mathbf{A}_{ik} - \mathbf{B}_{jk})|^{p} \right)^{\frac{1}{p}} \tag{E.1}$$

The Minkowsky distance is the pth root of the sum of the absolute differences to the pth power between corresponding elements.

E.1.2 Euclidian Distance

The Euclidean distance is presented below.

$$\mathbf{D}_{ij} = \sqrt{\sum_{k=1}^{p} (\mathbf{A}_{ik} - \mathbf{B}_{jk})^2} \tag{E.2}$$

where the summation is relative to k over columns 1 to p.

The Euclidean distance is simply the square root of the squared differences between corresponding elements of the rows (or columns). This is probably the most commonly used distance metric.

The Euclidean distance is actually a special case of Minkowsky distance for $p = 2$.

E.1.3 Mahalanobis Distance

The Mahalanobis distance is defined as

$$\mathbf{D}_{ij} = \sqrt{(\overline{a}_i - \overline{b}_j)\mathbf{C}^{-1}(\overline{a}_i - \overline{b}_j)^T} \tag{E.3}$$

where $\mathbf{C}^{-1}$ is the inverse of the feature-covariance matrix of vectors from $\mathbf{A}$ and $\mathbf{B}$. The distances are obtained by letting $\overline{a}_i$ and $\overline{b}_j$ represent the ith and jth row vector of $\mathbf{A}$ and $\mathbf{B}$ respectively.

Wilson and Martinez (1997) multiplies $\mathbf{D}_{ij}$ by $\det(\mathbf{C})^{\frac{1}{p}}$, instead of calculating the square root of the inner expression, in order to obtain the mathematical correct normalized result.

The Mahalanobis distance is effectively a weighted Euclidean distance where the weighting is determined by the sample variance-covariance matrix.

E.1.4 Block Distance

The block distance is defined as

$$\mathbf{D}_{ij} = \sum_{k=1}^{p} |(\mathbf{A}_{ik} - \mathbf{B}_{jk})| \tag{E.4}$$

The block distance is the sum of the absolute differences between corresponding elements of the rows. Note that this is a special case of the Minkowsky distance with $p = 1$.

The block distance is also known as the city block or Manhattan distance.

E.1.5 Chebychev Distance

The Chebychev distance is defined as

$$\mathbf{D}_{ij} = \max_{k=1}^{p} |(\mathbf{A}_{ik} - \mathbf{B}_{jk})| \tag{E.5}$$

The Chebychev distance is the maximum absolute difference between corresponding elements.

E.1.6 Chi-square Distance

The Chi-square (χ^2) distance is defined as

$$\mathbf{D}_{ij} = \sum_{k=1}^{p} \frac{1}{sum_k} \left(\frac{\mathbf{A}_{ik}}{size_{\overline{a}_i}} - \frac{\mathbf{B}_{jk}}{size_{\overline{b}_j}} \right)^2 \tag{E.6}$$

sum_k is the sum of column k for all vectors from $\mathbf{A}$ and $\mathbf{B}$. $size_{\overline{a}_i}$ is the row sum of the $\overline{a}_i$th vector, analogous for $size_{\overline{b}_j}$.

The chi-square distance, as opposed to the Euclidian distance metric, somewhat compensates for feature variance differences.

APPENDIX F

Prediction Methods

Many computer vision applications involve repeated predicting (or estimating) of the system quantities that change over time, i.e. spatio-temporal tracking. These dynamic quantities are called the system *state*. The system state here is represented by a vector. In addtion to this output vector from the *predictor*, there is an input vector containing *measurements* from sensor data (from e.g. feature extraction). The system state can also be modelled as a stochastic process.

This chapter deals with methods for predicting the state of the system at time t_k typically based on the state at previous discrete time instances t_{k-1}, t_{k-2}, ..., t_{k-p}. For the hybrid approaches in section F.2 the estimations are partly based on a system state model.

F.1 Data Driven Prediction

By assuming smooth state transition (monotonic, predictable behaviour), adequate sampling and low noise it is possible to predict the state by methods based only on past measurement data. Statistical and linear algebra methods can be applied for determining the optimal fit for a given data set. From the best fit regression analytical parameters are extracted, which can then be used for extrapolation/prediction.

The n degree polynomial fitting and extrapolation technique for prediction is described below. A particularly simple example of this fitting method is simple linear extrapolation, which is briefly discussed thereafter.

F.1.1 Polynomial Fit and Extrapolation

Fitting a n degree polynomial curve to a set consisting of m ($m = p+1$) data points is essentially a least-squares problem. The calculations are performed independently for each quantity in the state vector. Curve fitting is thus a 2D problem with (x, y) data points for each sample consisting of $(time, value)$. The overall goal of the fitting process is to minimize the squared error (e.g. from noise), and obtain the appropriate analytical parameters for the polynomial. These parameters can then be used for extrapolating the curve, i.e. prediction. Figure F.1 shows a simple example of linear fitting data points to a curve.

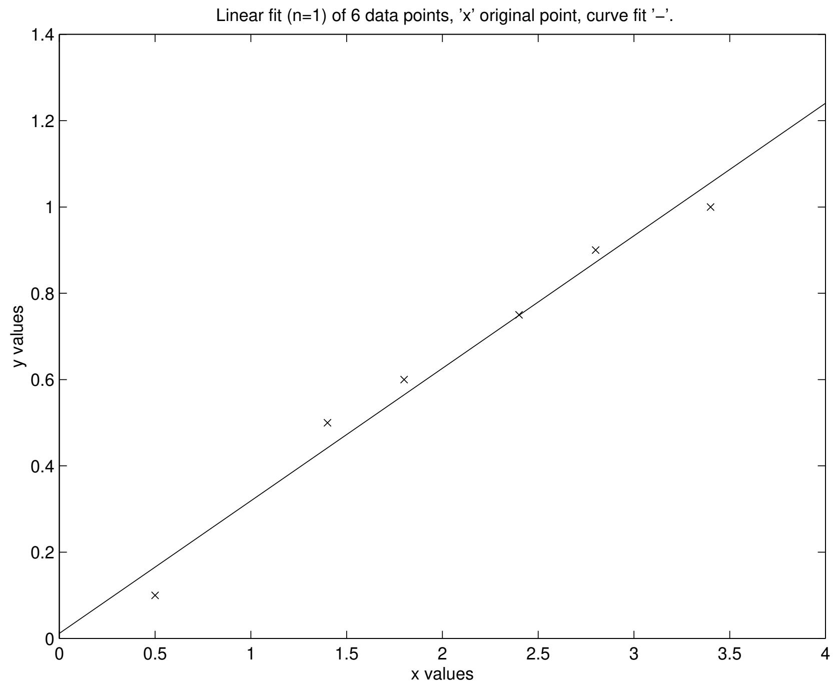


Figure F.1: Example of polynomial curve fitting.

Polynomial Fitting

Statistical methods exists for solving the regression problem (Shanmugan and Breipohl, 1988, pp. 529-534), however, the generalized linear algebraic approach from (Strang, 1998) shown below is much more elegant.

$\overline{t}$ is a column vector representing the sequence of m ordered, discrete time stamps (sometimes called a "controlled" stochastic variable), and $\overline{b}$ the corresponding data values (the "dependent" random variable). $\overline{x}$ contains the n paramters for the polynomial (optimal solution case), and $\mathbf{A}$ is the $m \times n$ matrix defined in equation F.1.

$$\mathbf{A}\overline{x} = \overline{b} \text{ is } \begin{array}{c} c_0 + c_1t_1 + c_2t_1^2 + \ldots + c_nt_1^n = \overline{b}_1 \\ c_0 + c_1t_2 + c_2t_2^2 + \ldots + c_nt_2^n = \overline{b}_2 \\ \vdots \\ c_0 + c_1t_m + c_2t_m^2 + \ldots + c_nt_m^n = \overline{b}_m \end{array} \text{ and } \mathbf{A} = \begin{bmatrix} 1 & t_1 & t_1^2 & \ldots & t_1^n \\ 1 & t_2 & t_2^2 & \ldots & t_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \ldots & t_m^n \end{bmatrix} \quad \text{(F.1)}$$

The equation $\mathbf{A}^T\mathbf{A}\hat{x} = \mathbf{A}^T\overline{b}$ gives $\hat{x} = (c_0, c_1, ..., c_n)$, which is the best fit for $m > n$ (with several intermediate equations omitted). $\hat{x}$ contains the n least-squares approximated parameters for the polynomial.

Provided $\mathbf{A}^T\mathbf{A}$ is invertible, solving for $\hat{x}$ then gives

$$\hat{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\overline{b} \quad \text{(F.2)}$$

For $m < n$ the solution is

$$\hat{x} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\overline{b} \quad \text{(F.3)}$$

Alternatively, the singular value decomposition (SVD) can be applied to ensure that the optimal solution is always obtained, regardless of singularity problems and the size of $\mathbf{A}$.

Extrapolation

Now that the best, analytical parameters for the polynomial has been found, the future value at time u, $y(u)$, can be predicted by using equation F.4.

$$y(u) = c_0 + c_1 u + c_2 u^2 + \ldots + c_n u^n \tag{F.4}$$

Figure F.2 shows an example of predicting a future value by extrapolating.

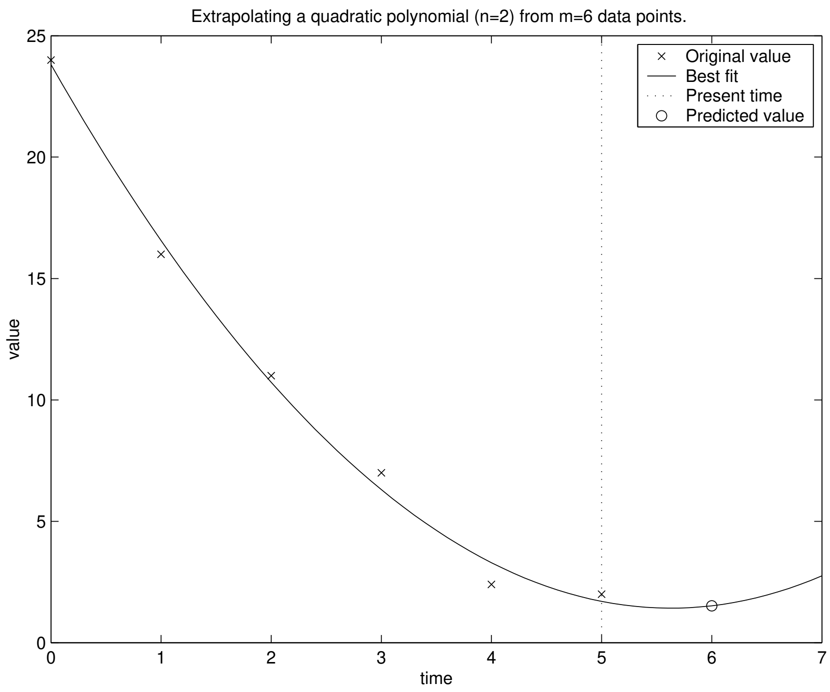


Figure F.2: Example of extrapolating a quadratic polynomial and predicting the next value.

Concluding Remarks

The choice of n depends on the application, but feasible n-values will typically be in the range $[1-3]$. If n is too high the risk of fitting the noise becomes inherent, which is undesirable. However, if n is too low it is impossible to capture unlinearities. To maximize performance n could possibly differ for the quantities in the state vector.

F.1.2 Minimal Memory, Linear Extrapolation

A special case of polynomial fitting and extrapolation method described above for $n = 1$ and $m = 2$ could be called "minimal memory, linear extrapolation". Based on just the current and previous measurements the future state is determined. The attractiveness of this method is its simplicity and speed, since it does not require any matrix calculations, just basic scalar operations. Equation F.5 shows how to predict the next value $y(u)$ from just current time and measurement (t_2, b_2), and the previous state (t_1, b_1).

$$y(u) = \frac{u - t_1}{t_2 - t_1} * (b_2 - b_1) + b_1 \tag{F.5}$$

Obviously this approach is sensitive to noise, but is conversely also more responsive to abrupt changes, compared to predictors which estimates using several samples.

F.2 Hybrid Model/Data Driven Prediction

As opposed to the solely data driven approached mentioned above, hydrid model/data prediction uses a state model of the observed signal process. This state model can include kinematic parameters like position, velocity and acceleration (Birk et al., 1996, p. 57). The state model is based on a statistical model of the signal process.

Below two approaches, Kalman filtering and Condensation, are briefly discussed without any equations, as a description of their mathematical background is outside the scope of this report.

F.2.1 Kalman Filter

The Kalman filter is essentially a set of mathematical equations that implement a linear predictor-corrector estimator that is *optimal* in the sense that it minimizes the estimated error covariance. It is important to note that Kalman filtering assumes that the noise is all zero-mean Gaussian, and white (Shanmugan and Breipohl, 1988). Even if the Gaussian assumption is not satisfied it can be shown that the Kalman filter is the best linear predictor for non-linear system (Birk et al., 1996, p. 58).

Its ability to incorporate the effects of noise (from both measurement and modelling), and its computational structure, has made it very popular for use in computer vision tracking applications.

The discrete Kalman Filter is an iterative procedure (figure F.3) containing several elements which are described hereafter.

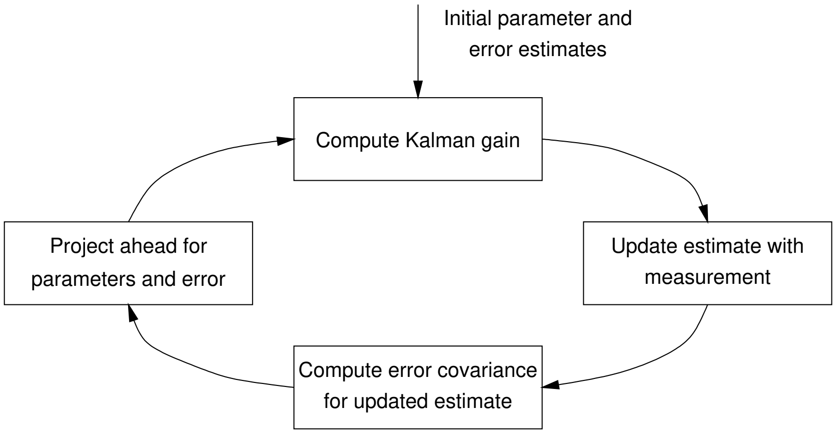


Figure F.3: Kalman filter loop from (Hosie, 1995).

The filter is supplied with initial information, including the measurement error covariance, and estimates of the initial parameters and associated error, and these are used to calculate a gain matrix. The error between the parameter estimates and the measured data is determined and multiplied by the gain matrix to update the parameter estimate and estimated error. The updated error and parameters are used as input to a model of behaviour to predict the projected error and parameters at the next time instance (Hosie, 1995).

Initially, when the model parameters are only rough estimates, the gain matrix ensures that the measurement data is highly influential in estimating the state parameters. Then, as confidence in the accuracy of the parameters grows with each iteration, the gain matrix values decrease, causing the influence of the measurement data in updating the parameters and associated error to lessen.

For the sake of efficiency the filter is typically implemented using a recursive approach, so that any prediction takes the same time to perform, and the complete minimization process is thus avoided for every new measurement.

F.2.2 Condensation

For some problems the "CONDENSATION algorithm" (**Con**ditional **Dens**ity Propag**ation**) has been shown to be outperform the Kalman filter. This is particularly true for tracking curves in dense visual clutter (Isard and Blake, 1998). In this case Kalman filtering is inadequate because it is based on Gaussian decisions which, being unimodal, cannot represent simultaneous alternative hypotheses. The Condensation algortihm can, however, be applied for any type of prediction other than curves.

The Condensation algorithm uses stochastic, factored sampling, previously applied to the interpretation of static images, in which the probability distribution of possible interpretations is represented by a randomly generated set. Condensation uses learned dynamical models (typically obtained by Kalman tracking in an environment that permits simple segmentation), together with measurements, to propagate the random set over time. The result is highly robust tracking of agile motion, and moreover the algorithm runs in real-time (Isard and Blake, 1998).

The original algorithm was designed for tracking single objects. Recently Koller-Meier and Ade (2001) have presented an improved version of the Condensation algorithm for tracking multiple contours in a cluttered enviroment.