Illumination-Invariant Object Segmentation

Towards an NPR Video Art Application

M.Sc. thesis Intelligent Systems
Wicher Visser

Supervisor
Nicolai Petkov

August 5, 2007

Abstract

Common generic segmentation methods are obstructed by sudden changes in illumination. Significant increase of brightness by light switching on and shadows cast by objects often cause these methods to produce erroneous classifications.

To enable illumination-invariant segmentation, the Collinear Vector Model discussed in this thesis constructs RGB color vectors from local pixel neighborhoods. Variations in brightness only influence the length of these vectors by a scalar value. Therefore an orthogonal distance measure can be employed to determine the local color similarity under illumination invariance. In the presence of additive noise, the vector collinearity is estimated by finding the minimum orthogonal distances from the vectors to the unknown noise-free signal.

The distance minimization can be defined as a smallest eigenvalue problem. This minimum is incorporated into a Bayesian framework, which allows for maximization of the a-posteriori probability (MAP) of the decision. The resulting value is compared against a static and an adaptive threshold. The classification labels are considered to be sampled by a Markov random field (MRF) to model the pixel interdependencies. The corresponding energy function is defined as the integration of the evidence over a spatial neighborhood. This induces spatial compactness and smooth edges in the foreground mask. Performance is measured using both the PETS 2001 dataset and a specific illumination test set.

The Collinear Vector Model is used in an interactive video art application. Therefore, the second focus of this work discusses the academic interest in art. One of the many purposes of art is to attract and stimulate the curiosity of people. To this end a video distorting mirror has been devised that produces time-delayed 'floating' copies of objects in movement. The Collinear Vector Model replaces static parts of the scene to emphasize this motion. The proposed *Pool of Intentions* algorithm will be deployed as part of the Science LinX program of the University of Groningen to express the creativity and transparency of academic research.

Preface

The thesis in front of you is the final work of my M.Sc. graduation project, part of the study Computing Science at the University of Groningen. The research described in this thesis has been carried out in the domain of Image Analysis, which is part of the research programme Intelligent Systems. It serves as documentation of my study, which has been conducted from November 2006 to August 2007.

The main focus of this thesis is an object segmentation algorithm for the purpose of video filtering and tracking, designed in the context of illumination invariance. Segmentation performance has been tested on the dedicated and publicly available PETS 2001 dataset. A second experiment shows the quality of segmentation in the context of strong and rapid illumination changes. The separation of background from foreground is used to replace the static regions from an interactive video filter by a fixed background. This video filter produces an artistic effect that appeals to the curiosity and attention of the spectator will be deployed as part of the Science Linx project of the University of Groningen.

The elaborate overview of the literature presented in this thesis could be extended to all the documents explored by the author. It is only for the restriction of time and pages that this has not been addressed in full. Likewise, most of the algorithms developed on the basis of the literature have been omitted for the sake of brevity.

Acknowledgements

It is a pleasure to thank the many people who made this thesis possible.

Firstly, I would like to express my gratitude to professor Nicolai Petkov, whose guidance and inspiration provided encouragement, sound advice and good ideas. The freedom of research he granted was both instructive and stimulating. Due to him research is a joy.

Thanks go out to Andreas Griesser, Ph.D. student at the ETH Zürich, for his eagerness to share knowledge on his illumination-invariant classification method and supplying his source code and test data. I am looking forward to join him as a colleague!

Kurt Lust, assistant professor in mathematics a the University of Groningen, clarified a few things regarding matrix theory. His explanations have been a great help.

For the development and experimentation three computer systems have been provided by Jurjen Bokma, member of the computer system support of the Institute of Mathematics and Computing Science. I appreciate his help and patience in setting up the Linux systems. His enthusiastic assistance finding a suitable digital camera was more than I would ever had expected.

A special token of gratitude to Binne Simonidis, good friend and fellow student. His working spirit and dedication has been a catalyst to my own attitude. The thorough comments on my thesis were a good help. Furthermore, his urge for cigarettes and coffee provided excellent opportunities to reflect my thoughts and discharge any troubles of mind. The evenings sessions of Battlestar Gallactica and Heroes served as a pleasant distraction.

I would like to thank, in order of appearance, Laurens van der Starre, Fabio Bracci, Jens Rasmussen, Alex Haan, Laura van Achteren, Barend van de Wal, Marieke Rumpff, Harmke Duynisveld and everybody else I forget to mention (sorry!) for the hilarious laughs, hours of coffee breaks and good company. Without their presence writing my thesis would have been dull. I will miss you all!

Lastly, and most importantly, my appreciation to my parents, Tjalling Visser and Tine Visser. Their faith and patience appeared infinite despite my communicational conciseness ("Het gaat goed.") and was a magnificent support. Tjalling, your industrious attitude has always been my example. Tine, the interest you continuously showed provided the energy to conclude this work.

Many thanks to you all!

Contents

Part I

Introduction

Chapter 1

Introduction

"The introduction of many minds into many fields of learning along a broad spectrum keeps alive questions about the accessibility, if not the unity, of knowledge."

Edward Levi (1911-2000)

Humans are of curious nature. They tend to ponder on the underlying message or functionality of what they observe, especially when presented with something that appears simple yet is not directly understood. If the viewer is also able to influence what is presented to him, his interest is further stimulated and he is likely to participate actively in the process of presentation and observation.

This almost automatic behavior of humans can be exploited to attract and capture the attention of someone casually passing by. In the context of the Science Linx exhibition project, coming students will be encouraged to join the University of Groningen. To stir up their creative and curious minds a project has been initiated that attempts to stimulate these prospective adolescents. The project should make use of a video application that appeals interactively to the viewer.

To this end an artistic video application is discussed that will present distorted images of a real-time video stream. The aimed distortion should result logically yet implicitly from the recorded video. To emphasize the artistic, interactive effect the recorded viewer should be segmented from the background and projected into a different scene, e.g. a beach or forest.

1.1 Research Questions

The two topics of this project will be addressed separately in this thesis. Prior to these discussions the research questions concerning the two focuses need to be established. Although both research questions are presented in the respective parts of the thesis (see section 1.2 for an overview of the organization) we will introduce these here for completeness.

The interactive video application calls for the following research question:

> How should an artistic, digital video application be developed that stimulates the fascination of the spectator and persuades him to contemplate on the underlying mechanism of the presented effects?

To answer this question the following topics are investigated in the part two of this thesis:

1. How would video art be fitted into scientific research?
2. What would be an interesting and easily produced artistic effect?
3. What is required to obtain this effect?
4. Is the resulting effect of the developed artistic filter as expected?

The second focus of this thesis concerns the segmentation of the foreground objects (the viewers) from the background. This presents another research question:

> How could a foreground-background segmentation algorithm be developed that is, or approximates, illumination invariance such that changes in brightness do not affect the segmentation (and optionally tracking) performance of the algorithm in realistic scenarios?

This question requires the addressing of the following topics, which can be found in part three:

5. What is the behavior of illumination and noise in signals processed by a CCD camera?
6. How could vector collinearity be used as a foundation of illumination invariant foreground-background segmentation?
7. Which issues should be resolved to obtain an implementation of this algorithm?
8. What is the performance of the algorithm in the presence of strong illumination changes?
9. How is the algorithm ranked in comparison with competitive tracking algorithms based on the PETS 2001 dataset?

1.2 Thesis Organization

The thesis is divided into five parts. This introduction contains the first part. It shortly introduces the dual focus of this thesis, provides an abbreviated version of the research questions discussed in later parts of this work and discusses the purpose of applying the research in the public domain.

The second part is concerned with a Non-Photorealistic Rendering (NPR) filter. It presents a broad overview of NPR techniques and proposes the design and implementation of a video distorting mirror.

Part three proposes a video segmentation technique that should render background-foreground extraction invariant to changes in illumination. This *Collinear Vector Model* is based on the assumption of (observed) color constancy in the presence of brightness variation.

Part four comprises the conclusion of both focuses of research, the bibliography of literature and the index. It also discusses the implementation of the model as part of the interactive video artwork *Pool of Intentions*, to be presented at the Science Linx exhibition of the University of Groningen.

Appendices containing the images of and results on the test sets are located in the last part, along with the developed algorithms.

Part II

Video Distorting Mirror

Chapter 2

Introduction

"There is nothing more difficult to take in hand, more perilous to conduct or more uncertain in its success than to take the lead in the introduction of a new order of things."

Niccolo Machiavelli (1469-1527)

Eversince mankind has been blessed with creativity can one find things that base their construction on the sole purpose of pleasing the human senses. For example, the stone age cave walls were the canvas of the world's earliest painters. Although men's creativity is seemingly limitless, the expression of it is bounded by the tools the artist possesses. The advancement of technology is an important ingredient to the production of new expressions of art. As with each invention the artist is invited to explore a new realm, in today's technology-driven society the artistic possibilities are abundant. Among all the various styles and forms, modern art applies the newest of all competences in its creational process. A particular area is the field of *digital media* that enables the artist to manipulate pieces of art in previously unimaginable ways.

Although the advantages are somewhat disputed by those who prefer the title of artist to be restricted, the ease that digital media offer in the process of creating art enables the greater public to join the artistic community. While the explosion of practitioners would flood the market, with many pieces of art likely being of marginal quality, it will raise the level of artistic mastership as artists are triggered to improve their skills by the intensified competition. Furthermore, by the leap into the digital world, art no longer has to be constructed physically[1]. Having dropped the physical limitations the process of creating art can now be focussed on the manipulation of complete objects, instead of having to draw multiple brush strokes for each object. Taken into the extreme, the complete process can be automated. Based on some input information, a painting can be created by a set of rules. This does not mean that the creativity has been lost; it merely has been shifted to an indirect and more abstract position: the creation of the rules.

The automatic process of creating art enables it to be applied in an interactive manner. A scene, optionally manipulatable by a spectator, can be the source of information. By presenting the output again to the spectator, an interactive piece of art has been established. This principle

[1]One may start a discussion about the physicality of a digital form, since it may boil down to manipulating magnetic bits. This, however, is a discussion we will not let ourselves to be pulled into. In this report, we regards something that resides in the digital world *not* to be of physical form.

does not limit itself to creating paintings or drawing. One can extend this principle to artistic videos, as long as the real-time[2] requirement is fulfilled.

2.1 Research Question

This part of the thesis focusses on the design of an interactive digital video filter. The application that is aimed for should hold both artistic and communicative functionality. Furthermore, the presented effect should motivate the spectator to engage into interaction with the video filter. These considerations correspond to the following research question:

> How should an artistic, digital video application be developed that stimulates the fascination of the spectator and persuades him to contemplate on the underlying mechanism of the presented effects?

To answer this question the following topics need investigation:

1. How would video art be fitted into scientific research?
2. What would be an interesting and easily produced artistic effect?
3. What is required to obtain this effect?
4. Is the resulting effect of the developed artistic filter as expected?

2.2 Trajectory

Chapter 3 initiates the discussion of digital art with a definition of Non-Photorealistic Rendering. We then proceed by motivating the research and development in this field and present an overview of examples of such systems.

This introductory part is followed by chapter 4 dedicated to the model of the video distorting mirror. The design phase is followed by a discussion on the set-up of the system and the implementation of the filter (chapter 5). An evaluation by experimentations and demonstrations in chapter 6 concludes the subject.

[2]The notion of 'real-time' is a common one for interactive processes. The notion of 'real' indicates that the process takes place in approximately the same time as would in the real world. For a process that has no real-world counterpart, its combined reaction- and operation-time should be shorter than the maximum delay allowed.

Chapter 3

Non-Photorealistic Image Rendering

"Painting is the representation of visible forms... The essence of realism is its negation of the ideal."

Gustave Courbet (1819-1877)

3.1 Introduction

Non-Photorealistic Rendering is one of the basic definitions from which we shall build this research. Clearly, a thorough understanding of it is required before one could advance to more complex discussions. In this chapter we will supply the reader with an introduction into the field of Non-Photorealistic Rendering of images.

3.2 Definition

When we inspect the syntactical definition of Non-Photorealistic Rendering, we notice that it is actually a contradiction of another definition: Photorealistic Rendering. Hence, it seems more likely to illustrate the definition of Non-Photorealistic Rendering by its negation.

3.2.1 Photorealistic Rendering

In 1964 researcher Ivan Sutherland originated the field of Computer Graphics as an offspring of his research on user-machine communication systems [Sut64]. From that moment onward the dominative strive of researchers in this field has been to create images that are indistinguishable from photographs. The latest advances in Computer Graphics have made feasible a flawless modelling and rendering[1] of virtual environments and realistic three-dimensional games.

[1] In Computer Graphics, rendering is the numerical process of generating the pixel colors of an image from a high-level description of a 2D or 3D scene. These descriptions deal with the objects' material characteristics, illumination, shading, depth and camera view.

A well-known motivation for the successes in Computer Graphics was the continuous increase of labor expenses while simultaneously hardware became more inexpensive. There was however another important influence that favored its rapid and successful development. During the 1960's one of the most popular painting styles in Northern America was the so-called *Photorealism* [Kin99]. Artist belonging to this movement attempted to create paintings that closely approached the reality and quality of detail of photographs. One of the leading examples of this movement was Richard Estes (figure 3.1).

Figure 3.1: Richard Estes' 'Paris Street Scene' (1972) oil on canvas painting is one of the leading examples of the artistic movement of Photorealism. Estes specialized in painting street scenes with elaborate window reflection that approach photographic images.

When we take a closer look at the history of image rendering, this strive for portraying reality is somewhat curious. Especially the idea that the 'ideal' shape of an illustration is its most realistic form has found great resistance. Although there have been some artists who aimed on displaying the reality in its finest details (e.g. the Neoclassicist Jean-Auguste-Dominique Ingres [oA06]), the vast majority of artists tried to record the world in a *subjective* way [Gom60]. One of the main arguments against this ideal is that photorealism only offers an image from a single point in time, from a single viewing angle and with a restricted sight of view. Furthermore, artefacts (e.g. noise introduced by pixel voltage amplification in a CMOS-camera [RJB+03]) caused by the creation of photographic images are often visibly present. Like other rendering techniques, photographic rendering has its advantages and disadvantages.

With the ongoing advances in Computer Graphics and many important questions having found

answering, questions arose where to find new challenges. One of the domains into which interest has flowed focusses on the idea that photorealism is only one of multiple rendering methods. This new field of research is accordingly called *Non-Photorealistic Rendering* and has since found an increase of attention.

3.2.2 Non-Photorealistic Rendering

We have defined Photorealistic Rendering in the previous section, which enables us to determine what Non-Photorealistic Rendering is: the process of creating images that do not comply with reality. The domain of Non-Photorealistic Rendering (henceforth called NPR) is a very broad one by definition. Any process that can render an image that does not conform to reality can be encapsulated into this domain.

Since NPR might occupy itself with any rendering technique that does not produce photorealistic images, we feel the need to inspect the global goal of NPR. Although we will address the motivation of NPR research further on in this chapter (section 3.3), an important element cannot be left unnotified. NPR images often have a *communicative* objective. By omitting detail in the process of displaying an object in ways that defy reality, other kinds of information conveyed by an image can be emphasized. For example, an architect does not require bricks in a wall to be visible since it would only distract his attention from the important information (i.e. the construction of the building). The omission of these elements would strengthen relevant parts and clarify the information within the image. NPR images would thus serve a user's aid by displaying just those elements that convey the information required by the user.

Gooch and Gooch [GG01], in their comparison with photorealism in the computer graphics scene, put it this way:

> In NPR images are instead judged by how effectively they communicate. [...] NPR involves stylization and communication, usually driven by human perception. Knowledge and techniques long used by artist are now being applied to computer graphics to emphasize specific features of a scene, expose subtle attributes, and omit extraneous information to give rise to a new field.

Not all scenes or images lend themselves as input to the artistic process. It is important for researchers to bear in mind for what kind of scenes and images and to what purpose the algorithm is developed. If mistakenly chosen, NPR algorithms may produce images of low quality[2] with respect to the requirements. However, NPR algorithms are useful in the task of enhancing suitable images by significant amounts.

3.2.3 Definition Inconsistencies

It is argued that the definition of NPR is not a just one. Many of the techniques that are used to create non-photorealistic images do not apply any rendering. Instead they apply modelling or post-processing methods to produce the desired effects. Despite common effort to alter its name, the term NPR remains to prevail in denominating the field of creating non-photorealistic images.

[2]The notion of quality for images quantified by their communicative power is a hard one which we shall not address here.

One other inconsistency in the definition of NPR is the difference in interpretation of 'non-photorealism', or, for that matter, 'photorealism'. Common among artist, the term 'photorealism' refers to the school of painting that focusses on simulating the effect of a camera lens, including all distortions and effects by reflection involved. Yet, in the field of computer graphics, this term describes the domain of images that are identical to reality. Hence, the distinction between the two definitions depends on the means of how an image is conceived; either by camera or by eye. Since our approach is one of simulating artistic styles, rooted in the field of computer science, we adopt the latter definition.

3.3 Motivation

For what cause would one indulge oneself in research on non-photorealistic rendering techniques? Would it have been induced out of pure interest, or can it be for the sake of practical benefit? A tip of the veil had already been lifted when we addressed the definition of NPR. Not surprisingly, there are other motivations for the investigations into this field.

3.3.1 Simulation and Tools

Art often appeals to us, yet it is hard to explain how it succeeds in attracting our attention. Artist often apply styles that can be categorized into e.g. classic, baroque, romantic or impressionism. By studying and sketching out the craft of typical artistic styles, researchers hope to prevent them from vanishing into oblivion.

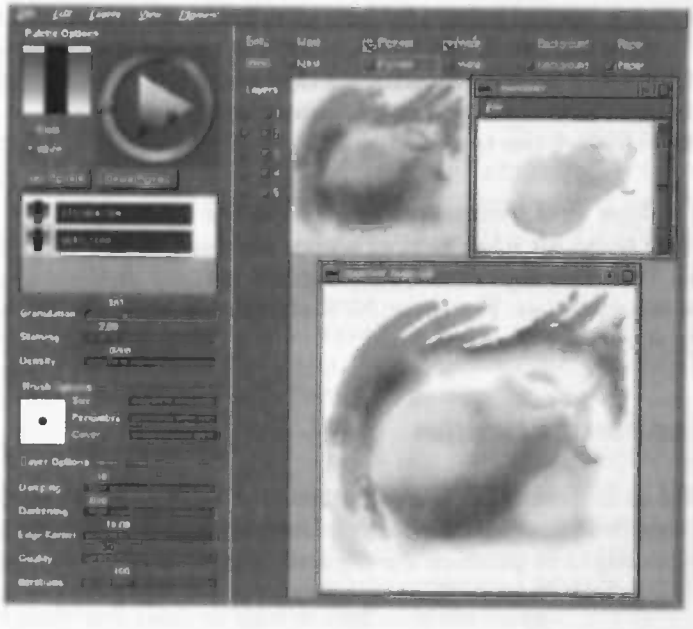


Figure 3.2: An interactive painting application for simulating and producing watercolor images.

This descriptive investigation of artistic crafts enables for interactive and automatic rendering system, such as the one depicted in figure 3.2. These digital tools simulate particular styles. They provide non-skilled users the means to express themselves more easily and stimulate expert artists to excellence and reflect. Most of these interactive systems require the user to supply strokes by using a mouse device [Hae90]. Alternative forms let the user interact by a pen-device, as in the tool by Baxter et al. [BSL01]. An interesting approach is taken by DeCarlo and Santella, who have developed a system that uses eye tracking to determine stroke location and thickness [DS02, SD02].

Simulation of the artistic process is not only concentrated on the craft itself. Researchers have focussed on mimicking the physical properties of brushes, pencils and canvases. This requires the digital simulation of substrate, diffusion and gravity [Coc91a]. The foundation of the painting, e.g. paper or canvas, can be simulated by modelling the characteristics of the material. A more thorough approach has been carried out by Curtis et al. modelled fluid flow, absorption and particle distribution to simulate watercolor paintings [CAS+97].

3.3.2 Information

Images, and hence paintings, not only have an esthetic purpose. Often they are created to convey information, be it, for example, to stir up old memories or to inform someone of an upcoming event. Furthermore, images are applied for clarification, like construction schematics that instruct the user how to build his table. These schematics generally are not realistic as they often show elements that are otherwise invisible or suppressed by its surrounding. Visualizing what is otherwise hidden could be achieved by separating parts of an object, thereby disclosing its internals. This is the *exploded diagrams* technique, which is often used in drawings that aim to display the composition of objects (figure 3.3).

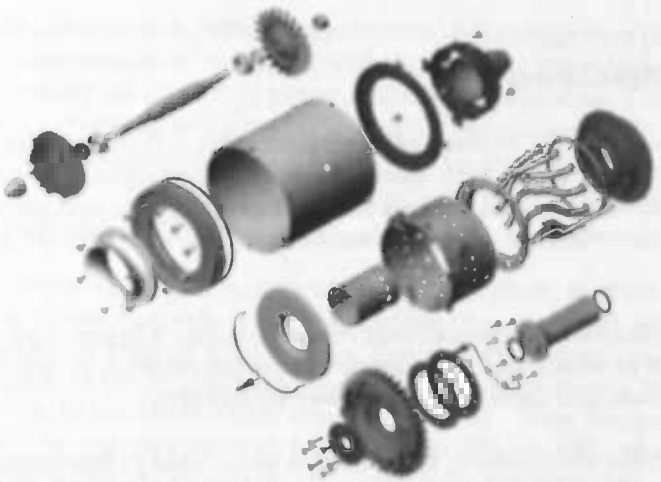


Figure 3.3: Exploded diagram of the rotor system of a helicopter engine (courtesy of www.wren-turbines.com).

NPR can contribute to the user the notion of approximation. From a psychological point of view, photorealistic images seem to imply that the depicted object is exact in terms of aspect and appearance. Consequently, photorealistic techniques are not particularly suited for depicting what

is merely an inexact representation, for it may exaggerate the accuracy of the simulated scene with respect to reality. As an example, the Piranesi system [Ric97] has proven its superiority to photorealistic rendering of architectural designs (figure 3.4).

Figure 3.4: Non-photorealistic rendering of a 3D architect's model by the Piranesi painting system (courtesy of Robin Lockhart, OBM International).

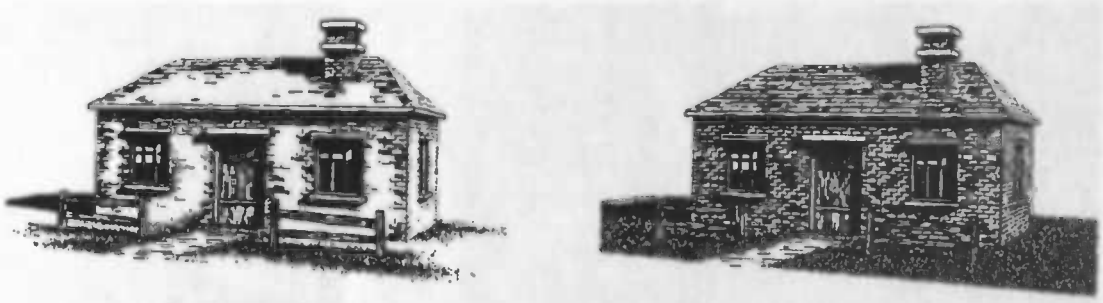
3.3.3 Viewer Stimulation

Images are paramount in the process of communication. They transfer information to the viewer at various levels of detail, with more efficiency and effectiveness than any other means. Highly suitable in advertisements, images can stimulate viewers and increase their attention. As Margret Hagen described, in a panel discussion at Siggraph 1998 [Phi88] on the design of effective images:

> "The goal of effective representational image making, whether you paint in oil or in numbers, is to select and manipulate visual information in order to direct the viewer's attention and determine the viewer's perception."

By leaving details to the imagination, the viewer is stimulated to interpretation of what would otherwise merely be observed. Grundland et al. noted that, by moulding the viewer's impression of an image, visual elements caused by rendering styles can play a constructive role in visual communication [GGD05]. As a consequence, the transfer of information can be applied more effectively. Also, by inviting the viewer to supplement what is observed with his own imagination, the viewer may become intertwined with the image.

Stimulation of attention can be seized in multiple ways. A common approach, however difficult to attain, is often referred to as *information by indication*. It employs a strategy of omission of



(a) Texture by indication.

(b) Texture by presentation.

Figure 3.5: Example of indicational drawing. The viewer's imagination is stimulated by omitting texture details (left). Compare this with texture rendered at full (right). With the same technique, by emphasis on particular parts, the transfer of information can be controlled (courtesy of Winkenbach and Salesin).

detail and indication of texture by marginal use of suggestive strokes. An appealing example is found in the work of Winkenbach and Salesin [WS94]. They devised a semi-automatical pen-and-ink system that could generate illustrations with an economical use of strokes that appear more attractive than a fully rendered example (compare figure 3.5(a) and 3.5(b)).

3.3.4 Multiresolution and Compression

When it comes to data compression, NPR images have an advantage over photorealistic images. Conventional image compression is concerned with the reproduction of photorealistic quality of images and do not consider the context of presentation into the process. These techniques strive to reduce data loss to a minimum, while the NPR-based compression might well be inherent to a lossy approach. As discussed in section 3.3.2, details can be omitted from a NPR image without effecting, maybe even boosting, the perceptual quality. As such, compression by reduction of image detail can be mutually beneficial.

In case an image has been created on a computer using a stylistic method such as painting or pen-and-ink drawing (these methods will be discussed in section 3.4) the individual strokes and dots can function as the base of compression. The strokes, or dots, can be combined or left out to produce an image of lower resolution that requires less storage. Grundland et al. have extended this approach to multiresolution images [GGD05]. Their images can be of varying resolution, dependent on the user-defined level of grain (figure 3.6). Multiresolution can be of use in the progressive display of images in multimedia presentations, where data is often transmitted over a narrow bandwidth network.

Another motivation of stroke-inspired compression can be found in the distorting effect of common lossy compression techniques. Especially for artistic images, in which details often convey the essence of the piece of art, artefacts of distortion caused by compression may yield a marraged image. The resolution approach of Grundland et al. aims on conserving the particular stroke details.

Figure 3.6: Example of an image rendered by paint stroke simulation at multiple resolutions (courtesy of [GGD05]).

3.3.5 Entertainment

While photographic images portray what is real—or, at least, present a notion of reality—and indicate to the viewer the truthfulness of the depicted situation, NPR pictures traditionally have a different purpose. Theirs is to entertain the spectator. The casualness in style and content with which the viewer is approached allows cartoons and caricatures to be taken with a wink (although they often convey an informative or moral message). An example of such an image that strongly appeals to the imagination is depicted in figure 3.7(a).

Not since the origination of the animation industry around 1910 have NPR images seized the attention of the mainstream public. To them the NPR scene brought enjoyment and a temporary escape from daily life. In the often fictional realm of cartoons and animations they could forget their sorrows from their otherwise precarious world. On this objective, the entertainment of the public, has many NRP image productions been focussed.

As ministers of originality, artist joined the entertainment scene and experimented with the NPR capabilities. Quickly they tempted the current artistic standards by introducing what is now referred to as *modern art*. With the introduction of modern computers came video art, in which artist utilized the temporal property of this new medium to express their intentions. In recent years interactive video art have appeared more frequently on exhibitions. Work by William Forsythe and Camille Utterback, the latter depicted in figure 3.7(b), can be seen as representative of this new development.

3.4 Examples

The dominate share of research in the field of NPR has been focussed on automatically reproducing artistic painting and drawing styles, with or without the aid of the user. Gooch and Gooch, in their surveying book on Non-Photorealistic Rendering [GG01], have presented the most prominent of projects within this area of research.

To better describe the NPR research and their offspring—the NPR applications—Gooch and Gooch have categorized these into three domains:

(a) What's on a mans mind (Sigmund Freud).

(b) 'Drawing from life' (Camille Utterback) in the American Museum of Natural History, New York.

Figure 3.7: Two examples of NPR art as an entertaining medium: a humoristic caricature by Sigmund Freud expressing his theory of male psychology (left) and a still from the interactive video exhibition by Camille Utterback presenting spectators as a composition of the four proteins of DNA.

- *Artistic media simulation.* This type of application replaces the classic tools of the artists. The application simulates the physical properties and interaction between the artistic components. Examples of artistic components are the applicators (brush, pencil, eraser, etc.), the substrate (canvas, paper, wood, etc.) and media (aquarelle paint, ink, charcoal, etc.).

- *User-assisted image creation.* The generation of artistic images has been semi-automated, leaving difficult tasks of the rendering process to the user. These tasks typically comprise segmenting the image into regions, selection of the brush type and the artistic style. The system would then fill in the image segments by simulating the painting or drawing technique. These applications simulate the painting techniques and thereto incorporate the skills of artist experts. This enables non-skilled users to produce images with a hand-crafted look and feel.

- *Automatic image creation.* These systems generate images without any user interaction. They simulate specific artistic styles and require an input image as reference. Because scenes in the reference image vary in difficulty (e.g. number of segments or detail) this strongly restricts the quality of the output images. Instead of simulating artistic media or painting techniques, the goal of automatic NPR systems is the simulate a painting as a whole.

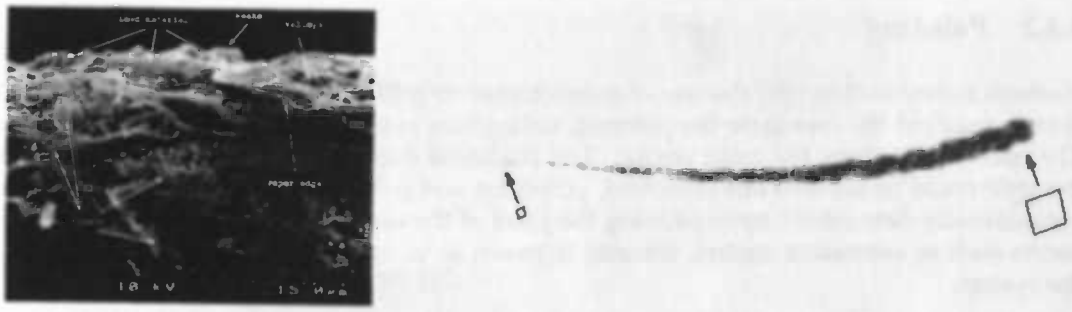
The survey of Gooch and Gooch [GG01] is limited to painting and drawing systems. The field of NPR, however, includes many more image rendering techniques as has been explained in section 3.2.2. We will therefor present an anthology of NPR research categorized by type rather than usage.

3.4.1 Drawing

From the standard computer line-drawing algorithms NPR researchers produced modifications, such as the *PencilSketch* drawing system by Allan Vermeulen and Peter Tanner [VT89], that are capable of creating hand-drawn effects. Because ink and graphite do not vary in tone, problems arise to indicate shade or texture [Tat04]. The production of multiple strokes put together can be difficult and would often not yield the desired effect. Jodoin et al. have proposed to generate texture by synthesis of a hand-drawn sample [JEGPO02]. Salisbury et al. allow users to select previously defined stroke textures or to create new ones [SALS96]. In their system each texture sample is stored as an image in several tone values. A more recent system [SWHS97] allows users to indicate for each region the stroke texture, direction field and tonal value. The system then adds strokes based on the rules of the selected texture until the desired tone is achieved. After each stroke the system evaluates the tone by comparison with a blurred version of the original illustration. To attain a consistent shading new strokes are drawn at locations with the largest tonal difference with respect to the original image. An example of an image drawn by the system of Salisbury et al. is given in figure figure 3.8.

Figure 3.8: Image of a raccoon created by the drawing system of Salisbury et al. using orientable stroke textures [SWHS97].

Maro Costa Sousa and John Buchanan approach the problem in a fundamentally different way. Based on their observations by electron microscope on the interaction between drawing materials (pencil, paper, eraser, blender) they have designed a system that simulates these interactions [SB99b, SB99a, SB00] (figure 3.9(a)). They model graphite pencils by hardness and shape. The lead (writing core) of their pencil is composed of graphite, clay and wax. Whereas the proportions between graphite and clay determine the hardness, wax is used to lubricate the pencil tip during movement on the paper. A three-dimensional polygon describes the shape of the pencil



(a) Electron microscope image.

(b) Path stroke by pencil shape variation.

Figure 3.9: Cross-sectional view from a real drawing by electron microscope (figure 3.9(a)). The image depicts the lead material to adhere to the paper fibers [SB00]. Figure 3.9(b) shows a stroke path of a pencil by the drawing system [SB99a]. The influence of the pencil shape varies along the path.

which effects the way the graphite is distributed on the paper surface. A three-dimensional height field is used to simulate the drawing paper. This field is defined in the horizontal and vertical dimensions by a grid, for which each point is assigned a maximum local lead volume (height). For each grid point that is effected by the pencil the amount of lead, wax and clay is computed at run-time. Due to the constant interactions pencil shape and paper heights vary. This causes the local texture and shape of strokes to vary along their path (figure 3.9(b)). Blenders and erasers are modelled by the same algorithms, but are also allowed to absorb and distribute lead particles. An example of a drawing by this system is depicted in figure 3.10.

(a) Source image.

(b) Drawing with 6H pencil.

(c) Smudging and erasing.

Figure 3.10: Results produced by the drawing system of Sousa et al. A high contrast photograph (figure 3.10(a)) has been rendered using a 6H pencil (figure 3.10(b)). Medium-soft pencils with light pressure have been applied interactively to create additional strokes. Figure 3.10(c) is the result of application of smudge and eraser onto image 3.10(b) to create the darker tones, shadow and some of the face lines [SB00].

3.4.2 Painting

Haeberli introduced in 1990 the use of paint strokes to generate digital paintings [Hae90]. His system required the user to do the painting, aiding him in the creation of the digital paint only. The user had to draw the exact stroke. The thickness depended on the mouse speed, while the style could be selected out of curved, pointillist and polygonal form. The stroke color was automatically determined by examining the pixel of the underlying original image. Enhancements such as saturation control, contrast between local regions and noise were provided by the system.

Digital painting systems can be classified in two subfields: those that simulate the characteristics of artistic media and those that simulate the artistic process [GCS02]. The former approach focusses on low-level painting, whereas the latter is high-level. The approaches do not exclude each other. Instead, combining them results in a rich painting system grounded on physically realistic rendering while supplying the user with familiar and efficient tools.

The simulation of artistic media can be further divided into work that simulate the physics of media and those that simulate the look. Haeberli's rudamentory work can be shared among the latter one and is regarded by many as its founder. Similar to the work of Sousa et al. discussed in section 3.4.1, the physics simulation concentrates on the applicator (the brush), the medium (paint) and substrate (canvas).

Brush simulation is the basis of the work by Steven Strassmann [Str86], who's method produces sumi-e images[3]. To model the sumi-e brush, a set of color values define the color to be applied by each brush bristle at a certain pressure. Integer values indicate the amount of ink stored at each bristle position. When a stroke *quad* (basic surface of a stroke) is created each integer is decremented to simulate the placement of ink onto the substrate. A result of Strassmann's system is shown in figure 3.11.

David Small attempted to predict the interactions between pigment and water in combination with paper fibers [Sma91]. Cockshott et al. propose their "wet and sticky" method to model wet paint on a substrate [Coc91b, CE91, CPE92]. Both systems apply similar techniques, but the one of Cockshott et al. is more elaborate. Like the work of Souse et al, Cockshott introduces a parameterized grid, called the *intelligent canvas*, that describes the absorbency, horizontal and vertical orientation, and the type and volume of paint that each grid cell stores. These cells are deformable cubes which tops are open in order for the paint to flow in. The paint is modelled by *paint particles* and are described by color, liquid content, viscosity, drying rate, mixing ability and transparency. This enables the modelling of many types of paint, such as watercolor, oil and acrylic. The simulation process is controlled by the *painting engine*, which updates the grid cells based on gravity and age, diffusion, mixture of the paint particles. Aging reduces the liquid content of the paint. Gravity influences the movement of the paint by acting on diffusion and surface tension. The painting engine mixes the paint in each cell that has influenced by diffusion and gravity. Finally, the image is rendered by one-to-one mapping of grid cells onto pixels.

The third target of physically-based simulation, media like watercolor and acrylic, is the focus in the work by Curtis et al. [CAS+97]. They extend the work of Cockshott and Small on paper substrate to fluid flow and light interaction. *Wet-area masks* represent the areas of the substrate water has been flowed into. The masks thereby limit the water flow and provide boundary conditions for the fluid flow computations. This flow is modelled by three layers: the *shallow-water layer*, the *pigment-deposition layer* and the *capillary layer* (figure 3.12). The shallow-water

[3]Sumi-e is a type of brush painting that originated in China. The ink causes smooth images to be created.

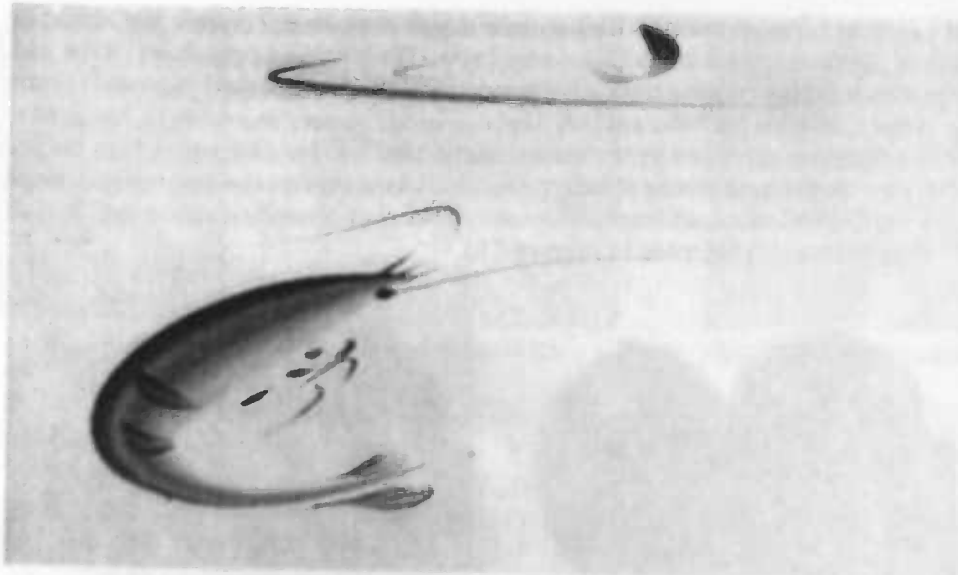


Figure 3.11: Strassmann's "Shrimp and Leaf" image in sumi-e painting style [Str86].

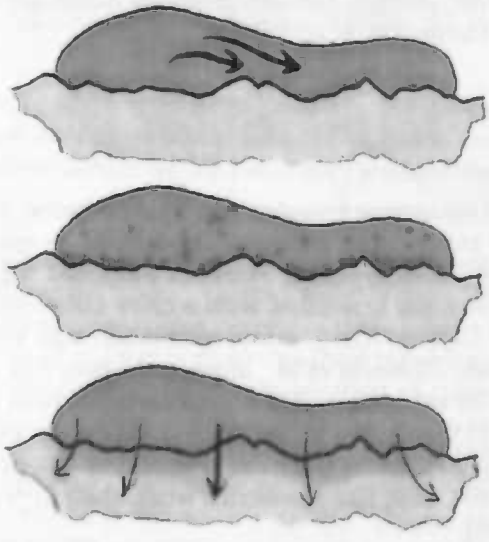


Figure 3.12: The three layers of Curtis et al. fluid model. The shallow-water layer (top) allows water and pigment to flow. The pigment-deposition layer (center) absorbs and desorbs pigments. The capillary layer (bottom) controls the capillary flow through the substrate [CAS+97].

layer allows the pigmented water to flow across the surface, bounded only by the wet-area mask. During this process the water can lift, carry or deposit pigment on the paper. Fluid flow and interaction is controlled by water velocity, pressure, substrate slope, viscosity, viscous

drag and pigment concentration. The pigment-deposition layer controls the absorption and desorption of pigment from the shallow-water layer. The physical properties of the individual pigment particles influence the physically-based simulation. Absorbed pigment is transferred throughout the substrate by the capillary layer, which causes the wet-area mask to expand. This capillary flow is controlled by the saturation and the fluid holding capacity of the substrate. Curtis et al. have developed three painting programs based on this system: interactive painting, automatic watercolorization and three-dimensional non-photorealistic rendering. A result from the second application is presented in figure 3.13.

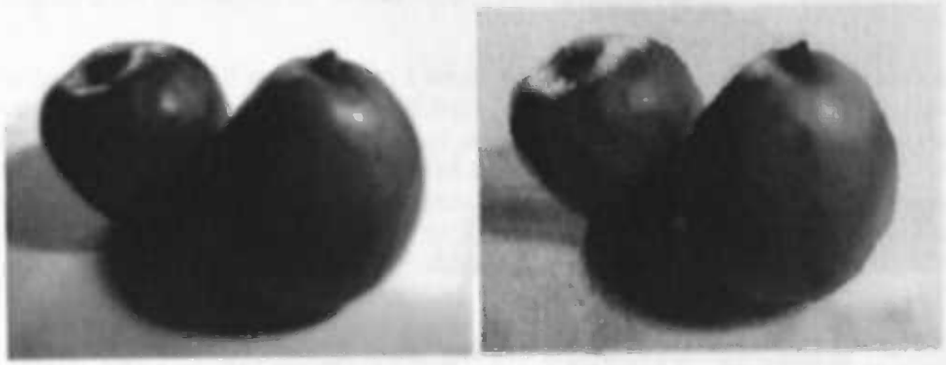


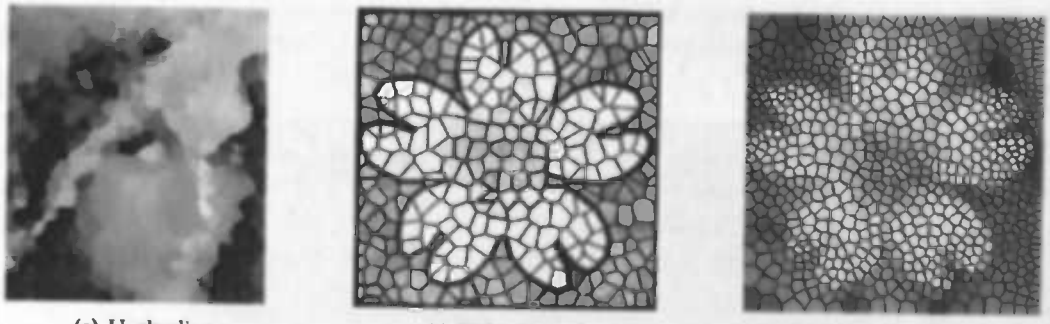
Figure 3.13: Source image (left) and automatically generated watercolor image (right) by the automatic watercolorization system of Curtis et al. [CAS+97].

3.4.3 Mosaics

Battiato et al. [BDBFG06] categorize mosaics into four domains: *crystallization mosaics*, *ancient mosaics*, *photomosaics* and *puzzle image mosaics*. Crystallization mosaics are composed of a set of randomly shaped images tiled together. Haeberli proposed the use of Voronoi diagrams to construct tiles [Hae90]. Each tile was filled with a color obtained from the covered region in the source image (figure 3.14(a)). The Voronoi tessalation was performed randomly, without considering image features. Dobashi et al. improved upon this idea by taking into account image edges [DHJN02]. To minimize color fluctuations between neighboring tiles, assuming a smooth color gradient in the source image, they define a squared difference error function. The function is iterated to determine the most resembling color between pixels of the source and target image in each tile. Still, like Haeberli's work, their approach suffers from local color fluctuations (figure 3.14(b)).

Faustino and Figueiredo let the size of the tiles take into account image densities by applying a density function on a Centroidal Voronoi Diagram (CVD) [FDF05]. The CVD is first seeded by sampling the image using a quad-tree. The tiles that compose the diagram are then created using a density function. The density function computes gray-scale image gradients using central differences. The diagram adapts to the mass distribution of the density function, yielding large tiles in regions with low image detail and small tiles in regions with high image detail. Especially, due to the image gradient, tiles are aligned along edges. An overview of the process is displayed in figure 3.15.

Hausner [Hau01] initiated the research on automatically producing ancient mosaics. A novel



(a) Haeberli. (b) Dobashi et al. (c) Faustino and Figueiredo.

Figure 3.14: *Overview of crystallization mosaics by various authors.*

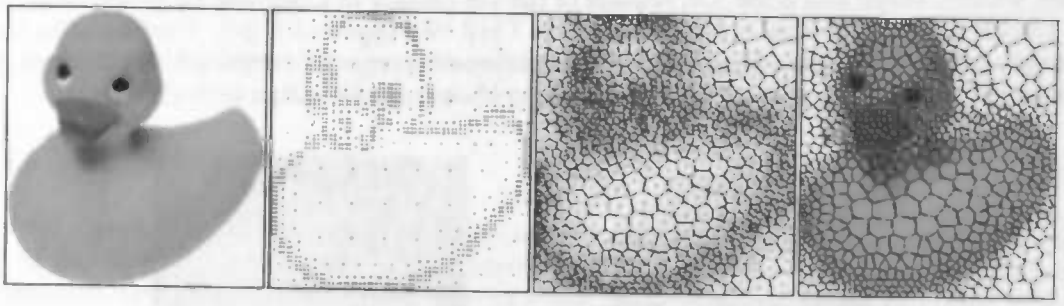
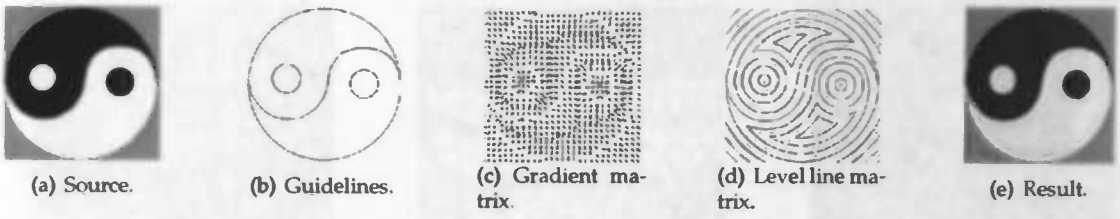


Figure 3.15: *Mosaic generation by adapting to image gradients. The source image (left) is sampled by a quad-tree, where the dots indicate leaf centers (center left). A CVD is constructed using an image density function (center right). The individual tiles are filled with color averages from the source image (right).*

technique, applicable for any kind of NPR rendering, has been proposed by Schlechtweg et al. [SGS05]. They propose a multi-agent system, called RenderBots, in which each agent represents one stroke. A competition between agents is carried out to distribute themselves throughout the image. Each agent has a set of rules that describes their behavior and defines the rendering style. More specialized is research grounded on observation of ancient mosaic techniques. As Battiato et al. [BDBFG06] point out, ancient mosaicists used precision in orientation, shape and size of tiles along with emphasizing dominant edges. Also, tiles were not gridded but instead put into random order to avoid the creation of artificial edges. Di Blasi and Gallo [DBG05] used these proven methods, yielding impressive results. Edges, or *directional guidelines*, are located in a binary image created by equalization, smoothing and thresholding the luminance channel of the source image. To determine the location and orientation of the tiles, Di Blasi and Gallo compute a gradient matrix and a level line matrix of the image from the directional guidelines. The level line matrix produces chains, onto which the centers of the tiles are placed. Tiles overlap is prevented by clipping them. An overview of the algorithm is depicted in figure 3.16.

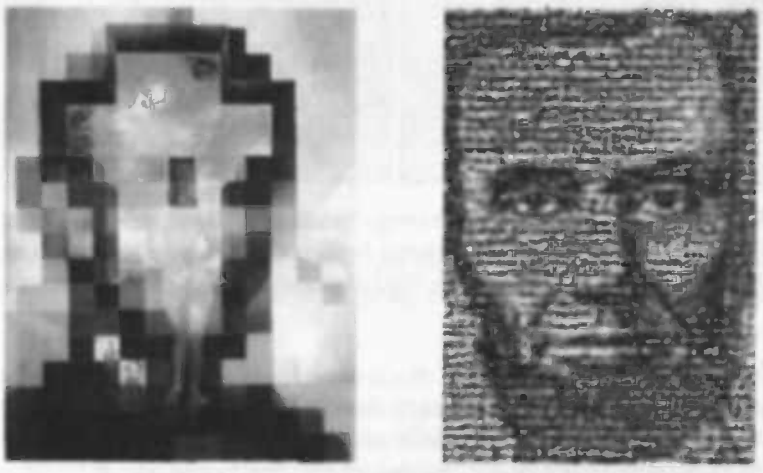
The technique of photomosaics generates an image by matching color statistics from image thumbnails to regions in a source image. Very few research has been carried out in this domain. The most well known works have been created by artists, such as Salvador Dali with his "Gala mirando el mar mediterráneo" (figure 3.17(a) which at a distant of 30 meters becomes a portrait



(a) Source. (b) Guidelines. (c) Gradient matrix. (d) Level line matrix. (e) Result.

Figure 3.16: Overview of the ancient mosaics algorithm by Di Blasi and Gallo [DBG05].

of Abraham Lincoln. Dali incorporated a small image of Lincoln as a tile. The first computer-generated photomosaics originate from Robert Silvers [SH97] who, during his graduate work on MIT Media Lab, devised an algorithm to generate mosaics that match the original image in tone, texture, shape and color. On request of the US Library of Congress, he created a mosaic of Lincoln by using images from the American Civil War (figure 3.17(b)). Photomosaics have recently gained popularity. Supported by the advance of computer hardware, applications like Photoshop [Ado03] are now capable of rendering photomosaics within seconds.



(a) Salvador Dali. (b) Robert Silvers.

Figure 3.17: Examples of photomosaics.

Mosaic rendering typically is time consuming due to the large amount of image processed. Di Blasi et al. [DBGP05] have introduced an algorithm that produces puzzle image mosaics in acceptable computation time. A more advanced method is proposed by Kim and Pellacini [KP02]. Most likely inspired from the Renaissance painter inventor Giuseppe Arcimboldo, who composited heads from clumps of vegetables, they introduced the Jigsaw Image Mosaic (JIM). This mosaicing technique, although similar to the one of photomosaics, uses arbitrarily shaped tiles to compose the final image within a container (i.e. set of regions to be tiled). They define the problem of tile location as a *mosaicing energy function*, which is to be minimized to find the optimal result. The energy within the system is described by color differences between source and target image and gap and overlap between tiles in the target image. When a solution has been found a refinement phase takes places by deforming unstable tiles and incorporating this deformation into the energy function. The refinement is then computed by minimization of the

extended energy function, balancing between the amount of deformation and reducing the gap, overlap and color differences. Finally, the resulting tiles are filled with images from a library. An overview is presented in figure 3.18.

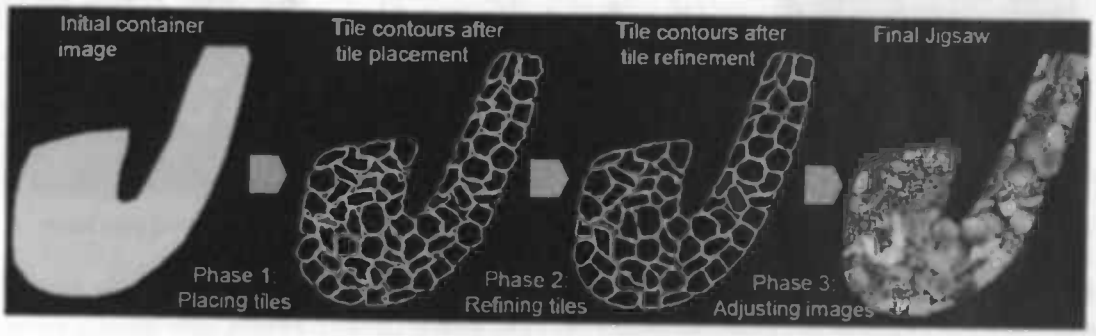


Figure 3.18: Phases of the Jigsaw Image Mosaic algorithm by Kim and Pellacini [KP02].

3.4.4 Stained Glass Rendering

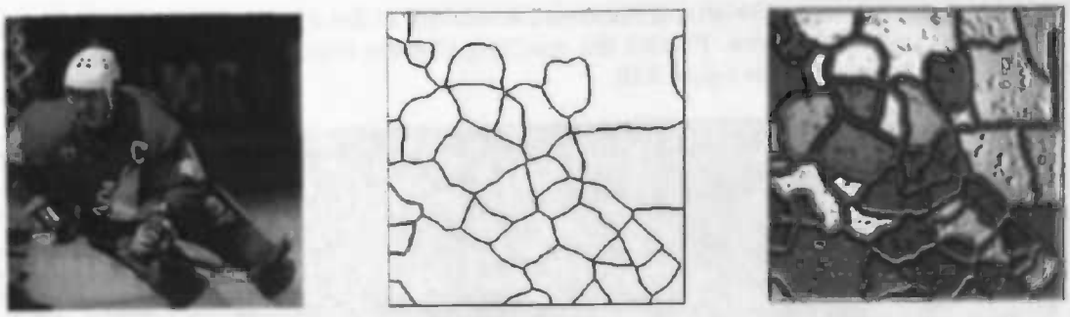
Stained glass images relate to mosaics and tiling in that image segmentation, region alignment based on edges and region filling are required to produce convincing output. Only very few researchers have attempted to mimic stained glass. One of the first works dedicated to automatic generation of stained glass images was the undergraduate project of Mark Grundland [Gru97]. Also, commercial image processing software has focussed on these algorithms. However, David Mould mentions in 2003 that these applications, such as Adobe Photoshop [Ado03], are incapable of adapting to the underlying content (the source image) [Mou03]. Also, Stephen Brooks, Kiichi Urahama and Kohei Inoue have researched the automatic generation of stained-glass images [Bro06, UI04, Ino].

To obtain a stained glass image, Mould segments a source image into regions which he post-processes to avoid highly concave[4] surfaces, excessively small or large and *islands* and *bottlenecks*. This is achieved by parallel erosion and dilation of all regions (figure 3.19(b)). To produce the glass colors Mould compares the average color from each region to a set of seven colors, typical for medieval stained glass, and selects the replacing color that has the smallest distance in RGB space. The selected color is then shifted in a random direction to produce color variations throughout the image. When mapping the regions onto the image, large displacements near the region boundaries create the surrounding leading (figure 3.19(c).

Stephen Brooks uses an existing stained-glass image as reference to the creational process [Bro06]. The source and reference image are both segmentation on multiple scales, using a standard watershed transform and a color image segmentation method of Deng and Manjunath [DM01]. The user can optionally split and merge regions from the source image, mixing the various scales, to produce a final segmentation.

To select the proper colorization, the texture and color statistics of each source region are compared to those of every reference region. Alternatively, this comparison can be made with images of real stained glass. Color comparison is performed by a distance map in feature vector

[4] An object is concave if at least one line segment between two of its vertices goes outside the object, where the vertices are located on the object contour. An object that is not concave is called convex.



(a) Original image. (b) Segmentation and opening. (c) Stained glass result.

Figure 3.19: Example of Mould's stained glass filter [Mou03]. The original image (figure 3.19(a)) is segmented and all regions are simultaneously eroded and dilated (center). Finally, after filtering extreme-size regions, the stained glass is rendered (right).

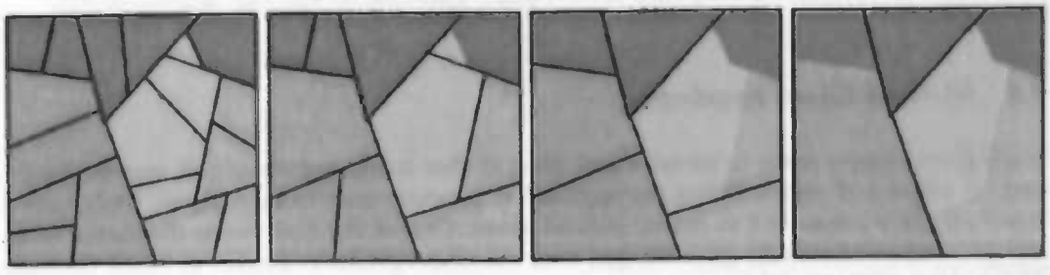


Figure 3.20: Segmentation by Brooks' stained glass filtering algorithm on multiple scales. The colors indicate the ground truth of the segmentation, whereas the black lines shows the computed region boundaries.

space, where the dimensions of this space are spanned up by the histogram bins of the colors in $L^*a^*b^*$ colorspace. Texture statistics are derived from the mean texture contrast computed by the eigenvalues of the second moments of the region. Texture comparison is performed by a Manhattan distance measure. The color statistics of the most closely related reference region are then adopted. To obtain realistic stained-glass regions (i.e. glass imperfections), Brooks applies Perlin noise and discontinuity-sensitive warping, which produces small color facets. The user can optionally replace colorized regions by solid, pre-defined stained-glass images. Finally, the leading boundaries are rendered by fitting them to cubic splines and matching the result to a mask by means of image analogy [HJO+01].

Figure 3.21: Example of Brooks' stained glass filter [Bro06]. The source image (left) is segmented and colorized based on a reference image (center) producing the final result (right).

Chapter 4

LinearDelay Algorithm

"Use time, or time will use you."

Salvador Dali (1904-1989)

In early 2006 the Pinakothek München [Pin] exhibited a project in its modern art section that raised the public interest. William Forsythe, a choreographer of The Forsythe Company [For], presented an interactive video installation by which he invited people to dance. Reacting upon their own distorted reflections, spectators were challenged to find the artist in themselves.

This type of interactivity, in which art and motion are combined, is the goal of research in this chapter. The presentation of a transformed image of the viewer invites him to interact. To achieve this an analysis of the *City of Abstracts* project is carried out in section 4.1. Then, in section 4.2, we hope to apply Computer Vision techniques to obtain a similar effect.

4.1 City of Abstracts

The choreographic project *City of Abstracts* by Forsythe has been around since 2000. It has been presented in many cities, such as Düsseldorf, Dresden, München and Moscow. They have witnessed passers-by to hold their pace and flock around the video installation. It has been positioned in areas where people often walk by, mainly at access routes, flights of stairs and central halls and plazas. A high-resolution video camera, located at a raised and distant position, records all movement within its field of view. The recordings are processed by a filter that outputs a delayed and deformed image (figure 4.1). They seem to possess characteristics that are typical for figures created by Salvador Dali.

The video installation presents an astonishing effect of shapes floating through space. The deformation effect appears to apply on regions of movement only; image regions that are not subject to motion remain unaffected. Objects in motion appear stretched and react upon their movement with varying time delay. Several seconds pass before the true reflection becomes visible again, but only after the object has come to a complete standstill.

Figure 4.1: A demonstration of the distorting effect by City of Abstracts at the Pinakothek der Moderne in München.

The distorted image presented by the interactive installation functions as feedback. By this the spectators are provoked to investigate and reflect their choreographic skills from the correlation between their self-image and the projected representation. The *City of Abstracts* enables the common man to discover and stimulate the artist and dancer within himself. Besides broading the practice of choreography among the public, Forsythe redefines the meaning of dance.

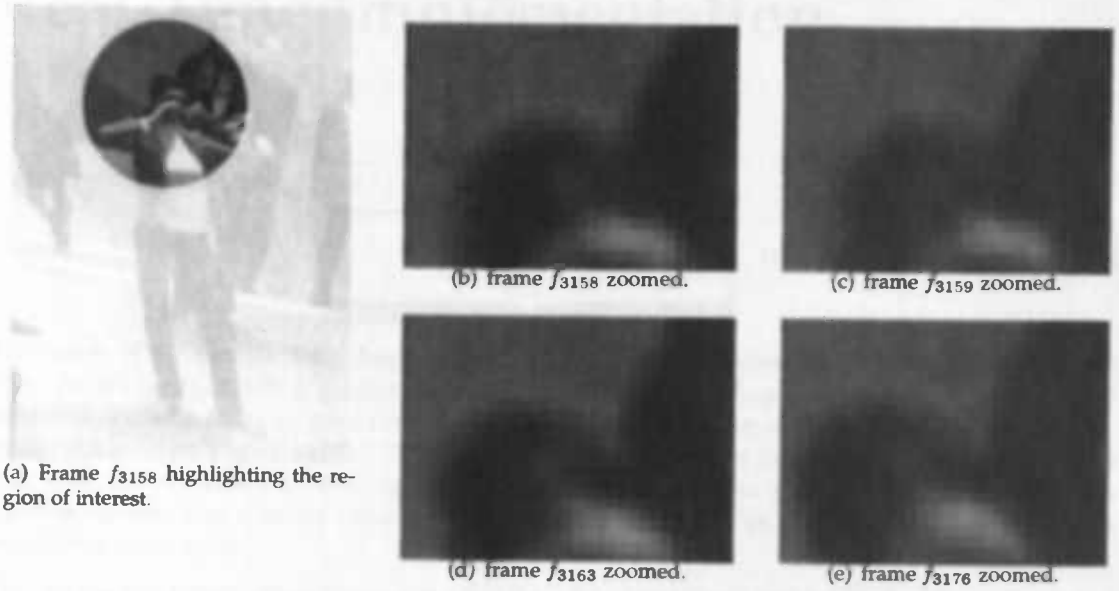
4.1.1 Motion Analysis

When inspecting an example from the distorting process it seems that the deformation of motion fuses with its delay. When, at a certain time instant, a person initiates a movement none of the scanlines in the representation witness a distortion. Hence, at zero time delay the region of interest in the output is unaffected (figure 4.2(b)). Then, in the first frames, the top few scanlines become subject to distortion (figure 4.2(c)). At subsequent time instances more scanlines, in top to bottom order, are included into the set of distorted scanlines (figure 4.2(d) and 4.2(e)) until the complete image region converted by the person's body becomes transformed.

The conclusion of movement causes a similar effect as the initiation. In this case the top scanline belonging to the image region of the body immediately displays the current situation. Lower

scanlines react with delay, depending on the distance of the scanline to the top of the image region. If the movement remains zero the distortion would ultimately seize and the representation of the person would become real again.

To formalize this effect let us define f_t as a frame on time t. The frame in which movement of the concerned image region is initiated is denoted f_0. At frames following this initiation, the number of scanlines that become subject to the distortion increases linearly with the time difference $\mathrm{d}f_t = f_t - f_0$. The number of distorted scanlines can be described by $\alpha \cdot \mathrm{d}f_t$, where α is a scalar. For example, if at $t = 0$ scanlines 1 to 100 exhibit motion, the algorithm would produce a distortion at scanline 10 at time $t = \lfloor \frac{1}{\alpha} \rfloor = 0$. In continuation, scanline 2 will show a distortion at $t = 1$ and scanline 100 at $t = 50$.



(a) Frame f_{3158} highlighting the region of interest.

(b) frame f_{3158} zoomed.

(c) frame f_{3159} zoomed.

(d) frame f_{3163} zoomed.

(e) frame f_{3176} zoomed.

Figure 4.2: Stills from the City of Abstracts promotion video. The images in (b)-(e) are magnifications of the highlighted area in (a) at succeeding time instants. The zoomed images display a delayed transformation that increases with the scanline index.

4.2 LinearDelay Filter

The gradual distortion of moving objects indicates a varying delay among the scanlines. More precisely, the delay appears to increase with the scanline index, if ordered from top to bottom. Because only a vertical distortion is present, i.e. among scanlines, any horizontal—within scanline—filtering can be neglected. Based on this observation and the motion analysis of the *City of Abstracts* demonstration in the previous section we will now present our own video distorting mirror algorithm. From hereon we will reference to this algorithm as the *LinearDelay* filter.

Let us define x to be the input sequence, where $x(i, j, t)$ is its function on the image location (i, j) at time t that yields a pixel value. Similarly, let us define y to be the output sequence, with $y(i, j, t)$ its function. The output is obtained by transforming the input based on

$$y(i,j,t) = x(i,j,t') \qquad \text{where } t' = \left\lceil \frac{t-i}{\alpha} \right\rceil . \tag{4.1}$$

Hence, scanline i in the output sequence at time t equals the same scanline in the input sequence at time t'. The value of t' depends on the scalar α and the location i of the scanline within the image. By the definition of t' the delay by which scanline i is displayed increases with the value of the scanline index i. This effect is schematically depicted in figure 4.3 for $\alpha = 1$.

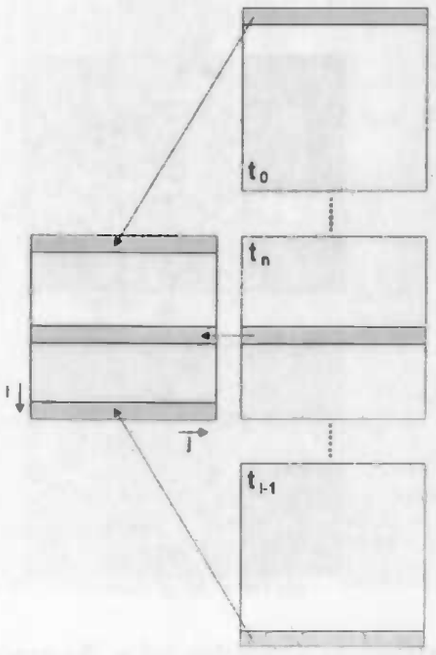


Figure 4.3: Schematic overview of the LinearDelay filter for $\alpha = 1$. The filter operates on the output at the current image (t_0). For each row $i \in \{0, \ldots, height - 1\}$ in this image, the filter replaces it by row i from the image recorded at time $t = t_0 - i$ (i.e. i frames back in time).

Chapter 5

Real-Time Implementation

"You may delay, but time will not."

Benjamin Franklin (1706-1790)

The merit of the entertaining characteristic of the *LinearDelay* filter lies in its interactive property. As a video distorting mirror, its application serves to please and challenge its users. To optimally benefit from its potential, any implementation of the video filter should obey real-time performance requirements. The filter should be capable of processing video frames at the rate they are presented by the capturing device. This real-time requirement demands for an efficient system that directly interacts with the necessary devices. This chapter discusses how to achieve these goals.

During the implementation process, the focus on the algorithmic architecture and programming should not be distracted by any peripheralities. Low-level video processing, such as extracting video frames from stream, decoding frames and exporting them to a display device is a cumbersome process. It should therefore be avoided in the implementation process if possible. Fortunately, video players exist that offer this, and more, functionality to the user. More in particular, MPlayer enables a transparent interface into which filter modules can be plugged with surprising ease.

Prior to the creation of the video plug-in we need to orientate on the exact set-up of the hardware devices, which will be the subject of discussion in section 5.1. After having obtained a clear image hereon, it is required to inspect the internals of MPlayer in more detail in section 5.2 and 5.3. Finally, in section 5.4, we will focus on the design of the video filter.

5.1 Hardware Set-up

The system on which the development and the experimentation of the video filter will be carried out comprises a 650MHz Pentium III Coppermine processor with 256KB level-1 cache that has been supplied with 250MB main memory. A digital web-cam (Logitech QuickCam Messenger) will be deployed to deliver a constant video stream. The camera is capable of capturing 640×480

images at a rate of 30fps (frames per second). An impression of the hardware set-up is presented in figure 5.1.

Figure 5.1: An impression of the hardware set-up of the system on which the real-time video filter will be developed.

5.2 MPlayer

MPlayer—its full name is *The Movie Player* [MPl06]—is a video player suitable for many systems, but mainly designed to operate on Linux platforms[1]. It is capable of processing a large variety of input formats and supports numerous output formats. This wide support and the presence of easy-to-use options renders it highly suitable for the task of real-time video processing.

The functionality of MPlayer is composed of five elements. Each of these elements offers a standard interface, such that new codecs[2] and filters can be easily included into the program.

[1]The *Pool of Intentions* application discussed in chapter 13 uses the *LinearDelay* filter using another (yet similar) implementation in a Windows XP environment.

[2]The word codec is an abbreviation of the term coder/decoder and serves to compress or decompress data. Examples of video codecs are MPEG, WMV and DivX.

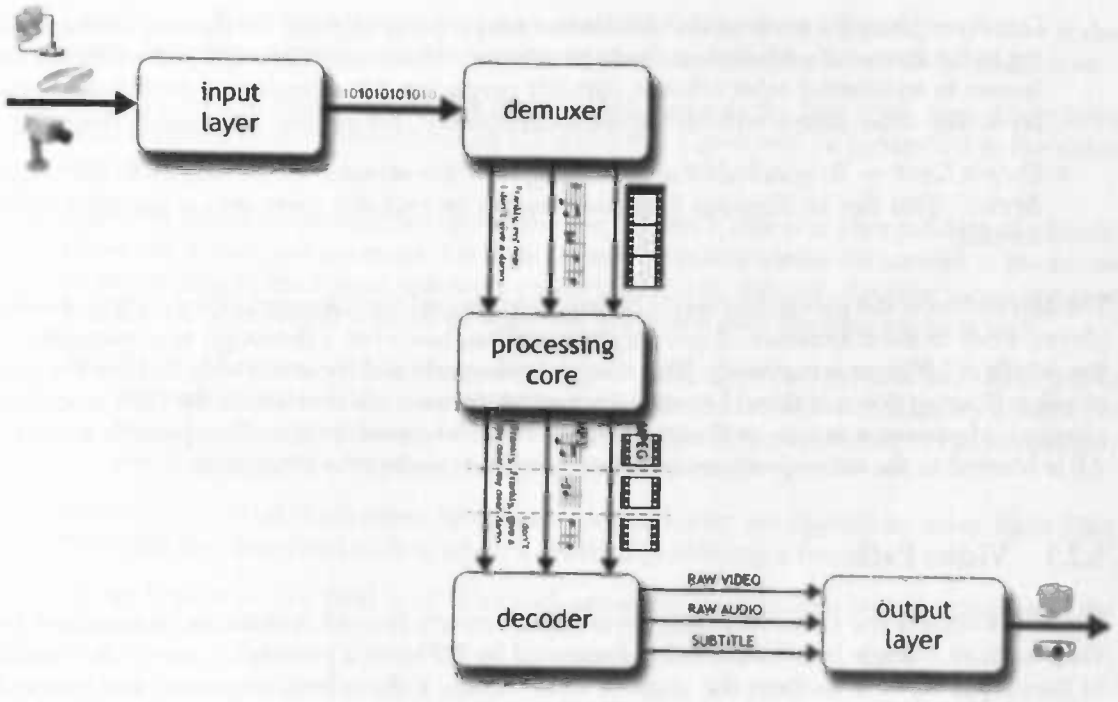


Figure 5.2: Diagram of MPlayer's processing flow. Data is read from a device, split into various channels (video, audio, subtitle), synchronized, decoded into a raw format and presented to an output device.

The tasks of the five standard elements are clearly distinct due to the extensive modularity of MPlayer, albeit some dependency cannot be prevented (e.g. knowledge on the type of frame encoding is required among the video process units). The video processing layers (see figure 5.2) can be listed as follows.

- ◇ *Input Layer* — The input layer reads a video and/or audio stream from a media source. This media source can either be a file, standard input (stdin), video CD, DVD or a network.
- ◇ *Demuxer* — The input stream usually consists of multiple channels, containing audio, video and DVD subtitle data. These channels are often intertwined and need to be separated. Such is the task of the demuxer, which demultiplexes[3] all input into individual streams.
- ◇ *Processing Core* — With all streams available separately, the core video processor attempts to synchronize the video frames with the audio and subtitles. Since the offset and bitrate of video and audio, depending on the format, may vary and hardware devices may cause significant delays (up to seven seconds) simple synchronization based on frame timestamp does not suffice. Instead, for the (possibly varying) timespan of each video frame, it has to be calculated how large the audio sample needs to be.

[3]Multiplexing originated from the field of telecommunications where multiple, usually unrelated signals transmitted through cables had to be combined into a single signal. Video and audio content on a DVD normally are multiplexed to prevent the DVD player to skip back and forth between the audio and video stream (which are positioned in series if not multiplexed).

- ◇ *Decoder* — Once the audio and video frames are properly aligned, the decoder corresponding to the format of each stream has to be selected. The decoder extracts and reformats the frames to an internal color scheme, suitably preparing it for display by the fifth and final layer. The video filters, with which we are concerned, are applied posterior to decoding.
- ◇ *Output Layer* — To conclude the processing flow the streams are passed on to an output device. This device displays the video frames or encodes them into a specified video format.

The description of the processing layers describes in general the information flow within a video player. Prior to the commence of any implementation, however, a thorough understanding of the details of MPlayer is necessary. This comprehension should focus not only on how the path of video filtering flows, it should as well discuss the framework into which the filter module is plugged. Moreover, a notion on the encoding of video frames and their colorspaces is required. All is covered in the subsequent sections, before we advance to the filter itself.

5.2.1 Video Path

Figure 5.3 depicts the video decoding layer of MPLayer's flow of operations. It describes the video path of a single frame from being presented by MPlayer's processing core to its transfer to the output layer. It harbors the chain of video filters; a chain both important and optional. The filters provide post-decoder adjustments in color format, intensity and saturation. They even offer basic motion tracking, picture-in-picture and removal features. Yet all can be omitted without inflicting damage to the video path.

Insight into the path of video frames is vital to understanding the interface details upon implementation. First of all, we will discuss the processing flow of video frames. This, and its functionality, is enumerated below.

1. *Decoding Request* — Following the frame synchronization phase, MPlayer' processing core requests the decoding of the compressed video frames.
2. *Codec Request* — The codec core determines the proper decoder based on the codec description in the frame header. It subsequently calls the selected video codec.
3. *Format Request* — Before decoding the first frame the codec needs to know in what kind of format its decoded frame has to be stored. This mainly involves selecting the proper colorspace, which depends on the preference of the output device.
   - 3a. *Colorspace Selection* — The initialization procedure of the codec support unit will inspect a list of optional color formats. It queries both the video codec and the video output streamer (through the initialization procedure of the video filters) for support on the formats. Finally, it determines the preferred (i.e. best internally usable) colorspace based on the outcome of the queries.
4. *Allocation Request* — A buffer has to be allocated in memory to store the new, decoded frame, a task managed by the codec support unit.
   - 4a. *Buffer Allocation* — If direct rendering is supported, the codec support unit will ask the video output layer, by using the video filter interface, for the buffer allocation. If not supported, the codec support unit will itself allocate a frame buffer.

5. *Decoding* — With the colorspace and an available buffer at its disposal, the video codec starts decoding. After completion it submits the decoded video frame to the codec core.

6. *Filter Call* — The codec core passes the decoded frame to the first filter. If no filters have been specified (during invocation of MPlayer) this signal will be forwarded to the video output layer.

7. *Allocation Request* — Similar to the video codec, a filter's task is to alter the data of a frame. However, it does not reformat this data. Instead it manipulates the content of the frame while obeying to the format selected by the video codec. As such, the filter requires a new frame to be constructed to store its outcome. For this it calls the filter support unit.

   7a. *Buffer Allocation* — Like the codec support unit, the filter support unit attempts to request the video output layer (through the interface of the next filter, if available) to allocate a direct rendering buffer. It this fails (i.e. direct rendering is not supported) the filter support unit allocates an internal buffer.

8. *Filter Chain* — If multiple filters have been specified, they are applied in series. Each filter then calls its subsequent sibling when it finished processing a frame.

9. *Output Request* — The final 'leaf' filter will prone the video output layer for display of the video frame.

5.2.2 Plug-in Framework

The plug-in framework consists of a variety of functions, its responsibility ranging from construction, initialization and destruction of the filter to frame processing. Not all functions re-

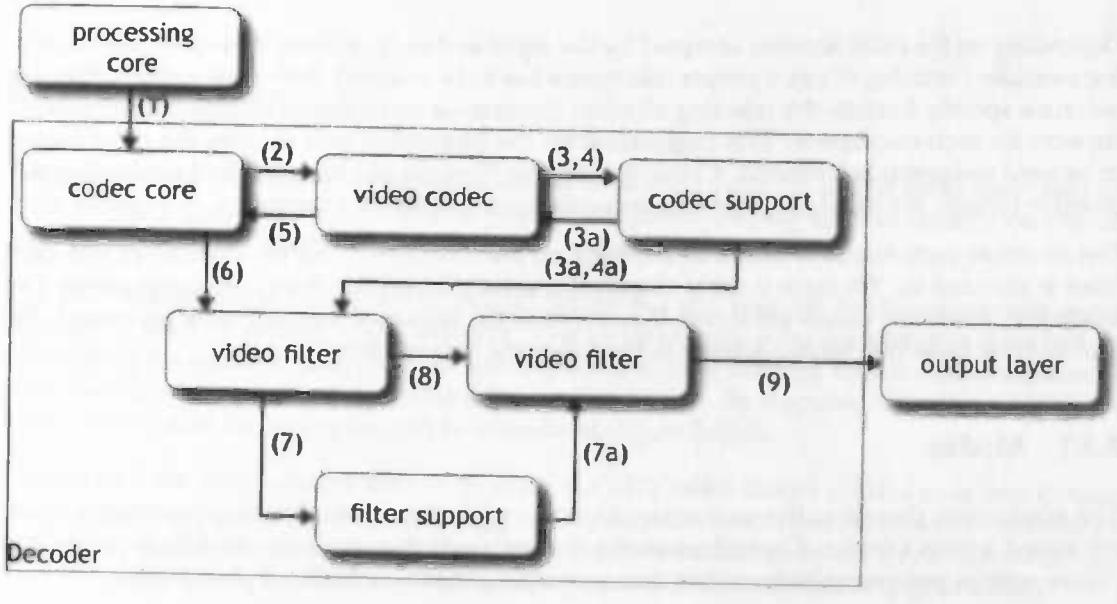


Figure 5.3: The video decoder layer of MPlayer's processing flow determines and calls the appropriate codec, which in return outputs a decoded frame. Each filter then subsequently manipulates this frame.

quire implementation, since the tail filter of the filter chain provides a default *fallback* implementation. The relevant functions are summarized below.

- ◇ *open(·)* — The obligatory *open(·)* function is called by the codec core when the filter is appended to the filter chain. By means of this call the filter receives its parameter options and should construct a *vf_instance_t* structure. This structure controls the interface connection of the video filter with the framework. Exemplary to the structure is the preferred resolution of the frame and the pointers to the local implementation of the framework.
- ◇ *query_format(·)* — Prior to initialization, the *query_format(·)* function is called to investigate upon the support of the colorspaces. This request is part of the colorspace selection process in the decoder layer (see section 5.2.1).
- ◇ *config(·)* — Initialization is taken care of by *config(·)*. However, due to the dynamic characteristics of the memory allocation process, in which knowledge on image dimensions and encoding is required, the buffers can not be initialized here. This task has been transfered to *put_image(·)*'s maiden call, during which *configure_buffers(·)* is invoked.
- ◇ *put_image(·)* — The processing heart of the video filter is the *put_image(·)* function. Any conversion, filtering or transformation of image data is performed here. To obtain an output buffer it queries the next filter.
- ◇ *uninit(·)* — This function is called to destruct the filter and to free any memory the filter had privately allocated.

5.3 Frame Formats

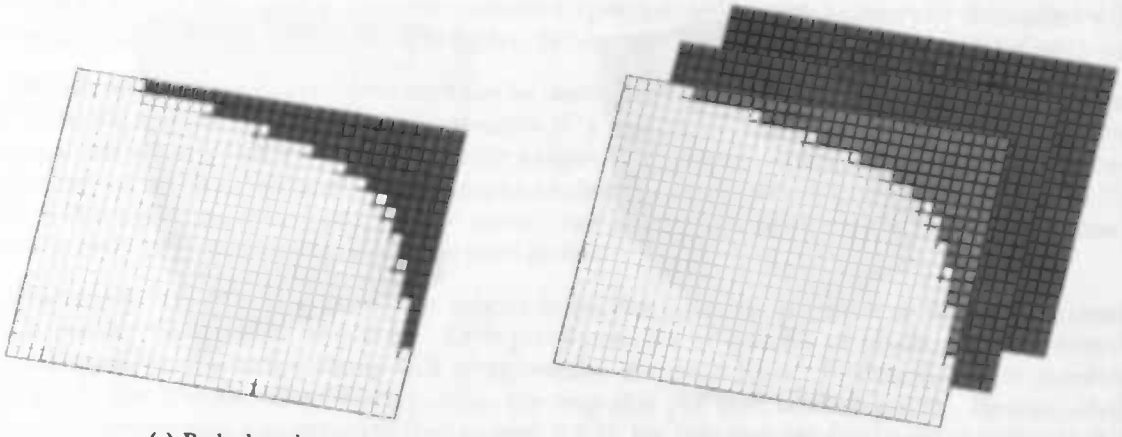
Depending on the color formats accepted by the input and output layer, as well by the processing modules (decoder, filter), a proper colorspace has to be selected. Although a video filter can enforce a specific format—by rejecting all other formats—a more decent approach is to provide support for each colorspace. This negotiation by the processing core renders the color format to be used unknown beforehand. Consequently, the filter should be capable of processing any possible format. Knowledge of the common colorspaces is therefore required.

The structure each frame is stored in depends on the colorspace and the number of bits each color is encoded in. Yet there is more to utilizing pixel information than colorspace alone. The mode into which the pixels are stored is relevant to the sequence they are to be processed. We will address these two format aspects in the following two sections.

5.3.1 Modes

The pixel mode describes the way into which the color values, independent of their format, are stored within a frame. Dependent on the type of mode the sequence into which pixel color values require processing differs. Two common modes exist: packed and planar mode.

In *packed* mode all color components have been interleaved together into one array. For each pixel, the color components are accumulated into a small pack of bitvalues (figure 5.4(a)). These bitvalues together define the pixel color, based on the selected colorspace.



(a) Packed mode. (b) Planar mode.

Figure 5.4: *Impression of an image in packed and planar mode. Color component data is either stored in sequence (packed) or separate (planar).*

Planar[4] modes have the color values separated among three images (figure 5.4(b)). Each image holds pixel values of a single color component (red, green or blue in case of RGB). Dependent on the colorspace (see section 5.3.2) the resolution, or size, of each of the three images may differ. The term 'planar' may cause confusion because, with a stack of three separate images, we actually behold three dimensions. It does not refer to the entire mode, however, but indicates that each image—or plane—is two-dimensional. At a pixel location only a single value has been stored, unlike the packed mode.

5.3.2 Colorspaces

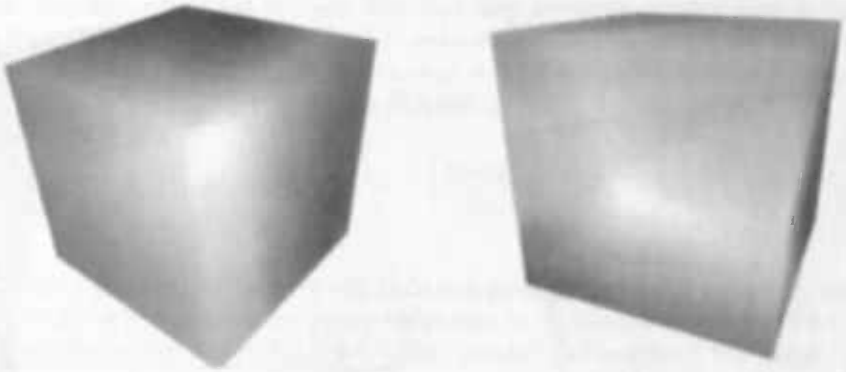
When someone, foremostly a painter, attempts to create a color he would most likely mix the three primary pigments, red, blue and yellow. Electronic display devices usually do not use such a color scheme, but utilize the RGB or YUV colorspace instead.

The RGB colorspace is composed of red (R), green (G) and blue (B) color components and is commonly used by analog VGA monitors. The value of each color components describes the intensity of the color. RGB applies an additive color mixing strategy that describes the kind of light that should be emitted to produce the specified color. By mapping the color components onto a cube's axes its colorspace can be visualized (figure 5.5(a)).

Instead of three color components as in RGB, the YUV color model utilizes only two chrominance components, with U ranging from green to red and V from green to blue. The luminance, or brightness, of both color components jointly define is controlled by the third component, the Y. In the normal YUV model the components are encoded with equal size (figure 5.5(b)).

Since YUV is based on RGB, both can be derived from each other. RGB to YUV conversion is

[4]Something is planar if it involves, relates to, or is situated in two dimensions.



(a) RGB colorspace. (b) YUV colorspace.

Figure 5.5: The RGB and YUV colorspaces mapped onto a cube. The face of the YUV cube shows the colors defined by the U and V components at maximum brightness (Y = max). Contrary to the RGB colorspace, the 'color' white is not located on the edge of the cube, but somewhere on the face.

defined by

$$Y = +(0.257 * R) + (0.504 * G) + (0.098 * B) + 16 \quad (5.1)$$
$$U = -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128 \quad (5.2)$$
$$V = +(0.429 * R) - (0.368 * G) - (0.071 * B) + 128 \quad (5.3)$$

and YUV to RGB conversion by

$$R = 1.164 * (Y - 16) + 1.596 * (V - 128) \quad (5.4)$$
$$G = 1.164 * (Y - 16) - 0.391 * (U - 128) - 0.813 * (V - 128) \quad (5.5)$$
$$B = 1.164 * (Y - 16) + 2.018 * (U - 128) \quad (5.6)$$

The YUV color model can be motivated by physiological evidence[5]. The photoreceptor cone cells in the human eye respond to long (yellowish-green), medium (bluish-green) and short (blue-violetish) wavelengths [WS82]. At the same time rod cells react on light intensity, yet are more sensitive than their conal counterparts [KSJ00]. Advantage can be taken upon this discrepancy by reducing the number of bits for the chrominance components and rely on the interpolative power of the eye. While standard YUV encoding requires 24 bits (8 bits per component), compression can yield a total of 12 or 16 bits per pixel (2 or 4 bits per color component).

5.4 Video Filter

The *LinearDelay* filter applies a delaying effect to the video stream. The algorithm does not alter pixels, nor does it change lines of pixels; it merely duplicates pixel lines from the buffers to the output frame. The buffer memory allocation, frame storage and output generation required

[5]Color models exist that represent human color perception more closely, such as HSL or HSV. Their main disadvantage, however, is their dependency on particular devices.

herefore will be discussed in the next sections. Experimentation with memory optimization will also be addressed. To initiate this discussion, however, we need to address a few global notions.

The rate of the LinearDelay filter effect can be controlled by a user-determined parameter. When increasing the rate, the delay in appearance of a pixel line is diminished linearly. Since the frame rate does not alter—at least not due to user interference—this suggests that the number of required buffers decreases. For example, examine the case where `rate = 2` (default is `1`). To obtain a doubled speed-up, the filter should not retrieve one pixel line per buffer, but instead has to copy two subsequent lines from each buffer.

Adherent to the discussion on color modes in section 5.3.1 the structure of the video frames is based on the number of planes. Each plane consists of a series of pixels. The combined pixels span up the entire frame and wrap around the pixel lines. To determine the number of pixels per line, the `width` or `stride`, the step size per line, is to be polled. Because, due to color component compression (see section 5.3.2), the bits per pixel may differ between the planes the chromatic shift can be requested. The chromatic shift controls the factor in which the address of a pixel has to be shifted to map with the actual location of the pixel in the bit array, and is stored in `chroma_x_shift` and `chroma_y_shift` for the horizontal and vertical displacement respectively.

5.4.1 Memory Allocation

To store frames as they are presented to the filter by the codec core, buffer memory has to be allocated. This memory allocation is dynamic, since the size of images may vary[6]. Consequently, because the frame size is unknown of the initialization and configuration phase, buffer allocation takes place at the *put_image(·)* function. The primary task of *put_image(·)* at its first invocation is this buffer allocation. To this purpose *configure_buffers(·)* is called. The code is given below. Although it is mostly self-explanatory, we will offer a short description furtheron.

```
1  // Determine the number of required buffers.
2  n_buffers = ceil(height / rate);
3
4  // Construct the planes (1 if packed, >1 if planar).
5  for (i = 0; i < num_planes; i++)
6  {
7    // Allocate buffer to store old frames in.
8    buffer[i] = calloc(n_buffers, sizeof(unsigned char*));
9
10   // Determine the buffer size.
11   bufsize = stride[i] * height;
12
13   // Allocate all buffers.
14   for (j = 0; j < n_buffers; j++)
15   {
16     buffer[i][j] = malloc(bufsize);
17   }
18 }
```

[6] It may even occur that the size of the frames vary over time, while they are presented to the filter. We drop this assumption, since the goal of the effect is to operate in conjunction with a camera, thereby providing fixed formatted frames.

Buffer allocation involves a twofold process. The first step involves construction of the planes. Dependent on the color mode (see section 5.3.1) one or more buffer arrays are required. Each plane will maintain a buffer array. At base rate (`rate = 1`) the number of these buffers is equal to the height of the image. At higher rates, it is defined by

$$\texttt{n\_buffers} = \left\lceil \frac{\texttt{height}}{\texttt{rate}} \right\rceil. \tag{5.7}$$

Next, the buffers are allocated. All buffers will hold complete frames, although only the current frame requires complete storage. Memory optimization is addressed in section 5.4.4. For the moment, the size of the buffer depends on the width (stride) and height of the image.

5.4.2 Frame Storage

When the first frame is received by the filter, the *put_image(·)* function is called. Following buffer allocation, the buffers have to be initialized. Since only a single frame is available, all buffers are filled with instances of this frame. During subsequent invocations of the function, every new frame is stored in a buffer.

To store a frame, each plane is copied to the reserved buffer. To determine whether multiple planes have to be copied (the color mode would be planar), `MP_IMGFLAG_PLANAR` flag can be checked. One should act with caution, because the color components of the YUV colorspace can be compressed (see 5.3.2). Consequently, the pixels in some planes might span less bits. To cope with this possible difference, the chromatic shifts (`chroma_x_shift` and `chroma_y_shift`) are applied on the width and height of the image, respectively.

```
1  // Copy the first plane to the buffer.
2  memcpy(buffer[0][bufidx], planes[0], stride[0]*height);
3
4  // Check if the color mode is planar.
5  if (flags & MP_IMGFLAG_PLANAR)
6  {
7    // Determine the size of the memory chunk.
8    memsize = width>>chroma_x_shift * height>>chroma_y_shift;
9
10   // Copy the other planes.
11   memcpy(buffer[1][bufidx], planes[1], memsize);
12   memcpy(buffer[2][bufidx], planes[2], memsize);
13 }
```

The buffer list is circular, where buffer `0` is the successor of buffer `n_buffers-1`. The variable `bufidx` points towards the buffer containing the oldest frame, which is to be replaced by its youngest sibling. After storing the new frame, the `bufidx` is advanced to the next buffer:

$$\texttt{bufidx} = (\texttt{bufidx} + 1) \ \% \ \texttt{n\_buffers} \tag{5.8}$$

As a result, the buffer list will always contain the `n_buffers` newest frames.

5.4.3 Output Generation

With all frames, past and present, at our disposal it now boils down to selecting the proper pixel lines from each frame. The age of the frames increases when traversing the buffer list backwards. I.e. the frame at buffer index `31` is older than the one at index `32`. Due to the circularity of the buffer list we can easily skip through the individual frames. Taking into account the filter rate the computation of the buffer index `bufidx` obeys

$$\texttt{bufidx} = \text{round}\left(\frac{\texttt{lineidx}}{\texttt{rate}}\right) \ \% \ \texttt{n\_buffers} \ . \tag{5.9}$$

In some cases the frame is stored in bottom-to-top order. This causes the line selection to process the bottom line from frame buffer 0, while the top line should be taken. To cancel this undesirable effect one is able to manually indicate the direction of the line selection process: either top-to-bottom (standard) or vice versa. To determine which line has to be selected from the frame buffer, following `bufidx`, the `lineidx` is computed:

$$\texttt{lineidx} = \begin{cases} \texttt{lineidx} & \text{if } \texttt{direction} = \texttt{down} \\ \texttt{height} - 1 - \texttt{lineidx} & \text{if } \texttt{direction} = \texttt{up} \end{cases} \tag{5.10}$$

Each pixel line of the output frame is filled by selecting the respective line from the appropriate buffer, repeated for each plane. The code snippet below shows this process. Similar to storing complete frames, a `memcpy` is used to directly address the memory location of `buffer` and copy the memory chunk to `plane`. The `stride` indicates the size of the memory to be copied.

```
1 // Determine the memory offset.
2 offset = (lineidx * stride[0]);
3
4 // Copy a frame line.
5 memcpy(planes[0] + offset,
6         buffer[0][bufidx] + offset,
7         stride[0]);
```

In case the color components in planar mode have been compressed, thereby having reduced the number of bits per pixel, the offset[7] and memory size differ from the standard situation. The code then resembles the following. The chromatic shifts again control the compression factor of the bit sizes of the color components.

```
1 // Determine the memory offset. Offset is bit-shifted
2 // due to color component (U and V) compression.
3 offset = (lineidx>>chroma_y_shift * stride[i]);
4
5 // Copy a frame line.
6 memcpy(planes[i] + offset,
7         buffer[i][bufidx] + offset,
8         width>>chroma_x_shift);
```

[7] An offset in an array or in memory is an integer value indicating the distance or displacement from the beginning of the array or memory chunk up until the given value.

5.4.4 Memory Optimization

Memory usage can be optimized by storing only the required parts of the frames. While at $t = 0$ all pixel lines are required by the filter, top rows have become redundant for older frames ($t > 0$). In the default case of `rate` $= 1$, when integrating over the number of frames, `n_buffers` $= \left\lceil \frac{\texttt{height}}{\texttt{rate}} \right\rceil =$ `height` (cf. formula 5.7). For a 800×600 image this would require 180300 line copies with a total of approximately 650 Mb of memory (at 12 bits uncompressed planar YUV). Not enforcing memory efficiency would require about 1300 Mb of memory. This suggests that memory optimization should yield considerable speed-up gains.

However, the large number of copies in memory required to store a single new frame are hampered by memory hardware restrictions. The time for the CPU and RAM to process them causes the computational speed to drop significantly. For example, on the test system, the filter processes a 320×240 video at approximately 2.3 fps (table 5.1), far less than the usual requirement minimum of 15 fps.

	processing time	**average framerate**
Default (non-optimized)	0.01326733 s	75.37 Hz
Memory optimization	0.43089109 s	2.32 Hz

Table 5.1: Processing time of the LinearDelay filter on a 320×240 *video in planar, 12 bit YUV compressed color mode, running in either default or optimized mode. The filter utilizes* 73728000 *bytes of memory.*

Consequently, the memory optimization hinders the achievement of the real-time requirements. Especially in the case of high resolution images—which are the type of images the filter ultimately should process—the frame rate would drop far below suitable limits, thereby rendering it useless for its main purpose: interactivity.

Chapter 6

Experiments

"No amount of experimentation can ever prove me right; a single experiment can prove me wrong."

Albert Einstein (1879-1955)

In the previous two sections we have discussed the design and implementation of the *LinearDelay* filter. In this section the video filter will be put to the test to observe its effect. We inspect the filter in a realistic environment (section 6.1). This simulation tests the quality of the produced result of the filter with respect to the requirements (i.e. creating a similar distortion as the *City of Abstracts*). To determine the effect on the public a few demonstration have been carried out on various locations and to a variety of spectators. These demonstrations are described in section 6.3.

6.1 Frame Delay Test

The frames displayed in figure 6.1 represent the result of LinearDelay filtering at `rate=2` and the corresponding input video. Each frame has a dimension of 784 × 576 pixels. While the prior motion of the person has remained zero, he initiates an approximately linear motion in frame 1290. The effect reveals itself around frame 1360, in which a slight distortion of the top of the head can be observed. This delay of around 60 frames is motivated by the distance in the image between the top of the frame and the location of the head; a distance of 138 pixels. Due to `rate=2`, at frame 1290 + (138 / 2) = 1359, the scanline containing the first movement of the top of the head is processed. Similarly, the person in frame 1290 is 362 pixels in height. As a consequence, the feet in frame 1465 have not yet been distorted. The delay at that location (scanline 500) is 250 frames, which points to a frame prior to the initiation of motion.

6.2 Examples of Filter Effects

The LinearDelay filter produces floating bodies, as witnessed in figure 6.1. This effect is more strongly visualized by the series of frames in figure 6.2. Due to the continuous lateral motion

Figure 6.1: Result of the LinearDelay filter at `rate=2` *(right) on a source video (left). In top-to-bottom order, frames 1290, 1360, 1395, 1430 and 1465 from movie* `lineardelay1.avi` *are shown.*

Figure 6.2: Effect of the LinearDelay filter at `rate=2` *(right) on a lateral motion (left). The person exhibits a motion directed interactively leftwards and rightwards. Frames 200, 600, 750, 890 and 1000 from movie* `lineardelay1.avi` *are displayed in top-to-bottom order.*

Figure 6.3: Effect of the LinearDelay filter (at `rate=2` *right) on a pirouette motion (left). The person exhibits a rotation motion. Frames 550, 600, 650, 782 and 845 from movie* `lineardelay2.avi` *are displayed in top-to-bottom order.*

of the person in iteratively the leftward and rightward direction, a stretching effect is created. This emphasizes motion, most specially after the direction of motion is reversed.

A rotating motion, as presented in figure 6.3 causes a spiral effect. The sudden raise of hands, acted to created the pirouette silhouette, yields the hands and upper arms floating in space. This demonstrates a property of the filter: upward (or, from a image perspective, vertical) motion can result in multiple objects. Only if the displacement of the top scanline of the object is limited to `rate` will a temporary split not be created. This can easily be verified by considering, if a larger displacement, say `rate` + 1, takes place, at time t scanline s will contain a part of the object while only at time $t + 1$ scanline $s +$ `rate` is incorporated.

6.3 Public Demonstration

On the third of November the technical departments of the University of Groningen (R*u*G) promoted their studies by means of a faculty-wide information day. The aim of this event was to persuade prospective students in joining the R*u*G using attractive, stimulating and interactive demonstrations. One of the presentations featured the distorting effect of the interactive *LinearDelay* video filter. A camera and display device, strategically positioned near the main pathway of the demonstration area, provided interested visitors to interact with the filter.

Visitors generally reacted either reserved or passionately engaged the interactivity. The cause for this duality may be found in the reflexive property of the filter. The majority of this public consisted of adolescents, who often are insecure about their self-image. By presenting a direct - although transformed - feedback, the filter sometimes had a repulsive effect on these individuals. Nevertheless, the filter received much attention and enthusiasm from the general public, visitors and co-researchers alike.

Part III

Object Segmentation

Chapter 7

Introduction

"No pessimist ever discovered the secret of the stars, or sailed to an uncharted land, or opened a new doorway for the human spirit."

Helen Keller (1880-1968)

When one would ask a human to point out which part of an image comprises a moving object a flawless response would result. Even on a pixel level, a person could precisely indicate if it belongs to the object or the background. This also holds for objects that change in shape over time (e.g. by rotation) or with backgrounds that exhibit local motion (e.g. waving trees). Instead of reasoning on a flat image, the human visual system presents a decomposition of the scene into an arrangement of objects. Depth information is included, objects are tracking throughout the scene and it is even likely that contextual information, like sounds and previously stored memories of the perceived objects, is included into the recognition process.

What appears to be a trivial task for humans is an enormous challenge for a computer. Contextual information is usually unavailable or lacks sufficient abstraction. Depth information often is absent as it requires stereo-vision, which, if available, poses additional problems such as depth ordering and feature matching between the left and right image. Shape changes and background motion are also hard to incorporate into a model. The scene dynamics—local change in illumination and similarities in color between objects and the background—call for a model that utilizes the spatial context to strengthen its recognition. The difficulty computers have to flawlessly segment on high resolutions is caused by the lack of a complex model that combines all this information using a highly parallel system.

Among these problems, changes in brightness can cause the recognition to perform poorly. At low intensity (near black) or high saturation (near white) correct distinguishment is difficult to achieve. Sudden variations particularly trouble the recognition, caused by strong differences in colors between the observed data and the background reference. In an attempt to resolve the difficulties introduced by brightness, the research described in this thesis proposes invariancy to illumination.

In section 7.1 a general notion of image and video analysis is introduced, followed by a discussion on the application fields in which object recognition in videos plays an important role in section 7.2. The discussion in section 7.3 focusses on previous research conducted in relevant literature and section 7.4 elaborates on the research goals of the thesis.

7.1 Image Analysis

Object detection is part of the *image analysis* field. The general purpose of image analysis is to extract meaningful information from images. This task is generally defined as a threefold process, as is illustrated in figure 7.1. At the first layer, objects are segmented from the background. The segmentation is obtained by detecting object features such as contours (shape), colors and edges (appearance) and compare them to a stored model. Algorithms that ignore object characteristics have also been proposed, but often produce results of lesser quality. This is caused by the absence of domain information. Obviously, the knowledge of the shape and color of a car can boost the performance. However, domain invariance allows for a generic usage.

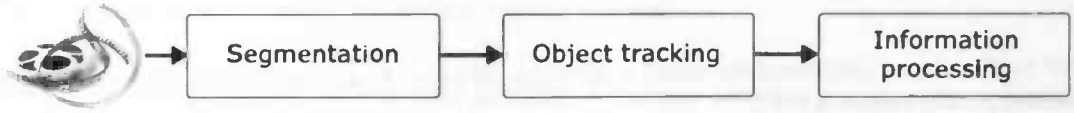


Figure 7.1: The process of image analysis can be divided into three parts, of which segmentation and object tracking are often intertwined.

At the second layer the segmented objects of interest, which are also denoted as *foreground objects*, are tracked throughout the video sequence. Tracking can be performed by matching the segmented foreground objects with a corresponding model. This process of association may involve a comparison of features, such as (again) shape and appearance. To mutually improve the performance, the tracking layer often provides feedback (e.g. probability of a perfect match) to the segmentation layer.

At the last layer the segmented and tracking objects are interpreted. This may be for any purpose and is highly domain dependent. One could locate the license plate of a tracked car and detect the license number, recognize odd behavior or analyze and predict the objects' trajectory. The yielding information is often presented to the user, although it may also be used by higher-level classification processes. Whatever its purpose, it should have been properly formatted to be processed easily.

7.1.1 Segmentation

Only the first step in image analysis is discussed in this thesis. Therefore, we will now only take a closer look at the process of segmentation. *Image segmentation* partitions an image into a set of disjoint regions. Each regions consists of pixels that have a relationship with each other: together they form an object, e.g. a car or person. To document the correspondence between pixel and object, segmentation methods typically produce layered masks. Each mask consists of binary values: 1 indicating that the pixel is part of the object and 0 the converse. The layered approach allows for independent processing of objects and enables depth ordering. To identify the layer a pixel belongs to, a label (an integer value) is stored at each pixel location. An example of a image segmentation can be found in figure 7.2.

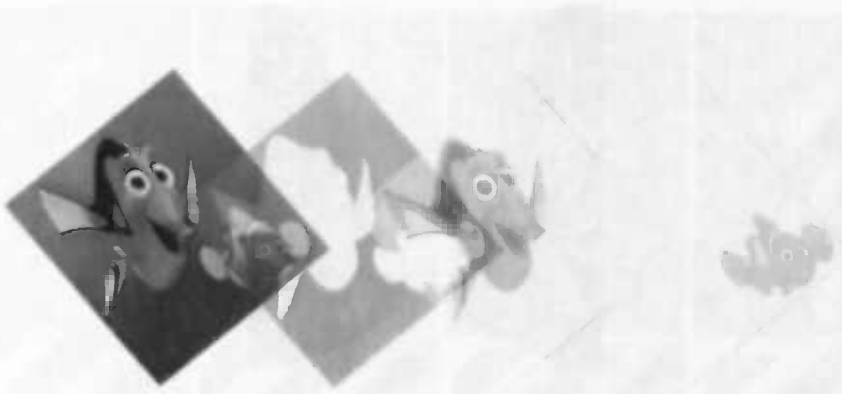


Figure 7.2: Example of an image segmentation. Each layer mask is an extraction from the image based on coherent pixels that together define an object. Background is usually represented by a mask too, although it is processed separately from the foreground layers.

Image segmentation is an *ill-posed problem* [Ter86]. The yielding solution is not always unique, if there is any. Consequently, a segmentation criterion is hard to define and optimal results cannot be ensured. To the contrary, a problem is a *well-posed problem* if

1. a solution exists;
2. the solution is unique;
3. the solution depends continuously on the data.

An example of ill-posedness is presented by figure 7.3. It presents two views: two black faces or a white vase. Segmentation can only be carried out if a definition of the object and the background is provided. The process of defining these conditions and constraints is termed *regularization*. Another cause of ill-posedness is the absence of a solution, which may occur when the object and background have similar appearance (e.g. in color or shape). In such a situation a decision cannot be made, as the pixels belong to both classes. Other main problems arising at image segmentation are due to noise, lack of contrast and boundary discontinuities.

The most common technique to obtain an image segmentation is by means of *thresholding*. A decision variable is computed based on the image statistics and salient object features. A pixel is then said to belong to the foreground if this variable exceeds a threshold value and is defined as background otherwise. Another approach is to determine the foreground objects by matching the expected object region with the object model. Such a model may be a parameterized by contour, edges, location of salient features, texture descriptors, etc. An overview of a large number of segmentation techniques is presented by [SK94].

In a video sequence objects are present in multiple successive frames. The background typically is static, with only minor fluctuations due to noise. Problems arise, however, if the background is subject to small movement, e.g. swaying leaves. A moving object presents the possibility to be segmented as foreground by utilizing the changing difference between its model and the background at each location. To detect this motion, three main approaches exist. *Temporal*

Figure 7.3: Do we see a vase or two faces? This is exemplary to the image segmentation problem. Without a definition of the object of interest the problem remains ill-posed.

differencing techniques compute the color intensity difference between two successive frames. Objects belong to the foreground if the difference is larger than some threshold. Although it is insensitive to gradual changes in brightness, this technique has a few important drawbacks. First of all, once an object has become static (i.e. exerts no motion) the temporal difference will no longer yield a foreground object. This effect also acts on homogeneous objects at low motion. Due to the small per-frame displacement the object overlaps at successive frames. The similarity of color throughout the object may cause pixels to be left undetected, yielding only the actual displacement (the contour) as foreground.

Another approach to object segmentation is presented by *background subtraction*. Instead of using the previous frame as reference, these techniques base the segmentation on a *background* or *reference frame*. This reference frame is often constructed prior to the start of the segmentation process. Usually, the first few (e.g. 100) frames in a video sequence produce an average of the background. These techniques assume initial *clean frames*: frames that do not include foreground objects. In the case such frames cannot be guaranteed, the background construction should model the optional presence of foreground pixels. An example of such an approach is presented by Calderara et al. [CMPC06]. During the detection phase objects are located due to their absence in the reference, which therefore produces a substantial difference. Thresholding this difference yields the final segmentation. Because the reference frame has been constructed at the start of the sequence, even gradual changes in brightness over time cause the segmentation to fail. To counter this, an adaptive background update could be employed or one could incorporate invariance to illumination into the model. The latter approach will be taken in this thesis.

Methods that compute the *optical flow* not only produce a segmentation, they also determine the motion vectors of the individual pixels. Optical flow techniques suffer from the *aperture problem*. Motion detection in a small local field, as occurs in the human visual system, is incapable of detecting the motion component parallel to a contour line or edge. This introduces ambiguity in the output as edges of different orientation and velocity can cause identical responses. Figure 7.4 shows a texture of diagonal lines which are observed to move in vertical direction, while the displacement actually is horizontal. A global smoothness constraint that assumes locally constant flow can be enforced to deal with this incapacity [LK81]. Such a global approach also enables a dense optical flow by predicting the flow information of homogeneous object by the motion boundaries. A local method, however, is less sensitive to noise [HS81].

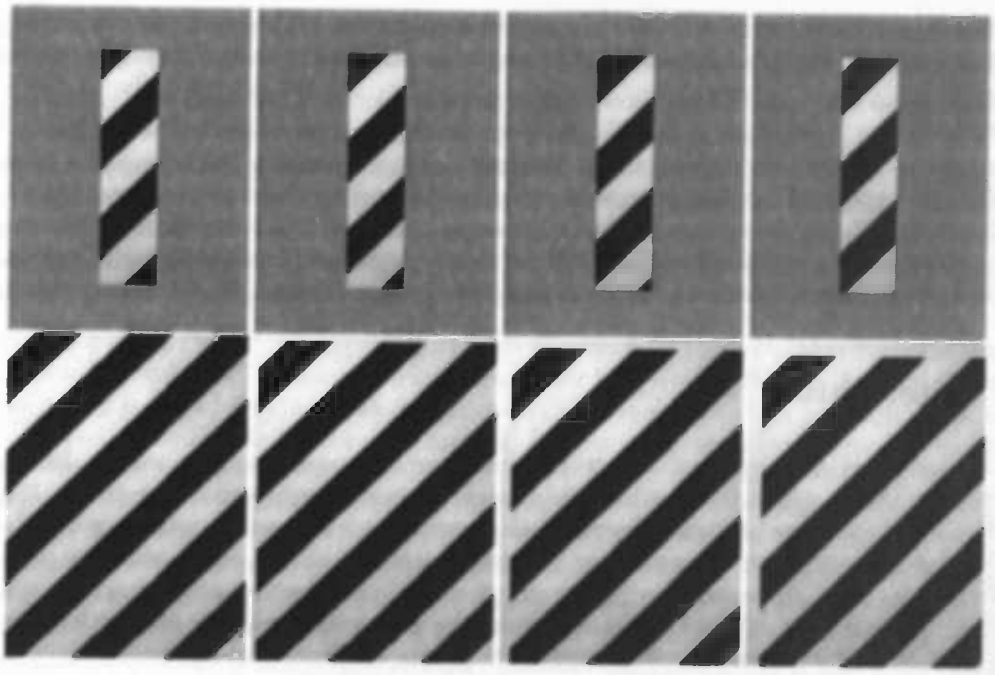


Figure 7.4: Illustration of the aperture problem: while the lines exert diagonal movement, horizontal or vertical motion is observed. This is caused by the small local field of perception, which is unable to observe motion parallel to contours. Only at a global level, when the object boundaries are taken into account, will the correct motion be perceived.

7.2 Applications

The information obtained from the segmentation and tracking of objects can be utilized in a variety of domains. This section briefly introduces the fields in which video analysis can be applied.

Video surveillance is one of the most familiar applications. Recent development of inexpensive hardware, offering the minimal computational requirements to process these applications, has led to broadly available systems. Using mostly real-time object tracking, video surveillance applications register trespassers on industry complexes, aggressive behavior at public sites, highway speeding and complex robotic tasks. Furthermore, with the segmented objects at hand, these applications lend themselves for further automatic processing of faces and license plates (recognition of perpetrators) and process monitoring.

Remote surgery and medical scanners have become important tools in modern medical science. These *medical systems* require applications that emphasize blood vessels, detect malignant tissue and locate miniscule injuries. Specialized recognition systems could provide this functionality by a 2D or 3D video analysis of the human body [WWdV03].

The current explosion of digital video data calls for automated classification to enable the retrieval among a vast amount of content. *Video indexing* techniques could provide the means to efficiently segment video portions. Video sequences are divided into scenes based on rapid global changes. The content of each scene is analyzed, often in combination with speech and

subtitle recognition [MfHCW02]. Representative elements of each scene, such as news-reader faces, are extracted and labels based on this content are attached to enable easy database searches.

Another important aspect of the recent urge for information is the need for fast video transmissions. Object segmentation can aid in better and faster *compression* of video data. By determining the objects and their motion, different coding strategies can be applied to various regions. Backgrounds exhibit only minor changes over time and can be stored by a simple model at low bit rate. Foreground objects, however, are most likely the focus of the scene and should likewise be modelled in detail. Their possibly rapid and complex motion require more sophisticated methods. Motion estimation has been incorporated in the MPEG standards.

7.3 Literature

The segmentation of foreground objects from a stream of images has been widely discussed by researchers. Many applications utilize the change of pixel intensity and/or color to determine the shape and locations of these objects. This approach is a necessary one, since often images captured at different time instances are the only source of information. Many segmentation methods have been proposed over the years, of which most include object tracking into their solution. For the sake of conciseness the overview presented below is limited to object segmentation, in line with our low-level approach. More elaborate surveys have been presented by Radke et al. [RAAKR05] and Karaman et al. [KGYS05]. Toyama et al. [TKBM99] listed a comprehensive overview of several background maintenance algorithms.

7.3.1 Image Difference and Parameterization

Object segmentation in videos by image thresholding was first achieved by simple temporal differencing [Ots79]. Following this rudimentary idea more sophisticated methods have been devised. *Change vector analysis* (CVA) is a generalization of simple differencing designed for multi-spectral images [Mal80]. It is based on a feature vector that records the pixel values for each spectral channel. The image difference is produced by subtracting two feature vectors. A drawback of these early methods is their use of a fixed threshold, which renders the segmentation a delicate process vulnerable to noise and variations in illumination.

An alternative approach is to classify objects based on their characteristics. This is often referred to as parameterization, as a set of parameters are computed to describe each object. Haley & Manjunath [HM99] have proposed a rotation-invariant texture recognition method that derives a feature set using modified Gabor filters at multiple resolutions. By comparing the feature sets between frames the objects can be matched and tracked. Edge detection also provides a solution to the segmentation problem. Gradient-based methods [SDC04], Laplacian or Difference of Gaussian (DoG) operators and Hough transform [KN95] have all been put to use to determine the object contours. A discussion about edge-based segmentation, among other topics on object tracking, is given by Lepetit & Fua [LF05].

7.3.2 Background Modeling

Background subtraction methods are based on a proper background model. Many researchers have proposed subtraction techniques that use a Gaussian density function to model the background [KCL+98]. A pixel is classified as foreground if the difference with the background

reference is not larger than a few standard deviations. In this literature, the mean and variance of the background are determined by computing the pixel statistics of the first frames. Hence, a Gaussian distribution models the probability of object detection for each pixel. To account for gradual changes in illumination, the means and variances can be updated at specific frames.

The Pfinder algorithm proposed by Wren et al. [WADP97] extended the Gaussian background model to multiple layers. Besides the background, a variable number of Gaussian distributions are included to model the different foreground objects. The pixel is assigned to the model of closest Mahalanobis distance. To update the model statistics (mean and variance), dynamic blob models are fitted onto the foreground pixels for each object.

Stauffer & Grimson [SG00] have generalized this approach by allowing also the background to be modelled by multiple Gaussian distributions. Their idea was to allow background dynamics into the scene, such as swaying trees and surface disturbances of cloth and liquids. A weight value determines the evidence for each model. Each pixel is compared against a fixed number of Gaussian models and matched to the closest model based on Euclidean distance. If the distances for all existing models exceed a fixed value (in their example they use a factor of 2.5 standard deviations), the least likely model for that pixel is discarded and re-initialized using the current pixel value and a small evidence. The models are sorted based on the evidence value and normalized by their standard deviation. To classify a pixel as background the evidence values are accumulated until the threshold value is reached. As a result, repetitive background motions can be effectively described. Another possibly desirable effect is that static objects become incorporated into the background. When again in motion, they are again segmented as foreground.

7.3.3 Illumination Invariance

Despite the recent advances in robust object detection, illumination change remains one the most important challenges in real-life applications. Fast changes in illumination produce significant signal variations, which are usually regarded as irrelevant. To account for these changes *illumination invariance* has become a research focus.

One of the simplest approaches to illumination invariance is the Color Angles method of Finlayson et al. [FCF96], which is based on a comparison between colorspace angles in the observed and reference image. The theory is based on the observation that, for narrow-band cameras, illumination induced color shifts correspond to simple scaling of color channels. The color vectors of each pixel are subtracted from the mean (i.e. the reference image) and normalized in length. The angles are constructed from the inverse cosine of the RG, RB and GB color products. The same derivation of angles is obtained from an edge image. A linear mapping by a diagonal matrix transform detects the amount of illumination change.

Prepatory to their Linear Color method, Drew et al. [DWL98] argue that, on approximation, difference in illumination of any object can be described by a linear relationship. Previous assumptions on this linearity were limited to single colored objects only. Hence, they argue that a full matrix transformation is required to model these changes. Multiple sources are accounted for by modelling the illumination as an accumulation of light rays. The color relationship between an observed and a reference image is estimated by a Discrete Cosine Transformation (DCT).

Toth et al. [TAM00] presented a motion detection algorithm that reduces the effect of rapid illumination changes by a homomorphic image filter. This simplified model, as they state, models

the image intensity as being produced by incident light, reflected by object surfaces. Toth et al. argue that illumination changes gradually over time. Furthermore, they state, reflectance consists of spatially high-frequency components. Illumination and reflectance have a multiplicative relation and can thus be separated by taking the logarithm of the intensity and applying high-pass filtering.

Drew et al. [DWL98] also presented a comparison between a number of techniques specially designed for illumination invariance. An alternative approach, called *color constancy*, derived from the human perception of color, will be discussed in section 8.4.1. A survey by Barnard [Bar98] discusses color constancy for the task of object detection.

7.4 Research Question

The research described in this part will deal with the segmentation of foreground objects from monocular videos. More in particular, a model is presented that limits the effect of brightness variation. To this ideal a foreground-background segmentation model is proposed aimed at illumination invariance. The discussion will concentrate on the process of segmentation—a high-level analysis of the produced segmentation is not the current focus. The main research question is the following:

> How could a foreground-background segmentation algorithm be developed that is, or approximates, illumination invariance such that changes in brightness do not affect the segmentation (and optionally tracking) performance of the algorithm in realistic scenarios?

To answer this question the following topics need investigation:

1. What is the behavior of illumination and noise in signals processed by a CCD camera?
2. How could vector collinearity be used as a foundation of illumination invariant foreground-background segmentation?
3. Which issues should be resolved to obtain an implementation of this algorithm?
4. What is the performance of the algorithm in the presence of strong illumination changes?
5. How is the algorithm ranked in comparison with competitive tracking algorithms based on the PETS 2001 dataset?

7.4.1 Trajectory

The remainder of this thesis part is concerned with addressing the topics introduced in the previous section. Chapter 8 focusses on the properties of light and presents a common color model. Furthermore, the effects of noise on the video data at varying illumination is addressed. An analysis of the behavior and distribution of noise would contribute to an effective model that reduces their influence. As the human brain is known as the optimal recognition system, the human visual system is briefly analyzed in the context of perception.

The collinear vector model presented in chapter 9 incorporates illumination invariancy on a principled basis. It utilizes probability theory to obtain a reliable segmentation. Knowledge of the human visual system is included to allow for low intensity processing.

Chapter 10 discusses the implementation of the proposed model. Experiments and results derived from a video segmentation dataset are elaborated in chapter 11. The model is evaluated on the basis of low brightness and for the purpose of tracking. Chapter 12 concludes this part by evaluating the model and results and proposing suggestions for improvement.

Chapter 8

Illumination, Color and Perception

"When the doors of perception are cleansed, man will see things as they truly are, infinite."

William Blake (1757-1827)

According to the Oxford Dictionary [Weh05] the term *illumination* origins from the Latin noun *lumen*, or light. It is defined as the action or process of lighting an object or scene. Static, non-changing, illumination produces a fixed experience: the perceived colors do not change over time. Realistically, however, illumination is a process dependent on external conditions. Sunlight is produced in a periodic fashion, changing in color and intensity at dawn, daylight, dusk and night, shadows of clouds and objects cause a local decrease of illumination and the reflectance of objects depend on surface material. To understand the effects of these processes the concepts of illumination, color and perception will be discussed in more detail in this chapter. A short explanation of illumination in section 8.1 and noise in section 8.2 initiates this discussion, followed by an inspection of the promising HSV color model and the ultimate example, the human visual system, in subsequent sections.

8.1 Illumination

The variety of illumination conditions hinder the reliable segmentation of objects from videos. Color is recorded at different values in light and dark scenes. E.g., the color of the left car in figure 8.1 differs from that of the right car, while only the intensity of the illumination has changed. One could argue that this problem can be solved by utilizing the normalized difference between the object and its surrounding. In fact, we will see in the discussion on color constancy in section 8.4.1 that spatial information is intensively used to normalize the effect of illumination. We could also inspect the color values of a pixel within the foreground region, both in the current frame I and reference frame J:

$$d_i = \frac{I_i - J_i}{J_i} \quad \texttt{with } i \in \{\texttt{R}, \texttt{G}, \texttt{B}\} \tag{8.1}$$

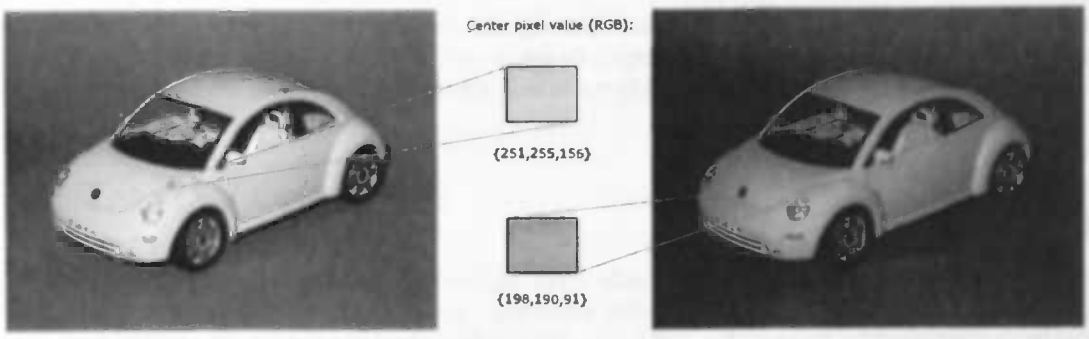


Figure 8.1: An example of the effect of illumination on the color values. Different levels of brightness yield significant differences in the recorded color values.

The reference frame—in this particular situation a model of the background—may be subject to illumination intensities that differ from the current frame of observation. The reference frame could have been generated many frames ago, after which the illumination has changed. An option would be to use the previous frame as reference. Unfortunately, a drastic change in illumination (e.g. by a sudden cast of shadow or a light switching on or off) could still produce a significant difference in intensity. Also, the amount of illumination by artificial light sources depends on the distance of the illuminated surface to the source. Foreground objects may therefor have a stronger illumination (if the light source is situated at the same side of the surface plane) or weaker illumination than the background—again compare the left and right car in figure 8.1.

Illumination is usually modelled by a positive multiplicative factor that modulates the signal. If we define $\mathbf{s}$ as the normal signal[1] the observed value $\mathbf{x}$ can be described as

$$\mathbf{x} = k \cdot \mathbf{s} \tag{8.2}$$

with k the illumination factor. The definition indicates that the illumination is equal among the color channels. This is a naive assumption: a red light source would not produce a change in illumination in the blue end of the frequency spectrum. A more realistic approach would be to model each of the color channels separately:

$$\mathbf{x} = \mathbf{l} \cdot \mathbf{s} \tag{8.3}$$

Another important observation is that illumination typically behaves as a global process. It affects the brightness of the entire scene, or at least in large regions. Due to this spatial low-frequency behavior it is reasonable to assume that the illumination is almost constant within a small neighborhood.

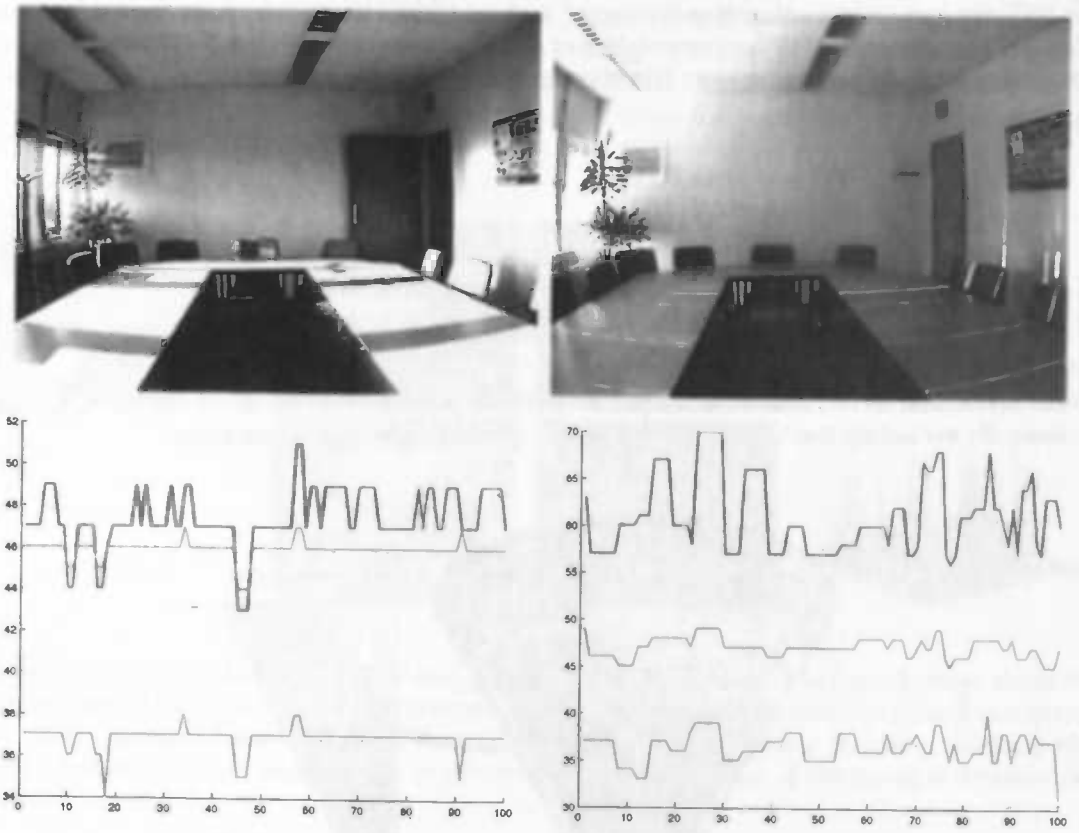
8.2 Image Noise

In the process of image acquisition by means of a CCD imaging system, two types of noise have an effect on the final color values: additive and multiplicative noise. Both types are produced by the sensor system of a CCD chip. Each chip consists of sensor elements sensitive to

[1]The normal signal is the standard, or norm, value of the signal.

a range of colors, which count the number of incident photons within their receptive range of wavelengths. For example, a red-sensitive element would fire only if a photon of wavelength between 575 and 700 nanometer reaches the sensor surface. This photon counting is subject to sampling errors, due to the random intervals at which photons arrive, and this worsens the effect of noise when less photons are caught.

Thermal disturbance between sensor elements on a CCD produce charges similar to those caused by incident photons. This is called *additive noise*, as it generates additional electronic discharges that add to the total amount of observed photons. Additive noise is also caused by the on-chip output amplifier. In contrast, the variation in sensor sensitivity results in multiplicative noise.



(a) Bright scene.

(b) Darkened scene.

Figure 8.2: Variance of the R,G and B components of a pixel at location (450,340)—top-right dark stripe on the table—for a staticly illuminated scene over 100 consecutive samples. The blue component shows a higher sensitivity to noise than the other components.

Color also has an effect on noise. CCD sensors are biased towards the red end of the color spectrum. To exclude the undesired infrared signals to be included into the acquisition, a filtration layer is placed over the sensors. However, this reddish sensitivity induces a relative insensitivity at the blue side of the spectrum. As of consequence, the blue channel will contain higher levels of noise compared to the green and red channels. Indeed, an analysis of the pixel compo-

nents of two image sequences at different brightness intensity display a larger variation among the observed color intensities for the blue channel (figure 8.2(a)). This effect increases, relative to the other channels, along with the reduction of brightness of the scene (c.f. figure 8.2(b)).

8.3 The HSV Colorspace

The HSV color model has been designed to more effectively model the individual aspects of color. It separates color from brightness. The color defined by the red, green and blue components in the RGB colorspace can be transformed to the HSV colorspace, and vice versa [Smi78]. In HSV, the hue (color), saturation (vibrancy) and value (brightness) can be controlled independently. To produce the HSV representative of a RGB defined color, an isomorphism exists that produces the values according to a bijective map[2]. The mapping follows

$$\begin{aligned} V &= MAX \\ S &= \frac{MAX-MIN}{MAX} \\ H &= \begin{cases} 0 + \frac{60(G-B)}{MAX-MIN} & \texttt{if } R = MAX \\ 120 + \frac{60(G-B)}{MAX-MIN} & \texttt{if } G = MAX \\ 240 + \frac{60(G-B)}{MAX-MIN} & \texttt{if } B = MAX \end{cases} \end{aligned} \tag{8.4}$$

with MAX and MIN defined as respectively the maximum and minimum of the R, G and B values. By excluding the V-component one can ignore brightness fluctuations.

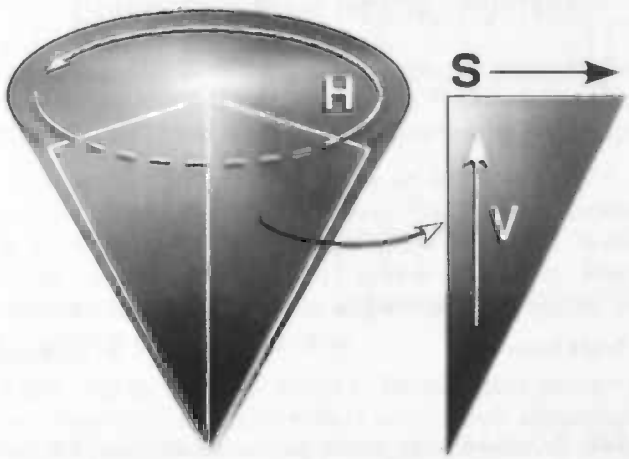


Figure 8.3: The HSV colorspace can be represented by a cone. The color type (e.g. yellow, purple, red) is modelled by the radial value of H on a circle. The saturation S and brightness V can be represented by a triangular region which, combined with H, form a cone.

[2] An isomorphism, from the Greek words *isos* (equal) and *morphe* (shape), is a unique bidirectional mapping of values onto each other in different domains that preserves the topological ordering.

A disadvantage of the HSV model is that the saturation is labile near zero and at singularity. This lability can be observed from the conal representation of HSV in figure 8.3. When the saturation approaches zero ($S \approx 0$) and at singularity ($S = 0$)—the color is located close to or on the central axis of the cone—discrimination from other colors becomes increasingly difficult. Furthermore, colors near $S = 0$ are much more vulnerable to noise. Small fluctuations may produce distinctly different colors.

Another problem with HSV arises at the edges of objects. Figure 8.4 shows the mean and standard deviation of the B (from RGB) and H (from HSV) component. The variation of the hue (bottom row) is significantly larger than the blue component (top row), which is representative for the maximum variation among the RGB components[3]. The large standard deviation of the hue is most apparent at locations of high texture, which contain many edges. Noise at edges is induced by tiny camera vibrations and 'photon leaking'. The effects are amplified by the non-linear computation of the hue component from RGB. If brightness (and optionally saturation) is to be ignored the large variation of color values may cause the segmentation system to be less reliable at edge and high texture locations.

Figure 8.4: Statistical behavior of HSV and RGB based on 100 samples. The top-left image shows the average image of the combined RGB components. The first row contains the statistics of the B component from RGB. The second row holds the H component from HSV. For each row the mean,' intensity plot of the standard deviation (scaled to its maximum value) and a 3D plot of the standard deviation are displayed.

The HSV model is not particularly suitable for color matching of noisy images. By separating color from brightness, the model ignores the effects that noise may cause. The red, green and blue tonal components at which colors are registered by digital cameras are affected by noise. These RGB values therefor do not reflect the true colors of the scene, but change over time according to a camera-specific, Gaussian distribution. Hence, splitting the RGB values into color and brightness components would produce significant color differences between recordings of a fixed scene.

[3]Because HSV is a transformation of RGB the distribution of the color components are not lost; the values are only redistributed according to the nonlinear transform rules. The statistics should prevail. Indeed, the distribution of S and V correct for the large deviation of H values.

8.4 The Human Visual System

The human visual system is responsible for the extraction of information from visible light within the perceived view. Its functionality is often compared to that of a camera, since both focus light emitted from an object onto a light-sensitive medium. For humans and most animals capable of color vision this medium (also known as retina) consists of rods and cones [Hub95].

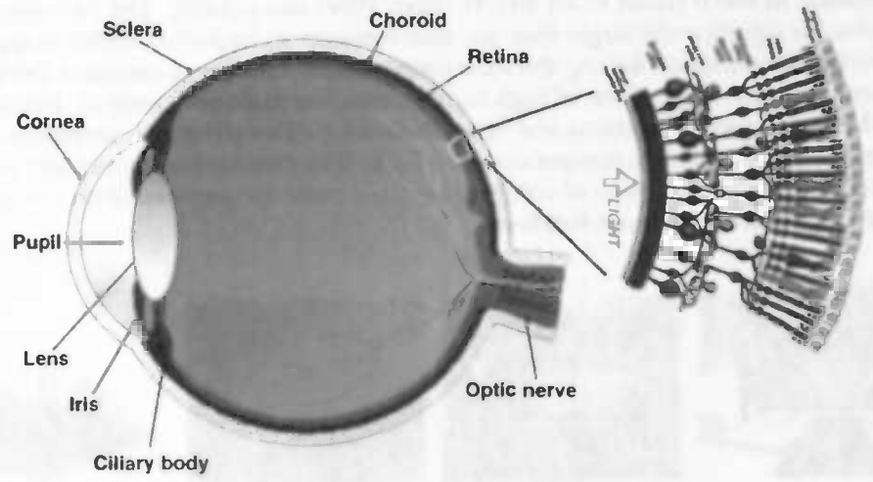


Figure 8.5: A schematic overview of the human eye. Light is focussed by the lens and projected onto the retina. The cone cells record the color information at daylight. Rods are sensitive to low illumination.

Perception of color is provided by *cone cells*, which typically come in three types: blue, green and red[4]. Cone cells are larger and less numerous than their rod counterparts, and are located at the center of the retina. Due to their high response, cone cells sense rapid temporal changes.

Rod cells are highly sensitive to light and can excite on a single photon. As such, they are responsible for night vision. To amplify vision at very low light intensities, rod cell signals are converged on a neuron. This integration also provides the visual system with sensitivity to peripheral movement, but causes a reduction of spatial resolution. Rod cells have a slow response (about 100 milliseconds), which further improves their light sensitivity. The drawback is a reduced ability to perceive temporal changes.

The specialization of rod cells to very low levels of illumination present a suitable component for low intensity object detection, at which noise causes much disturbance. The lower spatial resolution of rods allow for performing spatial averaging methods without much loss of quality.

8.4.1 Color Constancy

The human visual system is capable of observing a constant color of objects under varying illumination conditions. Both at daytime, in bright white light, and in the evening, when sunlight

[4]People suffering from color blindness may have four types of cones.

has a reddish color, is grass perceived as green. This effect is known as *color constancy* and assists us in accurately identifying objects. Land & McCann [LM71] indicated that light surrounding an object is used to obtain color constancy. They proposed the *retinex algorithm*, demonstrating that color perception follows a twofold process. First, the brightness recorded by each conal system is normalized in parallel. Secondly, comparisons between the conal system then produces chromatic contrast.

Brightness normalization is performed for each conal system independently by specialized neurons in the primary visual cortex. These so-called double-opponent cells operate as on-center off-surround inhibitors. Signals emitted from the center of a neuron's receptive field exhibit the response, while retinal surround signals have an inhibitory effect. This is in fact a subtraction of center and surround signals that eliminates absolute color values.

To achieve chromatic contrast the normalized signals from the conal systems are compared against each other. Two cones register short wavelength (blue) and long wavelength (yellow) respectively. The latter is splitted into yellow-red and yellow-green sensitivity, introducing the third cone. To determine e.g. blue, a specialized neuron receives a normalized excitatory input from the short wavelength conal system. The long wavelength system delivers a normalized inhibitory input. Subtraction of these signals produces the response of the neuron. A similar comparison is made between the yellow-red and yellow-green systems.

Brainard & Wandell [BW86] have shown that the retinex algorithm fails to model the human visual system. Nevertheless, color constancy can be exploited in assistance with illumination invariance methods.

Chapter 9

Collinear Vector Model

"Mathematics has the completely false reputation of yielding infallible conclusions. Its infallibility is nothing but identity. Two times two is not four, but it is just two times two, and that is what we call four for short. But four is nothing new at all. And thus it goes on and on in its conclusions, except that in the higher formulas the identity fades out of sight."

Johann von Goethe (1749-1832)

Mester et al [MAD01] have proposed an illumination invariant model in the context of decision theory. Their approach operates in RGB space, although any colorspace can be used with minor changes to the parameters. Their model operates on a local neighborhood to exploit the spatial low frequency of illumination. This model forms the basis of the Collinear Vector Model (CVM) discussed in this chapter. The foundation of the CVM is presented by the collinearity criterion in section 9.1. To find the optimal solution to the hypothetical collinear vectors the eigendecomposition of the combined vectors is determined in section 9.2. To simplify this decomposition, a generalization towards singular value decomposition is discussed (section 9.3). This yields an initial decision rule (section 9.4). To improve the model the hypothesis testing is fitted into a Bayesian framework (section 9.5), modelling the probabilities by a Gibbs/Markov random field (section 9.6).

9.1 The Collinearity Criterion

Assume a reference image B exists, which could have been constructed as a model of the background by accumulating the first N frames:

$$B_C(x,y) = \frac{1}{N}\sum_{i=0}^{N-1} I_C^i(x,y) \tag{9.1}$$

where x and y are the image locations and C is a color component (e.g. $C \in \{R,G,B\}$) and I^i denotes the i^{th} image in an image sequence.

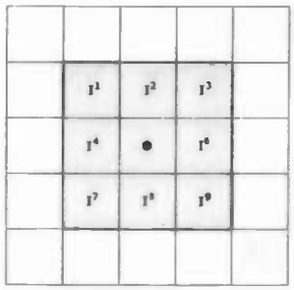


Figure 9.1: The feature vectors W_I and W_B are constructed from the multi-channel pixel values in a 3×3 neighborhood around the center (denoted by the $\bullet$).

In the presence of additive noise an observed signal O can be described by the unknown true signal s and a noise factor ϵ. The reference image B can thus be defined as the average of the observed signals: $B = s' + \epsilon_1$. In the absence of foreground objects, a static scene would yield signals that are close to the average: $s' \approx s$. According to equation 8.3, the effect of variation in brightness can be modelled by a multiplicative vector. Compared to the reference image, an observed image can thus be described as $I = l \cdot s + \epsilon_2$.

Noise is assumed to be spatially independent (see section 11.4 for a discussion). Furthermore, as illumination behaves as almost constant in a small neighborhood, a local window can exploit these advantages. For a pixel at (x, y) a window $W(x, y)$ is constructed from the pixels within a 3×3 square neighborhood (figure 9.1):

$$W_I = [\ I^1_{C1}\ \ I^1_{C2}\ \ I^1_{C3}\ \ I^2_{C1}\ \ I^2_{C2}\ \ I^2_{C3}\ \ \cdots\ \ I^N_{C1}\ \ I^N_{C2}\ \ I^N_{C3}\ ] \tag{9.2}$$

$$W_B = [\ B^1_{C1}\ \ B^1_{C2}\ \ B^1_{C3}\ \ B^2_{C1}\ \ B^2_{C2}\ \ B^2_{C3}\ \ \cdots\ \ B^N_{C1}\ \ B^N_{C2}\ \ B^N_{C3}\ ] \tag{9.3}$$

with $C1$, $C2$ and $C3$ color components and N the number of pixels in the local window. Both the reference vector W_B and the observation vector W_I are constructed in this way, based on the images B and I respectively.

To decide if the two feature vectors W_I and W_B are subject to variation in illumination they can be tested for collinearity. Two vectors are defined collinear if they are parallel. Collinear vectors

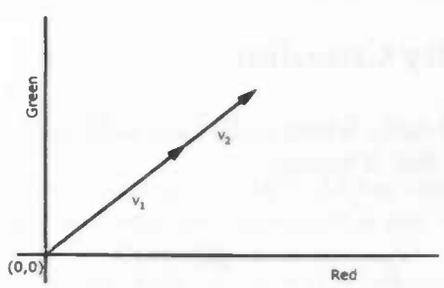


Figure 9.2: Two vectors v_1 and v_2 are collinear if the locations to which they point lie on the same line in combination with the origin.

are scalar multiples of each other (figure 9.2) and have the same origin. Collinearity thus models the multiplicative nature of illumination. In case two vectors lack this collinear property they can be regarded as different in color, and hence indicate the presence of a foreground object.

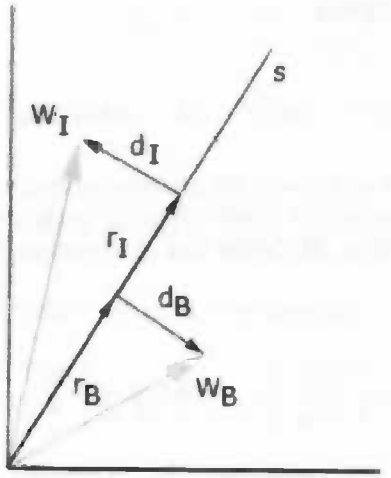


Figure 9.3: *Two-dimensional representation of the vector collinearity. Identical signals deviate by noise effects from the true signal s by a distance d.*

Mester et al. employ a hypothesis test to determine whether change is due to illumination or due to an object. They define the *null hypothesis* H_0 as whether W_B and W_I are collinear. The *alternative hypothesis* H_1 holds when dissimilarity between these feature vectors is statistically significant. Unfortunately, in the presence of noise perfect collinearity is hard to attain and one has to allow for some deviation. When Gaussian noise is assumed, the unknown "true signal" s can be estimated by minimizing the orthogonal distance of the two vectors (figure 9.3):

$$D^2 = |d_I|^2 + |d_B|^2 \tag{9.4}$$

Let us represent any of the distances d_I and d_B by d_*. Each distance can then be defined by

$$|d_*|^2 = |W_*|^2 - |r_*|^2 \tag{9.5}$$

with the vector $\mathbf{r}_*$ as the projection of W_* onto s. The vector $\mathbf{r}_*$ can be found by solving

$$|\mathbf{r}_*| = ||W_*| \cdot \cos\varphi_*| = |W_*^\top \cdot \mathbf{s}| \tag{9.6}$$

with $|\mathbf{s}| = 1$. As this may not be straightforward, appendix C provides a more elaborate deduction. Substituting $|\mathbf{r}_*|$ then yields

$$|d_*|^2 = |W_*|^2 - |W_*^\top \cdot \mathbf{s}|^2 \tag{9.7}$$

and the collinear deviation becomes

$$D^2 = |W_I|^2 + |W_B|^2 - |W_I^\top \cdot \mathbf{s}|^2 - |W_B^\top \cdot \mathbf{s}|^2 . \tag{9.8}$$

Unfortunately, as the signal s is unknown, the problem of finding the minimum distance cannot be solved in current form. However, $|W_I^\top \cdot \mathbf{s}|^2 - |W_B^\top \cdot \mathbf{s}|^2$ boils down to computing the correlation matrix of the vectors W_I and W_B. To clarify this observation, let us form the $2 \times N$ matrix M with

$$\mathbf{M} = \begin{bmatrix} W_I^\top \\ W_B^\top \end{bmatrix}, \qquad \mathbf{M} \cdot \mathbf{s} = \begin{bmatrix} W_I^\top \cdot \mathbf{s} \\ W_B^\top \cdot \mathbf{s} \end{bmatrix}. \tag{9.9}$$

Expansion of $\mathbf{M} \cdot \mathbf{s}$ produces

$$|\mathbf{M} \cdot \mathbf{s}|^2 = \mathbf{s}^\top \cdot \mathbf{M}^\top \cdot \mathbf{M} \cdot \mathbf{s} = |W_I^\top \cdot \mathbf{s}| + |W_B^\top \cdot \mathbf{s}| \tag{9.10}$$

which yields the distance measure

$$D^2 = |W_I|^2 + |W_B|^2 - \mathbf{s}^\top \cdot \mathbf{M}^\top \cdot \mathbf{M} \cdot \mathbf{s} \tag{9.11}$$

9.2 Eigendecomposition

The task of finding the collinearity of two vectors, in the presence of noise in the system, can be stated as finding the minimum distance D^2 of these vectors to the unknown "true signal". Since W_I and W_B are constant for an image at time t, the distance D^2 can be found by maximizing $\mathbf{s}^\top \cdot \mathbf{M}^\top \cdot \mathbf{M} \cdot \mathbf{s}$:

$$\mathbf{s}^\top \cdot \mathbf{M}^\top \cdot \mathbf{M} \cdot \mathbf{s} \longrightarrow \max \qquad \text{with } |\mathbf{s}| = 1 \tag{9.12}$$

Due to the construction of M from two vectors, $\mathbf{M}^\top \cdot \mathbf{M}$ has rank 2[1]. Therefore, the $\mathbf{M}^\top \cdot \mathbf{M}$ square matrix can be decomposed into two eigenvectors and corresponding eigenvalues [Azz96]. This eigendecomposition is of the form

$$\mathbf{V}^\top \cdot A \cdot \mathbf{V} = \Omega \tag{9.13}$$

[1]The rank defines the maximum number of linearly independent columns or rows of a matrix.

where A takes the place of the decomposed matrix ($A = \mathrm{M}^\top \cdot \mathrm{M}$), V is a square matrix of eigenvectors

$$V = \begin{bmatrix} X_1 & X_2 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \tag{9.14}$$

and Ω is a diagonal matrix of eigenvalues

$$\Omega = diag(\lambda_1, \lambda_2) = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}. \tag{9.15}$$

The eigenvalue equation in formula 9.13 can be rewritten as

$$A \cdot \mathbf{V} = \Omega \cdot \mathbf{V} \tag{9.16}$$

which shows that the eigenvectors are the characteristic vectors of matrix A and the eigenvalue define the characteristic lengths of these vectors.

Indeed, the eigenvectors are orthonormal and represent the axis that best fit the co-distribution of W_I and W_B. The eigenvalues are proportional to the variances along the principal axis. Due to construction of A from multiplication of M with itself, the matrix is rank deficient and the eigendecomposition simplifies to two eigenvectors and eigenvalues. The eigenvector corresponding to the largest eigenvalue represents the best estimate of the 'true signal' s. Furthermore, since the vectors of length d_I and d_B are orthogonal to s, the smaller eigenvalue can be used to compute D^2.

$\mathrm{M}^\top \cdot \mathrm{M}$ is the matrix of the observed vector W_I and reference vector W_B. Since these vectors consist of positive real values (the vector origin is located at $(0, 0)$), the matrix itself will not hold negative numbers. As a result, the smallest non-zero eigenvalue will represent the difference in similarity between the two data vectors.

9.3 Singular Value Decomposition

The eigendecomposition of the $\mathrm{M}^\top \cdot \mathrm{M}$, which is of size $3N \times 3N$, requires iterative numerical techniques[2], which unfortunately is computationally expensive (especially if applied for each pixel).

A generalized approach of finding the characteristic vectors of a matrix is provided by the Singular Value Decomposition (SVD) [Dep88]. Suppose the $m \times n$ matrix A consist of real or complex values. Then there exists a factorization of the form

$$\mathbf{A} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^* \tag{9.17}$$

[2]Generally it is almost always impossible to solve a high polynomial equation analytically (e.g. with the largest component being 6).

where $\mathbf{U}$ is a $m \times m$ unitary matrix, Σ is a $m \times n$ diagonal matrix and $\mathbf{V}^*$ is the conjugate transpose of a $n \times n$ unitary matrix. The columns of $\mathbf{U}$ are called *left singular vectors* and form an orthonormal basis for $\mathbf{M}$. Likewise, the columns of $\mathbf{V}$ are called *right singular vectors* and form an orthonormal basis for the decomposition of $\mathbf{M}$. The values on the diagonal of Σ are called the *singular values* and operate as scalars by which each left singular value is multiplied to produce the right singular values. For each singular value $s \in diag(\Sigma)$ produced by the singular value decomposition of matrix $\mathbf{A}$ holds

$$\mathbf{A} \cdot \mathbf{v} = \sigma \cdot \mathbf{u} \tag{9.18}$$

where $\mathbf{u} \in \mathbf{U}$ and $\mathbf{v} \in \mathbf{V}$ are the singular vectors. This equation resembles the eigendecomposition from equation 9.16. If the singular values are real and non-negative it can indeed be shown that the singular value decomposition coincides with the eigendecomposition:

$$\mathbf{A} = \mathbf{V} \cdot \Omega \cdot \mathbf{V}^\top \tag{9.19}$$

Furthermore, since we defined $\mathbf{A} = \mathbf{M}^\top \cdot \mathbf{M}$, the following two relations hold:

$$\begin{aligned} \mathbf{M}^\top \cdot \mathbf{M} &= \mathbf{V} \cdot \Sigma^\top \cdot \mathbf{U}^\top \cdot \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^\top \\ &= \mathbf{V} \cdot (\Sigma^\top \cdot \Sigma) \cdot \mathbf{V}^\top, \qquad (9.20) \\ \mathbf{M} \cdot \mathbf{M}^\top &= \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^\top \cdot \mathbf{V} \cdot \Sigma^\top \cdot \mathbf{U}^\top \\ &= \mathbf{U} \cdot (\Sigma \cdot \Sigma^\top) \cdot \mathbf{U}^\top. \qquad (9.21) \end{aligned}$$

Consequently, the squares of the non-zero singular values of $\mathbf{M}$ are equal to the non-zero eigenvalues of both $\mathbf{M}^\top \cdot \mathbf{M}$ and $\mathbf{M} \cdot \mathbf{M}^\top$. Since $\mathbf{M}$ contains real, positive values produced from the Euclidean vector lengths, finding the minimum orthogonal distance D^2 thus amounts to finding the smallest non-zero eigenvalue of the 2×2 matrix $\mathbf{M} \cdot \mathbf{M}^\top$, which can be computed in closed form without the use of iterative numerical techniques[3].

9.4 Decision Rule

As shown in the previous sections, the collinearity problem can be solved by finding the eigendecomposition of the matrix $\mathbf{M} \cdot \mathbf{M}^\top$ for the smallest non-zero eigenvalue. This yields a value $\mathbb{D}^2$ similar to the sought optimal distance ($\mathbb{D} \sim D$). Substituting the eigenvalue λ by $\mathbb{D}^2$, the minimum can be found by computing

$$\det\left(\mathbf{M} \cdot \mathbf{M}^\top - \mathbb{D}^2\mathbf{I}\right) = \det\begin{bmatrix} fore - \mathbb{D}^2 & cross \\ cross & back - \mathbb{D}^2 \end{bmatrix} \tag{9.22}$$

[3]The left singular vectors in $\mathbf{U}$ are the eigenvectors of $\mathbf{M} \cdot \mathbf{M}^\top$ and the right singular vectors in $\mathbf{V}$ are the eigenvectors of $\mathbf{M}^\top \cdot \mathbf{M}$, but this does not provide us with additional insight into the task of finding an optimal solution for D^2.

where $fore$, $back$ and $cross$ are the auto- and cross-multiplications of the vectors W_I and W_B in matrix M (see equation 9.9):

$$\begin{aligned} fore &= W_I \cdot W_I^\top \\ cross &= W_I \cdot W_B^\top \\ back &= W_B \cdot W_B^\top \end{aligned} \tag{9.23}$$

Expansion of equation 9.22 finally results in

$$\mathbb{D}^2 = \frac{1}{2} \cdot \left(fore + back - \sqrt{(fore - back)^2 + 4 \cdot cross^2} \right) \tag{9.24}$$

The effect of $\mathbb{D}$ can be understood by inspecting its value for some color vector combinations. If we assume two perfectly similar images I and B not interfered by noise, the color vectors for the I and B are equal. Consequently, the local feature vectors obey $W_I = W_B$ and therefore $fore = back = cross$ (see formula 9.23). Substituting these equalities in formula 9.24 yields $\mathbb{D} = 0$. Adding noise to the similar color vectors diverts W_I from W_B and causes $fore \neq back \neq cross$. With increasing dissimilarity between the color vectors, the distance measure $\mathbb{D}$ increases sublinearly (according to the square root).

9.5 Bayesian Estimation

The current decision rule uses prior knowledge (the reference frame) to predict the labelling of a pixel. However, as foreground objects have the tendency to be compact and smoothly shaped, one can utilize this information to improve the classification performance. For example, assume a pixel has not been classified as foreground despite that it belongs to a foreground object. This may be due to similarity in color between the reference frame and the current observation for this particular pixel. If the neighboring pixels are also part of the foreground and are classified correctly, the posterior knowledge of their classification could be exploited to strengthen the probability of our misclassified pixel. Likewise, false positives (caused by noise) could be removed by weakening their probability of belonging to the foreground by using the labelling of correctly classified background pixels in the surrounding.

The best classification of the pixel label L providing that the minimum distance estimate $\mathbb{D}^2$ can be found by employing a *maximum a-posteriori probability* (MAP):

$$\hat{L} = \texttt{argmax}_{l \in \{L_f, L_b\}} \; P(l|\mathbb{D}^2) \tag{9.25}$$

where L_f is a foreground label and L_b a background label. The MAP produces a quality measure based on both how well the segmentation labels conform to the observed data and how well they conform to the prior expectations [Bov00]. Our binary decision problem, in which a decision has to be made between the null hypothesis H_0 (background) or the alternative hypothesis H_1 (foreground) reduces this maximization[4]

[4]The simplification of the MAP problem prevents the use of expensive iterative methods, such as the Expectation-Maximization or conjugate gradient method.

$$\hat{L} \;:\; \frac{P(L_f|\mathbb{D}^2)}{P(L_b|\mathbb{D}^2)} \;\underset{b}{\overset{f}{\gtrless}}\; T \tag{9.26}$$

with T a threshold controlling the labelling estimation. A foreground label L_f is assigned to a pixel if the left side of the equation exceeds a threshold T. Using Bayes' theorem

$$P(A|B) \;=\; \frac{P(B|A)P(A)}{P(B)} \tag{9.27}$$

which states that the conditional probability $P(A|B)$ is proportional to the likelihood $P(B|A)$ times the prior probability $P(A)$, normalized by the constant $P(B)$, the maximization of the a-posteriori probability can be rewritten as

$$\hat{L} \;:\; \frac{P(\mathbb{D}^2|L_f)}{P(\mathbb{D}^2|L_b)} \;\underset{b}{\overset{f}{\gtrless}}\; T \cdot \frac{P(L_b)}{P(L_f)} \tag{9.28}$$

The conditional probability density functions have to defined in order to compute the left side of the equation. Note that these probabilities describe the pixel labelling based on the empirical data provided by $\mathbb{D}^2$. For perfectly collinear noise-free vectors hold $\mathbb{D}^2 = 0$. Because noise is sampled by a non-zero Gaussian distribution, the conditional probability density functions can be modelled by

$$P(\mathbb{D}^2|L_f) \;=\; \left(\frac{1}{\sigma_f \cdot \sqrt{2\pi}}\right)^N \cdot e^{-\frac{\mathbb{D}^2}{2\sigma_f^2}} \tag{9.29}$$

$$P(\mathbb{D}^2|L_b) \;=\; \left(\frac{1}{\sigma_b \cdot \sqrt{2\pi}}\right)^N \cdot e^{-\frac{\mathbb{D}^2}{2\sigma_b^2}} \tag{9.30}$$

within a local neighborhood of N pixels. The variances σ_f^2 and σ_b^2 describe the characteristic spread of the empirical distance $\mathbb{D}^2$ for foreground and background pixels, respectively. Background pixels have color vectors that are approximately collinear with respect to the reference image. Contrary, foreground pixels typically have a large variance, caused by the variety of non-matching colors between the observed and reference frame. Hence, the variance σ_f^2 is much larger than the variance σ_b^2 caused by noise ($\sigma_f^2 \gg \sigma_b^2$).

The inequality of formula 9.28 can be reformulated as

$$\left(\frac{\sigma_b}{\sigma_f}\right)^N \cdot e^{\frac{\mathbb{D}^2}{2\sigma_b^2} - \frac{\mathbb{D}^2}{2\sigma_f^2}} \;\underset{b}{\overset{f}{\gtrless}}\; T \cdot \frac{P(L_b)}{P(L_f)} \tag{9.31}$$

and some algebra reduces this to

$$e^{\frac{\mathbb{D}^2}{2}\cdot\frac{\sigma_f^2-\sigma_b^2}{\sigma_f^2\cdot\sigma_b^2}} \underset{b}{\overset{f}{\gtrless}} \left(\frac{\sigma_f}{\sigma_b}\right)^N \cdot T \cdot \frac{P(L_b)}{P(L_f)} \tag{9.32}$$

Taking the natural logarithm of this equation yields

$$\mathbb{D}^2 \underset{b}{\overset{f}{\gtrless}} \underbrace{\frac{2\sigma_f^2\cdot\sigma_b^2}{\sigma_f^2-\sigma_b^2}\cdot ln\left(\left(\frac{\sigma_f}{\sigma_c}\right)\cdot T\right)}_{T_S} + \underbrace{\frac{2\sigma_f^2\cdot\sigma_b^2}{\sigma_f^2-\sigma_b^2}\cdot ln\left(\frac{P(L_b)}{P(L_f)}\right)}_{T_D} \tag{9.33}$$

with a static threshold T_S and a dynamic threshold T_D.

9.6 Markov Random Field

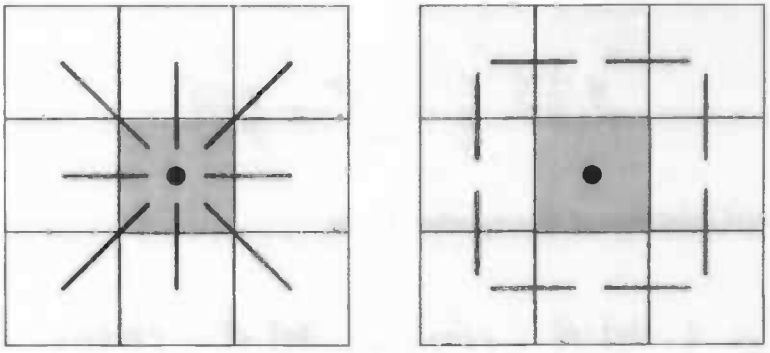
The dynamic threshold T_D in formula 9.33 depends on the prior probabilities of the foreground and background labelling and differs per pixel and per frame. As discussed in section 9.5, the smoothness and spatial compactness of foreground objects can be exploited. In this respect a pixel would have a higher chance of being foreground if many of its neighboring pixels have this classification. This reduces the number of false negatives and creates smoothly shaped objects. Likewise, false positives are discharged by the evidence of neighboring background pixels.

A Markov network or Markov random field (MRF) models this *a priori* knowledge on a principled basis. It provides a natural way to incorporate spatial correlations by modelling the contextual primitives in the images. The MRF is a generalization of the Markov process into two dimensions and describes the conditional independency of the input and output variables [Bov00]. It determines the probability of correct labelling, based on the labels in a local neighborhood, by

$$P(X) = \frac{1}{Z}\cdot\prod_k \phi_k\left(x_{\{k\}}\right). \tag{9.34}$$

The dependencies between the random variables in the neighborhood are represented by an undirected graph as weights to the edges. For each maximal clique k of this graph, $x_{\{k\}}$ is the state of the random variables in the kth clique. Because the probability of a specific value of X is requested, we limit x hereto ($X = x$) [Li01]. The potential function ϕ_k maps the possible joint probabilities of to non-negative real numbers. Z is a linear normalization constant based on the possible values of X.

The joint probability distribution required by the MRF usually is not available and its construction from the conditional distribution turns out to be very difficult for MRFs [Li01]. Fortunately,



(a) Center smoothness energy. (b) Surrounding smoothness energy.

Figure 9.4: The smoothness energy is determined by center pixel connectivity (E_C) and surrounding pixel connectivity (E_S).

the Gibbs joint probability distribution has been shown to be equivalent to the MRF [GG84] and provides a more convenient mechanism by describing the correlations in terms of an energy function:

$$P(X) = \frac{1}{Z} \cdot e^{-\beta E(X)} \tag{9.35}$$

with Z again an unspecified normalization constant. The inverse *temperature* β of the system describes the susceptibility of the system to the energy $E(X)$ in the system provided the random variables represented by X. In energy theory, temperature is the estimate of the average kinetic energy of atoms and molecules and indicates the flexibility in the system. A higher temperature (or, equivalently, a larger value of β) hence yields a more flexible system. To exploit the dynamics of the Gibbs random field, a value of $\beta = 1$ is chosen, which resembles a very high temperature[5]:

$$P(X) = \frac{1}{Z} \cdot e^{-E(X)} \tag{9.36}$$

Smooth boundaries produce adjacent pixels to be mostly similarly labelled. Hence, the *smoothness energy* can be obtained by inspecting the labelling of neighboring pixels. In a 3×3 neighborhood 16 comparisons have to be carried out: 8 concerning the center pixel and 8 among the surrounding pixels (figure 9.4). The number of matches are denoted by $\psi(L)$ and $\omega(L)$. As the labelling L is the relevant variable, the energy term can be described by

$$E(L) = E_C(L) + E_S(L) = c_C \cdot \psi(L) + c_S \cdot \omega(L) \tag{9.37}$$

[5]A system with $\beta = 1$ has high temperature. By the definition $\beta = \frac{1}{kT}$, at $\beta = 1$ the temperature T is the inverse of the Boltzmann constant k: $T = \frac{1}{k} = 7.243 \cdot 10^{22}$.

with $E_C(L)$ describing the smoothness energy at the center pixel (figure 9.4(a)) and $E_S(L)$ that of the surrounding pixels (figure 9.4(b)). Both energies are determined by counting the number of label matches, weighted by a constant. Replacing $E(X)$ in equation 9.36 by the right side of equation 9.37 (with $L = X$) for both the foreground and background objects produces

$$P(L_f) = \frac{1}{Z} \cdot e^{-(c_C \cdot \psi(L_f) + c_S \cdot \omega(L_f))} \tag{9.38}$$

$$P(L_b) = \frac{1}{Z} \cdot e^{-(c_C \cdot \psi(L_b) + c_S \cdot \omega(L_b))} \tag{9.39}$$

Substituting this into the definition of the dynamic threshold in equation 9.33 yields

$$T_D = \frac{2\sigma_f^2 \cdot \sigma_b^2}{\sigma_f^2 - \sigma_b^2} \cdot ln\left(\frac{\frac{1}{Z} \cdot e^{-(c_C \cdot \psi(L_b) + c_S \cdot \omega(L_b))}}{\frac{1}{Z} \cdot e^{-(c_C \cdot \psi(L_f) + c_S \cdot \omega(L_f))}}\right) \tag{9.40}$$

and reduces to

$$T_D = \frac{2\sigma_f^2 \cdot \sigma_b^2}{\sigma_f^2 - \sigma_b^2} \cdot (c_C(\psi(L_f) - \psi(L_b)) + c_S(\omega(L_f) - \omega(L_b))) \ . \tag{9.41}$$

9.7 Conclusion

The CVM presented in this chapter has been based on the assumption of vector collinearity of equal colors. In the presence of noise this collinear similarity is measured by estimating the 'true color' and computing the orthogonal distances $\mathbb{D}^2$ of the vectors to this signal:

$$\mathbb{D}^2 = \frac{1}{2} \cdot \left(fore + back - \sqrt{(fore - back)^2 + 4 \cdot cross^2}\right) \tag{9.42}$$

Hypothesis testing of the decision variable $\mathbb{D}^2$ is carried out using a static threshold T_S and a dynamic threshold T_D:

$$\mathbb{D}^2 \underset{b}{\overset{f}{\gtrless}} T_S + T_D \tag{9.43}$$

with

$$T_S = \frac{2\sigma_f^2 \cdot \sigma_b^2}{\sigma_f^2 - \sigma_b^2} \cdot ln\left(\left(\frac{\sigma_f}{\sigma_c}\right) \cdot T\right) \tag{9.44}$$

$$T_D = \frac{2\sigma_f^2 \cdot \sigma_b^2}{\sigma_f^2 - \sigma_b^2} \cdot (c_C(\psi(L_f) - \psi(L_b)) + c_S(\omega(L_f) - \omega(L_b))) \ . \tag{9.45}$$

Chapter 10

Implementation

"The desire to economize time and mental effort in arithmetical computations, and to eliminate human liability to error is probably as old as the science of arithmetic itself."

Howard Aiken (1900–1973)

This chapter discusses the issues regarding the implementation of the Collinear Vector Model (CVM). The outline of the source code is presented, followed by two solutions to render the performance of the CVM method more reliable in the presence of low illumination. An iterative approach to determine the energy in the Gibbs/Markov random field is also presented.

10.1 CVM Code Outline

The Collinear Vector Model has been developed using OpenCV [Ope06], the open source Computer Vision library built on the Intel Image Processing Library (IPL). OpenCV provides programmers with tools for use in real-time image segmentation, object tracking and motion estimation in a C environment. The library also enables the user easy access to input images and video streams and conveniently displays the output by means of a push method.

A frame of a video sequence is stored in `IplImage` format. This data structure contains image information such as width, height, number of channels (1 for gray-values, 3 for RGB), bits per pixel and image mask. The pixel values are stored in interleaved order. The `IplImage` data structure is capable of storing Gray, RGB, BGR and HSV formats, among others.

The CVM method has been built without the use of high-level image processing functionality. It utilizes only the IO methods for image retrieval and display, the `IplImage` data structure and its corresponding basic functions. Files part of the implementation are briefly described below:

- `FrameGrabber` – Provides a generic framework to the OpenCV IO library. It allows frames to be read from stream (e.g. camera), from video or from a sequence of separate images. If provided, the `FrameGrabber` is able to read both the input and ground truth frames.

- `CVMsegmentation` – Contains the implementation of the CVM algorithm. Provided the input frames are passed on by `segment` from the `FrameGrabber`, it produces a single mask representing the foreground objects. Although the composition of the CVM algorithm allows for a division into separate classes, thereby splitting the functionality, the individual steps of the algorithm have been combined for efficiency motives.

- `StatisticalAnalysis` – Delivers an analysis of the image statistics. When activated it produces the minimal, maximal and mean image values per pixel over the range of presented images, as well as their standard deviation, outputted to a text file. The class' functionality is not included into the CVM algorithm on default.

- `Evaluator` – Evaluates the performance of the CVM algorithm based on its produced masks and the ground truth images. The performance measures included are described in detail in section 11.1.1 and are printed to standard output.

- `segment` – The main class file, responsible for the interface between the `FrameGrabber` and the `CVMsegmentation`. It controls the parameter settings and operation of the algorithm.

The functionality of the core module, the `CVMsegmentation` class, has been divided into a number of methods. Their purpose are the following:

- `construct_background(·)` – Constructs the background reference frame by averaging over the 100 initial, *clean frames*. The auto-multiplication $W_B \cdot W_B^\top$ for each pixel is also computed since a fixed reference model is used. It is similar to the computation in `computeDistanceArguments(·)`.

- `computeDistanceArguments(·)` – Computes $W_I \cdot W_I^\top$ and $W_I \cdot W_B^\top$ for each pixel independently, prepatory to finding the *fore* and *cross* variables.

- `computeDistanceVariables(·)` – Computes the values of *fore*, *back* and *cross* using the results from `construct_background(·)` and `computeDistanceArguments(·)`. This is done by spatial summation in a small neighborhood around each pixel.

- `segment(·)` – Determines the decision variable $\mathbb{D}^2$ and the static and dynamic thresholds T_S and T_D. This is done by finding the pixel energies in the Gibbs/Markov random field in 8 iterations, following the computation of *fore*, *back* and *cross*. A threshold determines the label mask.

10.2 Brightness Intensification by Increased Vector Length

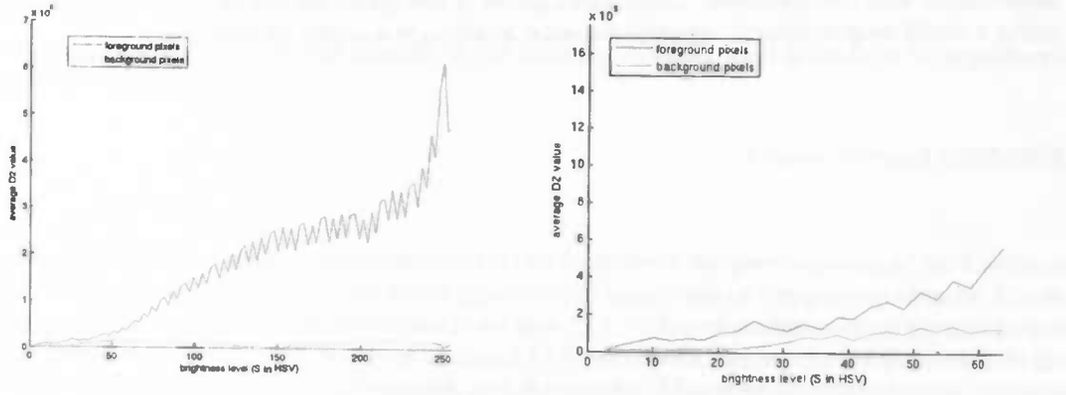
To render the Collinear Vector Model more effective at low intensity illumination, Griesser et al. [GDRNVG05] added an additional fixed component $\sqrt{O_{dc}}$ to the signal vectors W_I and W_B. The effect is that the additional component lifts the $3N$-dimensional plane[1] containing the colors in the local window to height $\sqrt{O_{dc}}$ in the $3N+1$ feature space. This produces an addition of O_{dc} to the values $fore$, $back$ and $cross$ (see equation 9.9).

[1] $3N$ dimensions, and not N, because we defined N in section 9.1 as the number of pixels in the local window.

10.3 Choosing the Parameter Values

Section 9.5 discussed the behavior of σ_f^2 and σ_b^2 in the context of the Bayesian estimation. It was argumented that σ_f^2 would be much larger than σ_b^2 due to the large difference between the observed frame and the reference frame (see figure 10.1(a)). Hence, Mester et al. [MAD01] argued that σ_b^2 could be neglected as $\sigma_f^2 - \sigma_b^2 = \sigma_f^2$, leaving a simplified $\frac{\sigma_f}{\sigma_b}$ as the coefficient of both the static threshold T_S and dynamic threshold T_D (see equation 9.33).

At low illumination this simplification does not hold. Both background and foreground pixels appear as almost black and the large difference fades, although the value of $\mathbb{D}^2$ remains slightly larger on average (figure 10.1(b)). However, since we are concerned with standard deviations, the $\mathbb{D}^2$ values of the background may exceed those of the foreground at low intensity. This becomes evident when inspecting the maximum value of $\mathbb{D}^2$ of the background pixels at each saturation level (purple dashed line in figure 10.1(b)).



(a) $\mathbb{D}^2$ at full saturation range.

(b) $\mathbb{D}^2$ at saturation range 0–60.

Figure 10.1: Maximum of the standard deviation of the decision measure $\mathbb{D}^2$ for each saturation level across 1190 sample images from the RIC *dataset (see appendix B). The values have been separated between foreground and background pixels.*

A fixed value of σ_f would either discard all low intensity foreground pixels (large σ_f) or introduce noisy foreground regions at medium and high intensity (small σ_f). The brightness intensification discussed in section 10.2 somewhat alleviates this problem, but does not justify the simplification. Instead a solution may be to adjust the variances to the level of intensity. To determine the values of σ_f and σ_b the intensity distribution is off-line monitored on a large number of frames. For each saturation level the value of σ_f^2 is based on the empirically determined standard deviations. These standard deviations, for both the true foreground and background pixels, are plotted in figure 10.1(a).

10.4 Iterative Energy Computation in MRF

The Gibbs/Markov random field, introduced in section 9.6 to model the smoothness and spatial compactness of foreground objects, uses the energy terms $E_C(L)$ and $E_S(L)$. The energy

flows from the labelling of the pixel neighborhood. This introduces a chicken-and-egg problem because, to determine the labelling, the pixel labels of the neighborhood have to be known in advance.

The energy terms are computed for each pixel in top-down, left-to-right order. This causes the pixels at the right and bottom side of the center pixel to have unknown labels. An iterative scheme is implemented to overcome this deadlock—an approach more often followed in literature. At each round the unknown pixel labels are approximated by using the previous labelling. For the known labels (those of the upper and left neighbors) the results of the current iteration are used. The special case of the first iteration, at which previous labels for that frame are not available, utilizes the label outcome of the previous frame. The first iteration of the first frame initiates with all pixels labelled as background.

The use of previous frames to obtain the surrounding labels is motivated by the observation that normal motion produces only small local changes in the labelling. The existing discrepancy is furthermore reduced by the iterative computation of the labels. This also improves the smoothness and compactness. Disadvantageous is the effect on the framerate as, even when using a small neighborhood, repetitive spatial analysis is a costly procedure.

Chapter 11

Experiments

"The true worth of an experimenter consists in his pursuing not only what he seeks in his experiment, but also what he did not seek."

Claude Bernard (1813-1878)

This chapter discusses the experiments carried out to observe the performance of the Collinear Vector Model presented in chapter 9 as a segmentation algorithm of foreground objects. Firstly, the design of the experiments is presented in section 11.1. It describes the motives for the dataset and experiment methodology. Sections 11.2 and 11.3 provides a description of the events in the dataset, how the performance is measured, and the results. This is all concluded by an analysis and discussion on the behavior of noise in section 11.4.

The purpose of the experiments is to determine to what amount a *non-learning* segmentation method can effectively detect foreground objects in the presence of brightness variation. One might argue that, as no model is constructed that describes the shape and appearance of objects, this limits the quality of the segmentation. Furthermore, the absence of a learning scheme disallows the system to adapt to ill-defined and unknown dynamics. As a result, unexpected and complex changes of objects and scene might not be properly modelled. This may cause the system to function poorly, e.g. in the case of flashing lights or object tracking in the presence of occlusions. However, a carefully designed system, albeit without an on-line learning scheme, should not be hampered by peculiar events. The following experiments are carried out to determine if the Collinear Vector Model adheres to these requirements.

11.1 Design

Performance evaluation is required to determine the effectiveness of any developed algorithm. It is important to gather the right data to test the design of the algorithm against specific situations and events. Until 2000, the research community measured their results by use of private data. Comparisons between published algorithms were hard to produce due to the large variety of data sets and metrics. This caused performance to be biased towards exaggeration of

the results, sometimes aided by the low relevancy of the datasets to the particular problem. A solution to this nonconformity was presented at the first Performance Evaluation of Tracking and Surveillance (PETS) workshop [Fer05]. The participants to this workshop presented and tested their algorithm on a standard dataset, thereby providing the first fully normalized algorithm comparison. The PETS evaluation has since established itself as one of the leading services evaluating tracking and segmentation performance.

To determine the general performance of the Collinear Vector Model in relation to modern, proven algorithms, the PETS dataset has been adopted. The PETS metrics have been designed for the evaluation of object tracking algorithms. However, PETS also serves as a measure on the quality of object extraction. It determines this latter quality by evaluating the object localization[1] and segmentation.

The PETS service does not evaluate the particularities of more specialized algorithms among which the Collinear Vector Model can be shared. Performance measurement at various illumination levels is a minor element of PETS evaluation and rapid changes in brightness are not an explicit part of the test videos. To estimate the performance of the Collinear Vector Model under varying illumination conditions datasets have been gathered. This Rapid Illumination Change (RIC) dataset exhibits sudden light changes which are explained in more detail below.

11.1.1 Performance Measure

How should we determine if a segmentation is sufficient? One could visually observe if the foreground masks resemble the shape of the objects present in the videos. This, however, will result in a measure in terms of "good", "fair" and "bad", yielding only a limited insight into the quality of the segmentation. A more detailed assessment thus boils down to an exact approach, in which the quality is expressed in terms of quantities[2].

Performance is a dual variable. Not only is it required to obtain a good segmentation, it also has to be achieved in reasonable time. Object tracking applications often require real-time segmentation algorithms. The quantification of performance can therefore be divided into two categories.

Qualitative performance — The quality of a segmentation can be effectively determined by the use of *metrics*. In mathematics, metrics are functions describing the distance between elements of a set. In the evaluation of a segmentation, metrics present a topology on the basis of segmentation results and the ground truth (to be discussed in the next section), from which reliable performance measurements can be deduced. Common metrics inspect the pixel-wise mismatches and misclassified pixels, often using observations on the number of *false positives* and *false negatives* in a segmentation. These measures correspond to the *type I* (falsely rejecting the null hypothesis) and *type II* errors (falsely accepting the null hypothesis) in statistics. For the PETS 2001 datasets, Young & Ferryman [YF05] proposed a metric standard that evaluates the quality of object segmentation in videos, which will be addressed in detail in section 11.2.3.

[1] For the purpose of tracking a perfect segmentation is not always required. More important is it to effectively detect and track the position or location of objects throughout a video. Object localization serves this goal, often by representing objects by a bounding box.

[2] In the context of human vision, one may determine the amount of similarity by inspecting the influence of individual neurons on the segmentation. Due to the center-surround organization, retinal neurons influence the result of many neural responses in its neighborhood. A penelization system based on weighted misclassifications of the surrounding double-opponent cells (see section 8.4.1 for an explanation) could provide a robust performance estimate on a sub-pixel level.

Quantitative performance — The computational effectiveness represents the speed of an algorithm. It can be determined by measuring the *framerate* of the algorithm: the number of video frames processed per second. This measure can be employed regardless of the precise purpose of the algorithm. We therefore present its definition in this section, rather than addressing it for each of the datasets. If compared to competitive methods, in conjunction with the qualitative performance measurements, the framerate provides an indication of the complexity of the algorithm.

Before the metric definitions can be presented it is necessary to introduce four general evaluation variables. These variables are described in table 11.1.

Variable	Description
N_{tp}	*Number of true positives:* the amount of correctly classified foreground pixels.
N_{tn}	*Number of true negatives:* the amount of correctly classified background pixels.
N_{fp}	*Number of false positives:* the amount of background pixels incorrectly classified as foreground.
N_{fn}	*Number of false negatives:* the amount of foreground pixels incorrectly classified as background.

Table 11.1: *Description of the basic evaluation variables.*

11.1.2 Ground Truth

The qualitative performance can only be determined if some kind of reference is available, which provides the actual segmentation of the objects. Such a reference is called a *ground truth*. It consists of a mask partitioned into a pixel labelling. Each label value corresponds to an object. Occlusions may split these objects into parts, but their labelling will remain the same[3]. These labellings may be split into layers, one for each object, and annotated by additional information, e.g. their bounding box. Construction of these ground truth datasets is a tedious and error-prone process, during which complex scenes have to be analyzed and segmented manually frame by frame. An alternative is to superimpose an automated script or partially transparent sequence onto a background video. This directly provides a ground truth labelling, but comes at the expense of less realistic data.

11.1.3 Test System

The CVM tests are run on a 2.8 GHz Intel Pentium IV system with 512 MB RAM. The installed operating system is a 2.6.17 Linux distribution.

11.2 The PETS 2001 Dataset

The PETS 2001 Metrics Evaluation Service[4] provides a facility to evaluate and compare object segmentation and tracking algorithms. It includes an annotated dataset of complex outdoor

[3]Image regions affected by these occlusions are labelled as part of the occluding object—they are no longer part of the occluded object.

[4]`http://www.cvg.cs.rdg.ac.uk/cgi-bin/PETSMETRICS/page.cgi?home`

scene, a metric standard comprising four performance measures and an online ranking. In the following sections the events occurring in the dataset, the metrics and the results of the Collinear Vector Model are discussed.

11.2.1 Dataset Description

The PETS *2001 Dataset 1 Camera 1* sequence features an elevated daylight campus view. Stills from the 2688 frames long video sequence are listed in appendix A. The images have been scaled down to 384 × 288 pixels in dimension. The events occurring in this dataset are described in table 11.2.

Frame	XML file	Event
110–653	*ObjectXML0.xml*	Presence of a red-coated female person. The object enters the scene from the left and exists at the right.
452–2688	*ObjectXML1.xml*	Presence of a blue car. The car enters the scene from the right and is parked at the center. The PETS2001 ground truth tracking stops representing the car at frame 707, since it has become static.
678–2688	*ObjectXML2.xml* *ObjectXML10.xml* *ObjectXML14.xml*	Presence of a white van. The van enters from the left and is parked at the right border of the scene at frame 997. At frame 1589 it continues its movement until frame 1959. From frame 2493 onwards, the van moves towards the left border of the scene.
771–1912	*ObjectXML3.xml* *ObjectXML4.xml* *ObjectXML5.xml*	Presence of three persons in brown and white shirts. The persons enter from the bottom-left and exits the scene due to the exclusion of a tree in the top of the scene.
830–1091	*ObjectXML6.xml*	Presence of a person in a brown shirt. The person enters the scene from the bottom-left and exists at the bottom-right, in front of the static white van.
924–1265	*ObjectXML7.xml*	Presence of a blue-shirted male. The person appears next to the parked blue car (see *ObjectXML1.xml*), remains static for a while and then proceeds to exit the scene at the bottom-right.
1338–1859	*ObjectXML8.xml* *ObjectXML9.xml*	Presence of a male in a white shirt and a female in a cream t-shirt. Both enter the scene at the bottom-right alongside each other and exit at the left.
1762–2267	*ObjectXML11.xml*	Presence of a white-topped person directly in front of the buildings, traversing from right to left.
2012–2571	*ObjectXML12.xml*	Presence of a female in blue skirt. The person enter the scene at the bottom-right and exits on the left.
2161–2665	*ObjectXML13.xml*	Presence of a black car which enters the scene at the right, turns around and exits on the left.

Table 11.2: *Description of the events in the* PETS *2001 Dataset Camera 1.*

The ground truth of these events has been published in an XML format. These files, of which a snippet is listed in appendix A.1, describe the foreground objects by a moving bounding box.

A per-pixel ground truth is therefore not available, which limits the tests to processing only the relevant frames (i.e. the frames containing foreground objects). Due to this representation, the events described in table 11.2 are ordered per object. Frames in which an object has become static (i.e. not moving) have been excluded from the XML description. These frames are therefore excluded from the evaluation.

11.2.2 Preparing the Data

The PETS 2001 metrics evaluate the segmentation based on the bounding boxes of the foreground objects. The CVM method is not specifically designed for the task of tracking and does not provide this representation. The output may contain many tiny foreground objects, induced by noise and color similarity. Isolated false positive pixels produce many tiny objects while false negatives could split coherent objects into multiple regions, producing multiple bounding boxes of a single ground truth object.

(a) CVM output (b) area opening (c) dilation (d) connected components (e) processed

Figure 11.1: Example of the post-processing steps on the PETS *dataset in preparation of the performance evaluation. The figures show the segmentation output of the* CVM *method and the results after each step: area opening, dilation, connected components labeling and the combined processed output.*

To obtain a proper set of bounding boxes of the CVM segmentation a few post-processing methods are applied. The focus of these methods is twofold: isolated pixels should be removed, while separated object regions need to be merged. To remove spurious sets of foreground regions *area opening* is applied to the binary segmentation mask:

$$\mathtt{mask}_{open} = \mathtt{mask} \circ^{N_{AE}} SE \tag{11.1}$$

where $\circ$ is the area opening operator and `mask` is the segmentation produced by the CVM algorithm. This morphological operator removes all foreground objects that consist of less than $N_{AE} = 15$ pixels. The structuring element SE has the shape

$$SE = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}. \tag{11.2}$$

The resulting image is repetitively *dilated* ($N_D = 3$ times) using a square structuring element of which the *connected components* are determined using 8-connectivity:

$$\mathtt{mask}_{dilated} = \mathtt{mask}_{area\_open} \oplus^{N_D} SE \quad (11.3)$$

$$\mathtt{mask}_{components} = connected\_components(\mathtt{mask}_{dilated}, connectivity = 8) \quad (11.4)$$

where $\oplus$ denotes a dilation. The dilation process ensures that neighboring foreground regions receive the same labelling and be grouped by a single bounding box. To remove the added foreground pixels this labelling is combined by the original segmentation (see equation 11.5). The above process is illustrated in figure 11.1.

$$\mathtt{mask}_{processed} = \begin{cases} \mathtt{mask}_{components}(x, y) & \mathtt{if\ mask} > 0 \\ 0 & \mathtt{if\ mask} = 0 \end{cases} \quad (11.5)$$

Variable	Description
N_{tp}	The number of inside the ground truth bounding boxes that have been classified as foreground.
N_{tn}	The number of outside the ground truth bounding boxes that have been classified as background.
N_{fp}	The number of outside the ground truth bounding boxes that have been classified as foreground.
N_{fn}	The number of inside the ground truth bounding boxes that have been classified as background.

Table 11.3: The PETS *definition of the true/false positive and negative variables.*

The ground truth definition of the PETS dataset does not lend itself for an exact evaluation of the false positive and false negative pixels as defined in section 11.1.1. Instead, the PETS metrics uses the definition presented in table 11.3 and depicted in figure 11.2.

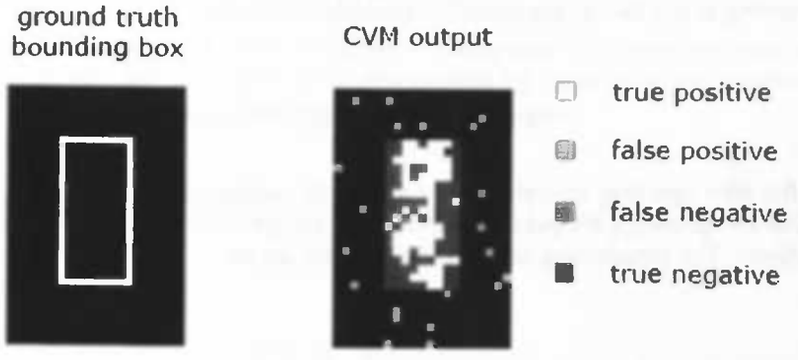


Figure 11.2: Definition of the true/false positives and negatives by the PETS *2001 dataset metrics. Incorrectly classified foreground pixels located outside of the bounding boxes are false positives. Incorrectly classified background pixels (inside the bounding boxes) are false negatives.*

11.2.3 Metrics

In [YF05], Young & Ferryman proposed four metrics to be standardized in the PETS 2001 dataset on motion segmentation. These metrics form the basis of the evaluation discussed later on. The variables N_{fp} and N_{fn}, defined in section 11.1.1, describe the amount of misclassification and are the foundation of the four metrics. Young & Ferryman suggest that

> [...] a low false positive score describes good object boundary identification. A low false negative score describes a good identification of foreground internal to the object.

This definition disregards the possibility of foreground and background holes. The false positive and negative scores could be better expressed as indicators of the *overclassification* ($N_{fp} > N_{fn}$) and *underclassification* ($N_{fp} < N_{fn}$) of a labelling problem. In fact, the semantic explanation of N_{fp} and N_{fn} is a specific case of this generalized definition and is only applicable if objects are represented by bounding boxes. Nevertheless, regardless this definition, the proposed metrics provide an effective tool to evaluate the classification. Their definition is given below. For each metric a lower value indicates better performance.

- **negative rate metric (NR)** — The pixel-wise mismatches between the ground truth and the segmentation result are computed by evaluating the false negative rate NR_{fn} and false positive rate NR_{fp}:

$$NR = \frac{1}{2}\left(NR_{fn} + NR_{fp}\right) \tag{11.6}$$

where

$$NR_{fn} = \frac{N_{fn}}{N_{tp} + N_{fn}} \tag{11.7}$$

$$NR_{fp} = \frac{N_{fp}}{N_{fp} + N_{tn}} \tag{11.8}$$

The false negative rate NR_{fn} describes the number of pixels incorrectly classified as background compared to the actual number of foreground pixels. The false positive rate NR_{fp} is defined likewise on the number of pixels incorrectly classified as foreground.

- **misclassification penalty metric (MP)** — The object segmentation is compared to the ground truth for each object. Because the Collinear Vector Model does not distinguish separate objects, this simplifies to a single object comparison. Misclassified pixels receive a penalty weighted by their distance to the object border. A low MP score indicates that the algorithm is proficient in identifying the object contour and segmenting it from the background scene.

$$MP = \frac{1}{2}\left(MP_{fn} + MP_{fp}\right) \tag{11.9}$$

where

$$MP_{fn} = \frac{\sum_{j=1}^{N_{fn}} d_{fn}^{j}}{D} \tag{11.10}$$

$$MP_{fp} = \frac{\sum_{k=1}^{N_{fp}} d_{fp}^{k}}{D} \tag{11.11}$$

where d_{fn}^{j} and d_{fp}^{k} are the distances of the j^{th} false negative and k^{th} false positive pixel to the ground truth object border. The distance to the border of the nearest ground truth object is taken. The divisor D and functions as a normalization factor, taking the total sum of the distances of pixels belonging to the objects in the computed segmentation.

- **rate of misclassifications metric (RM)** — Similar to MP, the RM metric determines the distance of incorrectly classified pixels to the reference object border. However, instead of producing a measure on the quantity of this type of error, it yields an average degree of error.

$$RM = \frac{1}{2}\left(RM_{fn} + RM_{fp}\right) \tag{11.12}$$

where

$$RM_{fn} = \frac{1}{N_{fn}} \sum_{j=1}^{N_{fn}} \frac{d_{fn}^{j}}{D_{diag}} \tag{11.13}$$

$$RM_{fp} = \frac{1}{N_{fp}} \sum_{k=1}^{N_{fp}} \frac{d_{fp}^{k}}{D_{diag}} \tag{11.14}$$

where D_{diag} is the diagonal distance of the frame.

- **weighted quality measure metric (WQM)** — The spatial difference between the estimated segmentation and the ground truth is determined by measuring the weighted distances d_{fn}^{j} and d_{fp}^{k} of the false negative and false positive pixels.

$$WQM = ln\left(\frac{1}{2}\left(WQM_{fn} + WQM_{fp}\right)\right) \tag{11.15}$$

where

$$WQM_{fn} = \frac{1}{N_{fn}} \sum_{j=1}^{N_{fn}} w_{fn}\left(d_{fn}^{j}\right) d_{fn}^{j} \tag{11.16}$$

$$WQM_{fp} = \frac{1}{N_{fp}} \sum_{k=1}^{N_{fp}} w_{fp}\left(d_{fp}^{k}\right) d_{fp}^{k} \tag{11.17}$$

where N is here defined as the number of pixels in the area of ground truth objects. The weighting functions w_{fp} and w_{fn} are treated differently:

$$w_{fp}(d_{fp}) = B_1 + \frac{B_2}{d_{fp} + B_3} \tag{11.18}$$

$$w_{fn}(d_{fn}) = C \cdot d_{fn} \tag{11.19}$$

with $B_1 = 19$, $B_2 = -178.125$, $B_3 = 9.375$ and $C = 2$ as in [YF05] and [AWK+05]. Function w_{fp} has been plotted in figure 11.3. The separate treatment of the weighting function was motivated by Aguilera et al. as false negatives (pixels missing in the foreground) contribute more to the visual degradation of 'good' segmentation at increasing distances than false positives (erroneously added foreground pixels). The particular selection of these values favors algorithms that provide larger foreground objects over conservative ones. Thus, it is biased towards overclassification.

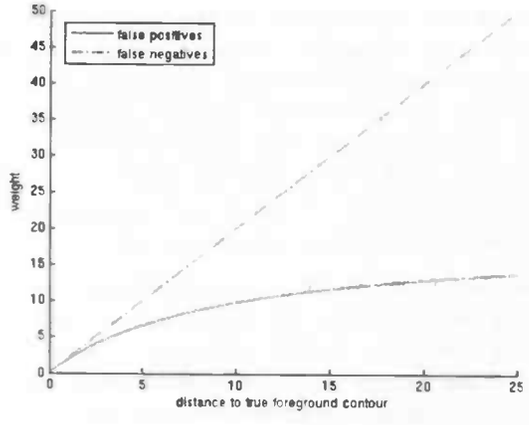


Figure 11.3: Behavior of the WQM weighting functions w_{fp} (solid) and w_{fn} (dashed) on the range of commonly observed distances.

11.2.4 Results

The performance measurements of the CVM algorithm on the PETS 2001 dataset present the opportunity of a comparison with the competitive field of object segmentation and tracking. Following this evaluation we will focus on the specific performance of (parts of) the specific dataset. The frames were processed at a framerate of 0.17134 fps.

Table 11.4 shows a comparison between the CVM algorithm and specialized object tracking algorithms[5] published by Young & Ferryman [YF05]. Although this is a somewhat unfair comparison (since the CVM algorithm has not been designed to discard non-relevant objects and

[5] A description of these methods goes beyond the scope of this report; the interested reader is recommended to refer to the referenced literature presented in table 11.4.

Algorithm	Metric			
	NR	MP	RM	WQM
Brightness and Chromaticity (BC) [HHD99]	0.1773	0.518	0.352	5.571
Five Frame Difference (DIF)	0.1764	5.183	0.261	5.837
Edge Fusion (EDG) [JDWR00]	0.1656	4.123	0.186	5.723
Gaussian Mixture Model (GMM) [SG99]	0.1520	4.060	0.176	5.746
Kernel Density Estimation (KDE) [EHD00]	0.1523	3.144	0.170	5.666
Color Mean and Variance (VAR) [WADP97]	0.1663	4.029	0.182	5.721
Wallflower Linear Prediction Model (WFL) [TKBM99]	0.1229	3.882	0.181	5.769
Collinear Vector Model (CVM)	0.0431	2.328	0.002	174.379

Table 11.4: Performance of the CVM *and competitive object segmentation and tracking methods evaluated by the standard* PETS *2001 metrics.*

only an ad hoc solution has been presented in section 11.2.2) it provides insight into the relative quality of performance.

The CVM method produces a low negative rate (NR) which indicates that the algorithm is better suited against noise and color variations compared to its competitors. Surprisingly, most of the erroneously segmented pixels have already been suppressed by the rudimentary filter. Failure of the other algorithms in reaching this level of performance may be due to tracking mechanisms that have the tendency to retain consistent sets of pixels which have been initially segmented as foreground by a change in brightness. The low quantity of false classifications also becomes apparent by the MP metric. It determines the degree of error, determined by the relative distance to the nearest ground truth bounding box for each misclassified pixel.

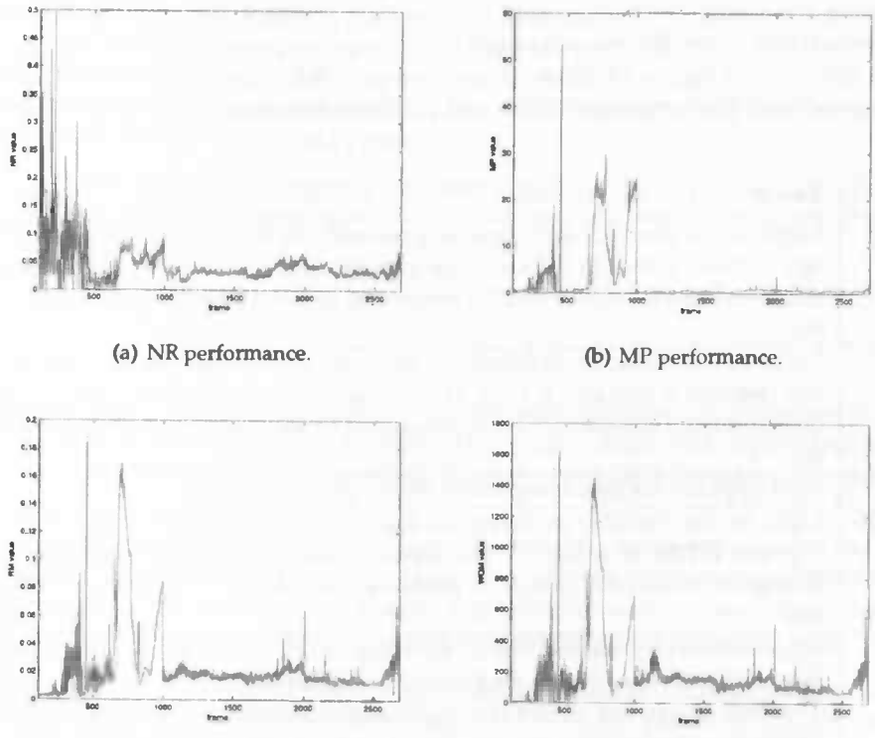
Unfortunately, the misclassified pixels produce very high WQM values. Comparing to the low MP and RM values, it can be concluded that the misclassified pixels are mostly located far away from the object borders (e.g. see figure A.3(b)). Assumed is that the good performance of the competitive algorithms on these metrics is caused by their effectivity of maintaining coherently shaped objects while excluding other, non-relevant pixel regions. This is most likely aided by the temporal information utilized by these methods. The CVM lacks this information and 'foolishly' accepts noisy, likely non-moving, groups of pixels.

	NR	MP	RM	WQM
average performance (mean)	0.0431	2.3277	0.0226	174.3792
false positives	0.0105	4.5933	0.0396	300.8460
false negatives	0.0757	0.0621	0.0056	47.9124

Table 11.5: Metric evaluation of the CVM *on the* PETS *2001 Dataset 1 Camera 1 dataset, with performance split up between false positive and false negative results.*

A performance split between false positives and false negatives supports this claim (table 11.5). The false negatives exhibit a performance similar to the averaged metric values of the competitive algorithm. These values are small foreground objects tend to remains small, yielding low distances of the misclassified false negatives (i.e. pixels classified as background inside the bounding boxes) to the ground truth contours. A high false positive performance furthermore indicates large distances of pixels erroneously classified as foreground.

The CVM performance has been plotted in figure 11.4 for each frame in the PETS 2001 dataset containing foreground elements. The input, ground truth bounding boxes, CVM output and enhanced masks are displayed in figure A.2 and A.3 in appendix A. The MP performance shows one minor (frame 450) and one major (frames 667–998) error. The decline in performance at frame 450 is caused by the absence of a ground truth bounding box for the entering car at the bottom right of the scene (figure A.2(b)). The same holds for the other the large error at the range 667–998 (see figures A.2(c) and A.2(d)). These omissions in the ground truth also cause a decrease in the measured performance by the metrics. The 'misclassified' pixels are indeed located at large distances from the ground truth bounding boxes, which is indicated by the high WQM error. The discussed example is representative for the other significant segmentation failures.



(a) NR performance.

(b) MP performance.

(c) RM performance.

(d) WQM performance.

Figure 11.4: Per frame plots of the measurements of the NR, MP, RM and WQM performance metrics.

Despite these inferior results the CVM results show an overall satisfactory performance (see the images in figure A.2), with the metric measures approaching or exceeding the performance of the specialized algorithms. In fact, only the WQM metric displays a coherently bad performance, which indicates that the CVM algorithm has the tendency to provide small foreground objects compared to the bounding boxes. This behavior has been expected, since the algorithm computes an image difference rather an object shape approximation (e.g. a bounding box).

11.3 The RIC Dataset

The PETS dataset tests the CVM algorithm on its overall performance in a complex, realistic environment. However, the specific design of the Collinear Vector Model to the effects of variation in illuminance has not been addressed by these tests. To determine the resilience of the CVM , a dataset is included into the experiments that features rapid illumination change (denoted as RIC). This section presents the events occurring in this dataset, the metrics used to evaluate the performance of the CVM algorithm and the results.

11.3.1 Dataset Description

The RIC dataset simulates the appearance of complex objects in a realistic indoor environment. During the scene, the background illumination changes both gradually and instantly. Computer-modelled three-dimensional objects are superimposed onto the background, both inside and outside the region of illumination change. Still taken from this RIC sequence are listed in appendix B. The images are 384 × 240 pixels in dimension. The events are outlined in table 11.6.

Frame	Event
134–176	Light in the corridor switches on gradually in the right upper part of the image (figures B.1(a) to B.1(d)). The average saturation of the concerned region changes from 0.2023 to 0.3132, measured in saturation based on a HSV analysis.
596–597	Light in the corridor switches off abruptly in the right upper part of the image (figures B.1(e) and B.1(f)). The average saturation of the concerned region changes from 0.3184 to 0.1988, measured in saturation based on a HSV analysis.
658	First appearance of a foreground object (figure B.1(g)).
969–1006	Light in the corridor switches on again in the right upper part of the image (figures B.1(k) to B.1(m)). The average saturation of the concerned region changes from 0.1901 to 0.3217, measured in saturation based on a HSV analysis.
1007	Appearance of a second object, entering at the right side of the image, superimposed onto the illuminated corridor (figure B.1(n)).
1145	The first object has exited the scene at the left side of the image, the second object remains present (figures B.1(o) and B.1(p)).

Table 11.6: Description of the events in the RIC *dataset.*

11.3.2 Metrics

To determine the proficiency of the CVM algorithm to deal with illumination change, the evaluation is focussed on the incorrectly classified pixels. Increase and decrease of light intensity may cause background pixels to be erroneously segmented as foreground. To quantify this effect, measures originating from the domain of information retrieval are adopted.

- **precision** — The proportion of pixels correctly detected as foreground out of the total number pixels segmented as foreground is computed by

$$\texttt{precision} = \frac{N_{tp}}{N_{tp} + N_{fp}} \tag{11.20}$$

where the sum of the true positives N_{tp} and false positives N_{fp} together describes the total number of pixels classified as foreground.

- **recall** — The proportion of pixels correctly detected as foreground out of all the foreground pixels is computed by

$$\texttt{recall} = \frac{N_{tp}}{N_{tp} + N_{fn}} \tag{11.21}$$

where the sum of the true positives N_{tp} and false negatives N_{fn} together describes the total number of true foreground pixels.

- **elusion** — The proportion of pixels incorrectly classified as foreground out of all the background pixels is computed by

$$\texttt{elusion} = \frac{N_{fp}}{N_{tn} + N_{fp}} \tag{11.22}$$

where the sum of the true negatives N_{tn} and false positives N_{fp} together describes the total number of true background pixels. The elusion measure is equivalent to the NR_{fp} variable from the negative rate metric (NR) of the PETS 2001 dataset (c.f. formula 11.8).

To produce a single notion on the classification error of true foreground pixels the average on the precision and recall should be taken. Since this relationship involves different rates (the units of the precision and recall are not similar, but have different divisors) this average should be computed by using the *harmonic mean* rather than the standard *arithmetic mean*.

Section 11.2.3 discussed the preference of overclassification over underclassified foreground objects, which called for a favorable weighting towards false positives. In the context of illumination variation, false positives indicate inadequate modelling of light intensity changes. In these measurements, false positives should therefore receive a larger punishment than false negatives. To accommodate this choice a *weighted harmonic mean* is used:

$$F_\alpha = (1 + \alpha)\frac{\texttt{precision} \cdot \texttt{recall}}{\alpha \cdot \texttt{precision} + \texttt{recall}} \tag{11.23}$$

For the test conducted on the RIC dataset the precision is weighted twice as much as the recall: the weighted harmonic mean has been set to F_2.

The measures discussed above determine the performance for a single frame. Taking the average of the measures for a sequence of frames produces a notion of the general performance.

Frames containing a large body of foreground pixels would produce a reliable estimate of the precision (equation 11.20) and recall (equation 11.21). However, these measures are much more vulnerable at frames featuring only a few foreground pixels (e.g. figure B.1(g)): misclassifications will have a larger impact on the precision and recall. Therefore, the mean of the precision and recall measurements may be biased towards frames with a low density of foreground pixels. A weighted average based on the number of true foreground pixels is proposed to account for this skewness. The weight is based on the number of true foreground pixels N_{tp} normalized by its maximum in the sequence:

$$WAP = \frac{1}{Z}\sum_{s\in S}(N_{tp}(s)\cdot \texttt{precision}(s)) \tag{11.24}$$

$$WAR = \frac{1}{Z}\sum_{s\in S}(N_{tp}(s)\cdot \texttt{recall}(s)) \tag{11.25}$$

where WAP and WAR are the *weighted average precision* and *weighted average recall* over the frames in S and Z is the total number of foreground pixels in S. The *weighted average elusion* (WAE) is computed similarly, using the false positive rate as weight:

$$WAE = \frac{1}{Z}\sum_{s\in S}(N_{fp}(s)\cdot \texttt{elusion}(s))\,. \tag{11.26}$$

The *average weighted harmonic mean* is deduced from WAP and WAR by

$$AF_\alpha = (1+\alpha)\frac{\texttt{WAP}\cdot\texttt{WAR}}{\alpha\cdot\texttt{WAP}+\texttt{WAR}} \tag{11.27}$$

11.3.3 Results

The experiments on the RIC dataset have been divided into an analysis of the presence and absence of foreground objects. While the former provides an insight into the segmentation performance, the latter shows the influence of significant illumination change. Appendix B contains the stills, ground truth images and segmentation results. The frames were processed at a framerate of 0.40124 fps.

Frames 672 to 968 contain an unimpeded presence of a foreground object in the absence of illumination change of which some stills are depicted in figure B.1(h) to B.1(k). Figure B.3 shows the segmentation results and corresponding ground truth on these frames. A first visual comparison shows that the CVM is able to segment the foreground objects except some minor regions at the legs and head of the person. The precision and recall, plotted in figure 11.6, show a somewhat unsteady behavior. This is caused by the complex movement of the foreground object. Both neck and upper and lower legs have colors similar to the background (wall and floor, respectively), producing incorrect classifications.

Two minima in the precision and recall are located at frames 772 and 933. The reduced performance at frame 772 is an effect of a temporal reduction of the number of true foreground pixels,

while N_{fn} and N_{fp} hold approximately the same quantity. This reduces both the precision and recall significantly. The segmentation at frame 933 is accompanied by an considerable increase of N_{fn} and N_{fp}, visible by the crude segmentation result (center column of figure 11.5). As the number of true foreground pixels remains on a similar level if compared to the predecessing and successing frames, the precision and recall show a strong decrease.

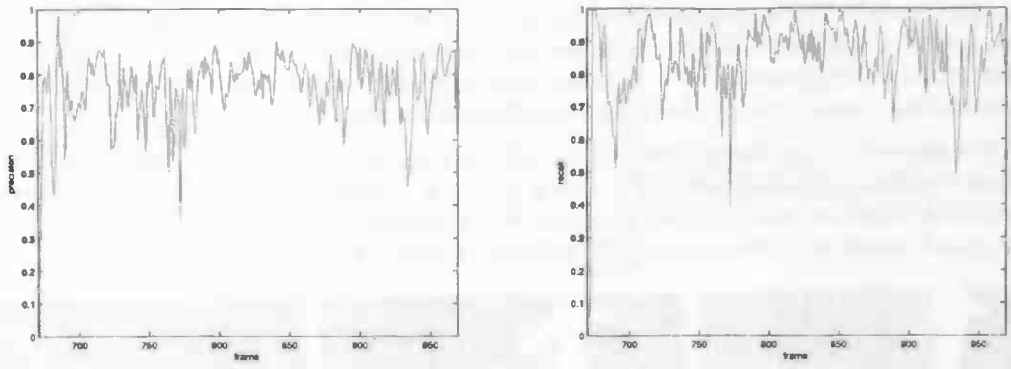
The CVM algorithm has the tendency to overclassify the data, producing larger than real foregrounds (right column of figure 11.5). This is brought about by the spatial smoothing property of the Gibbs/Markov random field energies. Nevertheless, the weighted average performance shows good results on the sequence 672–968 (see table 11.7).

Figure 11.5: Example of the preference of overclassification of the CVM *algorithm of frame 772 (top) and 933 (bottom). For both frames the ground truth (left), segmentation result (center) and false positives (right) are displayed.*

Metric	Measured performance (per sequence)		
	101–658	672–968	969–1190
weighted average precision (WAP)	–	0.7694	0.5695
weighted average recall (WAR)	–	0.8571	0.8476
weighted average elusion (WAE)	0.0690	0.0295	0.1065
average weighted harmonic mean (AF_2)	–	0.8257	0.7289

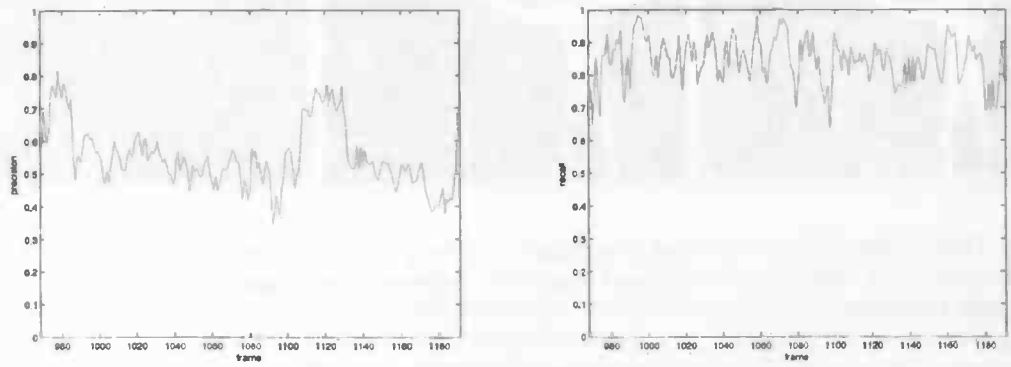
Table 11.7: Performance of the CVM *on subsets of the* RIC *dataset. The weighted average precision, recall and harmonic mean are excluded from the measure on frames 101–658, since these frames do not include foreground pixels.*

Frames 969 to 1190 follow the previous sequence (figure B.1(l) to B.1(p)). In contrast with sequence 671–968, sequence 969–1190 shows a significant performance reduction (figure 11.7). The weighted average precision has dropped by 26% to 0.5695, although the recall—the correctly classified foreground pixels out of all foreground pixels—has remained unchanged. As described in table 11.6 light intensity in the corridor increases. This induces the CVM to incorrectly classify large regions in the upper-right part of the scene as foreground, as can be observed from figure B.4.



(a) Precision (b) Recall

Figure 11.6: Precision and recall of the segmentation result on frames 672 to 968 from the RIC *dataset.*



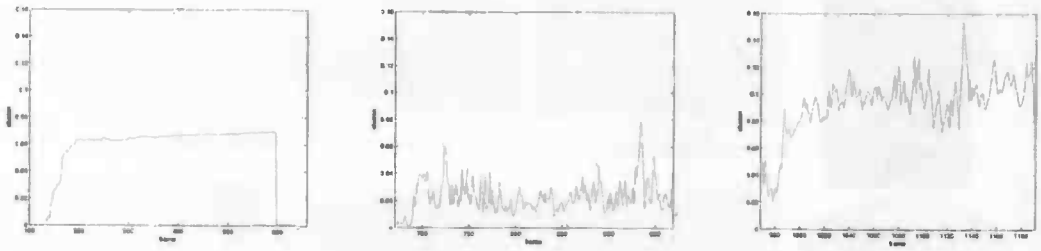
(a) Precision (b) Recall

Figure 11.7: Precision and recall of the segmentation result on frames 969 to 1190 from the RIC *dataset.*

A similar effect is observed from sequence 101–658. Here too, the variation in light intensity causes the CVM algorithm to produce false positives (figure B.2). Since foreground objects are not present in the scene, the amount of misclassification is completely determined by the elusion. Figure 11.8 shows the elusion for each of the three sequences. Note that, for the first sequence, the elusion performance remains zero until a change in illumination occurs at frame 135 (see figure 11.8(a)). It steeply increases as the scene becomes brighter, reaching a deflection point around frame 190 at which the intensity reaches its maximum. Between frames 595 and 597 the illumination is reduced to its normal level: the elusion correspondingly drops to zero.

The performance of the second sequence (frames 672–968, figure 11.8(b)) experiences only minor influence of the elusion. The third sequence, presented in figure 11.8(c), during which a change of illumination occurs in the presence of foreground objects, has the largest average elusion among the three sequences. This is caused by both the overclassification of the foreground

objects and the misclassification due to the changed illumination.



(a) Frames 101–658 (b) Frames 672–968 (c) Frames 969–1190

Figure 11.8: Elusion performance results on the three sequences of the RIC *dataset.*

The change in illumination causes a natural shift in the brightness and saturation distribution. The saturation of the frames before (frame 968) and after (frame 985) the change is shown in figure 11.9(b). More remarkably, the color distribution shows a similar distribution shift. The distribution of the hue[6] is affected by an increase and decrease in illumination between e.g. frames 134 and 176 and frames 596 and 597. An impure white light may cause a slight change in the color distribution. However, more likely would be the influence of noise which, at low light intensity, affects the color recording of the camera CCD. At increasing intensity, the true colors will be recorded more accurately, resulting in a, possibly significant, difference from the previous scene.

11.4 Noise Behavior

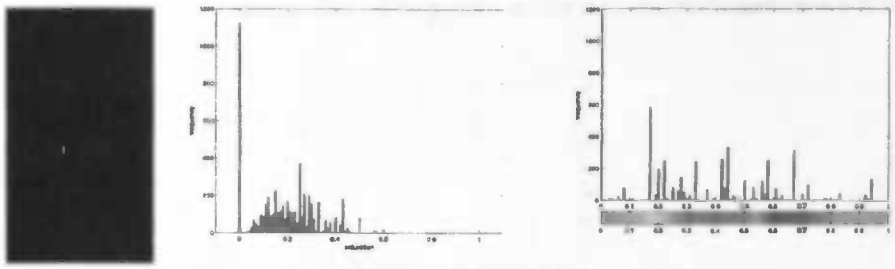
Image noise may cause trouble in a robust segmentation. Although all classification techniques inevitably suffer from this disturbance, it is of interest if, and to what extend, noise affects the quality of the CVM results. This section inspects the color variation induced by noise at various lighting intensities and discusses the influence of noise on the CVM method.

11.4.1 Noise and Low Brightness

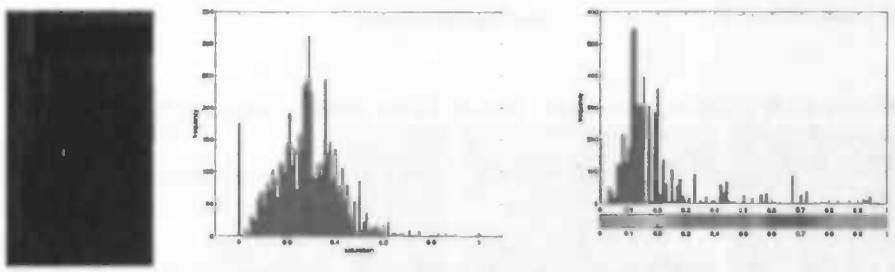
As was coined in section 8.2, noise affects low intensity scenes most heavily. Furthermore, the conclusion of section 11.3.3 demonstrated noise to induce shifts in recorded colors if the illumination variated. In continuation of this observation a test has been devised in which a scene is recorded on a large range of global light intensities, varying from daylight to sunset.

At each of the three levels of illumination shown in figure 11.10 the mean for each pixel over 100 sample images has been taken. These averaged images are displayed in the leftmost column of the figure. To measure the vulnerability of colors to noise, the hue has been extracted from the HSV averages. By excluding the saturation and brightness (i.e. the value component) in the HSV color model, this presents a notion on the level of noise-induced color fluctuations. The corresponding standard deviation then indicates the sensitivity to noise.

[6]The hue has been obtained by transcoding the RGB colorvalues of the input frames to HSV.



(a) Frame 968: low intensity (lights are off).



(b) Frame 985: high intensity (light are on).

Figure 11.9: Effect on the saturation and color distribution at varying illumination. Each row consists of a still of the upper right region in the RIC *sequence (left), a histogram plot of the saturation distribution—S from HSV—of the pixel values (center) and a histogram plot of the color distribution—H from HSV—of the pixel values (right).*

At high brightness (top row) the colors are almost constant. Only dark regions at the plant (at the left near the window) and the wall leaflet (right) exhibit a significantly increased variation. When the illumination diminishes the overall color deviation is strongly increased (center and bottom row), disallowing even a human observer to discern the scenic objects. However, in the full colorspace—including the saturation and brightness—a visual analysis of the scene can be made without too much effort (leftmost column).

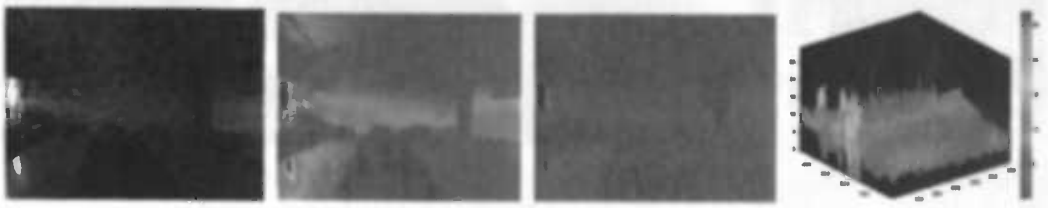
11.4.2 Noise and Test Statistic $\mathbb{D}^2$

The influence of noise on the segmentation performance under situations of low brightness can be made apparent by inspecting the distribution of the test statistic $\mathbb{D}^2$. Again, as in the previous section, a fixed scene at varying illumination has been inspected. The values of $\mathbb{D}^2$ have been recorded at each pixel over 100 consecutive frames. This empirical distribution has been plotted along with the average of these images in figure 11.11.

Each of the distributions plotted in figure 11.11 resembles a chi-square distribution. This is of no surprise to the alert reader who noticed that $\mathbb{D}^2$ is based on the summed squares of the vector elements W_I and W_B (see equation 9.23 and 9.24). Due to the absence of foreground objects in the scene, these vectors obey a zero-mean normal distribution with the variance dependent only on the level of noise. This leads to a chi-square distribution as depicted in figure 11.11.

(a) Full daylight (scene 1): mean RGB, mean S and standard deviation S in 2D and 3D.

(b) Recently initiated sunset (scene 3): mean RGB, mean S and standard deviation S in 2D and 3D.



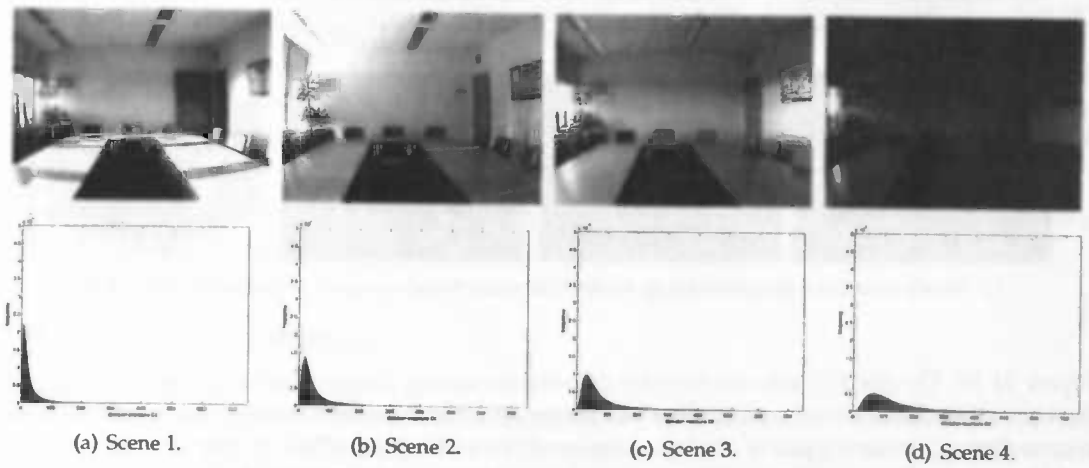
(c) Nearly completed sunset (scene 4): mean RGB, mean S and standard deviation S in 2D and 3D.

Figure 11.10: *The effect of noise on the color distribution among images at three illumination settings of the same scene. The leftmost images show an average over 100 successive images. The second columns of images show an intensity plot of the hue component (extracted from HSV) of these averages. The third and last column depict a 2D and 3D intensity plot of the standard deviation of the color values over the 100 sample images.*

Indeed, Mester et al. [MAD01] theoretically showed that $\mathbb{D}^2$ is proportional to a χ^2 distribution with $M-1$ degrees of freedom (with M the number of elements in both vectors W_I and W_B):

$$\mathbb{D}^2 \propto \sigma_d^2 \cdot \chi^2_{M-1} \tag{11.28}$$

with σ_d^2 the proportionality factor. Mester et al. indicated that, for signal vectors significantly larger than the expected value of the noise vector term, the multiplicative illumination factor would have no effect on the distribution. In practical situations, they motivate, low intensity scenes are uncommon and can therefore be excluded from further considerations. However, as the plots in figure 11.11 show, the variance of the distribution of test statistic $\mathbb{D}^2$ does increase considerably for images that are still subject to practical illumination conditions. This deviation in spread is governed by the proportionality factor σ_d^2.



(a) Scene 1. (b) Scene 2. (c) Scene 3. (d) Scene 4.

Figure 11.11: The empirical distribution of decision measure $\mathbb{D}^2$ at four different situations of illumination over all pixels in 100 consecutive images. The top row shows the average of these 100 images. The bottom row depicts the distribution of $\mathbb{D}^2$ for each illumination.

Chapter 12

Discussion

"All perception of truth is the detection of an analogy"

Henry David Thoreau (1817-1862)

"It's on the strength of observation and reflection that one finds a way. So we must dig and delve unceasingly."

Claude Monet

This part of the thesis discussed an approach towards illumination invariance using the Collinear Vector Model (CVM) based on the work of Mester et al. [MAD01]. Tests on the PETS 2001 dataset showed that the CVM algorithm is capable of segmenting relevant foreground objects if the CVM output is slightly enhanced by post-processing filters. To provide reliable data for a tracking mechanism, the performance requires some improvement. Furthermore, the segmentation results have to be clarified in the context of vector collinearity to obtain a meaningful analysis. The following sections focus on these clarifications and improvements.

12.1 Performance Clarification

Chapter 11 presented the observations on the RIC and PETS test results. Below a discussion is presented that aims to present an abstraction and generalization of these observations.

- The CVM algorithm has proven to be effective in segmenting objects in realistic scenes as is indicated by the results on the PETS 2001 dataset. The method is allows for natural variations in brightness throughout the scene. However, illumination change from or to colors close to black produce significant segmentation errors, as has been observed by the results on the RICdataset. The inability of the CVM algorithm to distinguish between foreground and background at these intensity levels is motivated by the high degree of noise relative to the color values.

- The use of a static background (i.e. fixed by the 100 first frames in the sequence) presents the CVM algorithm with an inability to adjust for incorrect segmentations. Because of this the performance will only restabilize if the background has transformed to its previous (or similar) state.
- The CVM model is quasi-invariant to illumination. The pseudo-invariance is based on the orthogonal distances of the feature vectors of the reference and input image to the estimated true vector. These distances vary by the level of brightness, due to the multiplicative nature of illumination.
- If noise is only additive (and not also multiplicative) and follows a Gaussian distribution, the reference image has averaged out the influence of noise in the signal. It then already presents a good estimate of the true vector, discarding the need for solving the optimization problem (finding the smallest eigenvalue of the combined feature vectors).
- The performance of the CVM algorithm is comparable to that of the competitive algorithm, of which some are dedicated to object tracking. However, if misclassification occurs in the CVM output, this is often caused by groups of pixels attained at locations distant from the ground truth objects. As discussed, these groups of pixels are the effect of the inability of the CVM algorithm to model color changes at low brightness or in the background. The absence of temporal knowledge in the model renders the algorithm less effective to tracking purposes.
- The framerate of the CVM algorithm (0.17134 fps for the 384 × 288 PETS 2001 dataset and 0.40124 fps for the 384 × 240 RICdataset) is insufficient for real-time purposes.

12.2 Improvements

Several shortcomings have been identified during the validation of the Collinear Vector Model in this thesis. These shortcomings concern theoretical aspects, performance improvements and extension towards object tracking. The proposed improvements are listed below.

- **Noise study:**
  Further study on the effects of noise on color values in the context of the collinear vector problem is required to understand to what extend illumination invariance is feasible. These examination should focus on low color intensities, at which noise shows to have a disastrous effect.
- **Color constancy:**
  The research field of color constancy provides proven theory and models that present a foundation for an extension and improvement of the Collinear Vector Model. As addressed briefly in section 8.4.1, color constancy models the human perception of color. For a succeeding study it is advised to focus on the combination of color constancy with vector collinearity.
- **Regional and global brightness:**
  The illumination invariance approach used in the CVM algorithm is based on orthogonal vector deviations from an estimated true signal. The model utilizes color values and intensity information by integration over a small spatial window. By extending to or including a regional and global brightness analysis a better estimation of the illumination may be obtained.

- **Different color model:**
  The RGB colorspace used for the CVM algorithm may not be well suitable for the purpose of color comparison. As discussed in chapter 8, components in a color model vary in noise sensitivity at different ranges of illumination intensity (e.g. blue in RGB and hue in HSV). An analysis of the effects of noise and color difference in other color models (e.g. the CIE model based on human perception) provides insight into optionally better models.

- **Gibbs/Markov random field:**
  The energy terms of the Gibbs/Markov random field can be extended by incorporating the proximity of salient edges or the spatial coherence of colors inside a local neighborhood. This information shall provide additional knowledge on the presence of foreground objects and likewise boosts the confidence level of thresholding.

- **Dynamic background:**
  The current model has not been designed to exclude small spatial changes in the background (e.g. waving trees or change in reflectance by a closing window). A non-static background model may cancel the effect of these non-relevant color fluctuations. The well-known Gaussian Mixture Model by Stauffer & Grimson [SG00] might prove useful for this purpose.

- **Temporal information:**
  The CVM algorithm is limited to processing of single frames. The segmentation results of previous frames are not incorporated, except its minor function as the initial energy in the Gibbs/Markov random field (see section 10.4). Temporal information, such as the labeling and colors of predecessing frames, may supply the algorithm with additional knowledge on the segmentation. For example, if the shape of the segmented objects are parameterized (i.e. represented by a contour description), matching object shapes among frames could improve the decision confidence of the segmentation.

- **GPU-based implementation:**
  The low quantitative performance of the CVM algorithm calls for improvements if it is be employed for real-time processing purposes. Griesser et al. [GDRNVG05] has developed a GPU-based version of the collinear vector algorithm of Mester et al. [MAD01] which shows a considerable speed-up. The CVM algorithm requires a similar implementation.

Part IV

NPR meets Computer Vision

Chapter 13

Pool of Intentions

"Creativity is the power to connect the seemingly unconnected."

William Plomer (1903-1973)

This chapter presents the *Pool of Intentions*, a video filter combining the LinearDelay and CVM algorithm. Section 13.1 introduces the idea behind the *Pool of Intentions* filter. The use of the CVM algorithm and its drawback is discussed in section 13.2, after which an alternative approach is presented in section 13.3 to simulate the effect of the CVM algorithm. The final result of the *Pool of Intentions* filter is shown in section 13.4.

13.1 Concept

It is well known that motion is the one of the most effective stimuli in human vision. The focus of the *Pool of Intentions* application is to attract and stimulate the attention of people by exploiting this knowledge.

The *LinearDelay* filter already expresses motion by applying a delaying effect to moving objects. The notion of movement could be further emphasized by extracting the moving objects out of their context. *Pool of Intentions* achieves this by replacing the static background by another image. The outline of this approach can be sketched as follows.

1. From a real-time recorded video, obtain a new frame.
2. Apply the *LinearDelay* filter to this frame. The filter uses the previous frames stored in its memory to construct the delayed output.
3. Apply object segmentation using the Collinear Vector Model to determine which image regions of the *LinearDelay* output are background. The CVM algorithm should be initialized by an initial sequence of 'empty' frames in which no foreground objects are present. (These frames could also be processed by the *LinearDelay* filter, since it only affects moving, hence foreground, objects).

4. Replace the pixels that have been designated as background during the previous step by their counterparts from another background image. Optionally, apply a fading effect to background pixels near the object contours to smoothly merge the images.

5. Present the result to the public.

13.2 Using the Collinear Vector Model

The experiments of the CVM algorithm on the PETS 2001 dataset in section 11.2 showed that the model is capable of effectively segmenting foreground regions under realistic conditions. Only when the change in illumination includes image regions of very low intensity will segmentation fail. This poses the CVM algorithm as a qualitatively sound candidate for the *Pool of Intentions* filter.

Unfortunately, the speed at which the CVM algorithm processes frames does not suffice the requirements of filtering real-time video streams. For 384×288 images a framerate of only 0.17134 fps was reached, which is similar to a processing time of approximately six seconds per frame.

13.3 Alternative Approach

To adhere to the real-time *Pool of Intention* requirements a fast segmentation algorithm is required. The CVM algorithm used a small neighborhood of pixels to construct the decision variable. To save processing time, the alternative approach presented in this section instead computes the segmentation by thresholding the per-pixel gray-value difference between the input and a background reference:

$$I_{mask}(x,y) = \begin{cases} 1 & \texttt{if } |I_{in}(x,y) - B(x,y)| > 25 \\ 0 & \texttt{otherwise} \end{cases} \tag{13.1}$$

where I_{in} is the image presented to the segmentation algorithm by the *LinearDelay* filter and B the background reference image constructed from the initial series of 'empty' input frames. Small foreground objects that are most likely introduced by noise are filtered by applying morphological area opening:

$$I_{open} = I_{mask} \circ^N \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} \tag{13.2}$$

where $\circ^N$ denotes the application of N iterations of area opening. To obtain the original foreground objects, I_{open} is AND-wise combined with the input image:

$$I_{and}(x,y) = \begin{cases} 1 & \texttt{if } I_{open}(x,y) == 1 \texttt{ and } I_{in}(x,y) == 1 \\ 0 & \texttt{otherwise} \end{cases} \tag{13.3}$$

To smoothly merge the foreground objects with the new background image a distance map $D(x, y)$ is computed of all the background pixels to the foreground object contours in I_{and}. The distances are determined by means of the $L2$-norm, thus using the Euclidean distance metric. Foreground pixels are assigned distance 0.

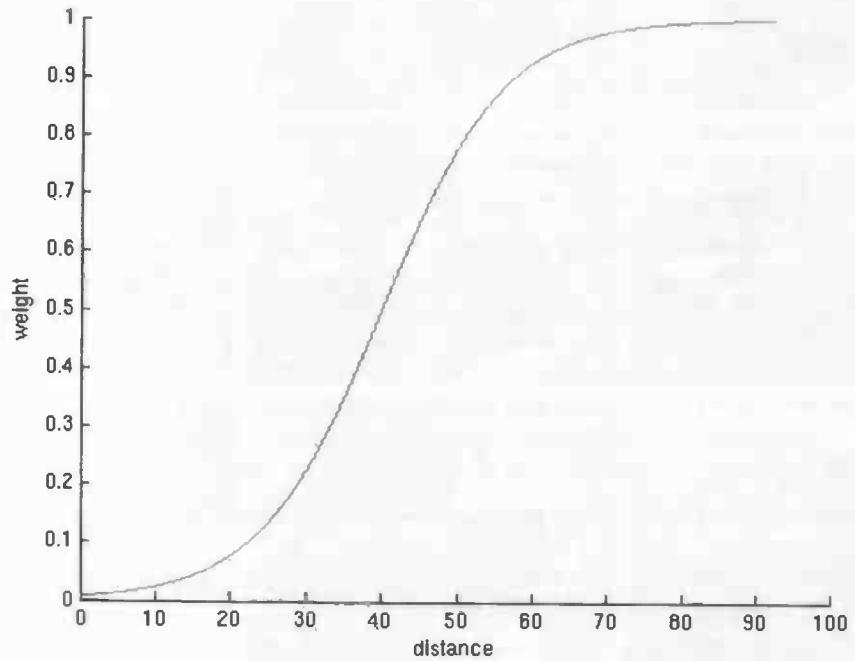


Figure 13.1: Logistic curve of the sigmoid function at smoothing factor $d = 8$ applied to the range of distance values $[0, 100]$.

Finally, the input and background images are combined using the distance values as smoothing weights. To obtain normalized weights and allow for a visually pleasing gradient the distance values are transformed by a special case of the logistic function, the sigmoid function (see figure 13.1):

$$w(x, y) = \frac{1}{1 + e^{-t(x,y)}} \tag{13.4}$$

where

$$t(x, y) = \frac{D(x, y) - 5s}{s} \tag{13.5}$$

and s is the smoothness factor controlling the strength of smoothing. This factor influences both the smoothing gradient and the distance at which the input image remains visually translucent.

A higher value of s allows a larger area around the foreground objects to be included into the foreground-background smoothing. Merging these images is achieved as follows:

$$I_{out}(x,y) = (1 - w(x,y)) \cdot I_{in}(x,y) + w(x,y) \cdot B(x,y) \quad (13.6)$$

13.4 Result

The alternative approach of the motion detection scheme presented in section 13.3, although less complex and rapid than the CVM algorithm, should prove itself useful for real-time application in both qualitative and quantitative evaluations.

Figure 13.2 shows the result of the *Pool of Intentions* filter by a sequence of images with a smoothness factor $s = 11$. The unaltered input images in the left column of this figure show a person walking into the scene from the right and taking a seat. At the right edge of the scene two other persons (in red and blue shirt) are present. The center column of the figure shows the result of the *LinearDelay* filter responding to any motion by a time delay that differs per image scanline. The green person 'flows' to his seat and produces a zig-zag output due to his wiggly behavior. Finally, motion detection determines the pixel locations at which change has occurred and replaces these pixels and its surrounding by the *LinearDelay* response (right column).

A few observations can be made that help to show the operation of the *Pool of Intentions* filter:

- The first row of images shows the green person walking towards the left side of the scene. Due to the delay, the *LinearDelay* filter shows the person only at the top few scanlines. The large region around the person's body in the center of the scene replaces the background image. This is caused by the frame difference registered for all pixels that are subject to the person's motion.
- The background images are smoothly merged with the recorded images. The foreground scene is gradually replaced by the background colors based on radial distances from the object in motion.
- The *Pool of Intentions* filter is implemented using multiple backgrounds used in alteration that fade into one another. This occurrence is visible in the filter output (right image) of the last row of images in figure 13.2.

The alternative approach to the motion detection scheme provides an improvement in computational speed. The CVM algorithm enforced a frame rate of approximately 0.17 fps for images of dimension 384 × 288. The alternative method allows a framerate of 8.0 fps with 800 × 600 images, a speed-up factor of 200.

Figure 13.2: Example of the Pool of Intentions filter splitted into the input (left), output of the LinearDelay filter (center) and output after motion detection (right).

Chapter 14

Conclusion

"Success is not final, failure is not fatal: it is the courage to continue that counts."

unknown quotee

This thesis discussed the design of an interactive digital filter to produce artistic effects that stimulates the spectator to act upon the presented video presentation. The distorted objects in motion have been extracted from the background and projected onto a different image. The results of the dual focus of this work are concluded in the following two sections. Chapter 13 discusses the merge of these two applications.

14.1 Video Distorting Mirror

The part of the thesis discussing the video distorting mirror was concerned with the following research question:

> How should an artistic, digital video application be developed that stimulates the fascination of the spectator and persuades him to contemplate on the underlying mechanism of the presented effects?

The research questions have been split up into four topics. The research described in part two of this thesis provides an answer to these topics, which will be summarized below.

1. **How would video art be fitted into scientific research?**
   In the field of Computer Graphics, Non-Photorealistic Rendering is concerned with the creation of non-realistic images. Their purpose is to aid artists in the production of new work, stimulate the cultivation of new realms of art and facilitate in the creation of digital masterpieces. It furthermore serves as a mean of investigation into the artistic process, allows for more suitable presentations of information and stimulates spectators to engage into visual communication. Analysis of the salient aspects in the artistic process may also allow for improved compression algorithms. Lastly, it provides new means of entertainment.

2. **What would be an interesting and easily produced artistic effect?**
   The *LinearDelay* algorithm is based on a delaying effect that differs among the scanlines in the video frames. This produces smoothened and stretched motion and functions as a digital distorting mirror.

3. **What is required to obtain this effect?**
   The *LinearDelay* algorithm has been developed as a plug-in to *mplayer*, providing an efficient and real-time performance. The algorithm has also been implemented using the *OpenCV* library for the purpose of merging with the object segmentation algorithm.

4. **Is the resulting effect of the developed artistic filter as expected?**
   A presentation at the information day of the University of Groningen has shown the filter to receive much attention and enthusiasm, despite the fact that the visitors were somewhat reserved. The video distorting filter provides the intended effect.

14.2 Collinear Vector Model

The part of the thesis discussing the Collinear Vector Model was concerned with the following research question:

> How could a foreground-background segmentation algorithm be developed that is, or approximates, illumination invariance such that changes in brightness do not affect the segmentation (and optionally tracking) performance of the algorithm in realistic scenarios?

The research questions have been split up into five topics. The research described in part three of this thesis provides an answer to these topics, which will be summarized below.

1. **What is the behavior of illumination and noise in signals processed by a CCD camera?**
   Variation in illumination can be modelled by a factor that influences the values of the color vectors. These factors may themselves be vectors, since the effect of illumination may vary between the color channels. Noise also affects the individual channels and can be both additive and multiplicative of nature.

2. **How could vector collinearity be used as a foundation of illumination invariant foreground-background segmentation?**
   A comparison of similar colors is affected by both noise and illumination. Difference in illumination will change the length of the color vectors, whereas noise would affect the direction. Colors are equal if their difference is induced by noise, which could be determined by measuring the orthogonal distances to the 'true' color vector. This renders the object segmentation quasi-invariant to illumination change.

3. **Which issues should be resolved to obtain an implementation of this algorithm?**
   Color information suffers from noise at (very) low brightness. To counter these effects partly, an extra dimension is added to the color vectors that increases the vector lengths and therefor provides a more reliable segmentation. The vulnerability of low color intensities also affects the value of the decision measure $\mathbb{D}^2$. Analysis of the distribution of this value showed that a fixed standard deviation is impractical.

4. **What is the performance of the algorithm in the presence of strong variation in illuminations?**
   The Collinear Vector Model is unable to achieve good segmentations if the illumination decreases to or increases from nearly black pixels. At these intensities, the color values experience high levels of noise that distort the true color values significantly. For common segmentation tasks, e.g. prepatory to object tracking, the model outperforms the algorithms from the PETS2001 algorithm competition. The drawback of the Collinear Vector Model is its high computational costs, which renders it impractical for real-time applications in its current form. A GPU-based implementation or a simplified or improved design is required for feasible usage.

5. **How is the algorithm ranked in comparison with competitive tracking algorithms based on the PETS 2001 dataset?**
   If boosted slightly by a simple post-processing filter (removing small objects), the Collinear Vector Model is capable of segmenting and tracking foreground objects throughout the scene. The performance exceeds that of the other methods except for one metric, which is caused by misclassified pixels at distant locations from the ground truth. The PETS 2001 dataset used in these tests showed to have some ground truth errors.

Bibliography

[Ado03] Adobe. *Adobe Photoshop CS Manual*. Adobe Systems, USA, 2003.

[AWK+05] J. Aguilera, H. Wildenauer, M. Kampel, M. Borg, D. Thirde, and J. Ferryman. Evaluation of motion segmentation quality for aircraft activity surveillance. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, October 2005.

[Azz96] A. Azzalini, editor. *Statistical Inference—Based on the likelihood*. Chapman & Hall/CRC, 1996.

[Bar98] K. Barnard. Modeling scene illumination colour for computer vision and image reproduction: a survey of computational approaches. Master's thesis, Simon Fraser University, 1998.

[BDBFG06] S. Battiato, G. Di Blasi, G.M. Farnella, and G. Gallo. A survey of digital mosaic techniques. In *Eurographics Italian Chapter Conference*, pages 129–135, 2006.

[Bov00] A. Bovik. *Handbook of Image & Video Processing*. Academic Press, Canada, 2000.

[Bro06] S. Brooks. Image-based stained glass. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1547–1558, 2006.

[BSL01] W.V. Baxter, V. Scheib, and M.C. Lin. Dab: Interactive haptic painting with 3D virtual brushes. In *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 461–468, 2001.

[BW86] D.H. Brainard and B.A. Wandell. Analysis of the retinex theory of color vision. *J. Opt. Am. A*, 3(10):1651–1661, 1986.

[CAS+97] C.J. Curtis, S.E. Anderson, J.E. Seims, K.W. Fleischer, and D.H. Salesin. Computer-generated watercolor. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 421–430, April 1997.

[CE91] T. Cockshott and D. England. Wet and sticky: supporting interaction with wet paint. In *In People and computers IV: Proceedings of the HCI '91 Conference*, pages 199–208, 1991.

[CMPC06] S. Calderara, R. Melli, A. Prati, and R. Cucchiara. Reliable background suppression for complex scenes. In *VSSN '06: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 211–214, New York, NY, USA, 2006. ACM Press.

[Coc91a] T. Cockshott. *Wat and sticky: a novel model for computer-based painting*. Phd thesis, University of Glasgow, 1991.

[Coc91b] T. Cockshott. *Wet and sticky: a novel model for computer-based painting*. PhD thesis, University of Glasgow, 1991.

[CPE92] T. Cockshott, J. Patterson, and D. England. Modelling the texture of paint. *Computer Graphics Forum.*, 11(3):217–226, 1992.

[DBG05] G. Di Blasi and G. Gallo. Artificial mosaics. *The visual computer*, 21(6):373–383, 2005.

[DBGP05] G. Di Blasi, G Gallo, and M. Petralia. Puzzle image mosaics. In *Proc. IASTED/VIIP 2005*, 2005.

[Dep88] E.F. Deprettere, editor. *SVD and signal processing: algorithms, applications and architectures*. North-Holland Publishing Co., 1988.

[DHJN02] J. Dobashi, T. Haga, H. Johan, and T. Nishita. A method for creating mosaic images using voronoi diagrams. In *Proc. Eurographics 2002*, pages 341–348, 2002.

[DM01] Y. Deng and B. Manjunath. Unsupervised segmentation of color-texture regions in image and videos. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 800–810, 2001.

[DS02] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 769–776, San Antonio, USA, 2002.

[DWL98] M.S. Drew, J. Wei, and Z. Li. On illumination invariance in color object recognition. *Pattern Recognition*, 31(8):1077–1087, 1998.

[EHD00] A.M. Elgammal, D. Harwood, and L.S. Davis. Non-parametric model for background subtraction. In *6th European Conference on Computer Vision*, pages 2:751–767, Dublin, Ireland, 2000.

[FCF96] G.D. Finlayson, S.S. Chatterjee, and B.V. Funt. Color angular indexing. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, pages 16–27, 1996.

[FDF05] G.M. Faustino and L.H. De Figueiredo. Simple adaptive mosaics. In *Proc. SIBGRAPI 2005*, pages 315– 322, 2005.

[Fer05] J.M. Ferryman, editor. *IEEE Int. Workshops on Performance Evaluation of Tracking and Surveillance (PETS)*, 2000–2005.

[For] The forsythe company. `http://www.frankfurt-ballett.de`.

[GCS02] B. Gooch, G. Coombe, and P. Shirley. Artistic vision: painterly rendering using computer vision techniques. In *Proceedings of NPAR*, pages 83–90, 2002.

[GDRNVG05] A. Griesser, S. De Roeck, A. Neubeck, and L. Van Gool. Gpu-based foreground-background segmentation using an extended colinearity criterion. In *Proceedings of Vision, Modeling, and Visualization (VMV) 2005*, pages 319–326, November 2005.

[GG84] S. Geman and d. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *IEEE Trans. Pattern Anal. Machine Intell.*, volume 6, 1984.

[GG01] B. Gooch and A.A. Gooch. *Non-Photorealistic Rendering*. A. K. Peters, Ltd., Natick, USA, 2001.

[GGD05] M. Grundland, C. Gibbs, and N.A. Dodgson. Stylized rendering for multiresolution image representation. In *Proceedings of SPIE*, pages 280–292, San Jose, USA, January 2005. Society of Photo-Optical Instrumentation Engineers.

[Gom60] E.H. Gombrich. *Art and Illusion: A Study in the Psychology of Pictorial Representation*, volume 35(5) of *Bollingen series*. Phaidon Press, London ,UK, 1960.

[Gru97] M. Grundland. Voronoimage, 1997. `http://www.cl.cam.ac.uk/~mg290/VoronoImage/index.html`.

[Hae90] P. Haeberli. Paint by numbers: abstract image representations. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 207–214, 1990.

[Hau01] A. Hausner. Simulating decorative mosaics. In *Proc. SIGGRAPH 2001*, pages 573–580, 2001.

[HHD99] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *IEEE ICCV'99 FRAMERATE Workshop*, 1999.

[HJO+01] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. In *Proceedings of ACM SIGGRAPH 2001*, pages 341–346, 2001.

[HM99] G.M. Haley and B.S. Manjunath. Rotation invariant texture classification. *IEEE Trans. Image Processing*, 8:256–269, 1999.

[HS81] B.K.P. Horn and B.G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[Hub95] D. Hubel. *Eye, Brain and Vision*. Scientific American Library, New York, 1995.

[Ino] K. Inoue. Generation of stained-glass-like images by using quantized mode filter.

[JDWR00] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld. Detection and location of people in video images using adaptive fusion of color and edge information. In *Proc. IAPR International Conference on Pattern Recognition*, pages 627–630, 2000.

[JEGPO02] P.-M. Jodoin, E. Epstein, M. Granger-Piche, and V. Ostromoukhov. Hatching by example: a statistical approach. In *Proceedings NPAR 2002 Symposium on Non-Photorealistic Animation and Rendering*, pages 29–36, 2002.

[KCL+98] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson. Advances in cooperative multi-sensor video surveillance. In *Proceedings of the DARPA Image Understanding Workshop*, volume 1, pages 3–24, November 1998.

[KGYS05] M. Karaman, L. Goldmann, D. Yu, and T. Sikora. Comparison of static background segmentation methods. In *Visual Communications and Image Processing (VCIP '05)*, July 2005.

[Kin99] M. King. The tyranny and the liberation of three-space: a journey by ray-tracer. *Digital Creativity*, 10(4):215–227, 1999.

[KN95] A. Kosaka and G. Nakazawa. Vision-based motion tracking of rigid objects using prediction of uncertainties. In *International Conference on rRbotics and Automation*, pages 2637–2644, 1995.

[KP02] J. Kim and F. Pellacini. Jigsaw image mosaics. In *Proc. SIGGRAPH 2002*, pages 657–664, 2002.

[KSJ00] E.R. Kandel, J.H. Schwartz, and T.M. Jessell, editors. *Principles of Neural Science*. McGraw-Hill, New York, USA, 2000.

[LF05] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: a survey. *Foundations and Trends of Computer Graphics and Vision*, 1(1):1–89, 2005.

[Li01] S.Z. Li. *Markov random field modeling in image analysis*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging understanding workshop.*, pages 121–130, 1981.

[LM71] E.H. Land and J.J. McCann. Lightness and retinex theory. *J. Opt. Am. A*, 61(1):1–11, 1971.

[MAD01] R. Mester, T. Aach, and L. Dümbgen. Illumination-invariant change detection using a statistical colinearity criterion. In *Proc. 23rd DAGM Symp.*, 2001.

[Mal80] W.A. Malila. Change vector analysis: an approach for detecting forest changes with landsat. In *Proc. of the 6th annual Symposium on Machine Processing of Remotely Sensed Data*, pages 326–335, 1980.

[MfHCW02] M. Mei-fang Huang, K. Chen, and H. Wang. Towards retrieval of video archives based on the speech content. In *Proc. International Symposium on Chinese Spoken Language Processing (ISCSLP2002)*, 2002.

[Mou03] D. Mould. A stained glass image filter. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 20–25, 2003.

[MPl06] Mplayer - the movie player, *version 1.0rc1*, October 2006. `http://www.mplayerhq.hu/`.

[oA06] National Gallery of Art. Neoclassicism: 18th- and 19th-century france, 2006. Washington, USA.

[Ope06] Opencv—the open source computer vision library, November 2006. `http://www.intel.com/technology/computing/opencv/`.

[Ots79] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[Phi88] D. Phillips. Siggraph '88 panels proceedings, 1988.

[Pin] Pinakothek münchen. `http://www.pinakothek.de`.

[RAAKR05] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: A systematic survey. *IEEE Trans. Image Processing*, 14(3):294–307, 2005.

[Ric97] P. Richens. Beyond photorealism. *Architects' Journal*, 1997.

[RJB+03] Y. Reibel, M. Jung, M. Bouhifd, B. Cunin, and C. Draman. Ccd or cmos camera noise characterisation. *The European Physical Journal Applied Physics*, 21(1):75–80, 2003.

[SALS96] M. Salisbury, S.E. Anderson, D. Lischinski, and D.H. Salesin. Scale-dependent reproduction of pen-and-ink illustrations. In *SIGGRAPH 96 Conference Proceedings*, pages 461–468, 1996.

[SB99a] M.C. Sousa and J.W. Buchanan. Computer-generated graphite pencil rendering of 3d polygonal models. *Computer Graphics Forum*, 18(3):195–208, 1999.

[SB99b] M.C. Sousa and J.W. Buchanan. Observational model of blenders and erasers in computer-generated pencil rendering. In *Grapics Interface Proceedings 1999*, pages 157–166, 1999.

[SB00] M.C. Sousa and J.W. Buchanan. Observational models of graphite pencil materials. *Computer Graphics Forum*, 19(1):27–49, 2000.

[SD02] A. Santella and D. DeCarlo. Abstracted painterly renderings using eye-tracking data. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 75–83, Annecy, France, 2002.

[SDC04] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):479–494, April 2004.

[SG99] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. International Conference on Pattern Recognition*, pages 246–252, 1999.

[SG00] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(8):747–757, August 2000.

[SGS05] S. Schlechtweg, T. Germer, and T. Strothotte. Renderbots - multi-agent systems for direct image generation. *Computer Graphics Forum*, 24(2):137–148, 2005.

[SH97] R. Silvers and M. Hawley, editors. *Photomosaics*. Henry Holt & Company, 1997.

[SK94] W. Skarbek and A. Koschan. Colour image segmentation—a survey. Technical report, Institute for Technical Informatics, Technical University of Berlin, October 1994.

[Sma91] D. Small. Simulating watercolor by modeling diffusion, pigment, and paper fibers. In *Proceedings of SPIE '91*, pages 140–146, 1991.

[Smi78] A.R. Smith. Color gamut transform pairs. In *Proc. SIGGRAPH 1978*, pages 12–19, 1978.

[Str86] S. Strassmann. Hairy brushes. *SIGGRAPH Comput. Graph.*, 20(4):225–232, 1986.

[Sut64] I.E. Sutherland. Sketchpad: A man-machine graphical communication system. In *DAC '64: Proceedings of the SHARE design automation workshop*, pages 6329–6346, New York, USA, 1964.

[SWHS97] M. Salisbury, M.T. Wong, J.F. Hughes, and D.H. Salesin. Orientable texture for image-based pen-and-ink illustration. In *SIGGRAPH 97 Conference Proceedings*, pages 401–406, 1997.

[TAM00] D. Toth, T. Aach, and V. Metzler. Bayesian spatio-temporal motion detection under varying illumination. In *European Signal Processing Conference (EUSIPCO)*, pages 2081–2084, Tampere, Finland, 2000.

[Tat04] L.G. Tateosian. Npr: Art enhancing computer graphics. Technical Report TR-2004-17, Knowledge Discovery Lab, North Carolina State University, 2004.

[Ter86] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(4):413–242, 1986.

[TKBM99] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *IEEE International Conference on Computer Vision*, pages 255–261, 1999.

[UI04] K. Urahama and K. Inoue. Stained glass images by using quatized mode filters. *Trans. ITE*, 58(10):1519–1521, 2004.

[VT89] A.H. Vermeulen and P.P. Tanner. Pencilsketch - a pencil-based paint system. In *Grapics Interface Proceedings 1989*, pages 138–143, 1989.

[WADP97] C.R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: real-time tracking of the human body. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(7):780–785, 1997.

[Weh05] S. Wehmeier, editor. *Oxford Advanced Learner's Dictionary*. Oxford University Press, Oxford, New York, 7th edition, 2005.

[WS82] G. Wyszecki and W.S. Stiles, editors. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley Series in Pure and Applied Optics, New York, USA, 1982.

[WS94] G. Winkenbach and D.H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 91–100, 1994.

[WWdV03] M.H.F. Wilkinson, T. Wijbenga, and G. de Vries. Blood vessel segmentation using moving-window robust automatic threshold selection. In *International Conference on Image Processing - ICIP 2003*, volume 2, pages 1093–1096, Barcelona, Spain, September 2003. IEEE Signal Processing Society.

[YF05] D.P. Young and J.M. Ferryman. Pets metrics: On-line performance evaluation service. In *The Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, October 2005.

Index

Part V

Appendices

Appendix A

PETS 2001 Dataset 1 Camera 1

A.1 Ground Truth

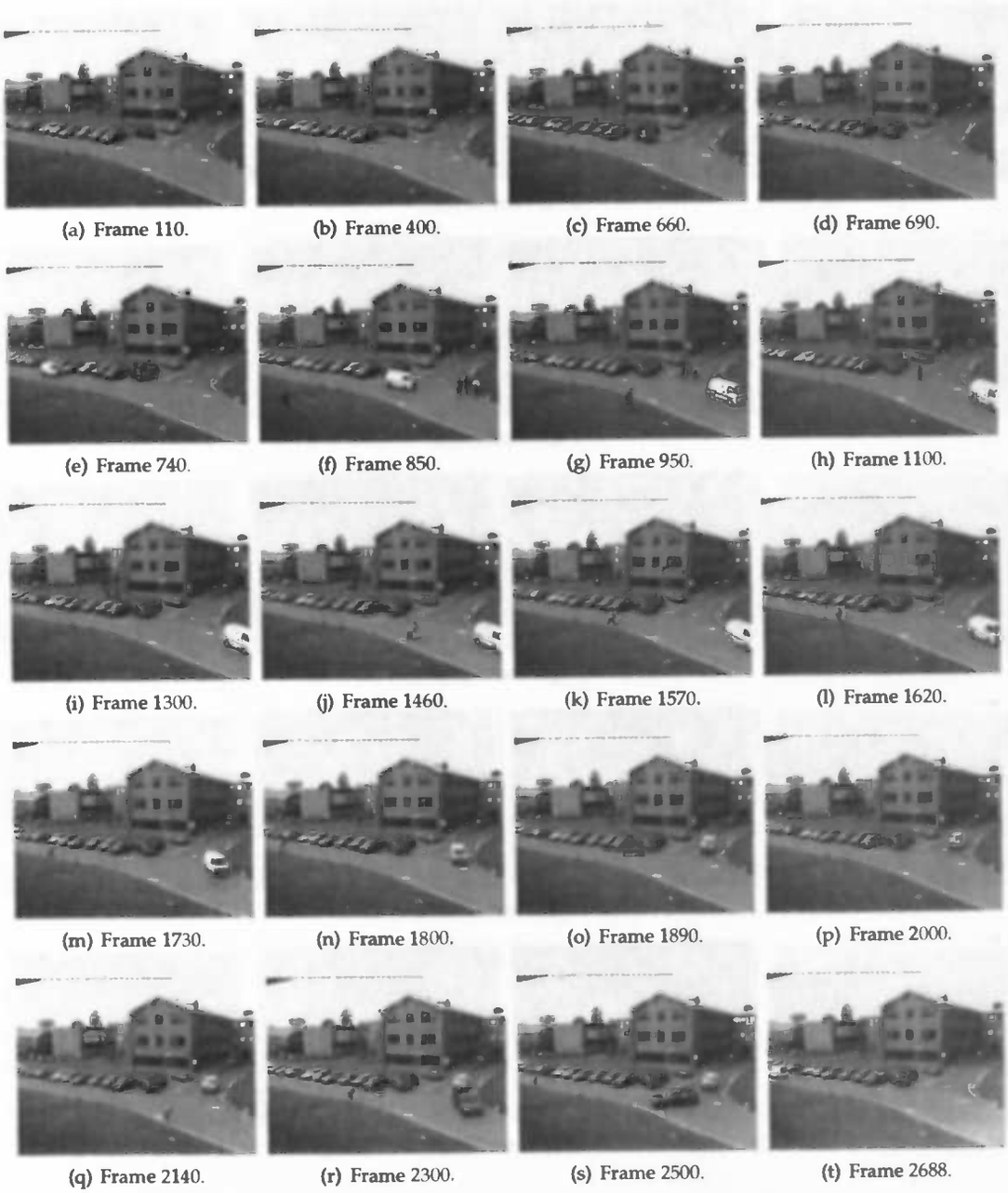
The ground truth of the PETS 2001 datasets is presented in an XML format, listing the location and shape of foreground objects by a description of the bounding boxes for each frame an object is present in. A snippet of one of the XML files, representing the first object appearing in the sequence, is given below.

```
<JPR_EVENT_SEQUENCE CLASS="PERSON" DESCRIPTION="Red-Coat Female"
FRAME_CREATED="110" FRAME_DESTROYED="653" ID="0">
  <OBJECT TIME="110">
    <OBSERVATION><BOUNDING_BOX BOTTOM="247" LEFT="1" RIGHT="15" TOP="287"/>
      <CENTROID CENTER_X="8" CENTER_Y="267"/>
      <POSITION BASE_X="8" BASE_Y="247"/>
    </OBSERVATION><OCCLUSION>NOTOCCLUDED</OCCLUSION>
  </OBJECT>

  <OBJECT TIME="111">
    <OBSERVATION><BOUNDING_BOX BOTTOM="246" LEFT="3" RIGHT="17" TOP="286"/>
      <CENTROID CENTER_X="10" CENTER_Y="266"/>
      <POSITION BASE_X="10" BASE_Y="246"/>
    </OBSERVATION><OCCLUSION>NOTOCCLUDED</OCCLUSION>
  </OBJECT>

  <OBJECT TIME="112">
    <OBSERVATION><BOUNDING_BOX BOTTOM="246" LEFT="4" RIGHT="18" TOP="286"/>
      <CENTROID CENTER_X="11" CENTER_Y="266"/>
      <POSITION BASE_X="11" BASE_Y="246"/>
    </OBSERVATION><OCCLUSION>NOTOCCLUDED</OCCLUSION>
  </OBJECT>

  ...

</JPR_EVENT_SEQUENCE>
```

A.2 Input and Output Images

This appendix contains a selection of PETS 2001 input images and their corresponding ground truth, CVM output and the enhanced results. The figures listed in the appendix are:

- Figure A.1 shows 20 frames of relevant events from the PETS *2001 Dataset 1 Camera 1* video.
- Figure A.2 and A.3 both show the ground truth contours, the output of the CVM algorithm and the enhanced results of the postprocessing module of five frames from the PETS *2001 Dataset 1 Camera 1* video.



(a) Frame 110. (b) Frame 400. (c) Frame 660. (d) Frame 690.

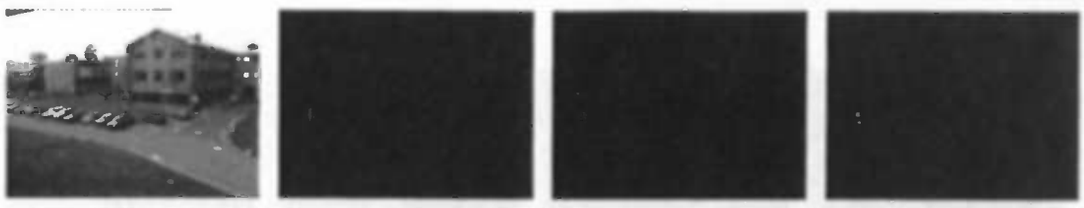
(e) Frame 740. (f) Frame 850. (g) Frame 950. (h) Frame 1100.

(i) Frame 1300. (j) Frame 1460. (k) Frame 1570. (l) Frame 1620.

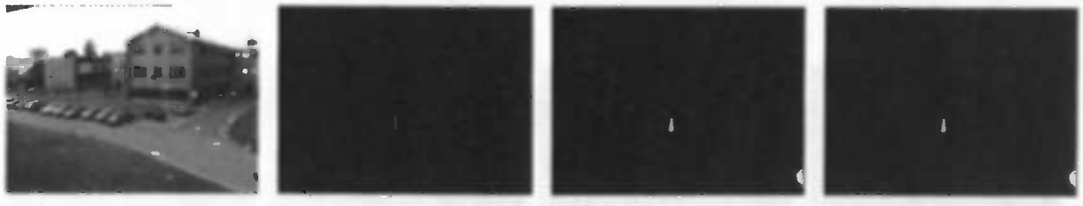
(m) Frame 1730. (n) Frame 1800. (o) Frame 1890. (p) Frame 2000.

(q) Frame 2140. (r) Frame 2300. (s) Frame 2500. (t) Frame 2688.

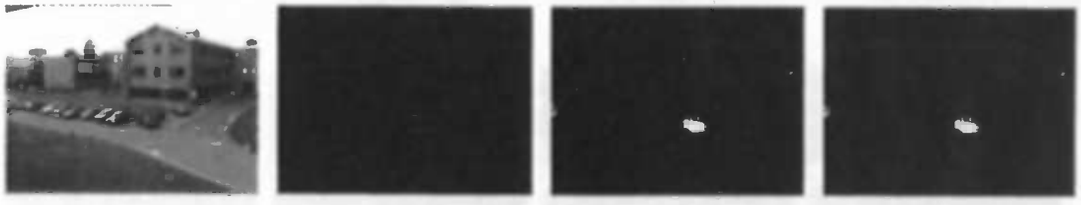
Figure A.1: *Stills from the* PETS *2001 dataset (total 2688 frames, dimension* 384×288*).*



(a) Frame 219.

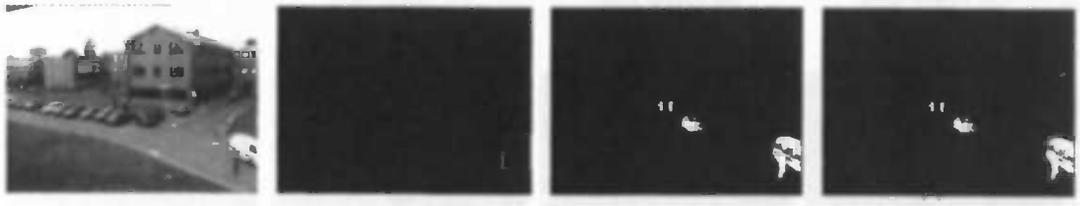


(b) Frame 450.



(c) Frame 677.

(d) Frame 997.

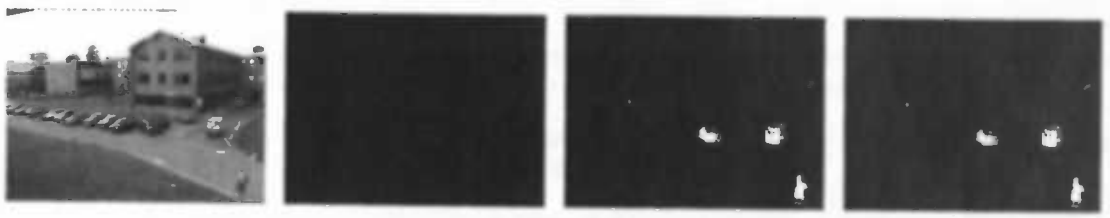


(e) Frame 1245.

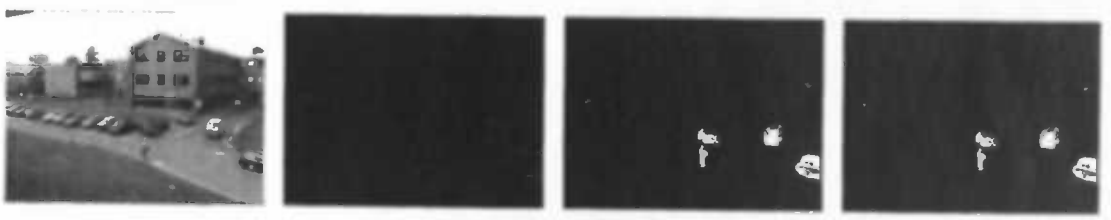
Figure A.2: Input (first column), ground truth bounding boxes (second column), CVM *output (third column) and enhanced mask (fourth column) of the* PETS *2001 dataset.*

(a) Frame 1510.

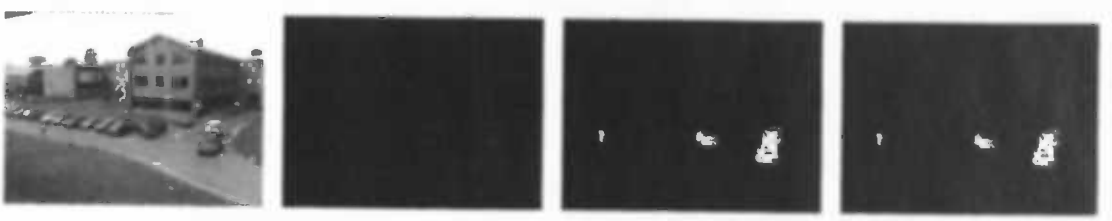
(b) Frame 1854.



(c) Frame 2033.



(d) Frame 2189.



(e) Frame 2434.

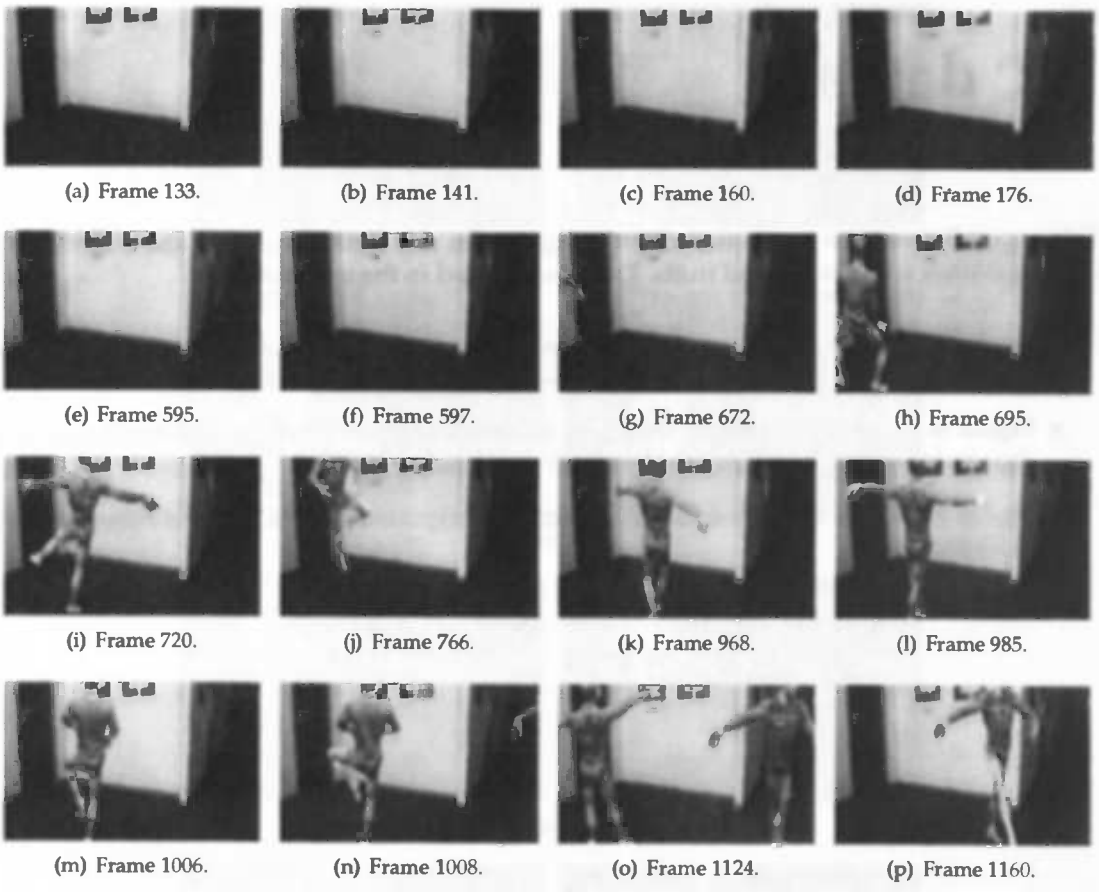
Figure A.3: Input (first column), ground truth bounding boxes (second column), CVM *output (third column) and enhanced mask (fourth column) of the* PETS *2001 dataset.*

Appendix B

RIC dataset

This appendix contains a selection of RICinput images and a comparison of the output of the CVM algorithm with the ground truth. The figures listed in the appendix are:

- Figure B.1 shows 16 frames of relevant events from the RICsequence in which rapid illumination change is present at frames 133–176, 595–597 and 968–1006.
- Figure B.2 and B.4 show the results of incorrect segmentations caused by illumination change from very low (near black colors) to normal brightness.
- Figure B.3 shows the segmentation quality of foreground objects, without strong changes in illumination.



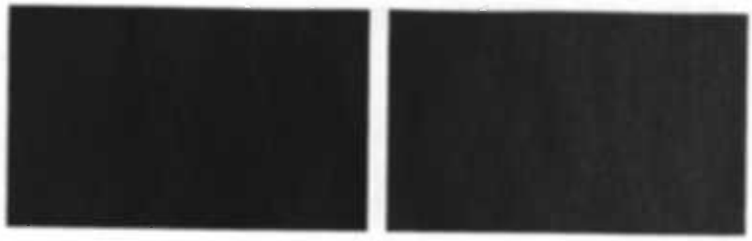
(a) Frame 133. (b) Frame 141. (c) Frame 160. (d) Frame 176.

(e) Frame 595. (f) Frame 597. (g) Frame 672. (h) Frame 695.

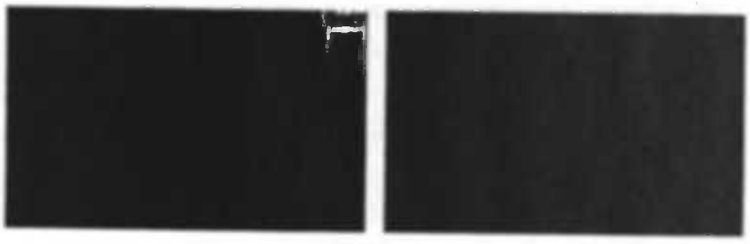
(i) Frame 720. (j) Frame 766. (k) Frame 968. (l) Frame 985.

(m) Frame 1006. (n) Frame 1008. (o) Frame 1124. (p) Frame 1160.

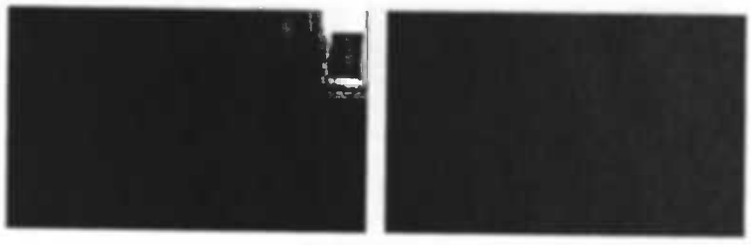
Figure B.1: Stills from the RIC *dataset (total 1191 frames, dimension* 384×240*).*



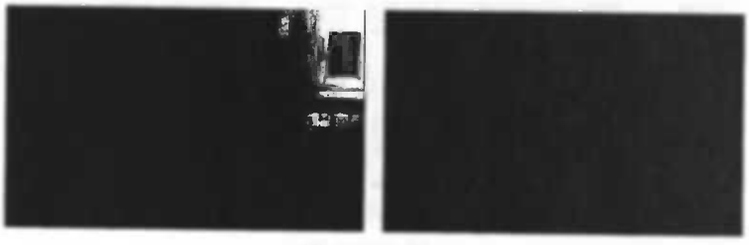
(a) Frame 133.



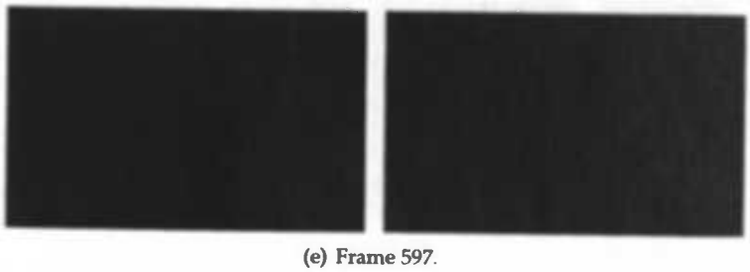
(b) Frame 141.



(c) Frame 160.



(d) Frame 176.



(e) Frame 597.

Figure B.2: *Segmentation results (left) and ground truth (right) of the* CVM *algorithm on sequence 101 to 658 the* RIC *dataset.*

(a) Frame 672.

(b) Frame 695.

(c) Frame 720.

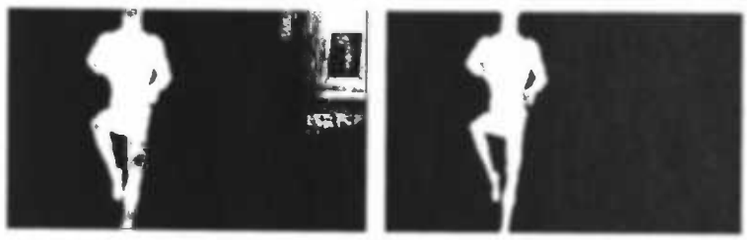
(d) Frame 766.

(e) Frame 968.

Figure B.3: Segmentation results (left) and ground truth (right) of the CVM *algorithm on sequence 672 to 968 of the* RIC *dataset.*

(a) Frame 969.

(b) Frame 985.



(c) Frame 1006.

(d) Frame 1124.

(e) Frame 1160.

Figure B.4: Segmentation results (left) and ground truth (right) of the CVM *algorithm on sequence 969 to 1190 the* RIC *dataset.*

Appendix C

The Collinearity Criterion

the orthogonal distance of the two vectors has been defined in section 9.1 by

$$D^2 = |d_I|^2 + |d_B|^2 . \tag{C.1}$$

Let us represent each of the distances d_I and d_B by d_*. The distance can then be defined by

$$|d_*|^2 = |W_*|^2 - |r_*|^2 \tag{C.2}$$

with the vector $\mathbf{r}_*$ the projection of W_* onto s. The vector $\mathbf{r}_*$ can be found by solving

$$|\mathbf{r}_*| = ||W_*| \cdot \cos\varphi_*| . \tag{C.3}$$

From the definition of the inner product (denoted by $\bullet$) we can deduce that

$$||W_*| \cdot \cos\varphi_*| = \frac{|W_* \bullet \mathbf{s}|}{|\mathbf{s}|} \tag{C.4}$$

$$= \frac{|W_*^\top \cdot \mathbf{s}|}{|\mathbf{s}|} . \tag{C.5}$$

With s of unity length ($|\mathbf{s}| = 1$) the above equation simplifies to

$$||W_*| \cdot \cos\varphi_*| = |W_*^\top \cdot \mathbf{s}| . \tag{C.6}$$

Substituting $|\mathbf{r}_*|$ with the righthand-side equivalent in equation C.6 then yields

$$|d_*|^2 = |W_*|^2 - |W_*^\top \cdot \mathbf{s}|^2 \tag{C.7}$$

and the collinear deviation becomes

$$D^2 = |W_I|^2 + |W_B|^2 - |W_I^\top \cdot \mathbf{s}|^2 - |W_B^\top \cdot \mathbf{s}|^2 . \tag{C.8}$$

Unfortunately, as the signal s is unknown, this problem of finding minimum distance cannot be solved in the current form. However, $|W_I^\top \cdot \mathbf{s}|^2 - |W_B^\top \cdot \mathbf{s}|^2$ boils down to computing the correlation matrix of the vectors W_I and W_B. To clarify this observation, let us form the $2 \times N$ matrix M with

$$\mathbf{M} = \begin{bmatrix} W_I^\top \\ W_B^\top \end{bmatrix}, \qquad \mathbf{M} \cdot \mathbf{s} = \begin{bmatrix} W_I^\top \cdot \mathbf{s} \\ W_B^\top \cdot \mathbf{s} \end{bmatrix}. \tag{C.9}$$

Expansion of $\mathbf{M} \cdot \mathbf{s}$ produces

$$|\mathbf{M} \cdot \mathbf{s}|^2 = \mathbf{s}^\top \cdot \mathbf{M}^\top \cdot \mathbf{M} \cdot \mathbf{s} = |W_I^\top \cdot \mathbf{s}| + |W_B^\top \cdot \mathbf{s}| \tag{C.10}$$

which yields the distance measure

$$D^2 = |W_I|^2 + |W_B|^2 - \mathbf{s}^\top \cdot \mathbf{M}^\top \cdot \mathbf{M} \cdot \mathbf{s} \tag{C.11}$$