A NEW BAYESIAN RELAXATION ALGORITHM FOR MOTION ESTIMATION AND SEGMENTATION IN THE PRESENCE OF TWO AFFINE MOTIONS

by

Alexander Strehl, Vordipl.-Inf. Univ.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

August 1998

A NEW BAYESIAN RELAXATION ALGORITHM FOR MOTION ESTIMATION AND SEGMENTATION IN THE PRESENCE OF TWO AFFINE MOTIONS

APPROVED:

Supervisor: _____

J. K. Aggarwal

_____

W. S. Geisler III

Dedicated to my parents

for their love, patience and measureless support

A NEW BAYESIAN RELAXATION ALGORITHM FOR MOTION ESTIMATION AND SEGMENTATION IN THE PRESENCE OF TWO AFFINE MOTIONS

Publication No. _____

Alexander Strehl, M.S.E.
The University of Texas at Austin, 1998

Supervisor: J. K. Aggarwal

Describing an image sequence in terms of a small number of coherently moving segments is useful for tasks ranging from autonomous vehicle navigation to video compression. A new probabilistic relaxation algorithm is proposed to perform motion estimation as well as object segmentation by using an iterative Bayesian approach. In the first stage, optical flow with local confidence estimates is computed. The second stage uses high confidence locations to alternately estimate parameters of two affine motions and segment the current image in two motion regions. This procedure is iterated in a stochastic relaxation framework to minimize the error of the fitted affine models until the motion parameter estimates converge. Applications of our motion estimation algorithm to stabilize sequences, detect and track objects and obtain their trajectories are presented. Our algorithm's performance is illustrated on synthetic and real image sequences.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Area Overview

Almost every task a human being accomplishes is guided by its visual system. Light intensity changes sensed with the eyes are analyzed to obtain critical information about the environment. Computer vision (CV) research deals with the use of modern computer technology to perform visual perception tasks. Presently, artificial vision systems perform successfully only in very constrained and rather simple environments, while humans effortlessly analyze complex natural scenes in realtime and almost without error. Learning from the nature's evolutionary solutions to improve artificial vision systems can be very rewarding, and links computer vision with neuroscience and psychology [1] [2] [3] [4].

In this thesis, dynamic scenes are investigated [5] [6] [7] [8] [9] [10]. Changes in a scene can be caused by camera motion, object motion, illumination changes, changes in object structure, size or shape. There is currently no system that takes all these effects into account. Most systems specialize in modeling one or two causes of these. In our case, we will only consider the effects of camera and object motion. Consequently, an image sequence can be classified into one of the four following categories:

- Stationary Camera and Stationary Objects (SCSO)

- Stationary Camera and Moving Objects (SCMO)
- Moving Camera and Stationary Objects (MCSO)
- Moving Camera and Moving Objects (MCMO)

The SCSO category is equivalent to static scene analysis and will not be considered further [11] [12] [7]. In the past, SCMO scenarios have received the most attention in the area of dynamic scene analysis. The objective of the analysis is usually to detect moving objects, recognize them and compute their motion characteristics. While the major part of the image does not change over time, the region of interest is the part of the image that changes. These changes are assumed to be due to object motion. Image changes can simply be detected by thresholding gray-level difference images. Segmentation can be done by clustering regions with changes [13] [14].

Static scenes observed with a moving camera (MCSO) have recently gained a lot of attention. In this scenario, estimating scene structure and obtaining a description of the camera motion are often the tasks to be solved [15] [16]. In this scenario, the observer moves through a rigid environment. Heeger and Jepson propose in [17] an algorithm to retrieve observer motion parameters from optical flow fields in the image plane. The motion of the observer can be represented by a translational and a rotational component. The rotational component yields no information about scene structure, and it is desired to separate this component from the translational component. Once the translational observer motion is known, this can be used to compute the scene structure from the flow field [18]. This is made possible by the fact

that feature points move slower with increasing distance from the translating observer (parallax problem). A comparison of present algorithms to compute observer motion can be found in [19]. Burger and Bhanu propose a robust system in [20] to estimate ego-motion[1] parameters for autonomous land vehicles by computing a fuzzy focus of expansion (FOE) from tracked features.

The most general (and probably the most difficult) case of dynamic scene analysis, namely MCMO, has been virtually ignored for a long time. Recently many systems have been proposed [21] [22] [23] [24] [25] [26]. No promising unified approach has yet been found. Compared to SCMO sequences, objects cannot be detected by simple difference images, because every image location is moving due to the camera motion. On the other hand, MCSO approaches fail because multiple rigid motions are present in MCMO scenes and the motion boundaries are unknown. Our proposed algorithm falls into this category and presents a solution for a subset of those sequences with two affine motions.

Once obtained, the extracted motion information is a very powerful cue to solve various tasks in diverse areas like robotics or multimedia. Applications include:

- 3–D scene structure estimation
- Active vision systems
- Autonomous vehicle navigation

[1] Camera motion is also called ego-motion, self-motion or observer motion throughout this thesis.

- Object detection, tracking and recognition
- Surveillance
- Efficient video encoding
- Video enhancement, stabilization

This list names only a few examples and currently the number of applications of versatile motion information seems to increase rapidly.

1.2 Problem Description

This thesis presents a new approach to the motion estimation problem in the moving camera and moving object case (MCMO). We assume that one independently moving object (IMO) is observed by a moving (or stationary, as a special case of moving) camera providing monocular gray-level images. The environment is not further constrained and no additional information on camera parameters, camera motion or scene structure is available.

The desired result is a compact description of the camera and object motion obtained only by analyzing a gray-level image sequence. We model these two motions, each as an affine transform of its coordinates. This is called the affine motion model and has six parameters per motion. So in our case the compact description is presented in the form of six parameters for the camera and six parameters for the object motion. Knowledge about the IMO's starting position in conjunction with the computed motion characteristics enables us to track it over an image sequence. Other applications of the obtained motion in-

formation that are presented in this thesis are simultaneous image stabilization [27] [28] and mosaicing [29] [30].

1.3 Previous Approaches

Recently many advances have been made in multiple motion scenarios. Active contour trackers can be used to track an object over time with a moving camera [31] [32]. However this approach requires objects with a distinct contour. In general a parametric function, capable of modeling this contour, must be known in advance. Other drawbacks of this approach include its failure in cases involving occluded or fragmented motion and the necessity to initialize the contour. Irani presents in [33] a method for detecting and tracking multiple transparent motions. Black and Anandan propose a robust estimation framework for multiple motions in [21] and apply it to several standard techniques for recovering optical flow. Adiv in [34] groups segments of the optical flow based on rigid 3–D motion constraints. Wang and Adelson [35] use optical flow to segment multiple motions through a k-means parameter clustering technique. Various advances have been made by Davis using systems employing velocity tuned filters [36].

1.4 Contribution of This Thesis

In this thesis a new probabilistic framework to estimate affine motion parameters from monocular gray-level images is introduced. The proposed algorithm performs the following two steps:

1. Obtain velocity vectors[2] and confidence estimates for each pixel using template-matching. We use the maximum likelihood estimate of all possible displacements as the velocity vector and the variability of the likelihood over the template searchwindow as a reciprocal confidence value. (Chapter 2)

2. Iteratively cluster high confidence pixels based on their affine motion parameters. This is done by relaxing the probabilities that a velocity vector is generated by a certain motion class. Deciding which motion caused the image velocity at a certain pixel is done with a Bayes classifier [37] . (Chapter 3)

This approach assumes very little knowledge about the presented scene. We do not require moving objects to be compact, so fragmented motion poses no problem. The object's shape does not have to be known in advance, unlike in a contour tracking framework [31] [32].

However, we do assume that the present motion can be approximated satisfactorily with an affine model and that two motion classes are present in the scene (e.g., background motion[3] and object motion). This is a very reasonable assumption in our scenarios, for example an independently moving object (first motion class) viewed by a camera on a moving platform (background is second motion class).

[2]Optical flow, displacement field and velocity vectors are used synonymous in this thesis and denote the 2–D vector field representing the estimated pixel-wise motion direction and magnitude between two successive frames.

[3]Background motion is caused by the moving camera. Background motion equals camera motion with inverse orientation.

The advantages of the presented algorithm are that it is based on a versatile probabilistic framework throughout all stages and that it is designed for a rather unconstrained environment. Figure 1.1 gives a graphical overview of our system and its environment. The light shaded area represents our system. Two gray-level frames are used to compute optical flow with local confidence estimates. In the second stage, illustrated with the dark shaded area, this flow and previous motion parameters are used to estimate the inter-frame motion parameters. This is done in a new iterative relaxation framework, our original contribution. Within this framework Bayesian segmentation of the flow and motion parameter estimation are performed alternately until the estimates converge.

1.5 Organization

The rest of this thesis is organized as follows. Chapter 2 describes and explains the first stage of our algorithm, the computation of optical flow. Why we did not use a standard method for optical flow computation is explained in section 2.1. How we compute local pixel velocities and confidence values is described in sections 2.2 through 2.5. Section 2.6 shows how the presented flow computation is applied in a Gaussian resolution hierarchy. Chapter 3 describes how the results from the algorithm presented in chapter 2 are used to conduct motion analysis in a probabilistic relaxation framework. The relaxation iteratively estimates parameters (section 3.2) and segments (section 3.3) the reliable parts of an image pair. Upon convergence the estimation of the desired parameters for motion between two frames is completed. Chapter 4 explains how our proposed algorithm is used when more than two frames are available. Chapter

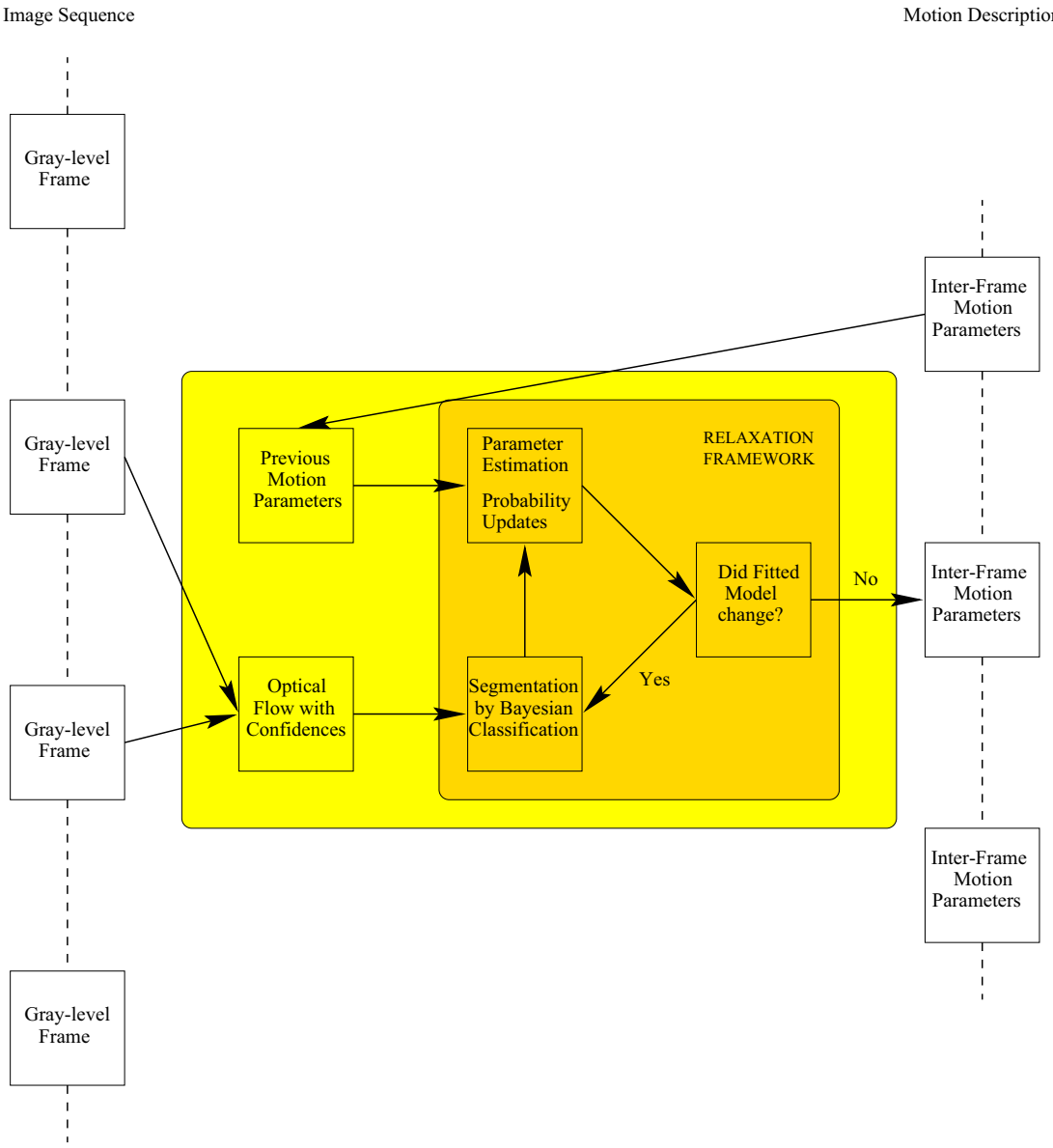


Figure 1.1: Overview of our proposed motion analysis system.

5 demonstrates applications of the output of our algorithm to obtain trajectories (section 5.1), perform image stabilization and object tracking (section 5.2). Two illustrative examples are presented with the stages of our algorithm and are explained at the end of chapters 2, 3 and 5. Chapter 6 discusses the results of our algorithm on multiple longer example sequences. In chapter 7, conclusions and proposals for future work are given. Appendix A gives some details on the software package implemented for this thesis.

Chapter 2

COMPUTING OPTICAL FLOW VECTORS AND THEIR CONFIDENCES

2.1 Similarities and Differences to Other Methods

The foundation of optical flow computation lies in the assumption that image intensity is conserved. In other words, a gray-level pattern in the image at time t will also be in the image at time $t+\delta t$, but eventually at a different location. This location may be displaced from the original location by u in x-direction and v in y-direction. Equation 2.1 formulates this mathematically, as proposed first by Horn and Schunk in [38].

$$I(x, y, t) = I(x + u_{x,y,t}, y + v_{x,y,t}, t + \delta t) \tag{2.1}$$

For an adequately small inter-frame time δt, this assumption is true and for larger δt this still is a good working hypothesis. The field of all displacement vectors $(u_{x,y}, v_{x,y})^T$ for each pixel location $(x, y)^T$ is called optical flow. Optical flow computation is a highly non-trivial and well-known problem in the computer vision literature, and many approaches have been documented. A comparison of optical flow techniques can be found in [39].

We use our own flow algorithm because most existing algorithms do not provide local confidence estimates for their output. Our algorithm is a modified version of Singh's approach [40] and employs a template-matching

methodology. A confidence estimate for each pixel's velocity vector is propagated through the algorithm to identify reliable regions where problems like the aperture problem [1] [40] or motion boundaries are not significant.

2.2 Approach

Our approach to computing the optical flow is based on the intensity conservation equation. Equation 2.2 is a generalized version of Horn and Schunk's original constraint [38] that takes into account a linear transformation (parameters a and m) and zero–mean signal independent additive Gaussian noise n.

$$I(x, y, t) = a \cdot I(x + u_{x,y,t}, y + v_{x,y,t}, t + \delta t) + m + n \tag{2.2}$$

The desired velocity vector $(u_{x,y,t}, v_{x,y,t})^T$ (the index t will be omitted from now on for easier reading) is computed for each pixel in the image by comparing a small neighborhood (template) in the first image at location $(x, y)^T$ with the neighborhood in the second image at a displaced location $(x + u, y + v)^T$. This comparison, which will be explained in section 2.3, returns the likelihood $c_{x,y}(u, v)$, a measure of how well the sub-images match for a certain displacement $(u, v)^T$. We obtain measures for all possible displacements within a searchwindow by performing this template-matching for all possible displacements. We normalize all these correlation measures to fulfill the constraints for a probability mass function. The normalized correlation measures as a function of 2–D displacement is called correlation surface $c_{x,y}$. The displacement with maximum correlation is our maximum likelihood estimate for the optical flow vector at this location (section 2.4). Following an

approach from Singh [40], we fit a continuous bivariate Gaussian probability density function (pdf) to the correlation surface $c_{x,y}$ by computing moments as described in section 2.5. This fitted Gaussian pdf's standard deviation $\sigma_{x,y}$ is indirectly proportional to the confidence in this pixel's displacement estimate.

Most other techniques designed to only compute optical flow smooth the optical flow field finally by assuming a global smoothness constraint [41] [42] [43]. However we do not filter the raw flow field and we pass it unchanged together with the confidence values to the second stage of our system (chapter 3). Smoothing at this point may yield a lower global error for the price of less accuracy at high confidence locations. We decided to proceed this way because only the accuracy of the high confidence locations is relevant for our algorithm's second stage.

2.3 Dissimilarity and Correlation Measures

In order to find out the position to which a gray-level pattern moved, a measure to compare two different patterns of equal sizes is necessary. Wu proposes in [43] a dissimilarity measure $\rho(f,g)$ for two sub-images f and g of the same size $(x_{\max})\mathrm{x}(y_{\max})$. The two sub-images we will compare will be the template and a candidate image. The size of the template in $x-$ and $y-$ directions is $(x_{\max})\mathrm{x}(y_{\max})$. The correlation measure $\rho(f,g)$ is basically the correlation of the image gray-levels over a window, and is computed as follows:

$$\rho(f,g) = \frac{\sum_{x=1}^{x_{\max}} \sum_{y=1}^{y_{\max}} \left(f(x,y) - \overline{f}\right) \cdot \left(g(x,y) - \overline{g}\right)}{\sqrt{\left(\sum_{x=1}^{x_{\max}} \sum_{y=1}^{y_{\max}} \left(f(x,y) - \overline{f}\right)^2\right) \left(\sum_{x=1}^{x_{\max}} \sum_{y=1}^{y_{\max}} \left(g(x,y) - \overline{g}\right)^2\right)}} \quad (2.3)$$

where

$$\overline{f} = \frac{1}{x_{\max} y_{\max}} \sum_{x=1}^{x_{\max}} \sum_{y=1}^{y_{\max}} f(x,y) \tag{2.4}$$

and

$$\overline{g} = \frac{1}{x_{\max} y_{\max}} \sum_{x=1}^{x_{\max}} \sum_{y=1}^{y_{\max}} g(x,y) \tag{2.5}$$

This dissimilarity measure is based on equation 2.2, and is the basis for the correlation measure discussed later in this section. In the beginning of the research another dissimilarity measure, known as the sum of squared differences (SSD), was taken into consideration. This very simple measure is defined in equation 2.6. Even though it is computationally less expensive, it was dismissed because it was not robust in realistic and noisy scenes.

$$\mathrm{SSD}(f,g) = \sum_{x=1}^{x_{\max}} \sum_{y=1}^{y_{max}} (f(x,y) - g(x,y))^2 \tag{2.6}$$

Equation 2.7 shows how the correlation measure $c_{x,y}(u,v)$ is now computed from the dissimilarity measure $\rho_(f,g)$ of the two sub-images f and g. The two sub-images compared are f from the original location in the first frame and g from a potential displaced location in the second frame.

$$c_{x,y}(u,v) = \exp\left(-k\rho\left(s_w(x,y,t), s_w(x+u, y+v, t+\delta t)\right)\right) \tag{2.7}$$

Here $s_w(x,y,t)$ denotes a sub-image from the original image sequence $I(x,y,t)$ at time t centered at pixel (x,y). The horizontal and vertical dimensions of this sub-image are w. By proper normalization through k, we assure that

$$\sum_u \sum_v c_{x,y}(u,v) = 1 \tag{2.8}$$

and, obviously, equation 2.7 implies

$$c_{x,y}(u,v) \geq 0. \tag{2.9}$$

We compute this likelihood measure $c_{x,y}(u,v)$ for all possible displacements $(u,v)^T$ of pixel $(x,y)^T$ within a searchwindow. The size of the searchwindow is set by the user. At this point we now have computed the correlation surface $c_{x,y}$ inside the searchwindow for pixel $(x,y)^T$. All values outside the window are assumed to be zero, meaning that displacements greater than the searchwindow do not occur (or, in other words, happen with probability zero).

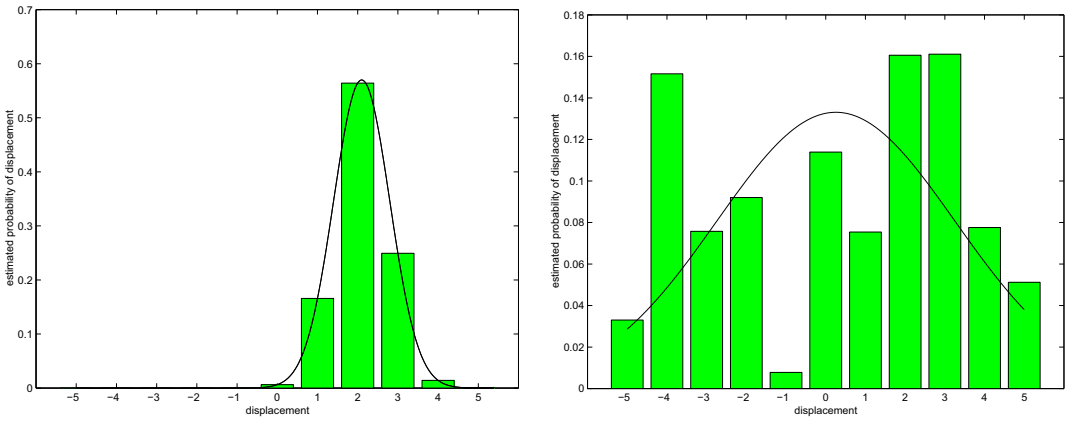
2.4 Maximum Likelihood Estimate

Given the correlation surface and the fact that each of its entries $c_{x,y}(u,v)$ is a likelihood measure for a certain displacement, we simply pick the displacement with maximum likelihood (ML) as our desired velocity estimate $(u_{x,y}, v_{x,y})^T$. This is expressed in mathematical notation by equation 2.10.

$$\begin{pmatrix} u_{x,y} \\ v_{x,y} \end{pmatrix} = \text{argmax}_{u,v}(c_{x,y}(u,v)) \tag{2.10}$$

This scheme limits the precision of our optical flow algorithm to pixel accuracy. Figure 2.1 illustrates this selection for a 1–D case. The x-axis lists all possible displacements in the searchwindow, and the height of the bar illustrates the likelihood of this displacement. Hence, the velocity estimate picked would be 2 in the high confidence case and 3 in the low confidence case.

If there are multiple displacements yielding the global maximum of the correlation surface, we assign a displacement estimate of 0 and mark it



high confidence low confidence

Figure 2.1: Illustration of ML estimation and confidence computation in 1–D.

with a high uncertainty. This case can be neglected for most real world applications. However for simple synthetic test sequences this special case must be considered.

2.5 Confidence from Gaussian Fitting

After obtaining the estimate for the local pixel velocity vector as described in the last section, we now discuss how we compute the associated confidence value for this pixel. Every entry $c_{x,y}(u,v)$ of the correlation surface denotes the probability of a certain discrete inter-frame displacement $(u,v)^T$ at pixel location $(x,y)^T$. We interpret the correlation surface $c_{x,y}$ as a discrete approximation of the continuous parametric 2–D Gaussian pdf

$$p_{x,y}(u,v) = \frac{\sqrt{|C_{x,y}^{-1}|}}{2\pi} \exp\left(-\frac{1}{2}((u,v)^T - \mu_{x,y})^T C_{x,y}^{-1}((u,v)^T - \mu_{x,y})\right). \quad (2.11)$$

While computing the correlation surface $c_{x,y}$ we made sure that it is a 2–D probability mass function by introducing constraints 2.8 and 2.9. So,

now the pdf's mean vector $\mu_{x,y}$ is estimated with least squared error from $c_{x,y}$ as the momentum

$$\mu_{x,y} = \sum_u \sum_v c_{x,y}(u,v) \begin{pmatrix} u \\ v \end{pmatrix} \tag{2.12}$$

and the 2x2 covariance matrix $D_{x,y}$ is estimated as

$$D_{x,y} = \begin{pmatrix} \sigma_{x,y,1} & \sigma_{x,y,3} \\ \sigma_{x,y,3} & \sigma_{x,y,2} \end{pmatrix} = \sum_u \sum_v c_{x,y}(u,v) \cdot ((u,v)^T - \mu_{x,y})((u,v)^T - \mu_{x,y})^T. \tag{2.13}$$

However, the assumption that the u– and v–directions are stochastically independent and have equal variances is sufficient for our applications. Hence we can use a simpler representation of the covariance matrix with only one parameter $\sigma_{x,y}$. This covariance matrix $D_{x,y}$ can now be written as

$$C_{x,y} = \begin{pmatrix} \sigma_{x,y} & 0 \\ 0 & \sigma_{x,y} \end{pmatrix} \tag{2.14}$$

where

$$\sigma_{x,y}^2 = \sum_u \sum_v c_{x,y}(u,v) \left\| \begin{pmatrix} u \\ v \end{pmatrix} - \mu_{x,y} \right\|^2. \tag{2.15}$$

The confidence values are the eigen-values of the covariance matrix [40]. In our case of a diagonal covariance matrix, the two confidence values (one for each dimension) are equal and are indirectly proportional to the standard deviation. Now we have an estimate for the displacement vector $(u_{x,y}, v_{x,y})^T$ (from equation 2.10), and associated with that a confidence value $1/\sigma_{x,y}$ (from equations 2.12, 2.15).

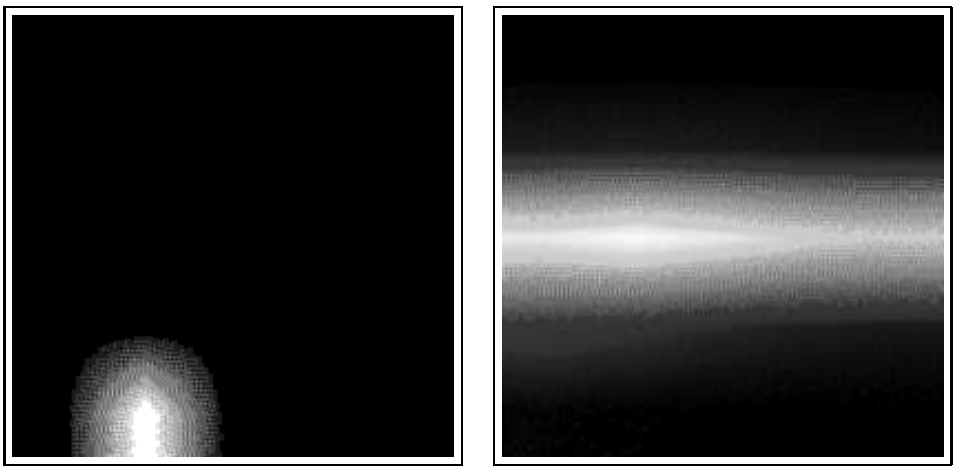
Figure 2.1 shows the entries of the correlation surface as bars and the fitted Gaussian pdf as an overlaid curve in a 1–D setting. At high confidence locations, the bars have one maximum and have a small variability. In

the low confidence environment, the function is multi-modal and broad. High variability characterizes this behavior.

Figure 2.2 shows two exemplary correlation surfaces for the actual 2–D setting. The brightness of a pixel is proportional to the likelihood of its displacement from the image center. Hence a single white spot in the center of the image would illustrate the case where a displacement of $(0, 0)$ is very likely and all other displacements are very unlikely.

A high confidence in motion displacement is seen at image locations which have sufficiently distinct features (e.g., corners) and thereby induce a unimodal and sharply peaked correlation surface $c_{x,y}$ such as seen in figure 2.2 (A). This surface would be interpreted as a high-confidence location with estimated displacement towards the lower left of the searchwindow.

Low confidence occurs at locations with ambivalent or missing motion cues, such as in regions with aliasing or regions without texture. A low confidence correlation surface is shown in figure 2.2 (B). This correlation surface corresponds to a location with a horizontal line segment. Due to the aperture problem, this induced a high uncertainty in the x-direction. If we differentiate confidence in the x- and y-directions, we would say that we are very confident that there is no displacement in y-direction and we have low confidence in our estimate that the displacement in x-direction is slightly to the left of the searchwindow.



(A) (B)

Figure 2.2: Illustration of 2–D correlation surfaces with high (A) and low (B) confidence.

2.6 Resolution Hierarchy – Coarse to Fine

In order to speed up the processing and to be able to deal with large motions, a resolution hierarchy is used. Please refer to [44], [45], [46], [47] or [48] for more detailed information on the advantages and techniques of resolution hierarchies for motion computation. We use a Gaussian image pyramid and our processing scheme is illustrated in figure 2.3. The base of the pyramid is the image in its original resolution (figure 2.3, $h = 1$). Image dimensions are reduced to 50% in each direction from one image to the next until the image is smaller than a user-set threshold. Optical flow computation starts at this low-frequency image (figure 2.3, $h = 3$). The velocity estimates obtained at the lowest resolution is used as the initial guess for the next higher resolution level (figure 2.3, $h = 2$). After refining the estimate at the higher resolution, the results are again propagated to the next level. This process continues until the original resolution level is reached. In other words, the major motion is

recovered in the early stages and refined in the later stages. There is neuro-psychological evidence that the human visual system uses a similar system.

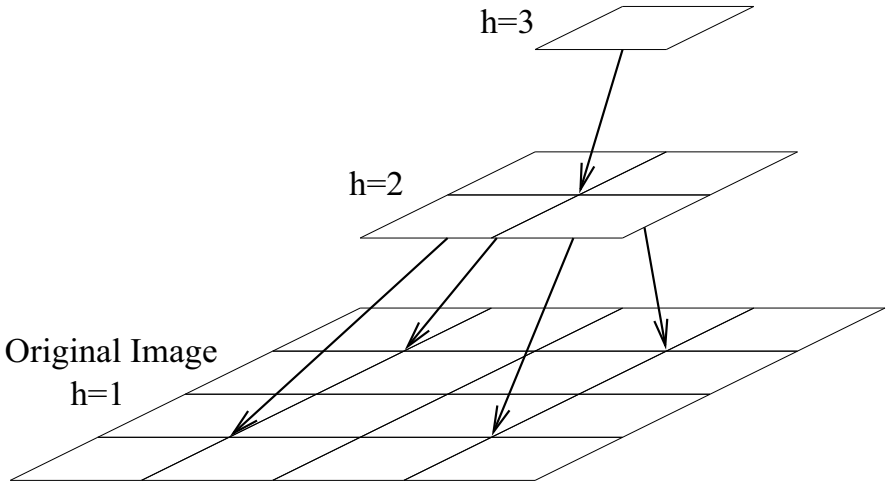


Figure 2.3: Illustration of hierarchical optical flow computation.

Confidence information is obtained on all stages of the hierarchy but is not passed on to higher resolution stages. Thus the final confidence values only rely on computations on the full image resolution.

By choosing this hierarchical structure, we inferred a certain amount of smoothness of the displacement vectors over the image. However this hierarchical procedure enables the algorithm to deal with large inter-frame motions. The algorithm's ability to cope with large motions is best in the center of the image. At the image margin only small motions can be correctly detected. This is due to the fact that locations closer to the margin have fewer pixels in their neighborhood, and a larger inter-frame motion requires more neighbors for correct estimation.

2.7 Treatment of the Image Margins

The treatment of the borders is an important issue in most image processing tasks. Because we use a double-windowed approach to obtain our flow estimates, we need information derived from the environment of each pixel to compute the estimate. A pixel on the border does not have this environment. Here, we define how the image extends beyond its margins. We considered the following implementations:

- A pixel outside is black
- A pixel outside has random gray-level
- A pixel outside is like its closest neighbor inside
- A sub-image partly or totally outside is equal to the Euclidean closest sub-image entirely inside

This list is ordered in ascending experimental performance results for our algorithm. However, it is also possible to compute flow estimates for only these pixels with sufficient environment. This implies cropping the image and, hence, reduced output frame sizes. The preferred method depends in whether "full-size" or "minimum error" output has higher priority.

2.8 Illustrative Examples

The entire system described in this thesis was implemented in C++. The end of chapters 2, 3 and 5 show practical examples of the algorithm applied to example sequences. The two examples consist of two frames each, and were

chosen to illustrate our algorithm and support its explanation in the text. Results on longer, more complex sequences are presented later in chapter 6.

The synthetic Blood Cells Sequence consists of two images of blood cells moving at resolution 300x300 pixels. All cells move horizontally 5 pixels to the right. The ones in the left half of the image do not move vertically and the ones in the right half move 3 pixels upwards. The template size is 9x9 pixels and the displacement searchwindow is 5x5 pixels on each of the 4 resolution levels. The optical flow field is sub-sampled by the factor 5 for printing purposes. The resolution of the final output is 200x200 pixels.

Figure 2.4 shows the original two-image sequence truncated to the size of the algorithm's output. In figure 2.5 (A) the flow is depicted as a vector field, and in figure 2.5 (B) the confidence values are shown. In the confidence image, the brightness is proportional to the uncertainty about the flow estimates.

It is interesting to note that the motion boundary between the left and the right half of the scene appears as a bright line in the confidence image. This signals that the flow computation is rather unreliable in this area. This makes sense, because optical flow is not defined at motion boundaries. Many scenes include motion boundaries caused by depth discontinuities, occlusion or independently moving objects. A motion boundary can be viewed as an edge in the optical flow. Many efforts in flow computation are directed towards the preservation of edges like this by robust filtering [21]. Because in our case we are not interested in the flow itself, but its use for motion estimation, we do not try to obtain good flow vectors everywhere. For our later processing we only need a sparse optical flow field. Hence, it is more important to have accurate

vectors at some locations than to have good vectors at every location.

Another region of high uncertainty is the bone-shaped bright blob in the lower center region of the image. When looking at the corresponding region in the original image, you will see that this region is homogeneously white. Optical flow is a local measurement, and there are no gray-level features in this neighborhood. Neither corner nor edge elements are present. More generally speaking, this area has no texture and no features to provide motion cues. Hence, it is impossible to estimate this location's velocity vectors. Consequently, this region appears bright in the confidence image, denoting low confidence.

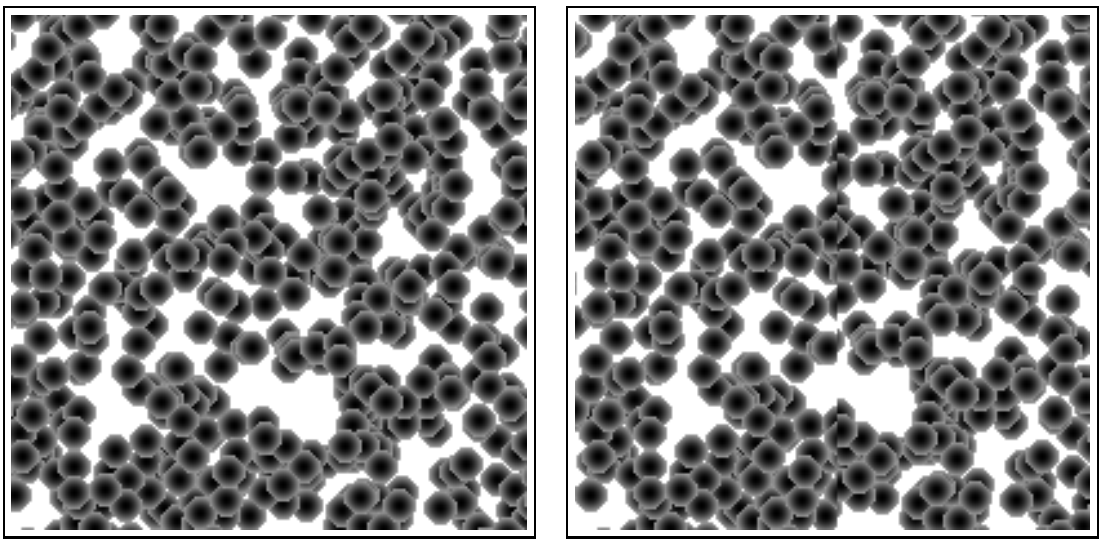
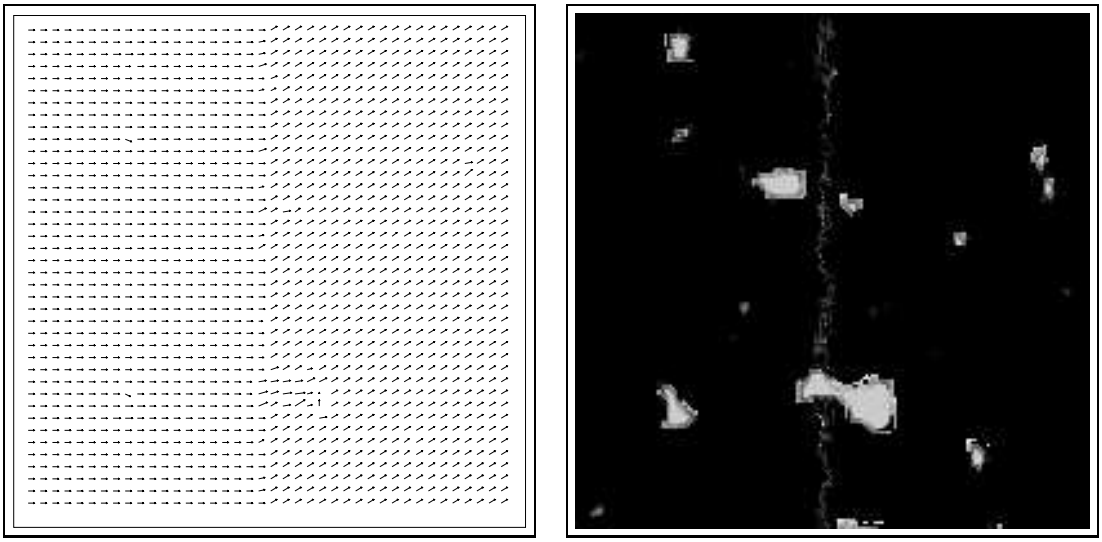


Figure 2.4: The two images of the synthetic Blood Cells Sequence.

The second example depicts a model vehicle driving to the right on a cluttered table viewed from a downward-moving camera. The original image resolution in this case is 320x200 pixels, and is reduced in 2 resolution levels



(A) (B)

Figure 2.5: Computed pixel-wise optical flow (A) and confidence values (B) for the synthetic Blood Cells Sequence. In the confidence image, black = high confidence, white = low confidence.

by the algorithm to an output of size 256x176 pixels. The template size is 11x11 pixels, and the searchwindow size is 7x7 pixels. The original two-image sequence is shown in figure 2.6. Figure 2.7 (A) shows the flow vector field sub-sampled by 5 and figure 2.7 (C) and (D) show the x- and y-component of each vector from the field as a gray-level image. In figure 2.7 (C) the vehicle is distinguished from the background because only the vehicle has horizontal flow components. In figure 2.7 (D) the gradual increase in the absolute y-component of the flow towards the lower image border nicely illustrates the recovered effects of parallax. Figure 2.7 (B) shows the confidences in the optical flow field for corresponding locations.

Figure 2.6: The two images of the Vehicle on Table Sequence.

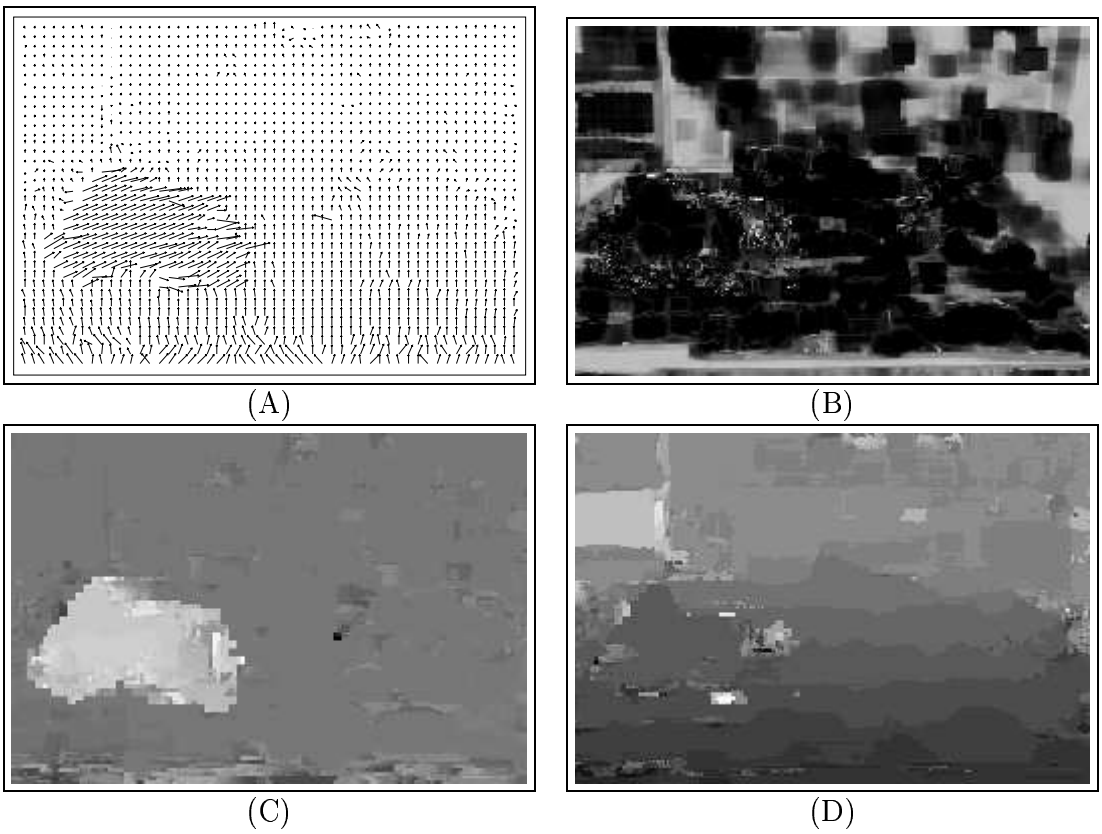


Figure 2.7: Computed pixel-wise optical flow (A) and confidence values (B) for the Vehicle on Table Sequence. In the confidence image, black = high confidence, white = low confidence. The x-component (C) and y-component (D) of the optical flow.

Chapter 3

AFFINE PARAMETER ESTIMATION AND BAYESIAN MOTION CLASSIFICATION

3.1 Statistical Framework

In chapter 2 we obtained the optical flow field. However, our desired result is not a large vector field but a highly compact description of the motion in the scene. This description can now be obtained in the form of parameters of motion models. Various motion models have been proposed in the computer vision literature. Three popular models in increasing order of complexity are:

- Translational Model
- Affine Model
- Planar Model

The translational model assumes a translation of the pixels in the image plane. It has two parameters, namely the x- and y- offset. The translational model can be viewed as a special case of the affine model. This model constrains image velocities to be an affine transform of the 2-D image coordinates. Hence it has six parameters, four in form of a 2–D transformation matrix (θ_1, θ_2, θ_4 and θ_5) and two for the 2–D offset (θ_3, θ_6). It is formally

defined by equation 3.1 and is a special case of the planar model [27].

$$\begin{pmatrix} u_r \\ v_r \end{pmatrix} = \begin{pmatrix} \theta_{i,1} & \theta_{i,2} \\ \theta_{i,4} & \theta_{i,5} \end{pmatrix} \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \begin{pmatrix} \theta_{i,3} \\ \theta_{i,6} \end{pmatrix} \tag{3.1}$$

The planar model is capable of describing the motion of a plane under perspective projection. Besides these models, various 3–D approaches exist. These 3–D models are successfully used today for scenes with only one motion, the camera motion. The displacement pattern can be very complex in the image. It depends on motion of camera and objects, as well as on the entire scene structure. When trying to model these in *full* complexity in the presence of *multiple* motions, the models needed have too many degrees of freedom and are currently yielding very little robustness and low accuracy, so simplifying models have to be employed.

Our proposed algorithm provides a framework for motion estimation in which the motion model can be easily exchanged. For our practical implementation, we use an affine motion model. The affine motion model is well researched and can model scenes with some parallax satisfactorily. Affine motion has six parameters and hence we need a minimum of three correspondences (pixel velocities) for the estimation of the affine parameters. However, there are many more vectors available (typically over 1000 pixel vectors). Posed as an optimization problem, the question is how to fit the affine model to the pixel velocity data and how to decide which pixel contributes to which affine motion. In other words, how do we minimize error in the estimation of the parameters (and hence maximize the quality of the model fit) and how do we segment the image? In this chapter we propose a new Bayesian relaxation framework to answer these questions.

First, we extract a set of points with highly reliable displacement vectors from the optical flow field. We call this set R; it is obtained by selecting the N most reliable locations.[1] High reliability is defined here in terms of high confidence or low variability σ of the pixel velocity. N can be set by the user or can be a function of the image size (e.g. select 10% of all image pixels). A certain data point of this set R will be denoted as r in the rest of the thesis and has a certain known location $(x_r, y_r)^T$ and direction $(u_r, v_r)^T$ (the displacement from chapter 2). The rest of this section gives a brief overview of the new Bayesian relaxation framework (as illustrated in the dark shaded area in figure 1.1).

Assuming that exactly two affine motions are present in the sequence, we partition the set of reliable points R into three disjoint sets R_0, R_1 and R_2, representing rejection, primary, and secondary motion classes respectively. The initial partitioning assigns points with a smaller than average magnitude of displacement to R_1 and all other points to R_2. Hence, the rejection class is always empty at the initialization of the relaxation.

In the next step, we estimate the affine motion parameters $\theta_{i,1...6}$ for each class $i \in \{1, 2\}$. Equation 3.1 is assumed to hold for each point $r \in R_i$ and, hence, with $n_i = |R_i|$, a linear system of equations with six unknown scalars $\theta_{i,1...6}$ and $2n_i$ ($\gg 6$) equations is given. We use the pseudo-inverse and the Gauss–Jordan method to compute the least squared error estimate for $\theta_{i,1...6}$.

[1] In the C++ implementation we decided to approximate the sorting in descending order of confidence and selecting the first N pixel locations by selecting points with confidence above a certain threshold. This threshold is computed through a binary search so that N pixel are selected.

The estimates for $\theta_{i,1...6}$ are now used to reclassify each point $r \in R$. Equation 3.11 is used to compute the model-based value for u_r and v_r for each point and each motion $i \in \{1, 2\}$. We assume that the measured displacement is the model based displacement affected by Gaussian zero–mean noise or an outlier. Thus $p(r|m_i)$, the probability that point r's displacement was caused by motion i, can be represented by a bivariate Gaussian pdf with the model-based displacement for point r under motion i as the mean and the unity matrix as the covariance matrix. Moreover, equal *a priori* probabilities $P(m_i)$ are assumed for each class. Now we can compute the *a posteriori* probability $P(m_i|r)$ using Bayes' rule [37].

Utilizing a Bayes' classifier, we find the new label l_r for point r. If the maximum *a posteriori* probability drops below a certain user–defined threshold t, a safe decision cannot be made and hence we assign this point the rejection class label 0. After determining the new labels l_r for all points $r \in R$, we re–estimate the parameters $\theta_{i,1...6}$ as described above ($l_r = i \Leftrightarrow r \in R_i$). We update our probabilities through the new parameters and classify again. We iterate this alternating estimation and classification process until our parameters do not change from the current estimate to the next. The parameter optimization has converged. Section 3.2 and 3.3 give a more detailed insight into the estimation and classification steps, respectively.

Basically, both motion classes are treated equally. However, we have to decide which class represents foreground and which background. Under the reasonable assumption that the IMO accounts for less than 50 % of the visible area and has as distinct features as the background we can make our

decision simply as follows: The class with more members will be considered the background class and its motion will be assumed to be caused by camera motion. Accordingly, the class with fewer members will be considered the object class undergoing image motion caused by camera and object motion.

This fails, for example, if we want to analyze a sequence of an airplane flying in a homogeneous blue sky. However, it is not possible to correctly recover object and camera motion in this case, because the background does not have enough features. On a featureless background, it is impossible to tell which component of the motion is caused by the camera and which by the object moving.

3.2 Estimation Step (Yielding Probability Updates)

Every location r was assigned to a certain motion class i during initialization. To estimate the parameters for the motion class i, we consider only locations r that are currently labeled as i. Each motion class i has six scalar parameters, $\theta_{i,1}$ through $\theta_{i,6}$, which we assume to fulfill the affine motion constraint (equation 3.1) for all n_i locations, $r \in R_i$, that currently are labeled i. At this point u_r, v_r, x_r and y_r are known quantities for all $n_i = |R_i|$ locations r. So, equation 3.1 holds for each location r, and we obtain the linear equation system 3.2 with six unknown scalars $\theta_{i,1...6}$ and $2n_i$ equations (two for each

location r).

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{pmatrix} \cdot \begin{pmatrix} \theta_{i,1} \\ \theta_{i,2} \\ \theta_{i,3} \\ \theta_{i,4} \\ \theta_{i,5} \\ \theta_{i,6} \end{pmatrix} = \begin{pmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{pmatrix} \tag{3.2}$$

This equation system 3.2 can be separated into two systems 3.3 and 3.4 of three unknown scalars each ($\theta_{i,1\ldots3}$ and $\theta_{i,4\ldots6}$) and n_i equations (one per location) each.

$$\begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \cdot \begin{pmatrix} \theta_{i,1} \\ \theta_{i,2} \\ \theta_{i,3} \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \tag{3.3}$$

$$\begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \cdot \begin{pmatrix} \theta_{i,4} \\ \theta_{i,5} \\ \theta_{i,6} \end{pmatrix} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \tag{3.4}$$

In our case we can assume $n_i \gg 3$ and, therefore, we have to use an approximative method to solve for the motion parameters $\theta_{i,1\ldots6}$. We decided to use the pseudo-inverse approach, which minimizes the sum of the squared error, to reduce the number of equations from n_i in equations 3.3 and 3.4 to 3 in equations 3.5 and 3.6.

$$\begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \cdot \begin{pmatrix} \theta_{i,1} \\ \theta_{i,2} \\ \theta_{i,3} \end{pmatrix} = \begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \tag{3.5}$$

$$\begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \cdot \begin{pmatrix} \theta_{i,4} \\ \theta_{i,5} \\ \theta_{i,6} \end{pmatrix} = \begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \tag{3.6}$$

The closed form solution of the motion parameter estimates is now given by equations 3.7 and 3.8.

$$\begin{pmatrix} \theta_{i,1} \\ \theta_{i,2} \\ \theta_{i,3} \end{pmatrix} = \left(\begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \right)^{-1} \cdot \begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \tag{3.7}$$

$$\begin{pmatrix} \theta_{i,4} \\ \theta_{i,5} \\ \theta_{i,6} \end{pmatrix} = \left(\begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \right)^{-1} \cdot \begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \tag{3.8}$$

Further simplification by explicit notation of the matrix operations leads to equations 3.9 and 3.10.

$$\begin{pmatrix} \theta_{i,1} \\ \theta_{i,2} \\ \theta_{i,3} \end{pmatrix} = \begin{pmatrix} \sum x^2 & \sum xy & \sum x \\ \sum xy & \sum y^2 & \sum y \\ \sum x & \sum y & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \sum xu \\ \sum yu \\ \sum u \end{pmatrix} \tag{3.9}$$

$$\begin{pmatrix} \theta_{i,4} \\ \theta_{i,5} \\ \theta_{i,6} \end{pmatrix} = \begin{pmatrix} \sum x^2 & \sum xy & \sum x \\ \sum xy & \sum y^2 & \sum y \\ \sum x & \sum y & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \sum xv \\ \sum yv \\ \sum v \end{pmatrix} \tag{3.10}$$

The matrix inversion can simply be obtained by the standard Gauss-Jordan method. With this scheme, we are able to obtain a least squared error estimate of the six parameters $\theta_{i,1\ldots6}$ for each motion class i in the image.

3.3 Classification Step (Yielding Segmentation)

The estimated parametric model is now used to classify each pixel in our set of reliable pixels R. We employ a Bayesian classifier to decide for

each reliable pixel whether it belongs to the primary or secondary motion class or is an outlier. Explicitly dealing with outliers as a separate class enhances the robustness of our algorithm. By excluding highly improbable velocities from the estimation process we avoid that these erroneous locations arbitrarily worsen our results. The classification step described in this section leads to a segmentation of the reliable parts of the image.

Using the model with the current set of parameters θ_j, we can now compute the model-based displacement $(\hat{u}_r, \hat{v}_r)^T$ for a position r under the hypothetical assumption that this location was generated by motion j. The model-based displacement is the displacement vector we would expect, assuming our current model and its parameters are correct. We compute the expected displacement (equation 3.11) using the same constraint we used to estimate the affine parameters (equation 3.1). However, we solve for $(\hat{u}_r, \hat{v}_r)^T$ instead of $\theta_{i,1\ldots6}$.

$$\begin{pmatrix} \hat{u}_r \\ \hat{v}_r \end{pmatrix} = f_j \begin{pmatrix} x_r \\ y_r \end{pmatrix} := \begin{pmatrix} \theta_{j,1} & \theta_{j,2} \\ \theta_{j,4} & \theta_{j,5} \end{pmatrix} \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \begin{pmatrix} \theta_{j,3} \\ \theta_{j,6} \end{pmatrix} \tag{3.11}$$

Assuming that the estimated displacement $(u_r, v_r)^T$ (see chapter 2) is the ideal (according to the model) displacement $(\hat{u}_r, \hat{v}_r)^T$ affected by independent zero–mean additive Gaussian noise n with normalized deviation 1 (equation 3.12), the probability $p(r|m_j)$ that the displacement vector r was generated by the j-th motion can be computed (equation 3.13).

$$(u_r, v_r)^T = (\hat{u}_r, \hat{v}_r)^T + \text{Noise} \tag{3.12}$$

$$P(r|m_j) = \mathcal{N}_{\hat{u}_r,1}(u_r) \cdot \mathcal{N}_{\hat{v}_r,1}(v_r) \tag{3.13}$$

$$\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2) \tag{3.14}$$

Assuming equal *a priori* probabilities for both motions ($P(m_1) = P(m_2) = 0.5$), we can now use Bayes' rule to classify each displacement vector r. A location r is assigned to the motion by which it was created with the maximum *a posteriori* probability $P(m_i|r)$ (equation 3.15). How the decision is made for the new label l_r is expressed formally in equation 3.16. In the rejection case ($l_r = 0$), the maximum *a posteriori* probability was too low to make a good decision and the location r is considered undecided or, in other words, an outlier.

$$P(m_j|r) = \frac{p(r|m_j) \cdot P(m_j)}{p(r)} = \frac{p(r|m_j) \cdot P(m_j)}{\sum_j p(r|m_j) \cdot P(m_j)} \tag{3.15}$$

$$l_r = \begin{cases} \text{argmax}_j(P(m_j|r)) & \text{if argmax}_j(P(m_j|r)) > t \\ 0 & \text{else} \end{cases} \tag{3.16}$$

We reclassify all locations r following this rule. At this point the algorithm goes back to estimate the motion parameters again. This iteration is performed as long as at least one classification of a location r changes. If no changes occur, the algorithm terminates and the current motion parameters are considered the final result for the current pair of images.

3.4 Illustrative Examples

Here the results of the motion analysis on the examples introduced in chapter 2 are presented. For the representation of the parameters, the x and y coordinates of a pixel have to be defined. In our case the origin of the axes is located in the upper left corner with pixel (0,0). The positive x-axis goes rightward from there and the positive y-axis downward. The motion parameters $\theta_{1,\ldots,6}$ are computed in reference to this axis system.

Table 3.1 lists the final motion parameters of the synthetic Blood Cells Sequence, which were obtained after one iteration step. The existing two motions are correctly recovered. The matrix parameters θ_1. θ_2, θ_3 and θ_4 are all zero indicating that the motions present are only translational. The primary motion and secondary motion[2] are described as a translation of $(5, -3)^T$ and $(5, 0)^T$ in the image plane, which are the exact parameters used to generate the sequence.

Motion Class	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	Error	Size
Primary Motion m_1	0.0	0.0	5.0	0.0	0.0	-3.0	0.0	59.1%
Secondary Motion m_2	0.0	0.0	5.0	0.0	0.0	0.0	0.0	40.9%
Rejected								0.0%

Table 3.1: Final motion parameters for the synthetic Blood Cells Sequence.

Figure 3.1 (A) depicts the final segmentation. In the final classification image, members of the primary motion are gray, members of the secondary motion are white and the points not considered are black.

[2]Primary motion, motion class 1 and background motion are used equivalently. Accordingly secondary motion, motion class 2, foreground motion and object motion are also synonymous.

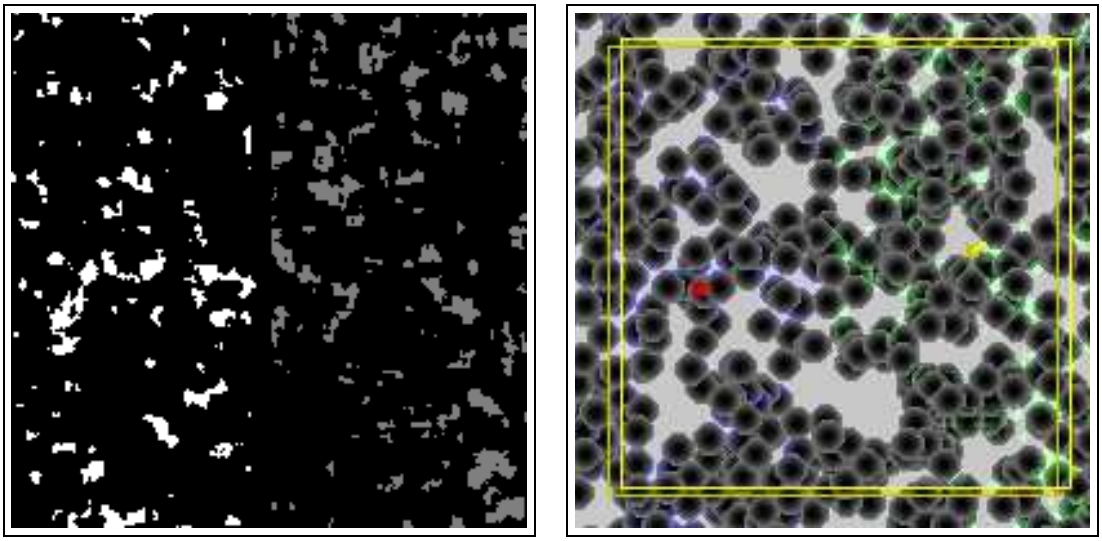
The original scene with the overlaid segmentation is shown in figure 3.1 (B). Here members of the primary motion class are colored green and members of the second motion class are colored blue. Pixels not considered (because they are unreliable or rejected by the classifier) remain uncolored. Primary motion corresponds to background motion and secondary motion to object motion.

The yellow square represents the detected centroid of the primary motion. Attached to this square is a yellow line that points where this centroid moves to in the image plane according to our motion analysis. Accordingly the red square is the centroid of the secondary motion and the red line originating at the red square shows direction and magnitude of the secondary motion centroid as estimated by our algorithm.

Moreover there are two yellow boxes in the image. The darker box illustrates the field of view of the camera in the first frame. The brighter box shows where this field of view has moved to in the second frame according to the results of our motion algorithm. In this case it illustrates that the background frame of reference moves to the upper right between the two frames. These two boxes depict the background or camera motion to the upper right or lower left respectively.

The final Bayesian classifier was trained only on the reliable locations. Figure 3.2 shows the classification results if the final Bayesian classifier is applied to the entire image, not only to the reliable locations.

The results on the Vehicle on Table Sequence are discussed in more detail. Our algorithm converges in four iteration steps. Figure 3.3 and 3.5



(A) (B)

Figure 3.1: Final classification (A) and motion analysis illustration (B) of the synthetic Blood Cells Sequence.

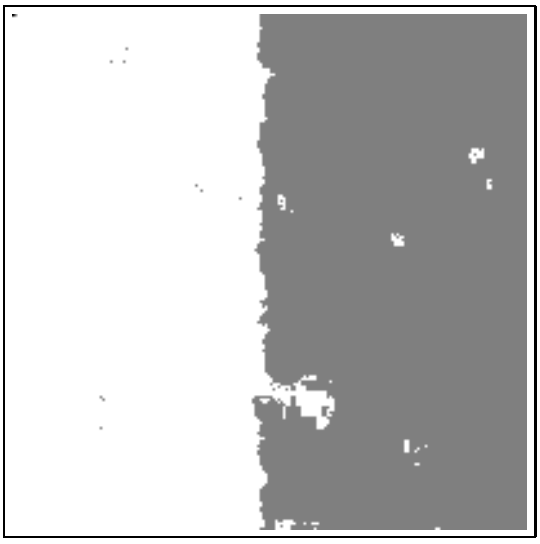


Figure 3.2: Full classification of the synthetic Blood Cells Sequence.

show graphically how the parameters of primary and secondary motion change over the iterations. The class specific average error per velocity vector and the proportional class size for primary and secondary motion are graphed in figures 3.4 and 3.6, respectively. The class specific average error is computed as the average Euclidean distance between the velocity vector from the flow field and the velocity vector modeled through our algorithm.

Table 3.2 lists the total mean error of fit in pixels and the size of the rejection class over all four iteration steps. The total mean error is continuously decreasing from the initial iteration 1 to the final iteration 4.

Figure 3.7 show the corresponding classifications for the first iterations 1 through the final iteration 4 (figure 3.7 (A) through (D)). The forced full classification of the scene is shown in figure 3.8.

An illustration of the motion analysis results, using colors as described earlier for the synthetic Blood Cells Sequence, is shown in figure 3.9. Please note that the two yellow boxes show how our algorithm nicely models the 3–D parallax in the scene with an affine transformation.

Iteration	Total Error in Pixels	Rejected
1	0.667770	0.000000 %
2	0.591061	2.252956 %
3	0.427410	2.922150 %
4	0.369866	0.000000 %

Table 3.2: Total mean error and rejected proportion for Vehicle on Table Sequence.

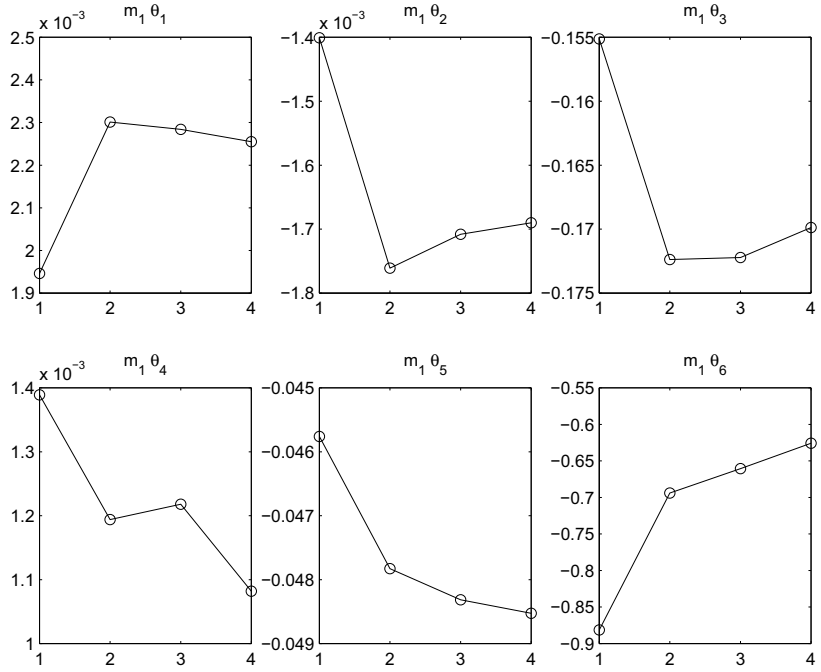


Figure 3.3: Class 1 motion parameters for the Vehicle on Table Sequence.

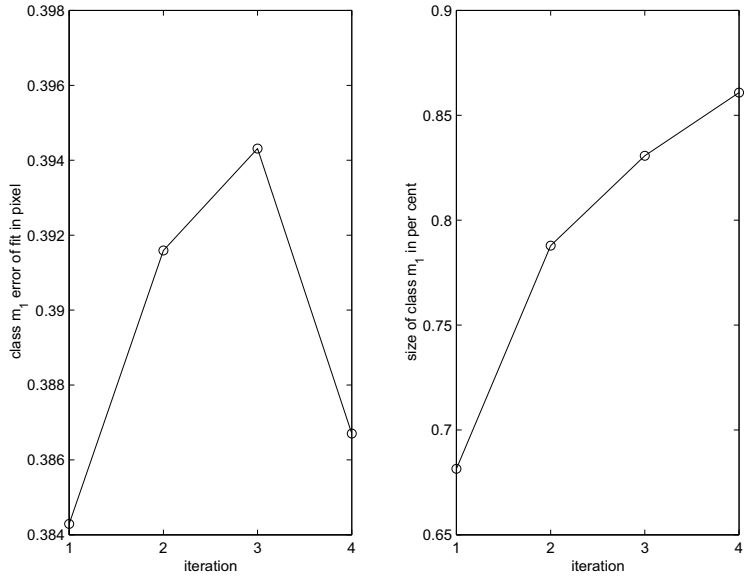


Figure 3.4: Error of fit and size of class 1 for the Vehicle on Table Sequence.

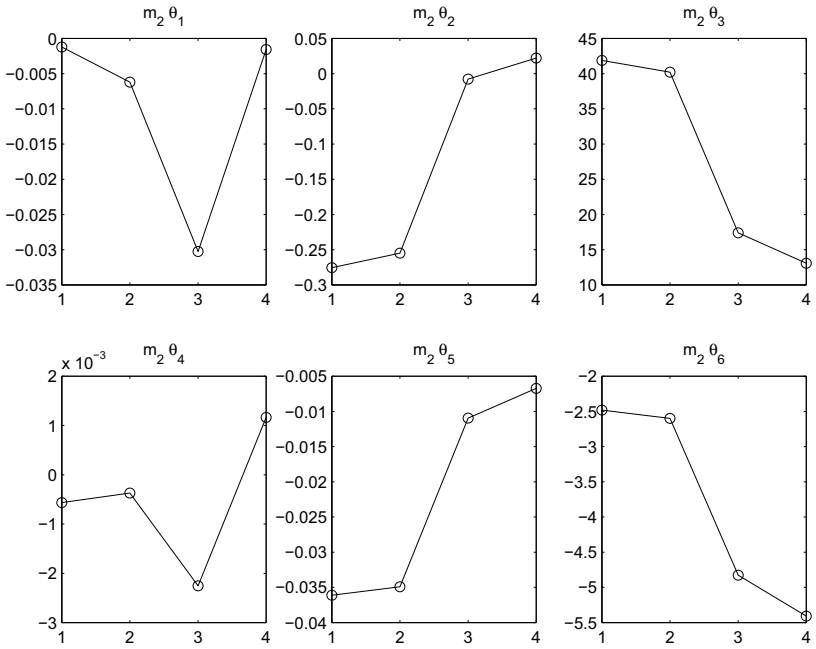


Figure 3.5: Class 2 motion parameters for the Vehicle on Table Sequence.

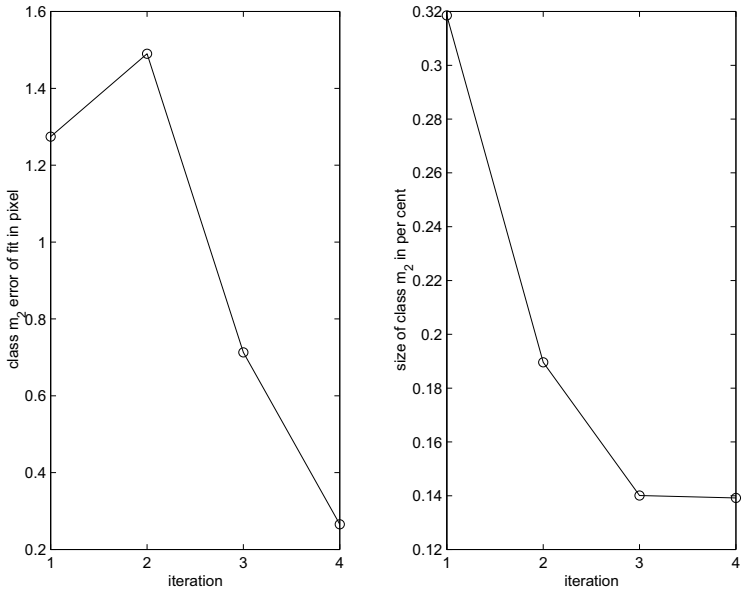


Figure 3.6: Error of fit and size of class 2 for the Vehicle on Table Sequence.

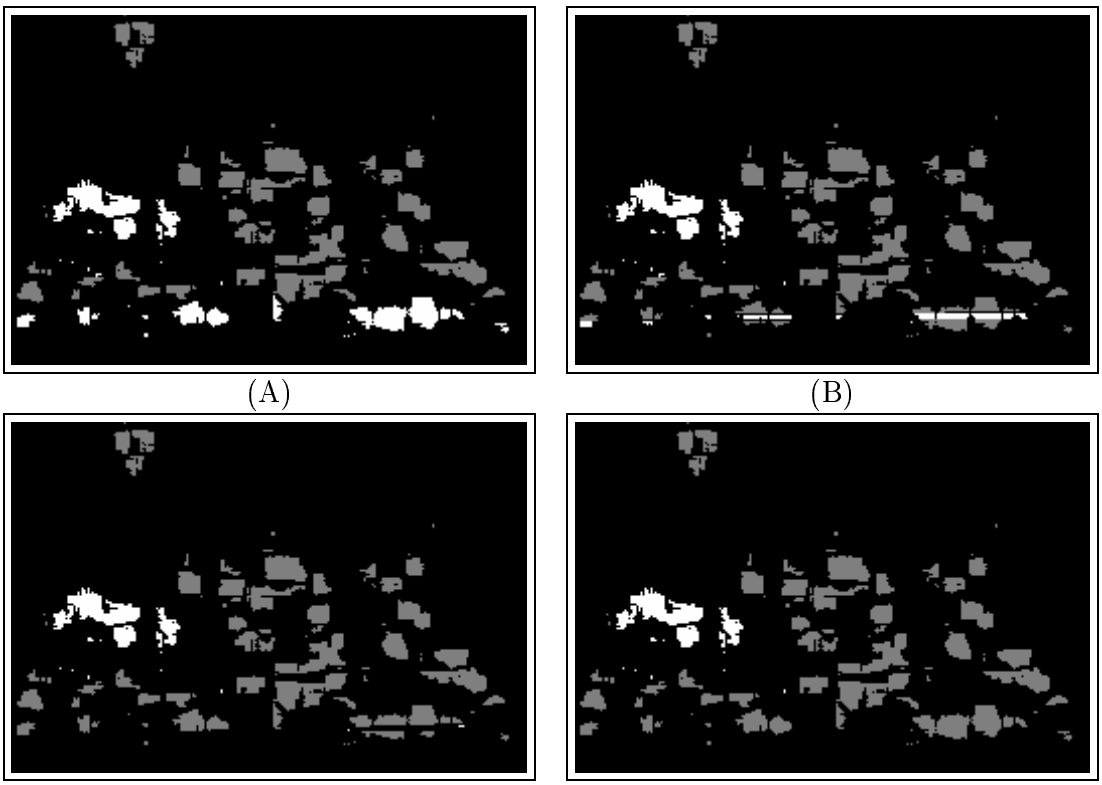


Figure 3.7: The subsequent classifications in the four iteration steps of the Vehicle on Table Sequence.

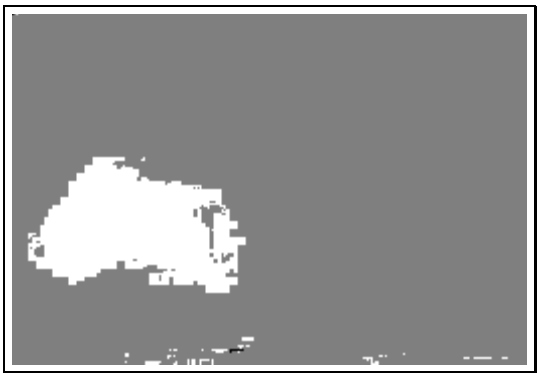


Figure 3.8: Full classification of the Vehicle on Table Sequence.

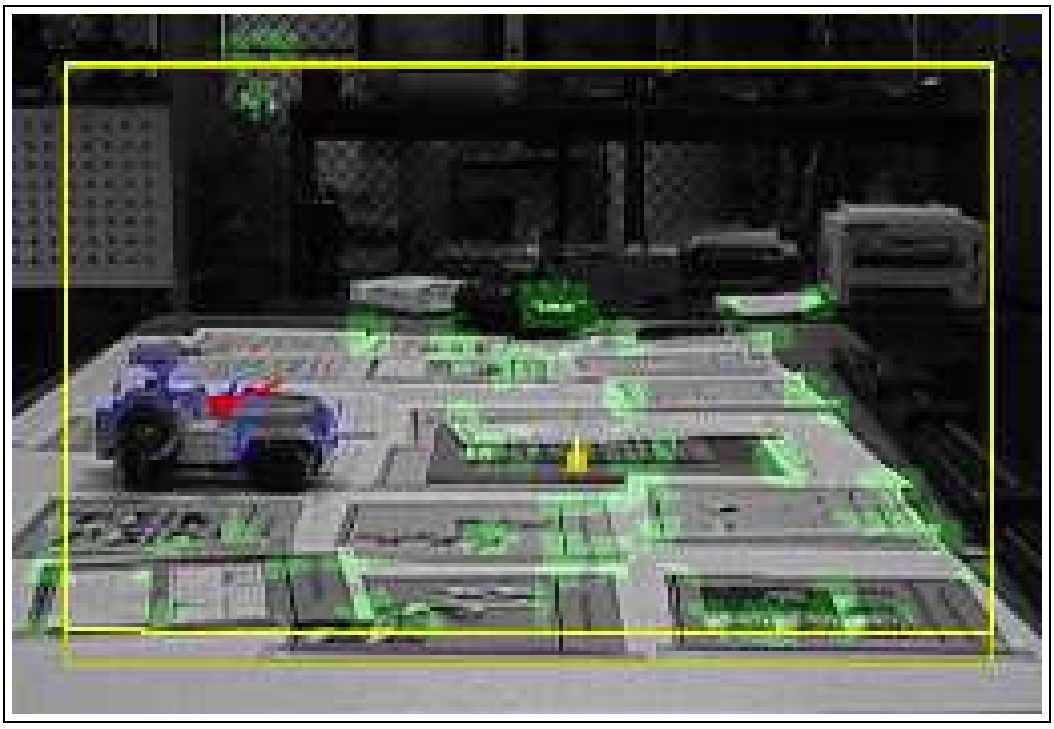


Figure 3.9: Motion analysis illustration of the Vehicle on Table Sequence.

Chapter 4

PROCESSING OF LONGER IMAGE SEQUENCES

4.1 Integrating Information from Multiple Frames

Our system, as described so far, has only analyzed two successive frames. Of course, in most real-world applications, longer sequences have to be analyzed. This chapter presents our approach to use our new framework in this setting. The relaxation framework described in chapter 3 is a local optimization algorithm and its success, like all of its kind, strongly depends on its initialization [49]. So when we process a longer image sequence we initialize the current motion parameters with the results from the previous pair of images before we start the relaxation. This solves the initialization problem and additionally provides a speed-up in processing. Implicitly however it assumes a certain smoothness and consistency of motions over time. This is very reasonable because neither objects or camera will rapidly change their motion parameters due to inertia given an appropriately high frame rate.

4.2 Class Consistency

Another difficulty arises when processing longer sequences: We have to make sure that the two classes extracted from each pair of frames are labeled consistently. In other words the background class in one frame must be the background class in the next frame. Background and foreground classes should

not be swapped from one frame to the next. This correspondence can be established with one of the following three strategies

- The class with more members is always the background.
- A class gets the same label as the class with the most similar parameters in the last frame.
- A class gets the same label as the class in the last frame whose centroid is closest to the current class' centroid.

The first strategy should be used if the background always has more features than the background. The second strategy is useful if the first is not given but the present motions are smooth and do not change abruptly over time. The last strategy is employed when the other two cannot be used. The best choice has to be made dependent on the scene (object size, distance and inertia) and the used imaging technique (optical, infra-red, range images [50], camera mounting).

4.3 Empty Classes

Eventually it may happen that one class has no locations assigned during the relaxation process. In other words one of the two classes becomes empty in the process of classifying and updating probabilities. This is a strong indicator that the processed sequence contains only one motion. One of the two classes vanishes if the motion in the scene can be satisfactorily described with only one affine motion. So all points are classified into the same motion class and the other one is empty. This means that we can no longer estimate the

affine parameters for the empty class. We could stop processing at this time and argue that there is only one motion present in the scene, or that the two motions are not distinct enough. However we assumed for our approach that two affine motions are present, so instead we repeat the relaxation procedure, however, we will not carry over the parameters from the last frame. Instead, we reinitialize the two classes by splitting them at the mean of their velocity magnitudes, as we did in the processing of the first two frames. If one class becomes empty again, we accept this by assuming only one affine motion and move on to the next frame. In the next frame, we will assume two motions again, hoping that now there are two sufficiently distinct motions.

Chapter 5

APPLICATIONS

5.1 Obtaining Trajectories

One possible application of the motion description obtained by our algorithm is to obtain the trajectory of an independently moving object observed by a moving camera. To accomplish this, we first must compute the affine motion parameters of both classes and the centroids of the motion segments over the entire sequence. The affine motion parameters are computed as described in chapter 3. Equation 5.1 describes how the centroid $(X_{j,0}, Y_{j,0})^T$ of the j-th motion ($j \in \{1, 2\}$) is computed in frame 0. R_j is the set of all locations in the image assigned to motion class j as used in chapter 3.

$$\begin{pmatrix} X_{j,0} \\ Y_{j,0} \end{pmatrix} = \frac{1}{|R_j|} \sum_{r \in R_j} \begin{pmatrix} x_r \\ y_r \end{pmatrix} \tag{5.1}$$

We are interested in computing the projection of the 3–D trajectory in the scene onto the 2–D image plane. From the affine parameters and the IMO centroid in the first frame, a trajectory of the IMO is obtained by applying the affine transformation with the estimated parameters for the secondary motion class (object motion, $j = 2$) to the object centroid in the first frame.

Compensating for camera motion is conducted by subtracting the effects of the primary motion transformation (camera motion, $j = 1$) from the

object centroid's motion. This is done successively for all frames to obtain a trajectory corrected for camera motion.

Equation 5.2 gives the recursive formula for computing all successive object centroid locations $(X_{2,t}, Y_{2,t})^T$ (with $t \in \{1, 2, \ldots, \text{lastframe}\}$) in the frame of reference of the first frame 0. Let $f_j^{t-1,t}$ denote the coordinates transformation from frame $t-1$ to t under motion $j \in \{1, 2\}$. The function $f_j^{t-1,t}$ corresponds to f_j as defined in equation 3.11 in chapter 3.

$$\begin{pmatrix} X_{2,t} \\ Y_{2,t} \end{pmatrix} = \begin{pmatrix} X_{2,t-1} \\ Y_{2,t-1} \end{pmatrix} + f_2^{t-1,t} \begin{pmatrix} X_{2,t-1} \\ Y_{2,t-1} \end{pmatrix} - f_1^{t-1,t} \begin{pmatrix} X_{2,t-1} \\ Y_{2,t-1} \end{pmatrix} \tag{5.2}$$

This sequence of object centroid locations $(X_{2,t}, Y_{2,t})^T$ is the desired 2–D trajectory.

5.2 Image Stabilization and Object Tracking

Here image stabilization means image registration while taking an independently moving object into account. For these applications, not only inter-frame motion descriptions are desired, but transformations that describe the motion to the current frame from a frame of reference which may be several frames ago. The total transformation $f_j^{0,t}$ over multiple frames can be decomposed as the concatenation of successive inter-frame motions:

$$f_j^{0,t} = f_j^{t-1,t} \circ f_j^{0,t-1} \tag{5.3}$$

The stabilized image regarding motion j can now be obtained by applying the inverse transform $(f_j^{0,t})^{-1}$ to project the current frame t into the coordinates of the reference frame 0. In order to compensate for background

motion or object motion we use $j = 1$ or $j = 2$, respectively.

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = (f_j^{0,t})^{-1} \begin{pmatrix} x_t \\ y_t \end{pmatrix} \tag{5.4}$$

If this inverse transform, as described by equation 5.4, is applied with $j = 1$ to all pixels $(x_t, y_t)^T$ in frame t and the estimated motion parameters (chapter 3) were correct, the background in the projected image will be stationary compared to the frame of reference 0. If this transform is successfully applied to all frames of a sequence only the motion of the object will remain. Of course this assumes not only correctly estimated parameters but also that the motions present can be satisfactorily modeled with the assumed affine motion model.

If subsequent frames are superimposed after they have been backprojected to the same frame of reference, a mosaic of the observed scene is obtained [29] [30]. When new parts of the scene become visible due to the camera motion, these are added to a scene image larger than each single frame. In this stabilized and mosaiced sequence we can now apply dynamic scene analysis tools for the SCMO case, for example difference images.

On the other hand, when we use $j = 2$ the coordinate transform will compensate for the object motion in the scene. Doing so for several frames will yield a sequence with a stationary object and and moving background. This is equivalent to tracking the object. Once detected in the first frame, the backprojection tracks the object by keeping it stationary in the transformed sequence.

Of course the described backprojection for either case, $j = 1$ or $j = 2$,

is only meaningful up to a reasonable maximum number of frames. The magnitude of motion increases as more frames lie between the current frame and the frame of reference. The errors accumulate because of the concatenation architecture, so a reset to a new frame of reference eventually becomes necessary. The actual decision rule on how to reset the frame of reference depends on the actual stabilization or tracking application.

5.3 Illustrative Examples

In the synthetic Blood Cells Sequence the left side of the image is considered the object and the right side the background. In figure 5.1 the object centroid is illustrated by the red square and the red line extending downwards from there expresses that the left side of the image is moving downward in frame of reference of the right side. Recalling that the left half translates $(5, 0)^T$ pixels and the right half translates $(5, -3)^T$ pixels in the image plane, the trajectory of the centroid of the left half in reference to the right half is 3 pixels downward. This fact was correctly recovered and is illustrated in figure 5.1. The sequence transformed such that the right half of the scene (the background) stays stationary is depicted in figure 5.2. Accordingly, in figure 5.3 the left half of the scene does not move. The stabilized and mosaiced sequences are always produced at double height and width. This enlarged size is necessary to properly illustrate registration of displaced images while performing stabilization and mosaicing.

The usefulness of this applications may become more apparent in the Vehicle on Table Sequence. Figure 5.4 show the computed trajectory of the

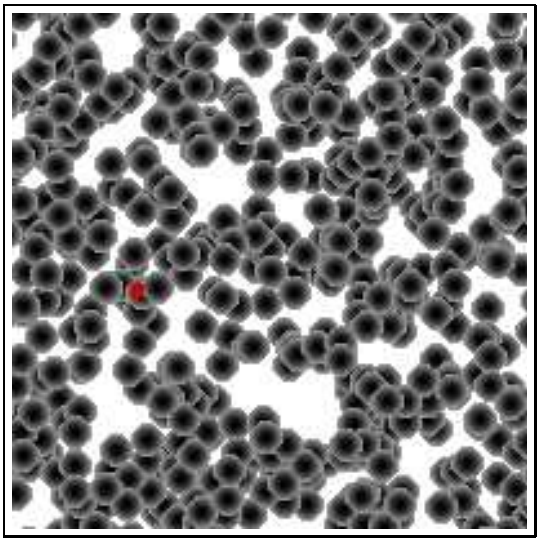


Figure 5.1: Trajectory of the synthetic Blood Cells Sequence.

vehicle on the table. The vehicle's motion is no longer described as motion to the right and upwards as in figure 3.9. We successfully computed the path of the vehicle horizontally to the right as indicated by the red line. This is due to the correction for the camera motion downwards. In figure 5.5 the sequence is depicted compensated for camera motion. Despite the presence of strong parallax the background registration shows no visible errors. The sequence backprojected to obtain a stationary object is depicted in figure 5.6.

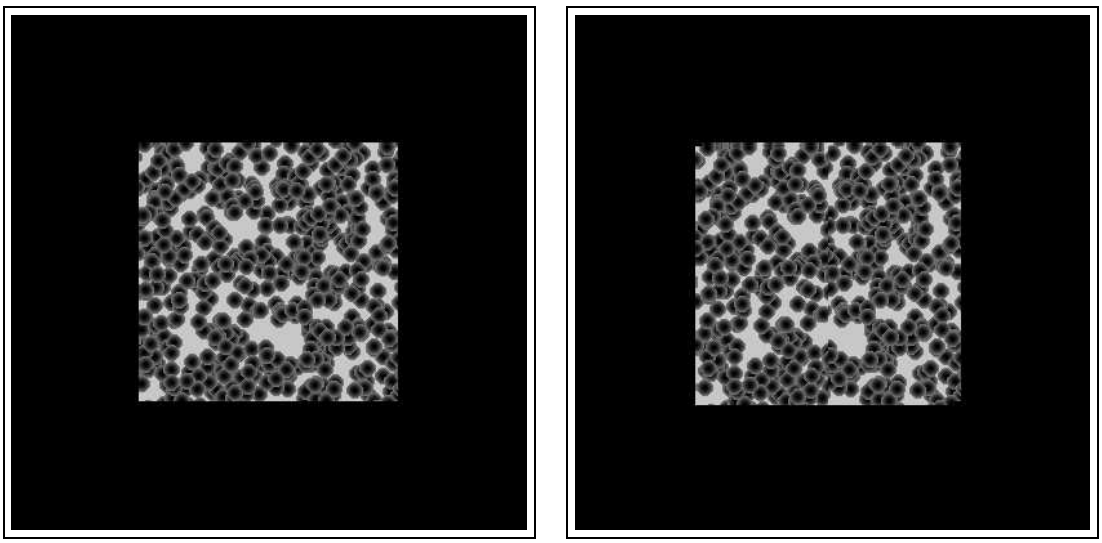


Figure 5.2: Background stabilized synthetic Blood Cells Sequence.

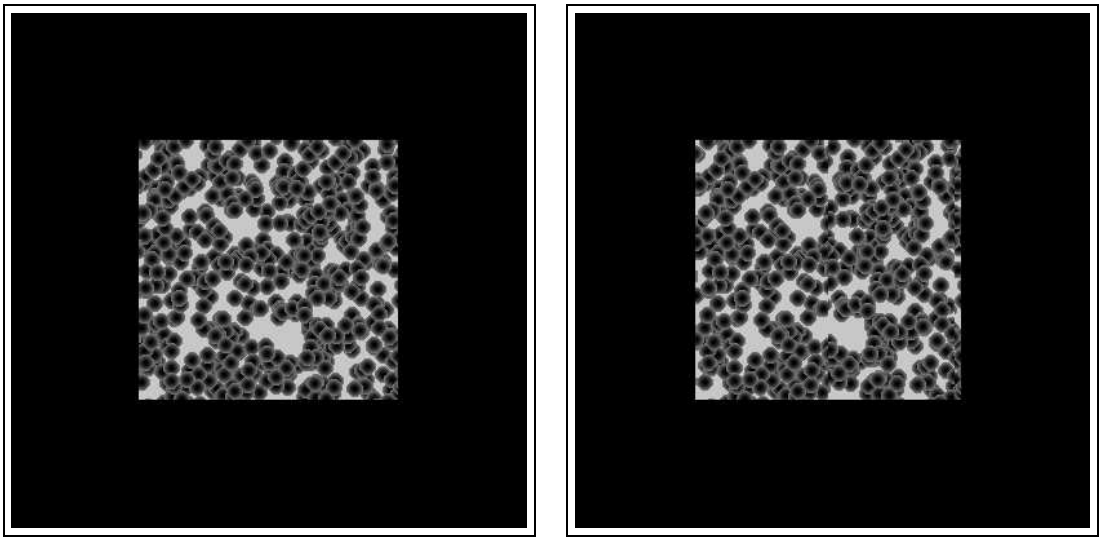


Figure 5.3: Object stabilized synthetic Blood Cells Sequence.

Figure 5.4: Trajectory of the Vehicle on Table Sequence.

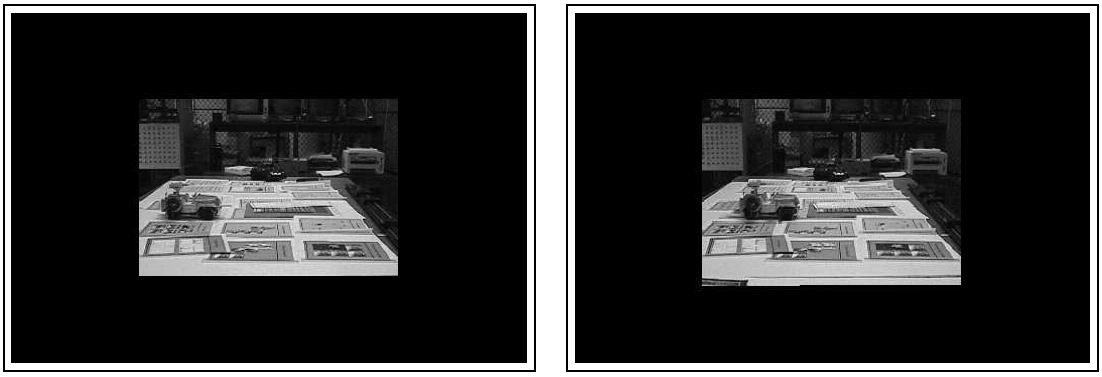


Figure 5.5: Background stabilized Vehicle on Table Sequence.

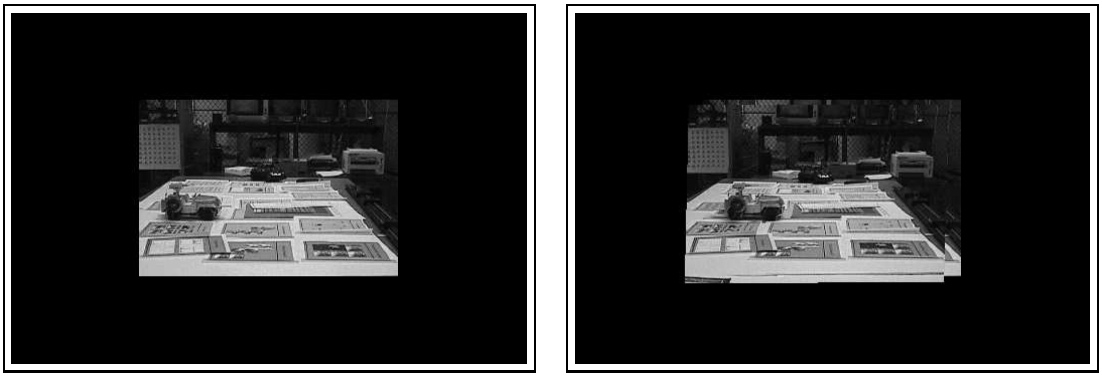


Figure 5.6: Object stabilized Vehicle on Table Sequence.

Chapter 6

DISCUSSION OF EXAMPLES

6.1 Can on Table Sequence

This sequence was taken frame by frame with a digital camera at a resolution of 320x200. It shows a moving can on a table. The camera tracks the can and tries to keep the can in the center of the view. The camera was hand-held and the motion of the camera is not smooth. The prevailing ego-motion is rotational and a minor translational component. Figures 6.1 to 6.6 show the frames of the sequence and the results of our algorithm. The figures are arranged one per page so the reader can try to "watch" the sequence and the results like a flip-book. It may be hard to see the effects of our algorithm in the disjoint frames, but it becomes intuitively clear when seeing them as a movie.

The sequence of (D) images shows how our algorithm can be used to stabilize the original sequence in reference to the background. In the (D) sequence, the effects of the camera motion are successfully removed by using the motion parameters our algorithm estimated to inverse-transform camera motion effects. All images are projected into the frame of reference of the first image and subsequently superimposed. The match of the superimposed new image, which was projected in the frame of reference of the first image, to the previous images is a measure for the quality of the motion estimation.

The series of (E) images uses the output of our algorithm to compensate for object motion. The (E) sequence is generated like the (D) sequence but this time the object motion parameters are used. Hence, object motion is removed and this results in a tracking sequence: the object remains in the same image location regardless of where the camera moves. (Of course, the object has to be kept within the angle of view.) In the Can on Table Sequence we were trying to track the can over time. While we were manually keeping the can in view, our algorithm was used to refine the tracking (stabilize the can) and keep it almost motionless in the center of each frame.

Figure 6.7 integrates information from the (D) and (E) series. It finally shows the trajectory of the object. This trajectory shows the 2–D path of the object in a stabilized background frame of reference. In other words it shows where the object moves in the scene, not where the object moves in the image plane.

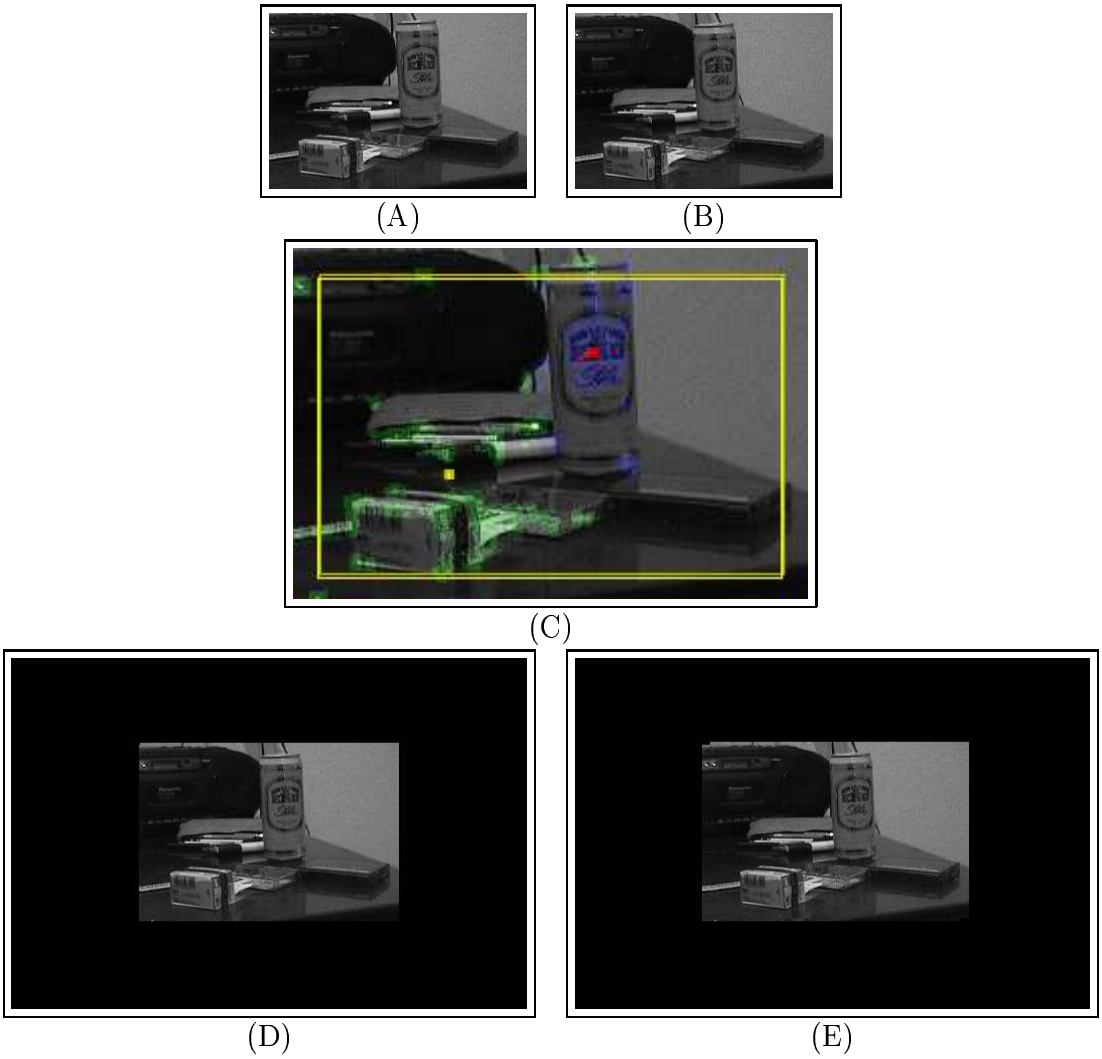


Figure 6.1: Frames 0 (A) and 1 (B) of the Can on Table Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 1.

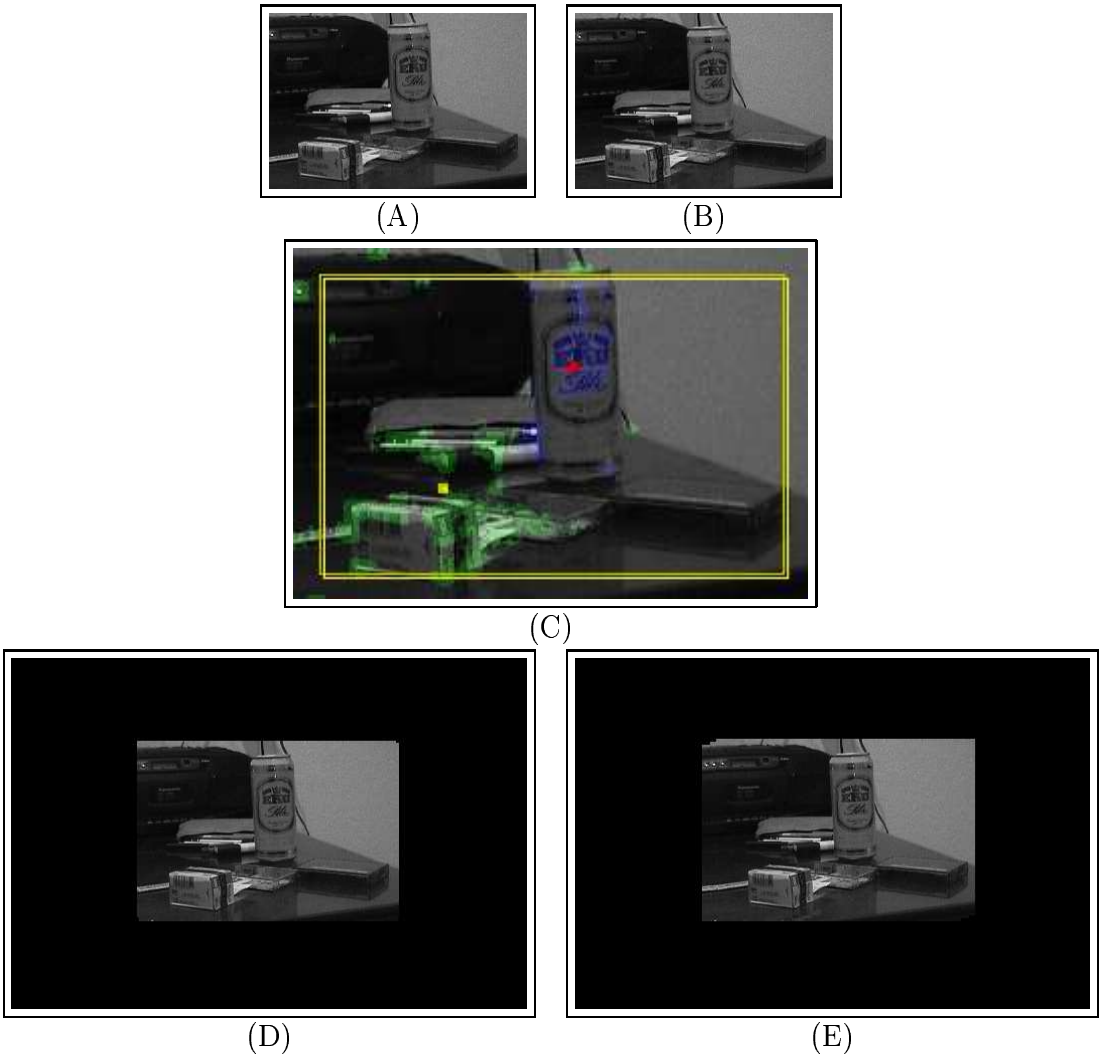


Figure 6.2: Frames 1 (A) and 2 (B) of the Can on Table Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 2.

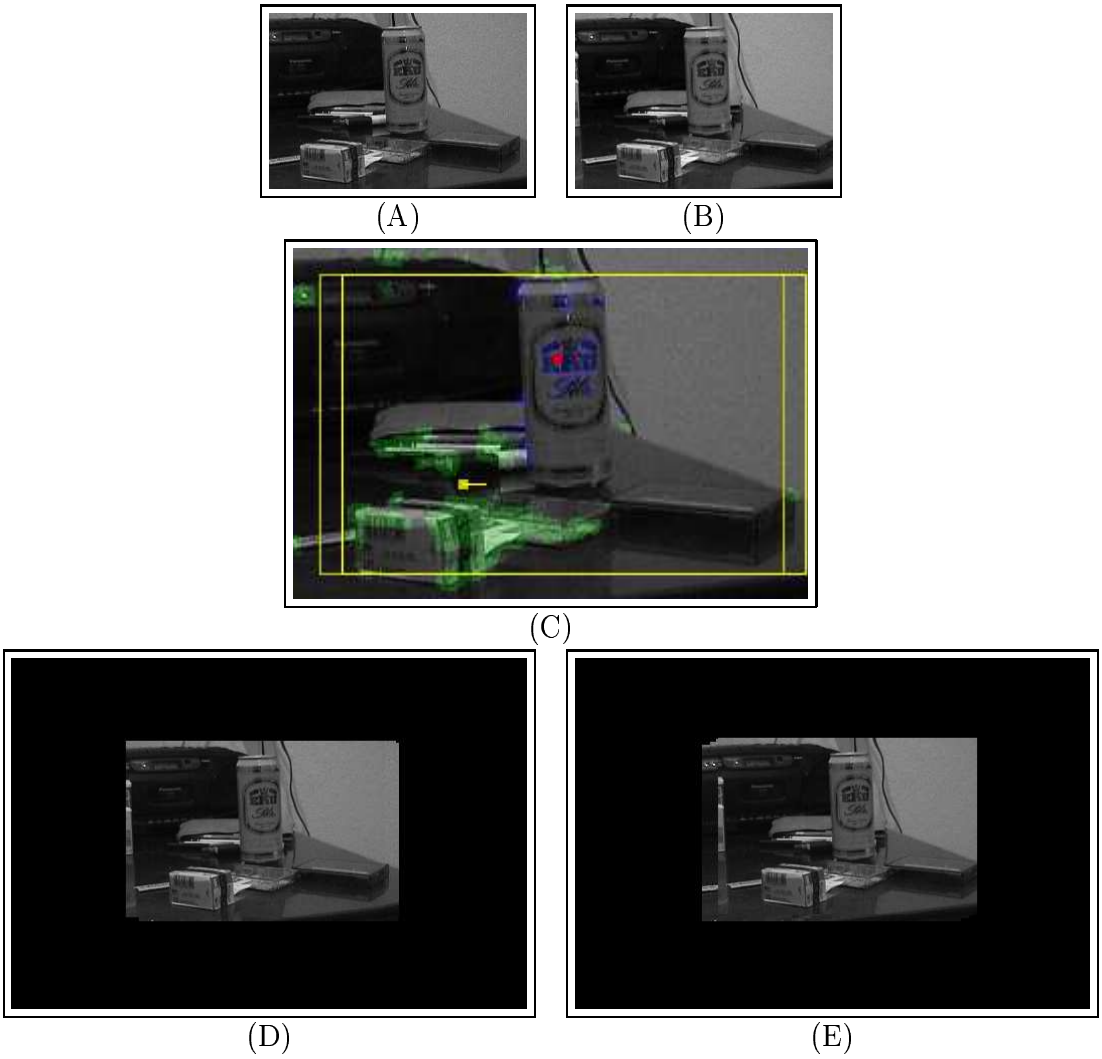


Figure 6.3: Frames 2 (A) and 3 (B) of the Can on Table Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 3.

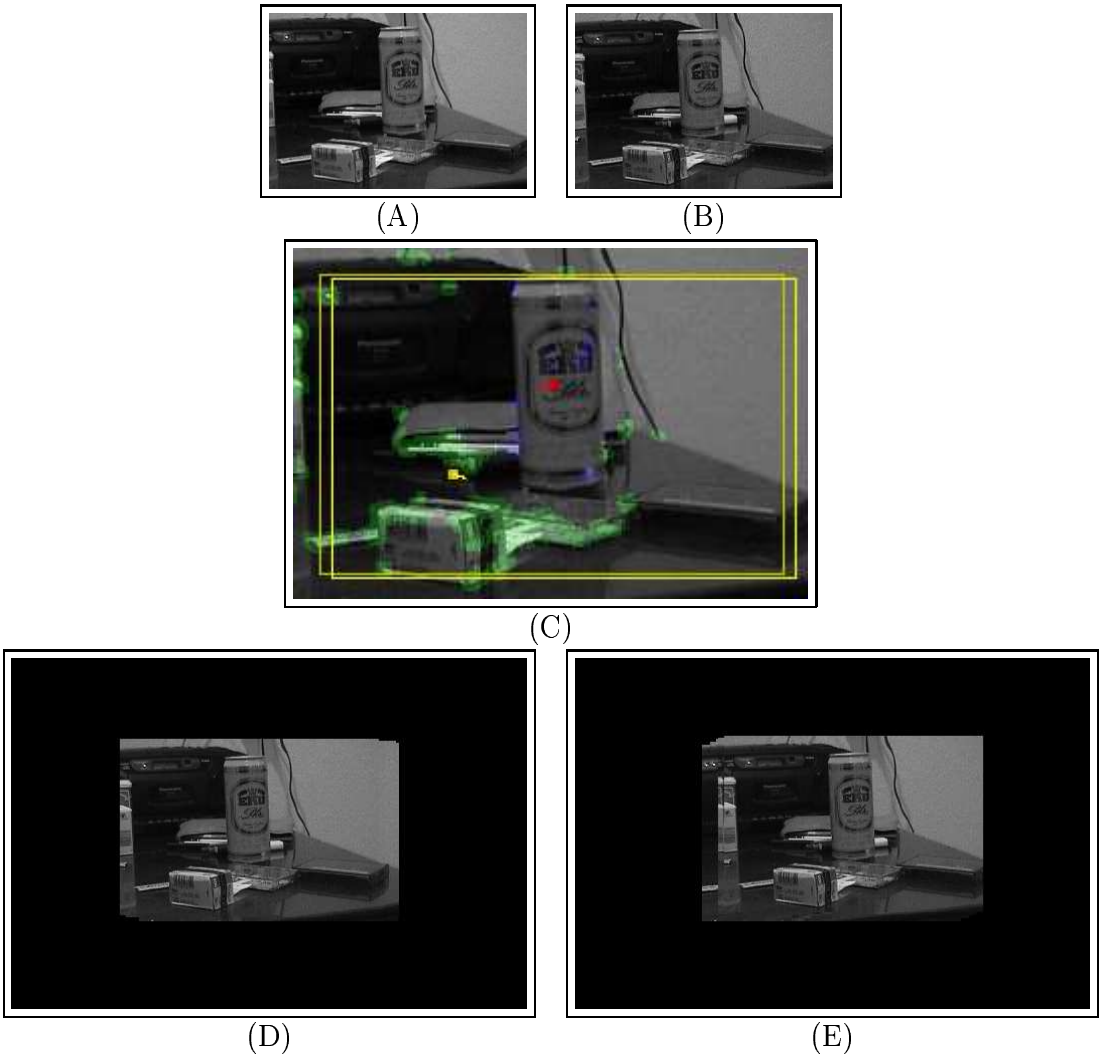


Figure 6.4: Frames 3 (A) and 4 (B) of the Can on Table Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 4.

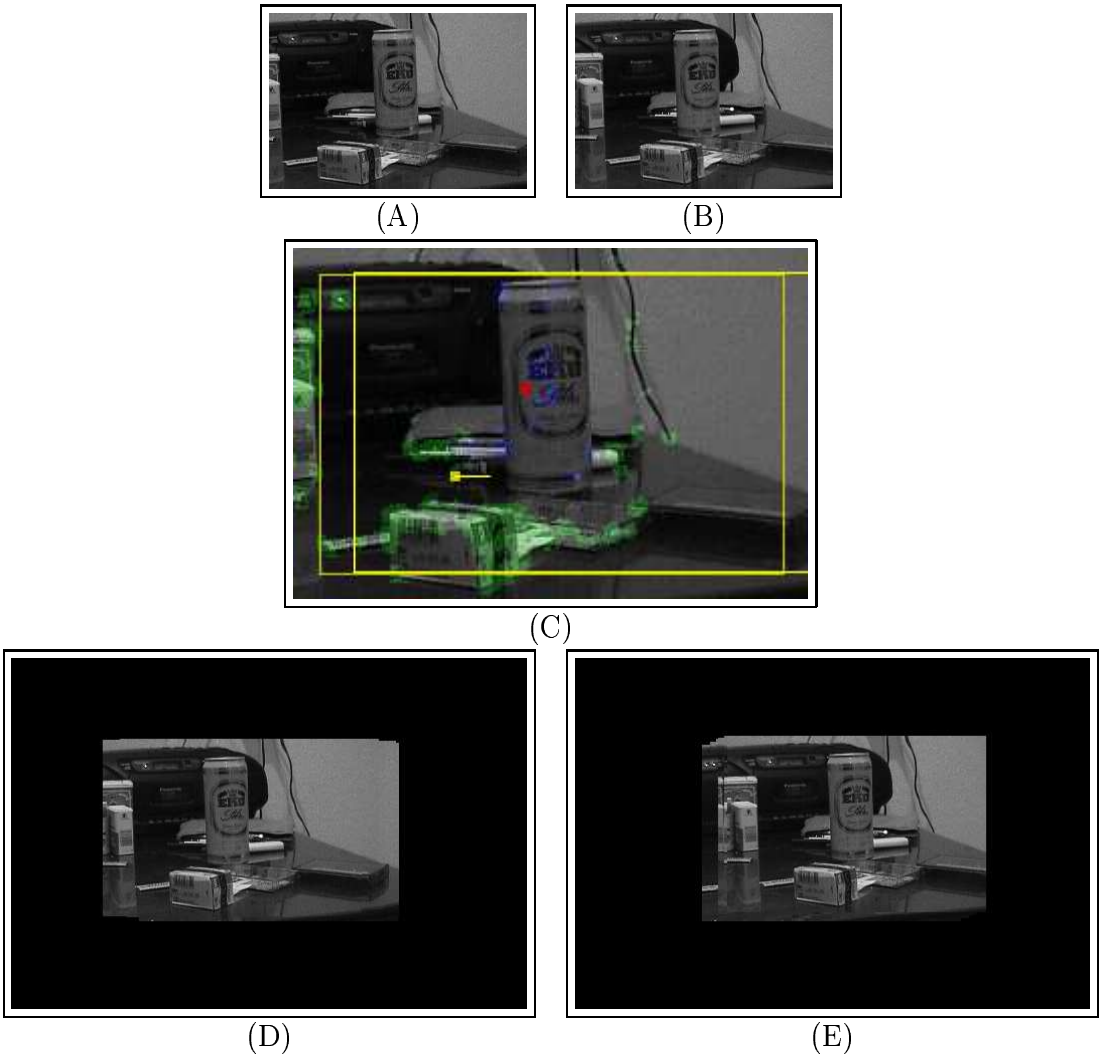


Figure 6.5: Frames 4 (A) and 5 (B) of the Can on Table Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 5.

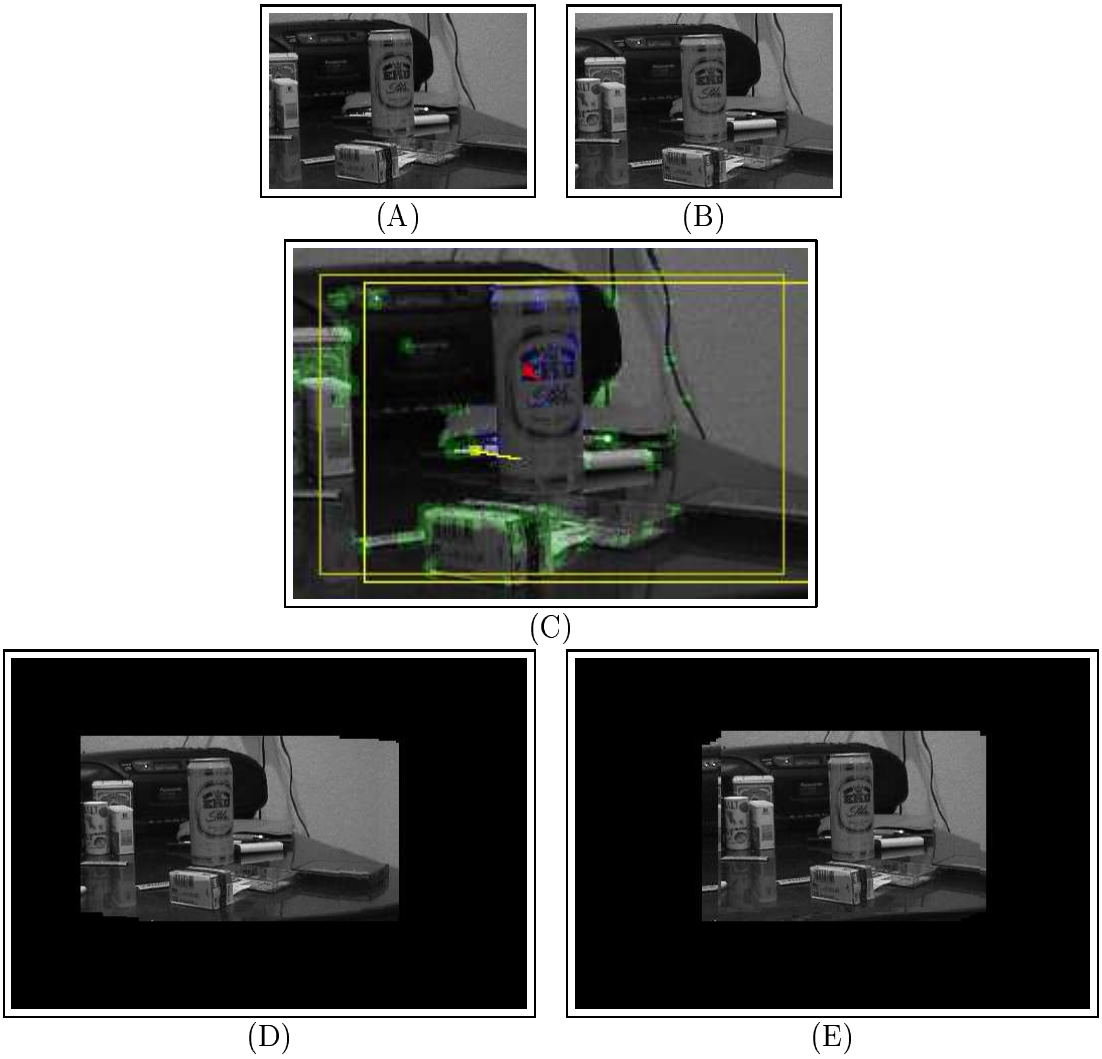


Figure 6.6: Frames 5 (A) and 6 (B) of the Can on Table Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 6.

Figure 6.7: Object trajectory obtained from Can on Table Sequence. The trajectory is backprojected and overlaid with the first frame 0.

6.2 Car Sequence

This sequence shows a car driving to the left on a road viewed by a downward panning camera. Figures 6.8 to 6.15 show the original image pairs (A) and (B), the motion analysis illustration (C), background stabilized (D) and object stabilized images (E). The recovered trajectory of the object's centroid is depicted in figure 6.16.

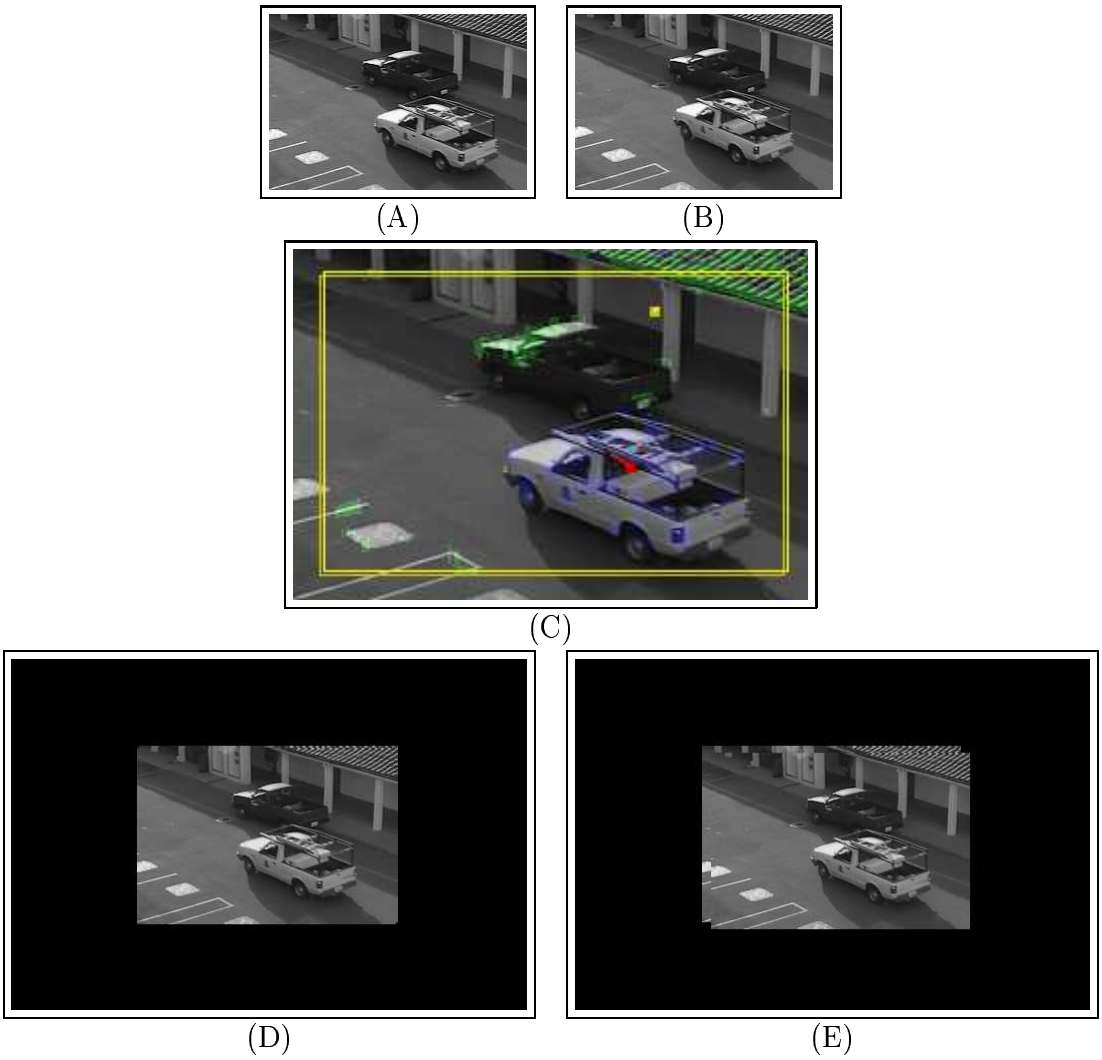


Figure 6.8: Frames 0 (A) and 1 (B) of the Car Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 1.

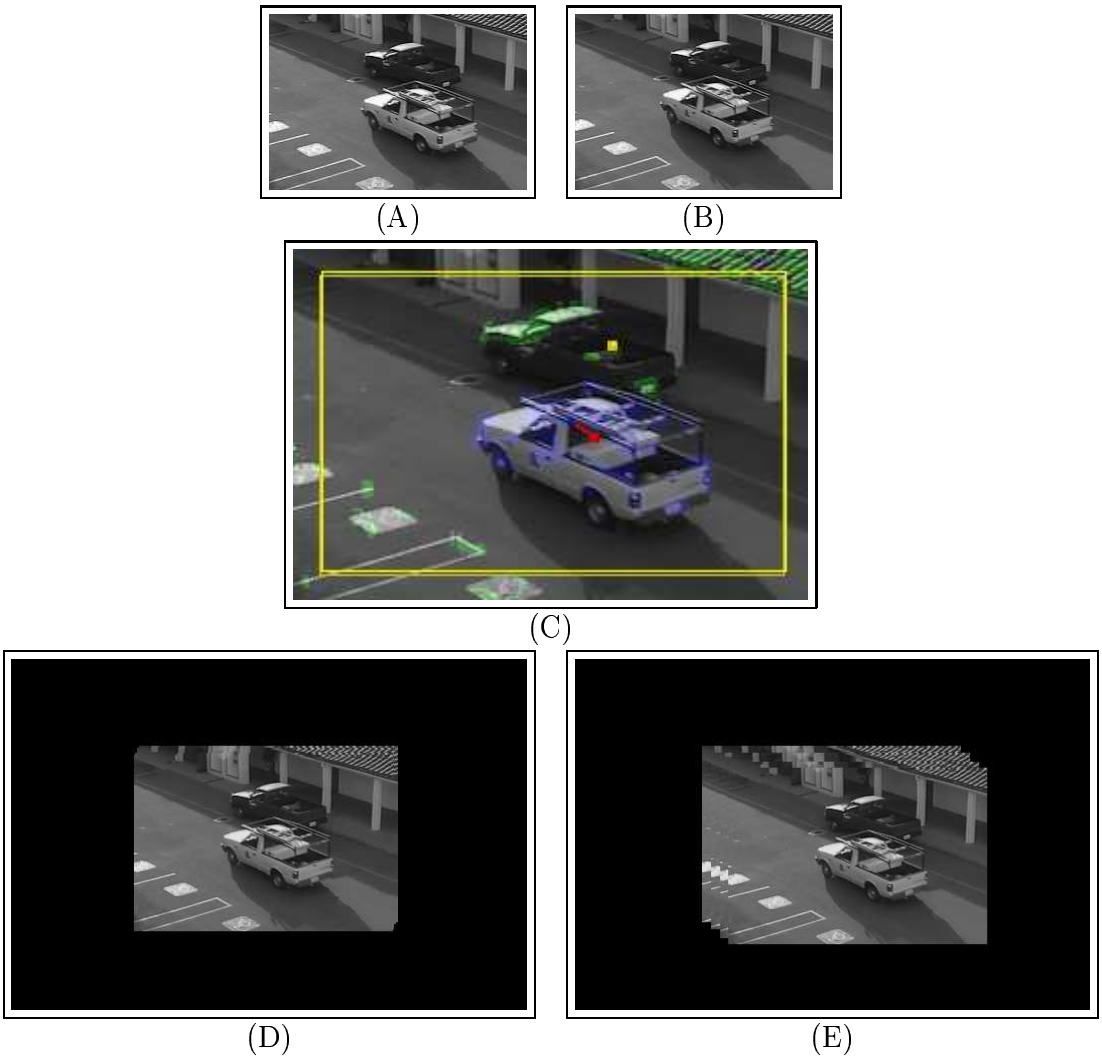


Figure 6.9: Frames 2 (A) and 3 (B) of the Car Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 3.

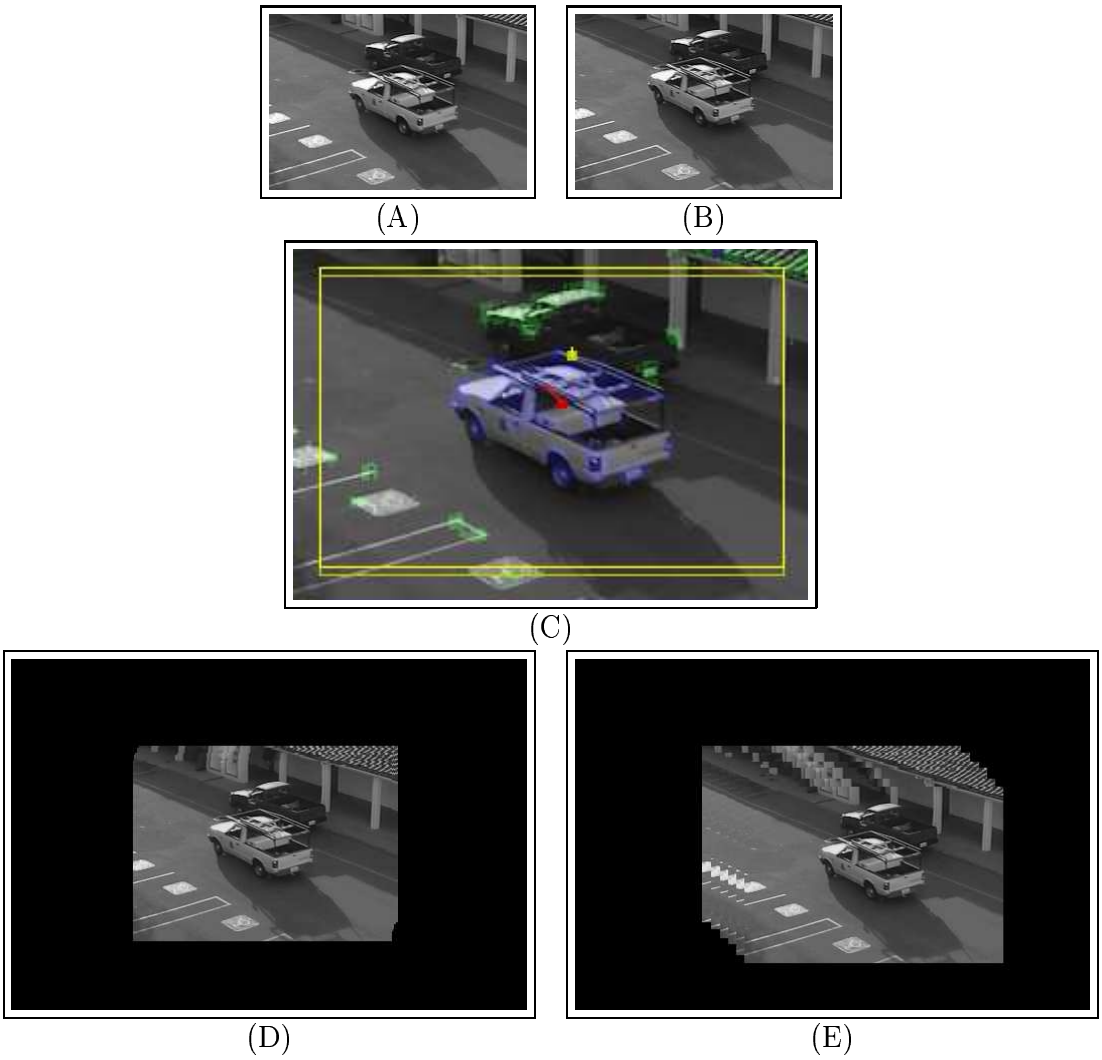


Figure 6.10: Frames 4 (A) and 5 (B) of the Car Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 5.

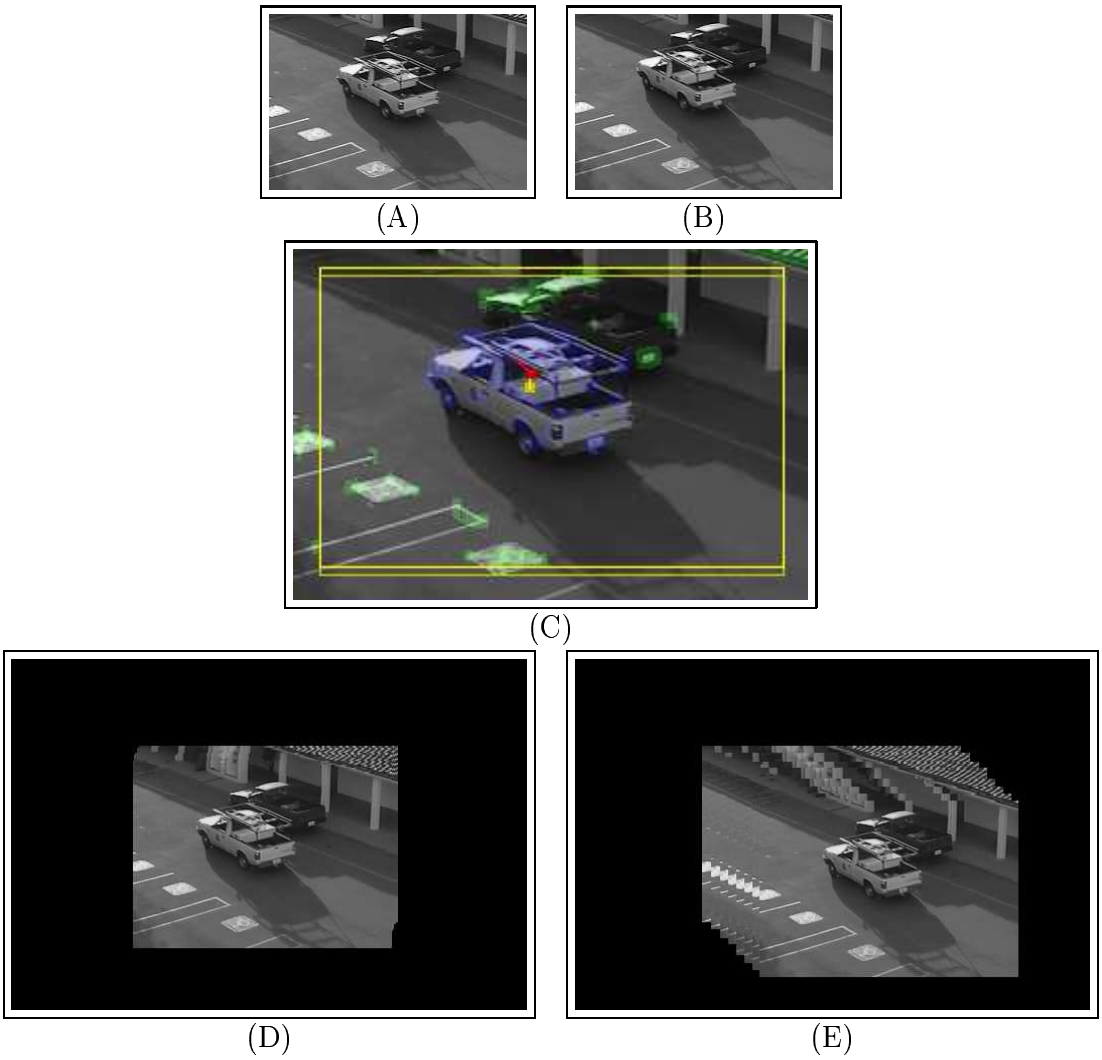


Figure 6.11: Frames 6 (A) and 7 (B) of the Car Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 7.

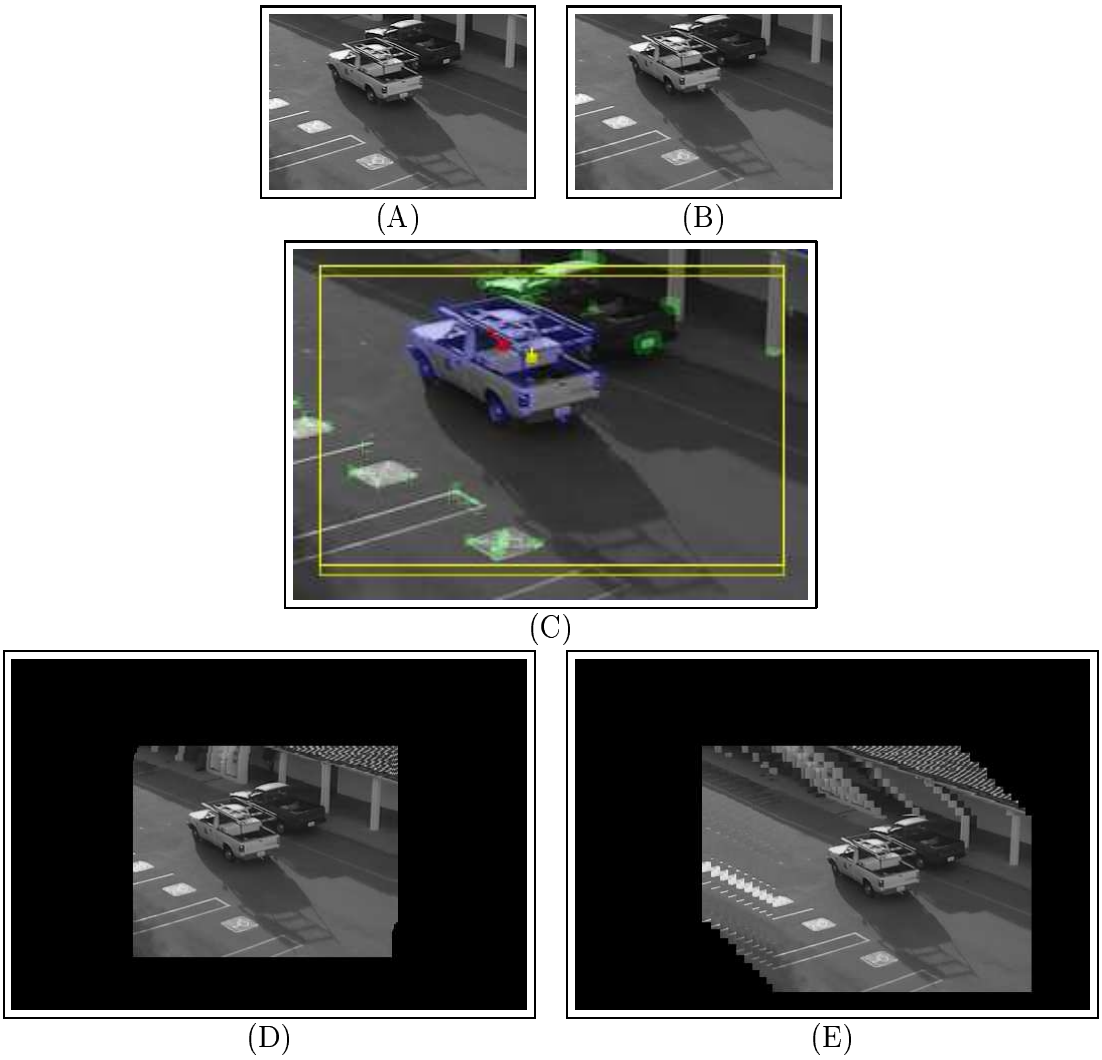


Figure 6.12: Frames 8 (A) and 9 (B) of the Car Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 9.

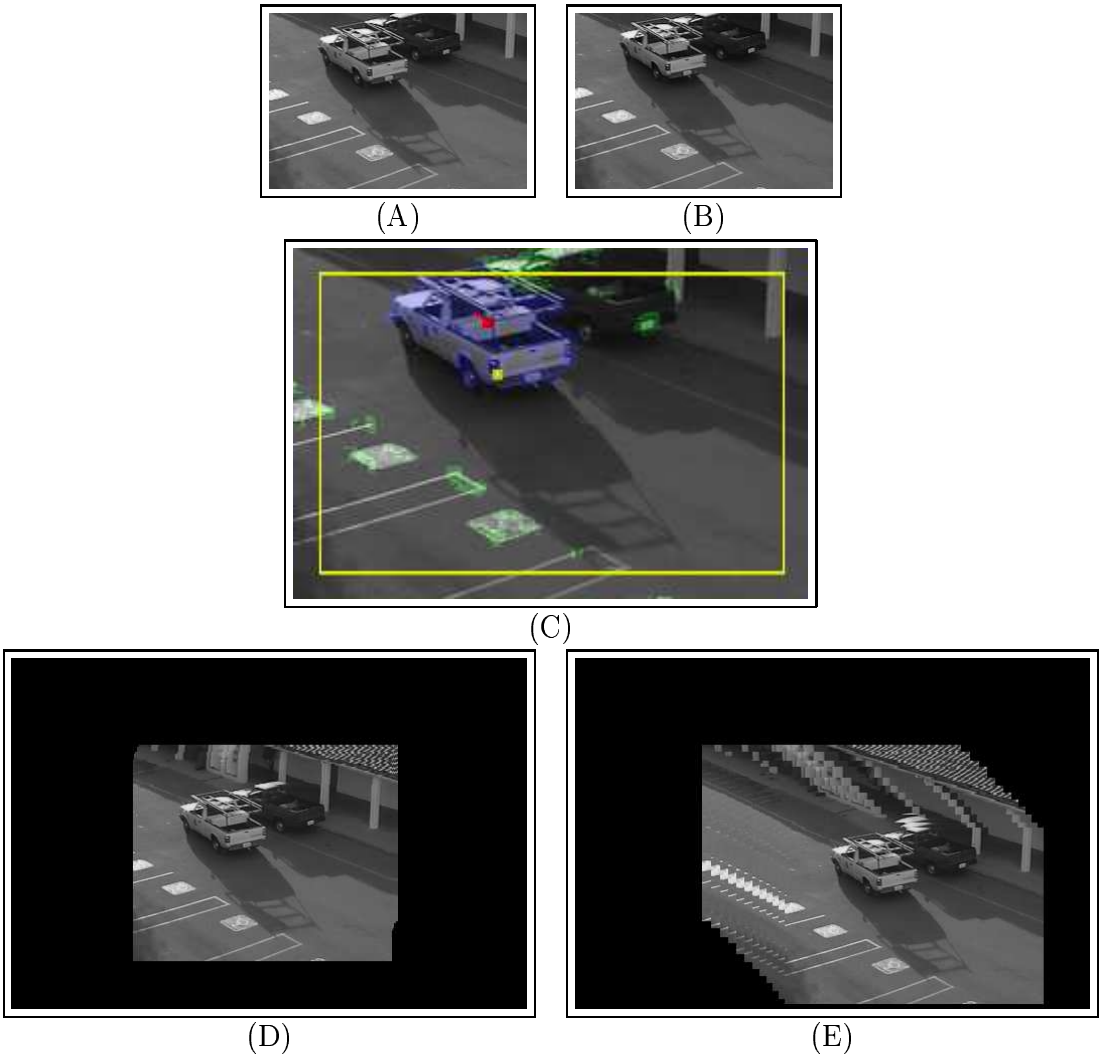


Figure 6.13: Frames 10 (A) and 11 (B) of the Car Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 11.

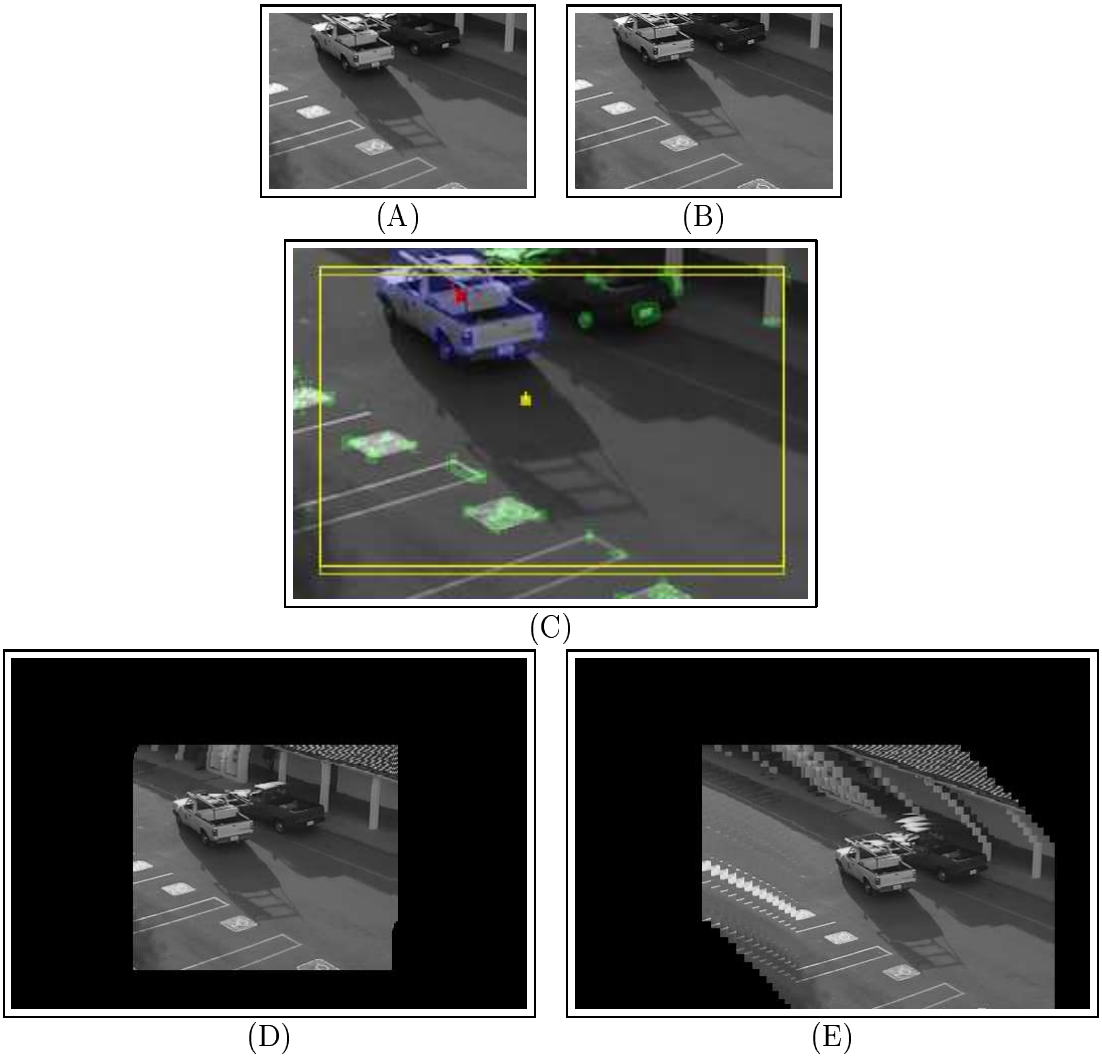


Figure 6.14: Frames 12 (A) and 13 (B) of the Car Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 13.

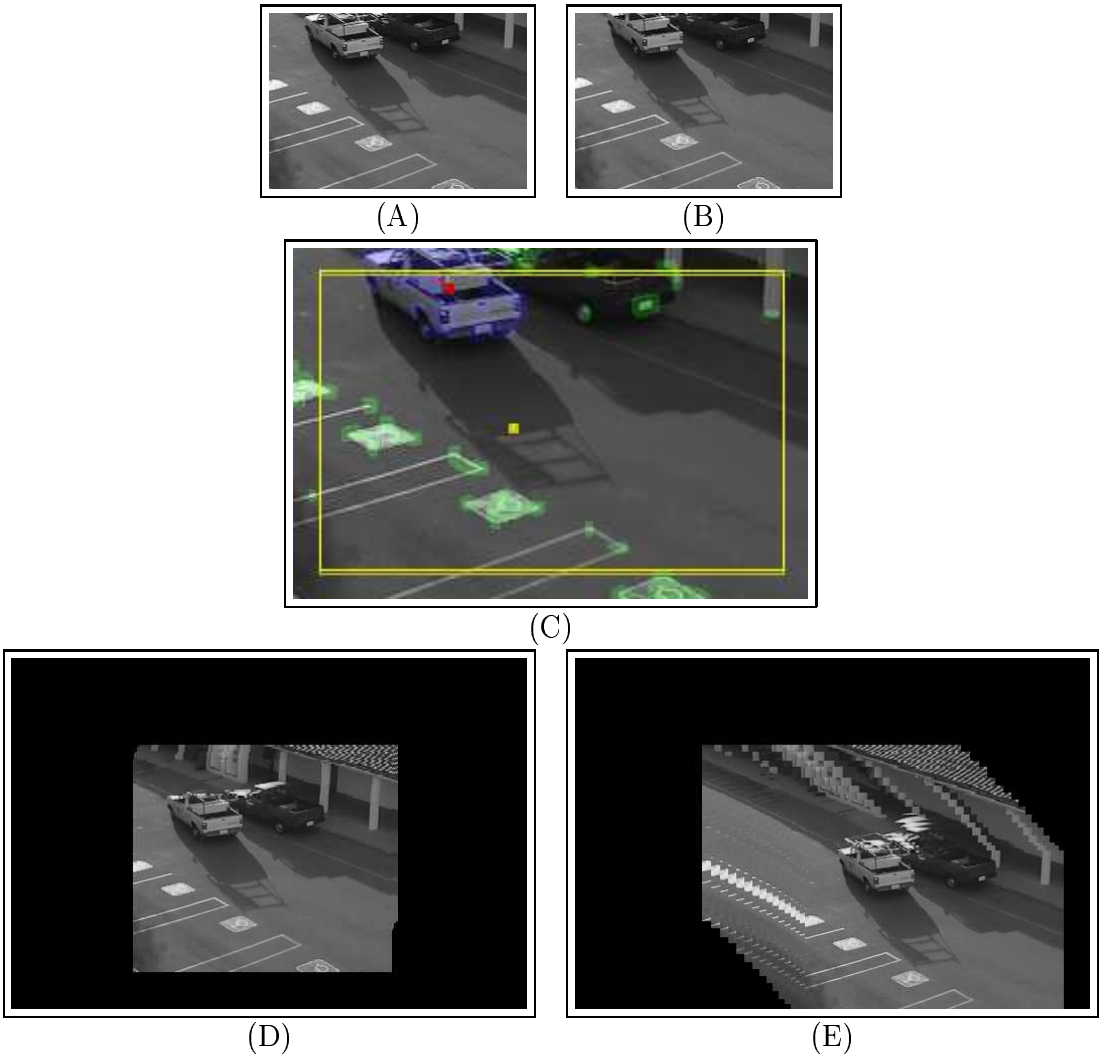


Figure 6.15: Frames 14 (A) and 15 (B) of the Car Sequence. Illustration of inter-frame analysis results (C). Mosaiced image with stabilized background (D) and image stabilized in reference to object (E) after processing frames 0 to 15.

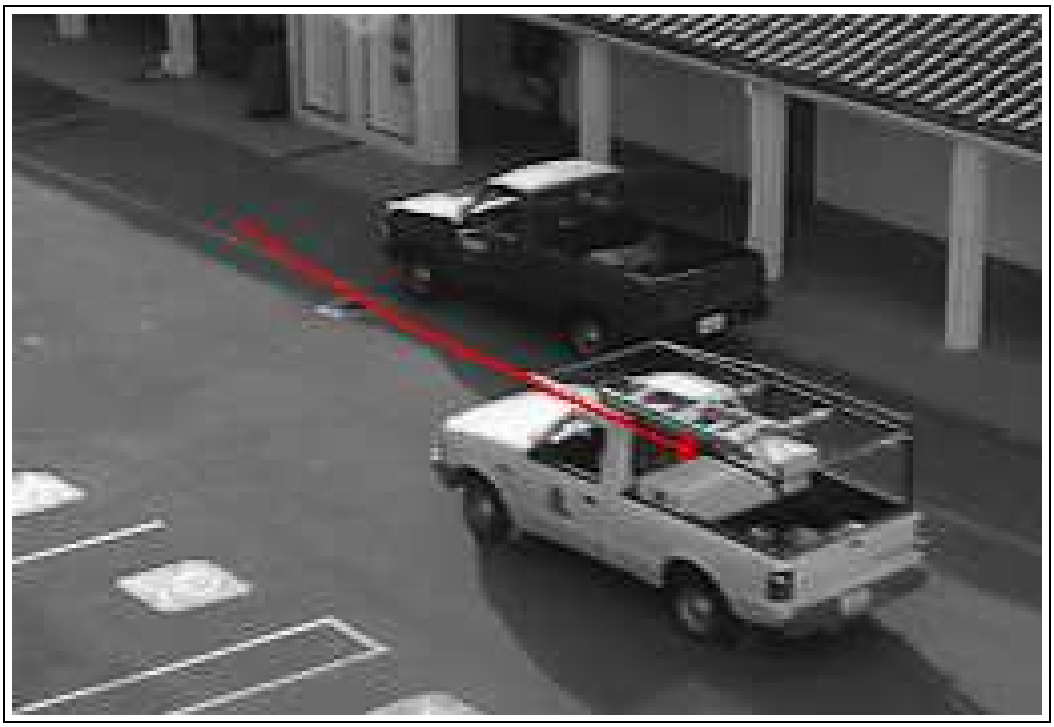


Figure 6.16: Object trajectory obtained from Car Sequence. The trajectory is backprojected and overlaid with the first frame 0.

6.3 FLIR ATR Sequence

We also applied our algorithm to a Forward Looking Infra Red (FLIR) image sequence used for Automated Target Recognition (ATR). The original sequence consists of 290 frames. The sequence can be partitioned into two phases. In the first phase, the camera is approaching a bright feature on the ground plane. No significant independently (from the camera) moving object is present. This is not according to our assumption of two motions, and consequently this part of the sequence gets over-segmented. However, our algorithm still provides satisfactory stabilized images that are compensated for camera motion.

During the second phase, a vehicle enters the scene from the right and moves towards the center of the image, while the camera is still moving forward. The entire sequence was sub-sampled before processing. We decided to use every third frame and consequently we processed 97 frames. The first phase of the sequence includes frames 0 to 64, and the second, frames 65 to 96. The quality of this sequence is very poor for our purposes compared to common visual sequences. The sequence exhibits:

- Low contrast
- Base gray-level fluctuation
- Artifacts on the right image margin (scan-line ends)
- Stationary contamination (black spots) in the optical system with more contrast than in the actual scene

- Very few features in scene
- Only two bright reliable blobs to track
- Sudden and very large image displacements (struck camera)

We decided to cut four pixels on each side because of the artifacts on the right image margin. For future experiments, the utilization of a contrast-normalizing preprocessing stage should be considered. The fact that only two good trackable features are present limits the accuracy of the estimation. This becomes obvious, especially towards the end of the sequence as errors accumulate.

Figures 6.17 to 6.23 show the first part of the original sequence and our motion analysis results. Please note that only every tenth pair is displayed. Due to the fact that this part of the sequence does not contain two distinct motions as is a required for our algorithm the motion analysis results are oversegmented.

Figures 6.25 (A) to 6.31 (A) present the background stabilization results for the first part of the sequence. The images are still properly registered despite the severe violation of the two motions assumption. The forward motion is clearly recovered and illustrated by the size reduction of the backprojected frames in the (A) sequence towards the end. The object stabilized series (figures 6.31 (B) should in principle be the same as the (A) series in the absence of an IMO. However fewer vectors are used to estimate object motion. Hence, the total object motion transformation accumulates more errors over time and the (B) sequence deviates significantly form the (A) sequence. The trajectory as

presented in figure 6.24 shows that the IMO did not move much in the scene over time. In other words, there is no IMO, which is true.

For printing purposes the gray-level palette of all images in this thesis has been spread to use the entire range. In the stabilized and mosaiced images, however, this was not done to maintain the same reference palette for all frames. Hence fluctuations in base gray-level of the low quality input sequence are visible in the stabilized and mosaiced images (figures 6.25 to 6.31 and 6.43 to 6.52) but remain unnoticeable in the print of the original sequence and motion results (figures 6.17 to 6.23 and 6.32 to 6.41).

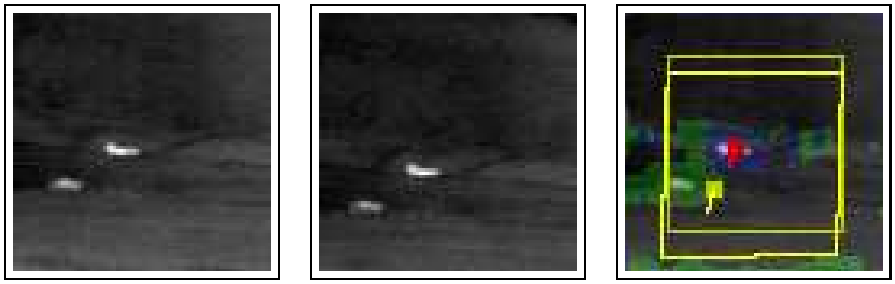


Figure 6.17: Frame 0, frame 1 and motion analysis of FLIR ATR Sequence.

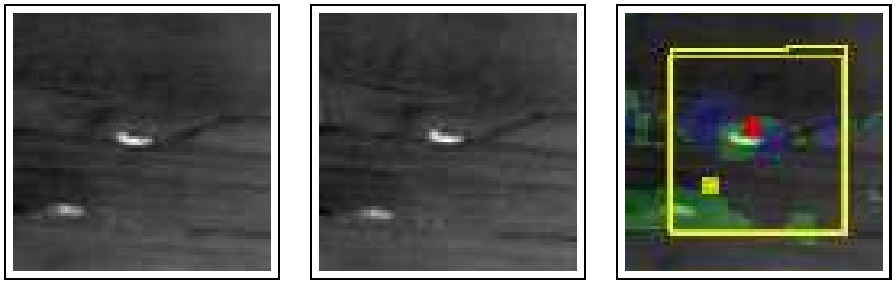


Figure 6.18: Frame 10, frame 11 and motion analysis of FLIR ATR Sequence.

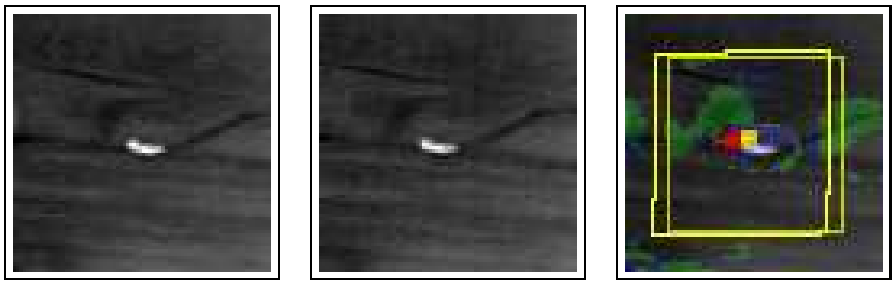


Figure 6.19: Frame 20, frame 21 and motion analysis of FLIR ATR Sequence.

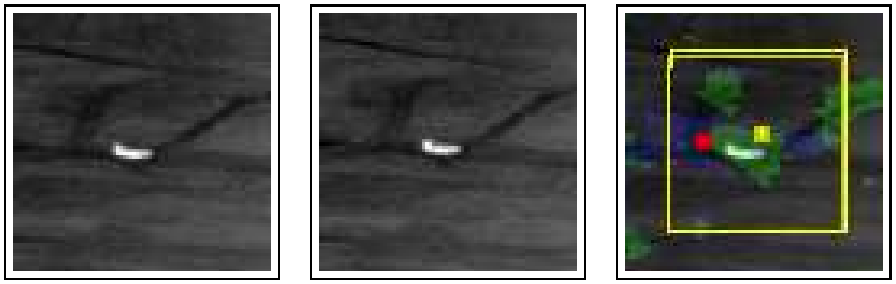


Figure 6.20: Frame 30, frame 31 and motion analysis of FLIR ATR Sequence.

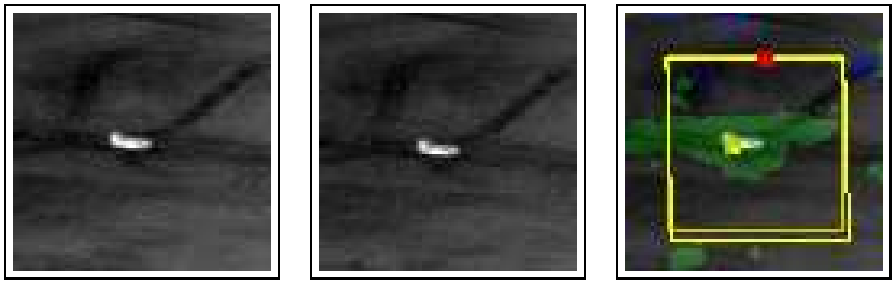


Figure 6.21: Frame 40, frame 41 and motion analysis of FLIR ATR Sequence.

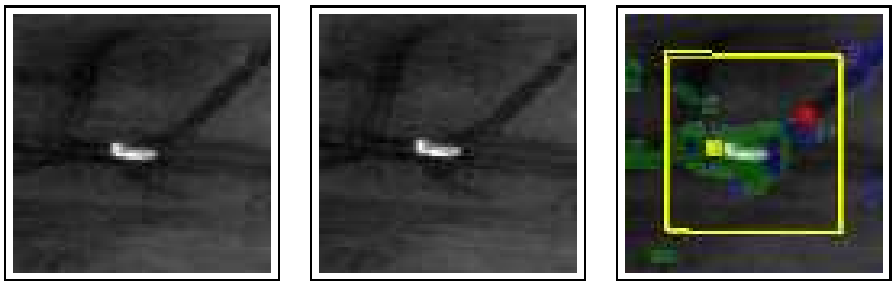


Figure 6.22: Frame 50, frame 51 and motion analysis of FLIR ATR Sequence.

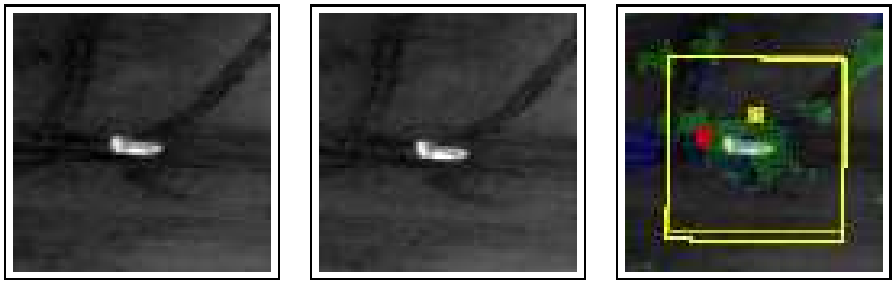


Figure 6.23: Frame 60, frame 61 and motion analysis of FLIR ATR Sequence.

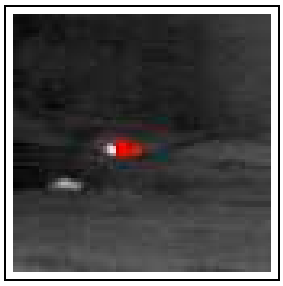
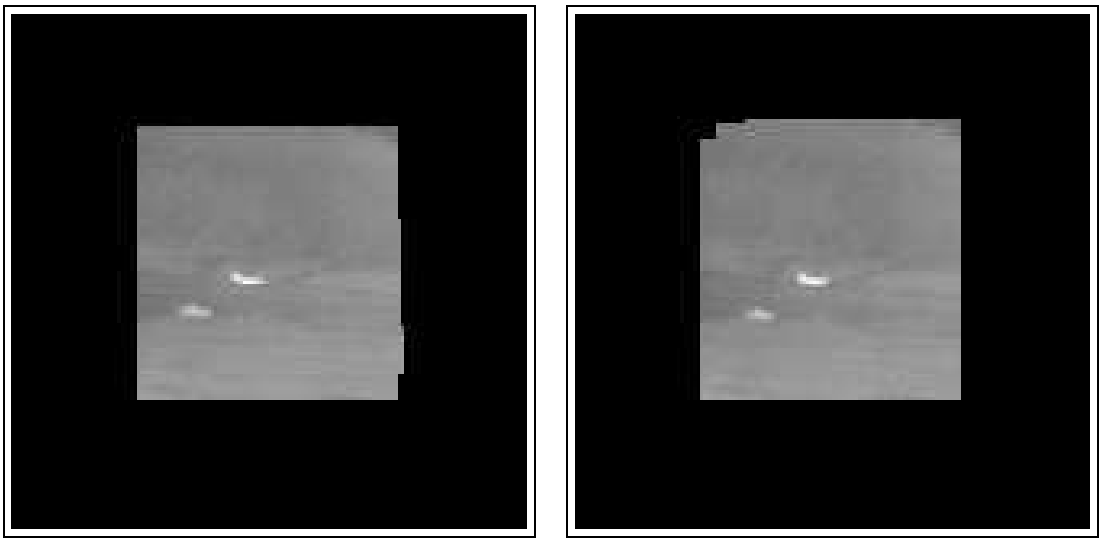
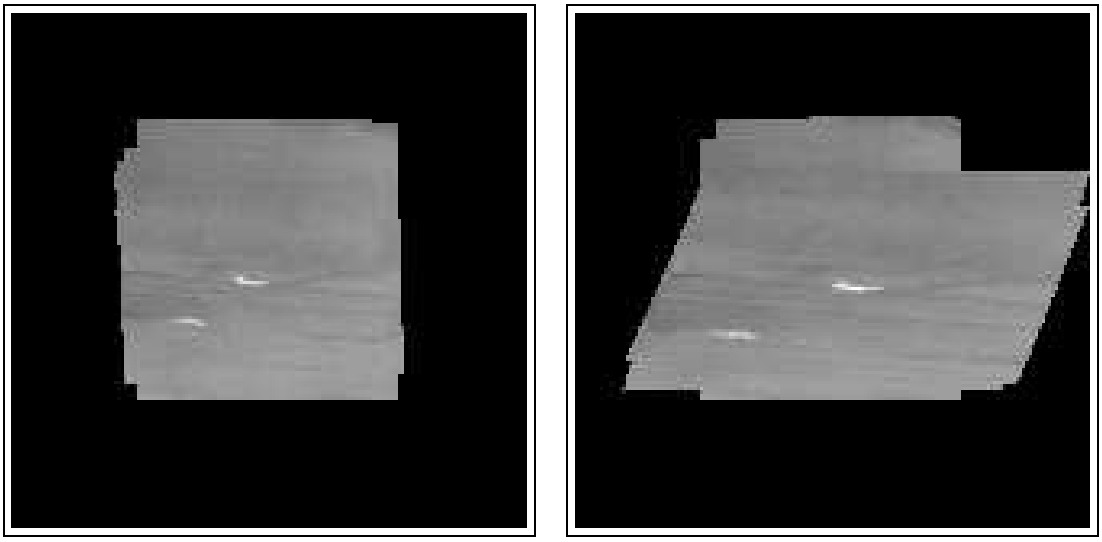


Figure 6.24: Object trajectory obtained from FLIR ATR Sequence. The trajectory is backprojected and overlaid with the first frame 0.



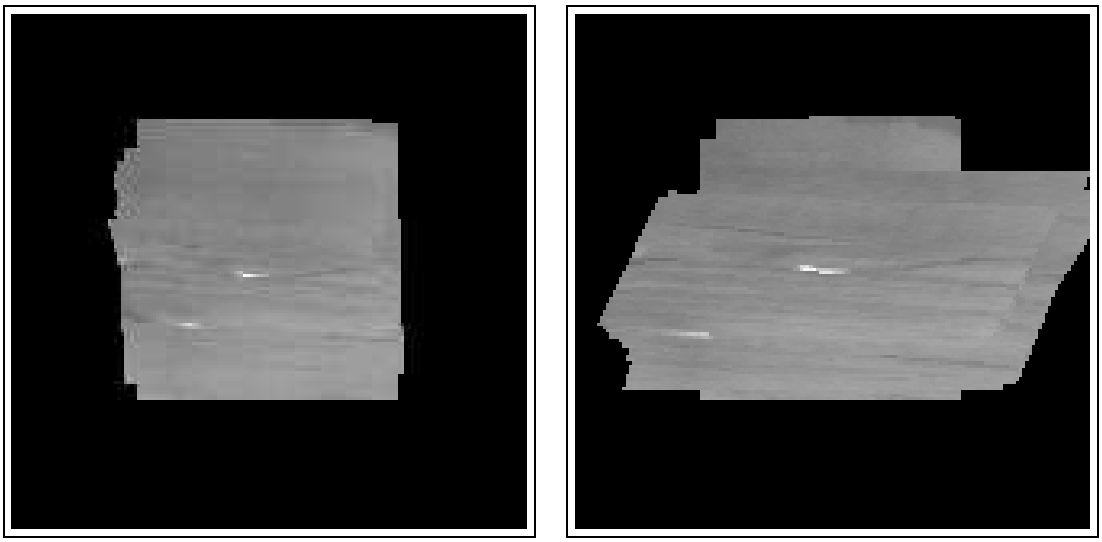
(A) (B)

Figure 6.25: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 0 to 1 of the FLIR ATR Sequence.



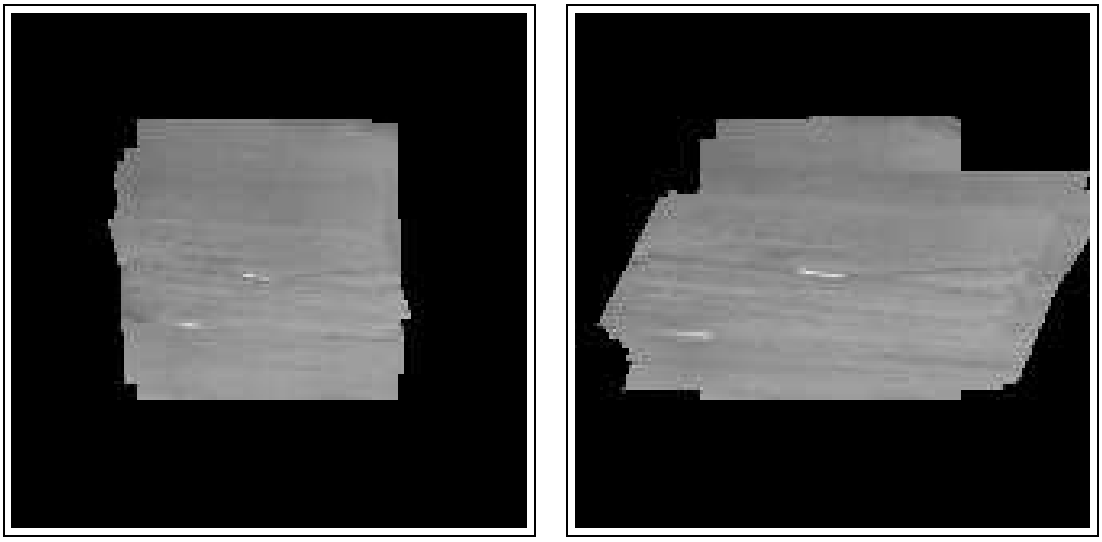
(A) (B)

Figure 6.26: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 0 to 11 of the FLIR ATR Sequence.



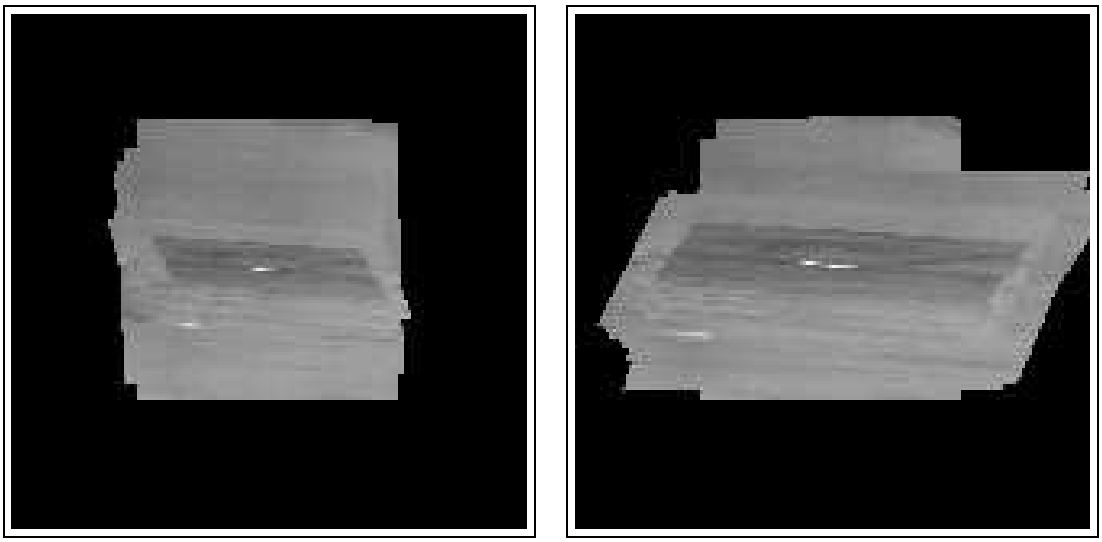
(A) (B)

Figure 6.27: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 0 to 21 of the FLIR ATR Sequence.



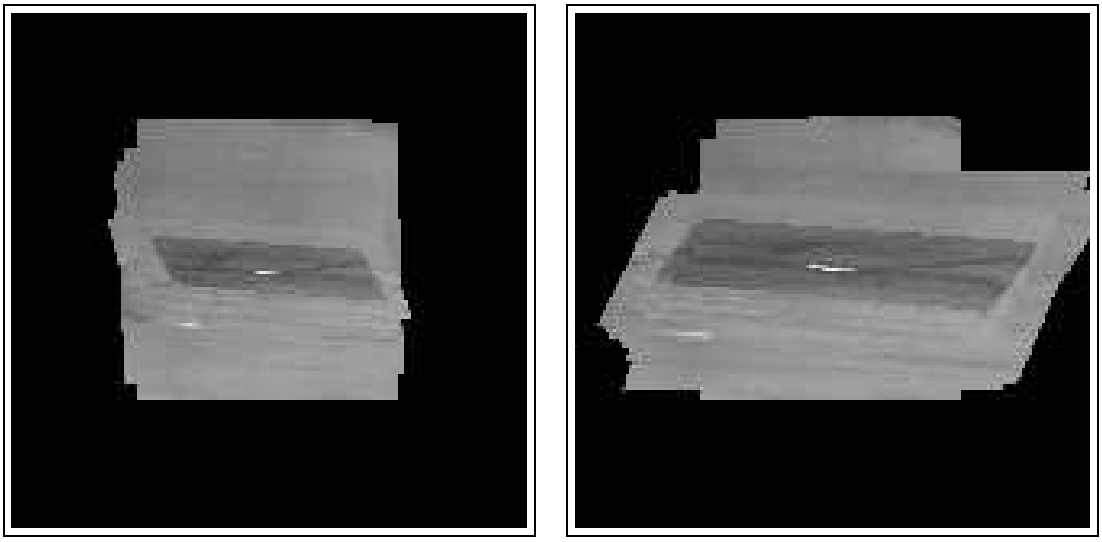
(A) (B)

Figure 6.28: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 0 to 31 of the FLIR ATR Sequence.



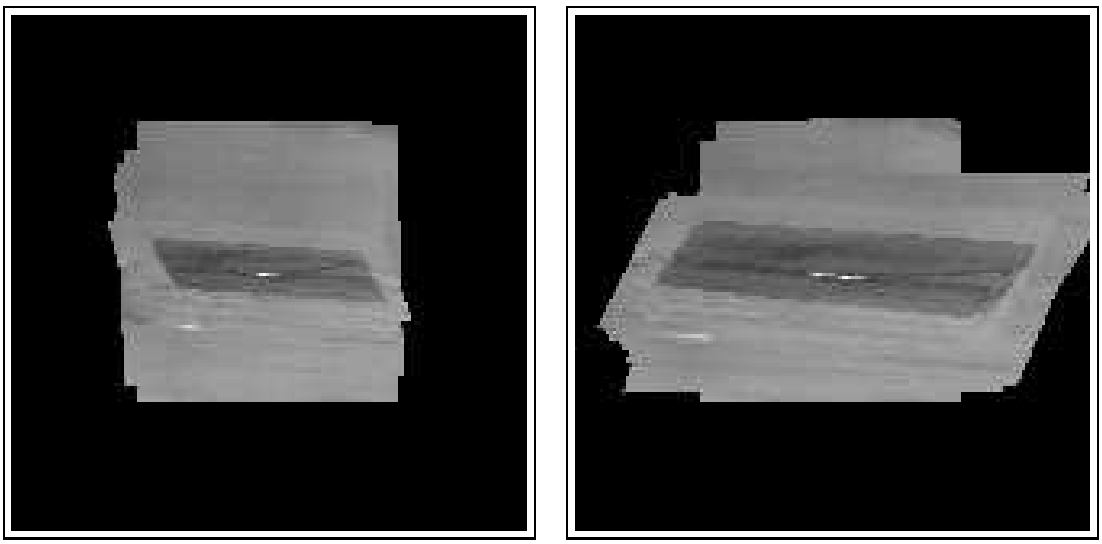
(A) (B)

Figure 6.29: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 0 to 41 of the FLIR ATR Sequence.



(A) (B)

Figure 6.30: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 0 to 51 of the FLIR ATR Sequence.



(A) (B)

Figure 6.31: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 0 to 61 of the FLIR ATR Sequence.

Figures 6.32 to 6.41 depict the second part of the sequence and our motion analysis illustrations. In this part a truck driving to the left enters the scene from the right. The independently moving truck is successfully segmented from the background. The background in this case mainly consists of the stationary tank in the center of the view. In frame 79 to frame 80 the two vehicles have moved to close together so that their motion estimation becomes unstable. The camera gets struck heavily from frame 80 to frame 81 and both objects are displaced far from their previous locations. Our system cannot recover the correct location correspondences for this pair of frames with the used parameters.

The stabilization and mosaicing results are presented in figures 6.43 to 6.52. Due to the fact that the stabilized sequences are totally dependent on all results since the frame of reference, errors accumulate over time and the stabilized sequences cannot recover from the major misinterpretation between frames 80 and 81. As described in chapter 4, a reset to a new frame of reference would be necessary. In figure 6.42 the beginning of the trajectory is accurately recovered. Starting with the bend towards the upper image margin, the trajectory is erratic. Considering the low quality of this sequence with a lack of features and significant jumps in camera orientation and base image brightness, the presented results are very good.

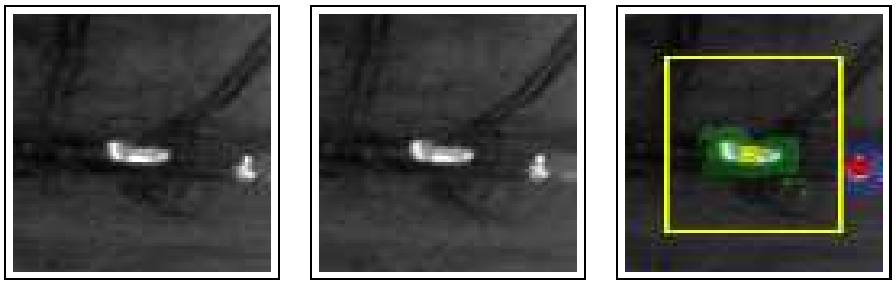


Figure 6.32: Frame 72, frame 73 and motion analysis of FLIR ATR Sequence.

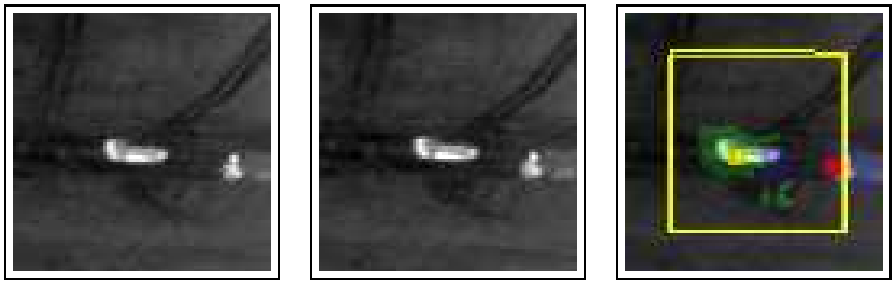


Figure 6.33: Frame 73, frame 74 and motion analysis of FLIR ATR Sequence.

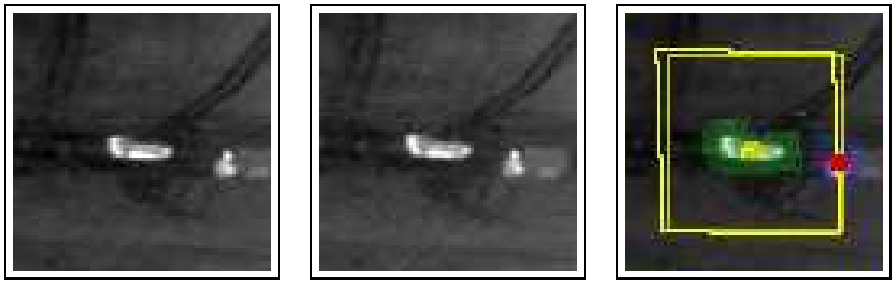


Figure 6.34: Frame 74, frame 75 and motion analysis of FLIR ATR Sequence.

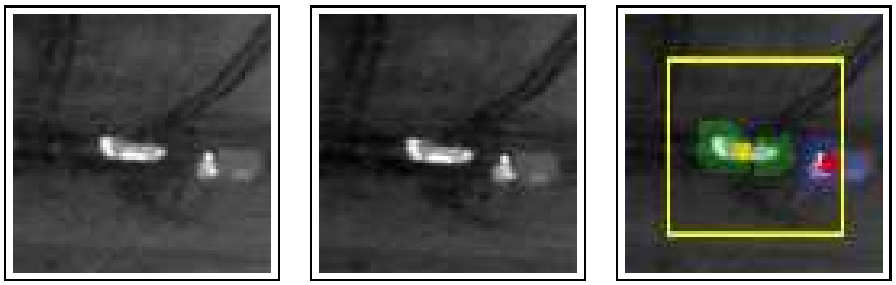


Figure 6.35: Frame 75, frame 76 and motion analysis of FLIR ATR Sequence.

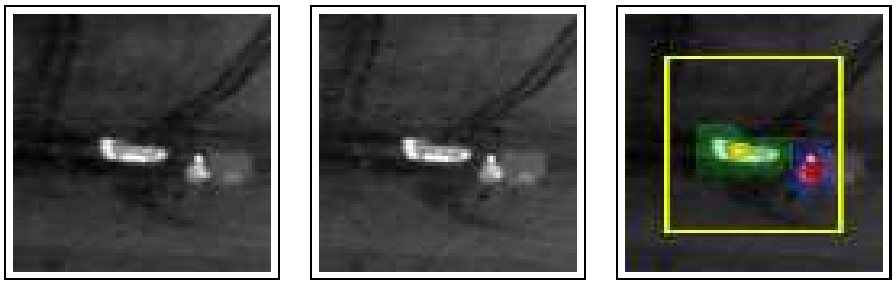


Figure 6.36: Frame 76, frame 77 and motion analysis of FLIR ATR Sequence.

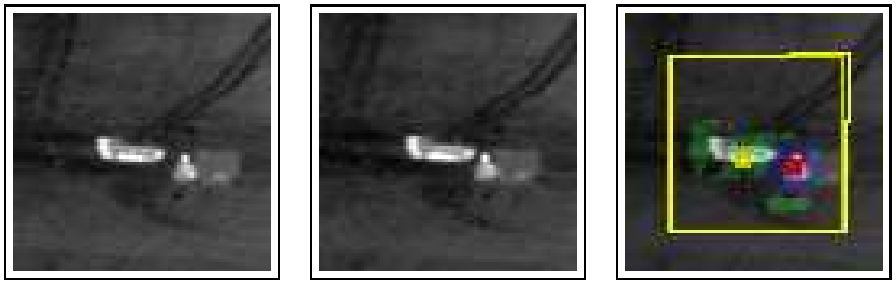


Figure 6.37: Frame 77, frame 78 and motion analysis of FLIR ATR Sequence.

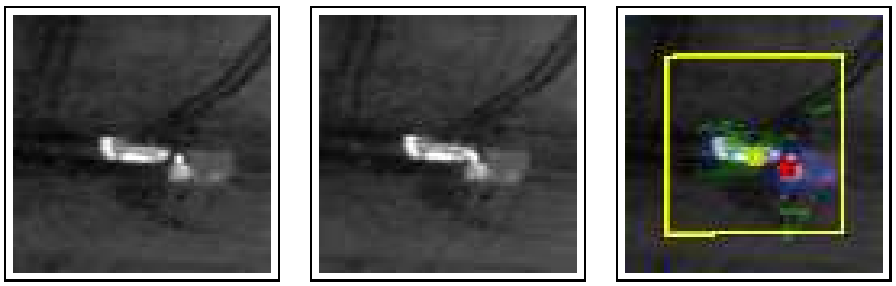


Figure 6.38: Frame 78, frame 79 and motion analysis of FLIR ATR Sequence.

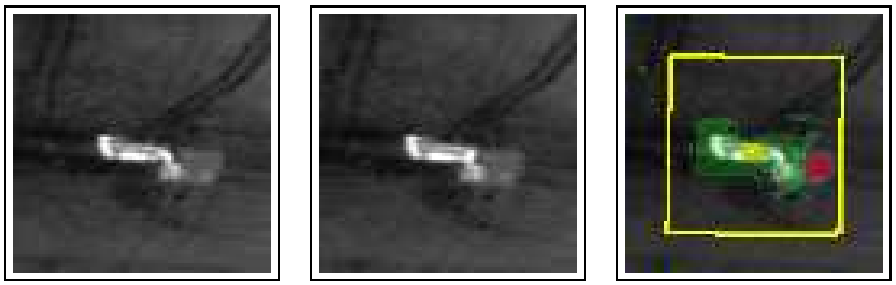


Figure 6.39: Frame 79, frame 80 and motion analysis of FLIR ATR Sequence.

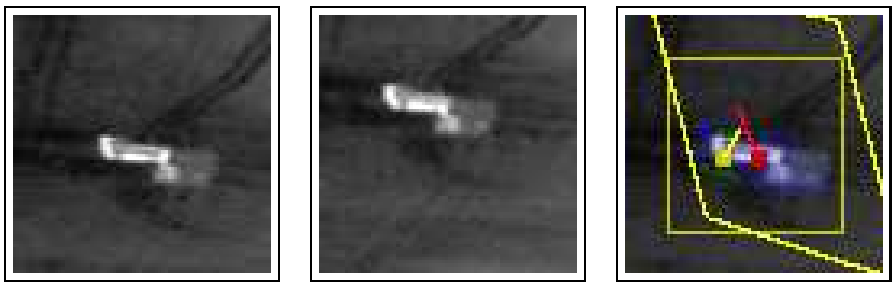


Figure 6.40: Frame 80, frame 81 and motion analysis of FLIR ATR Sequence.

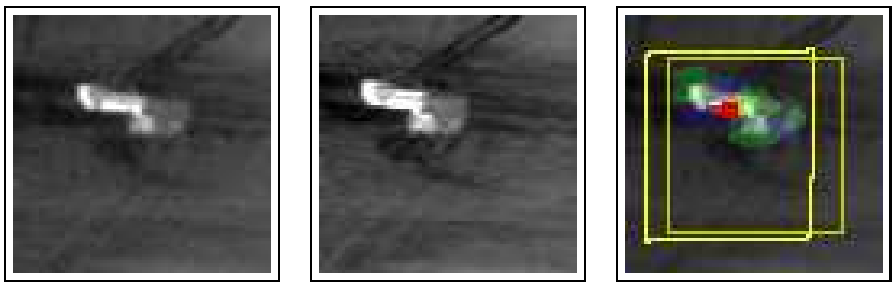


Figure 6.41: Frame 81, frame 82 and motion analysis of FLIR ATR Sequence.

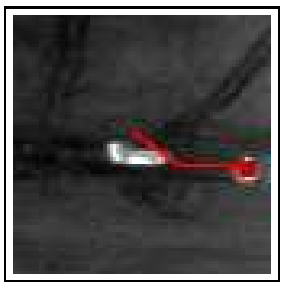
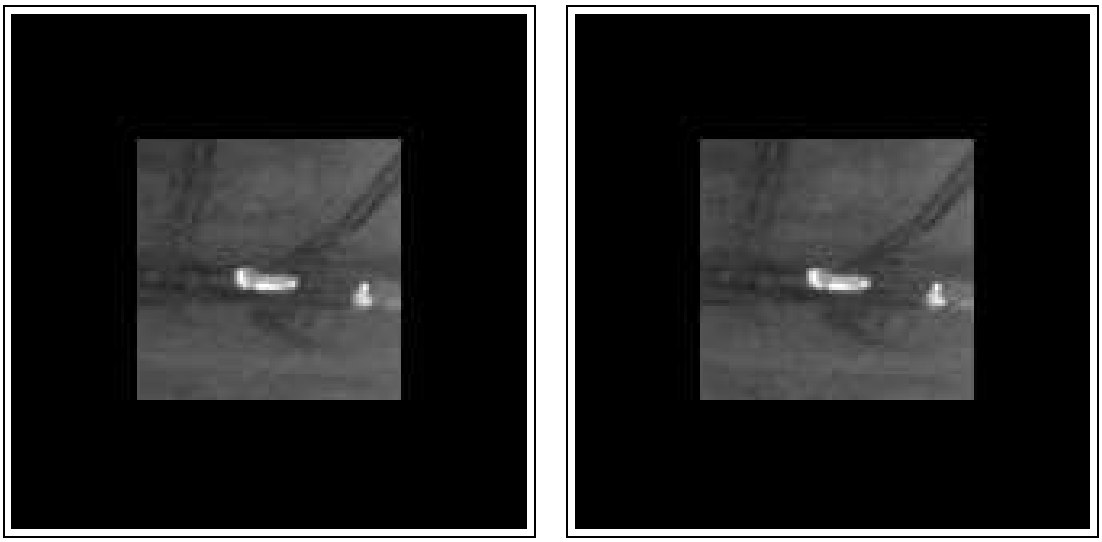
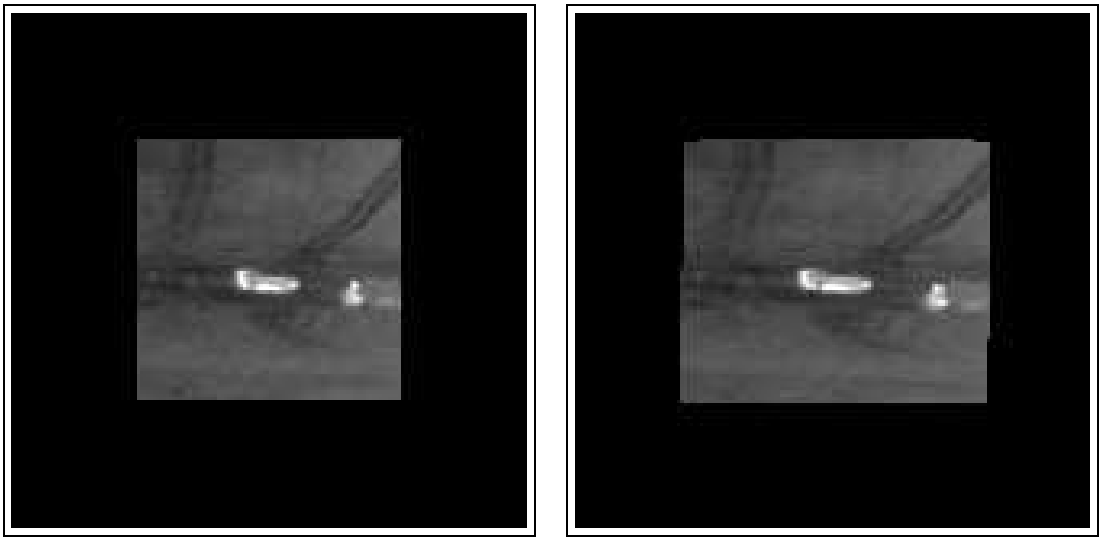


Figure 6.42: Object trajectory obtained from FLIR ATR Sequence. The trajectory is backprojected and overlaid with the first frame 72.



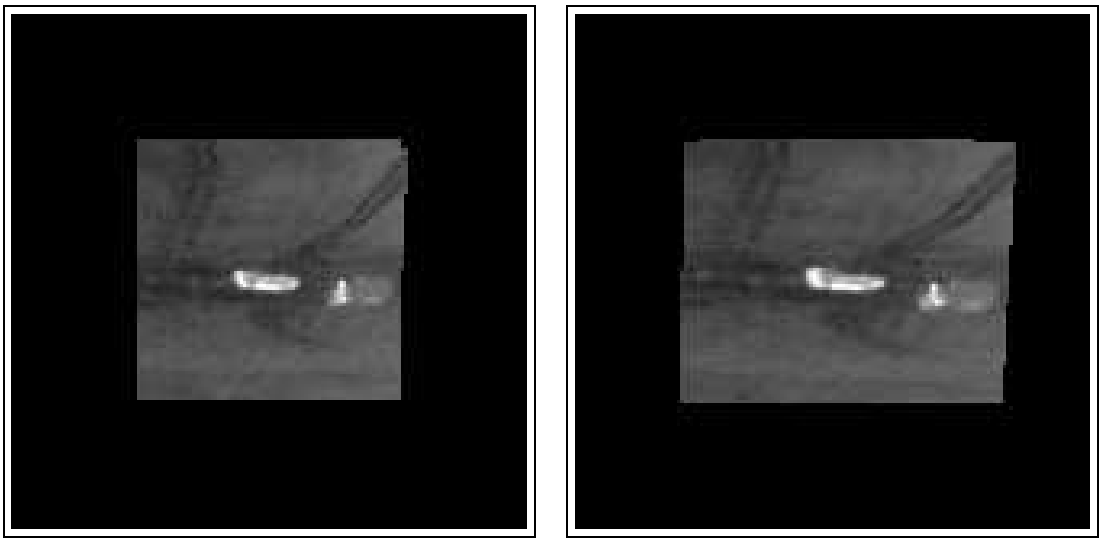
(A) (B)

Figure 6.43: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 73 of the FLIR ATR Sequence.



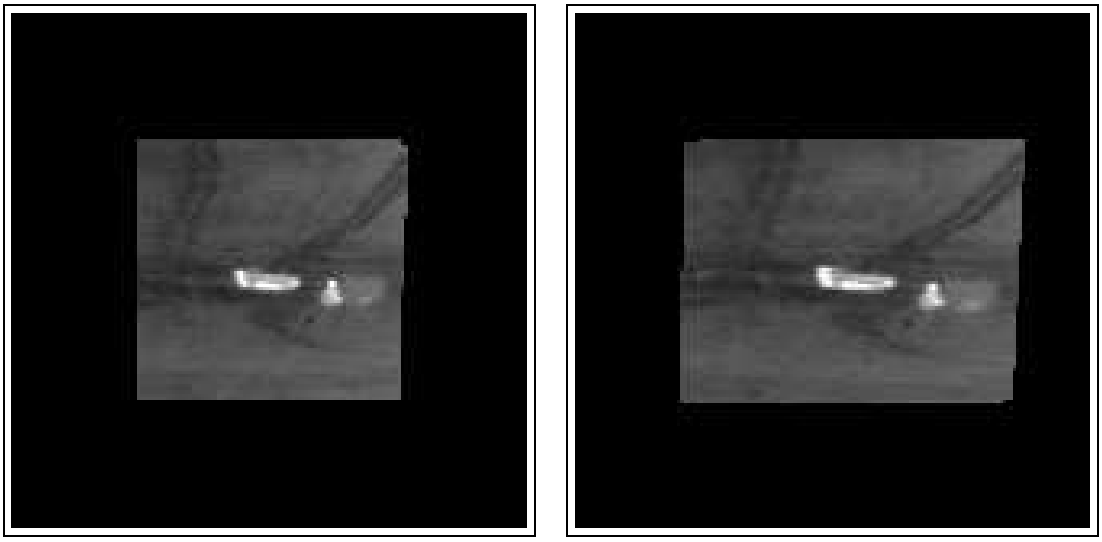
(A) (B)

Figure 6.44: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 74 of the FLIR ATR Sequence.



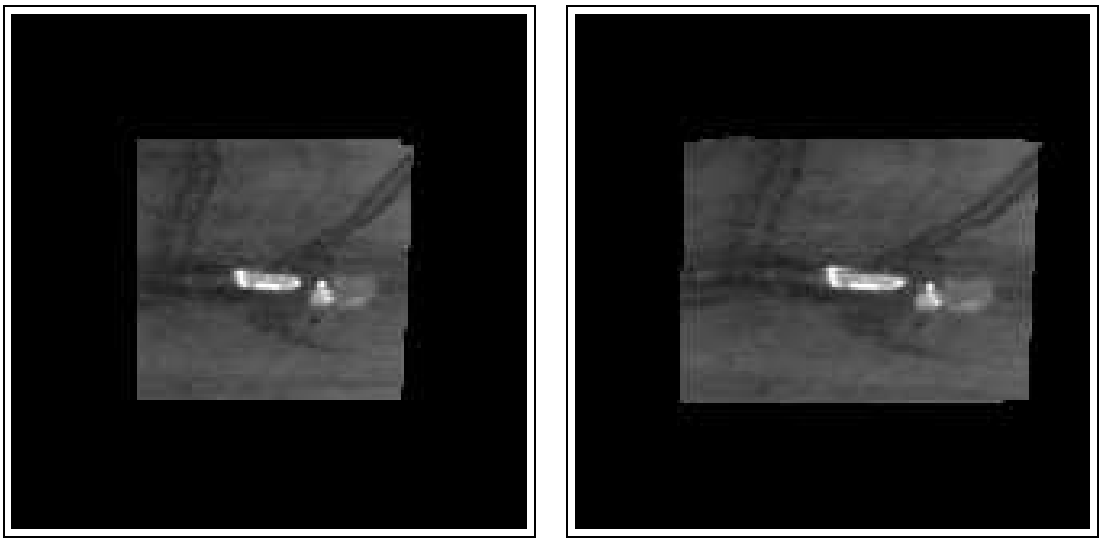
(A) (B)

Figure 6.45: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 75 of the FLIR ATR Sequence.



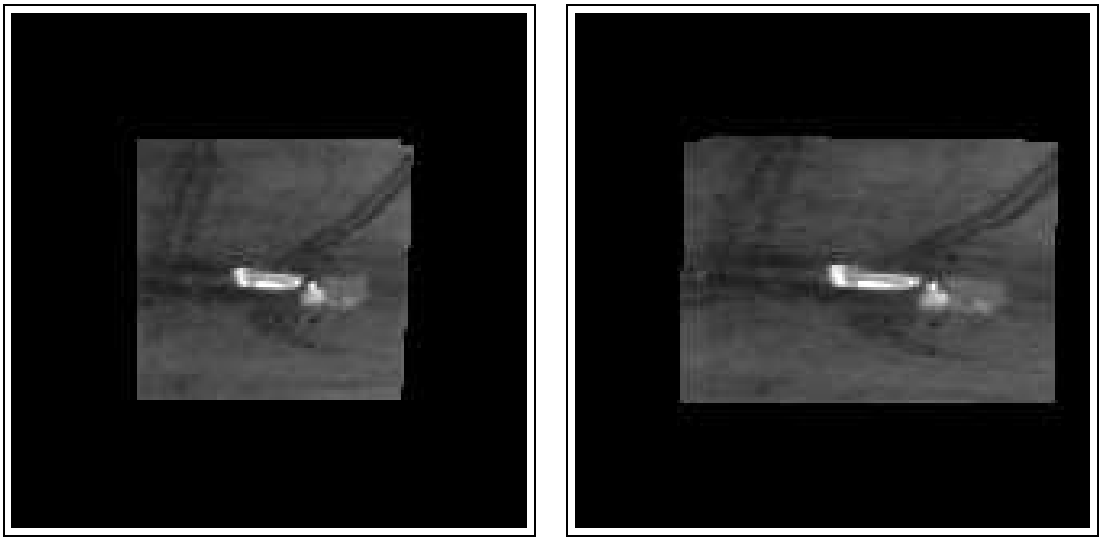
(A) (B)

Figure 6.46: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 76 of the FLIR ATR Sequence.



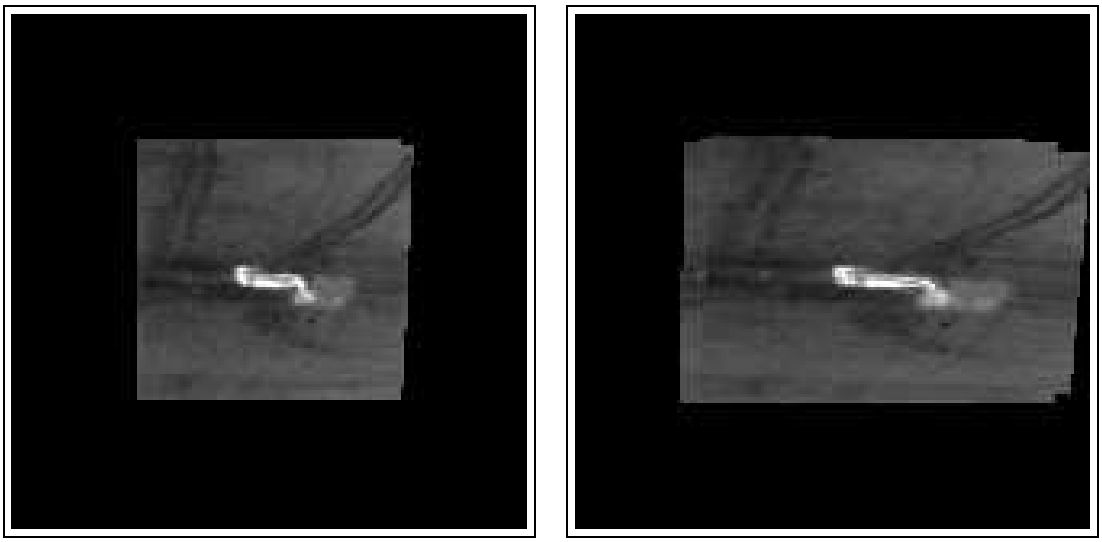
(A) (B)

Figure 6.47: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 77 of the FLIR ATR Sequence.



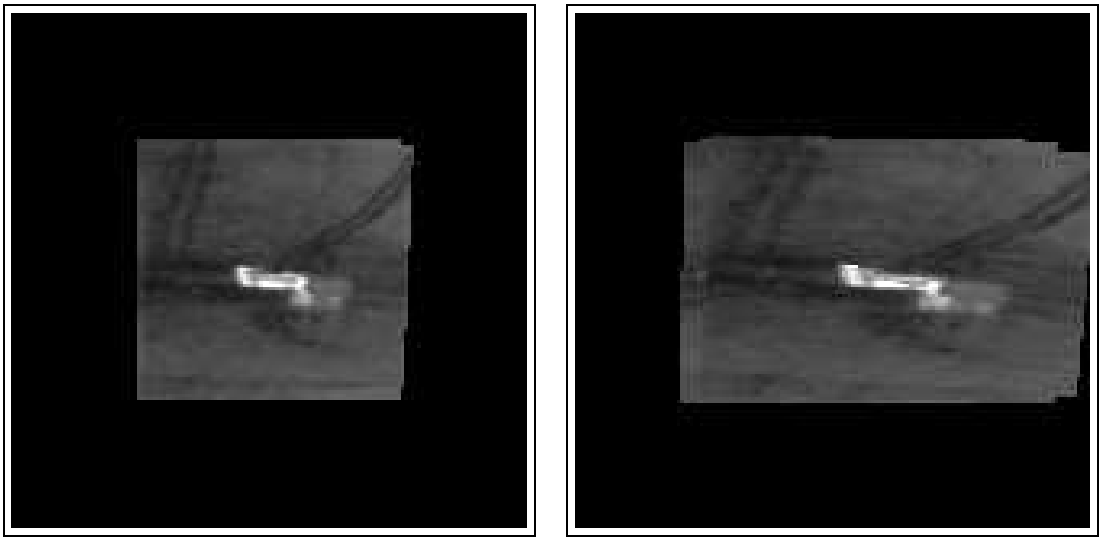
(A) (B)

Figure 6.48: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 78 of the FLIR ATR Sequence.



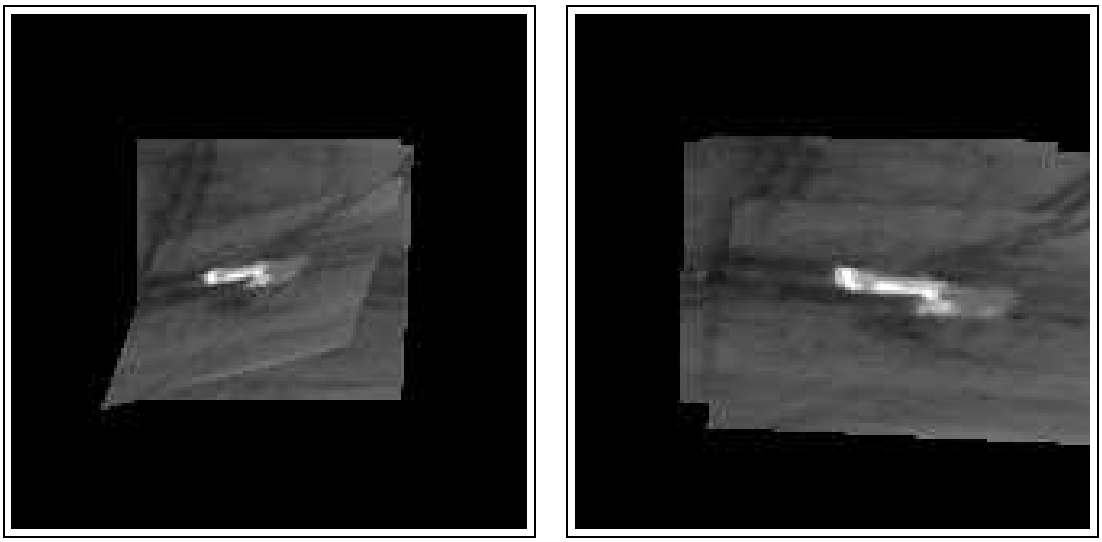
(A) (B)

Figure 6.49: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 79 of the FLIR ATR Sequence.



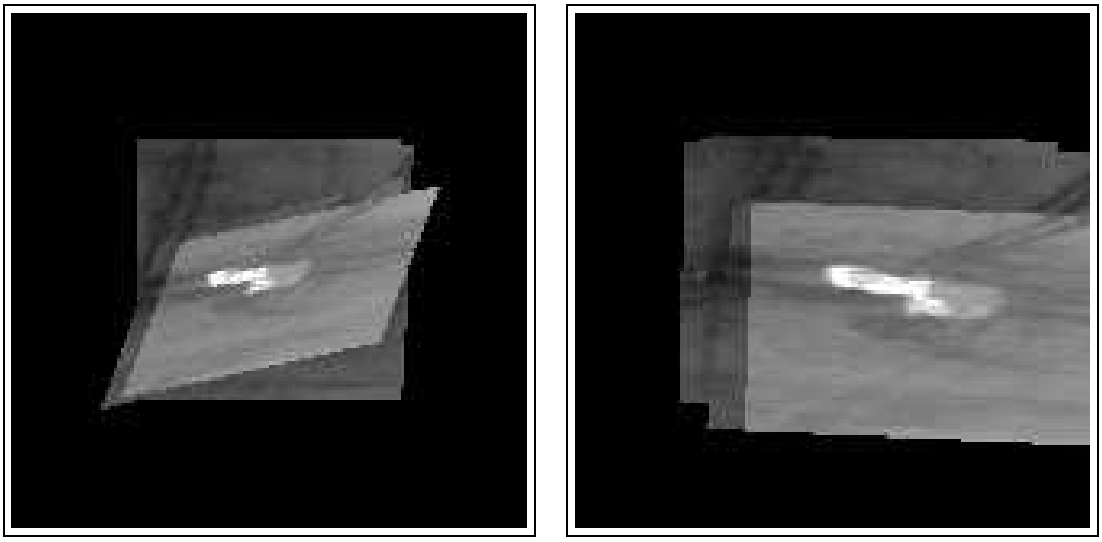
(A) (B)

Figure 6.50: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 80 of the FLIR ATR Sequence.



(A) (B)

Figure 6.51: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 81 of the FLIR ATR Sequence.



(A) (B)

Figure 6.52: Mosaiced image with stabilized background (A) and image stabilized in reference to object (B) after processing frames 72 to 82 of the FLIR ATR Sequence.

Chapter 7

CONCLUSION

7.1 Summary

In this thesis we have presented a new algorithm for estimation of two affine motions through the use of a Bayesian formulation in the image sequences of moving objects observed with a moving camera. We have addressed the problems of optical flow computation, motion segmentation and estimation within a new probabilistic framework. The new system assumes very little knowledge about the presented scene. It neither requires the object and background to be separated by contour lines, nor that the object is one compact region. Hence fragmented multiple motions pose no problem. Results and applications of our algorithm to synthetic and real-life image sequences have been demonstrated.

7.2 Future Work

Our proposed algorithm provides a Bayesian framework which allows for an easy change of the motion model. The currently used affine model can be replaced by another model, such as the planar or projective model [27]. To implement another model, only a parameter estimator routine is needed. This parameter estimator can then be used instead of the estimator described in section 3.2. Moreover, knowledge about the probability distributions of the

optical flow vectors can be used by replacing the current Gaussian family with another distribution family [51]. Another direction of future work could be the incorporation of multiple motion models. Based on an error measure from the Bayesian approach, the algorithm then could pick the appropriate model for a presented sequence.

While the previous proposals for future improvements addressed the motion analysis stage (chapter 3), which is our original contribution, the optical flow module (chapter 2) can also be modified. If a real-time system is desired the use of available on-chip optical flow solutions can be considered. On the other hand, instead of computing the optical flow and selecting reliable locations, we could also select features (e.g., with a Moravec operator [52]) and track them [53]. This would provide fewer but more robust velocity vectors to the motion estimation stage. However, the effect on motion estimation performance needs to be investigated.

It may also be rewarding to apply a robust spatio-temporal filter to the parameters in order to be able to deal with occlusion or estimation errors over multiple frames. A different approach for future extensions could be the integration of other sensor-data. Global Positioning System (GPS) information could be used in combination with our algorithm to build an active vision system able to track the independently moving object. Another idea would be to use the obtained motion information to predict the behavior of camera and object or predict future image frames.

Appendix A

DESCRIPTION OF SOFTWARE

The proposed algorithm was implemented by the author of this thesis in ANSI C++ and can be run portably on any UNIX platform. It requires the TIFF image format libraries and a C++ compiler. Both are easily available for almost all systems. The system was tested under AIX, Solaris and Linux. The processing is split into two executable modules, namely `of` and `amas`. The program `of` does the optical flow computation with confidence values as described in chapter 2 from a pair of `.tif`-images and writes out proprietary `.of`-files for future processing. These files are read by `amas` which conducts the motion analysis according to chapter 3. The `amas` program writes out TIFF images illustrating the results as well as the numeric estimated parameter values. The following two sections give details on usage, command-line parameters and options of the implemented programs.

A.1 Optical Flow Computation – `of`

The program `of` computes the optical flow and confidence values as described in chapter 2. The image sequence has to be available as TIFF images with filenames `filename-stub##.tif`. Here `##` is the two digit number of the frame. This filename-stub must be given to `of` as the command-line parameter. Upon unsuccessful execution a short help text is printed. The following options

can be set when calling `of`:

- `-w #` sets the diameter of the searchwindow in pixels. This parameter determines the maximum displacement that the program could theoretically recover. The maximum detectable displacement can be computed as $\frac{\text{windowsize}-1}{2} \cdot 2^{\text{hierarchylevels}-1}$. The number of hierarchy levels is determined by the image size in pixel. `of` will create another hierarchy level by halving the image size in the x- and y-directions until the image size is smaller than 32x32. Please note that computation-time grows quadratic with the window size parameter. The default value for the windowsize is 7.

- `-t #` sets the diameter of the template in pixels. This parameter sets the size of the sub-images or blocks that are compared to estimate the displacement vectors. Increasing this parameter affects the computation-time with a quadratic growth. Moreover increasing this parameter results in a stronger rigidity assumption. The default value for this parameter is 9.

- `-s #` sets the number of the first frame the program will process. The default value for the first frame is 0 and cannot exceed 99 due to a filename convention implying that the frame number has two digits.

- `-e #` sets the number of the last frame the program will process. The default value for the last frame is set to the filename convention based maximum of 99.

- `-b #` sets the number of pixels that are cut off on all four borders of each image of the sequence before processing. This is useful for images that have distortions close to the border. Low-quality visual cameras or FLIR cameras eventually have noisy or erratic gray-levels at the beginning and end of a scan-line. With this parameter these pixels can be excluded so the results are not corrupted. The default value for this parameter is 0, assuming a high-quality camera with no distortions at the border.

- `-n` makes the program process the data only at the original resolution. No resolution hierarchy is employed. Only the original resolution is considered and the maximum detectable displacement hence is $\frac{\text{windowsize}-1}{2}$.

A.2 Motion Estimation – `amas`

The `amas` program expects the completion of `of`'s execution and follows the same call-syntax and filename-conventions as `of`. It also expects the filename-stub as the command-line parameter. While `amas` is processing it sends current numeric results on motion estimation to `stdout` and messages on computation status to `stderr`. If not called correctly, it prints out a short help text. Options using the same character as options in `of` have the same effect. These options are `w`, `t`, `s`, `e`. The options `b` and `n` are not used by `amas`. The following additional options can be used:

- `-m #` sets the mode of how `amas` selects which motion class represents foreground and which background (see section 4.2). Mode 1 is the default and means `amas` will pick the larger motion class to be background in the first frame and will not enforce this in the rest of the sequence. This

means that that class label will be based on motion parameter similarity in subsequent frames. Mode 2 is like mode 1, but `amas` will enforce the larger motion class to be background in every frame of the sequence. Modes -1 and -2 correspond to modes 1 and 2, respectively. Whenever using a negative mode `amas` will exchange background and foreground as compared to the corresponding positive mode.

- `-p #` sets the percentage of pixels that should be used for the motion estimation. `amas` will automatically select the n most reliable pixels, where n is the given percentage of the total number of pixels per frame. By default `amas` will select ten per cent (.1) of the total number of image pixels.

- `-f` makes `amas` write out all images generated during processing.

- `-x` makes the program use the entire image. No margins are cut off to provide sufficient pixel environment for all processed locations. Thus erratic border locations may be included into the estimation. This is only recommended if the objects are often close to the image margins and motions are rather small.

- `-c` makes `amas` use the translational motion model (equation A.1) instead of the affine motion model (equation 3.1). The translational motion model is the simplest motion model and a special case of the affine motion model for $\theta_1 = \theta_2 = \theta_4 = \theta_5 = 0$. It is also called constant motion model because motion vectors ares not dependent on their image location. Using this feature is recommended wherever a constant mo-

tion model is appropriate, for example, when processing sequences from a pan-tilt camera.

$$\begin{pmatrix} u_r \\ v_r \end{pmatrix} = \begin{pmatrix} \theta_{i,3} \\ \theta_{i,6} \end{pmatrix} \tag{A.1}$$

Some parameters for `of` and `amas`, which in general do not need modification, can be changed by updating the `defines.h` file and rebuilding the executables.

This thesis, animated graphics of the discussed examples, and recent advances of the author's research are available on the Internet at the author's web-site `http://rhine.ece.utexas.edu/~strehl`.

BIBLIOGRAPHY

[1] B. A. Wandell, *Foundations of Vision.* Sinauer Associates, Inc., 1995.

[2] E. H. Adelson, "Mechanisms for motion perception," *Optics & Photonics News*, vol. 2, pp. 24–30, November 1991.

[3] W. S. Geisler and M. S. Blanks, *Handbook of Optics, Vol. 1: Fundamentals, Techniques, & Design.* New York: McGraw-Hill, 1995.

[4] D. G. Albrecht and W. S. Geisler, "Motion selectivity and the contrast-response function of simple cells in the visual cortex," *Visual Neuroscience*, vol. 7, pp. 531–546, 1991.

[5] J. K. Aggarwal, "Motion and time-varying imagery — an overview," in *Workshop on Motion: Representation and Analysis (Charleston, SC, May 7–9, 1986)*, IEEE Publ. 86CH2322-6, pp. 1–6, 1986.

[6] J. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images: A review," *Proceedings IEEE*, vol. 76, pp. 917–935, August 1989.

[7] H. Niemann, *Pattern Analysis.* Berlin: Springer, 1981.

[8] A. Mitiche, *Computational Analysis of Visual Motion.* Plenum Press, 1994.

[9] A. Movshon, "Visual processing of moving images," in *Images and Understanding*, pp. 122–137, Cambridge University Press, 1990.

[10] S. Ullman, *The Interpretation of Visual Motion*. MIT Press, Cambridge, 1979.

[11] R. Kasturi and R. C. Jain, *Computer Vision: V.1. Principles*. IEEE Press, 1991.

[12] R. Kasturi and R. C. Jain, *Computer Vision: V.2. Advances and Applications*. IEEE Press, 1991.

[13] S. Yalamanchili, W. N. Martin, and J. K. Aggarwal, "Extraction of moving object descriptions via differencing," *Computer Graphics and Image Processing*, vol. 18, pp. 188–201, February 1982.

[14] R. Jain, W. N. Martin, and J. K. Aggarwal, "Segmentation through the detection of changes due to motion," *Computer Graphics and Image Processing*, vol. 11, pp. 13–34, September 1979.

[15] S. Shah and J. K. Aggarwal, "Modeling structured environments using robot vision," in *Proceedings of 1995 Asian Conference on Computer Vision, Singapore*, pp. 297–304, December 1995.

[16] C. Fermuller and Y. Aloimonos, "Qualitative egomotion," *International Journal of Computer Vision*, vol. 15, pp. 7–29, June 1995.

[17] D. J. Heeger and A. D. Jepson, "Subspace methods for recovering rigid motion: Algorithm and implementation," *International Journal of Computer Vision*, vol. 7, no. 2, pp. 95–117, 1992.

[18] J. K. Aggarwal and A. Mitiche, "Structure and motion from images: Fact and fiction," in *Proceedings of the Third Workshop on Computer Vision: Representation and Control*, pp. 127–128, October 1985.

[19] T. Y. Tian, C. Tomasi, and D. J. Heeger, "Comparison of approaches to egomotion computation," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 315–320, 1996.

[20] W. Burger and B. Bhanu, "Estimating 3-D egomotion from perspective image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 1040–1058, November 1990.

[21] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding*, vol. 63, pp. 75–104, January 1996.

[22] W. J. MacLean, A. D. Jepson, and R. C. Frecker, "Recovery of egomotion and segmentation of independent object motion using the EM algorithm," in *Proceedings of the 5th British Machine Vision Conference*, pp. 13–16, 1994.

[23] P. J. Burt, J. R. Bergen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and H. Shvaytser, "Object tracking with a moving camera," in *Proceedings, Workshop on Visual Motion, (Irvine, CA, March 20–22, 1989)*, pp. 2–12, 1989.

[24] S. Tsuji, Y. Yagi, and M. Asada, "Finding of objects moving in a pathway by a moving observer," in *Proceedings, Eighth International Conference*

on Pattern Recognition (Paris, France, October 27–31, 1986), IEEE Publ. 86CH2342-4, pp. 1103–1106, 1986.

[25] M. Irani, B. Rousso, and S. Peleg, "Detecting and tracking multiple moving objects using temporal integration," in *Proceedings European Conference on Computer Vision, Berlin, Germany* (G. Sandini, ed.), vol. 588 of *LNCS*, pp. 282–290, Springer, May 1992.

[26] D. Nair and J. Aggarwal, "Detecting unexpected moving obstacles that appear in the path of a navigating robot," in *Proceedings of the First IEEE Conference on Image Processing, Austin, TX*, vol. 2, pp. 311–315, November 1994.

[27] L. Wixson, J. Eledath, M. Hansen, R. Mandelbaum, and D. Mishra, "Image alignment for precise camera fixation and aim," in *Proceedings IEEE Computer Society Conference of Computer Vision and Pattern Recognition*, pp. 594–600, 1998.

[28] M. Irani, B. Rousso, and S. Peleg, "Recovery of ego-motion using image stabilization," in *International Conference on Computer Vision and Pattern Recognition*, pp. 454–460, March 1994.

[29] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P.Burt, "Real-time scene stabilization and mosaic construction," in *Image Understanding Workshop*, pp. 457–465, 1994.

[30] H. S. Sawhney, S. Ayer, and M. Gorkani, "Mosaic based 2D&3D dominant motion estimation for mosaicing and video representation," in *Inter-*

national Conference on Computer Vision (E. Grimson, ed.), pp. 583–590, IEEE, June 1995.

[31] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *Proceedings IEEE Computer Society Conference of Computer Vision and Pattern Recognition*, pp. 232–237, 1998.

[32] W. Kasprzak and H. Niemann, "Visual motion estimation from image contour tracking," in *Computer Analysis of Images and Patterns, 5th International Conference* (D. Chetverikov and W. Kropatsch, eds.), pp. 363–370, Springer, 1993.

[33] M. Irani, B. Rousso, and S. Peleg, "Computing occluding and transparent motions," *International Journal of Computer Vision*, vol. 12, pp. 5–16, January 1994.

[34] G. Adiv, "Determining three-dimensional motion and structure from optical flow generated by several moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, no. 4, pp. 384–400, 1985.

[35] J. Y. A. Wang and E. H. Adelson, "Spatio-temporal segmentation of video data," in *Proceedings of SPIE on Image and Video Processing II, vol 2182, San Jose*, pp. 120–131, February 1994.

[36] C. Morimoto, D. Dementhon, L. Davis, R. Chellappa, and R. Nelson, "Detection of independently moving objects in passive video," in *Proceedings of Intelligent Vehicles Workshop, Detroit, MI*, pp. 270–275, September 1995.

[37] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1972.

[38] B. K. P. Horn and B. G. Schunk, "Determing optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[39] J. Barron, D. Fleet, S. Beauchemin, and T. Burkitt, "Performance of optical flow techniques," in *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Champaign, IL*, pp. 236–242, June 1992.

[40] A. Singh, *Optic Flow Computation: A Unified Perspective.* IEEE Press, 1990.

[41] B. K. P. Horn and B. Schunck, "Determining optical flow: A retrospective," *Artificial Intelligence*, vol. 59, pp. 81–87, February 1993.

[42] H. Niemann, J. Arnold, and G. Sagerer, "On the accuracy of optical flow computation using global optimization," in *Ninth International Conference on Pattern Recognition (Rome, Italy, November 14–17, 1988)*, (Washington, DC), pp. 1094–1096, Computer Society Press, 1988.

[43] Q. Wu, "A correlation-relaxation-labeling framework for computing optical flow – template matching from a new perspective," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 843–853, September 1995.

[44] E. H. Adelson, E. P. Simoncelli, and W. T. Freeman, *Representations of*

Vision, ch. Pyramids and Multiscale Representations, pp. 3–16. Cambridge University Press, 1991.

[45] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani, "Hierarchical model–based motion estimation," in *Proceedings European Conference on Computer Vision, Berlin, Germany* (G. Sandini, ed.), vol. 588 of *LNCS*, pp. 237–252, Springer, May 1992.

[46] R. Battiti, E. Amaldi, and C. Koch, "Computing optical flow across multiple scales: an adaptive coarse-to-fine strategy.," *International Journal of Computer Vision*, vol. 6, no. 2, pp. 133–145, 1991.

[47] S. J. Roan, J. K. Aggarwal, and W. N. Martin, "Multiple resolution imagery and texture analysis," *Pattern Recognition*, vol. 20, pp. 17–31, 1987.

[48] P. Meer, S.-N. Jiang, E. S. Baugher, and A. Rosenfeld, "Robustness of image pyramids under structural perturbations," *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 307–331, December 1988.

[49] J. Hornegger and H. Niemann, "Optimization problems in statistical object recognition," *Lecture Notes in Computer Science*, vol. 1223, pp. 311–326, 1997.

[50] B. Sabata and J. K. Aggarwal, "Surface correspondence and motion computation from a pair of range images," *Computer Vision and Image Understanding*, vol. 63, pp. 232–250, March 1996.

[51] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger, "Probability distributions of optical flow," in *Proceedings of IEEE Conference on Computer*

Vision and Pattern Recognition, Mauii, Hawaii, pp. 310–315, June 1991.

[52] H. P. Moravec, "Towards automatic visual obstacle avoidance," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence, Cambridge, MA* (R. Reddy, ed.), pp. 584–584, William Kaufmann, August 1977.

[53] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA*, pp. 593–600, IEEE Computer Society Press, June 1994.