Poznań University of Technology
Faculty of Electrical Engineering
Institute of Electronics and Telekommunications
ul. Piotrowo 3A, 60-965 Poznań, Poland

Piotr Steć

Unassisted Colour Video Segmentation Using Fast Marching Method

Doctoral Dissertation

Advisor: Professor Marek Domański

Poznań 2004

Politechnika Poznańska
Wydział Elektryczny
Instytut Elektroniki i Telekomunikacji
ul. Piotrowo 3A, 60-965 Poznań

Piotr Steć

Automatyczna segmentacja barwnych sekwencji wizyjnych z wykorzystaniem metody szybkiej propagacji

Rozprawa Doktorska
Przedłożona Radzie Wydziału Elektrycznego
Politechniki Poznańskiej

Promotor: prof. dr hab. inż. Marek Domański

Poznań 2004

Contents

List of Figures

List of Symbols

t	time of contour propagation,
n	frame number from the sequence,
$(x, y) = \mathbf{x}$	spatial pixel coordinates,
I	image,
$I(x, y)$	pixel feature vector at (x, y),
$Y(x, y)$	pixel intensity at (x, y),
κ	contour local curvature,
C	curve in the active contour method,
D	discrete difference,
E	energy of a curve,
F	contour propagation speed,
dI	image difference,
dM	motion difference,
$\vec{\mathcal{N}}$	normal vector,
ϕ	level set surface,
$\mathbb{R}$	set of real numbers,
$\langle . \rangle$	mean operator

Chapter 1

Introduction

1.1 Scope of the Work

Accurate *video segmentation* plays an important role in many applications, such as object-based video coding, video surveillance, interactive multimedia [93] and video editing. Recent international standards such as MPEG-4 [30, 56, 58, 67] and MPEG-7 [57, 81] are specially designed to support these applications. For example, MPEG-4 allows different video objects to be encoded separately and transmitted within separate data streams.

A lack of efficient real-time segmentation techniques is a significant obstacle in widespreading and practical applications of MPEG-4 object-based compression. This object-based approach gives the possibility of manipulating the transmitted objects. A video object can be easily replaced by another one and transmitted using different bandwidths depending on the required picture quality (e.g., less important objects can be transmitted with a lower bitrate). To take advantage of these features, the video sequence must be precisely segmented into objects. MPEG-4 and MPEG-7 do not specify how to extract objects. The only requirement is that the objects must be defined prior to coding or indexing.

In most cases, video sequences of images segmentation is aimed at the extraction of the so-called *semantic objects* (see Figure 1.1) that have meaning for humans (e.g., man, car, table, etc.), but they are very hard to define using low level features. Only such features can be extracted directly from a video sequence.

It makes the process of the extraction of such objects from the highly textured background very complicated.

Another type of video segmentation is *temporal segmentation*. Its purpose is to extract shots limited by cuts and transitions within video sequences. Since it is a different problem than object detection, this work will not deal with this kind of segmentation. An extensive review of temporal segmentation can be found in [71].

Segments extracted from video sequences can be used in the following disciplines of video processing:

- **Video surveillance.** For example, automatic detection of intruders in the video picture.
- **Video transmission.** Here segmentation will permit the compression and transmission of different objects in separate data streams.
- **Editing.** Keying without a special background, automatic rotoscopy, etc.
- **Visual information search.** Automatic indexing of visual databases. Possibility of creating visual queries for such a database.
- **Vehicle navigation.** Automatic obstacle detection, recognizing road signs, trajectory prediction based on the actual traffic condition.
- **Interactive video services.** Distant education systems, on-line shopping.

The desired quality of segmentation and the speed of the process of segmentation depend on the application. For off-line video editing applications, the most important issue is quality, while speed is a secondary matter. On the contrary, in many real-time applications such as vehicle navigation and surveillance, the most important feature is the detection and location of moving objects, while the determination of the exact object boundary is not so important. A method that would be fully automatic, fast and accurate is still an open problem in video segmentation.

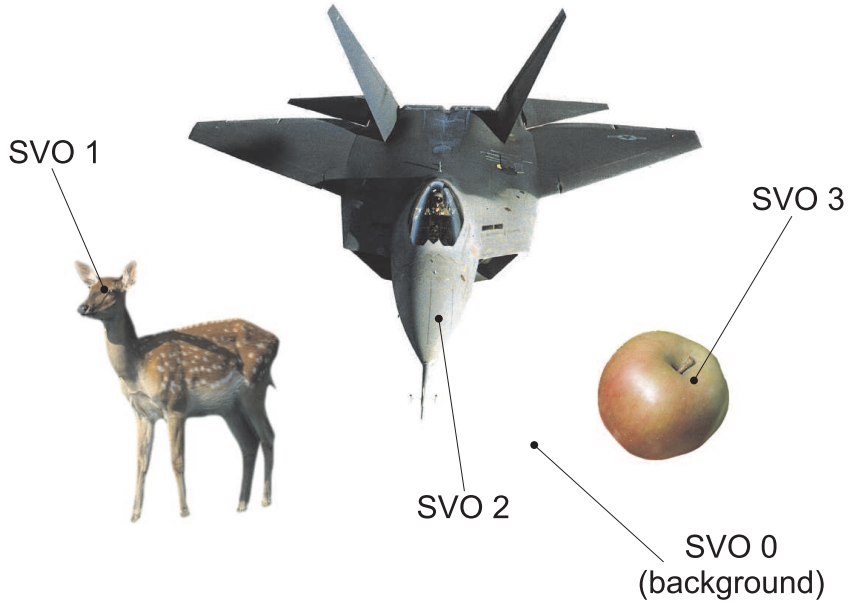


Figure 1.1: Example of sematic visual objects

In specially prepared sequences, for example, those created by chroma-keying, object extraction is relatively simple, so it can be performed fast and accurately. In artificial, computer-generated sequences, there are usually no problems because of the detailed scene information. However, object extraction from real-life sequences is a very complicated task. Segmentation and object tracking are fundamental and yet not well solved problems in computer vision. Many recent papers, e.g., [25, 33, 41, 45, 51, 82, 132, 133] are devoted to this problem and describe more or less accurate methods.

It is possible to detect a semantic object in a video sequence thanks to its motion. However, in practice, it is a very difficult task, because the estimation of a dense and highly accurate motion field is still an unsolved problem.

1.2 Assumptions, Goals and Thesis

The following assumptions are made in this dissertation:

- Segmentation is performed on natural sequences, i.e., sequences that represent the natural world as perceived through a camera, and not created by

computer graphics tools.

- No object-oriented processing is applied (no chroma-keying, etc.).
- Parameters of the camera are unknown.
- Segmented sequences are non-interlaced (progressively scanned).

Sequences made with the use of specially prepared environments such as the blue box are not taken into consideration. Such sequences have been successfully segmented for a long time even by simple analog devices.

The goal of this work is the development of an unsupervised video segmentation algorithm. The algorithm should detect semantic objects that are moving with respect to the background. The complexity of this algorithm should be low enough to allow real-time implementation using processors that will be available within two or three years.

The proposed method is intended for the preparation of the source material for object-based video processing and requires a high quality input, i.e., no compression artifacts, low noise level, etc. For the sake of simplicity, the deliberations are restricted to non-interlaced (progressive) sequences only [138, 153]. Interlaced video is used mostly by television both for broadcasting and surveillance. However, modern TV standards (such as HDTV [110, 153]) offer the possibility of displaying non-interlaced material. It is more compatible with cinema standards and provides better picture quality.

In a video sequence, semantic objects can be detected by motion analysis, thus proper motion estimation is very important in video segmentation. Motion estimation is beyond the scope of this work. This part of video processing is widely used in video segmentation; however, it is a discipline for itself. Moreover, motion estimation algorithms are still under intensive development in many scientific centers all over the world. The speed and accuracy of motion estimation is still being improved. Algorithms developed in this work will exploit the clas-

sical motion estimation methods. Improved motion estimation may replace the algorithms considered in the dissertation.

Active contour methods [17, 65, 72, 97, 140] are often used in the segmentation of static images as well as video sequences (see Chapter 4). Among active contour methods, a relatively new Fast Marching Method [113, 119, 124] (see Section 4.4) is considered in this dissertation. The flexibility of this method allows adapting it to a wide range of problems. Additionally, the Fast Marching Method has computational complexity of class $O(n \log n)$, where n is the total number of points in the computational domain [119]. Low computational cost is very important for applications that require speed. In order to limit the dissertation to a reasonable size, the research was limited to applications of the Fast Marching Method.

Thesis of the dissertation

The Fast Marching Method can be used for real-time unsupervised extraction of semantic objects from frames in colour video sequences with both a static and a moving background.

This thesis will be proved by means of a verified proposal of respective techniques as discussed in Chapters 5 and 6.

1.3 Thesis Outline

Chapter 2 will highlight some general problems of image processing related to video segmentation that are essential to the method developed in this work. This chapter will also present problems directly related to video segmentation. In Section 2.3.3, there will be proposed a procedure for efficiency improvement for the existing motion estimation methods. Additionally, an original method of quality evaluation for video segmentation will be proposed in Section 2.4.3.

Chapters 3 and 4 will give an overview of video segmentation methods. Algorithms developed in this thesis are related to the group of methods presented in Chapter 4.

Chapter 5 will present an original method of foreground-background segmentation based on the Fast Marching Method [119]. The contribution of this chapter includes:

- development of an automatic initialization method,
- adaptation of the fast marching algorithm to the given problem,
- automatic calculation of the stop condition,
- method of segmentation quality improvement.

Chapter 6 will present an original method of the detection of multiple objects in a certain class of video sequences. The method is based on multi-label fast marching [124, 125], yet it contains significant extensions. The contribution of this chapter includes:

- initialization with any number of labels (the number of labels is unknown prior to segmentation),
- simplification of propagation speed, which makes the algorithm more efficient,
- dynamic motion vectors regularization,
- solution of the problem of unidirectional contour propagation.

Chapter 7 will contain some concluding remarks as well as directions of further research.

Chapter 2

Selected Problems of Video Segmentation

2.1 General Problems of Video Segmentation

The most important task in video segmentation is to extract semantic objects. The problem is that there is no exact definition of the semantic object. Generally, it is defined as an object that has meaning for humans. However, this definition is very context dependent. For example, one person can consider a car as one object while another person can distinguish the wheels, the body of a car and the windows as separate objects. It is easy to notice that semantic decomposition of the scene can be done on different scales.

The high-level definition of the semantic object is ambiguous. It is hard to find a combination of low-level features that will clearly define such objects. Usually, the low-level feature that is used for the detection of the semantic object is its motion. Leaving out problems with motion estimation and considering the perfect and dense motion field it is not easy to find semantic objects. Objects with complex motion are difficult to segment. For example, walking people are such a difficult case, because during the walk one leg is always static against the surface while other parts of the body are moving at different speeds and in different directions. The objects that are not moving are undetectable using this approach.

It has to be mentioned that the term *motion* used here is related to movement in the image plane of the video, and not necessarily to real displacements of objects. In some cases it is possible to detect objects that are not moving in a real world. When a static object is seen against a static background that is far behind the object and the camera is in motion, there can be measured the so-called *apparent motion* caused by the parallax effect. Nevertheless, the moving camera causes a lot of problems with object detection because, as will be shown later, most segmentation algorithms are based on frame to frame differences. As a consequence, they require a static or a motion-compensated background.

Video segmentation can be performed only within one shot. After a cut or during a transition, the differences between consecutive frames are too big for motion estimation and segmentation. Then the segmentation of a material already cut is considered, the borders between the takes have to be detected first. Such a process is called *temporal segmentation* [9, 35, 71, 104] and is considered as a separate problem in video processing.

Overlapping objects that appear in video sequences are a problem similar to the moving background. In some parts of the sequence one object can be a background for another. In this case, it is hard to detect the border between them using the difference between the frames. It is a big problem for methods that consider a stationary background.

Another problem in video segmentation are reflective and transparent objects. Since the interior of such objects is constantly changing while the objects are moving, it is very hard to estimate the motion properly for such objects and, as a consequence, to detect such objects in a scene. A particulary vivid example of such an object is water, where complex visual properties exist, such as reflections and complex motion.

Noisy video sequences may be very hard to segment, not to mention viewer perception issues. Of course, it depends on the amount of noise that the sequence contains. The more noise, the more problems with segmentation. Since noise

distribution is different in every frame, detecting objects by the frame difference is difficult. Also, the methods of dense motion field estimation [53, 74, 76] have problems in case of noise. The application of noise-reduction filters [1,90,106,136] can help; nevertheless, this kind of filters usually introduce blurring and can be time-consuming in more advanced implementations.

Despite some general problems with using motion as a feature that makes object detection possible, motion estimation is a very complex task itself. This subject is presented in Section 2.3.

2.2 Colour Models

2.2.1 RGB Colour Space

This is the most popular colour space in image processing. It consists of three chromatic components: **R**ed, **G**reen and **B**lue [34,112]. The RGB colour space is orthogonal and the colour gamut of this space forms a cube. In the center of the coordinate system there is a black point, while the point defined by the maximum values of the red, green and blue components is white. On the line defined by the maximum values of the components and the center of the coordinate system there are all achromatic colours.

This colour space is most popular in digital representation of images (storage) and in display devices. Also, a large number of acquisition devices use this model.

Despite its popularity, the RGB colour space has some disadvantages:

- high correlation of components, which makes the RGB model unsuitable for compression;
- it is hard to intuitively describe the colour using the R, G, B components;
- perceived differences in colour are not equivalent to the distances in colour space (non-uniformity).

This model is not very suitable for segmentation applications because of its

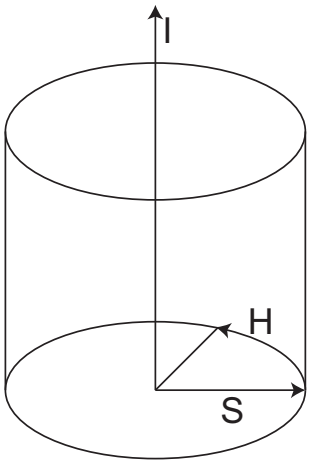


Figure 2.1: Cylindrical IHS colour space

sensitivity to illumination changes. Handling shadows or scenes with changing lighting is very hard using the RGB colour model.

2.2.2 Perceptual Colour Spaces

Human eye contains receptors of the red, green and blue colours. Despite this fact, it is very hard for an untrained person to describe colour in the RGB space. It is much more natural to describe colour in the sense of its intensity (brightness, lightness), hue and saturation [34, 112]. Hue represents the dominant wavelength in the stimuli. Intensity is related to the energy of the light while saturation is related to the differences in amplitude between the dominant wavelength and the remaining part of the spectrum. The maximum of saturation gives pure chromatic light whilst zero saturation gives gray levels dependent on intensity. Zero intensity gives black and the maximum of the intensity gives white.

The separation of luminance and chrominance gives the possibility of handling easily illumination changes and shadows in segmentation applications [36, 64, 68, 148].

A typical perceptual colour space is the IHS[1] cylindrical space (Figure 2.1). In this space, **I**ntensity is represented on the cylinder axis, **S**aturation is a distance

[1]Variations of this space are the HSV and HLS colour spaces.

from the axis, while **H**ue is an angle of rotation from the point of zero hue.

Because of the complex transformation from the orthogonal RGB space to the cylindrical colour space, there can be met different forms of representing H, which are often a compromise between the accuracy of representation and the simplicity of calculation. An exact calculation of H with the given RGB components can be done according to the following formula [48, 139]:

$$H = \cos^{-1} \frac{0.5((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}}, \tag{2.1}$$

where if $B > G$, then $H = 360 - H$.

These colour models have some disadvantages:

- conversion from other models is not straightforward (there are many conversion schemes used) and is computationally expensive,
- for low saturation values, colour information becomes less reliable,
- computation of differences is inconvenient because of the angular character of the hue component (extreme values of this component are perceptually almost the same).

2.2.3 Television Transmission Colour Spaces

These colour spaces were designed to minimize the bandwidth required for image transmission. In these models, chrominance is defined as differences between chromaticity components and luminance, hence this type of colour space is called the *opponent colour space* [34, 112]. Because of a very low correlation between the components, opponent colour spaces are widely used in image compression.

The YUV colour space is the basis of the PAL TV signal. The components of this space are defined as follows:

$$\begin{aligned} Y &= 0.177R + 0.812G + 0.011B \\ U &= -0.147R - 0.289G + 0.437B = 0.493(B-Y) \\ V &= 0.615R - 0.515G - 0.100B = 0.877(R-Y). \end{aligned} \tag{2.2}$$

The YIQ colour space is used in the NTSC TV coding standard. The luminance component Y is identical as in the YUV (Equation 2.2) space. The I component represents the orange-cyan axis and the Q component represents the magenta-green axis. These two components are defined as follows:

$$\begin{aligned} I &= 0.596R - 0.274G - 0.322B = 0.74(R-Y) - 0.27(B-Y) \\ Q &= 0.211R - 0.523G + 0.312B = 0.48(R-Y) + 0.41(B-Y). \end{aligned} \tag{2.3}$$

The YC_bC_r was designed as an alternative to YUV and YIQ, independent of the TV coding system. It is widely used in the digital coding of images. The luminance component Y is identical as in the YUV (Equation 2.2) space, while the chromaticity components are defined as follows:

$$\begin{aligned} C_b &= -0.169R - 0.331G + 0.500B = 0.564(B_Y) \\ C_r &= 0.500R - 0.418G - 0.081B = 0.713(R-Y). \end{aligned} \tag{2.4}$$

The YC_bC_r colour model is very convenient from the segmentation point of view [4, 20, 73, 109]. Firstly, it separates luminance from chrominance, which makes the handling of shadows and illumination changes very natural. Secondly, it does not represent disadvantages of the IHS colour space: conversion to YC_bC_r is straightforward and there is no problem with the calculation of differences. An additional advantage of this colour space is that it is used by video encoders and can be used directly by a segmentation algorithm designed for object-based video coding.

Considering all the advantages described earlier, the YC_bC_r colour space has been chosen as the working colour space for segmentation algorithms developed in this work. Another argument for using this colour space is the fact that most video sequences are available in the YC_bC_r or similar YUV and YIQ spaces.

2.3 Motion Estimation

2.3.1 Importance of Motion Estimation in Video Segmentation

To perform the extraction of moving semantic objects from a colour video sequence, additional information about the consistence of these objects is needed.

For video compression, block matching techniques [96, 138] are mostly used. Those motion estimation algorithms do not provide a dense motion field and are not intended to discover a real motion from a sequence. Their goal is to find the closest match for the present block of pixels in the next frame. The above assumptions make this kind of algorithms unapplicable for video segmentation.

Although motion information is essential for the method considered in this thesis, the development of a new motion estimation algorithm is not intended. There is intensive research in this field being conducted. One of the available algorithms will be used.

2.3.2 Problems with Motion Estimation

Here some problems related to motion estimation are emphasized. A detailed analysis of the problems along with examples of motion estimation algorithms can be found in [96, 138].

Apparent motion

Changes that we observe in video sequences are a result of the projection of a 3-D space onto a 2-D image plane. It is impossible to determine real object motion without implicit knowledge about scene geometry. In natural sequences, such information is unavailable. Such a 2-D projection of a 3-D motion is called *apparent motion*. If a point moves in an image sequence, its motion is ambiguous. This means that the same observed motion can be caused by all points whose initial and final positions lie on the same line that crosses the center of projection (see Figure 2.2). Such a 2-D vector of displacement in the image plane is also

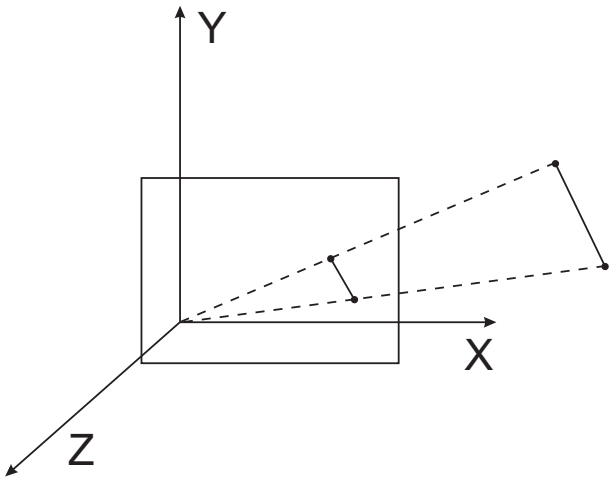


Figure 2.2: Apparent motion

called the *correspondence vector*.

Due to moving objects, pixels in consecutive images from the sequence change its brightness. The rate of this change can be followed in time-space coordinates. The vector that defines this changes is called the *optical flow vector* and is defined as

$$(v_x, v_y) = (dx/dt, dy/dt), \tag{2.5}$$

at a particular point $(x, y, t) \in \mathbf{R}^3$.

Theoretically, the projected motion and the optical flow should be identical. In practice, these two values can be different because of:

- Movement of the smooth regions. When there is no gradient within the moving area, the optical flow cannot be computed. For example, a uniformly colored circle that rotates around its center will not produce any optical flow (see Figure 2.3).

- Changes in illumination. When lighting in the scene varies from frame to frame, the optical flow exists even if there is no physical motion in the scene. The optical flow different from the real object motion can be also generated by shadows or changes in shading caused by the rotated surfaces (see Figure 2.4).

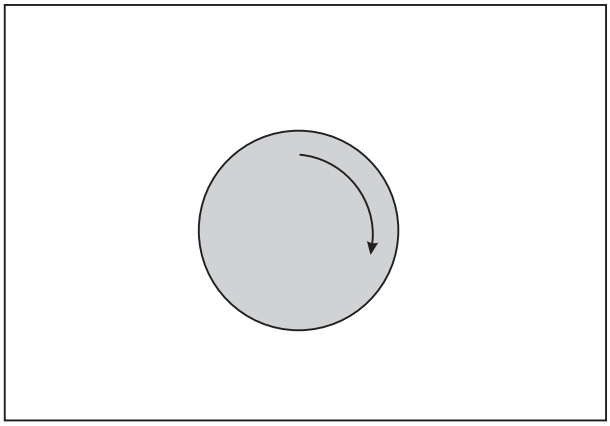


Figure 2.3: Motion that does not produce the optical flow

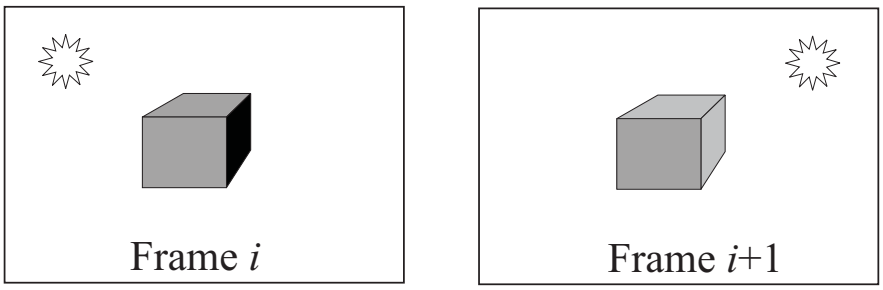


Figure 2.4: Optical flow produced by illumination changes

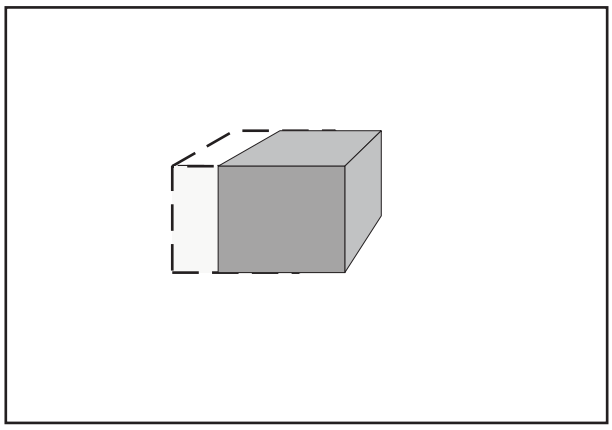


Figure 2.5: Moving object is covering some parts of the background while uncovering others

Occlusion

When an object in the scene moves from frame to frame, it covers some parts of the background that were uncovered in the previous frame whereas some parts of the background remain uncovered (see Figure 2.5). The part of the image that was uncovered has no corresponding region in the previous image from the sequence. This makes the computation of the optical flow impossible. By analogy, a similar problem exists for the covered part of the picture.

Aperture

Most motion estimation methods perform within a certain window. A consequence of that is the fact that the solution to the motion estimation problem is not unique. It is only possible to determine the optical flow orthogonal to the spatial image gradient. This kind of flow is called the *normal flow*. For example, when a smooth square rotated by 45 degrees moves horizontally and its edge is observed by a small window, it is impossible to determine the real direction of the movement. Only the normal flow can be observed (see Figure 2.6).

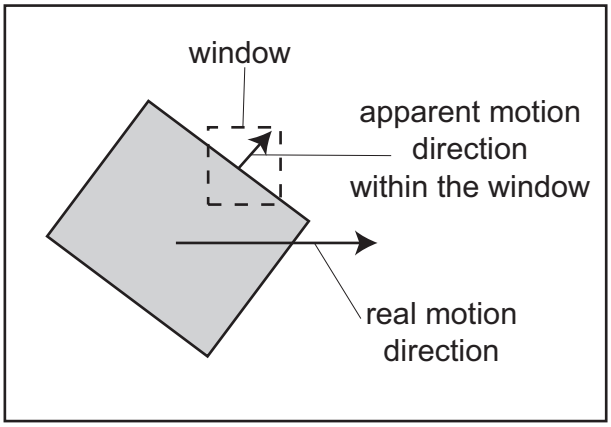


Figure 2.6: Aperture problem

2.3.3 Some Methods of Dense Motion Estimation

The development of a motion estimation method is not the purpose of this work. To avoid implementation problems, two state-of-the-art dense motion field estimation methods were used. These methods include the *Horn-Shunk* method and the *Lucas-Kanade* method. The former was chosen because of its robustness to get stuck in local minima, the latter because of its speed connected with relatively good reliability. Both methods have a very good support in the literature. An additional factor was the performance comparison found in [38], where both methods obtained the highest results.

Horn-Shunk Method

The motion estimation method introduced by Horn and Schunck in [53] is based on the optical flow equation. According to this equation, the intensity $I(x, y, t)$ of the image remains constant along the motion trajectory:

$$\frac{dI(x, y, t)}{dt} = 0. \tag{2.6}$$

According to the chain rule, the above equation can be expressed as follows:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0. \tag{2.7}$$

The values of $\frac{dx}{dt}$ and $\frac{dy}{dt}$ are components of the motion vectors to be estimated and will be denoted by

$$u = \frac{dx}{dt} \quad \text{and} \quad v = \frac{dy}{dt}, \tag{2.8}$$

for horizontal and vertical directions, respectively.

For the sake of clarity, let I_x, I_y, I_t be the image gradients with respect to x, y and t, respectively. Now, Equation 2.7 can be expressed as follows:

$$(I_x, I_y) \cdot (u, v) = -I_t. \tag{2.9}$$

From Equation 2.9 it is possible to determine only the movement component in the brightness gradient direction. In order to calculate the real optical flow, an additional smoothness constraint must be applied. The smoothness constraint is usually formulated in terms of Laplacians:

$$\nabla^2 u = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2, \tag{2.10}$$

$$\nabla^2 v = \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2. \tag{2.11}$$

To calculate the flow vectors, the sum of errors in intensity displacement must be minimized

$$\mathcal{E}_b = I_x u + I_y v + I_t, \tag{2.12}$$

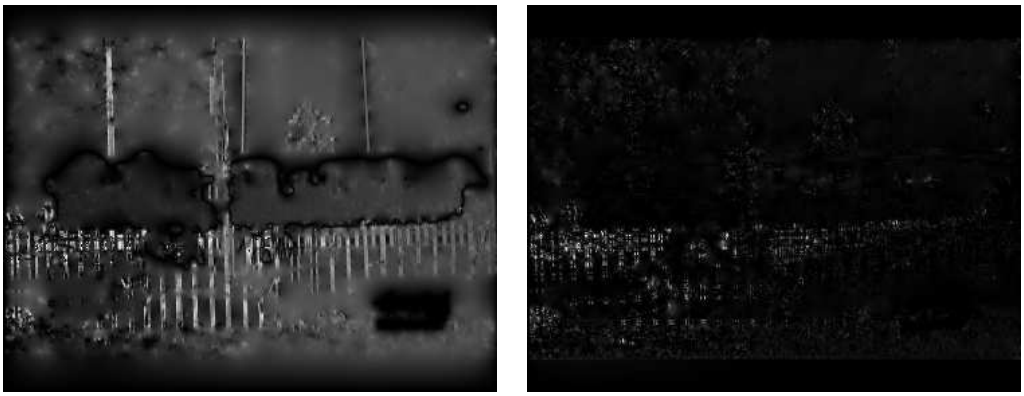
and the smoothness constraint derived from (2.10, 2.11):

$$\mathcal{E}_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2. \tag{2.13}$$

Finally, the total error to be minimized has the following form:

$$\mathcal{E}^2 = \iint_{\Omega} (\mathcal{E}_b^2 + \alpha^2 \mathcal{E}_c^2) dx dy, \tag{2.14}$$

where Ω is the image area and α is the weighting factor that controls smoothing.



(a) (b)

Figure 2.7: Example of motion field magnitudes obtained by the Horn-Schunck algorithm (500 iterations). (a) – horizontal component, and (b) – vertical component

The above problem can be solved iteratively using the Gauss-Seidel method according to the following equations:

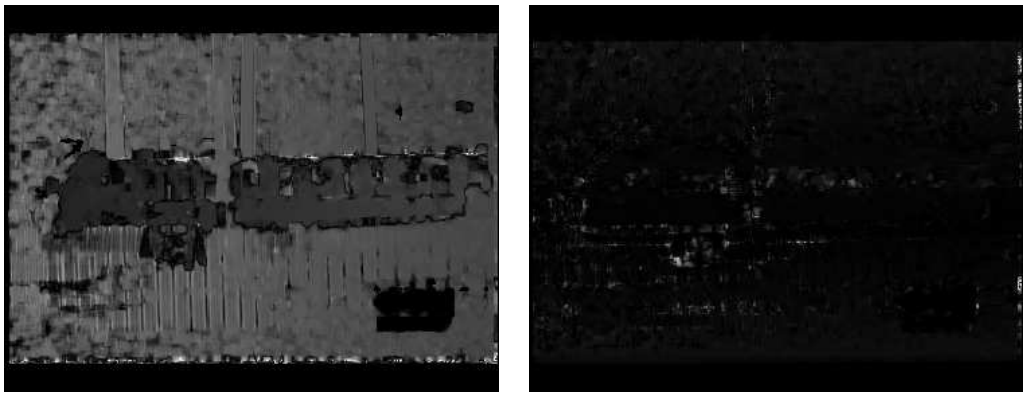
$$u_{n+1} = \bar{u}_n - I_x \frac{I_x \bar{u}_n + I_y \bar{v}_n + I_t}{\alpha^2 + I_x^2 + I^2 y}, \tag{2.15}$$

$$v_{n+1} = \bar{v}_n - I_y \frac{I_x \bar{u}_n + I_y \bar{v}_n + I_t}{\alpha^2 + I_x^2 + I^2 y}, \tag{2.16}$$

where $\bar{u}$ and $\bar{v}$ are the local averages of motion components excluding the present pixel. The higher value of the parameter α, the smother the resulting motion field, and the same edges in the motion field have less correspondence to the underlying object edges. Too small values of α makes the algorithm stuck in the local minima. As the initial guess of (u, v), zero or the estimate from the previous frame at the same pixel is usually chosen. The iterations are stopped when there is no change in the motion field between two consecutive iterations. An example motion field obtained from the 'Bus' sequence is presented in Figure 2.7.

Lucas-Kanade Method

This method was introduced in [76] as an image registration and stereo disparity calculation technique. It was also successfully applied as a motion estimation method [8, 24, 74].



(a) (b)

Figure 2.8: Example of motion field magnitudes obtained by the Lucas-Kanade algorithm (1 iteration). (a) – horizontal component, and (b) – vertical component

The first step of the algorithm is the computation of the spatial and temporal gradients I_x, I_y, I_t over the smoothed version of the images from the sequence. Typically, this is done using a 5×5 convolutional filter.

In the next step, the following linear equation is constructed:

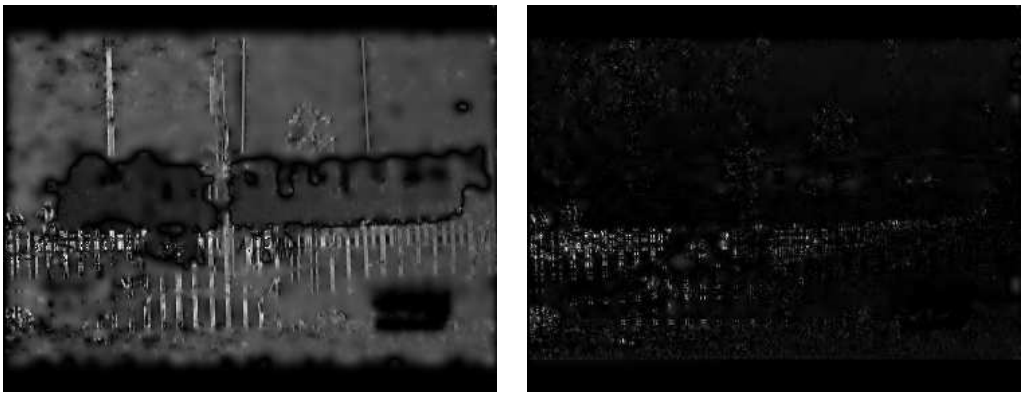
$$\begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = - \begin{bmatrix} \sum_W I_x I_t \\ \sum_W I_y I_t \end{bmatrix}, \tag{2.17}$$

where $W(x, y)$ is a square window with the centre placed at (x, y) with weights assigned to the elements according to the Gaussian function, namely, weights values are higher for central pixels and lower for peripheral pixels.

The solution of this equation with respect to v_x and v_y gives the motion components for the given pixel. An example motion field obtained with the Lucas-Kanade method is shown in Figure 2.8.

The advantages of this method include:

- speed,
- accuracy,
- no iteration required (motion vectors are obtained in one pass of the algorithm).



(a) (b)

Figure 2.9: Example of motion field magnitudes obtained by the Horn-Schunck algorithm with the result of the Lucas-Kanade method used as an initial guess (1+20 iterations). (a) – horizontal component, and (b) – vertical component

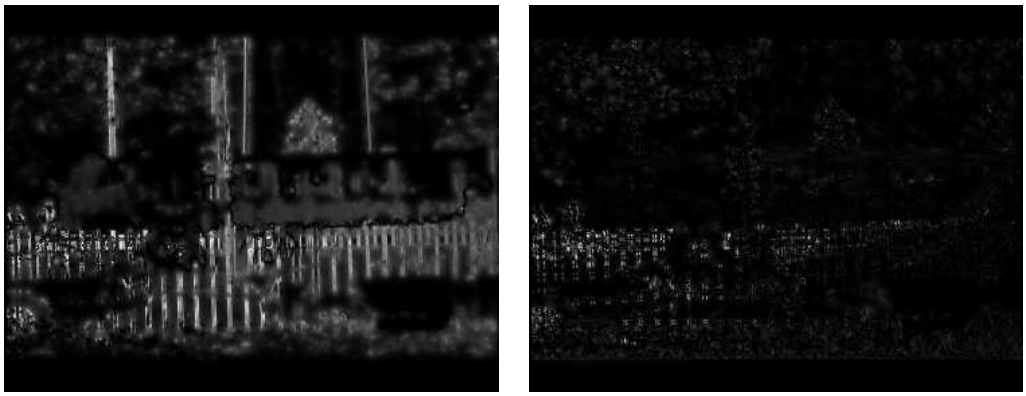
As disadvantages one can consider:

- problems with the solution of the ill-posed versions of Equation 2.17,
- maximum detected displacement dependent on the size of the window W.

Combined Method

To speed up the calculations, a combined method which merges the Horn-Schunck (HS) and Lucas-Kanade (LK) algorithms is proposed in this dissertation. To obtain a smoother motion field, after the LK estimation, several iterations of the HS algorithm are performed with the result from the first step used as an initial guess (Figure 2.9). This gives a significant reduction of computational cost in comparison to the traditional HS method (Figure 2.7) with similar quality. When starting from zero, the HS method after 20 iterations gives a motion field which is far from the real motion present in the sequence (Figure 2.10).

A comparison of the HS method with and without initialization is presented in Table 2.1. The error between the frames was calculated as a mean square of



(a) (b)

Figure 2.10: Example of motion field magnitudes obtained by the Horn-Schunck algorithm after 20 iterations. (a) – horizontal component, and (b) – vertical component

motion compensated frame differences:

$$MSE = \frac{1}{w \cdot h} \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} (I_n(x,y) - I_{n-1}(x+mx, y+my))^2, \qquad (2.18)$$

where w and h are image dimensions, mx my are motion vectors at the current pixel and n is the frame number. In most cases, the initialized method obtained better results in a much shorter time.

2.4 Quality Measures of Segmentation

Segmentation quality assessment is an important part of the video segmentation problem. The problem is that it is very difficult to obtain an objective quality measure that conforms with subjective assessments made by human observers. Quality evaluation is very often omitted in publications. A reliable quality measure is essential for the development of segmentation methods.

2.4.1 Subjective Methods

All subjective test methods are based only on assessments made by viewers. The tests should be performed under controlled viewing conditions. Such viewing conditions are defined in the ITU-T Recommendation P.910 [98] for image and

Table 2.1: Comparison of motion estimation quality for the Horn-Shunk (HS) method started from zero and the HS method initialized by the Lucas-Kanade method (LK-HS). MSE – mean square error between two motion compensated frames

Sequence	**Frame no.**	**Time HS (ms)**	**MSE HS**	**Time LK-HS (ms)**	**MSE LK-HS**
Bus	60	9680	441.1	1324	411.1
Bus	120	9682	238.2	1318	210.5
Bus	240	9692	302.8	1324	298.3
Coast	1	9637	72.5	1332	141.9
Coast	130	9742	67.9	1340	133.8
Coast	264	9667	81.1	1341	148.5
Dance	45	9650	110.3	1308	98.2
Dance	100	9892	233.5	1328	190.9
Dance	160	9810	168.8	1337	138.8
Mobile	50	9677	183.1	1389	219.0
Mobile	130	9893	177.9	1357	228.2
Mobile	270	9791	96.4	1347	168.9
Stefan	30	9643	121.9	1320	111.8
Stefan	50	9740	21.4	1347	20.4
Stefan	215	9745	55.7	1339	54.7

video quality assessment, and can be adapted for segmentation quality assessment as well.

An example of subjective tests applied to video segmentation can be found in [84]. The test sequences were categorized by the number of visible objects, place (indoor/outdoor) and illumination changes. The sequences were viewed using high quality displays in a quiet surrounding according to the ITU-T recommendations.

The evaluation described in [84] was performed by a group of 24 people. Half of this group were experts related to video technology, while the other half included unexperienced users. Since colour was essential in the tests, all groups had to pass the colour blindness test. The testers gave marks from a five point scale on a form containing the criteria shown in Table 2.2.

The testing was performed according to a strict procedure. At the beginning, the tester was watching an original, unsegmented sequence for a maximum of

Table 2.2: Evaluation criteria and performance metrics (from [84])

	Algorithm evaluation criteria	Evaluation Method
1	Segmenting moving semantic objects from the background	Subjective assessment
2	Tracking individual regions throughout the video sequence	Subjective assessment
3	Providing an accurate region or, preferably, object boundaries	Subjective assessment
4	Distinguishing between moving objects and image perturbations (e.g., camera noise, rain)	Subjective assessment
5	Segmenting objects into associated sub-regions	Subjective assessment
6	Eliminating or correctly identifying shadows	Subjective assessment
7	Low computational complexity	Run-time data
8	Low number of configuration parameters	Run-time data
9	Illumination invariance	Post-assessment analysis
10	Segmenting well outdoor sequences	Post-assessment analysis

three minutes. During this time, any part of the sequence could be seen. After that time, the entire segmented sequence was presented once. Then the sequence was divided into three parts. Every part could be seen for five minutes at most and after that time it should be scored, before viewing the next part.

The results of the tests were the subject of statistical analysis. This approach seems to be good since it promotes methods that are better perceived by the viewer. However, such methods have some pitfalls. It is known that a statistical test requires a large number of data to be representative. Moreover, the selection of the testing group could have an impact on the results because of the individual preferences. It is unclear whether the testing group should be chosen randomly or represent the target customers of the tested segmentation method.

Subjective tests have some serious drawbacks:

- they are hard to manage (many people must be engaged),
- they are very time consuming,
- they are expensive.

2.4.2 Objective Methods

There are two groups of such methods:

- **Evaluation without ground truth**

  The techniques estimate segmentation quality by the use of measurements of the object extracted. Such measures can be computed automatically without user interaction.

- **Evaluation with ground truth**

  Reference segmentation is used. The reference for natural sequences can be prepared only by a human operator. The method results in the measure of errors between the reference segmentation and the segmentation assessed.

Evaluation without Ground Truth

Here as an example the method of Erdem *et al.* [31] can be presented. They assume that the colour histogram should remain unchanged inside the object if segmentation is correct. With this in mind, another measure that can be mentioned that characterizes the quality of segmentation is inter-frame *colour histogram differencing*. The histogram is calculated from the differences between the image segments at the frame n and $n-1$. Another approach is to build the histograms of the image segments at the frame n and a smoothed histogram of objects from the images $\{n-i,\ldots,n+i\}$. The histogram at the frame n is denoted by H_n. It is represented as a vector containing concatenated histograms from all colour components. The smoothed version of this histogram denoted as $H_{n,av}$ is constructed as follows:

$$H_{n,av}(j) = med\{H_{n-i}(j),\ldots,H_{n+i}(j)\}, \qquad j=1,\ldots,B, \tag{2.19}$$

where *med* denotes the median and B is the number of bins in the histogram. The calculations consists in measuring the characteristics on the opposite sides of the object contour. For the evaluation purpose, the YCbCr colour space is

used. The points of measure are determined by short lines spaced equally and aligned normally to the boundary of the segment (Figure 2.11). Along these lines,

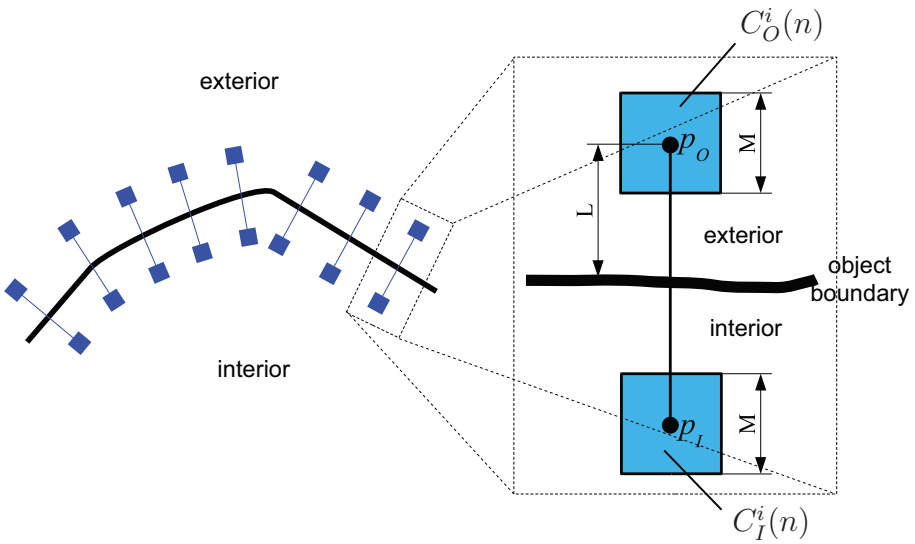


Figure 2.11: Normal lines along the object boundary and the areas on which metrics are computed. p_I – interior point, p_O – point outside the object.

a *colour difference metric* is calculated. The metric is defined as follows:

$$0 \leq d_{CB}(n) = 1 - \frac{1}{K_n} \sum_{i=1}^{K_n} d_{CB}(n;i) \leq 1 \tag{2.20}$$

$$d_{CB}(n;i) = \frac{||C_O^i(n) - C_I^i(n)||}{\sqrt{3 \times 255^2}} \tag{2.21}$$

where K_n is the total number of normal lines placed in equal distances (Figure 2.11) at the object boundary at the frame n, and $C_O^i(n)$ and $C_I^i(n)$ are average colour values calculated in the $M \times M$ neighbourhood for the interior end exterior end of the normal line, respectively. Since the difference between $C_O^i(n)$ and $C_I^i(n)$ is calculated as a distance in the Euclidean space, denominator in Equation 2.21 provides normalization to the range $\langle 0, 1 \rangle$.

The difference between the histograms H_n and $H_{n,av}$ is calculated using four metrices: L_1, L_2, χ^2 and *histogram intersection*. Because the length of the his-

tograms can be different, the scaling parameters R_1 and R_2 have to be introduced:

$$R_1 = \sqrt{\frac{N_{H_n}}{N_{H_{n,av}}}}, \qquad R_2 = \frac{1}{R_1}, \tag{2.22}$$

$$N_{H_n} = \sum_{j=1}^{B} H_n(j), \qquad NS_{H_n} = \sum_{j=1}^{B} H_n^2(j), \tag{2.23}$$

$$N_{H_{n,av}} = \sum_{j=1}^{B} H_{n,av}(j), \qquad NS_{H_{n,av}} = \sum_{j=1}^{B} H_{n,av}^2(j). \tag{2.24}$$

Another metric used to measure segmentation quality is the *motion metric*. When computing this metric, the authors assume that the motion of the background is zero or can be compensated. This is a very serious limitation since in the large amount of video sequences an accurate compensation of the camera motion is impossible. Another assumption is that motion boundaries exactly coincide with colour boundaries. This cannot be ensured by most motion estimation algorithms due to the occlusion effect and other problems that make such estimation inaccurate. The metric itself is defined as follows:

$$d_M(n) = 1 - \frac{\sum_{i=1}^{K_n} d_M(n;i)}{\sum_{i=1}^{K_n} w_i}, \tag{2.25}$$

$$d_M(n;i) = d(v_O^i(n), v_I^i(n)) \cdot w_i, \tag{2.26}$$

$$w_i = R(v_O^i(n)) \cdot R(v_I^i(n)), \tag{2.27}$$

where $v_O^i(n)$ and $v_I^i(n)$ are average motion vectors at the ends of the normal lines used for measuring the background and the object, respectively. The function $d(.)$ is the distance between two vectors:

$$d(v_O^i(n), v_I^i(n)) = \frac{\|d(v_O^i(n), v_I^i(n))\|}{\|d(v_O^i(n)\| + \|v_I^i(n))\|}, \tag{2.28}$$

and R, which denotes the reliability of the motion vector at the point p^i, is expressed as

$$R(v^i(n)) = e^{\left(-\frac{\|v^i(n) - b^i(n+1)\|^2}{2\sigma_m^2}\right)} \cdot e^{\left(-\frac{\|c(p^i;n) - c(p^i+v^i;n+1)\|^2}{2\sigma_c^2}\right)}, \tag{2.29}$$

where $b^i(n+1)$ is the backward motion vector and $c(p^i;n)$ is the image colour. The parameters σ_m and σ_c are chosen empirically.

The overall score is computed as follows:

$$D = \mu D_{CB} + \beta D_{CH} + \gamma D_M, \tag{2.30}$$

$$\mu + \beta + \gamma = 1. \tag{2.31}$$

The parameters μ, β and γ are used to balance the importance of the score components. For perfect segmentation, the value of this measure should be equal to 1.

Another attempt to create a standalone evaluation method was made by Correira and Pereira in [22]. They defined some metrics that are used for segmentation quality assessment. These metrics are divided into two groups:

- **individual object evaluation** – when the object is evaluated independently of the rest of the image,
- **overall evaluation** – when all objects from the sequence are evaluated at the same time.

In the first group of metrics, one can find *shape regularity*. This quality measure can be defined as compactness (*compact*):

$$compact(E) = \max\left(\frac{perimeter^2(E)}{75 \cdot area(E)}, 1\right), \tag{2.32}$$

where E is the evaluated object. Alternatively, it can be defined as a combination of circularity and elongation (*circ_elong*):

$$circ\_elong(E) = \max\left(circ(E), \max\left(\frac{elong(E)}{5}, 1\right)\right), \tag{2.33}$$

$$circ = \frac{4 \cdot \pi \cdot area(E)}{perimeter^2(E)}, \qquad elong(E) = \frac{area(E)}{(2 \cdot thickness(E))^2}. \tag{2.34}$$

Thickness is defined as a number of morphological erosion steps needed to completely erase a segment. The constant values are chosen empirically.

Another metric is *spatial uniformity*. It is defined as texture variance or spatial perceptual information. The latter case, however, does not fit into the objective

evaluation method since it relies on the subjective viewer response according to the ITU-T recommendation P.910 [98].

The *temporal stability* metric is defined as a combination of absolute differences of metrics defined earlier, namely, size, elongation and circularity. The *motion uniformity* metric is defined as a variance of motion vector values inside the object.

In the second group there are defined two metrices. One of them is a *local contrast to neighbours* defined as

$$contrast = \frac{1}{4 \cdot 255 \cdot N_b} \sum_{i,j} (2 \max(DY_{ij}) + \max(DU_{ij} + max(DV_{ij})), \quad (2.35)$$

where N_b is the number of border pixels and DY_{ij}, DU_{ij} and DV_{ij} are differences between colour components on the opposite sides of the border. The second is the *difference between neighbouring objects*. Here, many features describing the objects (such as shape regularity, spatial uniformity, motion uniformity, etc.) can be taken into account while computing the differences.

The results given by these measures are strongly content dependent. Not all of them are adequate for all kinds of sequences. This feature is characteristic also for other stand-alone segmentation evaluation methods. There can be easily found an example segmentation that fulfils all the assumptions mentioned earlier, but is still incorrect. In many disciplines, quality assessment is based on some kind of reference. The problem with video segmentation is that the reference is hard to obtain and it is not always easy to define the "right" segmentation.

Evaluation with Ground Truth

This group of segmentation quality assessment methods exploits reference segmentation done mostly manually. The way of preparing reference segmentation depends on the application. For example, in one application reference segmentation would result in objects with shadows, while another application would imply objects without shadows in reference segmentation.

A properly prepared reference segmentation can help in the selection of segmentation methods that are best-suited for a given task. The problem of reference segmentation will be discussed later in this chapter.

Segmentation evaluation with ground truth requires a definition of measure that will describe the difference between the assessed object and the reference object. This measure should reflect the general similarity of objects and should be insensitive to less important errors like missalignments. This problem has been considered by several authors [32, 44, 95, 150], who proposed a number of quality indices.

Erdem and Sankur [19] introduced motion penalty as a discrepancy between ground truth and the segmented object. This assumption goes too far, because it requires an almost perfect motion field. According to this measure, a very good shape will be rejected because of a motion different from ground truth. In Chapters 5 and 6 it will be shown that segmentation can be performed even with an erroneous motion field.

The above-mentioned motion penalty is defined as a distance function between two time evolution surfaces described by the motions of the reference object $M_{g_i}^n$ and the tested object $M_{s_i}^n$ at the frame n. The distance function is expressed as follows:

$$DM_i = \frac{1}{N} \sum_{n=1}^{N} \frac{||M_{g_i}^n - M_{s_i}^n||}{||M_{g_i}^n|| + ||M_{s_i}^n||}, \tag{2.36}$$

where i is the number of objects in the current frame.

Another measure used by Erdem and Sankur is the *missclassification penalty*. It is based on the distance between the boundaries of ground truth and the boundaries of the segmented object. The set of reference objects belonging to one frame is defined as $G = \{g_i, i = 1, \ldots, M\}$, while the set of segmented objects is defined as $S = \{s_i, i = 1, \ldots, M\}$, where M is the number of objects. These sets are accompanied by label functions that denote object membership

for the pixel in the location (k, l) at the frame n:

$$L_G^n(k,l) \in \{1, \ldots, M\}, \tag{2.37}$$

$$L_S^n(k,l) \in \{1, \ldots, M\}. \tag{2.38}$$

The above functions are used by the indicator function expressed as follows:

$$I^n(k,l) = \begin{cases} 1, & \text{if} \quad L_G^n(k,l) \neq L_S^n(k,l); \\ 0, & \text{if} \quad L_G^n(k,l) = L_S^n(k,l). \end{cases} \tag{2.39}$$

Finally, the value of the penalty has the following form:

$$DP_i^n = \frac{\sum_{(k,l)\in(s_i \cup g_i)} I^n(k,l) w_{g_i}(k,l)}{\sum_{\forall(k,l)} w_{g_i}(k,l)}, \tag{2.40}$$

where $w_{g_i}(k,l)$ is a modified Chamfer distance described in [87].

Another quality index is the *shape penalty*, which describes the differences in shape between ground truth and the object detected during the segmentation process. To describe the shapes, the authors use the turning angle function. The shape penalty measure has the following form:

$$DS^n = \frac{1}{M} \sum_{i=1}^{M} \left(\frac{\sum_{s=1}^{P} |\Theta_{g_i}^n(s) - \Theta_{g_i}^n(s)|}{P 2\pi} \right), \tag{2.41}$$

where Θ is the turning angle function and P denotes the number of contour elements. To make two turning angle functions comparable, they must be resampled prior to the computation to have the same length.

The final measures of quality is a weighted average of motion, missclassification and shape penalties. Weights can be adjusted in order to fit the measure into certain needs. The weights have to be chosen experimentally.

Another method of segmentation evaluation can be found in [59]. In this publication Izquierdo *et al.* presented a method based entirely on contour comparison. The contours of the reference object and the tested object are compared after a piecewise linear approximation. To make this approximation more reliable, the authors developed a new scale-space-based method.

To make a good approximation of a given contour, the most relevant points must be found. Such points are the points of a higher curvature. First, the contour is represented as a parameterized curve:

$$C : S \subseteq \mathbb{R} \rightarrow \mathbb{R}^2, \quad C(s) = [x(s), y(s)], \tag{2.42}$$

where s represents the curve arclength. A linear scale-space representation is obtained by curve evolution. In the subsequent steps of the evolution, the parameterized curve is smoothed by convolution with the Gaussian kernel. A detailed description of this procedure can be found in [59]. The points used later for contour approximation can be calculated on any scale by finding the local curvature extremes.

The contour of the object is approximated recursively starting from the coarsest scale. For each segment, the point with the largest value of approximation error is sought. Then, on a more detailed scale, the local extremum closest to that point becomes a new point of approximating polygon. The recursion stops when the desired level of accuracy is reached.

As a metric of comparison for simplified contours of the reference object and the segmented object, the authors used the L_2 norm between turning angle functions Θ of these contours. The function is measured from the reference point. This point is the point of the maximum curvature from the contour at the coarsest level of approximation. To make two turning angle functions comparable, they are normalized to the unit length. The distance between two polygons $\tilde{C}_0^1(s)$ and $\tilde{C}_0^2(s)$ is defined as follows:

$$dist_T(\tilde{C}_0^1, \tilde{C}_0^2) = \left\| \Theta_{\tilde{C}}^1 - \Theta_{\tilde{C}}^2 \right\|_2, \tag{2.43}$$

where$\|.\|_2$ is the L_2 norm and T is the given error for polygonal approximation.

Thanks to normalization, the functions are invariant with reference to translation, scaling and rotation. It is a convenient property; nevertheless, not only the shape itself is important for segmentation evaluation. An object segmented

out with the wrong position or size can be considered as an error by an observer but will be assumed valid by this algorithm.

2.4.3 Evaluation with Soft Reference Objects

Despite the chosen similarity measure, there is a common problem with the definition of reference segmentation. It is commonly assumed that segmentation is binary. Namely, pixels are considered as either belonging to the object or not. In the image, anyway, there is at least one pixel on the object border with uncertain membership, due to low sampling resolution and blurring by optics. For these pixels, the resulting values are a mixture of light coming from the object and from the background. The problem is even more important for moving objects, where, due to motion blur, the uncertain boundary can be even thicker. Ground truth is always defined by a human and such uncertain pixels can cause problems. It is hard to decide which pixel on the border should be considered as belonging to the object or to the background. This leads to situations where for the same sequence reference segmentations can differ when they are prepared by different people. Reference segmentations may differ even when they are prepared twice by the same person for the same frame from the sequence.

To overcome this problem, a new evaluation procedure is proposed in this dissertation. Ground truth must be prepared in a different way than it is done typically. It can be assumed that the operator is able to determine which pixels belong to the object for sure and which pixels belong to the background. Namely, two segments are prepared: the object and the background. Between these segments there must be at last one pixel uncertainty area. It is obvious that for more blurred objects, the gap between the object and the background will be bigger.

To be more general and to permit an extension to non-binary segmentation, deliberations are made in the continuous domain, i.e., they result in continuously valued segmentation masks. It is assumed that the area inside the object has the value equal to 1 and the area outside the object the value 0. The continuous-

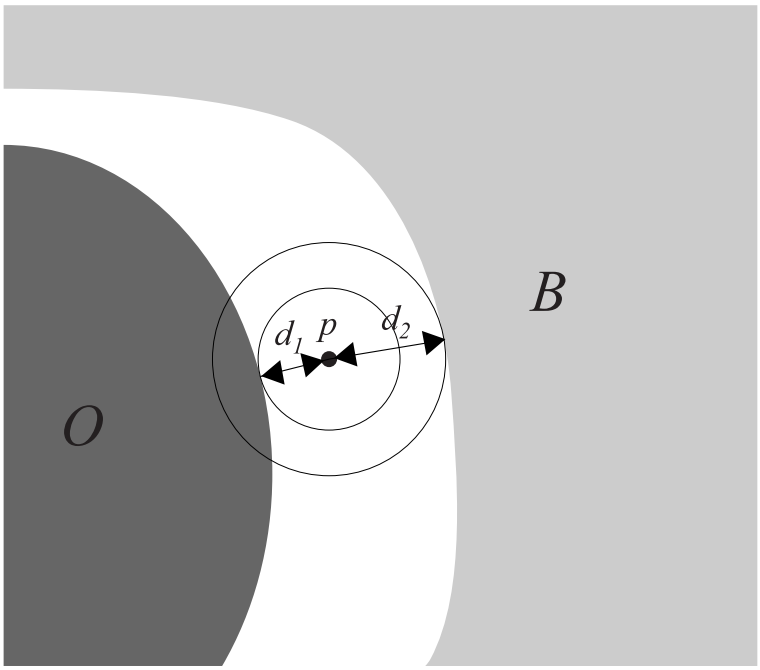


Figure 2.12: Value of the point p is calculated according to the distances d_1 and d_2. O – object, B – background

valued mask is obtained by finding at the point p the ratio of the distances from the object and background segments, d_1 and d_2, respectively (Figure 2.12). The value of the membership function M is equal to the distance from the background to the sum of distances to the background and the object:

$$M(p) = \frac{|d_2|}{|d_1| + |d_2|}. \tag{2.44}$$

To make a comparison possible, the tested object has to be represented in a similar way as the reference. Since all present segmentation algorithms give discrete results, the segmentation mask must be converted to the continuous form. This is done by connecting the centers of the border pixels with straight lines and interpolating the uncertainty surface between these two contours (Figure 2.13). This gives a line with the angle equal to 45° on the border cross section for binary segmentation. For a more blurred segmentation, the angle will be lower. Figure 2.13 shows three possible positions of the evaluated object edge (solid line). The absolute segmentation error is computed as a volume of differences between the evaluated object and the reference object computed outside the tolerance area.

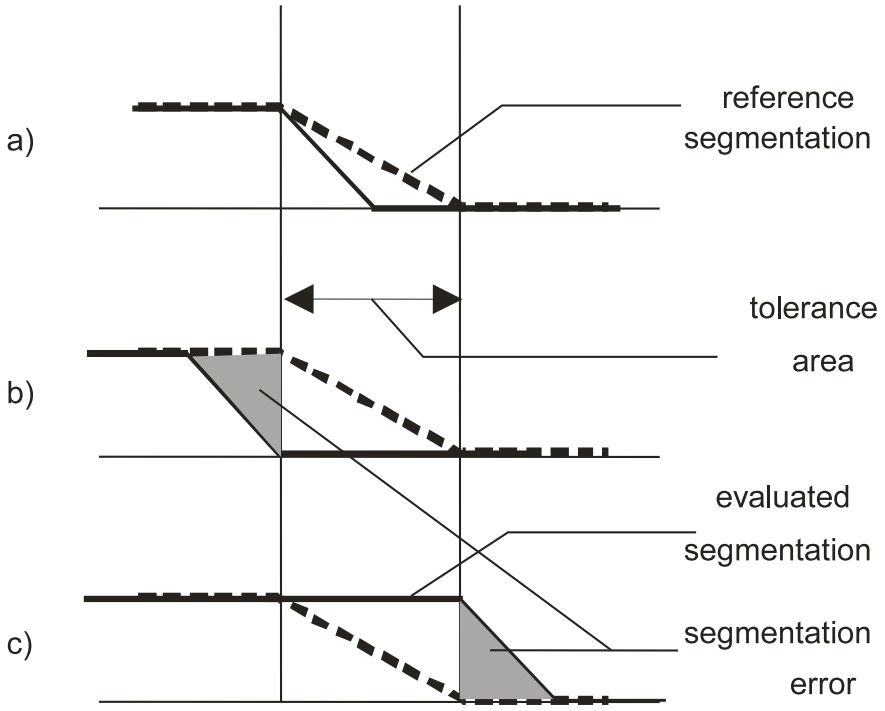


Figure 2.13: Cross-section through the object edge. Areas that are erroneous in the presence of tolerant ground truth are marked in gray

Namely, let V_r be the volume under the reference object under assumption that the pixels which are fully assigned to the reference have value 1, while the pixels that are totally outside the reference has value 0. Values of the remaining pixels are computed according to Equation 2.44. Let V_e be the volume of the erroneous area that is marked with gray in Figure 2.13. When the binary segmentation is evaluated, calculation of the V_e can be simplified by assumption that all the boundary pixels of the segmentation have value equal to 0.5 while remaining pixels have value 1. Finally, the soft reference index SR can be defined as follows:

$$SR = \frac{V_e}{V_r} \cdot 100\%. \tag{2.45}$$

The error value V_e can be used as an input to some metrics proposed in the literature [19, 59, 87] to make it more comparable to the object area. However, current SR index definition is good enough to make the proper quality assessment.

Presently, there are no methods of segmentation that give soft segments. However, it seems to be a natural direction of development in this field and such algo-

rithms will appear soon. This evaluation method will naturally handle problems with the evaluation of these new segmentation methods.

Chapter 3

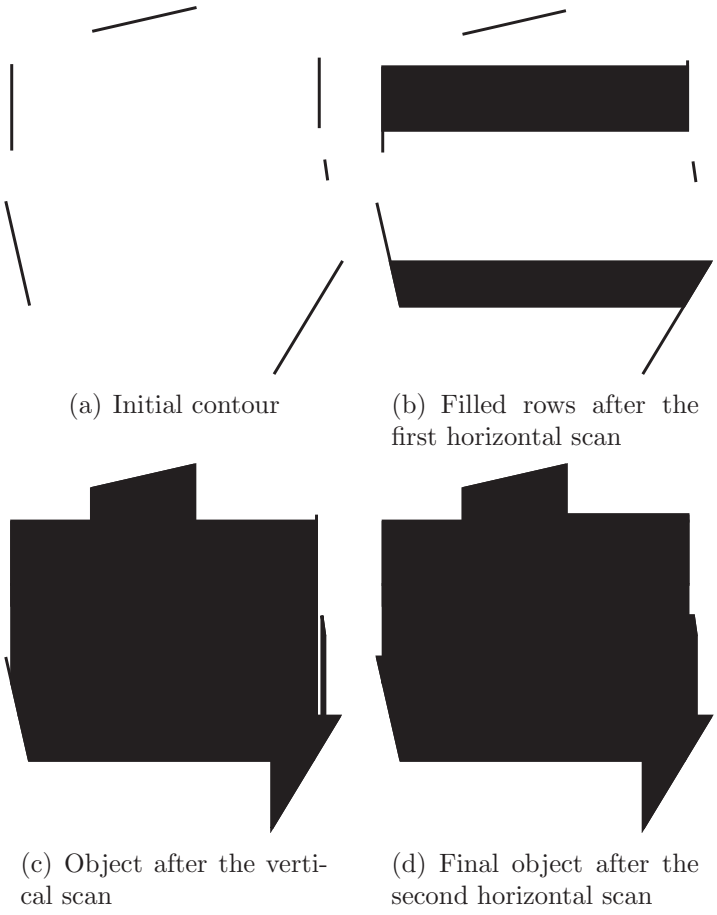
Region-Based Segmentation

3.1 Segmentation Based on Differences

In the case of a static camera, techniques based on the *frame difference* (FD) are the most straightforward techniques for moving object detection. They are often used to segment video sequences [12, 18, 29, 85, 92]. To avoid the influence of noise, the differences between consecutive frames are computed within some window. A change is detected when the frame difference exceeds some predefined threshold. The main problem is that the most significant differences are usually located on the object border. This is especially visible when the object has a uniform interior. The biggest problem is to obtain all pixels that belong to the object from such incomplete data.

An example technique for filling in an incomplete boundary can be found in [86]. First, the binary change detection mask is cleaned up by removing single pixels. This is done using a morphological operator. Next, the remaining contours are filled. Every row of the change mask is scanned for pixels containing borders. The portion of the row contained between two pixels is considered as the object interior and is filled. Then the operation is repeated for columns and again for rows. The process is shown in Figure 3.1.

When the displacement of an object in a scene is small from frame to frame, it is difficult to detect frame differences. An extension to segmentation based on the difference that solves this problem is the *accumulative difference picture* [60].



(a) Initial contour

(b) Filled rows after the first horizontal scan

(c) Object after the vertical scan

(d) Final object after the second horizontal scan

Figure 3.1: Filling of an object using an incomplete boundary

All changes in a sequence are measured against the reference frame. The value of the accumulative difference picture is increased by one for every pixel that is different from the corresponding pixel from the reference frame. To reduce the noise influence, the difference between the pixels must be higher than some predefined threshold. This technique is particulary useful for the segmentation of the "head and shoulders" sequences, where the segmented object presents a small movement that oscillates around one position.

A similar approach that is based on building a reference background over several frames is presented by Li *et al.* in [75]. The method is limited to a static camera only and an empty background must be acquired before the segmentation process. To deal with changes in lighting conditions and shadows, the method

uses the HSV colour space.

First, the statistical model of the background has to be built over the first few images. The probability density function for the i-th point is defined as follows:

$$f(\mathbf{x}_i(t)) = \frac{1}{\left(\sqrt{2\pi}\right)^3 \prod_{K=H,S,V} \sigma_{Ki}} \exp\left(-\frac{1}{2} \sum_{K=H,S,V} \left(\frac{x_{Ki}(t) - u_{Ki}}{\sigma_{Ki}}\right)^2\right), \quad (3.1)$$

where $\mathbf{x_i} = (x_{Hi}, x_{Si}, x_{Vi})^T$ is the i-th pixel, K is the value of the colour at the i-th pixel, u_{Ki} is the mean colour value over the N tested background images and σ_{Ki} is its standard deviation.

Li *et al.* [75] discovered that colour components show higher variation in some image areas than in others. This is mainly the result of the properties of the HSV colour space. When the V value is low, the values of H and S become less reliable. Also, for low saturation, the H component becomes less stable. Under such conditions, the values of pixels are more sensitive to camera noise and lighting conditions. To overcome this problems, the authors proposed to divide the pixels into four classes according to the σ_{Ki} value, namely V, VS, VH and VHS. The resulting image is then smoothed using a 5×5 median filter.

The decision of assigning a pixel to the foreground or the background is based on multiple clues. However, it is assumed that if a change in the value of V exceeds H_{th}, the pixel is assigned to the background. The value of the threshold is set empirically to 75. The set of background pixels is denoted by B while of foreground pixels by F. *Condition1* is met when the V change is high while H and S remain on the same level. This happens in the shadow areas. *Condition2* decides whether the background needs to be updated because of the global lighting conditions change. The full decision process is presented in the following pseudocode:

```
for each pixel I
  if |ΔV| > H_th then
    i ∈ F
  else if Condition1 is met then
    i ∈ B
  else if Condition2 is met then
```

```
    α_V = α_VIllum
    {check the proper components of H, S, V
    according to to type model}
    if |x_Ki(t) − u_Ki| < α_K α_Ki then
      for every usable component K
        i ∈ B
  else i ∈ F
```

After the processing, the result is smoothed using the following convolution kernel:

$$\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array} .$$

When capturing a reference frame is impossible and when, additionally, the camera is moving, the reference background can be reconstructed using the *mosaicing* [10, 11, 142, 146] technique. The main idea of this technique is to build the background image using aligned parts that come from different frames. The problem is that the process of aligning frame parts (*registration*) is difficult in the case of a moving camera. Moreover, this technique is only applicable to off-line processing, since it requires two passes: in the first pass the mosaic is built, in the second pass the actual segmentation is done.

An exception from this rule is surveillance application [88] of mosaicing, where the camera tracks the same scene and the camera movement is limited. In such a case, the mosaic can be built in the "clean" area tracked by the camera. When the reference background is ready, the tracking of any changes is easy.

Intruder detection can be much faster when the camera is equipped with a position tracking device. In such a case, the position of the acquired frame on the mosaic can be determined very fast without exhausting global motion estimation.

3.2 Motion Segmentation

Motion video segmentation methods do not rely on differences between consecutive frames but take into account the motion field estimated for every frame. Usually [13, 63, 108, 126], a dense motion field is used for segmentation to ensure the highest possible segmentation quality. However, some authors attempt to apply the sparse motion field used in video compression. The method developed by Si and Park [62] uses the DCT coefficients and motion vectors provided by the MPEG-4 encoder. This is a very attractive assumption because no additional computation of motion is required; however, this motion field has two main drawbacks. The first one is the fact that the motion field is sparse. In a compression process, the vectors are estimated for the blocks. The second problem is the fact that motion estimation algorithms used in the compression process are not intended to find the real motion vector for a given block but to find the most similar block in the previous frame. This means that the motion field which is very good for compression may be totally useless from the segmentation point of view. This can lead to significant segmentation errors.

3.2.1 Segmentation Using Parametric Motion Models

In some approaches [7, 55, 134], authors try to segment a scene by finding areas with motion that conform to some parametric model. To reflect an apparent motion of 3D objects onto the image plane, a number of parametric models are used, including:

1. affine model with six parameters:

$$\begin{aligned} x' &= a_1 x + a_2 y + a_3 \\ y' &= a_4 x + a_5 y + a_6, \end{aligned} \tag{3.2}$$

2. model that takes into account perspective projection:

$$\begin{aligned} x' &= \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1} \\ y' &= \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}, \end{aligned} \tag{3.3}$$

3. parabolic surface model under perspective projection:

$$\begin{aligned} x' =& a_1x^2 + a_2y^2 + a_3xy + a_4x + a_5y + a_6 \\ y' =& b_1x^2 + b_2y^2 + b_3xy + b_4x + b_5y + b_6, \end{aligned} \tag{3.4}$$

where (x, y) and (x', y') are corresponding points in the frames n and $n + 1$, respectively.

3.2.2 Multi-Valued Segmentation with K-Means

Video segmentation is often performed using approaches similar to static image segmentation. A very popular algorithm that permits the segmentation of multivalued images is K-means. In its adaptation to video segmentation, motion information is used as an extension of colour information [5, 14, 69].

For example, in [69], the algorithm treats a number of consecutive images as one three dimensional image. Additionally, the authors introduce a *spatial proximity constraint* that prevents the creation of disconnected regions of the same class. A modified version of the K-means algorithm which introduces the connectivity constraint works in the following way:

Step 1 After a certain small number of iterations of the standard K-means algorithm there are K regions with intensity centers CI_k. Then spatial centers are computed as $\mathbf{CS}_k = (CS_{k,X}, CS_{k,Y}), k = 1, \ldots, K$:

$$CS_{k,X} = \frac{1}{M_k} \sum_{m-1}^{M_k} p^k_{m,X} \tag{3.5}$$

$$CS_{k,Y} = \frac{1}{M_k} \sum_{m-1}^{M_k} p^k_{m,Y}, \tag{3.6}$$

where $p^k_{m,X}, p^k_{m,Y}$ are the horizontal and vertical coordinates of the pixels belonging to the class s_k, accordingly. From the same set of data, differential motion centers CV_k are computed:

$$CV_k = \frac{1}{M_k} \sum_{m-1}^{M_k} ||\mathbf{P}^k_{m,t+1} - \mathbf{P}^k_{m,t}||, \tag{3.7}$$

where $\mathbf{P}^k_{m,t}, m = 1, \ldots, M_k$ are the pixels of the k-th region at the time t. The mean area of all segments is defined as

$$\bar{A} = \frac{1}{K} \sum_{k=1}^{K} M_k. \tag{3.8}$$

Step 2 For every pixel, a generalized distance is calculated according to the following equation:

$$D(\mathbf{p}, k) = \frac{\lambda_1}{\sigma_I^2} ||I(\mathbf{p}) - CI_k|| + \frac{\lambda_2}{\sigma_V^2} ||V(\mathbf{p}) - CV_k|| + \frac{\lambda_3}{\sigma_S^2} \bar{A} \frac{||\mathbf{p} - \mathbf{CS}_k||}{A_k}, \tag{3.9}$$

where $V(\mathbf{p}) = ||\mathbf{p}^k_{t+1} - \mathbf{p}^k_t||$, σ_I, σ_V, σ_S are the standard deviations of the respective parameters and λ_1, λ_2, λ_3 are regularization parameters.

Step 3 A component labelling algorithm is applied to find all connected regions. To each region, a unique label is assigned. Regions of a size below a certain threshold are not labelled. The result of this process is L connected regions. For these regions, intensity, spatial and motion centers are calculated.

Step 4 If changes in the intensity centers are below a certain threshold, then **stop**, else proceed to **Step 2** with $K = L$.

A general problem with the K-means algorithm is that the number of required segments must be known before segmentation. If there is no information about scene contents, it is impossible to automatically determine the number of required segments. A solution to this problem was shown in the method presented by Habili and Ngan in [46]. The authors use a multi-feature K-means algorithm. As the components of the feature vector they use the colour in the YC_bC_r space, the position of the pixel and its motion. A block diagram of this segmentation is presented in Figure 3.2 with $F(x, y, t)$ denoting the pixel value and $L(x, y, t)$ denoting the cluster membership.

A complete set of n feature vectors is defined as $C = \mathbf{u}_1, \ldots, \mathbf{u}_n$, while $C_1, \ldots, C_k$ is the partition of this set into k clusters. For the clustering purposes, the authors minimize by means of the K-mens algorithm sums of squared

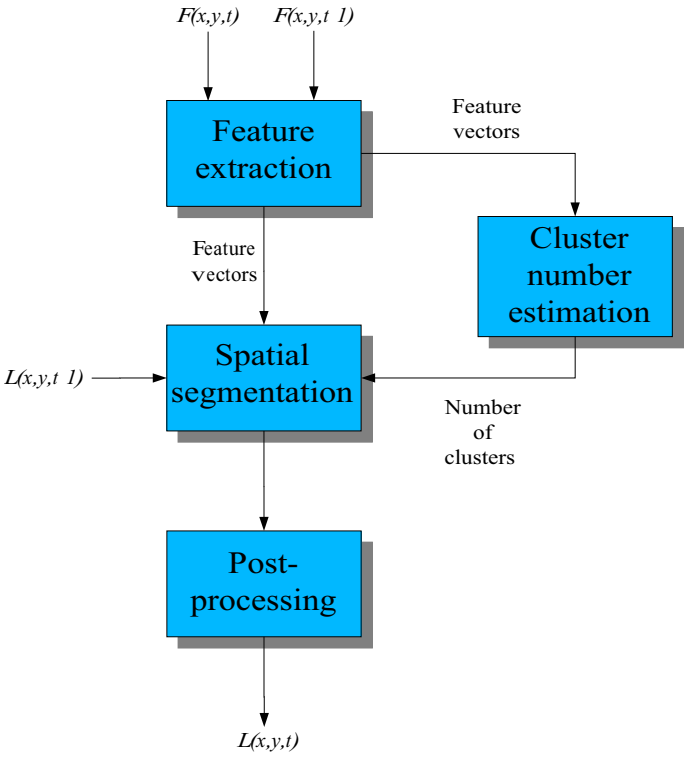


Figure 3.2: Block diagram of the clustering-based segmentation

errors defined as

$$J = \sum_{i=1}^{k} \sum_{\mathbf{u} \in C_i} d^2(\mathbf{u}, \mathbf{m}_i), \tag{3.10}$$

where

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{u} \in C_i} \mathbf{u} \tag{3.11}$$

is the mean of the samples belonging to one cluster and

$$d(\mathbf{u}, \mathbf{m}_i) = ||\mathbf{u} - \mathbf{m}_i|| = \sqrt{\sum_{l=1}^{f} (u_l - m_{il})^2} \tag{3.12}$$

for the f-dimensional feature space.

All image features have different ranges of possible values. To make them comparable and unify their influence on the clustering process, a normalization procedure must be performed. In order to do that, for every feature the standard

deviation σ_j is computed:

$$\sigma_l^2 = \frac{1}{n}\sum_{q=1}^{n}(u_{ql} - m_l)^2, \tag{3.13}$$

where

$$m_l = \frac{\sum_{q=1}^{n} u_q l}{n}. \tag{3.14}$$

After the normalization it is easier to control the importance of each feature by assigning the weights w_l. Taking into account the weights, Equation 3.12 can be expressed as follows:

$$d'(\mathbf{u}, \mathbf{m}_i) = \sqrt{\sum_{l=1}^{f} w_l \frac{(u_l - m_{il})^2}{\sigma_l^2}}. \tag{3.15}$$

Because the K-means algorithm works for a fixed number of clusters, some method of the estimation of this number must be employed. In this implementation, the authors are investigating the behaviour of J for different numbers of the clusters k. It is done sequentially for $k = 1, \ldots, k_{max}$. The best number of clusters $\hat{k}$ is estimated based on the observation that the value of J decreases rapidly for a growing k until $k = \hat{k}$, then slows down and reaches zero for $k = n$. It means that a corner on the J versus k plot appears for the best k. In practice, this is realized by the following condition:

$$\frac{J_k}{J_{k+1}} < \theta, \tag{3.16}$$

where θ is an empirically chosen sensitivity threshold.

Algorithms based on K-means offer high segmentation quality, even in the presence of noise. Nevertheless, their iterative nature and computational complexity make this group of methods inappropriate for real time applications.

3.2.3 Bayesian Segmentation

Bayesian methods are very often applied to the video segmentation problem [15, 91, 103, 145]. The general idea of Bayesian segmentation is to find a maximum a

posteriori probability (MAP) estimation of the segmentation X for some given observations O:

$$P(X|O) = P(O|X)P(X), \tag{3.17}$$

where $P(O|X)$ is conditional probability modelled as white Gaussian noise with a zero mean and $P(X)$ is the *a priori* likelihood. The segmentation X is usually modelled as a *Markov random field*.

An example of such an approach can be found in [68]. Here, a pixel is treated as a seven-element vector $\mathbf{x} = [x, y, Y, U, V, u, v]^T$, where (x, y) are spatial coordinates, (Y, U, V) denotes colour and (u, v) is a motion at the pixel position.

The image is segmented into n classes denoted by c_i for $i = 1, .., n$. It is assumed that the assignment for each pixel in the previous frame is known and given by $L_{t-1}(x, y)$, where t denotes the time index and $1 \leq L_{t-1}(x, y) \leq m$.

The probability that the pixel (x, y) belongs to the class c_i is given by $P(c_i|\mathbf{x}_t(x, y))$. Using the Bayes rule, it can be shown that

$$P(c_i|\mathbf{x}_t(x, y)) = \frac{P(\mathbf{x}_t(x, y)|c_i)P(c_i)}{P(\mathbf{x}_t(x, y))}. \tag{3.18}$$

Equation 3.18 can be used to find for each pixel probability of belonging to each class. The class with the maximum probability is then assigned to this pixel according to the following equation:

$$L_t(x, y) = \arg\max_i \left\{ \frac{P(\mathbf{x}_t(x, y)|c_i)P(c_i)}{P(\mathbf{x}_t(x, y))} \right\}, \tag{3.19}$$

where $1 \leq i \leq n$. It can be further simplified because the denominator is always positive and independent of i. Additionally, the authors introduce the logarithmic likelihood relationship for better class identification:

$$L_t(x, y) = \arg\max_i \{\ln(P(\mathbf{x}(x, y)|c_i)) + \ln(P(c_i))\}. \tag{3.20}$$

The segmentation process starts by iteratively computing for each pixel the probability of belonging to one of the Gaussian distributions and recomputing

the parameters of these distributions. This process is repeated a fixed number of times. In the second step, these regions are used to compute classes, which results in a new regions distribution. The second step is repeated until the regions stop changing.

The large number of computations and the iterative nature of this method make it very hard to implement in a real time.

3.3 Summary

There can be observed a lack of methods that fulfil all the assumptions given in Chapter 1. Most of the methods require a static background. If the background can move, it must be motion compensated, which results in a static background as well. Some of the methods based on K-means can segment the frame only into a fixed number of objects. If any method can deal with more complex cases, it cannot perform in a real time.

Chapter 4

Active Contour Segmentation

4.1 Snakes

A snake is an active contour model introduced by Kass *et al.* in [65, 66] and also applied in [61, 89, 94, 140, 141]. It is a method of manipulating a closed planar curve with a set of markers placed along the curve. Originally, a snake curve was approximated by straight segments that connect markers.

The parametrization of the planar curve C is given as follows:

$$C : [0, 1] \rightarrow \mathbb{R}^2, p \rightarrow C(p). \tag{4.1}$$

This curve is placed over a two dimensional image I.

The following energy is distributed along the curve:

$$E(C(p)) = \alpha \int_0^1 E_{int}(C(p))dp + \beta \int_0^1 E_{img}(C(p))dp + \gamma \int_0^1 E_{con}(C(p))dp, \tag{4.2}$$

where the term E_{int} keeps the curve regular and smooth, E_{img} is the component dependent on image properties and E_{con} is the constraint that keeps the curve in a certain domain. The image-dependent component is defined to attract the curve to desired image features, for example, edges. During propagation every marker is shifted in a direction that will lead to the minimization of contour energy.

This kind of active contour has some disadvantages. It requires initialization in the close neighborhood of the expected solution. Additionally, the approximation error increases as the distance between the markers increases. Moreover,

moving markers can create self-intersections of the propagating curve. This can be reduced by a properly defined internal energy function that will increase the rigidity of the contour. However, such a constraint makes the contour unable to detect small details.

To provide better shape representation, snake markers can be connected using B-splines. This technique was used in [21, 28, 107]. Thanks to B-splines, shape can be better approximated using fewer markers, which increases the performance of the method.

4.2 Geodesic Active Contours

Geodesic active contours represent a technique for curve evolution to minimize some metric. This technique was introduced in [17] as an alternative for snakes. If the metric to be minimized is defined depending on the object boundary, it can be used for image segmentation [43, 72, 147].

Let the curve $C(p) = \{x(p), y(p)\}$, where $p \in [0, 1]$ is an arbitrary parametrization. The function that describes the propagating contour is an arc-length parameterized functional:

$$S[C] = \int_0^{L(C)} g(C)ds, \tag{4.3}$$

where ds is the Euclidean arc-length and L the Euclidean length of $C(p)$. The function $g(.)$ is a scalar edge detector function. The value of this function should be minimal on the object boundary. Typically [17, 80], this function is defined as

$$g(C) = \frac{1}{1 + |\nabla \hat{I}|^n}, \tag{4.4}$$

where $\hat{I}$ is the segmented image smoothed with the Gaussian function and $p \in \{1, 2\}$. The value of this function is taken at the point of the curve that is going to be propagated.

Curve propagation towards a local minimum is performed using the gradient

descent method by the following equation:

$$\frac{\partial C}{\partial t} = g(C)\kappa\vec{\mathcal{N}} - (\nabla g(C) \cdot \vec{\mathcal{N}})\vec{\mathcal{N}}, \tag{4.5}$$

where t is the time moment of curve evolution, $\vec{\mathcal{N}} = \frac{\nabla C}{|\nabla C|}$ is the normal vector to the curve C, and κ is the curvature defined with the following equation:

$$\kappa = \nabla \cdot \frac{\nabla C}{|\nabla C|}, \tag{4.6}$$

which is equal to the divergence of a vector field formed by normal vectors.

Contrary to the classical snakes, geometric active contours do not depend on curve parametrization. Since the value of the function g is equal to zero only on the object boundary, this method is relatively independent of the initial conditions.

An application of geodesic active contour to video segmentation was also proposed by Gastaud and Barlaud in [40]. This method deals with a moving camera, which is a common problem in video segmentation. It is impossible to detect moving objects using solely image differences. The key idea of the method is to create one big picture containing all the background that appears in the camera during the entire sequence. This technique is called *mosaicing* [37, 49, 135]. It is not suitable for on-line application because it requires two passes:

- first, the background mosaic is created,
- then, active contour segmentation is applied in order to detect moving objects.

During the segmentation stage the position of the current image has to be found in the mosaic image. Then the contour is propagated on the basis of the absolute difference between the background and the current image.

Geodesic active contours have also been applied to such problems as medical image segmentation, volume reconstruction, object tracking.

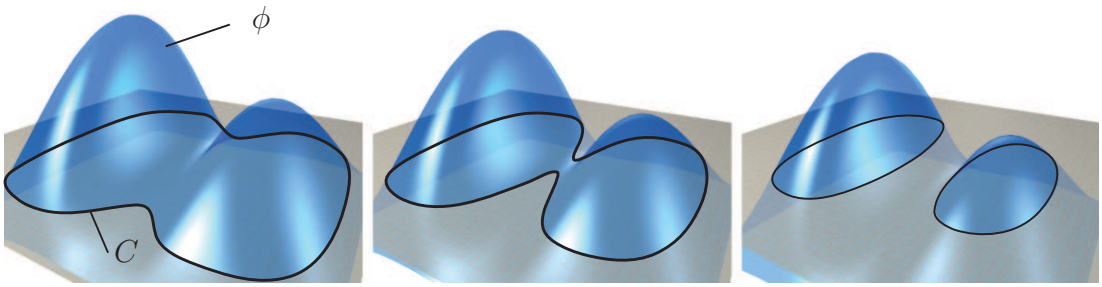


Figure 4.1: Contour defined by a zero level set of a propagating surface

4.3 Level Set Methods

The Level Set Method developed by Osher and Sethian in [97] provides a solution to stable curve evolution with changing topology. The idea consists in embedding the propagating curve C as a zero level set of a surface ϕ. Then, instead of propagating a parametric curve, one can propagate a strictly defined and well-behaved function. The propagating contour can be always found taking a zero level set of the surface ϕ at a certain time, which is shown in Figure 4.1. The only problem is to define the propagation of ϕ in such a way so that its zero level set will follow the original evolution equation $\frac{\partial C}{\partial t}(p) = F(\kappa)\vec{\mathcal{N}}$.

The above can be illustrated by the equations

$$\begin{cases} C(p,0) = \{(x,y) : \phi(x,y,0) = 0\}, \\ C(p,t) = \{\phi(x,y,t) = 0\}, \quad C(t) = \phi^{-1}(0). \end{cases} \tag{4.7}$$

The zero level set of the surface ϕ is given by the equation

$$\phi(C(t),t) = 0. \tag{4.8}$$

Since the curve C must always match the zero level set of the surface ϕ during evolution, the values of $\phi(C(t),t)$ should be always zero while the time changes. Namely, the partial derivative of ϕ with respect to t should be zero. By the chain rule,

$$\nabla\phi(C(t),t) \cdot \frac{\partial C}{\partial t} + \frac{\partial \phi}{\partial t} = 0. \tag{4.9}$$

The curve C is evolving in a normal direction $\vec{\mathcal{N}} = \frac{\nabla\phi}{|\nabla\phi|}$ with a given speed F, namely,

$$\frac{\partial C}{\partial t} \cdot \vec{\mathcal{N}} = F. \tag{4.10}$$

From the above equation,

$$\frac{\partial C}{\partial t} = F \cdot \frac{|\nabla\phi|}{\nabla\phi}. \tag{4.11}$$

Substituting Equation 4.11 into Equation 4.9 yields an evolution equation for ϕ:

$$\frac{\partial \phi}{\partial t} + \nabla\phi(C(t), t) \cdot F \cdot \frac{|\nabla\phi|}{\nabla\phi} = 0, \tag{4.12}$$

$$\frac{\partial \phi}{\partial t} + F|\nabla\phi| = 0, \tag{4.13}$$

assuming that the initial condition $\phi(x, y, 0)$ is given. Usually [2, 79, 102], the initial surface is defined by the signed distance function from the propagating curve. The function returns positive values inside the contour and negative ones outside of it.

The advantages of the level set approach are:

- It can be generalized to higher dimensions of C, for example, the method can propagate a three dimensional surface that forms a zero level set of a four dimensional hypersurface.
- Topology changes of the propagating contour are handled in a natural way. Despite the splitting and merging of the propagating curve, the function ϕ remains single valued.
- It can be adapted to solving many problems by a redefinition of the speed function F.
- The values of the speed F may vary from positive to negative ones, which permits very complicated propagation of the curve.

Exemplary applications of this method are presented in [6, 42, 120].

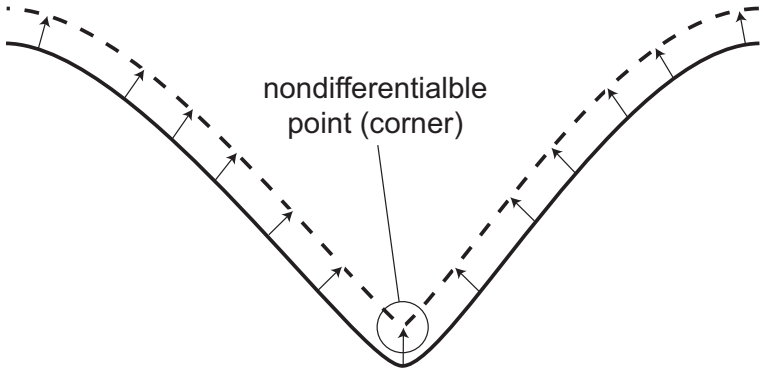


Figure 4.2: Possible evolution of a concave part of the contour

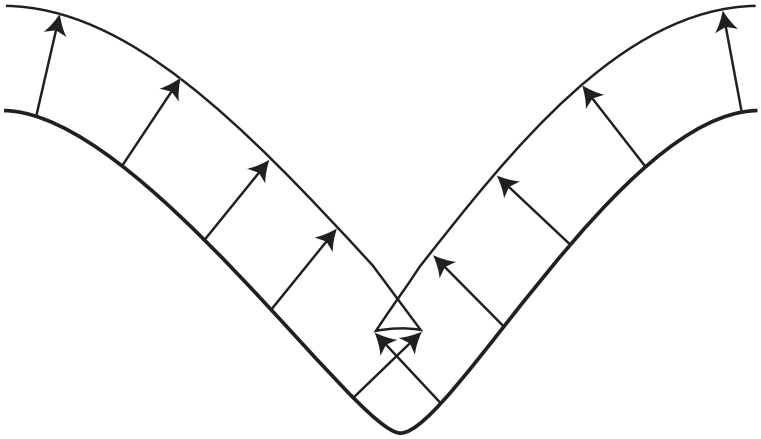


Figure 4.3: Swallowtail solution

4.3.1 Numerical Schemes

During evolution, the contour C as well as the level set function ϕ may become undifferentiable. For example, such a situation is possible in concave parts of the propagating curve, which is illustrated by Figure 4.2. To make the propagation of the contour possible under any conditions, a numerical scheme that produces an entropy-satisfying weak solution is required [97, 111, 114].

When the contour that formed the corner is further propagated along its normal direction, it can cross itself forming the so-called swallowtail solution (Figure 4.3). This situation is very hard to handle by the classical active contour methods. To overcome this situation, two strategies can be used.

One of the strategies is to treat the moving contour as a propagating wave.

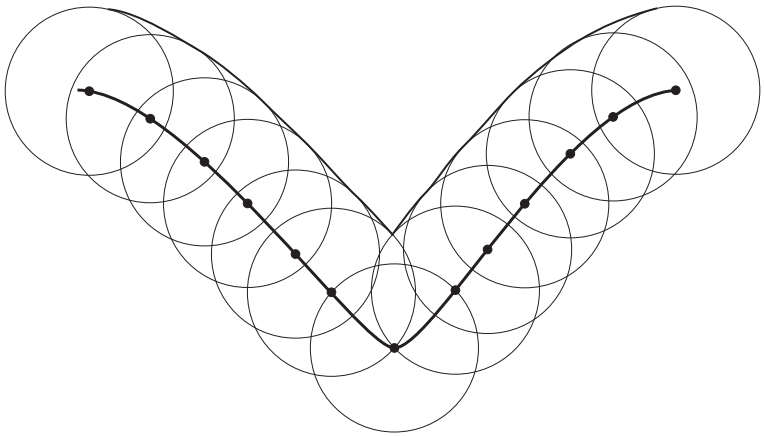


Figure 4.4: Illustration of a solution based on the Huyghens principle

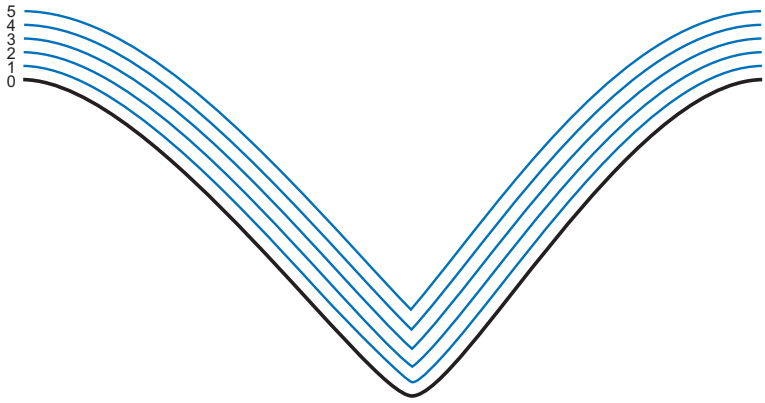


Figure 4.5: Curve evolution that uses an entropy-satisfying solution to the gradient

By the Huyghens principle, every point reached by a propagating wave becomes the source of a new wave. Following this rule, the propagation can be performed by shifting a circle of a small radius along the curve. This circle will mimic waves starting from every point of the contour. The line created by the edges of the circles will be the new position of the propagating curve (Figure 4.4).

Another way to overcome the problem with self-crossing contours is the introduction of an entropy condition [115]. This condition states that a point once visited by a contour cannot be visited by another part of the contour (Figure 4.5),analogously to the fire moving through the forest. The trees once burnt cannot be burnt one more time.

A numerical scheme producing the correct weak solution was proposed in [97].

The surface ϕ is approximated on a rectangular grid where ϕ_{ij} denotes the value of the function at the grid point with the (i, j) coordinates. The step of the surface evolution with the speed F defined for every point has the following form:

$$\phi_{ij}^{n+1} = \phi_{ij}^{n} - \Delta t[\max(F_{ij}, 0)\nabla^{+} + \min(F_{ij}, 0)\nabla^{-}], \tag{4.14}$$

where

$$\begin{aligned}\nabla^{+} =&[\max(D_{ij}^{-x}, 0)^2 + \min(D_{ij}^{+x}, 0)^2 + \\ &\max(D_{ij}^{-y}, 0)^2 + \min(D_{ij}^{+y}, 0)^2]^{1/2},\end{aligned} \tag{4.15}$$

$$\begin{aligned}\nabla^{-} =&[\max(D_{ij}^{+x}, 0)^2 + \min(D_{ij}^{-x}, 0)^2 + \\ &\max(D_{ij}^{+y}, 0)^2 + \min(D_{ij}^{-y}, 0)^2]^{1/2},\end{aligned} \tag{4.16}$$

and n is the number of evolution step and Δt is the size of the time step per iteration. The finite difference operator D is constructed according to the following scheme:

$$D_{ij}^{+x} = \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x}, \qquad D_{ij}^{-x} = \frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x}, \tag{4.17}$$

$$D_{ij}^{+y} = \frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta y}, \qquad D_{ij}^{-y} = \frac{\phi_{i,j} - \phi_{i,j-1}}{\Delta y}, \tag{4.18}$$

where Δx and Δy denote grid spacing.

4.3.2 Level Set Segmentation

The level set approach has been widely used for image [47, 52, 123, 137] as well as video [50, 82] segmentation.

A level set based method proposed by Konrad and Ristivojević in [70] treats a sequence of images as the so-called "video cube." It means that segmentation is performed in a 3D space, where two dimensions are spatial dimensions of the sequence and the third dimension is time. This methods exploits ideas similar to those applied to 3D shape recovery from a cloud of points [151, 152].

The problem is defined as the maximum *a posteriori* probability estimation (MAP). The method consists in evolving a parameterized surface ϕ in the x, y, t

space. The surface is estimated on the basis of a subset of the image sequence $\mathcal{I} = \{I_n : n_0 \dots n_0 + N\}$, where n_0 is the starting frame number and N is the number of frames. It is assumed that motion is affine with constant or slowly varying velocity. The motion inside ϕ is denoted by $\boldsymbol{p}$ while outside by $\bar{\boldsymbol{p}}$. Additionally, it is assumed that motion trajectory for each image pixel can be computed from $\boldsymbol{p}$ or $\bar{\boldsymbol{p}}$. The MAP is then formulated as follows:

$$\max_{\phi,\boldsymbol{p},\bar{\boldsymbol{p}}} p(\phi, \boldsymbol{p}, \bar{\boldsymbol{p}}|\mathcal{I}) = \max_{\phi,\boldsymbol{p},\bar{\boldsymbol{p}}} p(I_t|\phi, \boldsymbol{p}, \bar{\boldsymbol{p}}, \mathcal{I} \setminus \{I_n\}) pp(\phi, \boldsymbol{p}, \bar{\boldsymbol{p}}|\mathcal{I} \setminus \{I_n\}), \tag{4.19}$$

where I_n is an image from the sequence at the frame n and p is probability density. This method is able to track objects without strong intensity edges on the object boundary. However, it requires a static background and only one object can be segmented. The authors also report high computational complexity of this approach, which, along with the requirements for the large number of frames to be processed simultaneously, makes this method unsuitable for real-time application.

In the literature there can be also found approaches to video segmentation that do not employ motion information. An example is the method presented by Harper and Reilly in [50]. The segmentation process is based on colour information only and in fact could be applied to static image without any modification. The method is based on the classical Level Set Method.

The method is limited to the detection of human faces in video sequences. As a feature space, the normalized rgb space was chosen. Since $\bar{r} + \bar{g} + \bar{g} = 1$, only the $\bar{r}$ and $\bar{g}$ components are used in the segmentation process. To cover the wide range of skin tones that vary for human races and lighting conditions, a 2D histogram of the distribution Υ is built on the basis of the data from a database prepared earlier. To remove noise and outliers, some morphological operations are performed. Based on the number of samples in the database, a threshold t_{skin} is computed. Using this threshold, propagation speed for the level set algorithm is computed. The speed depends on the likelihood of the pixel to belong to the

distribution Υ:

$$F = \left(\log \left(\frac{\Upsilon(r,g)+1}{t_{skin}} \right) \right). \tag{4.20}$$

The logarithm in Equation 4.20 helps to visually linearize colour distribution.

The strong dependence on the predefined database does not make this method very flexible. The colour of the skin is not unique in nature. It is highly probable that some objects can be wrongly segmented. The wider range of colours in the database, the higher the probability of segmentation errors. Too small a number of features used in segmentation (especially video segmentation) often leads to problems with the detection of meaningful objects.

4.4 Fast Marching Methods

4.4.1 Description of the Fast Marching Algorithm

The *Fast Marching Method* [119] is an extremely fast version of the Level Set Method, but it has some limitations. Namely, curve propagation speed must be of a constant sign and a curve may propagate in one direction only. There exists an opinion that this disadvantage should prevent fast marching equations from being directly applied to video segmentation. Here, we are going to show that fast marching can be applied directly in a quite efficient way. The main advantage of the Fast Marching Method is its low computational complexity, which is $O(N \log N)$, where N is the total number of points in the area of interest.

In a two dimensional case, a curve (contour) propagates with the speed $F(x,y)$ and arrives at the point (x,y) at the time $T(x,y)$.

For the surface $T(x,y)$, there is

$$|\nabla T| F = 1. \tag{4.21}$$

The idea is to sweep the curve ahead by considering a set of points in a narrow band around the existing front and to march this narrow band forward, freeze the values of the existing points and bring new ones, also referred to as *trial points*,

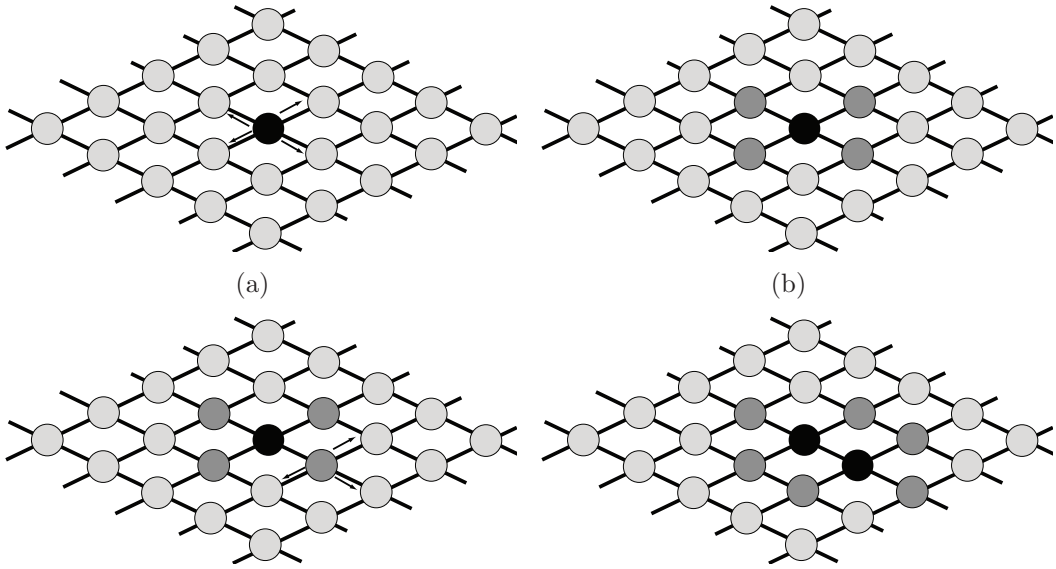


Figure 4.6: Front propagation: (a) – starting point, (b) – neighbour points marked as trial, (c) – point with the smallest time value chosen for propagation, (d) – point chosen in the step (b) is marked as visited and its neighbours are added to the trial list

in the narrow band structure (Figure 4.6). It is very important to select properly a grid point in the narrow band to be updated. This can be done using the heap sort algorithm. Every new accepted value is inserted into a sorted list of the trial points values. Therefore, finding the next point to update takes almost no time [113].

Equation 4.21 can be solved using the following numerical scheme

$$\left[\max(D_{ij}^{-x}T, -D_{ij}^{+x}, 0)^2 + \max(D_{ij}^{-y}T, -D_{ij}^{+y}, 0)^2\right] = \frac{1}{F_{ij}^2}, \tag{4.22}$$

where D^+ and D^- are the forward and backward difference operators and F_{ij} is propagation speed at the point (i, j).

The algorithm proceeds as follows:

1. All points in the initial conditions are tagged as *Accepted*.
2. All points in the neighbourhood of the accepted points are tagged as *Trial*.
3. The remaining grid points are tagged as *Far*.
4. The begin loop: find a *Trial* point with the smallest value of T.

5. The chosen *Trial* point is added to *Accepted* and removed from the *Trial* list.
6. All neighbours of the point chosen in Step 4 that are not *Accepted* are tagged as *Trial*.
7. The values of all new *Trial* points are recomputed according to Equation 4.22.
8. Return to Step 4.

4.4.2 Fast Marching Segmentation

This method has been applied to a wide range of problems including image segmentation [78, 143, 149], shortest path finding [27, 144] and other problems [3, 99, 121].

Because of its unidirectional propagation the Fast Marching Method is not very often applied to video segmentation. Nevertheless, some applications can be found in the literature [124, 127, 129]. An example of such an application is the method presented by Sharma and Reilly [122], based solely on colour. It is intended as a video segmentation method but it is in fact static image segmentation since no motion or frame difference is used. Here colour decides about propagation speed for the Fast Marching Method. Pixels from the image $I(x, y)$ are classified into four classes. Each class has a priority assigned. The pixels that are most likely to be a face have the priority 4 while those forming a background the priority 1.

Here a typical problem with fast marching used in a segmentation task is also present. Namely, due to one-way propagation of the contour, it tends to overshoot the desired object boundary. To overcome this problem, the authors proposed the following stop condition: The percentage of pixels with the priority 4 and belonging to the level set are computed and compared to the threshold value. If the result is above the threshold, it is assumed that a solution was reached.

To make colour invariant of lighting conditions and shadows, the YCbCr colour space was used. Choosing this colour space has some advantages. It gives a convenient separation of colour and intensity and in many cases it is the stan-

dard working space (e.g., in the MPEG and JPEG compression). Additionally, it was shown that skin tones fall into a quite narrow range of the Cb and Cr values. However, a problem with the missegmentation of the objects with colour similar to a face (e.g., wooden furniture) was also reported.

The idea of the multi-label fast marching method was first introduced by Sifakis and Tziritas in [124, 125]. The motion of the objects in the scene is detected on the basis of frame differences computed with statistical methods. Initially, pixels are divided into three classes: changed pixels, unchanged pixels and uncertain pixels.

After labelling two sets of pixels as *static* and *mobile*, the label propagation is started using the fast marching algorithm. Both regions grow to an unlabelled space with respective speeds. Propagation speeds for both classes are defined explicitly. The algorithm stops propagation when two regions meet on the object boundary and there is no unlabelled pixel left.

Because each of the two classes has an individually defined speed, extension to more classes is not straightforward. It is impossible to segment the sequence to an unknown number of segments using this approach.

4.5 Summary

Most active contour methods require clear distinction of properties for object interior and exterior. As was shown in this chapter, a typical solution to this problem is checking if the point is moving or not. This is possible when frame background is stationary or motion compensated. A method that deals with the problem of a moving background and offers high segmentation quality will be presented in Chapter 5.

Another problem with active contour segmentation is that it divides the frame into two areas, namely, the foreground and the background. Even when the foreground can consist of several separated parts, these parts are undistinguishable. When two foreground objects overlap, they are detected as one. A method of ac-

tive contour segmentation that can divide the frame into multiple objects will be presented in Chapter 6.

Chapter 5

Two-Step Object Segmentation

The Fast Marching Method (see Section 4.4) is a very efficient end elastic algorithm; nevertheless, it has an essential drawback. Namely, contour propagation can be performed only in one way. This makes the correction of wrongly segmented objects impossible, for example, when the contour leaks inside the object. This problem was also encountered by other authors (see Section 4.4). As a consequence, most researches [23, 82, 83, 101] prefer the slower Level Set Method because it permits bidirectional contour propagation. Here, a method will be presented for this kind of problems.

To exploit the speed of the Fast Marching Method and to avoid its drawbacks at the same time, **an original two-step method will be proposed in this dissertation**. In the first step, an image from a sequence will be coarsely but fastly segmented using the Fast Marching Method. A stop condition for this step must be defined in such a way so that the contour stops at some distance from the expected object boundary. This prevents the contour from leaking into the object. Such an error cannot be corrected since the Fast Marching Method can move the contour only in one direction. The second step is intended to improve segments from the first step by merging them with the results of colour-based static segmentation. In static segmentation, the edges of segments are well correlated with real object edges, but the image is usually oversegmented. Static segmentation cannot detect semantic objects without some knowledge of a high

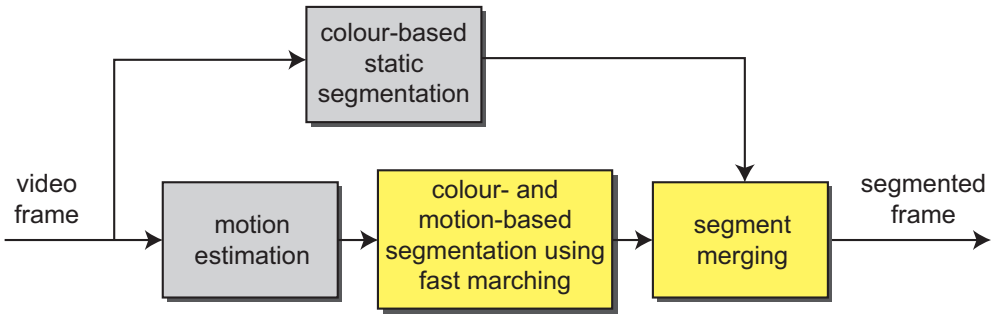


Figure 5.1: Architecture of the segmentation algorithm. Yellow blocks denote originally developed parts

level. In a typical video segmentation, only low level features are available. The speed of the static segmentation algorithm is a secondary matter, since segmentation is essential only in the contour neighbourhood; the segmented area may be significantly reduced.

A general block diagram of the method is presented in Figure 5.1. Blocks marked in gray use the standard methods, whilst parts represented by yellow blocks were developed in this work. Such a construction of the algorithm permits simple replacement of parts of the algorithm as well as pipelined and parallel processing.

5.1 Colour and Motion-Based Segmentation Using Fast Marching

The key element in the Fast Marching Method that allows adapting it to many problems is the contour propagation speed F (see Section 4.4). In this work, the speed function F is defined as a product of two terms, i.e., an image-related term F_1 and a curvature-related term F_2:

$$F = F_1 F_2. \tag{5.1}$$

In the original Fast Marching Method, there is no curvature-dependent term. A curvature term that is typically used in the Level Set Method (Equation 4.6) can have both positive and negative values and therefore cannot be used directly.

Propagation speed in the fast marching algorithm must be of a constant sign. The multiplication of the propagation speed F_1 with the positive curvature term F_2 allows keeping this condition. Here the neutral element of the curvature equals 1. Values from the range $\langle 0, 1)$ will slow down the propagation, while values greater than 1 will speed it up. A more detailed explanation of this aspect will be given in Section 5.1.2.

The next sections will give definitions of propagation speed components. Also, there will be proposed an automatic initialization method as well as a stop condition for the first stage of initial segmentation.

5.1.1 Image-Based Term

The definition of an image-based part of propagation speed is an important issue for the method. The detection of the moving object cannot rely solely on the motion field, since in many cases motion estimation methods produce erroneous motion vectors. On the other hand, using only image properties for the segmentation of semantic objects is inadequate when both the object and the background are highly textured. The main idea of this method is to exploit the advantages of motion and image segmentation. As was shown in Section 2.3, the smoother the image, the less reliable the motion vectors. Usually, in smooth areas of moving objects, motion vectors are zero or have random values due to ill-posed numerical problems. On the contrary, motion can be estimated well near object edges and on highly textured areas. If both the object and the background are highly textured, the object boundary is more difficult to detect using only image properties.

Considering the above observations, the properties of the speed function can be defined as follows:

- contour propagation speed should be high for uniform image areas,
- for uniform image areas, the influence of motion values should be negligible,

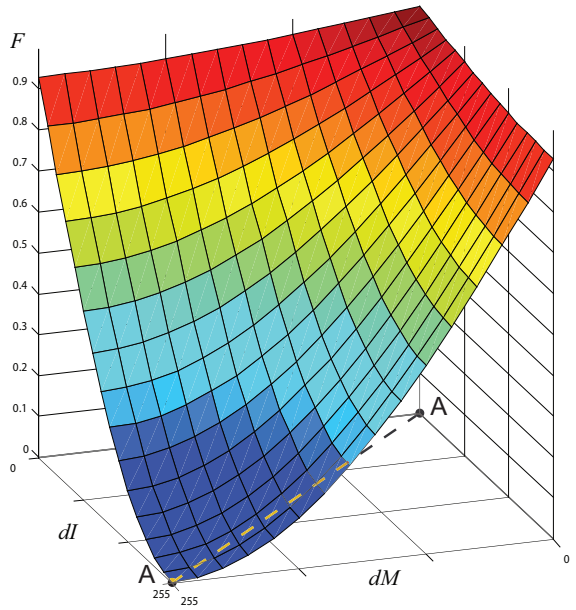


Figure 5.2: Profile of propagation speed defined with Equation 5.2

- propagation speed should be high for areas with a uniform motion field,
- where motion is uniform, image properties should have no influence on propagation speed.

Propagation speed defined with the above properties should be low on the object boundary and high for the rest of the image. The first attempt at defining the speed function presenting these properties was made in [127, 128]. The image-dependent part of the speed was defined as follows:

$$F_1(x,y) = 1 - \left(\cos\left(\frac{\pi/2}{1+dI(x,y)}\right)\cdot\cos\left(\frac{\pi/2}{1+dM(x,y)}\right)\right), \qquad (5.2)$$

where dI and dM are image and motion differences, respectively. The definitions of these differences will be provided later in this section. An S-shaped cross-section (Figure 5.3) of the speed profile should help in precise object detection. Such a function has good properties and the detection of objects was good, and yet the speed was not satisfactory due to complex calculations. Additionally, image and motion differences had different ranges of values, i.e., the range of the motion difference was only a fraction of the image difference. As a result, the speed was

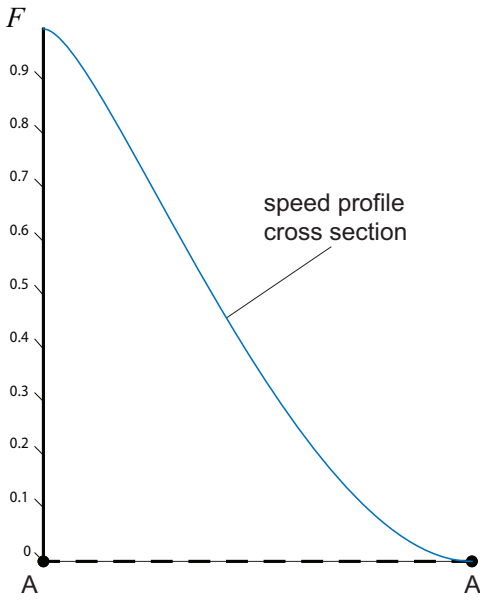


Figure 5.3: Cross section of the speed profile from Figure 5.2 along the A-A line

varying in the ranges $\langle 0.5, 1\rangle$. Segmentation quality was on a satisfactory level; nevertheless, the complex shape of the function made the calculations inefficient. It was necessary to find a function with similar properties but with a simpler definition.

Further research led to the speed function that was introduced in [130]:

$$F_1(x,y) = \frac{1}{min(\alpha \cdot dI(x,y), \beta \cdot dM(x,y)) + 1}, \tag{5.3}$$

where α and β are normalizing coefficients. Such a function fulfils all conditions given at the beginning of this section and is defined in a much simpler way than the function given in Equation 5.2. The general form of speed is similar to those presented in [125]. However, instead of statistical properties of the image, colour and motion differences were used.

The maximum value of the colour change depends on image representation in computer memory and is fixed. From the definition of image difference (Equations 5.9, 5.10 and 5.11), it can be shown that the maximum image difference has

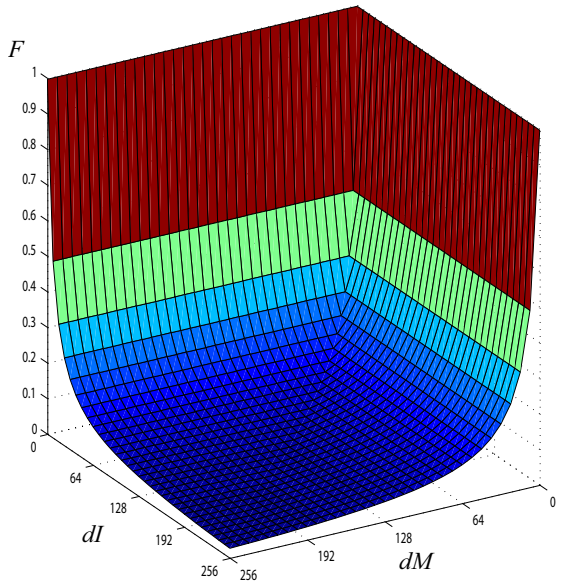


Figure 5.4: Profile of propagation speed defined with Equation 5.5

the following value:

$$dI_{max} = 255^2\sqrt{2}. \tag{5.4}$$

The maximum values of differences in luminance and chrominance are equal to 255 and $255\sqrt{2}$. The difference in chrominance is defined as the length of a vector with endpoints defined by colour values of two points in the C_b, C_r plane (Equation 5.11). The maximum value results from Equation 5.9, where both values are combined by multiplication.

The maximum value of the difference in motion can vary from sequence to sequence. This assumption helps to simplify Equation 5.3 and leave only one normalizing coefficient:

$$F_1 = \frac{1}{\min(dI, \alpha \cdot dM) + 1}. \tag{5.5}$$

To keep the sense of function min from Equation 5.5, the values of dM must be mapped to the same range as the image difference by multiplying them with the α coefficient. This parameter is measured during the motion estimation phase based on the minimal and maximal values of horizontal and vertical motion components.

Finally, the definition of the normalizing coefficient has the following form:

$$\alpha = \frac{dI_{max}}{dM_{max}}, \tag{5.6}$$

where dM_{max} is the maximum motion difference calculated according to Equation 5.12. As input values, two extremum displacements are used. These displacements are found during the motion estimation process.

Computation of Image Differences

Image difference estimated by typical convolution filters is not suitable for contour speed computation. In such a difference image, edges are not well defined and are influenced by noise. Applying some noise reduction techniques such as smoothing or median filtering leads to creating gaps on the edges. The fast marching algorithm is very sensitive to such defects. Even a small gap on the object edge can lead to the leaking of the contour inside the object. This kind of segmentation error cannot be corrected since the Fast Marching Method can propagate the contour in one direction only. Using more sophisticated edge detectors such as the Canny operator [16] can improve the weak edge definition, but on the other hand, the problem of gaps in the object boundary still exists.

In order to overcome problems with gaps it was necessary to calculate differences in a way that will prevent the contour from leaking into objects through small gaps. The calculation procedure proposed in [128] is based on comparing image properties from some area visited by the contour to those that are at the front of the contour. The definition of the luminance component is expressed as follows:

$$dLum(x,y) = \left| \frac{\sum_{\mathcal{W}} Y(x,y)}{|\mathcal{W}|} - Y_\sigma(x,y) \right|, \tag{5.7}$$

where $\mathcal{W}$ is a set of pixels that were visited by the contour and are covered by a certain window centered at (x,y) (Figure 5.5), and Y_σ is the luminance value from the current frame blurred with the Gaussian convolution kernel in order to reduce the influence of noise. The radius of the window and the radius of the convolution

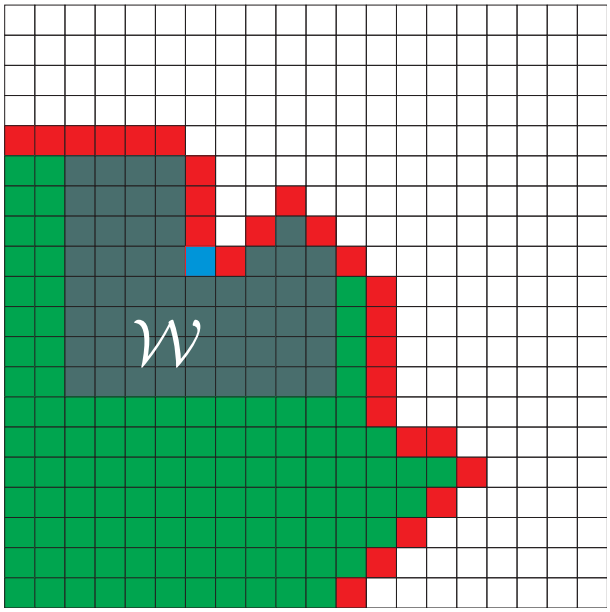


Figure 5.5: Difference operator that compares the current point with the area visited by the contour

kernel are the same. This helps to avoid contour leaking through gaps that are smaller than the window diameter. However, in some cases, where propagation direction was parallel to the object boundary, edges were not detected properly and the results of the segmentation were wrong.

To avoid the problem described above, the object edge should be detected even if the active contour is in a perpendicular position to the object edge. The most desirable situation is when the contour is parallel to the object edge. The propagation of the active contour can be turned towards the object by slowing down the parts of the contour that are close to the object boundary while the remaining part propagates with higher speed.

To achieve the behaviour described above, a difference operator that aligns itself to contour orientation was introduced by the author in [130]. The shape of the operator forms a four arms cross, where each arm covers a 1/8 section of a circle (Figure 5.6). The arms of the operator marked with a are always aligned to the normal direction while the pair b is aligned with a tangent.

The term dI is computed using the curve-oriented difference operator and is

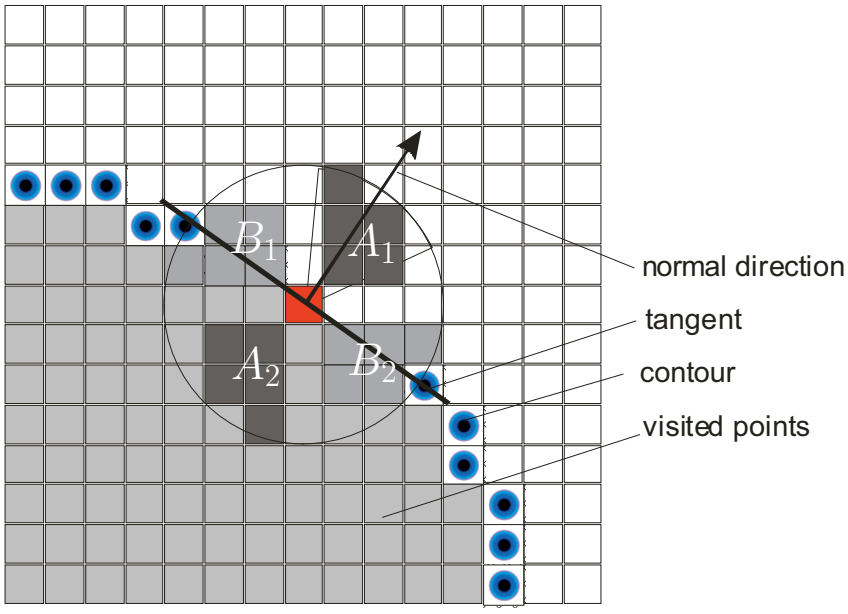


Figure 5.6: Curve-oriented difference operator

defined as follows:

$$dI = \max(dA, dB), \tag{5.8}$$

where dA is the difference between the parts A_1 and A_2 while dB is the difference between the parts B_1 and B_2. Both differences are defined analogically, thus only a definition for the dA will be given.

Every part of the cross gives an average of several pixels to reduce noise influence. The difference dA is composed with the differences in luminance $dLum$ and chrominance $dCrom$:

$$dA(x, y) = dLum(x, y) \cdot dChr(x, y). \tag{5.9}$$

The above definition promotes 'strong' edges, i.e., edges that appear in luminance as well as in chrominance. This prevents the edges of the shadows from being detected as the object edges. The difference in luminance for the parts A_1 and A_2 of the difference operator is defined as follows:

$$dLum(x, y) = \left| \frac{\sum_{A_1} Y}{|A_1|} - \frac{\sum_{A_2} Y}{|A_2|} \right|, \tag{5.10}$$

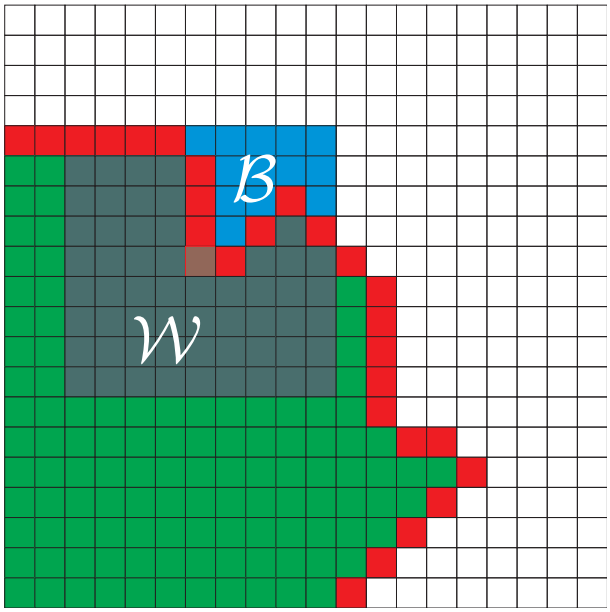


Figure 5.7: Areas on which the difference dM is computed

where A_1 is the part at the front of the contour, A_2 is the part behind the contour, Y denotes the luminance component and $|.|$ denotes the number of elements in the set. It is an absolute difference between the mean values of the luminance for the parts A_1 and A_2 of the difference operator.

Since chrominance components form vectors on the C_bC_r plane, a good measure of the difference among them will be the length of the vector that is a result of vector subtraction of components from the parts A_1 and A_2. To reduce noise influence, a values of the C_b and C_r are averaged over the A segments of the difference operator. Finally, the difference in chrominance can be written as

$$dChr(x,y) = \left| \left(\frac{\sum_{A_1} C_b}{|A_1|}, \frac{\sum_{A_1} C_r}{|A_1|} \right) - \left(\frac{\sum_{A_2} C_b}{|A_2|}, \frac{\sum_{A_2} C_r}{|A_2|} \right) \right|. \tag{5.11}$$

The term dM should be robust and resistant to noises in the motion field. The most important information is how the motion of the pixels behind the contour differs from the motion of the pixels at the front. Therefore, the motion difference is computed by a comparison of mean motion vectors at the opposite sides of the curve and as a sum of absolute differences between the horizontal and vertical

motion vector components:

$$dM = \left| \frac{\sum_{\mathcal{W}} V_x}{|\mathcal{W}|} - \frac{\sum_{\mathcal{B}} V_x}{|\mathcal{B}|} \right| + \left| \frac{\sum_{\mathcal{W}} V_y}{|\mathcal{W}|} - \frac{\sum_{\mathcal{B}} V_y}{|\mathcal{B}|} \right|, \tag{5.12}$$

where $\mathcal{W}$ is the set of pixels that form the part of the window placed at (x, y) that is currently located inside the curve, $\mathcal{B}$ is the set of pixels that forms the part of the window outside the curve, V_x and V_y are the horizontal and vertical motion components, respectively (Figure 5.7). A square window was used to simplify the calculations. Anisotropy introduced by the window shape does not have a significant influence on segmentation quality. Moreover, its regularity makes memory operations more efficient than for a circular window.

A sample evolution of the contour according to the method presented in this section is shown in Figure 5.8.

5.1.2 Curvature-Based Term

In the original Fast Marching Method [118, 119], there is no curvature-related term. However, practice has shown that without smoothing the contour used for segmentation may become too contorted. In such a case, the length of the narrow band increases radically, which implies a loss of efficiency. The idea of the smoothing term was borrowed from the original Level Set Method, but underwent an essential modification. In the Level Set Method the smoothing term is summed with the main term. This works when speed can change the sign. The smoothing term with the same sign as the main term increases speed whereas that with the opposite sign reduces it. In the Fast Marching Method, speed must have a constant sign, let us say, a positive one. When trying to use the same idea it is possible to obtain in some cases a negative speed, which is invalid. Moreover, if the term F_2 was limited to non-negative values, it would only be able to increase contour speed.

It was necessary to include the smoothing term in another way. The inclusion procedure should have a neutral element and allow increasing and decreasing propagation speed. Using multiplication allows keeping the essential features of

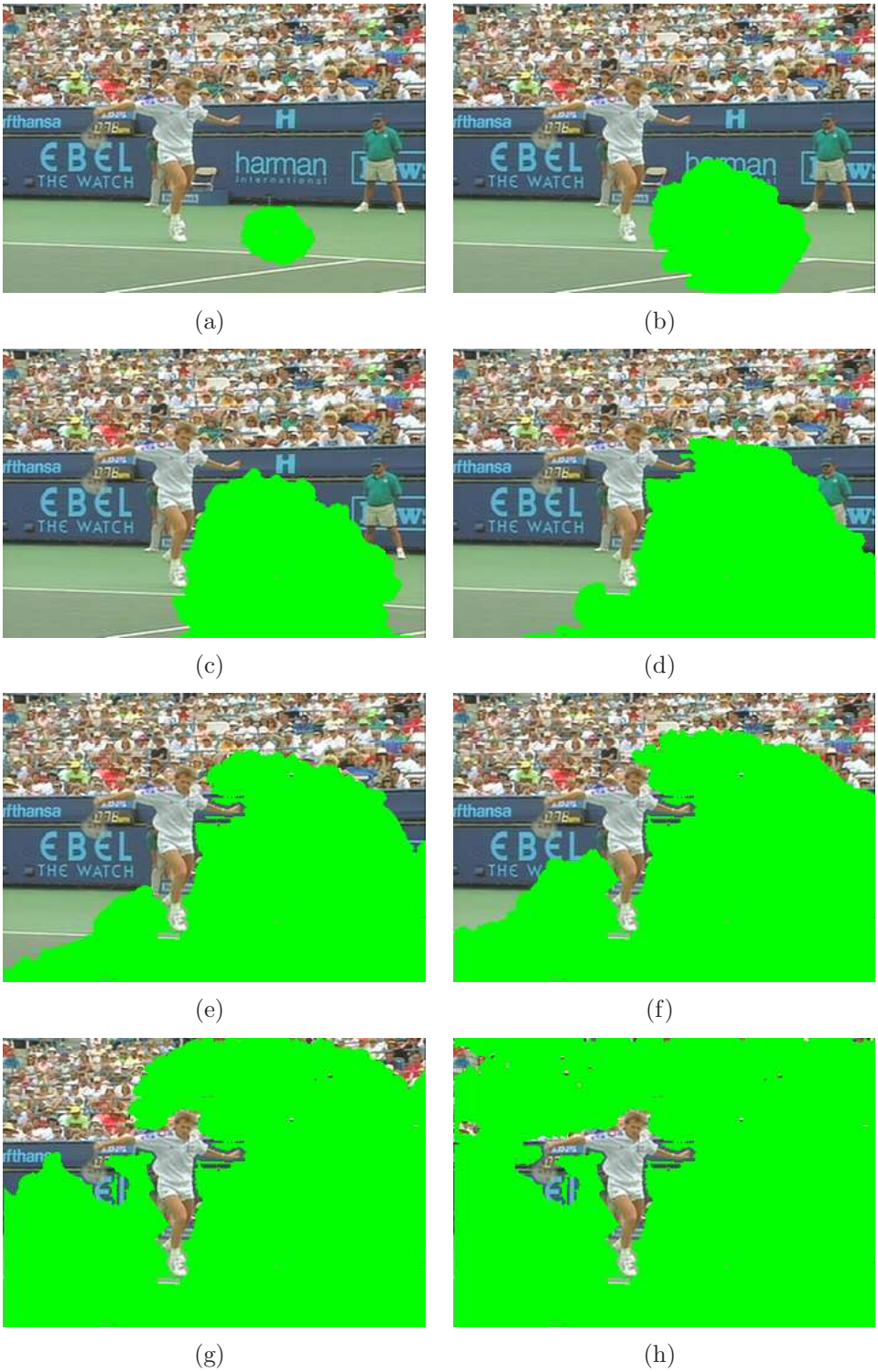


Figure 5.8: Curve evolution computed by the Fast Marching Method

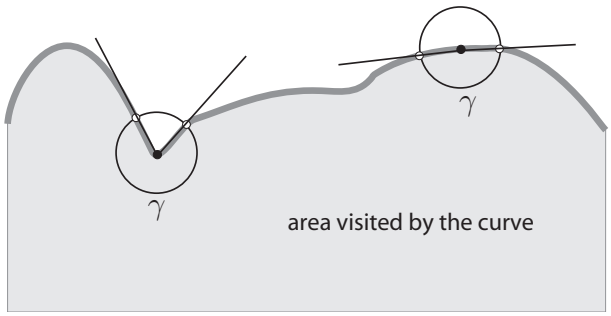


Figure 5.9: Local curvature definition. The angle (γ) is measured at the side already visited by the curve

the smoothing term, which can be found in the Level Set Method. F_2 is neutral when it has the value 1, it slows down the propagation for values less than 1 and increases speed for values greater than 1. A detailed description of the calculation of the curvature-based term is given in the subsequent section.

The standard curvature definition given by Equation 4.6 is not applicable to to the Fast Marching Method because it gives positive as well as negative values. This section offers a proposition of a new way of calculating the local curvature. Since the propagation of the contour does not have to be strictly curvature-dependent like in some level set applications [79, 97, 116], precise calculation is not required. In this case, the curvature term acts only as a smoothing term.

At the point of the curve where the component F_2 has to be estimated, a circle of a small radius is placed (Figure 5.9). The angle defined by the circle centre and curve intersection points defines the local curvature:

$$F_2 = \begin{cases} 0.1, & \text{if } \gamma < 90°; \\ 10, & \text{if } \gamma > 270°; \\ 1, & \text{in other case.} \end{cases} \tag{5.13}$$

The curvature-based term F_2 should have values below 1 for cusps and a value greater than 1 for corners. A simplified local curvature estimation is proposed to keep the performance of the algorithm at the high level. This term also helps to keep the contour as short as possible. The shorter the contour, the smaller the number of elements in the narrow band list to be sorted.

Alternatively, a non-negative curvature can be defined on the basis of the

Figure 5.10: Regions with the dfd value equal to zero

curvature from Equation 4.6. The definition of the positive curvature κ_p is as follows:

$$\kappa_p = \begin{cases} 1+\kappa, & \text{if } \kappa \geq 0; \\ \frac{1}{1+|\kappa|}, & \text{in other case.} \end{cases} \tag{5.14}$$

5.1.3 Initialization

Typically, video segmentation algorithms based on active contour methods are initialized at boundary pixels of a frame [40, 50, 122]. This is a very convenient assumption but objects that are partially visible in the frame may be not detected. This happens because the initial contour intersects such objects as well as the background.

The application of the Fast Marching Method [2, 117, 119] makes it possible to start in any point of the background and segment out object thanks to the possibility of performing a contour topology change. It is even possible to make foreground-background segmentation in presence of multiple moving objects.

A serious problem is to find automatically a good initialization point [131]. An initialization procedure based on displaced frame difference estimation will be proposed. It is quite possible that the largest, uniformly moving region will be the background region. The displaced frame difference at the point (x, y) is

computed as follows:

$$dfd = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1}|I_n(x+1, y+j) - I_{n-1}(mx+i, my+j)|, \tag{5.15}$$

where mx and my are the position of the point in the previous frame according to motion information and n is the frame number. Such a computation procedure has two purposes: reducing image noise influence and rejecting single points as starting region candidates. All connected pixels with the dfd value equal to zero are connected as one region (Figure 5.10). As the initialization point, the center point of the largest region is chosen.

5.1.4 Stop Condition

In typical applications of the active contour, the stop condition is defined depending on the energy value calculated along the whole contour. Moreover, in the classical geodesic active contour and level set methods calculations are performed along the whole contour even when some parts of the contour have reached the solution. In the implementation of the Fast Marching Method presented in this paper, the stop condition is defined locally. This means that those parts of the contour which reached solution are no longer considered in the calculation. When the speed of the contour at a certain point drops below the threshold value F_{th}, this point is considered to be within the solution. A good threshold estimate is the standard deviation of speed function values computed for the whole image:

$$F_{th}^2 = \frac{1}{n}\sum_{i=1}^{n}(F - \bar{F})^2, \tag{5.16}$$

where n is the number of points in the image, F is the contour speed at the n-th point and $\bar{F}$ is the mean speed in the image. A similar approach can be found in [46]. The points that have reached a solution do not propagate (the narrow band is not updated for them) and are removed from the narrow band. This approach allows reducing the length of the narrow band list and improving the performance of the algorithm. The algorithm stops when the narrow band list is empty.

The verification of the speed threshold calculated with Equation 5.16 was performed experimentally. The results of the experiment are shown in Figures 5.14 and 5.15. The contour stops at a proper distance from the object for threshold values around the computed value.

5.2 Enhancement Step

The first step of the presented algorithm stops the contour before the object boundary. This is necessary because the Fast Marching Method cannot move the contour back. If the object boundary is passed by the propagating curve at any point, the curve will leak into the object connecting it with the background.

In the second step of the algorithm, segment borders are moved towards the nearest edge detected in the picture by static image segmentation. The segmentation of the whole image is not necessary. Only the part that is contained in the band around the curve developed in the first stage is considered. The thickness of the segmentation area around the contour must be wider than the diameter of the difference operator described in Section 5.1.1. For simple and fast static segmentation algorithms, the segmentation of the whole frame will be effective enough and limiting the surface area will not be necessary. In most tests, a watershed was used as a static segmentation algorithm. This algorithm was chosen because

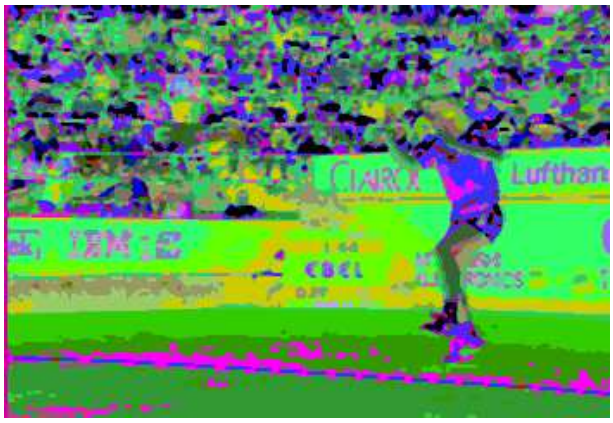
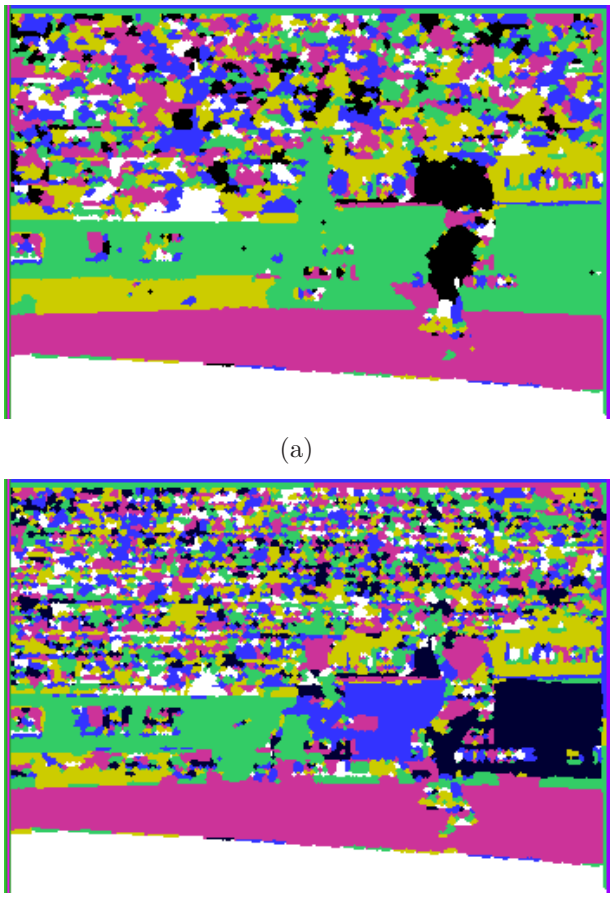


Figure 5.11: Segmentation by histogram clustering



(a)

(b)

Figure 5.12: Watershed segmentations with different levels

it allows controlling the size of the resulting segments. Thanks to this, the entire method was tested for different levels of segmentation. Example segmentations of the sample images are shown in Figures 5.11, 5.12(a) and 5.12(b).

After static segmentation is performed, its result must be merged with the result of fast marching segmentation. If it is assumed that a full frame is segmented, static segmentation can be done parallel to fast marching segmentation. It can be very convenient if computation time for both methods is similar. Since the whole method is intended to work in a real time, the merging procedure must be simple. Because the contour propagating in the first step stops at some distance from the object edge, it can be assumed that it has no common parts with the segments that belong to the object. This assumption simplifies the merging procedure to

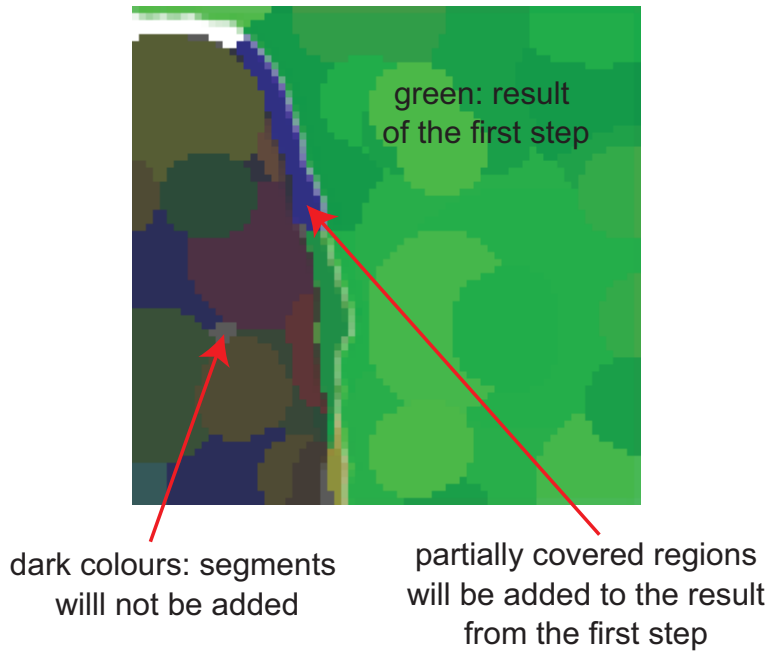


Figure 5.13: Illustration of the merging procedure

the Boolean union of the result from the first step and the segments produced in the second step (Figure 5.13). Namely, let sr_i be an i-th region from static segmentation and fmr a region from fast marching segmentation. The merging procedure can be written as follows:

```
for i=0 to n
if sr_i ∩ fmr ≠ ∅ add sr_i to fmr
```

where n is the total number of segments obtained from static segmentation. The segments with the common parts are connected.

5.3 Experiments

5.3.1 First Step, Threshold Estimation

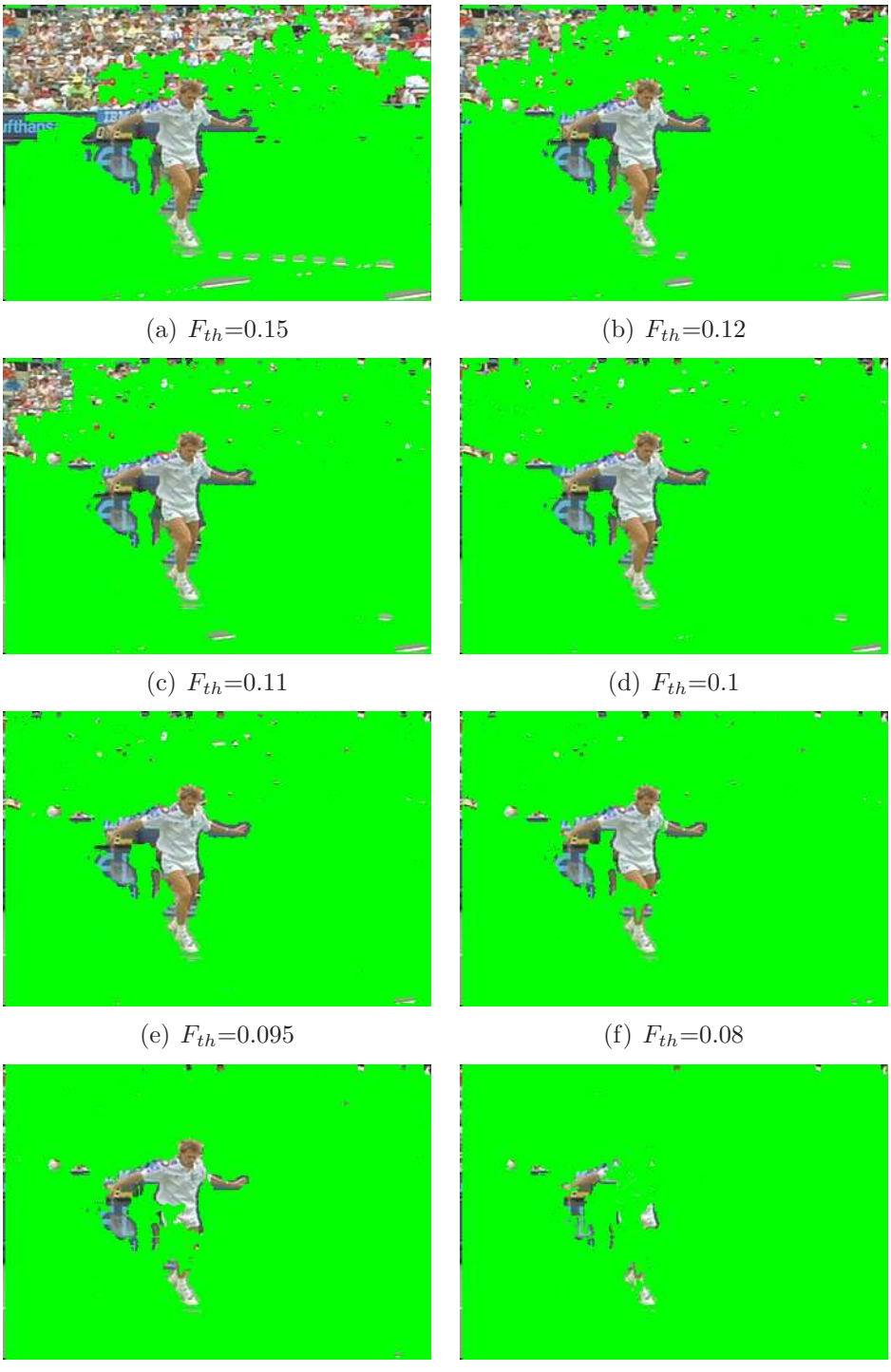
To ensure proper segmentation, the first step of the algorithm should stop propagating the segmenting contour at some distance from the expected edge. The stop condition based on the speed threshold defined in Equation 5.16 was tested experimentally. Segmentation with an automatically chosen threshold is good enough to ensure good results after the second step of the segmentation. Sample

results for a different level of the speed threshold are presented in Figures 5.14 and 5.15 to compare them with the threshold chosen automatically. The chosen threshold gave results suitable for processing in the second step of the algorithm. The contour is stopped at some distance from the object boundary. Some errors in the highly textured area are unavoidable because of the motion estimation errors. Most of them are usually corrected in the second step.

5.3.2 Quality of Final Segmentation

Example results of the segmentation are shown in Figures 5.16 and 5.17. Experiments shown that algorithm performs well when the motion can be estimated correctly. For the frames with large object displacements the segmentation is inaccurate because of the large motion estimation errors. More examples are included at the end of this chapter. The results of segmentation was assessed with some quality indices. Exemplary quality measures are shown in Table 5.1. A soft reference (SR) quality index was presented in Section 2.4.3. This index is compared with other indices that are using the binary reference. These indices are defined as follows:

- DPC – number of pixels that differ from reference segmentation,
- POD – percentage of DPC to the number of pixels in reference segmentation,
- MD – mean absolute distance between the contours of reference and the evaluated segmentation; the mean is calculated over all points belonging to the reference contour,
- MSD – mean distance between the contours of reference and the evaluated segmentation, where the displacement outside the reference is calculated as positive, while the displacement inside the reference is calculated as negative; the mean is calculated over all points belonging to the reference contour,



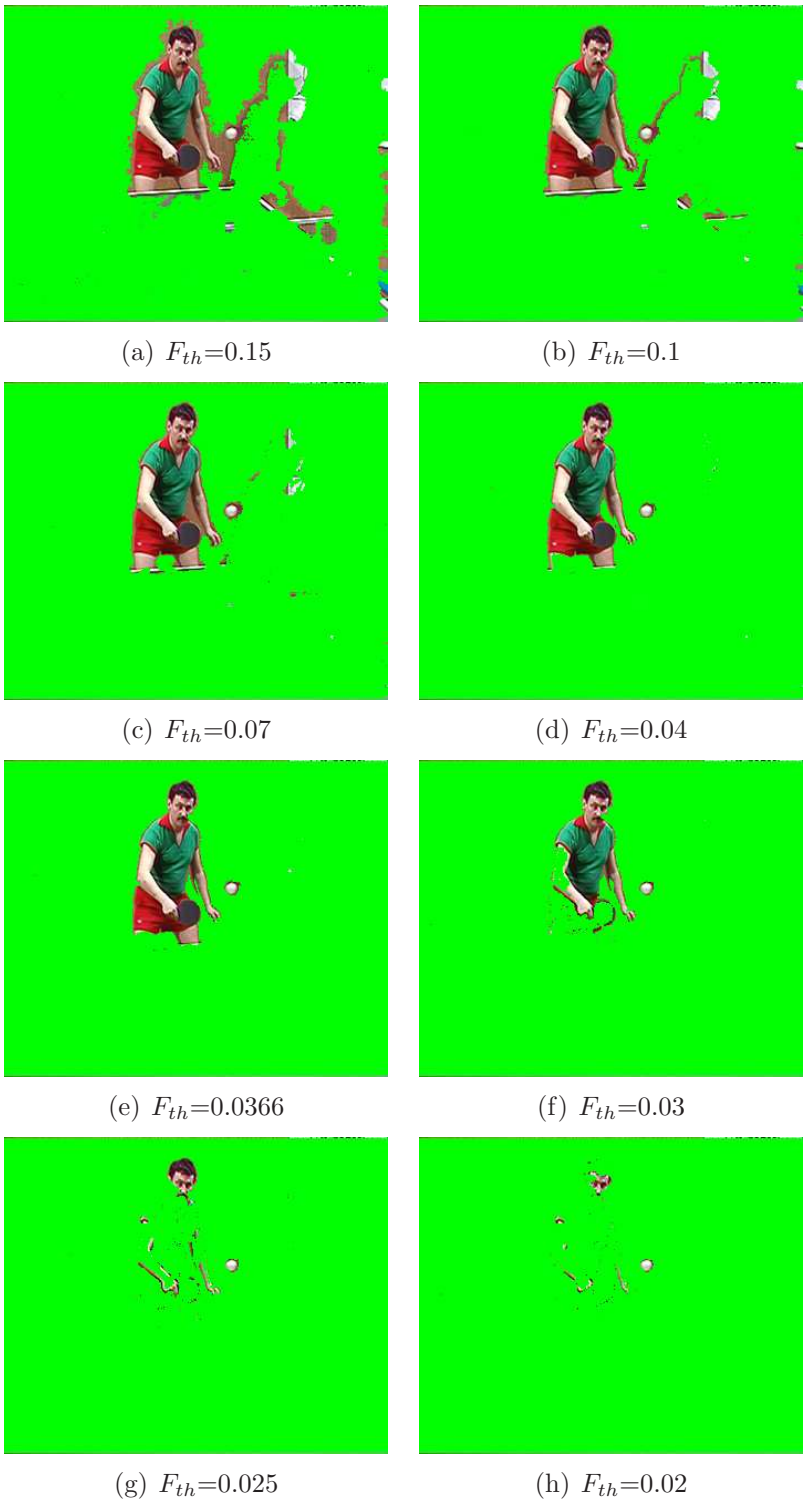
(a) F_{th}=0.15 (b) F_{th}=0.12

(c) F_{th}=0.11 (d) F_{th}=0.1

(e) F_{th}=0.095 (f) F_{th}=0.08

(g) F_{th}=0.07 (h) F_{th}=0.05

Figure 5.14: Results of fast marching segmentation (first step) of the sequence 'Stefan' for different levels of the F_{th} threshold. The computed threshold was equal to 0.095



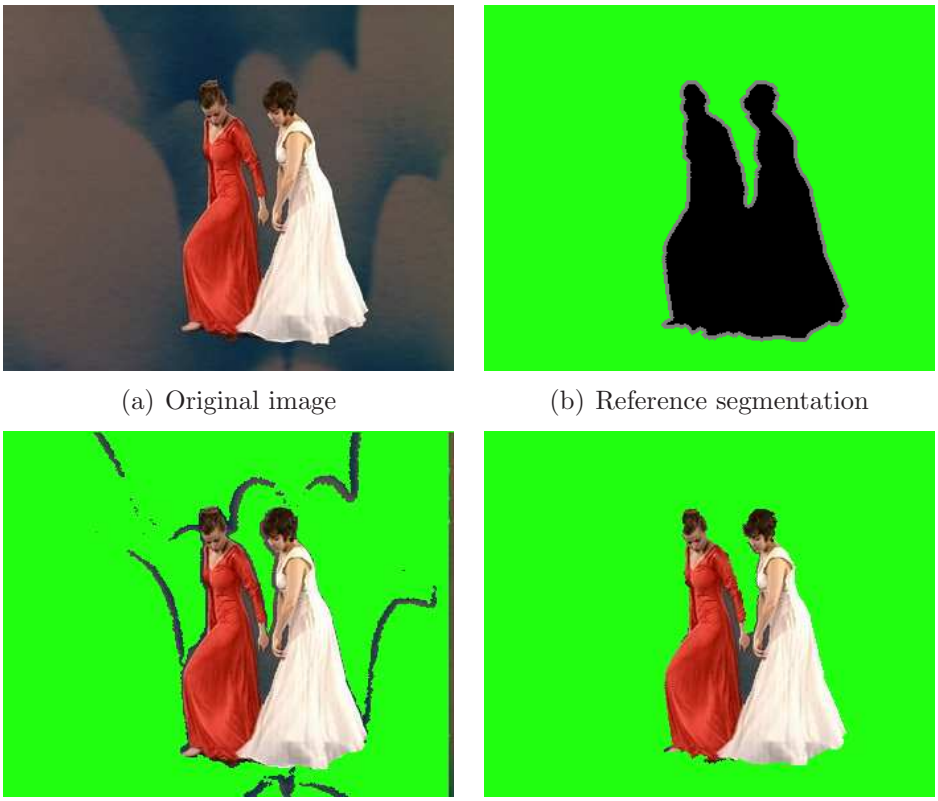
(a) F_{th}=0.15 (b) F_{th}=0.1

(c) F_{th}=0.07 (d) F_{th}=0.04

(e) F_{th}=0.0366 (f) F_{th}=0.03

(g) F_{th}=0.025 (h) F_{th}=0.02

Figure 5.15: Results of fast marching segmentation (first step) of the sequence 'Table Tennis' for different levels of the F_{th} threshold. The computed threshold was equal to 0.0366



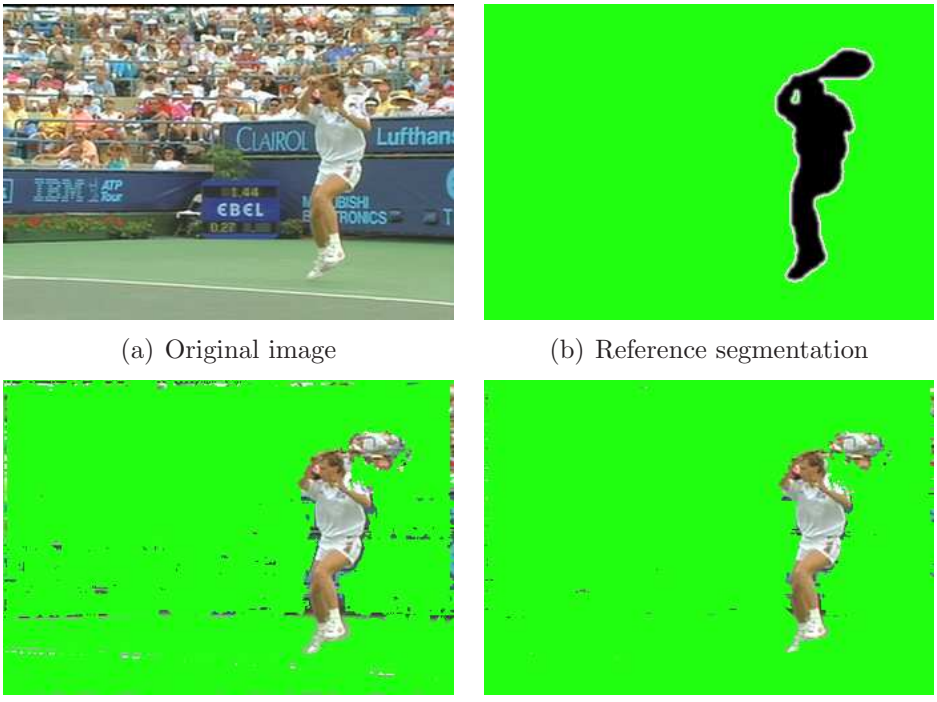
(a) Original image (b) Reference segmentation

(c) Segmentation after the first step (d) Segmentation after the second step

Figure 5.16: Frame 47 from the 'Dance' sequence segmented using two-step fast marching

- AD – absolute distance between the contours of reference and the evaluated segmentation, calculated as a sum of absolute distances over all points belonging to the reference contour,
- ASD – distance between the contours of reference and the evaluated segmentation, where the displacement outside the reference is calculated as positive, while the displacement inside the reference is calculated as negative, calculated as a sum of distances over all points belonging to the reference contour.

Unfortunately, most of the authors [18, 39, 62, 68, 70, 77, 82, 100, 102, 124] does not give any quantitative measures for their results. Even speed of the methods is seldom presented in publications. Usually only the pictures presenting segmen-



(a) Original image

(b) Reference segmentation

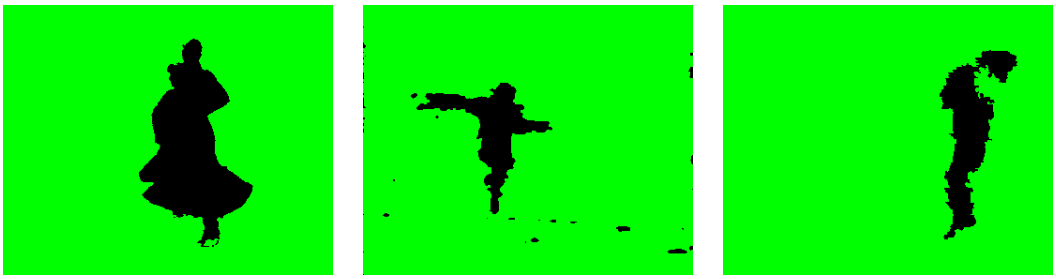
(c) Segmentation after the first step

(d) Segmentation after the second step

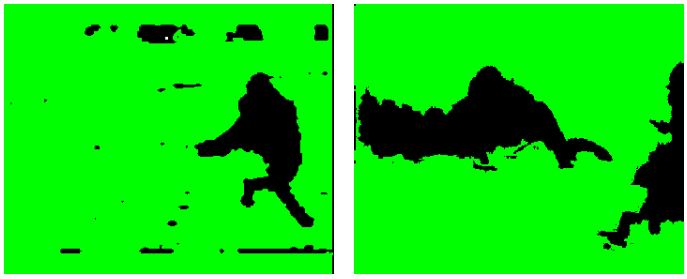
Figure 5.17: Frame 218 from the 'Stefan' sequence segmented using two-step fast marching

Table 5.1: Segmentation quality measured for exemplary frames. Frame number is in the braces. Quality indices: SR – soft reference, DPC – different pel count, POD – percent of difference, MD – mean distance, MSD – mean signed distance, AD – absolute distance, ASD – absolute signed distance

Seq. name	**SR**	**DPC**	**POD**	**MD**	**MSD**	**AD**	**ASD**
Dancer (100)	0.85	710	5.38%	0.81	0.43	687.72	367.25
Stefan (50)	31.23	1974	49.69%	3.2	-1.66	2137.8	-1108.78
Stefan (218)	37.96	2113	55.31%	3.83	-2.28	2171.43	-1165.87
Football (117)	13.56	3427	36.38%	4.39	-0.13	5841.56	-173.77
Football (340)	12.4	4769	20.76%	2.5	1.33	3941.53	2092.84



(a) Dancer (100). (b) Stefan (50) (c) Stefan (218)



(d) Football (117) (e) Football (340)

Figure 5.18: Segments evaluated in Table 5.1

tation results are available. This makes a precise comparison of the methods impossible. Instead of this, a functional comparison of some methods will be presented in Section 5.3.3.

When motion estimation fails, proper segmentation is impossible. Figure 5.19 shows such a situation. Segmentation presented in Figure 5.17 is less accurate because of motion estimation errors correlated with the highly textured background. Motion in the presented frame was too high for the implemented algorithms. It was observed that displacement higher then 3 pixels per frame was not detected correctly. Increasing aperture size in Lucas-Kanade algorithm could help in detecting larger displacement but at the cost of higher blurring of the object motion edges. However, for the tests, only basic motion estimation algorithms were used. Using a more precise estimation method would improve segmentation quality. Figures 5.20 to 5.25 show exemplary segmentations.It can be seen, that second step improved segmentation quality significantly. However, some small false objects was detected due to motion estimation errors.

(a) Segmentation after the first step

(b) Segmentation after the second step

Figure 5.19: Frame 200 from the 'Stefan' sequence segmented using two-step fast marching. Segmentation failed because of wrongly estimated motion

Table 5.2: Comparison of features from different segmentation methods. 0 – proposed method, 1 – method by Pargios [102], 2 – method by Sifakis [124], 3 – method by Konrad [70], 4 – modified K-means [69], 5 – DCT-based method [62]. + – feature exists, ∘ – partially exists, − – feature unavailable

	0	**1**	**2**	**3**	**4**	**5**
foreground-background segmentation	+	+	+	+	−	+
multiple objects	∘	∘	∘	−	+	∘
moving background	+	−	−	−	−	−
real time	+	−	−	−	−	+
colour segmentation	+	−	+	−	+	+
object tracking	−	+	−	+	−	−
overlapping objects	∘	∘	∘	−	+	∘
partially visible objects	+	−	+	−	+	+
high segmentation precision	+	+	+	+	+	−

5.3.3 Functional Comparison

In this section will be given a functional comparison of several video segmentation methods. Table 5.2 shows a breakdown of features available in the compared methods. The compared features are as follows:

- **foreground-background segmentation** – segmentation method divides frame into two classes only, without recognizing individual objects; minus sign informs that method is able to distinguish multiple objects,

- **multiple objects** – multiple objects can be detected in the scene; + means

that objects are marked individually by the algorithm; ∘ means that the algorithm can detect multiple, non-overlapping objects but they are not distinguished; − the algorithm can detect only one object,

- **moving background** – the algorithm can segment sequences taken with moving camera,
- **real time** – complexity of the method allows real-time implementation,
- **colour segmentation** – the method uses colour information during the segmentation process; − means that only gray-scale images are used,
- **object tracking** – the segmented objects can be tracked over time,
- **overlapping objects** – the method can detect overlapped objects; ∘ means that overlapped objects are detected as one object; − only one of the objects can be detected,
- **partially visible objects** – the method can detect objects that are not fully visible in the scene,
- **high segmentation precision** – the method is able to give pixel-level segmentation.

Most of the features are present in the method shown in this chapter. The presented method has a unique ability to segment objects over a moving background even when an object is only partially included in the scene.

5.3.4 Execution Times

The mean segmentation time for the sample set of sequences was 1250 ms on a computer with a 1.4 GHz AMD AthlonXP processor and 512 MB of RAM. Example timing for the different sequences is presented in Table 5.3. The speed of the segmentation was dependent on the length of the contour. If the contour was propagated smoothly, the segmentation time was shorter. If the segmented

object was complicated, the segmentation was performed for a longer time. The profiling of the code showed that 75% of the execution time was related to the sorting function used by the fast marching algorithm. It has to be mentioned that the code was optimized neither for speed nor for memory consumption and has a big overhead related to debugging. Nevertheless, the complexity of the method suggests a real time implementation. The current version does not use any advanced processing functions such as MMX or SSE. A significant performance gain could be achieved by rewriting the part responsible for fast marching propagation. Currently, sorting is realized using dynamically allocated objects. This solution is convenient from the application point of view but it is relatively slow. The Heapsort algorithm used in the Fast Marching Method can be implemented on static arrays. The maximum length of such an array is the number of pixels in the processed image. This gives the possibility of porting to the assembler all functions that are responsible for handling contour points. Additionally, fixed memory location gives the possibility of better memory cache utilization than in the case of dynamic memory allocation. Assembly language implementation of the most often executed functions can give a several times faster code. This could allow the method to approach the real time on the 1.4 GHz AMD AthlonXP processor. A bottleneck of the method is also the speed of the memory and system bus because of the large number of memory operations. This means that modern computers with fast memory are able to make calculations in a real time for sequences in the CIF resolution.

Table 5.3: Sample execution times of the proposed method for different sequences. The mean execution time was 1250 ms

sequence name	**frame number**	**execution time (ms)**
Table Tennis	7	1333
Table Tennis	91	892
Table Tennis	141	1359
Table Tennis	187	1347
Dance	24	1263
Dance	45	1230
Dance	96	1278
Dance	133	1202
Stefan	41	1239
Stefan	50	1323
Stefan	212	1301
Stefan	312	1228

(a) Segmentation after the first step (b) Segmentation after the second step

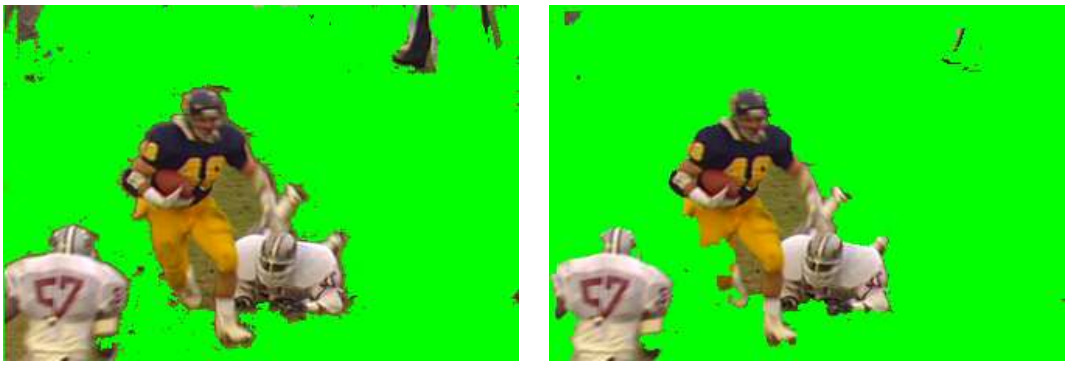
Figure 5.20: Frame 22 from the 'Basket' sequence segmented using two-step fast marching

(a) Segmentation after the first step (b) Segmentation after the second step

Figure 5.21: Frame 80 from the 'Basket' sequence segmented using two-step fast marching

(a) Segmentation after the first step (b) Segmentation after the second step

Figure 5.22: Frame 253 from the 'Basket' sequence segmented using two-step fast marching

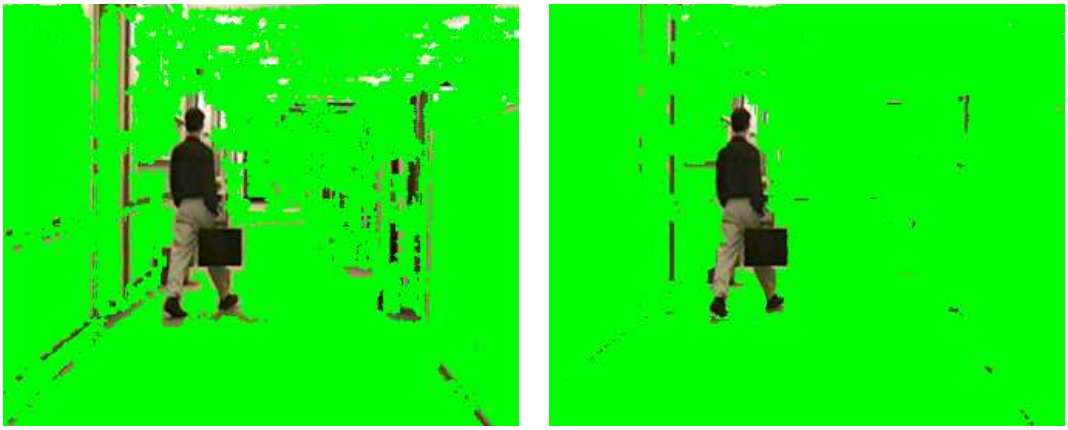


(a) Segmentation after the first step (b) Segmentation after the second step

Figure 5.23: Frame 215 from the 'Football' sequence segmented using two-step fast marching

(a) Segmentation after the first step (b) Segmentation after the second step

Figure 5.24: Frame 340 from the 'Football' sequence segmented using two-step fast marching



(a) Segmentation after the first step (b) Segmentation after the second step

Figure 5.25: Frame 43 from the 'Hall Monitor' sequence segmented using two-step fast marching

Chapter 6

Extraction of Multiple Objects Using Multi-Label Fast Marching

In some applications, foreground-background segmentation is not sufficient. The detection of all objects in the scene would be more appropriate. The original method that would be presented in this chapter is aimed at performing such a task. As the main tool the multi-label Fast Marching Method will be used. The idea of simultaneous propagation of multiple contours using the Fast Marching Method was introduced by Sifakis *et al* [124,125]. However the method presented in this chapter shares with the Sifakis approach merely the idea of multi-label fast marching. In the original method only two labels are used, and each of them has individual propagation speed. Such an approach is hard to extend to multiple object segmentation, especially when the number of objects is unknown. The approach presented here uses the same propagation speed for all labels, thus the number of labels does not influence algorithm work. Moreover, there is no limitation to a static or motion compensated background. An additional advantage of this approach is that it is easy to define the stop condition since the contours are propagating toward each other. In the original method, the algorithm stops when the contours meet. In the method presented in this section some additional actions can be performed when two segments meet.

6.1 Initialization

The initialization procedure is based on the displaced frame difference (*dfd*) between two consecutive frames and requires the dense motion field to be computed prior to the initialization. Here it is assumed that the motion field was computed using one of the already known methods, as was done in Section 2.3.3.

It is assumed that regions with zero *dfd* are likely to be inside objects presenting the same motion properties. The procedure is similar to that presented in Section 5.1.3. This makes such areas good starting points for contours propagation. The displaced frame difference at the point (x, y) is computed as follows:

$$dfd = \frac{1}{9} \sum_{i=-1}^{1} \sum_{j=-1}^{1} |I_n(x+1, y+j) - I_{n-1}(mx+i, my+j)|, \tag{6.1}$$

where mx and my are the position of the point in the previous frame according to motion information. Such a computation procedure has two purposes: reducing image noise influence and rejecting single points as starting region candidates. All connected pixels with the dfd value equal to zero are labelled as one region with an additional constraint on motion uniformity. This prevents the regions on the border between the object and the background with zero dfd but with different motion properties from merging into one region. Each region is assigned an individual label. Such regions will be seeds for contours propagated with the Fast Marching Method (Figure 6.1). The number of seed regions is always larger than the number of final segments.

6.2 Initial Segments Propagation

All segments initialized earlier are propagated outwards using a modified fast marching algorithm. The segment labels for the points visited by contours are positive integers. The trial points for each contour are marked with negative numbers of segment labels. All trial points from all segments are included into the same sorted list. Thanks to this, no additional time synchronization be-

Figure 6.1: Seed regions overlayed on the original frame from the sequence

tween the segments is required. This situation is naturally handled by the fast marching algorithm since it can propagate contours of any topology. At this stage of propagation, there is in fact no difference between the standard and the multi-label implementation apart from the fact that the new label for the trial point is inherited from the segment that propagates at the current algorithm step (Figure 6.2).

Propagation speed is based only on the current image properties. It is proportional to the inverse gradient value combined with blurred image components:

$$F = \frac{1}{\max(\nabla Y_\sigma, \nabla Cb_\sigma, \nabla Cr_\sigma) + 1}, \tag{6.2}$$

where σ denotes Gaussian blurring. Such a speed definition makes contour motion fast in smooth areas and slow as they approach edges. Thanks to this, contours are likely to meet on the object edge rather than inside the object.

6.3 Dynamic Regularization of the Motion Field

Since estimation starts from the points where motion vectors are estimated correctly, these correct vectors are propagated along with the contour and replace the originally computed motion vectors. This situation is correct until the contour

Figure 6.2: Segments during the initial stage of propagation

(a) (b)

Figure 6.3: Regularized motion field for the frame 190 from the 'mobile' sequence. a – horizontal component, b – vertical component

stays inside the object from which the propagation started. When the contour crosses the object border, motion vectors that belong to that object are assigned to the background or another object. Such a situation leads to two errors: segments are missaligned with real objects and some parts of the frame have erroneous motion vectors assigned. This situation will last until two contours meet. Section 6.4 will give a solution to the errors introduced in this step.

After segmentation is finished, every segment has a uniform motion field. The final result is shown in Figure 6.3.

Such a simplification reduces the application of this method to sequences with

translational motion of rigid objects. For example, traffic monitoring sequences fall into this group. Nevertheless, the application of a more sophisticated motion regularization method and using full motion information will extend the reliability of this method. This is possible since the only requirement of this segmentation method is motion consistency within the propagating segment.

6.4 Segments Merging and Pushing

The expansion of the segment described in Section 6.2 is performed as long as new trial points can be set on the area not visited by any of the propagating curves. When a new trial point is going to be set in a place occupied by a trial point from another segment, two actions can be performed: the segments can be merged or one contour can be pushed back by another.

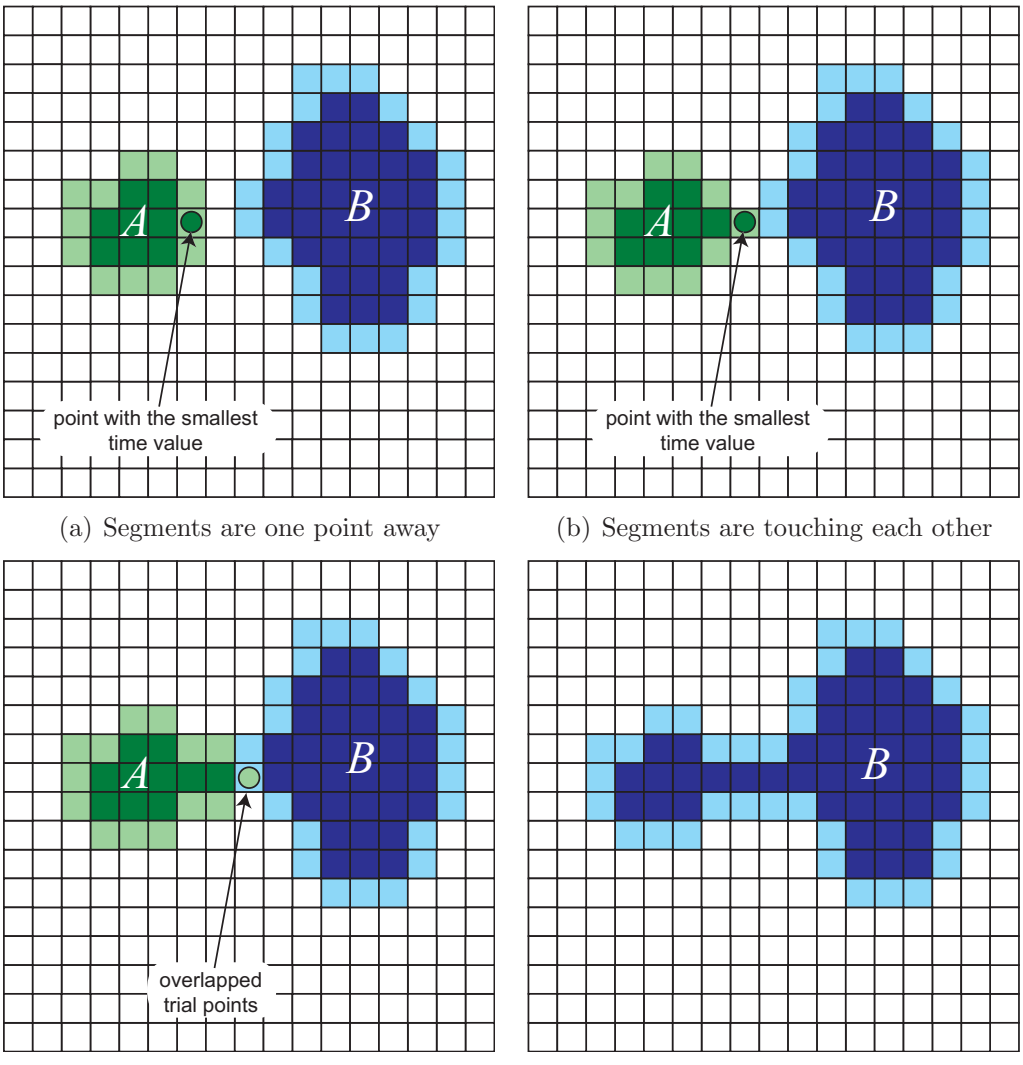
6.4.1 Segments Merging

When two segments meet, the motion of these segments is compared. The meeting point is a trial point from one segment that must be placed over a trial point from another segment (Figure 6.4(c)). Since motion within segments is the same for all points, it is sufficient to take one point from each segment for comparison. Motion from the segment A is compared with motion from the segment B according to the following expression:

$$|mx_A - mx_B| < \varepsilon \wedge |my_A - my_B| < \varepsilon, \tag{6.3}$$

where mx and my are motion vector components and ε is an empirically chosen merging threshold. During tests that were performed on a number of sequences, the best results gave $\varepsilon = 0.9$. This means that segments with motions different by less than one pixel per frame are connected. Motion vectors are estimated with sub-pixel accuracy. Additional research is needed to find a way of automatically adjusting ε. When the expression 6.3 is true, the segments A and B are merged.

To ensure maximum efficiency, labels from the smaller segment are changed



(a) Segments are one point away

(b) Segments are touching each other

(c) Trial point overlaps the existing one. Motion similarity is checked

(d) Segments merged together

Figure 6.4: Procedure of merging segments with high motion similarity. Trial points are marked in a brighter colour

to the value of those from the larger segment. Also, trial points from smaller segments are assigned with the value from larger segments (Figure 6.4(d)). Motion vectors from smaller regions are replaced with motion vectors from larger regions to ensure motion uniformity within segments.

6.4.2 Segment Pushing

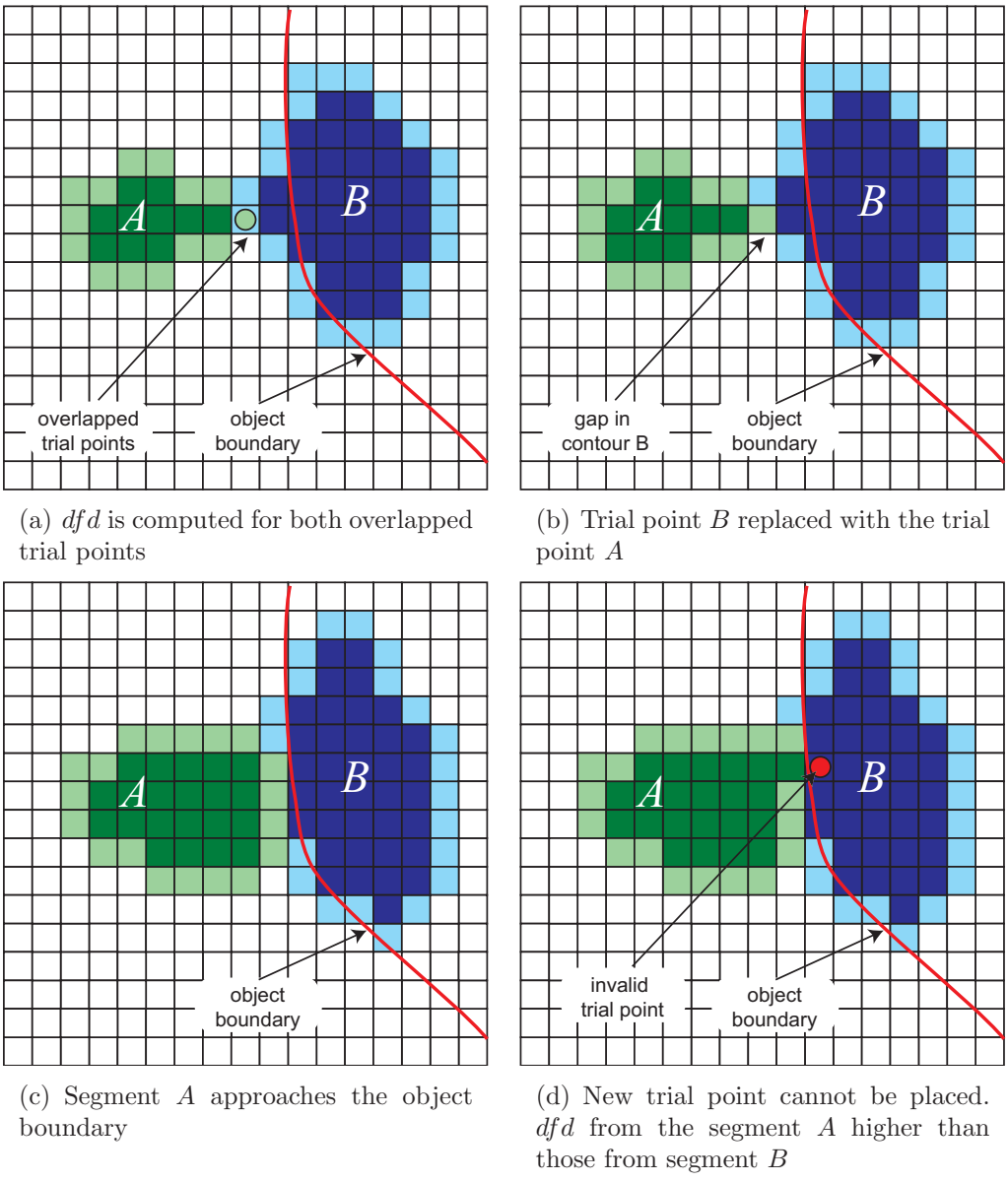
If two segments that meet are not classified to be merged, the propagating segment can push back another segment under certain circumstances.

When a trial point from the propagating segment A is going to be placed at the position (x, y) occupied by a trial point from another segment B (Figure 6.5(a)) and motion similarity is not high enough, then the displaced frame difference is computed for both segments:

$$dfd_A = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1}|I_n(x+1, y+j) - I_{n-1}(mx_A + i, my_A + j)|, \tag{6.4}$$

$$dfd_B = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1}|I_n(x+1, y+j) - I_{n-1}(mx_B + i, my_B + j)|, \tag{6.5}$$

where I_n is the n-th image from the sequence, mx, my are motion compensated positions of the pixels, and the indexes A and B denote the segment being the source of motion information. If $dfd_A < dfd_B$, then the trial point from the segment A replaces the trial point from the segment B. In the case when $dfd_A > dfd_B$, the trial point from the segment A is not placed and no further propagation is performed. The latter case means that the meeting point belongs to the segment B. At that point only the segment B has the possibility of propagating further, because the trial point from the segment A was not set. If the point considered lies on the object border, the segment B cannot propagate either because the trial point from B will have higher dfd than the point from A set earlier. The segment B can propagate further if the segment A passed its object border and the meeting took place on the object that belongs to B. The segment B will push back the segment A to the nearest object border.



(a) dfd is computed for both overlapped trial points

(b) Trial point B replaced with the trial point A

(c) Segment A approaches the object boundary

(d) New trial point cannot be placed. dfd from the segment A higher than those from segment B

Figure 6.5: Segment A with lower dfd pushes the segment B with higher dfd back to the object boundary

The replacement of the point from the trial list of the segment B creates a gap in the segment boundary (Figure 6.5(b)). Nonetheless, it has no influence on the further propagation of neither the segment B nor the segment A. The replaced point has no chance of propagating anyway because its dfd was higher than those of the segment A. The remaining portion of the segment B is propagated normally. The fast marching algorithm does not require a closed contour for propagation.

The segment A stops pushing back the segment B on the boundary of the object which has motion properties similar to those represented by the segment B. In such a case, the segment A cannot propagate further, because its dfd for the trial point that is going to be set inside the object occupied by the segment B will be higher than those for the segment B (Figure 6.5(d)).

When a contour has no possibility of propagating further, no new trial points are set. This implies the reduction of the total length of the sorted list used by the fast marching algorithm and the same performance improvement.

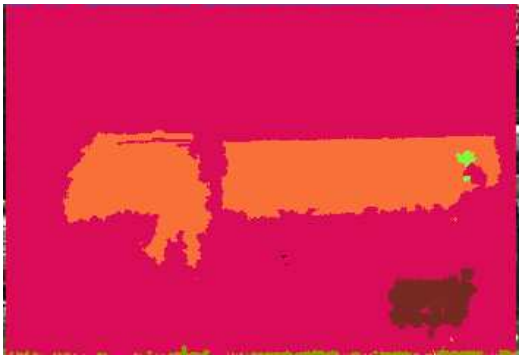
6.5 Stop Condition

The presented algorithm stops propagation when all image points are assigned to segments and there is no segment that could push back another segment. The algorithm cannot run infinitely because oscillations between segments are impossible. No segment can visit twice the same area. Namely, when a segment was pushed back by another segment, it cannot get back the lost pixels.

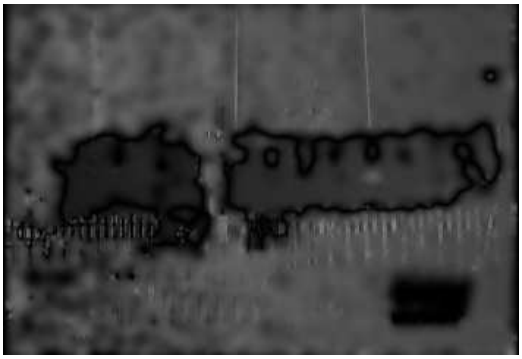
6.6 Experiments

The proposed method of segmentation using multi-label fast marching was evaluated experimentally. The algorithm was able to segment complex scenes with multiple overlapping objects and with objects partially visible in the scene. An example of such a scene is presented in Figure 6.6.

(a) Original image



(b) Segmented image



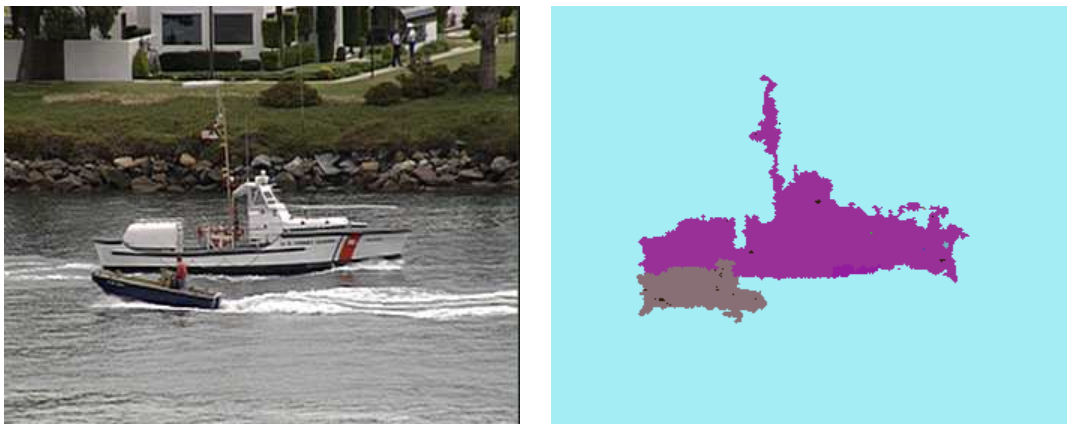
(c) Motion in the x direction

(d) Motion in the y direction

(e) Regularized motion in the x direction.

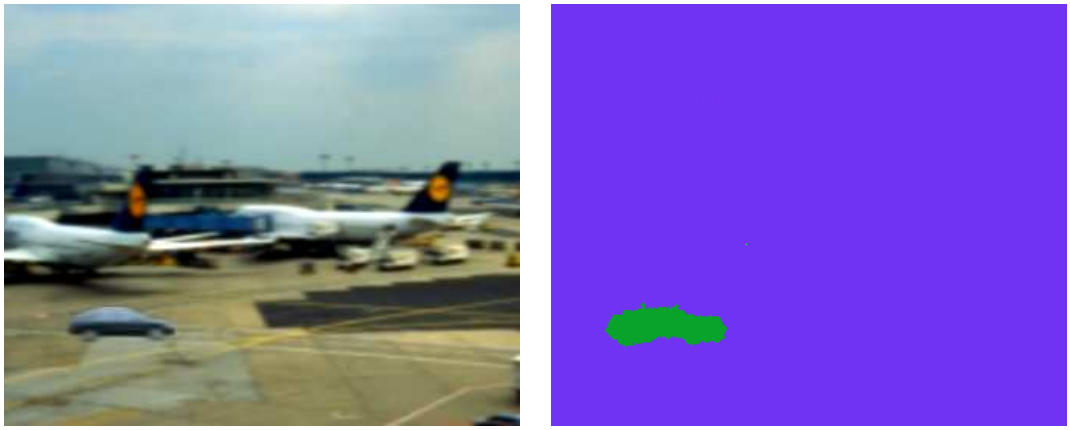
(f) Regularized motion in the y direction

Figure 6.6: Frame 112 from the 'Bus' sequence segmented using multi-label fast marching



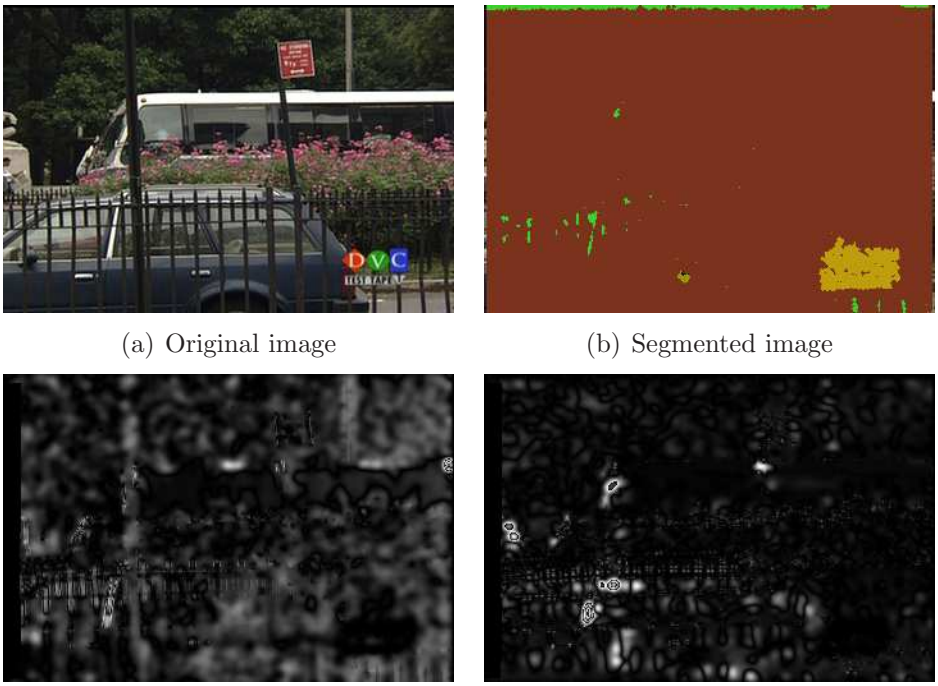
(a) Original image (b) Segmented image

Figure 6.7: Frame 78 from the 'Coast Guard' sequence segmented using multi-label fast marching



(a) Original image (b) Segmented image

Figure 6.8: Exemplary frame from the semi-artificial 'Car' sequence segmented using multi-label fast marching



(a) Original image (b) Segmented image

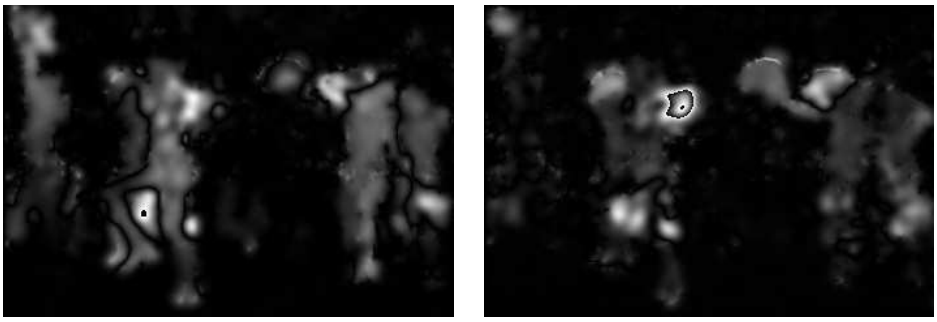
(c) Motion in the x direction (d) Motion in the y direction

Figure 6.9: Frame 8 from the 'Bus' sequence segmented using multi-label fast marching. Problems with the erroneous motion field

The algorithm requires a *partially reliable* motion field for correct performance. This means that the motion estimation algorithm must be able to produce at least some parts of the motion field, with motion vectors that point precisely onto the corresponding pixels from the previous frame (dfd for these points is zero). An example of such a motion field is shown in Figures 6.6(c) and 6.6(d). The algorithm fails when there is no reliable motion field. Figure 6.9 shows the segmentation of Frame 8 from the 'Bus' sequence. For this frame, motion was too fast for the currently implemented motion estimation algorithm. Motion vectors are mostly erroneous and the consequence is a wrongly segmented image. However, for the testing purposes, only simple classical motion estimation algorithms were implemented. The implementation of a fastest and more precise motion estimation method will improve the performance of the segmentation algorithm.

In this implementation only the sequences with translational and rigid motion can be segmented. However, this limitation is dependent on the motion regular-

(a) Original image (b) Segmented image



(c) Motion in the x direction (d) Motion in the y direction

Figure 6.10: Frame 166 from the 'Cheer' sequence segmented using multi-label fast marching. Problems with complex elastic motion

ization method, which is an integral part of the algorithm and can be replaced with another method without interference into the core algorithm. Figure 6.10 presents a sequence with complex non-rigid motion that causes problems for the algorithm.

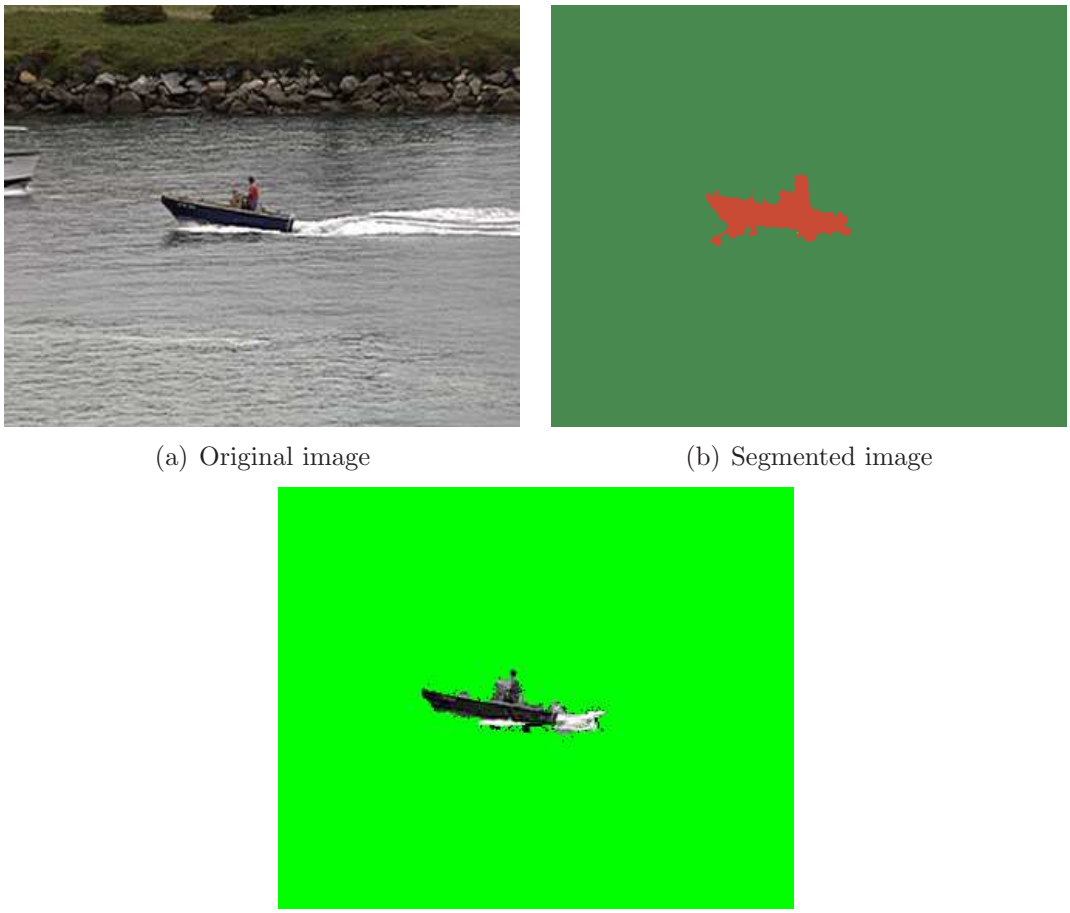
Computational efficiency of this method is similar to this presented in Chapter 5. Nevertheless, the total time of frame segmentation is now higher despite the much simpler propagation speed. The current speed definition allows calculating speed for the whole frame before propagation begins using fast convolution filters. During the propagation, speed is only read from the table. However, the total length of the propagated contour is much bigger than it was in the method from Chapter 5. As a consequence, segmentation with multi-label fast marching can last even three times longer. This is a proof of the fact that the biggest impact on the performance comes from the implementation of the sorting algorithm used by the Fast Marching Method.

Multi-label propagation was implemented using Sifakis's [124] approach, namely, by a single sorted list. This approach has some advantages when all the propagated contours must be synchronized in time. They share the sorted list, thus the fastest point of all from all the contours is always propagated. For Sifakis this was essential since two propagated contours were expected to meet on the object boundary. In his method, there is no possibility of correction when the boundary is missed.

In the method presented in this chapter, such time synchronization is not essential. It is important because the first stage of propagation is much faster than segment pushing but it is not critical. This opens the way for parallel implementation of propagating contours. Each contour can be propagated by a different thread until the contours are merged. During the merging, the thread the with the smaller segment is destroyed and its contour is took over by the thread with the bigger segment. The implementation of segment pushing can be virtually the same as in the version with the single list since threads do not process points already visited by the contour.

The performance analysis from Section 5.3 is also applicable in the case of multi-label fast marching. Despite longer execution times in comparison to the method from Chapter 5, for a single processing thread (about three times), the method presented in this chapter can be computed using a larger number of processors. Real time can be achieved using only four processors of the Pentium IV class. This is a very good property because complexity grows linearly with the frame size.

Figure 6.11 shows a comparison of a frame segmented with the method presented in this chapter and with the method presented by Patras in [105]. While segmentation quality is on a similar level, algorithm complexity is much lower in the method developed in this thesis. Patras uses for segmentation several steps, and each of them converges iteratively to the final solution. However, Patras's method deals better with large object displacements.



(a) Original image (b) Segmented image

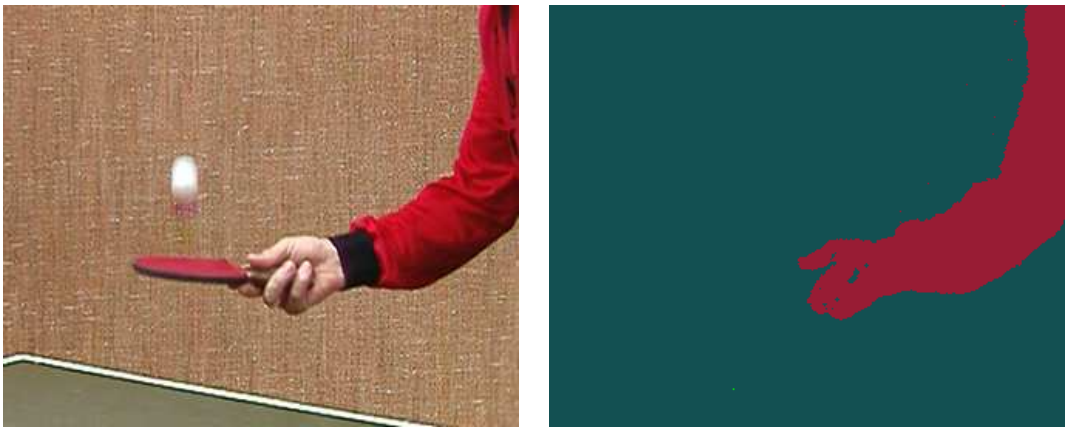
(c) Result obtained by Patras in [105]

Figure 6.11: Frame 2 from the 'Coast Guard' sequence segmented using multi-label fast marching compared to the result obtained by another author

Figures 6.12 to 6.15 show exemplary segmentations obtained by the method developed in this thesis. Objects with large displacements are usually undetectable, which is caused by limitation of the motion estimation method used for tests. In some cases the method tends to oversegment frame. Or live small false objects (Figure 6.15). A way of keeping the segments more consistent must be found in further development of the method.

(a) Original image (b) Segmented image

Figure 6.12: Frame 199 from the 'Coast Guard' sequence segmented using multi-label fast marching



(a) Original image (b) Segmented image

Figure 6.13: Frame 11 from the 'Table Tennis' sequence segmented using multi-label fast marching

(a) Original image (b) Segmented image

Figure 6.14: Frame 165 from the 'Mobile' sequence segmented using multi-label fast marching

(a) Original image (b) Segmented image

Figure 6.15: Frame 143 from the 'Bus' sequence segmented using multi-label fast marching

Chapter 7

Conclusions

7.1 The Two Original Methods

The algorithms developed in this dissertation allow segmentating colour video sequences in the presence of a moving background without the necessity for global motion compensation. Both algorithms are designed to segment individual frames without object tracking. The motivation of such an approach is the fact that there is a large number of object tracking algorithms [28,45,54,55,82] that require prior manual initialization of the object boundary. However, there exists a problem of automatic search of objects at the beginning of the sequence.

Two methods were proposed by the author. Both of them fulfil the assumptions given in Section 1.2, which proves the thesis given at the beginning of this dissertation.

The first algorithm separates foreground objects from the background. Combining the Fast Marching Method with static image segmentation allows developing a two-step method which is fast and accurate. The first step based on the Fast Marching Method is designed to provide fast and rough segmentation of the video frame while the second step refines the result using information that comes from static image segmentation. Correct performance of the method requires a reliable motion field on the object boundary. Motion estimation errors in highly textured areas may result in segmentation errors. The motion field in smooth image areas is ignored, thus segmentation errors in those areas have no

influence on the algorithm.

The second segmentation algorithm was designed to segment the video frame into multiple separated objects. The main assumptions are the same as in the first method, but they are additionally limited to rigid objects with translational motion. Here the main concern was algorithm speed and stability rather than segmentation quality. The current version of the algorithm cannot deal with complex motion and sometimes may produce oversegmented frames. Nevertheless, its generality permits further extensions and performance improvements.

7.2 Original Achievements

The Fast Marching Method developed by Sethian [119] has been used in some video segmentation application (see Section 4.4.2). However, it is not as popular as the other active contour method because of its ability to propagate contour only in one way, which causes many problems with segmentation quality. The new approach presented in this work lets exploit the speed of the Fast Marching Method and ensures high segmentation quality. The author's original contribution to the first method includes:

- Introduction of a second colour-based step that enhances the results of fast marching segmentation.

- Adaptation of the Fast Marching Method to frame segmentation that includes propagation speed design. Speed is designed in a way that permits segmentation even with an erroneous motion field.

- Reduction of segmentation errors by introducing a special difference operator that controls curve behaviour.

- Automatic initialization method that can deal with objects partially included in the frame.

The second algorithm developed in this work is based on the idea of multi-label fast marching that was introduced by Sifakis [124]. Nevertheless, the realization proposed in this thesis is totally different from the original. The original method is limited to a sequences with static background and is able to propagate only two segments with individually defined speed functions. The method presented in this work can deal with sequences with moving background and is able to propagate any number of segments. The author's original contribution to the second method includes:

- Development of the multi-label version of the fast marching algorithm that can perform with any number of contours propagating at the same time.
- Extension to multi-label propagation that removes the limitation of one-way contour propagation. This facilitates the adopting of the algorithm to a much wider range of problems than segmentation, which was reserved for slower, bi-directional active contour models.
- Significant simplification of the algorithm contrary to the method known in the literature [124, 125], which permits easy parallelization of calculations.

Additionally, a novel method of segmentation quality assessment was proposed in Section 2.4.3. This method is based on soft reference objects, which makes it more insensitive to false assessments due to an imprecise preparation of reference segmentations. Moreover, the method can be directly applied even to segmentation with fractional membership functions, which are a possible way of further development of image and video segmentation methods.

Some performance improvement of the motion estimation method was proposed in Section 2.3.3.

7.3 Direction of Future Research

Further research should include work on segmentation quality improvements as well as the reliability of the method in the presence of noise and motion estimation

errors. The integration of segmentation methods developed in this dissertation with object tracking techniques may lead to a significant improvement in performance.

The second of the proposed segmentation algorithms requires a more sophisticated method of motion regularization, which would make the segmentation of sequences with complex motion possible. Additionally, it will be a necessary verification if the proposed methods of merging and pushing objects are also applicable to objects with non-rigid motion.

Further research related to the second algorithm proposed may also include its adaptation to problems other than video segmentation, for example, medical image segmentation or the reconstruction of objects from unstructured data.

Further performance improvement can be achieved by the parallelization of the fast marching algorithm. An example was proposed by Dejnozkova and Dokladal in [26].

Bibliography

[1] N. Abdelmalek, "Noise filtering in digital images and approximation theory," *Pattern Recognition*, vol. 19, no. 5, pp. 417–424, 1986.

[2] D. Adalsteinsson and J. Sethian, "A fast level set method for propagating interfaces," *J. Comp. Physics*, vol. 118, no. 2, pp. 269–277, 1995.

[3] D. Adalsteinsson and J. Sethian, "A level set approach to a unified model for etching, deposition, and lithography I: algorithms and two-dimensional simulations," *Journal of Computational Physics*, no. 120, pp. 128–144, 1995.

[4] A. Albiol, L. Torres, and E. J. Delp, "An unsupervised color image segmentation algorithm for face detection applications," in *IEEE International Conference on Image Processing*, (Thessaloniki, Greece), pp. 681–684, 2001.

[5] Y. Altunbasak, P. E. Eren, and A. M. Tekalp, "Region-based parametric motion segmentation using color information," *Graphical Models and Image Processing*, vol. 60, pp. 3–23, January 1998.

[6] T. Aslam, "Level set methods for tracking discontinuities in reactive flow," in *SIAM 9th International Conference on Numerical Combustion*, (Sorrento, Italy), pp. 24–27, April 7-10 2002.

[7] A. Bab-Hadiashar, N. Gheissari, and D. Suter, "Robust model based motion segmentation," in *International Conference on Pattern Recognition*, (Quebec City, Canada), pp. II: 753–756, August 2002.

[8] J. L. Barron and H. Spies, "The fusion of image and range flow," *Lecture Notes in Computer Science*, vol. 2032, pp. 171–190, 2001.

[9] J. Bescós, A. Movilla, J. Menéndez, and G. Cisneros, "Real time temporal segmentation of mpeg video," in *International Conference on Image Processing*, vol. 2, (Rochester, New York, USA), pp. II–409 – II–411, September 2002.

[10] U. Bhosle, S. Chaudhuri, and S. D. Roy, "A fast method for image mosaicing using geometric hashing," *IETE Journal of Research*, vol. 48, pp. 317–324, May – August 2002.

[11] U. Bhosle, S. Chaudhuri, and S. D. Roy, "Background mosaicing for scenes with moving objects," in *National Conference on Communications (NCC)*, pp. 85–89, 2003.

[12] M. Bichsel, "Segmenting simply connected moving objects in a static scene," *Pattern Analysis and Machine Intelligence*, vol. 16, pp. 1138–1142, November 1997.

[13] M. Bober, M. Petrou, and J. Kittler, "Nonlinear motion estimation using the supercoupling approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 550–555, May 1998.

[14] G. D. Borshukov, G. Bozdagi, Y. Altunbasak, and A. M. Tekalp, "Motion segmentation by multistage affine classification," *IEEE Transactions on Image Processing*, vol. 6, pp. 1591–1594, November 1997.

[15] P. Brault and A. Djafari-Mohammad, "Bayesian segmentation and motion estimation in video sequences using a Markov-Potts model," in *WSEAS04 Int'l Conference on Applied Mathematics*, pp. 98–103, April 2004.

[16] J. F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[17] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[18] A. Cavallaro and T. Ebrahimi, "Accurate video object segmentation through change detection," in *IEEE International Conference on Multimedia and Expo*, (Lausanne, Switzerland), pp. 445–448, August 2002.

[19] Ç. Erdem and B. Sankur, "Performance evaluation metrics for object-based video segmentation," in *10th European Signal Processing Conference (EUSIPCO)*, (Tampere, Finland), pp. 917–920, September 4-8 2000.

[20] D. Chai and A. Bouzerdoum, "A Bayesian approach to skin color classification in ycbcr color space," in *IEEE Region Ten Conference (TENCON'2000)*, vol. II, (Kualu Lumpur, Malaysia), pp. 923–926, September 2000.

[21] R. Cipolla and A. Blake, "The dynamic analysis of apparent contours," in *IEEE International Conference on Computer Vision*, (Osaka, Japan), pp. 616–625, 1990.

[22] P. Correia and F. Pereira, "Standalone objective evaluation of segmentation quality," in *WIAMIS 2001 – Workshop on Image Analysis for Multimedia Interactive Services*, (Tampere, Finland), 16-17 May 2001.

[23] D. Cremers, "A multiphase level set framework for motion segmentation," in *Lecture Notes in Computer Science*, vol. 2695, pp. 599–614, Springer-Verlag, Heidelberg, 2003.

[24] K. Daniilidis and V. Kruger, "Optical flow computation in the log-polar-plane," in *Computer Analysis of Images and Patterns*, pp. 65–72, 1995.

[25] C. Debrunner and N. Ahuja, "Segmentation and factorization-based motion and structure estimation for long image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 206–211, February 1998.

[26] E. Dejnozkova and P. Dokladal, "A parallel algorithm for solving the eikonal equation," in *International Conference on Image Processing*, (Barcelona, Spain), September 14-17 2003.

[27] T. Deschamps and L. Cohen, "Fast extraction of minimal paths in 3D images and applications to virtual endoscopy," *Medical Image Analysis*, vol. 5, pp. 281–299, December 2001.

[28] M. Dubuission Jolly, S. Lakshmanan, and A. Jain, "Vehicle segmentation and classification using deformable templates," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 18, pp. 293–308, March 1996.

[29] E. Durucan and T. Ebrahimi, "Robust and illumination invariant change detection based on linear dependence for surveillance application," in *10th European Signal Processing Conference (EUSIPCO)*, (Tampere, Finland), pp. 1141–1144, September 2000.

[30] T. Ebrahimi and F. Pereira, *The MPEG-4 Book*. Prentice Hall, July 2002.

[31] Ç. Erdem, A. M.Tekalp, and B. Sankur, "Metrics for performance evaluation of video object segmentation and tracking without ground-truth," in *International Conference on Image Processing (ICIP)*, (Thessaloniki, Greece), pp. II:69–II:72, October 7-10 2001.

[32] M. R. Everingham, H. Muller, and B. T. Thomas, "Evaluating image segmentation algorithms using the Pareto front," in *7th European Conference on Computer Vision (ECCV2002), Part IV (LNCS 2353)* (A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, eds.), pp. 34–48, Springer, June 2002.

[33] R. Fablet, P. Bouthemy, and M. Gelgon, "Moving object detection in color image sequences using region-level graph labeling," in *International Conference on Image Processing*, (Kobe, Japan), pp. 939–943, 1999.

[34] M. D. Fairchild, *Color Appearance Models*. Addison-Weseley, 1997.

[35] R. Ford *et al.*, "Metrics for shot boundary detection in digital video sequences," *Multimedia Systems*, vol. 8, pp. 37–46, January 2000.

[36] A. R. François and G. G. Medioni, "Adaptive color background modeling for real-time segmentation of video streams," in *Proceedings of the International on Imaging Science, Systems, and Technology*, (Las Vegas, Nevada), pp. 227–232, 1999.

[37] A. Fusiello, M. Aprile, R. Marzotto, and V. Murino, "Mosaic of a video shot with multiple moving objects," in *IEEE International Conference on Image Processing*, (Barcelona, Spain), pp. II:307–II:310, September 14-17 2003.

[38] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills, "Recovering motion fields: An evaluation of eight optical flow algorithms," in *Ninth British Machine Vision Conference (BMVC '98)*, vol. 1, (Southampton, UK), pp. 195–204, September 14-17 1998.

[39] M. Gastaud, M. Barlaud, and G. Aubert, "Combining shape prior and statistical features for active contour segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 726–734, May 2004.

[40] M. Gastaud and M. Barlaud, "Video segmentation using active contours on a group of pictures," in *International Conference on Image Processing (ICIP)*, (Rochester, New York, USA), pp. II–81 – II–84, September 2002.

[41] D. Gatica-Perez, M. Sun, and C. Gu, "Semantic video object extraction based on backward tracking of multivalued watershed," in *IEEE International Conference on Image Processing*, pp. 145–149, IEEE, 1999.

[42] F. Gibou, R. Fedkiw, R. Caflish, and S. Osher, "A level set approach for the numerical simulation of dendritic growth," *Journal of Scientific Computations*, no. 19, pp. 183–199, 2003.

[43] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky, "Cortex segmentation: A fast variational geometric approach," *IEEE Trans. on Medical Imaging*, vol. 21(2), pp. 1544–51, December 2002.

[44] A. Goumeidane *et al.*, "New discrepancy measures for segmentation evaluation," in *International Conference on Image Processing*, (Barcelona, Spain), pp. II:411–II:414, September 14-17 2003.

[45] J. Guo, J. Kim, and C. Kuo, "An interactive object segmentation system for mpeg video," in *International Conference on Image Processing*, (Kobe, Japan), pp. 140–144, 1999.

[46] N. Habili and K. Ngan, "Automatic multi-cue vop extraction for mpeg-4," in *Picture Coding Symposium*, (Saint Malo, France), April 2003.

[47] T. Hanning, H. Farr, M. Kellner, and V. Lauren, "Segmentation of vector images by n-level-set-fitting," in *IEEE International Conference on Image Processing*, (Thessaloniki, Greece), pp. II: 793–796, October 2001.

[48] R. Haralick and L. Shapiro, "Glossary of computer vision terms," *Pattern Recognition*, vol. 24, no. 1, pp. 69–93, 1991.

[49] R. Hardie *et al.*, "High resolution image reconstruction from digital video with in-scene motion," in *IEEE International Conference on Image Processing*, (Santa Barbara, CA, USA), pp. I:153–I:156, October 1997.

[50] P. Harper and R. B. Reilly, "Color based video segmentation using level sets," in *IEEE International Conference on Image Processing*, vol. III, (Vancouver, Canada), pp. 480–483, IEEE, 2000.

[51] C. G. Harris and M. Stephens, "A combined corner and edge detector," in *Fourth Alvey Vision Conf.*, pp. 147–151, 1988.

[52] H. Hassan and A. Farag, "Mra data segmentation using level sets," in *IEEE International Conference on Image Processing*, (Barcelona, Spain), pp. II:173–II:176, September 14-17 2003.

[53] B. K. Horn and B. G. Rhunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[54] G. Iannizzotto and L. Vita, "Real-time object tracking with movels and affine transformations," in *IEEE International Conference on Image Processing*, vol. I, (Vancouver, Canada), pp. 316–322, 2000.

[55] M. Irani, B. Rousso, and S. Peleg, "Detecting and tracking multiple moving objects using temporal integration," in *European Conference on Computer Vision*, vol. 588 of *LNCS*, pp. 282–287, Springer-Verlag, 1992.

[56] ISO/IEC DIS 14496, *Generic coding of audio-visual objects.*

[57] ISO/IEC IS 15938-3 (2001), *Multimedia Content Description Interface – Part 3: Visual.* Tokyo, 1998.

[58] ISO/IEC JTC1/SC29/WG11 Doc. MPEG97/M2249, *MPEG-4 video verification model version 7.1.* Stockholm, 1997.

[59] E. Izquierdo, A. Pinheiro, and M. Ghanbari, "A robuts and efficient scale-space based metric for the evaluation of MPEG-4 VOPs," in *IEEE International Conference on Circuits and Systems*, (Geneva, Switzerland), May 2000.

[60] R. Jain, "Difference and accumulative difference pictures in dynamic scene analysis," *Image and Vision Computing*, vol. 2, pp. 98–108, February 1984.

[61] J. Jaksa, K. Ślot, and P. Szczypiński, "Face recognition using deformable models," in *Conference on Signals and Electronic Systems*, (Łódź, Poland), pp. 111–116, 2001.

[62] S. Ji and H. W. Park, "Region-based video segmentation using dct coefficients," in *IEEE International Conference on Image Processing*, pp. 150–154, IEEE, 1999.

[63] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *International Conference on Computer Vision*, (Vancouver, Canada), pp. II: 586–591, July 2001.

[64] S. K. Kang, A. Koschan, H. S. Zhang, J. K. Paik, B. R. Abidi, and M. A. Abidi, "Hierarchical approach to enhanced active shape model for color video tracking," in *IEEE International Conference on Image Processing*, (Rochester, New York), pp. I–888 – I–891, September 2002.

[65] M. Kass, A. Witkin, and D. Terauzopoulos, "Snakes: Active contour models," in *First International Conference on Computer Vision*, pp. 259–263, June 8–11 1987.

[66] M. Kass, A. Witkin, and D. Terauzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[67] A. Katsaggelos, L. Kondi, F. Meier, J. Ostermann, and G. Schuster, "Mpeg-4 and rate-distortion-based shape-coding techniques," *Proceedings of the IEEE*, vol. 86, pp. 1126–1154, June 1998.

[68] S. Khan and M. Shah, "Object based segmentation of video using color, motion and spatial information," in *Computer Vision and Pattern Recognition*, (Kauai, Hawaii), December 11–13 2001.

[69] I. Kompatsiaris and M. G. Strintzis, "Spatiotemporal segmentation and tracking of objects in image sequences," in *IEEE International Conference on Image Processing*, pp. 155–158, IEEE, 1999.

[70] J. Konrad and M. Ristivojević, "Joint space-time image sequence segmentation based on volume competition and level sets," in *International Conference on Image Processing (ICIP)*, (Rochester, New York, USA), pp. I–573 – I–576, September 2002.

[71] I. Koprinska and S. Carrato, "Temporal video segmentation, a survey," *Signal Processing and Image Communication*, vol. 16, pp. 477–450, January 2001.

[72] G. Kühne, J. Weickert, O. Schuster, and S. Richter, "A tensor-driven active contour model for moving object segmentation," in *IEEE International Conference on Image Processing*, vol. II, (Thessaloniki, Greece), pp. 73–76, 2001.

[73] S. Lee and C. Chung, "Hyper-rectangle based segmentation and clustering of large video data sets," *Information Sciences*, vol. 141, pp. 139–168, January 2002.

[74] S. Lim and A. El Gamal, "Optical flow estimation using high frame rate sequences," in *International Conference on Image Processing (ICIP)*, (Thessaloniki, Greece), pp. II:925–II:928, October 7–10 2001.

[75] N. Li, J. Bu, and C. Chen, "Real-time video object segmentation using HSV space," in *International Conference on Image Processing (ICIP)*, (Rochester, New York, USA), pp. II:85 – II:88, September 2002.

[76] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of DARPA Image Understanding*, pp. 121–130, 1981.

[77] H. Luo, A. Eleftheriadis, and J. Kouloheris, "Statistical model-based video segmentation and its application to very low bit-rate video coding," *Signal Processing: Image Communication*, pp. 333–352, April 2000.

[78] R. Malladi and J. A. Sethian, "A real-time algorithm for medical shape recovery," in *International Conference on Computer Vision*, (Bombay, India), pp. 304–310, January 1998.

[79] R. Malladi and J. Sethian, "Image processing: Flows under min/max curvature and mean curvature," *Graphical Models and Image Processing*, vol. 58, no. 2, pp. 127–141, 1996.

[80] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modelling with front propagation," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 17, pp. 158 – 175, February 1995.

[81] B. S. Manjunath, P. Salembier, and T. Sikora, eds., *Introduction to MPEG-7: Multimedia Content Description Interface*. John Wiley and Sons, 2002.

[82] A.-R. Mansouri and J. Konrad, "Motion segmentation with level sets," in *IEEE International Conference on Image Processing*, (Kobe, Japan), pp. 126–130, October 1999.

[83] A.-R. Mansouri, "Region tracking via level set pdes without motion compensation," *IEEE Trnasactions Pattern Analysis Machine Intelligence*, vol. 24, pp. 947–961, July 2002.

[84] K. McKoen *et al.*, "Evaluation of video segmentation methods for surveillance applications," in *10th European Signal Processing Conference (EUSIPCO)*, (Tampere, Finland), September 4-8 2000.

[85] R. Mech and M. Wollborn, "A noise robust method for segmentation of moving objects in video sequences," in *IEEE International conference on Acoustics, Speech and Signal Processing*, vol. 4, (Munich, Germany), pp. 2657–2660, April 1998.

[86] T. Meier and K. N. Ngan, "Video segmentation for content-based coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 1147–1167, Dec 1999.

[87] T. Meier and K. N. Ngan, "Automatic segmentation of moving objects for video object plane generation," *IEEE Trans. on Cirquits and Systems for Video Technology*, vol. 8, no. 5, pp. 525–538, 1998.

[88] A. Mittal and D. Huttenlocher, "Scene modeling for wide area surveillance and image synthesis," in *Computer Vision and Pattern Recognition (CVPR'00)*, vol. 2, (Hilton Head, South Carolina, USA), pp. 160–167, June 2000.

[89] L. Moisan, "Affine plane curve evolution: A fully consistent scheme," *IEEE Transactions on Image Processing*, vol. 7, pp. 411–420, March 1998.

[90] D. Muresan and T. Parks, "Adaptive principal components and image denoising," in *IEEE International Conference on Image Processing*, (Barcelona, Spain), pp. I:101–I:104, September 14-17 2003.

[91] D. Murray and B. Buxton, "Scene segmentation from visual motion using global optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 220–228, 1987.

[92] A. Neri, S. Colonnese, G. Russo, and P. Talone, "Automatic moving object and background separation," *Signal Processing*, vol. 66, pp. 219–232, April 1998.

[93] S. Nikiel and P. Steć, "Metody promocji dostępne w sieci WWW," in *Multimedialne i sieciowe systemy informacyjne*, pp. 305–311, Oficyna Wydawnicza Politechniki Poznańskiej, 1998.

[94] H. Nowak and K. Ślot, "Object classification with intermediate deformable models," in *European Conference on Circuit Theory and Design*, (Cracow, Poland), pp. I:240–I:243, September 2003.

[95] C. Odet, B. Belaroussi, and H. Cattin, "Scalable discrepancy measures for segmentation evaluation," in *IEEE International Conference on Image Processing*, (Rochester, New York), pp. I:785–I:788, September 2002.

[96] J.-R. Ohm, *Multimedia Communication Technology: Representation, Transmission, and Identification of Multimedia Signals*. Springer-Verlag, February 2004.

[97] S. Osher and J. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[98] I.-T. R. P.910, *Subjective video quality assessment methods for multimedia applications*. ITU-T, August 1996.

[99] D. L. Page *et al.*, "Simultaneous mesh simplification and noise smoothing of range images," in *IEEE International Conference on Image Processing*, vol. III, (Rochester, New York), pp. 821–824, September 2002.

[100] T. Papadimitriou, K. Diamantaras, M. Strintzis, and M. Roumeliotis, "Video scene segmentation using spatial contours and 3-D robust motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 485–497, April 2004.

[101] N. Paragios and R. Deriche, "A pde-based level set approach for detection and tracking of moving objects," in *IEEE International Conference in Computer Vision*, (Bombay, India), pp. 1139–1145, January 1998.

[102] N. Paragios and R. Deriche, "Geodesic active contours and level sets for the detection and tracking of moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 3, pp. 266–280, 2000.

[103] D. Park, "Adaptive bayesian decision-model for motion segmentation," *Pattern Recognition Letters*, vol. 15, pp. 1183–1189, December 1994.

[104] N. V. Patel and I. K. Sethi, "Video shot detection and characterization for video databases," *Pattern Recognition*, vol. 30, pp. 583–592, April 1997.

[105] I. Patras, E. Hendriks, and R. Lagendijk, "Video segmentation by map labeling of watershed segments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 326–332, March 2001.

[106] M. Pawlak and E. Rafajlowicz, "Non-linear local harmonic filters for edge-preserving image denoising," in *International Conference on Pattern Recognition*, (Quebec, Canada), pp. III: 895–898, August 2002.

[107] A. Pentland, "Automatic extraction of deformable part models," *International Journal of Computer Vision*, vol. 2, pp. 107–126, March 1990.

[108] M. Petrou, M. Bober, and J. Kittler, "Multiresolution motion segmentation," in *International Conference on Pattern Recognition*, vol. 1, (Jerusalem, Israel), pp. A:379–383, October 1994.

[109] S. L. Phung, A. Bouzerdoum, and D. Chai, "A novell skin color model in YCbCr color space and its application to human face detection," in *IEEE International Conference on Image Processing*, (Rochester, New York), pp. I–289 – I–292, September 2002.

[110] C. Poynton, *Digital Video and HDTV Algorithms and Interfaces*. San Francisco: Morgan Kaufmann Publishers, 1 ed., January 2003.

[111] E. Rouy and A. Tourin, "A viscosity solutions approach to shape-from-shading," *SIAM J. Num. Analysis*, vol. 29, no. 3, pp. 867–884, 1992.

[112] S. J. Sangwine and R. E. N. Horne, eds., *The Colour Image Processing Handbook*. Kluwer Academic Publishers, April 1998.

[113] J. Sethian and M. Popovici, "Three dimensional traveltimes computation using the Fast Marching Method," *Geophysics*, vol. 64, no. 2, pp. 516–523, 1999.

[114] J. Sethian, "Curvature and the evolution of fronts," *Communications of Mathematical Physics*, vol. 101, no. 4, pp. 487–499, 1985.

[115] J. Sethian, "Curvature and the evolution of fronts," *Comm. in Math. Phys.*, no. 101, pp. 487–499, 1985.

[116] J. Sethian, "Curvature flow and entropy conditions applied to grid generation," *Journal of Computational Physics*, vol. 115, pp. 440–454, 1994.

[117] J. Sethian, *Level Set Methods*. Cambridge University Press, 1996.

[118] J. Sethian, "Fast Marching Methods and Level Set Methods for propagating interfaces," in *29th Computational Fluid Dynamics*, vol. 1 of *VKI Lectures series*, von Karman Institute, 1998.

[119] J. Sethian, "Fast Marching Methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[120] J. A. Sethian and D. Adalsteinsson, "An overview of level set methods for etching, deposition, and lithography development," *IEEE Transactions on Semiconductor Devices*, vol. 10, no. 1, pp. 167–184, 1997.

[121] J. Sethian and D. Adalsteinsson, "An overview of level set methods for etching, deposition, and lithography development," *IEEE Transactions on Semiconductors Manufacturing*, vol. 10, no. 1, pp. 167–184, 1997.

[122] P. Sharma and R. Reilly, "Fast marching methods applied to face location in videophone applications using colour information," in *IEEE International Conference on Multimedia and Expo*, (Lausanne), August 2002.

[123] E. Sifakis, C. Garcia, and G. Tziritas, "Bayesian level sets for image segmentation," *Journal of Visual Communication and Image Representation*, vol. 13, pp. 44–64, March 2002.

[124] E. Sifakis and G. Tziritas, "Moving object localisation using a multi-label fast marching algorithm," *Signal Processing: Image Communication*, vol. 16, no. 10, pp. 963–976, 2001.

[125] E. Sifakis and G. Tziritas, "Video segmentation using fast marching and region growing algorithms," *EURASIP Journal on Applied Signal Processing*, pp. 379–388, April 2002.

[126] S. Sista and R. Kashyap, "Unsupervised video segmentation and object tracking," in *IEEE International Conference on Image Processing*, (Kobe, Japan), p. 26PP6, October 1999.

[127] P. Steć and M. Domański, "Video segmentation using fast marching methods," in *Int. Conf. Computer Vision and Graphics*, (Zakopane, Poland), pp. 710–715, 2002.

[128] P. Steć and M. Domański, "Efficient unassisted video segmentation using enhanced fast marching," in *IEEE International Conference on Image Processing – ICIP 2003*, (Barcelona, Spain), pp. 246–253, September 14-17 2003.

[129] P. Steć and M. Domański, "Foreground-background separation in video sequences using fast marching methods," in *European Conference on Circut Theory and Design – ECCTD '03*, (Cracow, Poland), pp. 89–92, 2003.

[130] P. Steć and M. Domański, "Two-step unassisted video segmentation using fast marching method," in *10th International Conference – CAIP 2003*, Lecture Notes in Computer Science, (Groningen, Holland), pp. 246–253, Springer-Verlag, 2003.

[131] P. Steć and M. Domański, "Fast two-step unassisted video segmentation technique evaluated by tolerant ground truth," in *5th International Workshop on Image Analysis for Multimedia Interactive Services*, (Lisboa, Portugal), April 21-23 2004.

[132] P. Steć, "Metody automatycznej segmentacji sekwencji wizyjnych," in *Poznańskie warsztaty telekomunikacyjne*, pp. 3.8–1 — 3.8–5, Instytut Elektroniki i Telekomunikacji Politechniki Poznańskiej, 2000.

[133] P. Steć, "Video segmentation," in *Symposium on Science Research Education*, (Zielona Góra), pp. 223–228, Technical University Press, 2000.

[134] E. Steinbach, P. Eisert, and B. Girod, "Model-based 3-d shape and motion estimation using sliding textures," in *Vision, Modeling, and Visualization*, (Stuttgart, Germany), pp. 375–382, November 2001.

[135] R. Szeliski, "Video mosaics for virtual environments," *IEEE Computer Graphics and Applications*, vol. 16, pp. 22–30, March 1996.

[136] A. Tavildar, H. Gupta, and S. Gupta, "Linear mmse filtering for restoration of images degraded by film grain noise," *Signal Processing*, vol. 6, pp. 225–234, 1984.

[137] S. Teboul, L. Blanc-Feraud, G. Aubert, and M. Barlaud, "Segmentation and edge-preserving restoration," in *IEEE International Conference on Image Processing*, (Santa Barbara, CA, USA), pp. II:470–II:473, October 1997.

[138] A. Tekalp, *Digital Video Processing*. Prentice Hall PTR, 1995.

[139] J. Tenenbaum, T. Garvey, S. Weyl, and H. Wolf, "An interactive facility for scene analysis research," Tech. Rep. 87, Stanford Research Institute, AI Centre, Stanford, 1974.

[140] D. Terzopulous, J. Platt, K. Fleischer, and A. H. Barr, "Elastically deformable models," in *SIGGRAPH '87*, pp. 205–14, ACM Press, 1987.

[141] D. Terzopulous, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3d shape and nonrigid motion," *Artificial Intelligence*, vol. 36, pp. 91–123, January 1988.

[142] L. Torres, D. García, and A. Mates, "On the use of layers for video coding and object manipulation," in *2nd Erlangen Symposium, Advances in Digital Image Communication*, (Erlangen, Germany), pp. 65–73, April 1997.

[143] R. Truyen, T. Deschamps, and L. D. Cohen, "Clinical evaluation of an automatic path tracker for virtual colonoscopy," *Lecture Notes in Computer Science*, vol. 2208, pp. 169–178, 2001.

[144] Y. Tsai, "Rapid and accurate computation of the distance function using grids," Tech. Rep. 17, Department of Mathematics, University of California, Los Angeles, 2000.

[145] N. Vasconcelos and A. Lippman, "Empirical Bayesian motion segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 217–221, February 2001.

[146] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing*, vol. 3, pp. 625–638, September 1994.

[147] C.-F. Westin, L. M. Lorigo, O. Faugeras, W. E. L. Grimson, S. Dawson, A. Norbash, and R. Kikinis, "Segmentation by adaptive geodesic active contours," in *Third International Conference On Medical Robotics, Imaging and Computer Assisted Surgery*, (Pittsburgh, Pennsylvania, USA), pp. 266–275, October 2000.

[148] Z. Wu, J. Bu, and C. Chen, "Detection and location of people in video streams by fusion of color, edge and motion information," in *IEEE International Conference on Image Processing*, (Rochester, New York), pp. III–449 – III–452, September 2002.

[149] J. Yan and T. Zhuang, "Applying improved fast marching method to endocardial boundary detection in echocardiographic images," *Pattern Recognition Letters*, vol. 24, pp. 2777–2784, November 2003.

[150] Y. Zhang, "A survey on evaluation methods for image segmentation," *Pattern Recognition*, vol. 29, pp. 1335–1346, August 1996.

[151] H.-K. Zhao, S. Osher, and R. Fedkiw, "Fast surface reconstruction using the level set method," in *1st IEEE Workshop on Variational and Level Set Methods*, (Vancouver, Canada), pp. 194–202, 2001.

[152] H.-K. Zhao, S. Osher, B. Merriman, and M. Kang, "Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method," *Computer Vision and Image Understanding*, no. 80, pp. 295–314, 2000.

[153] L. Zong and N. Bourbakis, "Digital video and digital TV: A comparison and the future directions," *Real-Time Imaging*, vol. 7, pp. 545–556, December 2001.