Region-of-Interest based Adaptation of Video to Mobile Devices

Tamir Nuriel

Region-of-Interest based Adaptation of Video to Mobile Devices

Research Thesis

Submitted in partial fulfillment of the requirements

for the degree of Master of Science in Electrical Engineering

Tamir Nuriel

Submitted to the Senate of

the Technion — Israel Institute of Technology

Tishrei 5770 Haifa October 2009

The research thesis was done under the supervision of Prof. David Malah in the Electrical Engineering Department.

Acknowledgements

I would like to express my deep gratitude to Prof. David Malah, for his devoted supervision and professional guidance throughout this research work.
I also wish to thank the staff of the Signal and Image Processing Lab (SIPL) for their help and technical support.
Finally, I owe a great debt to my dear wife, Tal, for her patience, understanding, and moral support. This work is dedicated to her.

The generous financial support of the Technion and Negev consortium is gratefully acknowledged.

Contents

List of Figures

Abstract

Mobile TV is a growing and promising market. The goal of this work is to develop methods for the adjustment of video in standard definition resolution to the smaller resolution used in mobile devices. The naive solution is to scale each frame to the desired size, thus reducing the size of all the objects. But, objects must be at a sufficient size to be recognized easily. Therefore, it was suggested in previous works to display only a Region Of Interest (ROI), which is a region including only the most important content.

In this work we focus on news broadcasting and interview scenes. The ROI is defined by a rectangle around the speaker's face. At the beginning of every scene, the ROI should be detected and tracked smoothly up to the end of the scene. Only the ROI is adjusted to the desired resolution. An editor can mark several ROIs for tracking. For example, marking two speakers in an interview scene and transmit only the active speaker in each frame.

We developed several tools for the editor. We present a novel algorithm for tracking the ROI. The algorithm estimates the global motion caused by the camera and the local motion caused by the speaker's movements. We use a pan, tilt and zoom camera model. We initially present an algorithm for estimating these parameters in the frequency domain. We show that only slices of the 2D Fourier transform are needed. By using the slice-projection theorem, we can compute these slices efficiently by computing the 1D Fourier transform of the horizontal and vertical projections. We reduce complexity by applying the 1D Mellin-transform on the slices in the frequency domain. This transform converts the scale parameter into a multiplicative parameter. We further reduce complexity by applying the 1D Mellin-transform on the projections in the spatial domain. We compute and compare the computational complexity

of the different methods. We also developed a statistical model for analyzing the performance of the motion parameters estimation.

The result of the tracking algorithm is jittery due to estimation errors caused by noise and interference. We reduce the jitter by using a Kalman filter. For more complicated cases (e.g., occlusion), we use a particle filter, obtaining also a more robust tracking.

In addition, we present a tool for the editor for detecting the ROI. The editor is required to click on the speaker's face image and the tool finds the ROI containing it, using a skin-color detector combined with a region growing algorithm.

Another tool developed is for detecting the ROI containing the current active speaker. We tested methods solely based on visual information, as motion vectors and pixels' intensities. Another method examined uses also the soundtrack. We track visual points on the speakers' faces and find the time instants of significant changes in their motion, and compare them to audio onset times. The correlation between the time instants of the events in video and audio is used to determine the current active speaker.

All the tools were tested by simulations as well as by real scenes recorded from TV.

Abbreviations and Notations

α	—	Scale factor.
$f(x, y)$	—	Frame intensity at pixel (x, y).
$F(f_x, f_y)$	—	Fourier transform of a frame at frequencies (f_x, f_y).
d_x	—	Shift in x-axis.
d_y	—	Shift in y-axis.
S	—	slices through the origin of the 2D Fourier transform.
$P(x)$	—	Projection on the x-axis.
M_f	—	Analytical Fourier-Mellin transform of frame f.
M_f^d	—	Discrete form of the analytical Fourier-Mellin transform of frame f.
$M_P(s)$	—	Mellin transform with parameter s on the projection P.
s	—	Parameter for the Mellin transform.
σ	—	Parameter for the analytical Fourier-Mellin transform.
W	—	Width of the frame.
H	—	Height of the frame.
μ_x	—	Mean of random variable x.
σ_x	—	Standard deviation of random variable x.
k	—	Frame number.
$(x(k), y(k))$	—	Location of the center of the ROI in frame k.
$w(k)$	—	Width of the ROI in frame k.
$h(k)$	—	Height of the ROI in frame k.
$v_x(k)$	—	Velocity of x component of the center of the ROI in frame k.
$v_y(k)$	—	Velocity of y component of the center of the ROI in frame k.
$v_w(k)$	—	Change of width of the ROI in frame k.
$v_h(k)$	—	Change of height of the ROI in frame k.
$S(k)$	—	State at frame k.

$z(k)$	—	Observations in frame k.
A	—	State transition model matrix.
H	—	Observation model matrix.
Γ	—	Control-input model matrix.
Q	—	Covariance matrix.
T	—	Time between consecutive frames.
$P(k\|i)$	—	Error covariance matrix in frame k given observations up to frame i.
$K(k)$	—	Kalman filter gain at frame k.
$L(k,i)$	—	Measure of the estimated state quality for a particle i in frame k.
$P_{UL}(k,i)$	—	Upper-Left point of particle i in frame k.
$P_{LR}(k,i)$	—	Lower-right point of particle i in frame k.
$w(k,i)$	—	Weight of particle i in frame k.
N	—	Number of particles.
N_{eff}	—	Effective number of particles.
$\vec{v}_{n,m}$	—	Motion vector of block n,m.
$I(x,y,t)$	—	Frame intensity at pixel (x,y) at time t.
$\mathcal{W}$	—	Window around visual point to track.
g	—	Gradient vector at the visual point to track.

Chapter 1

Introduction

In recent years Mobile TV is a growing and promising market. Most of the video content today have SD (standard definition) resolution. The SD resolution is different for various transmission methods, e.g. NTSC or PAL, and can be up to 720x576 pixels. On the other hand, the resolution of mobile devides displays, like cellphones or palm PCs, is usually smaller. Ordinary display resolutions are CIF (Common Intermediate Format), which is 352x288 pixels, or QCIF (Quarter CIF - one fourth of the CIF area as quarter implies that the height and width of the frame are halved). Therefore, production of a content suitable for watching video on small displays becomes an important task [3]. A simple and naive adjustment method is to rescale each frame of the original video in SD resolution to the desired output resolution. This method is simple and can be done automatically by a computer without the need of a human editor. However, this method has a siginificant disadvantage. It reduces the sizes of all the objects in the frame. Some objects may be too small for viewing propely. For example, a speaker's face may be too small for recognition.

An alternative adjustment method was suggested in [3]. A region in each frame is declared as a Region of Interest (ROI), and only this region will be displayed on the mobile devices. The process of marking the ROI in each scene is done by an editor in the editing phase, before saving the video for transmission. The editor marks manually the location of the ROI. This is added as a metadata to the compressed video stream. Just before the transmission of the edited video to the mobile device, the location of the ROI information is used,

and only the ROI is extracted and rescaled. The main disadvantage of this production process is that a human editor has to edit every scene manually and therefore it is time consuming. The process can be improved by using tracking algorithms. The ROI is marked at the start of every scene manually by the editor and a region tracking algorithm can be used to track the ROI to the end of the scene. However, most of the region tracking algorithms, which will be surveyed in chapter 2, have a very high computational complexity. Therefore, it is hard to perform the tracking in real-time on an ordinary editor's PC. For example, one of the methods for tracking a region is based on a color histogram. The algorithm searches in the current frame a region with a color histogram similar to the histogram of the ROI from the previous frame. Calculating color histogram for an entire region requires many operations. When we look for a region with similar color histogram in the current frame, we have to calculate the color histograms for each possible region in the vicinity of the region in the previous frame. This makes the algorithm not feasible for the typical computational power available.

In this work we develop several tools that make the production process semi-automatic. We will focus on news and interview scenes and thus we assume that the ROI is always a rectangle containing the speaker's face. First, we present a novel and efficient tracking algorithm. We assume that an initial rectangular ROI is given (manually or in an automatic manner) at the beginning of every scene. We track this ROI in the next frames, limiting ourselves to a rectangular ROI. The algorithm estimates the global motion caused by the camera movements and the local motion caused by the speaker's movements. We use a three parameters model to describe the camera movement. The parameters represent the zoom factor, pan (left or right shift) and tilt (up and down shift). An algorithm for estimating these parameters by applying a two-dimensional Fourier transform to the whole frame is presented. This algorithm has a high computational complexity becuase of the 2-D Fourier transform. But, we show that, actually, only two slices of this transform are needed. Using the well-known slice-projection theorem, we exploit the relation between slices of the 2-D Fourier transform and the 1-D Fourier transform of the horizontal and vertical projections. Hence, our motion estimation algo-

rithm is based solely on the horizontal and vertical projections. Thus, instead of performing a 2-D transform of each frame, we perform only 1-D transforms of the projections and thus reduces complexity.

Since we focus on news or interview scenes we can assume that the camera movement is relatively slow. Hence, it is realistic to assume that changes in the motion parameters of consecutives frames are small. We use as an initial estimation for the current frame the estimated parameters from the previous frame. Given an initial estimation of the motion parameters, we show that we can further reduce complexity by using the Mellin-transform in the spatial domain to estimate the current motion parameters. This transform converts the scale parameter into a multiplicative parameter, which can be estimated at a reduced complexity. We compare the computational complexity of the different methods. We also developed a statistical model, which is used for analyzing the performance of the motion parameters estimation.

The result of the tracking algorithm can be jittery due to small estimation errors caused by random noise and interferences. We suggest using a linear dynamic model for the region movement. For example, the region might have a constant velocity. We estimate its state from all the previous estimated motion parameters. By updating the region location according to the estimated state, we force its motion to be much less jittery and more robust. We can estimate the state using a Kalman filter, assuming the posterior probability density function of the state is Gaussian. If the true density is non-Gaussian (e.g., if it is bimodal), we suggest using a particle filter. The main idea is to represent the required posterior density function by a set of random samples. As the number of samples increase the particle filter approaches the optimal Bayesian estimate.

The ROI can have any size. Rescaling the ROI might cause distortion because the height/width ratio of the ROI may be different than the height/width ratio of the output display. Therefore, we will define another region as the Region Of Display (ROD). This region always contains the ROI and has the same propotions as the output display. That way, the distortion will be avoided. A more restrictive demand from our system is to extract one of two size options. The first option is to extract the entire frame and then rescale it to the desired

output size. The second option is to extract only a window which has the same size as the desired output size. We extract the pre-defined small region size (e.g., CIF) only when the ROI size is smaller than the display size. Otherwise, the whole frame will be rescaled and sent to the display.

There are many types of possible contents (e.g., news, sports, nature scenes), and thus it is hard to automatically detect the ROI. In this work we assume that the ROI always contains the speaker's face. We present a tool for the editor for detecting the bounding box of a face from only a single point marked by the editor on the speaker's face. This tool is used to help the editor mark the ROI at the start of every scene. We use a skin-color detector [27] for detecting the pixels of the face. A skin-color detector may detect also pixels outside the face as skin. Therefore, we further improve the face detection by combining the skin-color detector with the region growing algorithm, which segments the image by finding pixels with similar intensities.

We allow the editor to mark several ROIs at the beginning of every scene, and track all the ROIs to the end of the scene. But, only a single ROI is to be transmitted to the mobile device in any frame. The ROI to be sent is the ROI containing the active speaker. A tool was developed for detecting that ROI. We checked methods for active speaker detection that are solely based on visual information. Motion vectors are used to detect activity. We also used pixels' intensities to detect changes in the mouth area [8]. These methods were not robust enough. Another method that we examined uses also the soundtrack. We track visual points on the speakers' faces and find the time instants of significant changes in the motion characteristics e.g., acceleration. We then compare these time instants to audio onset times. The correlation between the time instants of the events in video and audio [12] is used to detect the current active speaker in each frame.

This work is organized as follows: Chapter 2 overviews previous work. Chapter 3 describes our proposed algorithms for tracking regions in video. Chapter 4 presents the computational complexity and a performance analysis of the tracking algorithms. Chapter 5 presents the usage of Kalman and particle filters for improved ROI tracking. In chapter 6, we describe methods for displaying the ROI without distortions and without jitter. Chapter 7 presents

methods for detecting an active speaker in an interview scene. In chapter 8 we present an algorithm for the detection of the ROI from a single click by the editor. Chapter 9 describes experimental results obtained in simulations as well as of real scenes recorded from TV. In chapter 10 the work is concluded and directions for future work are given.

Chapter 2

Previous Work

The first stage in our application is the detection of the region of interest. We focus in this work on a talking head or interview scenarios. Therefore, the ROI is defined always by the location of the speaker's head. Many algorithms exist for detecting faces. It was verified, using training data, that skin-colors are clustered in color space [10]. It is common to think that different people have skin-colors which differ significantly from each other due to the existence of different races. However, what really occurs is a larger difference in brightness / intensity and not in color.

Another method for detecting faces was suggested by Viola and Jones in [18]. In their work, a fast method for detecting faces based on simple features was proposed. An example for a feature called the two-rectangle feature is the difference between the sum of the values of two adjacent rectangular windows. A three-rectangle feature considers three adjacent rectangles and computes the difference between sum of the pixels in the extreme rectangles and the sum of the pixels in the middle rectangle. An example for such features is shown in Fig. 2.1. They introduced a new image representation called the "Integral Image" which allows the features used by their detector to be computed very quickly. Then, they used Adaboost [20] for learning. Adaboost is an algorithm that selects a small number of critical visual features from a larger set and yields extremely efficient classifiers.

A different method for detecting ROI is based on detecting salient objects in the frame, not necessarily faces. A visual attention system, inspired by

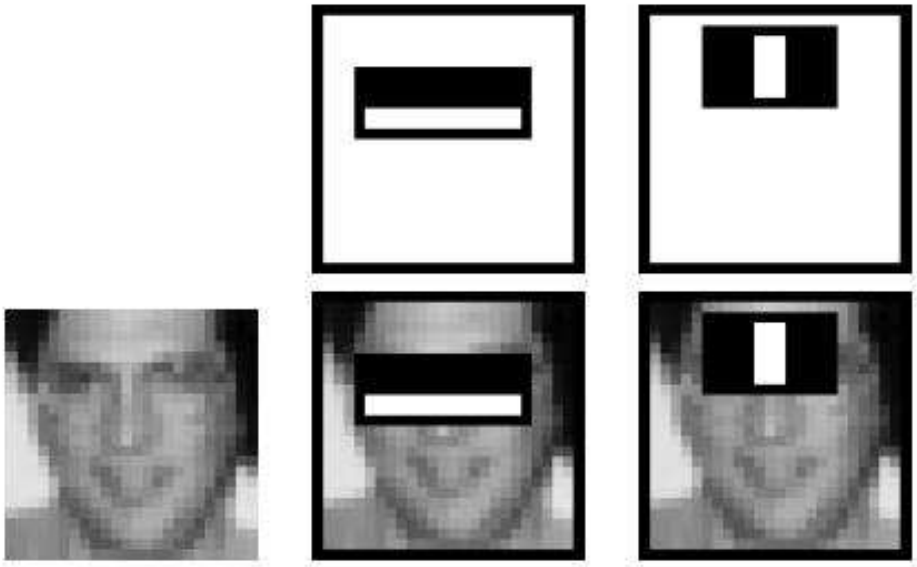


Figure 2.1: Features for the Viola-Jones face detector. Figure taken from [19]

the behavior and the neuronal architecture of the early primate visual system, was presented in [21]. Multi-scale image features are combined into a single topographical saliency map as shown in Fig. 2.2. The features are based on color, intensity and orientation. Several algorithms have been proposed for detecting a region of saliency using a modification of the saliency map. Fig. 2.3 demonstrates results for extracting a salient region suggested in [22].

For some applications, the ROI is marked at the start of every scene by an operator. The operator has to mark, for exmaple, a bounding region around the speaker's face. Marking a region requires indicating at least two points on the frame. A fast object selection that can be used as an auxiliary tool for the operator was proposed in [23]. They proposed an efficient semi-automatic algorithm for marking the bounding region. The operator needs to indicate only one point inside the object. Their work is based on the visual attention model presented by [21].

After detecting the ROI at the start of a scene, its location has to be tracked up to the end of the scene. One of the methods for tracking a region is based on a color histogram (e.g. [24]). The algorithm searches in the current frame for a region with a color histogram similar to the histogram of the ROI from

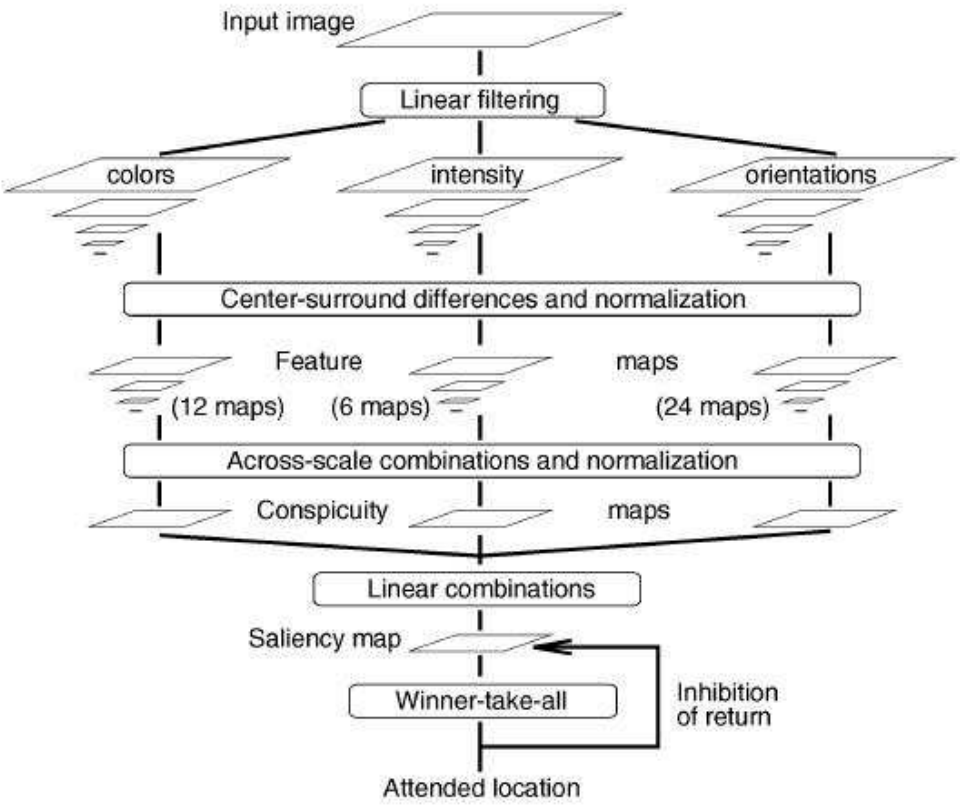


Figure 2.2: Visual attention model. Taken from [21]

the previous frame. There are several functions that can be used to measure the distance between two histograms. One of the most known functions is the Bhattacharya similarity, which measures not the distance, but the similarity between two histograms.

$$\sum_x \sqrt{f_1(x) f_2(x)}, \tag{2.1}$$

where $f_1(x), f_2(x)$ are histograms of the ROIs in two adjacent frames.

The search for the ROI in the current frame is in the area of the ROI in the previous frame. The ROI is used for display so we can assume it is always a rectangular. The ROI can be defined by its center, width and height. There are four parameters that can be changed: two for the center coordinates, one for the width and one for the height. The search area can be defined by small pertubations of these parameters. The search is usually performed by the

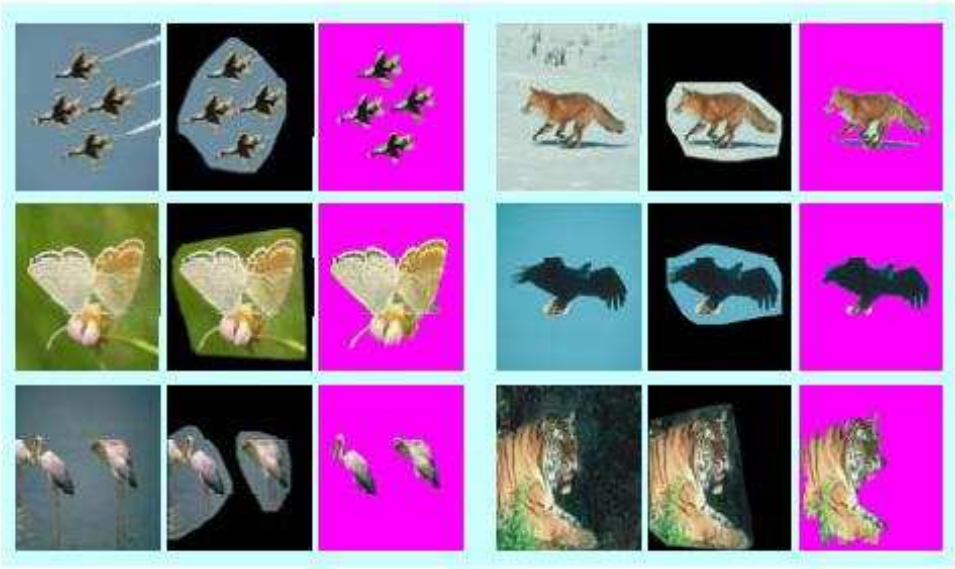


Figure 2.3: Results of extracting salient regions. Taken from [22]

N-step [1] or diamond search [2] methods, in the four-parameter space. Note that when calculating the histograms for all the regions, one can just update the previous results for faster computation.

In [25] the concept of spatiogram is introduced. The second-order spatiograms contain spatial means and covariances for each histogram bin. This spatial information still allows quite general transformations, as in a histogram, but captures a richer description of the region to increase robustness in tracking.

Another method of tracking is by using visual features. Finding features in the ROI in the previous frame and then detecting similar features in the current frame, can be used to track the ROI movement. In [26] it is proposed to use color cue and local scale-invariant feature transform (SIFT) features with particle filtering [30]. The particle weight is calculated firstly by color similarity measurement and then updated according to the distribution of SIFT matches.

A different approach for region tracking is by using a motion model. One method for motion estimatation is by using motion vectors. The computation of motion vectors has a high computational complexity. Therefore, these methods are suitable when the motion vectors have to be computed for other

reasons such as compression [38]. A combination of tracking by visual features and motion model was proposed in [37]. They suggested a fitting method of the motion vectors in a target region to the perspective motion model. The proposed method considers the motion vectors only of the feature points in the target region.

The tracking result might be too jittery for viewing. Noise can cause wrong estimated values of the ROI location. In order to solve the jitter problem use of a dynamic model is suggested in [9]. The dynamic model means that the region movement is controlled by certain parameters. For example, the region might have a constant velocity with a small random acceleration. The motion of the ROI can be much less jittery if we update it by estimating the state of the dynamic model.

In the Bayesian approach to dynamic state estimation, one tries to estimate the posterior probability density function of the state, based on all the set of received measurements. The Kalman filter, presented in [29], is one of the well-known methods for estimating the state of a model. Assuming a linear dynamic model and that both the state noise and the measurement noise processes are Gaussian, the Kalman filter is the optimal (in terms of MMSE) state estimator [31]. A Kalman filter is suggested in [9] for smoothing the motion of a tracked region.

A single Gaussian can not describe well the true posterior probability density function if it is non-Gaussian. Therefore, the Kalman filter will not work well in such a case. Particle filtering was introduced in [30] and tutorials can be found in [31, 32]. The main idea of particle filtering is to represent the required posterior density function by a set of random samples (particles) with associated weights and to compute estimates based on these samples and weights. The particle filter approaches the optimal Bayesian estimate when the number of particles increases. A particle filter is expected to yield an improvement in the estimation process when the density function is non-Gaussian.

Kalman filtering or particle filter reduces most of the noise in the tracking estimate. That may be sufficient for some applications. But, for broadcasting applications it is still too jittery. In [9] it was suggested to use a virtual camera. When a speaker is moving only within a small region, an experienced

camera operator usually does not move the camera at all. When the speaker is moving, the camera operator should move the camera smoothly in order to track him while keeping the video steady. The virtual camera detects if the speaker is in the same place and not moving significantly and extract the same region location for viewing. The virtual camera follows the speaker when he is moving.

The main disadvantage of all the methods described on this section is the high computational complexity required. In our work we develop efficient algorithms for detecting and tracking the ROI.

Chapter 3

Region Tracking by Horizontal and Vertical Projections

In this chapter we develop a novel algorithm for region tracking in video. At the beginning of a new scene the ROI location in the frame is marked by an operator. Our goal is to track the ROI location up to the end of the scene. Our algorithm is based on the horizontal and vertical projections of each frame. As will be shown, using these projections we can estimate the camera movements and the changes in the location of the region on the display. Camera movements change the location of the ROI. For example, consider the ROI location is known in a given frame and the camera moves to the right. The ROI location in the next frame should be moved also to the right to track the camera movement. Another example - consider the ROI location is known in a given frame and the camera zooms in. In the next frame the objects are bigger due to the zoom, and therefore the ROI size should increase by the same scale factor.

To track the ROI we need first to estimate the camera parameters by estimating global motion parameters between each two consecutive frames. After updating the ROI location, on the basis of the estimated global motion parameters, we need to find the local motion of the object of interest (e.g., a face) inside the ROI. For example, if the object moves to the right, we move

the location of the ROI also to right, so that object will remain in the center of the ROI.

Our work focuses on news and interview scenes, where the speakers are in front of the camera. In such scenes, the camera usually only perform zoom, pan and tilt operations. Since the speakers are far enough from the camera in a studio, we assume zooming is the result of a change in the camera's focal length and not by an approaching object to the camera. Hence, assume a three-parameter model for zoom, pan and tilt suffices to describe the global motion of the scene. The relation between two adjacent frames according to the assumed model is:

$$f_2(x, y) = f_1(\alpha x + d_x, \alpha y + d_y) \tag{3.1}$$

Where, f_1 denotes the previous frame and f_2 denotes the current frame. The global motion parameters are α - the zoom (scale) factor, d_x - the horizontal shift, and d_y - the vertical shift. Note that when $\alpha > 1$ the camera zooms out and when $\alpha < 1$ the camera zooms in.

We have three parameters to estimate. If there is no scale change (i.e., $\alpha = 1$), we can simply estimate the shifts, by using 2D correlation in the spatial domain. However, to estimate α, we need to find a transformation which is invariant to all the parameters, except the scale factor.

Transforming equation (3.1) into the frequency domain:

$$F_2\left(f_x, f_y\right) = \frac{1}{\alpha^2} e^{j2\pi(f_x d_x + f_y d_y)} F_1\left(\frac{f_x}{\alpha}, \frac{f_y}{\alpha}\right) \tag{3.2}$$

The shift parameters appear only in the phase of the Fourier transform. The relation between the magnitudes of the Fourier transforms of the current and next frames depends only on the scale parameter.

We do not need to calculate the entire two-dimensional Fourier transform. Instead, we can use the *projection-slice theorem* [5]. The theorem states that the Fourier transform of the projection of a two-dimensional function onto a line is equal to a slice through the origin of the two-dimensional Fourier transform of that function which is parallel to the projection line.

We demonstrate this theorem by an example: Consider the projection $P(x)$

on the x-axis, defined as:

$$P(x) = \int_{-\infty}^{\infty} f(x,y)\, dy \tag{3.3}$$

The 2D Fourier transform of $f(x,y)$ is given by:

$$F(f_x, f_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, e^{-2j\pi(xf_x+yf_y)} dxdy \tag{3.4}$$

The theorem states that the slice of the 2D Fourier transform, $S(f_x)$, and the projection $P(x)$ are a Fourier pair.

$$S(f_x) = F(f_x, 0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, e^{-2j\pi x f_x} dxdy$$
$$= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(x,y)\, dy\right] e^{-2j\pi x f_x} dx = \int_{-\infty}^{\infty} P(x)\, e^{-2j\pi x f_x} dx \tag{3.5}$$

Fig. 3.1 shows a graphical illustration of the projection-slice theorem in two dimensions.

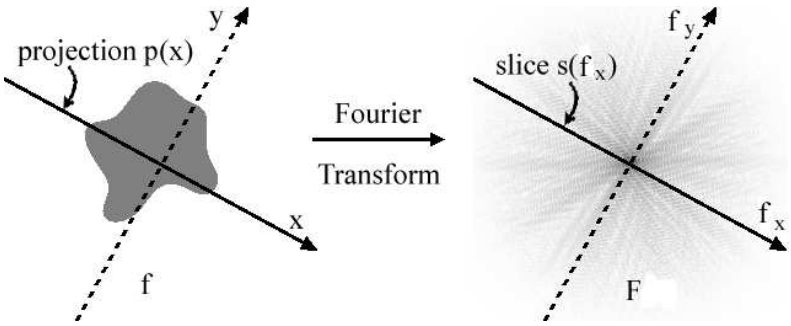


Figure 3.1: A graphical illustration of the projection slice theorem in two dimensions.

Using equation 3.2 we can estimate the scale α by taking the absolute value of a slice of the transform corresponding to the line $f_y = 0$ (or $f_x = 0$):

$$|S_2(f_x)| = |F_2(f_x, 0)| = \frac{1}{\alpha^2}\left|F_1\left(\frac{f_x}{\alpha}, 0\right)\right| = \frac{1}{\alpha^2}\left|S_1\left(\frac{f_x}{\alpha}\right)\right| \tag{3.6}$$

Where $S_1(f_x), S_2(f_x)$ are slices through the origin of the 2D Fourier transform of the previous frame and the current frame.

We use the slice-projection theorem as follows: Calculating the projections on the x-axis for the current and next frame and then taking the magnitude of the Fourier transform yields both sides of equation (3.6). In the next subsections we describe how to estimate the scale from these functions.

Fig. 3.2 shows an example of a speaking man moving to the left. We can see the corresponding change in the projections in Fig. 3.3. Horizontal projection is the projection on the x-axis and vertical projection is the projection on the y-axis.

Figure 3.2: Frames 1 and 100 from BBC movie.

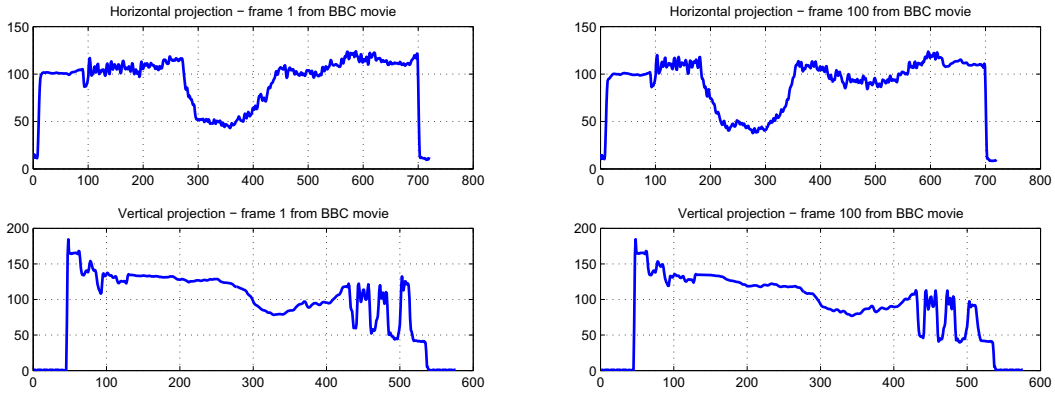


Figure 3.3: Horizontal and vertical projections for frames 1 and 100 from BBC movie.

In another example, the camera is zooming out while moving to the left.

The zoom action causes the pattern to shrink and move to the right. The distortion is non-linear. Motion due to zoom is proportional to the distance from the center of the image. The frames and the projections are shown in figures 3.4 and 3.5, respectively.

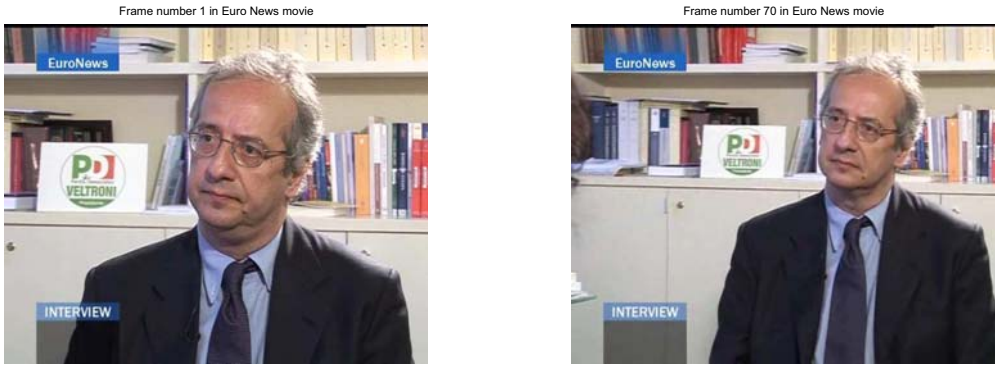


Figure 3.4: Frames 1 and 70 from Euro News movie.

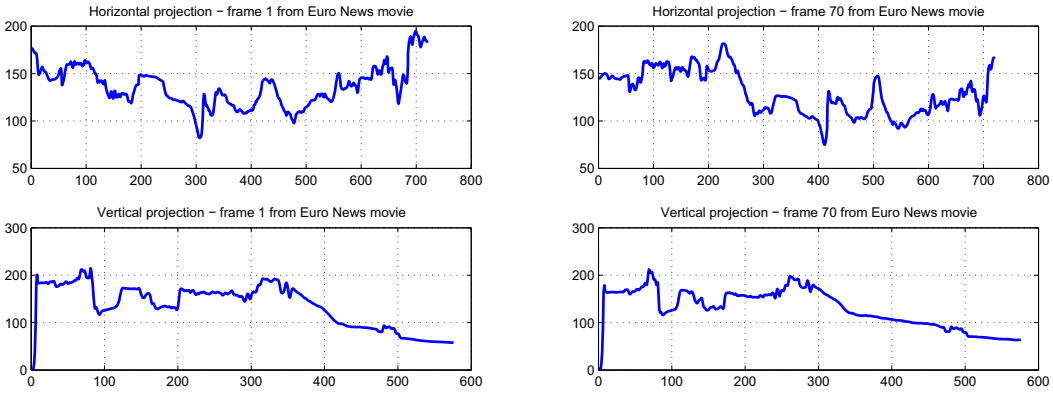


Figure 3.5: Horizontal and vertical projections for frames 1 and 70 from Euro News movie.

3.1 Scale Estimation using Cross-Correlation of Slices in the Frequency Domain

In the previous section we found the relations between slices of the Fourier transforms of frames before and after a scale change. Next, we want to estimate the scale from these slices. By converting the frequency axes in equation (3.6) to logarithmic scale we have:

$$|S_2\left(logf_x\right)| = \frac{1}{\alpha^2}|S_1\left(logf_x - log\alpha\right)| \tag{3.7}$$

Estimation of the scaling factor can be done by performing a cross-correlation between both sides of equation (3.7), and finding where the maximum is achieved. We work only in a part of the frequency band. Since natural images are smooth, high frequencies will not contains much information. We also don't want to work with frequencies close to zero becuase of the numerical problems when changing to logarithmic scale. Hence, we selected the band edges as:

$$f_{min} = 0.2\pi, f_{max} = 0.8\pi \tag{3.8}$$

The relation between the frequencies in logarithmic scale is:

$$\begin{aligned} \alpha &= base^{-lag} \\ base &= \left(\frac{f_{max}}{f_{min}}\right)^{\frac{1}{N-1}}, \end{aligned} \tag{3.9}$$

where N is the number of points in the logarithmic scale. Given the desired resolution we can calculate the number of points to use as will be shown in section 4.1.

We can repeat the estimation process for the vertical slice $f_x = 0$, and then average the scale estimation results.

We demonstrate this process in the following example. We took a frame from the Euro News movie. We rescaled it by a scale factor of 0.9 using bilinear interpolation. The original frame and the frame after the scale change are shown in Fig. 3.6.

The horizontal projections for these frames are shown in Fig. 3.7.

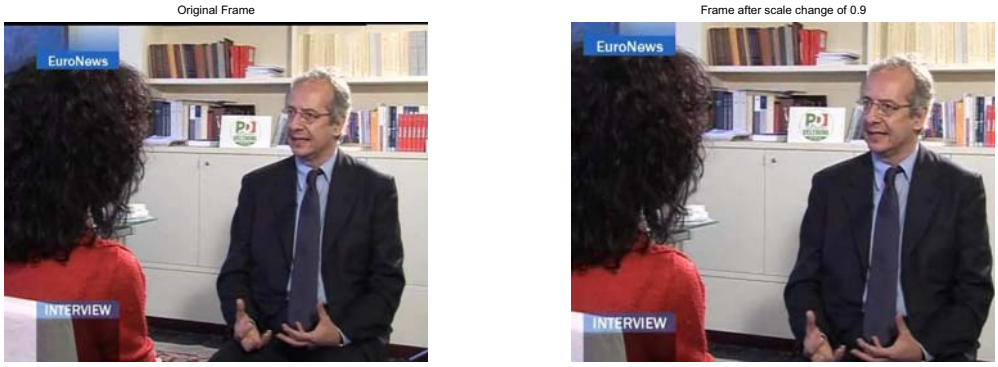


Figure 3.6: Original frame and the frame after scale change of 0.9.

In the next step, we calculate the Fourier transform of the projections. According to the slice-projection transform, we get the slices through the origin of the two-dimensional Fourier transform of the entire frame, which are parallel to the projection line. Before calculating the Fourier transform, we multiply the projections by windows to reduce finite-length effects. The magnitudes of the slices are shown in Fig. 3.8. Note, that the frequency axis is in logarithmic scale.

In the final step, we perform a cross-correlation between the magnitudes of the slices (in logarithmic scales). By equation (3.9), the lag where the maximum is achieved corresponds to the estimated scale. We can clearly see the peak at $\alpha = 0.9$ in Fig. 3.9.

3.2 Scale Estimation using the Mellin Transform in the Frequency-domain

The cross-correlation method for estimating the scale has high complexity. The resaon for the high complexity is that we need to calculate the cross-correlation for a large number of lags (typically, hundreds) in order to achieve good resolution for the scale estimation. In a typical scene, there are up to 1000 frames. We need a good resolution becuase the estimation errors are accumulated. For example, if instead of $\alpha = 1$ we estimate $\alpha = 1.001$ for 1000

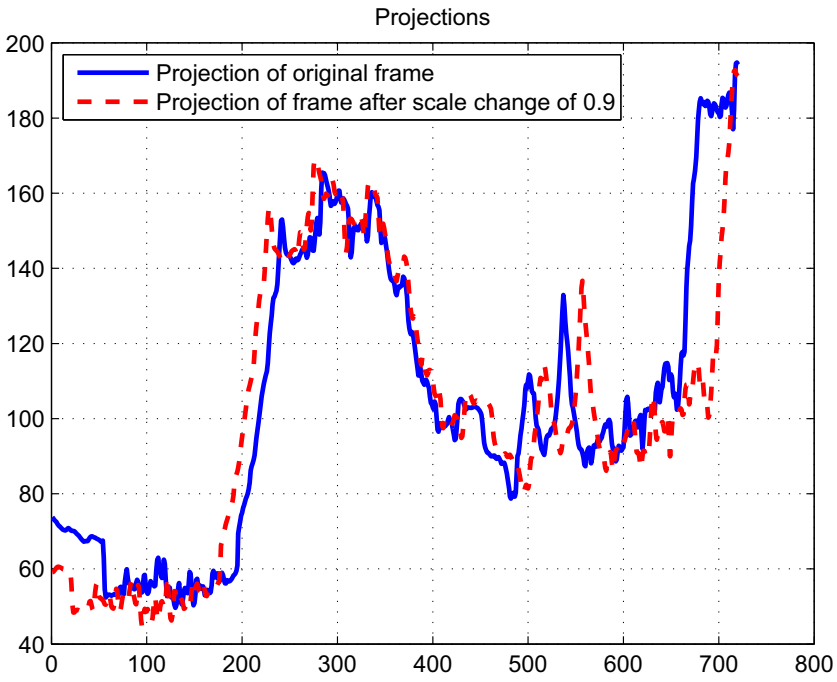


Figure 3.7: Horizontal projections of the original frame (solid line) and the frame after scale change(dashed line).

frames, we have a total scale factor of $\alpha = 1.001^{1000} \approx 2.7$. In section 4.1 we will discuss further the complexity of the algorithms considered in this chapter. In order to reduce the complexity we seek a transform that after applying it to the magnitudes of the slices, the scale parameter will become a multiplicative parameter.

We start by considering the 2D Analytical Fourier-Mellin transform [28]. This transform can be used to estimate scale and rotation. We show how it can be used for estimating scale and shifts parameters. Then, we will show that for our model we can use just a 1D Mellin transform, using the slice-projection theorem. The Mellin transform for scale estimation was introduced in [6].

Definition - Analytical Fourier-Mellin Transform (AFMT):

$$M_f(k,v) = \frac{1}{2\pi}\int_0^{\infty}\int_0^{2\pi} f(r,\theta)\, r^{\sigma - jv} e^{-jk\theta} d\theta \frac{dr}{r} \tag{3.10}$$

Where σ is a parameter.

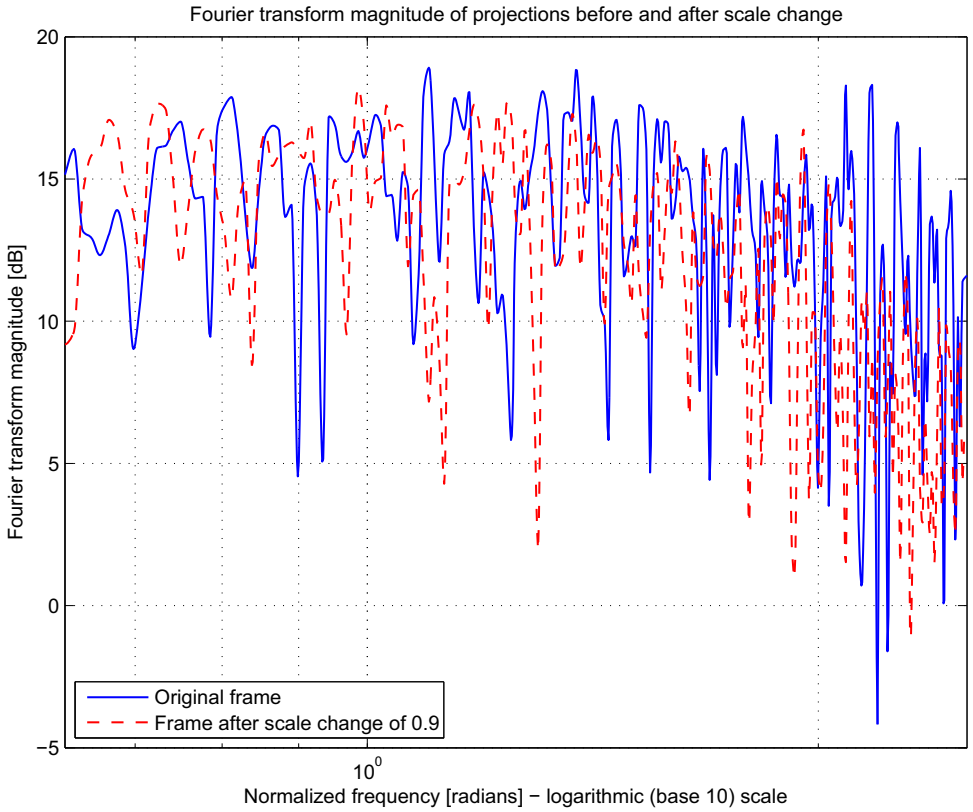


Figure 3.8: Magnitudes of Fourier transform of the projections before (solid line) and after (dashed line) the scale change.

For frames related by scale and rotation, $f_2(r,\theta) = f_1(\alpha r, \theta + \beta)$ we have:

$$\begin{aligned} M_{f_2}(k,v) &= \frac{1}{2\pi}\int_0^{\infty}\int_0^{2\pi} f_1(\alpha r, \theta+\beta)\, r^{\sigma - jv} e^{-jk\theta} d\theta \frac{dr}{r} \\ &= \alpha^{-\sigma+jv} e^{jk\beta} M_{f_1}(k,v) \end{aligned} \tag{3.11}$$

Where α is the scale parameter and β the rotation parameter.

In appendix A we show how to estimate the scale parameter using the AFMT for our model which is given by 3.1.

Calculating the AFMT on the entire image has high complexity. Instead, we will use the one dimensional Mellin transform on a slice from the Fourier transform. As before, the relation between the Fourier transforms is given by (3.2). And if we take the absolute value of the slice corresponding to $f_y = 0$ (or $f_x = 0$) we will have the relation given in equation (3.6).

Figure 3.9: Cross-Correlation of magnitudes of slices of 2D-Fourier transform of slices before and after scale change.

Definition - One-dimensional Mellin transform:

$$M(s) = \int f(x)\, x^{s-1} dx \tag{3.12}$$

Where s is a parameter.

The relation between the one-dimensional Mellin transform of the absolute values of the Fourier transforms of the projections is given by

$$\begin{aligned} M_2(s) &= \int |S_2(x)|\, x^{s-1} dx = \frac{1}{\alpha^2} \int \left| S_1\left(\frac{x}{\alpha}\right) \right| x^{s-1} dx \\ &= \frac{1}{\alpha^2} \int |S_1(x)|\, (\alpha x)^{s-1}\, \alpha dx = \alpha^{s-2} \int |S_1(x)|\, x^{s-1} dx \\ &= \alpha^{s-2} M_1(s) \end{aligned} \tag{3.13}$$

Now we can estimate the scale by:

$$\alpha = \left[\frac{M_1(s)}{M_2(s)}\right]^{\frac{1}{2-s}} \tag{3.14}$$

Near $s = 2$ we may have numerical problems. However, experimental results show that we should use values of s around 2. Normalizing the Fourier transforms by $F_1(0), F_2(0)$, we get

$$\begin{aligned} \left|\hat{S}_1(f_x)\right| &= \frac{|S_1(f_x)|}{|F_1(0)|} = \frac{|F_1(f_x,0)|}{|F_1(0)|} \\ \left|\hat{S}_2(f_x)\right| &= \frac{|S_2(f_x)|}{|F_2(0)|} = \frac{|F_2(f_x,0)|}{|F_2(0)|} \\ \left|\hat{S}_2(f_x)\right| &= \left|\hat{S}_1\left(\frac{f_x}{\alpha}\right)\right| \end{aligned} \tag{3.15}$$

Applying the 1D-Mellin transform to the normalized slices gives:

$$M_2(s) = \alpha^s M_1(s) \tag{3.16}$$

The scale estimation is now given by:

$$\alpha = \left[\frac{M_1(s)}{M_2(s)}\right]^{\frac{1}{-s}} \tag{3.17}$$

The parameter s has a significant effect on the estimation. We want to find an empirical value of s that yields the lowest error for the scale estimation. We used hundreds of frames from our video samples, changed their scale and then estimated the scale using different values of s. We then found the parameter s which resulted in the best result for each scale. We can see in Fig. 3.10, that the best empirical value of s is different for different scale factors.

Since we don't have the true scale, we suggest to use an iterative method to estimate the scale. We begin the estimation with $s = 1$, and then update it iteratively by the previous scale estimation until convergence. The results are shown in Fig. 3.11. We can see that the estimation results using the Mellin transform are not good as the results using the slice cross-correlation, even when using the iterative method. In the next subsections we will investigate the problem further and propose an improvement.

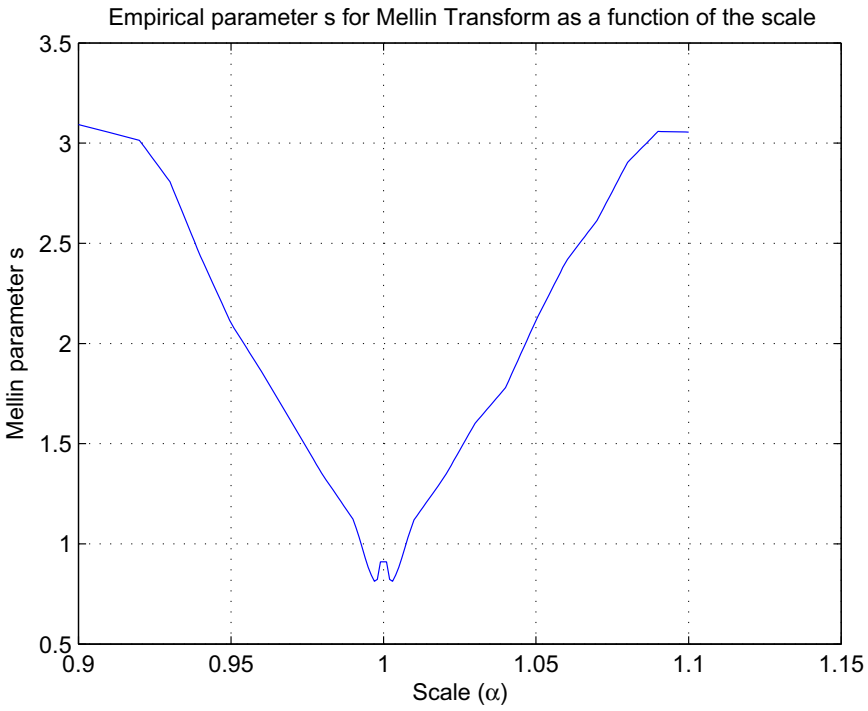


Figure 3.10: Empirical parameter s for Mellin as a function of the scale α.

3.3 Effect of Fixed-Size Display

The results for the scale estimation using the Mellin transform on the absolute of the Fourier transform are not good enough. The main reason for that are the effects caused by the fixed size of the display. The global motion causes some part of the frame not to appear in the next frame. We can see in Fig. 3.12 an example for the effects on a projection in a case of zoom in. The projection without the effects caused by the fixed-size display is longer than the projection with the effects. This is because after zoom in, some of the picture is out of the frame. Also, we can see that the amplitude of the projection with the effects is different than the projection without the effects. This is due to the fact that the most upper and lower rows are out of the frame.

We can solve this problem by zeroing the parts of the each frame that do not appear in the other frame. An example can be seen in Fig. 3.13.

After this operation, we have almost no edge effects. Some edge effects may still appear because we are working at a one-pixel resolution. Assum-

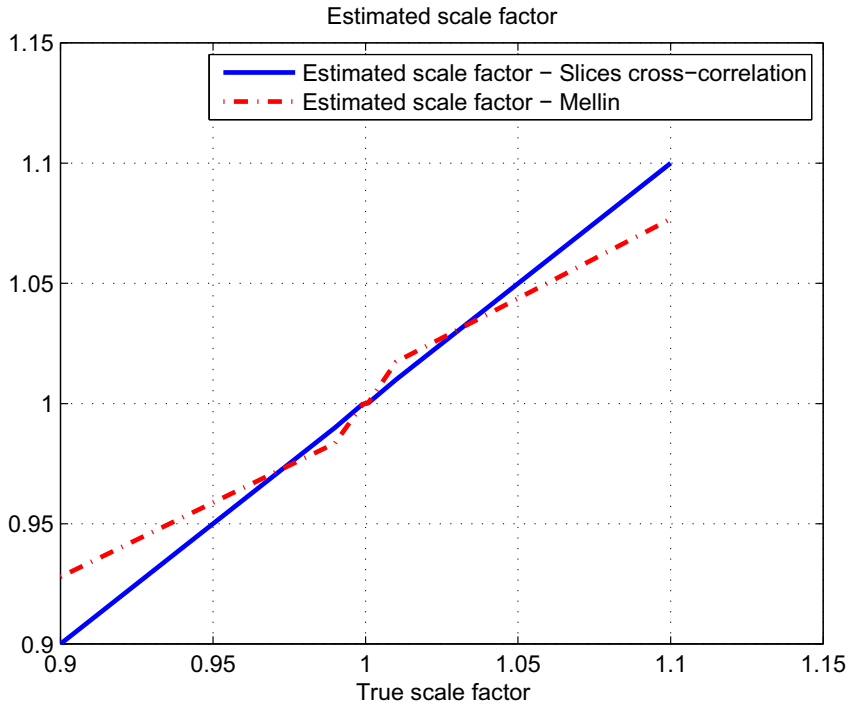


Figure 3.11: Scale Estimation results using iterative method.

ing that changes between scales and shifts for consecutives frames are small, and assuming we have previous scale and shifts estimation, we can use these estimations as a good guess for removing the edge effects.

3.4 Scale Estimation using Spatial-Domain Mellin transform

We assume that we have the previous frame estimated scale and shifts and that they are also good estimates for the current parameters. We perform the vertical and horizontal projections on the current frame after zeroing out parts of the frame to reduce the effect of the fixed-size. The relation between the projections after normalization by the number of summed pixels is:

$$P_2(x) = P_1(\alpha x + d_x) \tag{3.18}$$

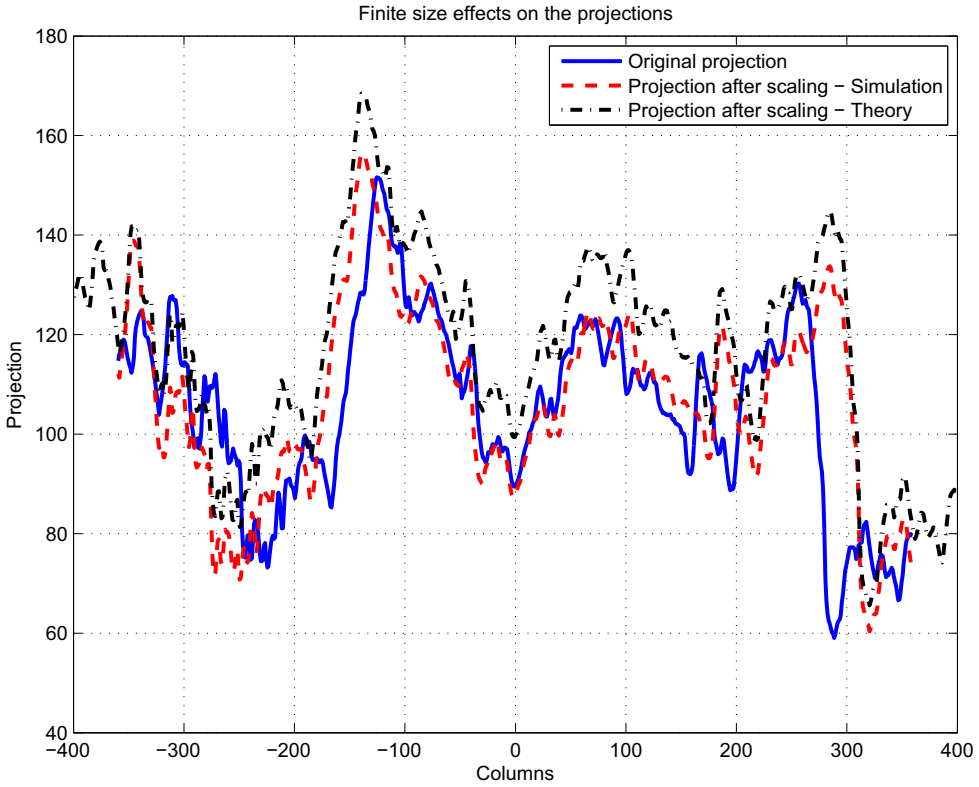


Figure 3.12: Fixed-size display effects on the projections.

Since we assume that the we have a good estimation for the shift d_x from the previous frame, we can perform shift correction and then the relation between the projections is:

$$P_2(x) = P_1(\alpha x) \tag{3.19}$$

Since we have the shifts estimations, we can apply the Mellin transform directly on the projections in the spatial domain. We can estimate the scale by the Mellin transform. The relation between the Mellin transforms of both sides of equation (3.19) is given by:

$$\begin{aligned} M_{p_2}(s) &= \int P_2(x)x^{s-1}dx = \int P_1(\alpha x)x^{s-1}dx \\ &= \int P_1(x)\left(\frac{x}{\alpha}\right)^{s-1}\frac{1}{\alpha}dx = \alpha^{-s}\int P_1(x)x^{s-1}dx \\ &= \alpha^{-s}M_{p_1}(s) \end{aligned} \tag{3.20}$$

(a) (b)

(c) (d)

Figure 3.13: Example of reducing fixed-size display effects. Image (a) is the previous frame. Image (b) is the current frame which is a scaled and shifted version of the previous frame. Image (c) is the previous frame after zeroing out the parts that do not appear in the current frame. Image (d) is the current frame after zeroing out the parts that do not appear in the previous frame.

Therefore, the scale estimation is:

$$\alpha = \left[\frac{M_{p_1}(s)}{M_{p_2}(s)}\right]^{\frac{1}{s}} \tag{3.21}$$

The estimation results for the Mellin transform in the spatial domain, after reducing the fixed-size display effects, are very good as can ce seen in Fig. 3.14.

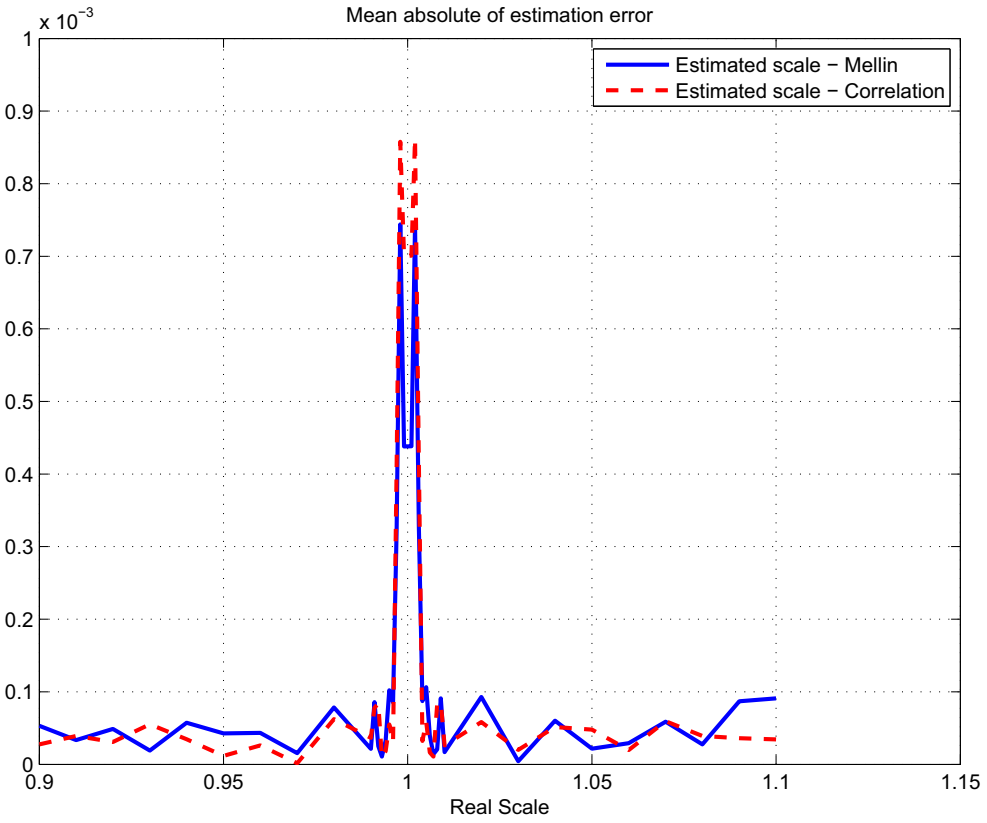


Figure 3.14: Comparison between estimation results using correlation in the frequency domain and spatial-domain Mellin transform.

We suggest estimating the scale and shift parameters between the first two frames in each scene using the slices cross-correlation method described in section 3.1. For the following frames, up to the end of the scene, we suggest estimating the scale using the spatial domain Mellin transform. After estimating the scale α, we will estimate the translation parameters as described

in section 3.5.

We repeat here the process for finding the empirical parameter s. We used hundreds of pictures, changed their scale and then estimated the scale using different values of s. Reducing the fixed-size display effect should result in very good performance for any parameter s. Therefore, we determined the parameter s that yields the best results for noisy frames. Gaussian noise with a standard deviation σ is added to every pixel in the frame, so that $SNR = 20dB$. The mean value of a pixel, $MeanPixelValue$, is 128.

$$SNR = 20log_{10}\left(\frac{MeanPixelValue}{\sigma}\right) \tag{3.22}$$

We then found the parameter s which yields the best result for each scale. The results are shown in Fig. 3.15. The solid line in the figure connects the discrete values of s for the true scale α that were tested.

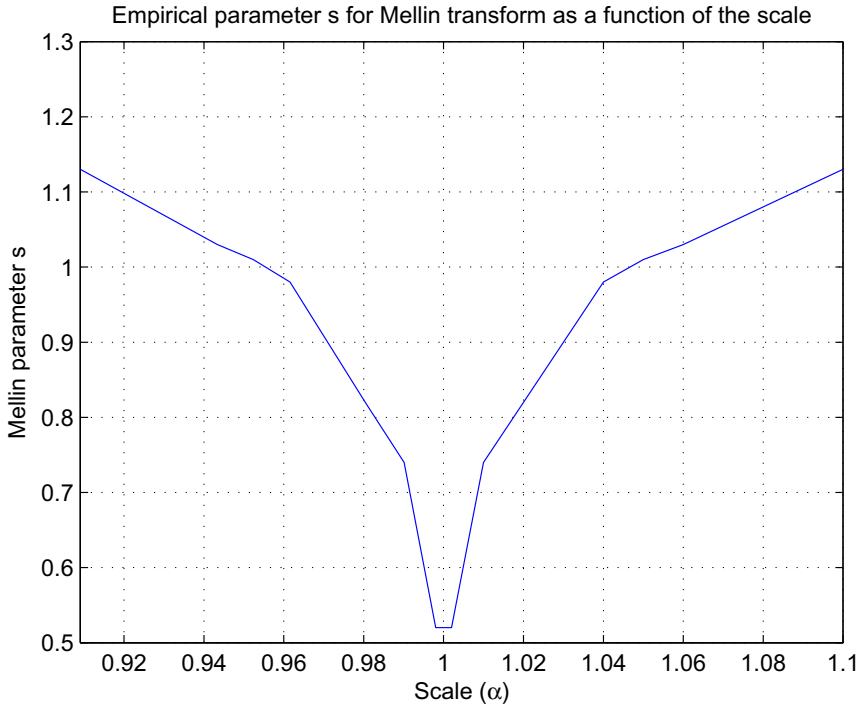


Figure 3.15: Empirical parameter s for spatial-domain Mellin transform as a function of the scale α.

The improvement using empirical parameter s is shown in Fig. 3.16. We

calculated the mean square error (MSE) of the estimated scale value using different values of the Mellin transform parameter s, for the true scale of $\alpha = 0.99$. We can see in the figure that using the empirical parameter s improves significantly the results.

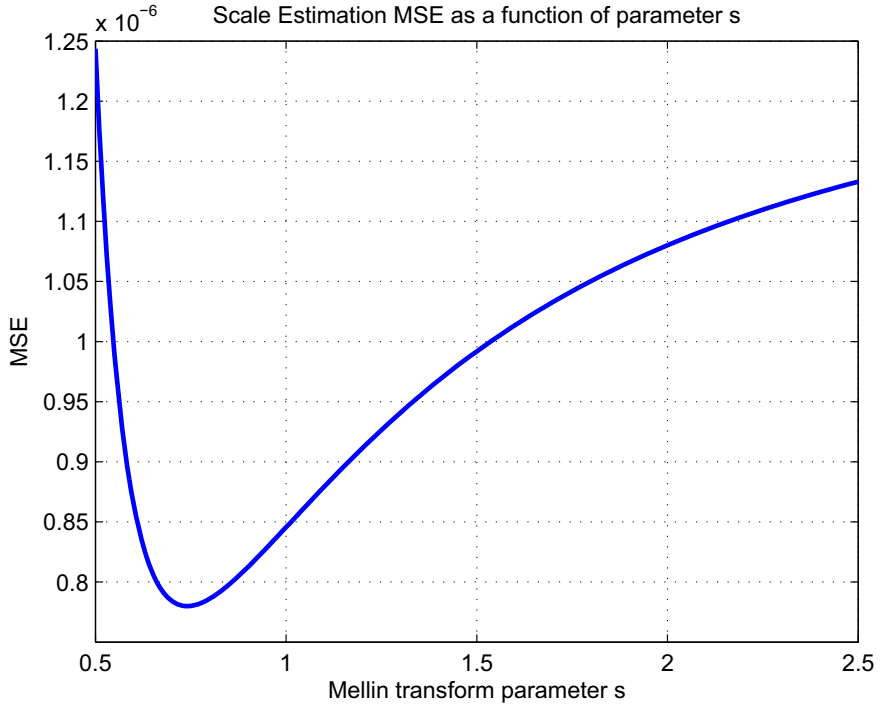


Figure 3.16: Scale Estimation MSE as a function of the Mellin transform parameter s.

3.5 Global Translation Estimation

After estimating the scale factor between any two consecutive frames, we can estimate the horizontal and vertical shifts between them. The relation between the projections is given by equation (3.18). The translation is estimated by first scaling the projection of the previous frame f_1 with the estimated scale α. We define the scaled projection as $P_1^r(x)$.

$$P_1^r(x) = P_1(\alpha x) \tag{3.23}$$

The relation between the scaled projection for the previous frame and the

projection from the current frame:

$$P_1^r(x) = P_2\left(x - \frac{d_x}{\alpha}\right) \tag{3.24}$$

The translation is estimated by performing cross-correlation between the scaled projection and the projection from the current frame f_2.

We demonstrate this by an example. We took a frame and created a new frame by scaling it by 0.9 and then shifting it by 5 pixels. After the scale estimation, by any of the methods described in the previous sections, we scale the projection of the previous frame so it has the same scale as the current frame. An example of the projections after scaling is given in Fig. 3.17. We can see that now the projections have the same scale, but are shifted. The shift is estimated performing a cross-correlation between the projections.

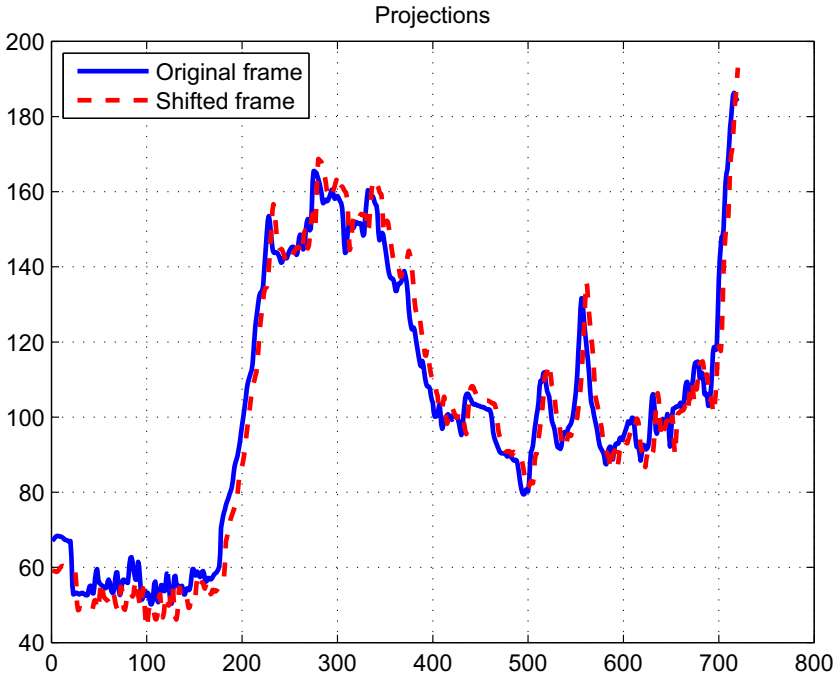


Figure 3.17: Projection of a frame and its shifted version.

3.6 Local Motion Estimation

In the previous sections we estimated the global motion caused by the camera. In this section we estimate the local motion, which is typically the motion of the speaker inside the ROI. The ROI window is updated according to the estimated global motion parameters. Next, we need to find if there was a local motion inside the ROI. For example, a talking man moving his head. Our assumption is that there is no local scale change (like in the case of a man approaching the camera).

We calculate horizontal and vertical projections only for the ROI in the previous frame and in the current frame. We scale, using bilinear interpolation, the projections from the previous frame with the estimated global scale factor. We then estimate local horizontal and vertical shifts by performing cross-correlation between the scaled projection for the previous ROI and the projection from the current ROI in the spatial domain.

Chapter 4

Computational Complexity and Performance Analysis of ROI Tracking Algorithms

In the previous chapter we have developed novel algorithms for ROI tracking in video. In this chapter we will analyse the algorithms. In the first section we will calculate the computational complexity of the algorithms. We will calculate the number of multiplications and additions needed for each algorithm. In the second section we will propose a statistical model. This model enable us to calculate the probabilty distribution function of the estimated scale. We will test the model results by simulations.

4.1 Complexity of ROI Tracking Algorithms

We compare in this section the complexity of scale estimation in two of the proposed methods. We compare the complexity of the method using cross-correlation of slices in the frequency domain, described in section 3.1 and the complexity of method using spatial-domain Mellin transofrm, described in section 3.4.

In the first method we perform slice cross-correlation in a logarithmic scale. We work only in the frequency band given in equation (3.8). The relation between the frequencies in a logarithmic scale is given by equation (3.9). We

mark by N_1 the number of points in the logarithmic scale. Then:

$$base = \left(\frac{f_{max}}{f_{min}}\right)^{\frac{1}{N_1-1}} \tag{4.1}$$

Given the desired resolution we wish to calculate the number of points to use. Since we use a logarithmic scale it is hard to define a resolution because, the resolution is different for different values of scale. We will find the resolution around the scale value of 1. The scale estimation is done by:

$$\alpha = base^{-lag} \tag{4.2}$$

Where lag is the value found for maximum cross-correlation.

The next possible value of scale after $\alpha = base^0 = 1$ in the logarithmic scale is $\alpha = base^1$. Therefore the resolution near $\alpha = 1$ is:

$$\Delta\alpha = base^1 - base^0 = base - 1 \approx resolution \tag{4.3}$$

Assume that we want a resolution of 0.001 we have:

$$\begin{aligned} resolution &= 0.001 \\ \Rightarrow N_1 &= 1388 \end{aligned} \tag{4.4}$$

We want to find scale values in the range $\alpha \in [0.8, 1.25]$, $\alpha_{min} = 0.8, \alpha_{max} = 1.25$. We need $2M - 1$ shifts in the correlation.

$$M = \frac{log\,(\alpha_{max})}{log\,(base)} = \frac{-log\,(\alpha_{min})}{log\,(base)} \approx 223 \tag{4.5}$$

Therefore, the number of operations (multiplications and additions) for the correlation is $(2M-1)\,N_1 \approx 617,660$. We can assume that the scale α does not change much between consecutives frames. Therefore, except for the pair of frames in each scene, we can use the previous estimation of the scale as an initial guess. Then, we should calculate the cross-correlation only in the vicinity of the previous estimated scale value. From empirical experience, it is enough to calculate the cross-correlation for 20 points around the previous estimated value. Therefore, for all the frames, except the first pair of frames,

we have only $54,132$ operations.

For the method using Mellin transform in the spatial domain, we don't need to perform a Fourier transform. Therefore, for this method the number of operations is equal to the length of the projection.

The rest of the complexity analysis is given in Appendix B. We assume the frame has width $W = 720$ and height $H = 576$ and frame rate of 30 frames per second.

The results are summarized in Table 4.1.

Method	Additions(MIPS)	Multiplications(MIPS)
2D Fourier transform	1,082.9	1,082.9
Slice cross-correlation - first pair	43.8	18.9
Slice cross-correlation	26.9	2.0
Spatial-domain Mellin transform	25.5	0.5

Table 4.1: Summary of algorithms complexity

Considering that we will perform the correlation method only once in a scene, we have that the average complexity is around $25.5MIPS$. It can be easily implemented for real-time applications on a Pentium with a 3GHz processor. The projections are the most consuming operation and they can be implemented using parallel computing, so the algorithm can work efficiently on a single computer with multiple processors.

4.2 Performance Analysis using a Statistical Model

In this section we calculate the variance of the scale estimation error using the spatial-domain Mellin transform method. For that purpose, we first calculate the probability distribution function of the estimated scale. We assume, as in the previous section, that the frame has width W and height H. We assume that a Gaussian noise with a standard deviation σ is added to every pixel in the frame. Using the pdf of the estimated scale we will calculate the distribution of estimation errors for different true scale values and different signal to noise ratios (SNR). The first operation is the projection. For example, we will

consider the vertical projection:

$$P(x) = \frac{1}{H}\sum_{i=1}^{H} f(x, y_i) \tag{4.6}$$

The projection operation reduces the noise variance by the value of height H. We mark with tilde the noisy functions.

$$\begin{aligned} \tilde{P}(x) &= \frac{1}{H}\sum_{i=1}^{H} f(x, y_i) + n(x, y_i) = P(x) + n(x) \\ \sigma_n^2 &= \frac{\sigma^2}{H} \end{aligned} \tag{4.7}$$

We perform the projection operation for two consecutive frames. Before applying the Mellin transform we have to shift one of the projections using the estimated shift from the previous frame as described in the previous section. Instead of calculating the projection's samples after the shift, we can use the same samples with a shifted x-axis. The Mellin transform in that case is:

$$M(s) = \int_x P(x - d_x)x^{s-1}dx = \int_x P(x)(x + d_x)^{s-1}dx \tag{4.8}$$

Therefore, we can assume that the noise after the adjustment is still Gaussian with the same standard deviation as before.

The noisy Mellin transform, for a fixed scalar parameter s, are given by:

$$\begin{aligned} \tilde{M}(s) &= \frac{1}{W}\sum_x \tilde{P}(x)\, x^{s-1} \\ &= \frac{1}{W}\sum_x [P(x) + n(x)]\, x^{s-1} \\ &= M(s) + \frac{1}{W}\sum n(x)\, x^{s-1} \\ &= M(s) + n_M \end{aligned} \tag{4.9}$$

The variance of the noise is now given by:

$$\sigma_{n_M}^2 = \frac{\sigma^2}{W^2 H}\sum_x \left(x^{s-1}\right)^2 \tag{4.10}$$

We know that the scale estimation is given by equation (3.21). Therefore,

we have:

$$\tilde{\alpha}^s = \frac{\tilde{M}_{p_1}(s)}{\tilde{M}_{p_2}(s)} = \frac{M_{p_1}(s) + n_{M_1}}{M_{p_2}(s) + n_{M_2}} \tag{4.11}$$

The noises n_{M_1}, n_{M_2} are both Gaussian noises with known variances. The noises are assumed to be uncorrelated. Therefore:

$$\begin{aligned} \tilde{M}_{p_1}(s) &\sim \mathcal{N}(M_{p_1}(s), \sigma_{n_M}) \\ \tilde{M}_{p_1}(s) &\sim \mathcal{N}(M_{p_1}(s), \sigma_{n_M}) \end{aligned} \tag{4.12}$$

We want to find the distribution of the estimated scale $\tilde{\alpha}$ by equation (4.11). As shown in Appendix C:

$$\begin{aligned} f_{\tilde{\alpha}}(\tilde{\alpha}) &= |s\tilde{\alpha}^{s-1}| \frac{b(\tilde{\alpha}^s) \cdot c(\tilde{\alpha}^s)}{a^3(\tilde{\alpha}^s)} \frac{1}{\sqrt{2\pi}\sigma_{n_M}\sigma_{n_M}} \left[2\Phi\left(\frac{b(\tilde{\alpha}^s)}{a(\tilde{\alpha}^s)} \right) - 1 \right] \\ &+ \frac{1}{a^2(\tilde{\alpha}^s) \cdot \pi\sigma_{n_M}\sigma_{n_M}} e^{-\frac{1}{2}\left(\frac{M_{p_1}^2(s)}{\sigma_{n_M}^2} + \frac{M_{p_2}^2(s)}{\sigma_{n_M}^2} \right)} \end{aligned} \tag{4.13}$$

Where,

$$\begin{aligned} a(\tilde{\alpha}^s)) &= \sqrt{\frac{1}{\sigma_{n_M}^2}\tilde{\alpha}^2 s + \frac{1}{\sigma_{n_M}^2}} \\ b(\tilde{\alpha}^s) &= \frac{M_{p_1}(s)}{\sigma_{n_M}^2} z + \frac{M_{p_2}(s)}{\sigma_{n_M}^2} \\ c(\tilde{\alpha}^s) &= e^{\frac{1}{2}\frac{b^2(\tilde{\alpha}^s)}{a^2(\tilde{\alpha}^s)} - \frac{1}{2}\left(\frac{M_{p_1}^2(s)}{\sigma_{n_M}^2} + \frac{M_{p_2}^2(s)}{\sigma_{n_M}^2} \right)} \\ \Phi(\tilde{\alpha}^s) &= \int_{-\infty}^{\tilde{\alpha}^s} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}\, du \end{aligned} \tag{4.14}$$

We can repeat the above estimation process for the horizontal projection. Hence, assuming that we have scale estimations $\tilde{\alpha}_V, \tilde{\alpha}_H$ using vertical and horizontal distributions, and that we have $f_{\tilde{\alpha}_V}(\tilde{\alpha}_V), f_{\tilde{\alpha}_H}(\tilde{\alpha}_H)$- the corresponding probability distribution functions, we can estimate the scale averaging of them.

$$\tilde{\alpha} = \frac{1}{2}(\tilde{\alpha}_V + \tilde{\alpha}_H) \tag{4.15}$$

For simplicity we will assume that the noises in two estimations are independent. This is only an approximation, since in the projections stage we

are summing the noise along the columns, for the horizontal projection, and along the rows for the vertical projections. Therefore, the noises that are being summed are from different pixles, except for only one. Its contribution is negligible and therefore, we can assume that after the projections the noises are independent. The probability distribution functions of $0.5\tilde{\alpha}_V, 0.5\tilde{\alpha}_H$ is given by:

$$\begin{aligned} f_{0.5\tilde{\alpha}_V}(x) &= 2f_{\tilde{\alpha}_V}(2x) \\ f_{0.5\tilde{\alpha}_H}(x) &= 2f_{\tilde{\alpha}_H}(2x) \end{aligned} \tag{4.16}$$

The probability density function of the sum of two independent random variables is the convolution of their separate density functions:

$$f_{\tilde{\alpha}}(\tilde{\alpha}) = \int_{-\infty}^{\infty} f_{0.5\tilde{\alpha}_V}(y)\, f_{0.5\tilde{\alpha}_H}(x-y)\, dy \tag{4.17}$$

We will now show an example of the estimated probability distribution function. We use $s = 1$ as the Mellin transform parameter, and assume that the mean value of a pixel, $MeanPixelValue$, is 128. Therefore, from equations (4.7) and (4.9), we can also assume that $M_{p_1}(s) = 128$. Given an SNR (signal to noise ratio) we can calculate the standard variation of the noise σ.

$$\begin{aligned} SNR &= 20log_{10}\left(\frac{MeanPixelValue}{\sigma}\right) \\ \Rightarrow \sigma &= 10^{-SNR/20} MeanPixelValue \end{aligned} \tag{4.18}$$

Using σ^2 we can calculate the variance of the noise after the projection and the Mellin transform $\sigma^2_{n_M}$ as described above.

Examples for probability distribution functions for different SNR values and the true scale α are shown in Fig. 4.1.

We performed simulations to test the model. We randomly selected frames from our movies. For each frame we created another frame with a scale change of 0.95,1 and 1.05. Then we added random Gaussian noise to each pixel in both frames, so that $SNR = 10dB$. We estimated the scale change using the spatial-domain Mellin transofrm as was described in section 3.4. We calculated an histogram of the estimated values. The normalized histogram is compared

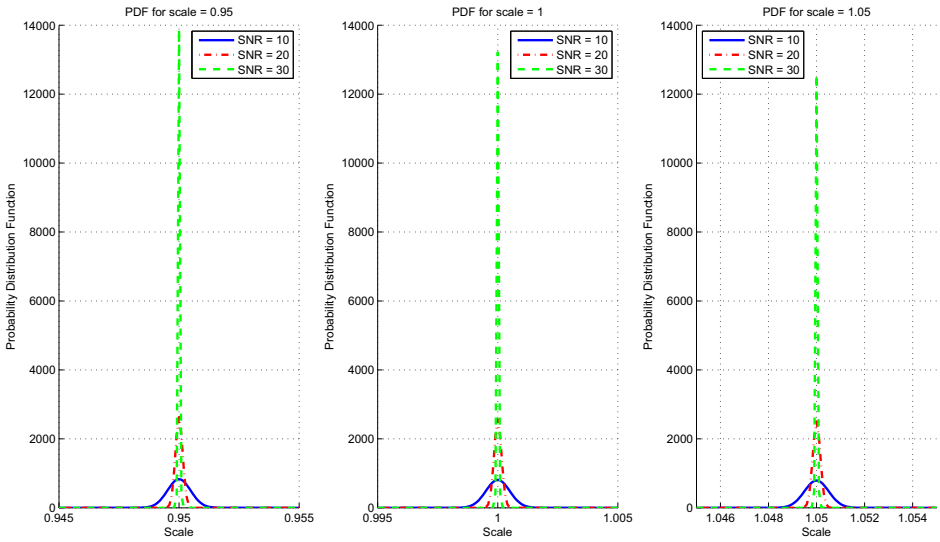


Figure 4.1: Examples for probability distribution functions of scale estimation.

to the probabilty density function in Fig. 4.2. We can see from the figure that the simulations results are quite similar to those expected from theory.

We used the same noisy frames to compare the estimation methods. We estimated the scale change using frequency-domain cross-correlation as was described in section 3.1. We calculated an histogram of the estimated values. The normalized histogram is compared to the normalized histogram of the estimated values using the spatial-domain Mellin transofrm method. The results are shown in Fig. 4.3. We can see that the spatial-domain Mellin transofrm method performs better than the frequency-domain cross-correlation.

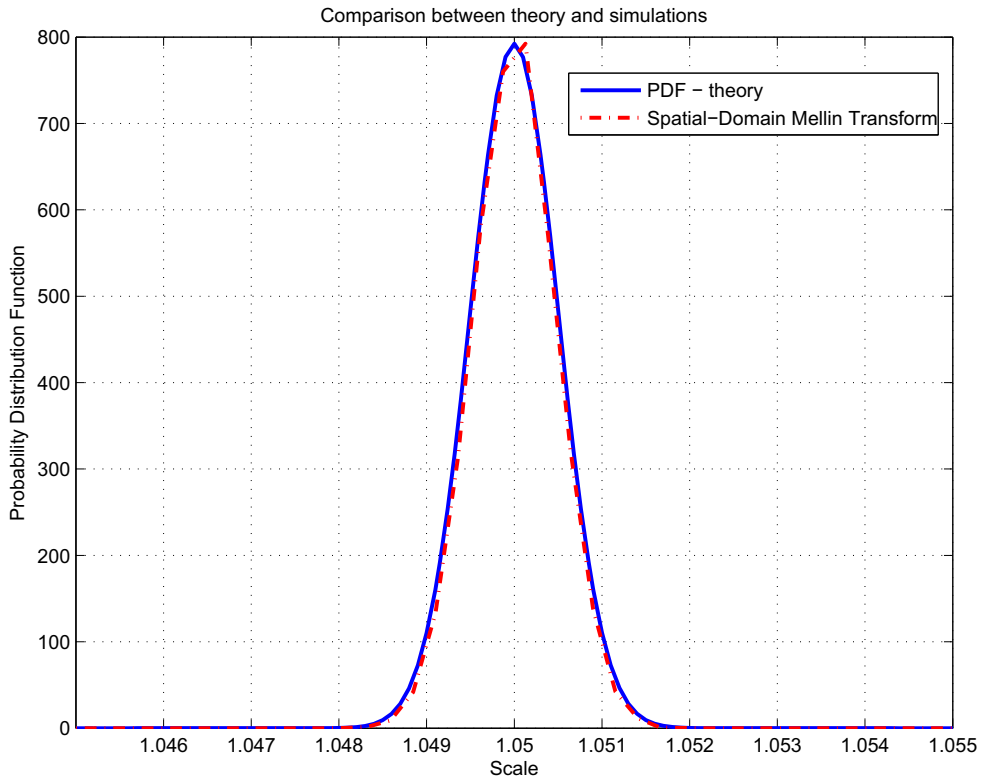


Figure 4.2: Comparison between estimation methods.

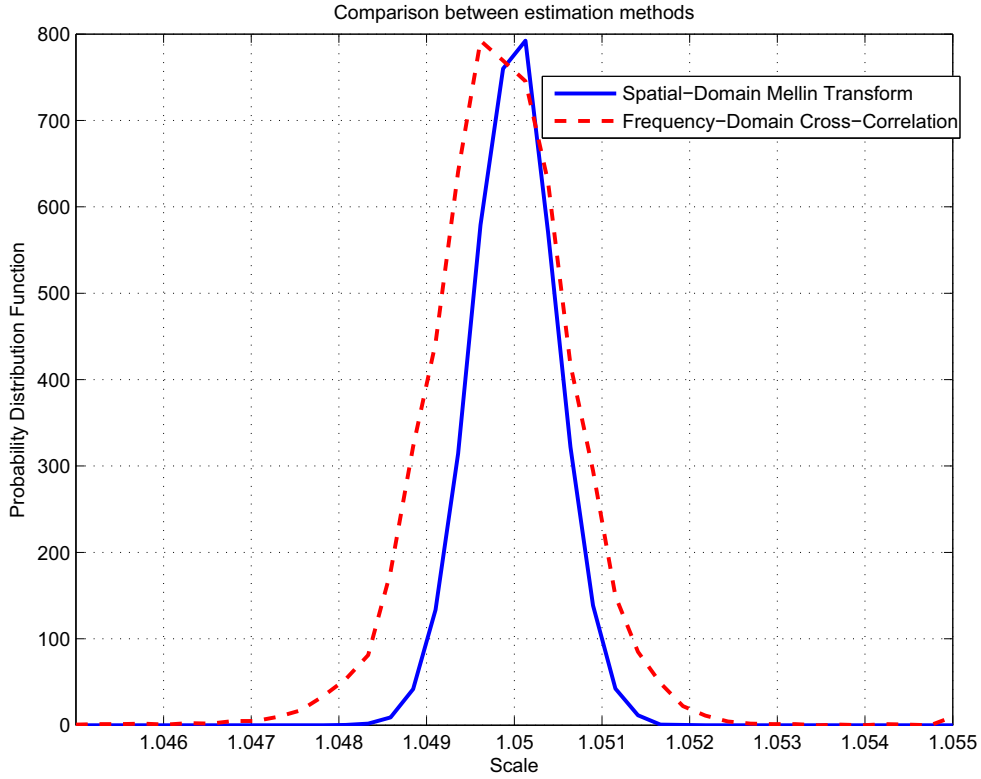


Figure 4.3: Comparison between estimation methods.

Chapter 5

Model-Based Region Tracking

In the previous chapter we introduced a novel algorithm for region tracking in video. We estimate the global motion caused by the camera movements and the local motion caused by the movement of the object (e.g., face) inside the region we want to track. The result of the tracking algorithm is the location of the region in every frame. This result can be jittery due to small estimation errors or small movements of the camera and the tracked object. Also, random noise and interferences can result in a wrong estimation of the ROI location in some of the frames. Consider, for exmaple, a reporter standing near a road. A passing car can cause a wrong estimation of the movement in several frames. In order to solve the jitter problem as well as the interferences problem we sugget in this chapter to use a dynamic model. The dynamic model assumes that the region movement has certain parameters. For example, the region might have a constant velocity with a small random acceleration. We use the estimated values of the global and local motions to estimate the state of the dynamic model, e.g. its velocity in a certain frame. By updating the region location according to the estimated state of the dynamic model, we force its motion to be much less jittery. Moreover, since the state of the dynamic model can be estimated from all the previous estimated values of the global and local motion, wrong motion estimated values for a few frames will have a minor effect on the tracking. Therefore, the tracking will be more robust and less sensitive to interferences, e.g., a passing car.

5.1 Dynamic Model

In this section we present the dynamic model that we use for describing the region movement.

We can define a state in frame k by:

$$S(k) = [x(k), y(k), w(k), h(k), v_x(k), v_y(k), v_w(k), v_h(k)]^T \tag{5.1}$$

Where
$x(k), y(k)$ - are the location coordinates of the center of the ROI.
$w(k), h(k)$ - are the width and height of the ROI.
$v_x(k), v_y(k), v_w(k), v_h(k)$ - are the corresponding velocities components.

We model the movement by:

$$S(k+1) = As(k) + n_s(k) \tag{5.2}$$

Where
$n_s(k)$ - is random noise, and

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We use $T = 1$ so that the units of the velocities are in pixels per frame.

The observations are given by:

$$z(k) = [x(k), y(k), w(k), h(k)]^T \tag{5.3}$$

$$z(k) = Hs(k) \tag{5.4}$$

Where,

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The observations vector $z(k)$ defines the location of the ROI in frame k.

5.2 Kalman Filtering

The Kalman filter, presented in [29], exploits the dynamics of the ROI, which governs its time evolution, to remove the effects of noise. It is used to get an improved estimate of the ROI location, e.g., in [39]. Since the results of the earlier described tracking methods are often too jittery, The Kalman filter can be used for smoothing the motion [9]. Assuming a linear model and that both state noise and measurement noise processes are Gaussian, as described in the previous section, the Kalman filter is an optimal (in terms of MMSE) state estimator [31].

We can model the movement, for example, in the x direction by:

$$\begin{aligned} x(k+1) &= x(k) + v_x(k)T + \frac{a_x(k)T^2}{2} + o(T^3) \\ v_x(k+1) &= v_x(k) + a_x(k)T + o(T^2) \end{aligned} \tag{5.5}$$

Where k is the frame number, x is the location, v_x is the velocity, and a_x is the acceleration. The movement in the y direction and the change in width or height can be modeled in the same way. We assume that the acceleration is a random noise. Therefore, we can re-write equation (5.2) as

$$S(k+1) = AS(k) + \Gamma a(k) \tag{5.6}$$

Where $S(k)$ and A are as defined in equation (5.2), and

$$\Gamma = \begin{pmatrix} \frac{T^2}{2} & 0 & 0 & 0 \\ 0 & \frac{T^2}{2} & 0 & 0 \\ 0 & 0 & \frac{T^2}{2} & 0 \\ 0 & 0 & 0 & \frac{T^2}{2} \\ T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{pmatrix}$$

$$a(k) = [a_x(k), a_y(k), a_w(k), a_h(k)]^T$$

The acceleration vector $a(k)$ is assumed to be Gaussian random noise with a known diagonal covariance matrix $Q_a(k)$.

We define:

$$Q(k) = cov(\Gamma a(k)) = \Gamma Q_a(k) \Gamma^T \tag{5.7}$$

Where *cov* denotes the covariance matrix.

We define $\hat{S}(k|i)$ as the estimate of the state at time k given observations up to and including time i.

We define $P(k|i)$ - the error covariance matrix (a measure of the estimated accuracy of the state estimate) at time k given observations up to and including time i. The matrix is initialized by a zero matrix.

The Kalman filter equations are:

Predict:

$$\begin{aligned} \hat{S}(k|k-1) &= A\hat{S}(k-1|k-1) \\ P(k|k-1) &= AP(k-1|k-1)A^T + Q(k-1) \end{aligned} \tag{5.8}$$

Update:

$$\begin{aligned} K(k) &= H^T \left(HP(k|k-1)H^T \right)^{-1} \\ \hat{S}(k|k) &= \hat{S}(k|k-1) + P(k|k-1)K(k) \left(z(k) - H\hat{S}(k|k-1) \right) \\ P(k|k) &= \left(I - K(k)H \right) P(k|k-1) \end{aligned} \tag{5.9}$$

Where $K(k)$ is the Kalman filter gain. The observation vector $z(k)$ is the estimated ROI location and is the output of the tracking algorithm described in the previous chapter.

We use the estimated state to update the ROI location in the current frame. We use the observation matrix H given in 5.4. After the tracking algorithm, the ROI location in frame k is given by the observation vector $z(k)$.

$$\hat{z}(k) = H\hat{S}(k|k) \tag{5.10}$$

After calculating the Kalman filter equations given in (5.8),(5.9), we replace the ROI location in frame k with the estimated observation vector $\hat{z}(k)$, which is calculated using the estimated state of the dynamic model (5.10). The estimated observation vector contains the width, height and center of the ROI, similarly to (5.3). The state of the dynamic model is estimated from all the previous estimated values of the global and local motion. Therefore, the motion of the ROI should be much less jittery and the tracking should become more robust to small errors in global and local motion estimation. Comparison of tracking with and without a Kalman filter is shown in Fig. 5.1. The y-component of ROI center before and after Kalman filtering is shown for the first 50 frames of a video clip, where the camera is zooming-in while moving to the left. We can see that after Kalman filtering the curve is smoother.

Experimental results will be given in chapter 9.

5.3 Particle Filtering

Particle filtering was introduced in [30] for estimating Bayesian models. As a result of the popularity of particle methods, several tutorials have already been published on the subject [31, 32]. In the Bayesian approach to dynamic state estimation, one tries to estimate the posterior probability density function of the state, based on all available information, including the set of received measurements. The main idea of particle filtering is to represent the required posterior density function by a set of random samples (particles) with associated weights and to compute estimates based on these samples and weights. As the number of samples increases, this Monte-Carlo characterization be-

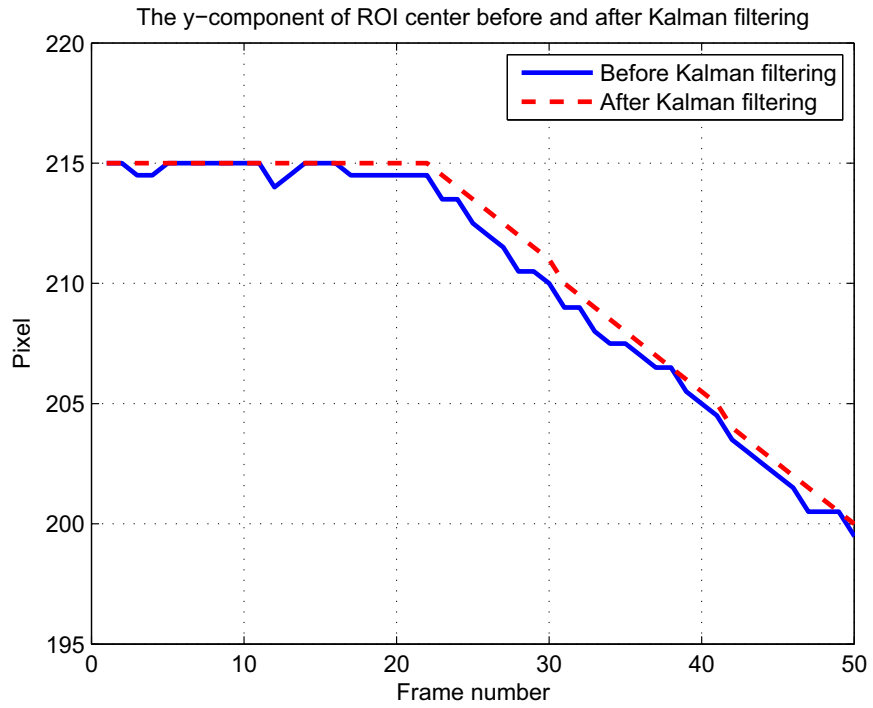


Figure 5.1: The y-component of ROI center before and after Kalman filtering.

comes an equivalent representation to the usual functional description of the posterior probability density function, and the particle filter approaches the optimal Bayesian estimate. If the true density is non-Gaussian (e.g., if it is bimodal or heavily skewed), then a Gaussian can never describe it well. For example, in case of interefence, such as a passing car in front of the speaker, the posterior probability density function is highly non-Gaussian. In such cases, particle filters will yield an improvement in performance in comparison to that of a Kalman filter and its extentions.

In this section we will describe the use of a particle filter for our tracking method and dynamic motion model.

Stage 1:

In the first frame of each scene, we are given the initial location of the ROI $[x(0), y(0), w(0), h(0)]^T$. We assume that the number of particles is N. We draw random particles with states around the initial location. The state of

particle i is given by:

$$S(0,i) = [x(0), y(0), w(0), h(0), 0, 0, 0, 0]^T + n_s^{init}(i) \tag{5.11}$$

Where $n_s^{init}(i)$ is the state noise in the first frame and particle i and it is a random Gaussian noise vector with standard deviation given by:

$$\sigma_{n_s^{init}} = \left[0, 0, 0, 0, \sigma_{v_x}, \sigma_{v_y}, \sigma_{v_w}, \sigma_{v_h}\right]^T \tag{5.12}$$

The values of $\sigma_{v_x}, \sigma_{v_y}, \sigma_{v_h}, \sigma_{v_w}$ are defined by the operator.

The initial ROI state is the weighted mean of all the particles. Every particle is assigned with an initial weight of $\frac{1}{N}$.

Stage 2:

We estimate the state of each particle by the model of the movement (5.2).

$$S(k+1,i) = As(k,i) + n_s(k,i) \tag{5.13}$$

Where $n_s(k,i)$ is the state noise in frame k and particle i and is a random Gaussian noise vector with standard deviation given by:

$$\sigma_{n_s^{init}} = \left[\sigma_x, \sigma_y, \sigma_w, \sigma_h, \sigma_{v_x}, \sigma_{v_y}, \sigma_{v_w}, \sigma_{v_h}\right]^T \tag{5.14}$$

Then, we estimate the current location of each particle by:

$$z(k,i) = HS(k,i) \tag{5.15}$$

Note that when calculating the current location we do not round the results to integers, in order to avoid quantization errors that affect the next stages.

Stage 3:

We estimate the global motion (scale, horizontal shift and vertical shift) of the frame as was described in sections 3.1 to 3.5.

Stage 4:

Each particle has a different location. Therefore, we need to estimate the local motion for each particle. The estimation of local motion was described in section 3.6.

Stage 5:

We update the location of each particle according to the global motion and local motion.

Stage 6:

Now we have two estimated values of the location for each particle. The first estimated value is given by updating the location according to the state of the particle (5.15). The second estimated value is given by updating the location according to the global and local motion estimations. We can calculate the distance between the locations. Each particle defines a region by its center, width and height. Instead, the location can be defined by two points, the upper-left point and the lower-right point. We will define these points as $P_{UL}(k,i), P_{LR}(k,i)$. We have two estimated values for these points. The distance between them is given by:

$$d(k,i) = \sqrt{\|P^1_{UL}(k,i) - P^2_{UL}(k,i)\|^2 + \|P^1_{LR}(k,i) - P^2_{LR}(k,i)\|^2} \qquad (5.16)$$

Indices 1, 2 donate the two estimated values of the points.

A measure of the estimated state quality for a particle i in frame k can be given by:

$$L(k,i) = e^{-d^2} \qquad (5.17)$$

This measure is bounded in $[0,1]$, which helps the stability of the calculations.

Stage 7:

We update the weights of each particle according to the quality measure of its estimated state. We also have to normalize the weights so the sum of them is one.

$$
\begin{aligned}
w(k,i) &= \frac{w(k-1,i), L(k,i)}{\sum_{j=1}^{N} w(k-1,j), L(k,j)} \\
\sum_{i=1}^{N} w(k,i) &= 1
\end{aligned} \tag{5.18}
$$

The current weighted mean state is given by:

$$
S(k) = \sum_{i=1}^{N} w(k,i) S(k,i) \tag{5.19}
$$

The location of the ROI in the frame k is given by:

$$
z(k) = HS(k) \tag{5.20}
$$

Stage 8:

We calculate a measure for the effective number of particles. It has been shown [33] that the variance of the particles weights can only increase over time, and thus, it is impossible to avoid a degeneracy phenomenon. This degeneracy implies that a large computational effort is devoted to updating particles whose contribution to the approximation is almost zero.

$$
N_{eff} = \frac{1}{\sum_{i=1}^{N} w^2(k,i)} \tag{5.21}
$$

N_{eff} is a measure for the effective number of particles. In the case where all the particles have the same weight, $w(k,i) = \frac{1}{N}$, we have that $N_{eff} = N$ as expected. When only one particle has weight 1 and the others have weight 0 then, $N_{eff} = 1$ as expected. If the effective number of particles is smaller than a threshold, we redistribute the particles according to their weights. We used the systematic resampling proposed in [34]. First, we create a new vector Q by cumulative summation over the particles weights. We omitted the frame index k for convenience.

$$
Q(m) = \sum_{i=1}^{m} w(i), \quad m = 1, \ldots, N \tag{5.22}
$$

Also, $Q(0) = 0$.

We draw one random sample $U(1)$ by:

$$U(1) \sim \mathcal{U}\left[0, \frac{1}{N}\right] \tag{5.23}$$

Where $\mathcal{U}[a, b]$ is the uniform distribution between a and b.

We define $U(i)$ by:

$$U(i) = U(1) + \frac{i-1}{N}, \quad i = 2, \ldots, N \tag{5.24}$$

We are now associating a number of offsprings $n(i)$ with each particle i.

$$n(i) = |\{U(j) : Q(i-1) \leq U(j) \leq Q(i)\}| \tag{5.25}$$

Where $|\cdot|$ denotes the number of elements in the group.

Finally, we resample by taking $n(i)$ times the particle i, for all $i = 1, \ldots, N$. Particles with with $n(i) = 0$ are deleted, and particles with $n(i) > 1$ are duplicated. After the resampling all the new particles weights are set to $\frac{1}{N}$.

Other resampling methods can be used as well. Comparison of resampling schemes for particle filtering, can be found in [35, 36].

For the next frame, we return to stage 2.

Comparison of tracking with and without a particle filter is shown in Fig. 5.2. The y-component of ROI center before and after particle filtering is shown for the first 50 frames of a video clip, with an interference between frames 30 to 40. The true location of the ROI should not be changed during these frames. We can see that the particle filter reduced the effect of the interference. In chapter 9 we will compare experimental results using Kalman filtering and particle filtering.

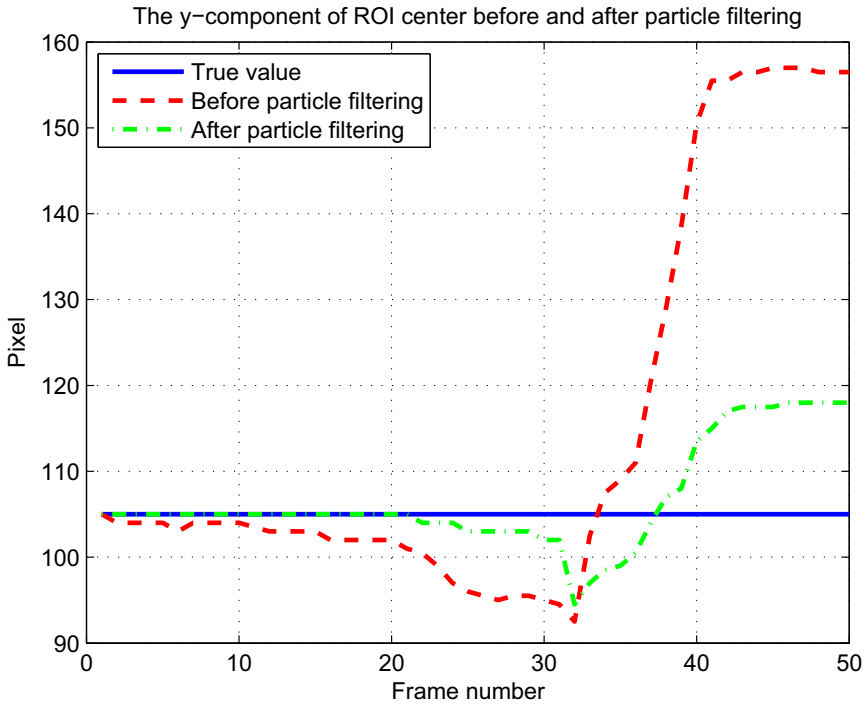


Figure 5.2: The y-component of ROI center before and after particle filtering.

Chapter 6

Region of Display Determination

The ROI is the region in the video that should be extracted for vieweing. But, we can not simply extract only the ROI for viewing becuase of two reasons. The first reason is that the speaker might move his head while talking. Tracking this movement cause the output video, which contains only the ROI, to be too jittery for the viewers' eyes.

The second reason not to extract only the ROI is that its size may be arbitrary. The output video has a pre-defined size (e.g., CIF). Extracting the ROI and scaling it to the pre-defined size will cause distortion. Therefore, we will define a region of display (ROD) as the region to be extracted. The ROD always contains the ROI. From our application requirements, we can extract only two different sizes of regions: The whole frame or a smaller pre-defined size (e.g., CIF). Therefore, the ROD can have only two possible sizes. When the ROD is determined to be the whole frame, we have to scale it to the output size. In this section we describe the process of finding the ROD location and size in each frame.

In the first frame of each scene we check the size of the ROI. If its height and width are both smaller than the pre-defined output size, with some margins (we use 30 pixels margin), we mark the ROD as a region with the pre-defined output size and locate its center at the center of the ROI. In the next frames, if the ROI is not getting close to the borders of the ROD, we do not change the

ROD location. That way, small movements of the ROI will not cause any jitter in the output display. When the ROI is getting too close to the borders of the ROD (e.g., closer than 5 pixels), we change only the ROD center location. We must have some delay since we do not want to change the ROD location according to the ROI in every frame because of its jittery motion. We track the ROI in the next several frames and calculate its new center. Only then, we move the ROD center to the new center of the ROI over the same few frames using a constant motion. The reason for changing the ROD center location over several frames is to make it look like a natural camera movement.

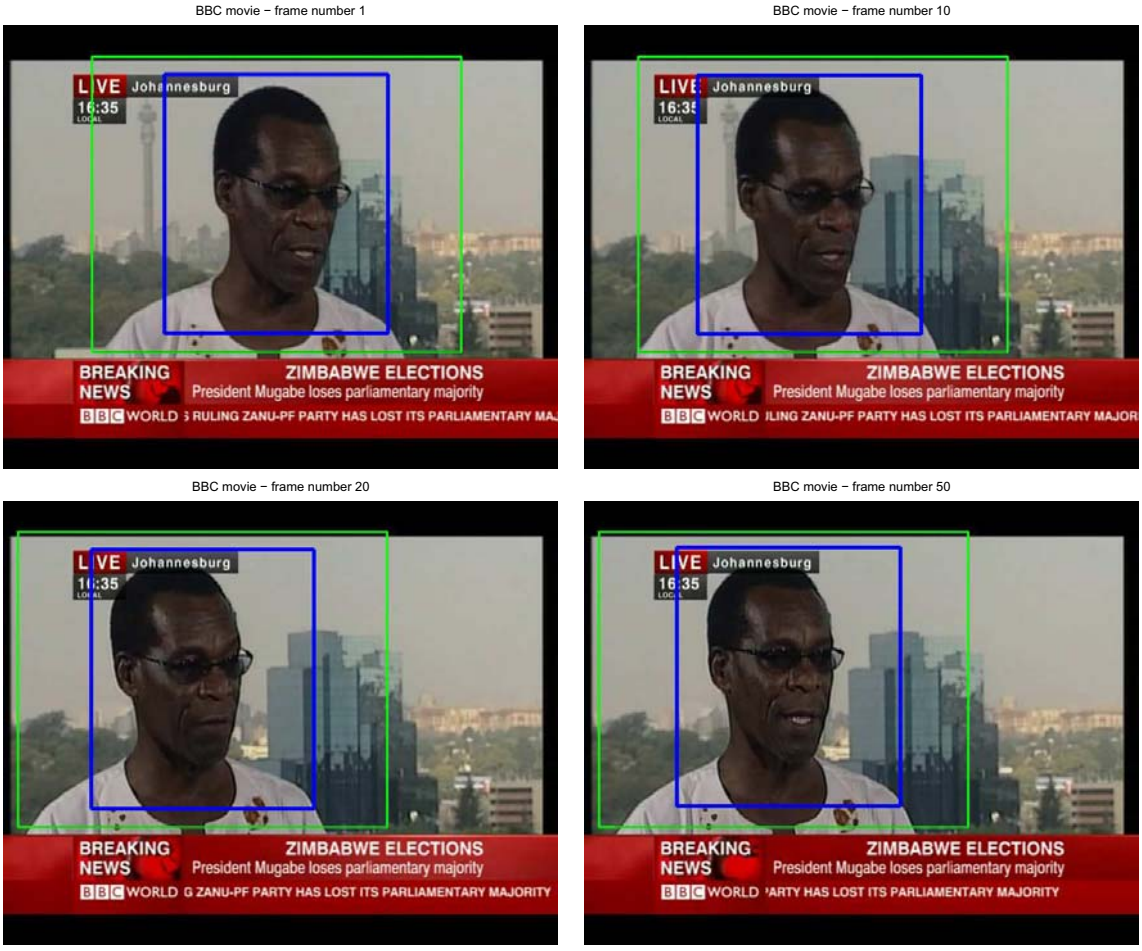


Figure 6.1: Frames from the movie "BBC" with the ROI marked as a blue rectangle and the ROD marked as a green rectangle. The ROD always contains the ROI.

An example is given in Fig. 6.1. We can see in the figure that in the first

frame of the movie the ROD has the same center as the ROI. In the next frames the man is moving his head to the left. We use here a delay of 20 frames. We can see that in frame 20 the ROD and the ROI both moved to the left and have the same center location. In frame 10, the location of the ROI is determined by the tracking algorithm. But, the location of the ROD is determined by a linear interpolation between the ROI center location in frame 1 and frame 20. Linear interpolation is needed in order to have the ROD move at a constant velocity. In frame 50, the man has moved to the right, but he didn't move much. The ROI has also moved to the right, but the ROD is in the same position as in frame 20.

The next case is when the ROI becomes bigger (due to a zoom-in) than the ROD size. We change the ROD size and mark the whole frame as a ROD. We apply the change in the size of the ROD in a single frame. Therefore, it looks like a scene cut.

An example is given in Fig. 6.2. We can see the initial ROI marked in frame 1. The ROI is bigger than the pre-defined output size and therefore the ROD is the whole frame. In frame 80 the ROI is getting smaller due to a zoom-out operation. The ROD in frame 80 is still the whole frame. In frame 81, the ROI is small enough so that the ROD changes its size to the smaller pre-defined output size. We can see that in frame 120 The ROI is even smaller, and the ROD is still of the same size.

Fig. 6.3 shows the ROD of the frames after rescaling to the desired output.

If the ROD has the size of the whole frame and the ROI is getting smaller (due to zoom-out) than the pre-defined output size, with some margins, we change the size of the ROD to the smaller size. Again, we apply the change in the size of the ROD in one frame so it looks like a scene cut. In order, not to update the ROD location too many times, we apply the hysteresis principle for all the cases mentioned above.

Figure 6.2: Frames from the movie "Euro News Zoom-Out" - with the ROI marked as a blue rectangle and the ROD marked as a green rectangle. The ROD always contains the ROI.

Figure 6.3: Frames from the movie "Euro News Zoom-Out" - output display.

Chapter 7

Active Speaker Detection

In an interview scene we may have several ROIs, one for each person in the scene. We track separately each ROI and determine its corresponding ROD. The ROI with the current active speaker is the one to be displayed. Therefore, we need to determine in each frame which is the ROI with the active speaker. An example for interview scene is shown in Fig. 7.1.

Figure 7.1: Interview scene - two regions with speakers.

7.1 Video Based Speaker Detection

7.1.1 Active Speaker Detection Based on Motion Vectors

In this section we assume we that have the motion vectors of the blocks inside each ROI. An example for motion vectors of an active speaker is shown in Fig. 7.2. We remove the effect of the global movement caused by the camera from the motion vectors. We use the global motion estimation described in the previous chapter. This way, the motion vectors correspond only to motion inside the ROI.

Figure 7.2: Motion vectors of an active speaker.

We define activity measurements for each ROI based on the motion vectors, after removing the effect of the global motion. The ROI with the highest activity measure is the current active ROI.

A simple activity measurement is the mean sum of squares of the current motion vectors:

$$\frac{1}{MN}\sum_{n,m}|\vec{v}_{n,m}|^2 \tag{7.1}$$

where N, M are the height and width of the region.

An example for this activity measure is shown in Fig. 7.3. Only the person in Region 1 was speaking during the whole scene.

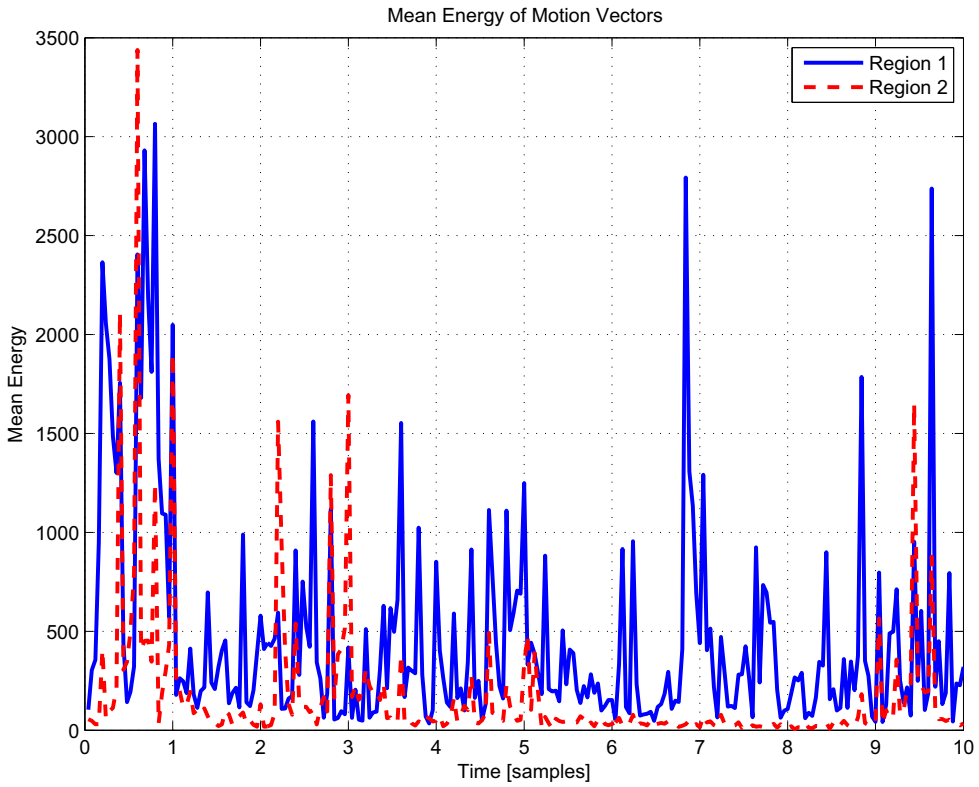


Figure 7.3: Mean energy of motion vectors.

It is clear that this algorithm will not work properly if the non-active speaker will not be still. Non-active speaker can move, for example, nod his head or smile. Therefore, algorithms based on motion only are not robust enough.

7.1.2 Active Speaker Detection Based on Pixels Intensities

In this section we assume that the motion vectors are not available since computing them has too high complexity. When opening the mouth, the number of pixels with low intensities is increasing [8]. The number of pixels with low intensities in the mouth area of each ROI in each frame is counted as suggested in [8]. The active region is the one with the highest variance. An

example is shown in Fig. 7.4.

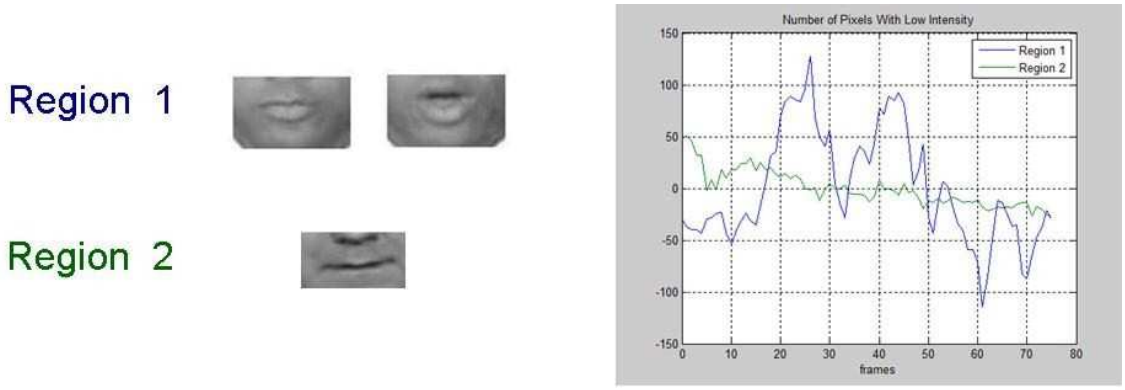


Figure 7.4: Number of pixels with low intensities.

This algorithm requires detection of the mouth location. The location of the mouth is set to be the lower half of the ROI in each frame.

The algorithm will not work properly if the speaker reads with his face down, when his mouth can not be fully seen in the frame.

7.2 Active Speaker Detection Aided by Audio Track Information

Correlation between events in the audio-track and in the video can be used to detect the pixels that are producing the sound [11]. For speech, the onsets times might be correlated with a high acceleration of the speaker lips [12]. In the next subsections we will describe an algorithm for onset detection and an algorithm for tracking pixels in video. Then we will describe how to use the correlation between onsets locations and features from the trajectories of the tracked pixles, in order to find the pixels that are related to the audio. Determing the pixels that correspond to the audio will help us determine the current active speaker.

7.2.1 Onsets Detection

In phonetics, onset refers to the beginning of a syllable, at which the amplitude rises from zero to an initial peak. There are many algorithms to detect the

onsets locations, e.g. [13, 14, 12]. In [12], several musical instruments can be playing different tunes simultaneously. Therefore, they suggested to detect new frequencies using the spectogram.

In our application we do not expect speakers to talk simultaneously. Therefore, a simple energy detector should be enough to detect the onsets location. We calculate the energy using a sliding window. The window we used is a Hamming window and the length of the window was set to be 0.1 second. We search for consecutive increases in the energy. When we find such consecutive increases, we check the energy before the first increase. If it is lower than a certain thrsehold, meaning that the increase occured after silence, we mark it as a candidate for an onset. Our goal is to detect instants of audio events to be correlated with the instants of video events. We define the time location of an audio event to be the time between the onset and the end of the rise in energy. After having a candidate for an onset, we search for the first decrease in the energy. The first decrease is marked as a candidate for an audio event end. We calculate the difference between the energy in the candidate location for end of the event and the energy in the candidate location of the onset. This difference is the total increase in energy during the audio event. If the total increase is larger than a certain thrsehold, then we declare it to be an audio event. Using this method we also have the start and end locations of the audio events. We noted during experiments that the consecutive increases in the energy can start before the sharp increase during the real location on the onset. Therefore, we look for the time location where 20% of the total increase in energy is obtained and mark this place to be the onset location. An example for an audio event is given in Fig. 7.5.

In order to use the same thrseholds for different audio volume, we need an automatic gain control (AGC) before calculating the energies. We use a simple agc. We divide the soundtrack into sections and the agc rescales the enregy in every section to a pre-defined value. The rescale in not perfomed on section which have initial very low energy, since it means that the section contain only silence and noise.

An exaple for audio events detction is given in Fig. 7.6.

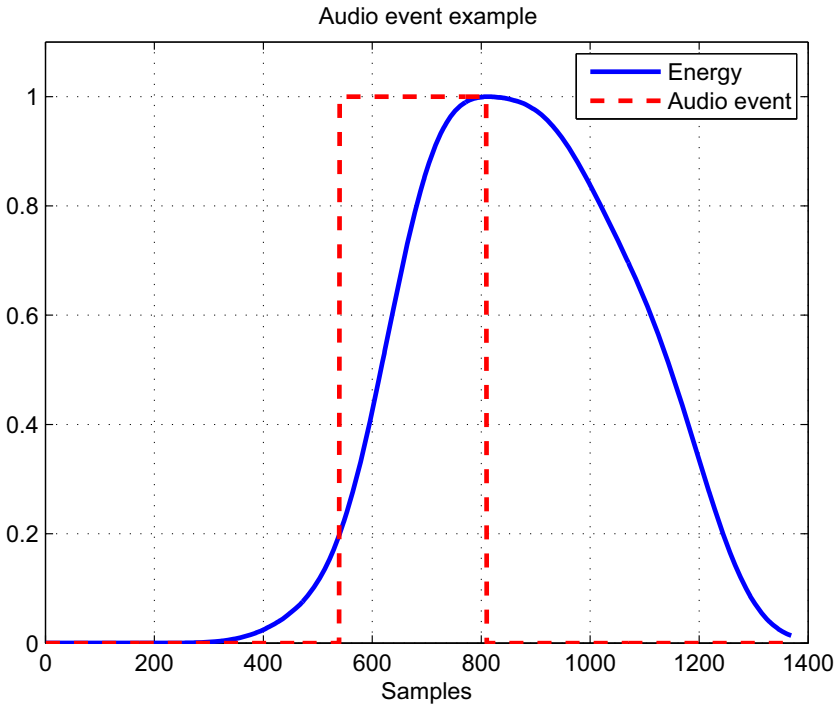


Figure 7.5: Example for an audio event.

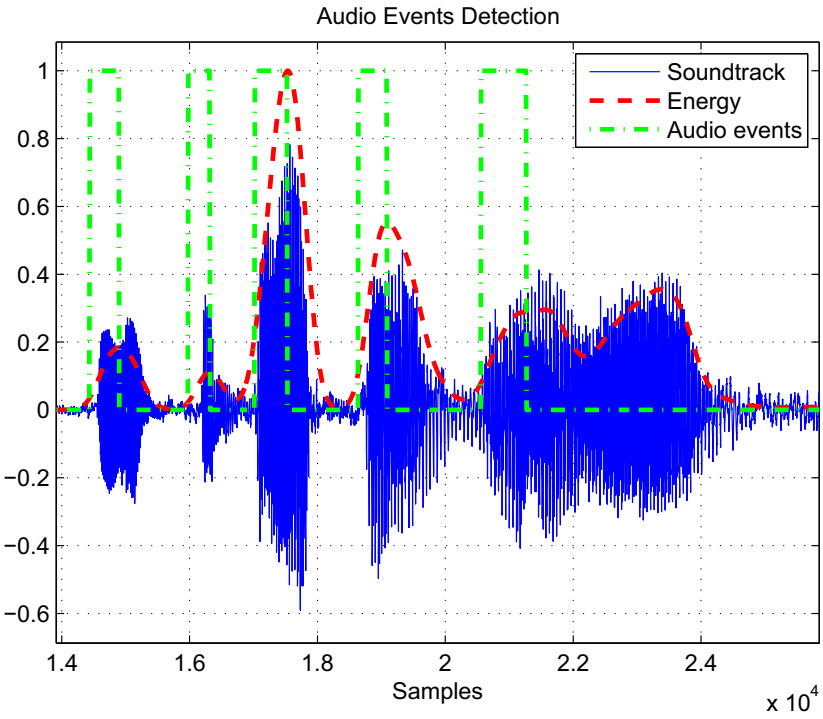


Figure 7.6: Example for detecting audio events locations.

7.2.2 Tracking of pixels in video

In this section we desccribe an algorithm for tracking pixels in video. We choose to work with a tracker known as the Kanade-Lucas-Tomasi (KLT) tracker [15], implemented by [17], as was suggested in [12]. The description of the tracker in this section is based on [15, 16, 17, 12].

The motion model for the KLT-tracker assumes displacement only.

$$I(x, y, t + \tau) = I(x - d_x, y - d_y, t) \tag{7.2}$$

Where I(x,y,t) is the value of pixel (x, y) in time t, and $d = (d_x, d_y)$ is defined as the displacement.

An important problem in finding the displacement d of a point from one frame to the next is that a single pixel cannot be tracked, unless it has a very distinctive brightness level with respect to all of its neighbors. Because of this problem, the KLT-tracker does not track single pixels, but blocks of pixels. We do not use a more complex model for small windows, such as affine motion. This is becuase of the danger of over-parametrization. We therefore only estimate the displcement vector d.

Define $J(\mathbf{x}) = I(x, y, t + \tau)$ and $I(\mathbf{x} - d) = I(x - d_x, y - d_y, t)$. Our local image model is given by;

$$J(\mathbf{x}) = I(\mathbf{x} - d) \tag{7.3}$$

The displacement vector d is then chosen so as to minimize the residue error defined by the following double integral over the given window $\mathcal{W}$:

$$\epsilon = \int_{\mathcal{W}} \left[I(\mathbf{x} - d) - J(\mathbf{x})\right]^2 w(\mathbf{x}) d\mathbf{x} \tag{7.4}$$

Where $w(\mathbf{x})$ is a weighting function, which can be Gaussian-like, to emphasize the central area of the window.

When the displacement d is much smaller than the window size, the linearization method can be used. The intensity function can be approximated by its Taylor series truncated to the linear term.

$$I(\mathbf{x} - d) = I(\mathbf{x}) - g^T d \tag{7.5}$$

where g denotes the gradient vector of I at $\mathbf{x}$.

Now we can write the error function from equation (7.4) as:

$$\epsilon = \int_{\mathcal{W}} \left[I(\mathbf{x}) - g^T d - J(\mathbf{x}) \right]^2 w(\mathbf{x}) d\mathbf{x} \tag{7.6}$$

This error function is a quadratic function of the displacement d. As a consequence, the minimization can be done in closed form. Differentiating the last expression of the error function given in equation (7.6) with respect to d and setting the result equal to zero yields the following vector equation:

$$\int_{\mathcal{W}} \left[I(\mathbf{x}) - g^T d - J(\mathbf{x}) \right] g w(\mathbf{x}) d\mathbf{x} = 0 \tag{7.7}$$

Since $(g \cdot d)g = (gg^T)d$ and d is assumed to be constant within $\mathcal{W}$, we have:

$$\left(\int_{\mathcal{W}} gg^T w(\mathbf{x}) dx \right) d = \int_{\mathcal{W}} \left[I(\mathbf{x}) - J(\mathbf{x}) \right] g w(\mathbf{x}) dx \tag{7.8}$$

This is a system of two scalar equations in two unknowns, which are the components of the displacement vector d. It can be rewritten with a symmetric 2x2 coefficient matrix G and a two-dimensional vector d.

$$\begin{aligned} Gd &= e \\ G &= \int_{\mathcal{W}} gg^T w(\mathbf{x}) dx \\ e &= \int_{\mathcal{W}} \left(I(\mathbf{x}) - J(\mathbf{x}) \right) g w(\mathbf{x}) dx \end{aligned} \tag{7.9}$$

Equation (7.9) is the basic step of the tracking procedure. For every pair of adjacent frames, the matrix G can be computed from one frame, by estimating gradients and computing their second order moments. The vector e, on the other hand, can be computed from the difference between the two frames, along with the gradient computed above. The displacement d is then the solution of system (7.9).

We also use an algorithm for automatic location of good points to track. This is done by recognizing that a good point is one that can be tracked well. The tracking equation is defined in equation(7.9). Consequently, a window $\mathcal{W}$ is chosen if its corresponding matrix G is well-conditioned. This implies

that the window can be tracked reliably [15].

7.2.3 Audio-Visual Correlation

In this section we will use the audio events locations and the trajectories of the tracked points in order to detect the current active speaker. We assume that we have several regions that we track during the scene and we have to determined in which of them is the current active speaker.

We use the software in [17]. At the beginning of a scene, we select points to track, as was decsribed in section 7.2.2. An example of good points to track is shown in Fig. 7.7.

Figure 7.7: Good points to track.

We track these points for the rest of the scene. We also calculate the audio events locations as was described in section 7.2.1. In [12] it was proposed to calculate the correlation between the audio events locations and the times of a significant acceleration of the visual points. The use of the acceleration is becuase we are interested in locating instances of significant temporal variation in the motion of a visual point.

In our work we added two modifications. First, we use the global motion estimation described in chapter 3 to eliminate the effect of the camera motion from tracked points. For every point $p_1 = (x_1, y_1)$ in the current frame, we use

the global motion prameters to calculate the estimated location of the point in the next frame $\hat{p}_2 = (\hat{x}_2, \hat{y}_2)$.

$$\hat{p}_2 = Ap_1 + b \tag{7.10}$$

Where

$$A = \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix} \tag{7.11}$$

α is the estimated global scale and b is a two element vector with the estimated global horizontal and vertical shifts.

Then we can use the location of the point in the next frame found by the KLT-tracker, $p_2 = (x_2, y_2)$ and subtract the estimated point location $\hat{p}_2$. The difference between the points coordinates is the change in location of the tracked point, without the effect of the camera motion.

The second modification is that we calculated the acceleration only in the vertical axis. This is becuase we are looking for a point on the speaker face that is correlated to his voice, and the mouth movements are mostly in the vertical axis. After calculating the acceleration in the vertical axis for each tracked point, we use a thrsehold to detect times of significant changes.

The audio events are given in a vector $AudioEvents(i)$ where i is the soundtrack samples index. At the audio events locations the vector is equal to 1, and otherwise it is 0. Similarly, for every tracked point we have a vector of acceleration onsets or visual events $VideoEvents(k)$ where k is the frame number. We use linear interpolation to obtain $VideoEvents(i)$ where i is the soundtrack samples index. Then we calculate the correlation coefficient between the vector of audio events and the vectors of acceleration onsets. Note that these two vectors are binary vectors. We denote their standard deviations as $\sigma_{VideoEvents}$ and $\sigma_{AudioEvents}$.

$$\rho = \frac{cov(VideoEvents, AudioEvents)}{\sigma_{VideoEvents}\sigma_{AudioEvents}}, \tag{7.12}$$

Where *cov* means covariance.

The tracked point with the highest correlation is declared to belong to the current active speaker. For our application, we need this algorithm only for

cases where there are multiple regions that can be the current ROI. We track visual points only inside these regions. The region with the point that achieve the highest correlation is declared as the current active ROI. In Fig. 7.8 we mark in red the visual point that was the most correlated with the audio onsets. We can see that the point is indeed on the speaker's face, near his mouth.

Figure 7.8: Best visual point associated to the audio.

Chapter 8

ROI Determination from a Single Click on a face

In the previous chapters we assumed that at the start of every scene, the ROI is marked by an operator. In this chapter we develop an auxiliary tool for the operator. Earlier it was assumed that the operator draws a bounding box defining the ROI, requiring at least two clicks on the image. Since our work focuses on news or interviews scenes, we assume that in most of the scenes the ROI is bounding the speaker's face. Here we propose an algorithm that automatically finds the ROI by indicating only one point inside the face, therefore requiring only one click. The algorithm is based on a combination of skin-color detection and region growing.

8.1 Skin Color Detection

It was verified, using training data, that skin-colors are clustered in color space [10]. It is common to think that different people have skin-colors which differ significantly from each other due to the existence of different races. However, what really occurs is a larger difference in brightness / intensity and not in color. An example is shown in Fig. 8.1 (taken from [10]). The skin-color pixels were obtained from a color face database containing images of people of different races, ages and gender [40].

However, using only skin color as a face detection algorithm is not good

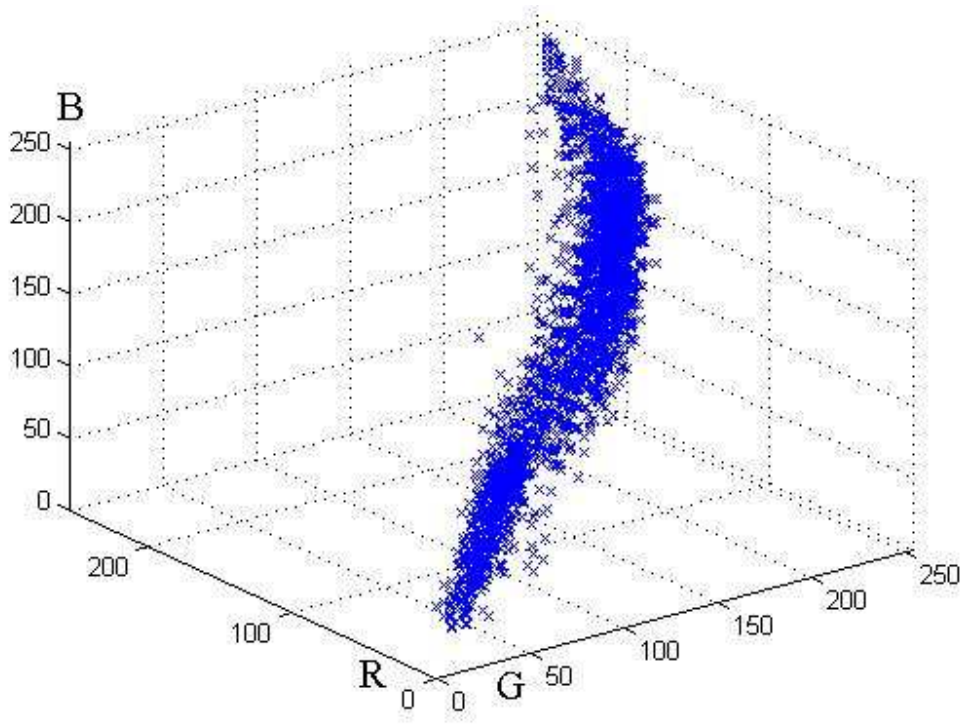


Figure 8.1: Skin color cluster in RGB space.

enough. We can see in Fig. 8.2 that the skin-color detector detects many wrong pixels as skin. For example a shelf behind the man has a color that is identified as skin color. Therefore, morphological operations and connectivity analysis are needed to detect faces.

We use the skin-color detector that was proposed in [27] for the YCbCr color space. A pixel is labeled as a skin-candidate if it falls within the region where $Skin(Cb, Cr) = 1$, which is defined by:

$$Skin(Cb, Cr) = \begin{cases} 1, & (Cb < \Gamma_{up}(Cr)) \cdot (Cb > \Gamma_{bottom}(Cr)) \\ 0, & otherwise \end{cases} \tag{8.1}$$

Where

$$\Gamma_{up}(x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0 \tag{8.2}$$

$$\Gamma_{bottom}(x) = \beta_2 x^2 + \beta_1 x + \beta_0 \tag{8.3}$$

(a) Frame (b) Skin Pixels

Figure 8.2: Skin pixels detection.

For the upper bound, the quadratic coefficients are $\alpha_2 = 0.0225$, $\alpha_1 = 6.1251$ and $\alpha_0 = 290$ while the lower bound coefficients are $\beta_2 = 0.0284$, $\beta_1 = 9.1477$ and $\beta_0 = 836$. These functions are shown in Fig. 8.3.

8.2 Face Segmentation using Region Growing

Region growing is an algorithm for detecting a homogenous region which has similar intensity values, starting with any point inside the region [41]. Starting with a point inside the face region, as indicated by the operator, the region is iteratively grown by comparing all unallocated neighboring pixels to the region. The difference between a pixel's intensity value and the region's mean is used as a measure of similarity. The pixel with the smallest difference is allocated to the respective region if the difference is smaller than a certain threshold. This process stops when the intensity difference between the region mean and any of the unallocated neighboring pixels is larger than the threshold.

We applied the region growing algorithm on the skin-color detector output. We also applied the region growing algorithm on the original frame with the same starting point, using only the intensities. An example is shown in Fig. 8.4. We can see that non of the outputs are good enough to locate the speaker's face. But if we combine the outputs by using a logic AND operation, we

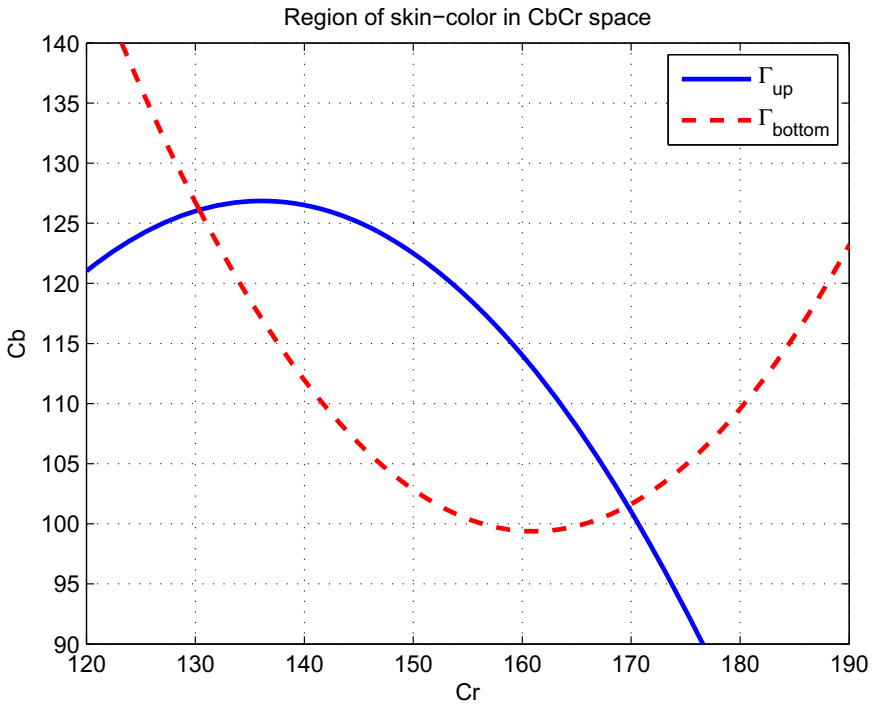


Figure 8.3: Skin-color detector.

get a better estimation of the face location. The final result after the AND operation is shown in Fig. 8.5. The face region is then defined by the bounding rectangle of the region.

(a) (b)

Figure 8.4: Region growing example. (a) Region growing on the skin-color detector output. (b) Region growing on the frame.

(a) (b)

Figure 8.5: Region growing result. (a) Combination of region growing on the skin-color detector output and region growing on the frame using an AND operation. (b) Face region is marked on the frame.

Chapter 9

Experimental Results

In this chapter we present test results of the algorithms developed in this work using simulation on real TV video clips as well as simulated ones.

9.1 Video Samples

We tested the algorithms on 3 short video sequences. In the first video clip the camera is static, but the talking man is moving his head. In the second clip, the camera is zooming-in while moving to the left. In the third clip, the camera is zooming-out while moving to the right. The first and last frames from each video clip is shown in Figs. 9.1, 9.2 and 9.3.

(a) (b)

Figure 9.1: Frames from "BBC" video clip (a) First frame (b) Last frame

(a) (b)

Figure 9.2: Frames from zoom-in video clip (a) First frame (b) Last frame

(a) (b)

Figure 9.3: Frames from zoom-out video clip (a) First frame (b) Last frame

9.2 Tracking Results

In order to test the tracking algorithm, and mainly the global motion estimation we first created a video sequence with known scale and shift values. We did it by taking the first frame from a real TV interview and created the rest of the frames by transforming it with the known parameters. It is important to note that we perform the transformation on the first frame, and not as a

cascade of transformations. This is because a cascade of transformations will cause distortion. We used a zoom factor of $\alpha = 0.99$ and horizontal and vertical shifts of $d_x = -3$ $d_y = 2$. We marked an ROI and tracked it using only the global motion estimation. The ROI is marked in blue in the following figures. We can see the initial ROI mark in Fig. 9.4, and in Fig. 9.5 the ROI in frame 40. The tracking provides a good result since the ROI contains exactly the same objects in both frame 1 and 30, despite the simulated camera movement.

Figure 9.4: Frame 1 from a movie with constant scale change between frames.

We created another movie with known parameters. In this movie the scale was changing over time. We wanted to test the use of scale and shift parameters from previous frame to reduce finite-size display effects, as was described in section 3.3, even when they are not accurate. The initial scale factor is $\alpha = 0.99$ and in frame 40 the final scale factor is $\alpha = 0.98$. Note that when α is decreasing the zoom-in factor $\frac{1}{\alpha}$ is increasing. We can see in

Figure 9.5: Frame 40 from a movie with constant scale change $\alpha = 0.99$ between frames.

Fig. 9.6 that the tracking algorithm provides a good result.

Figure 9.6: Frame 40 from a movie with an increasing zoom-in factor.

Next, we tested our algorithm on a real TV interview. We marked the ROI as the head of the interviewee. In the first scene, the camera zoomed-in and also moved to the right and up. The first frame is shown in Fig. 9.4. The last frame in the scene, frame 300, is shown in Fig. 9.7.

In the second scene, the camera zoomed-out and also moved to the left and down. The first frame is shown in Fig. 9.8 and the last frame from the scene, frame 159, is shown in Fig. 9.9. We can see that the algorithm tracked the head till the end of the scene.

In the next example, the camera was still, but the talking head moved during the scene. We can see in Figs. 9.10 and 9.11 that the tracking worked well and that the head is in the middle of the ROI in both the first frame and the last frame, frame 100. We expected the tracking algorithm to detect no camera movement when estimating the global motion parameters. Then, we

Figure 9.7: Last frame from a scene with a zoom-in.

Figure 9.8: First frame from a scene with zoom-out.

Figure 9.9: Last frame from a scene with zoom-out.

expect it to detect local motion (the head motion) to the left. In this case, our estimation erroneously detected the head motion as a camera shift to the left. This is because the head is a big object regarding to the frame size. Therefore, the head movement to the left looks as moving the camera to the left. After we updated the ROI using the global motion parameters, we calculated the local motion. In this case no local motion was detected because we already moved the ROI to the left. Therefore, despite the erroneous movement due to global motion estimation, to the left, the final tracking result is good.

Figure 9.10: First frame from a scene with talking-head moving to the left.

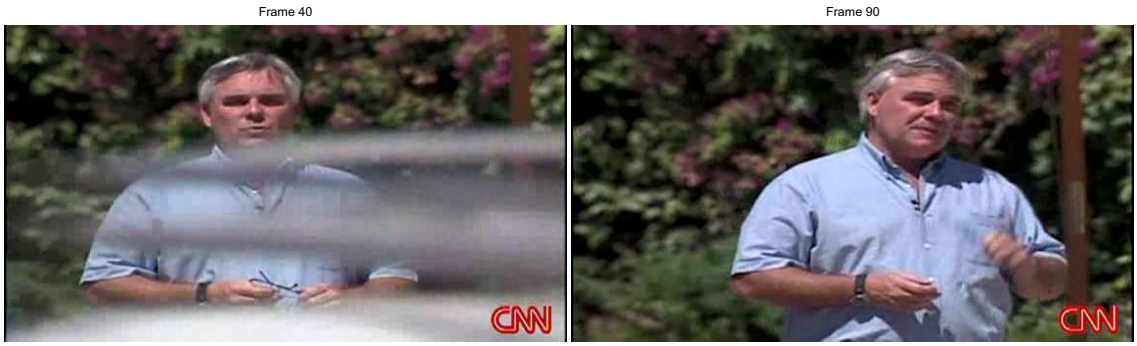
9.3 Kalman and particle filtering

In this section we test the tracking using a Kalman filter and particle filtering. In the video scene we chose, the reporter is standing across a road. Passing cars are almost hiding him for several frames. We chose this video becuase it has an interference, so that in some of the frames, the global motion estimation

Figure 9.11: Last frame from a scene with talking-head moving to the left.

is expected to be wrong. Assuming a dynamic model and estimating its state should improve the tracking, as was discussed in 5. Three frames from the video are shown in Fig. 9.12. In the first frame the reporter is standing and talking to the camera. In frame 40, a passing car is almost hiding the reporter. In the last frame the reporter is moving to the right.

(a)

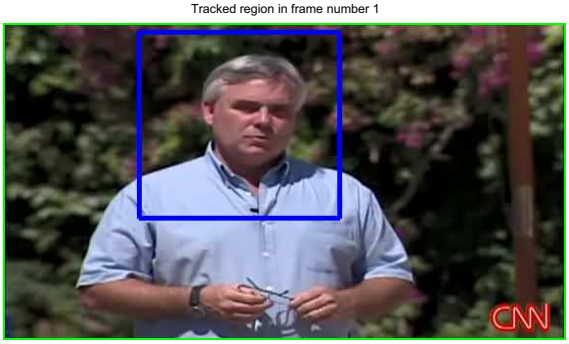


(b) (c)

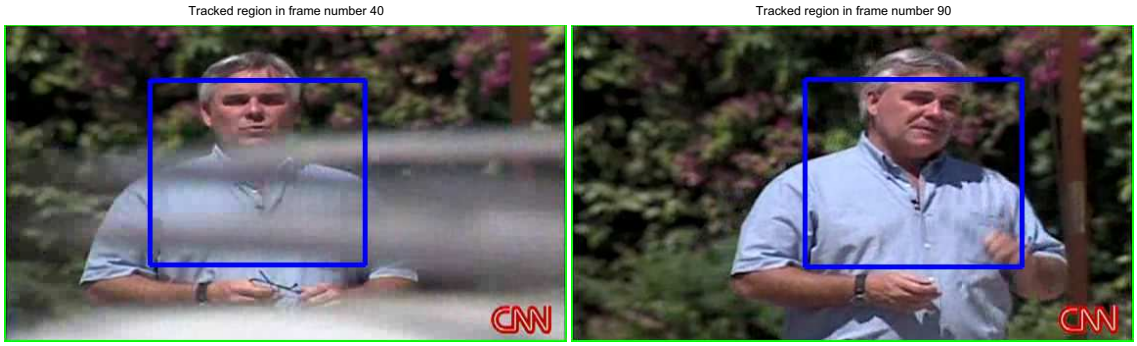
Figure 9.12: Frames 1, 40 and 90 from CNN mideast movie.

The results of the tracking algorithm, without Kalman or particle filtering, is shown in Fig. 9.13. We can see that when the car passed, the tracked region moved down erroneously. Since the motion in the frame when the car is passing doesn't match the global motion model assumed, the movement of the tracked region is erratic.

The results of the tracking algorithm with a Kalman filter is shown in Fig. 9.14. We can see that the region still moved down, but less than without the Kalman filter. This is becuase the state of the tracked region is estimated to be motionless before the passing car. When the car passes, the global



(a)



(b) (c)

Figure 9.13: Tracked region in CNN mideast movie in frames 1, 40 and 90.

motion that is detected is downward. The state is updated slowly, considering also the previous measurements, and therefore the movement is smaller. We can change the parameters of the Kalman filter so that the state updating process will be even slower. But, in that case, the Kalman filter won't be able to track the reporter when he moves to the right. We couldn't find parameters that satisfy the trade-off between the fast update of the state needed when the reporter moves and the slow uopdate of the state needed when the car passes by.

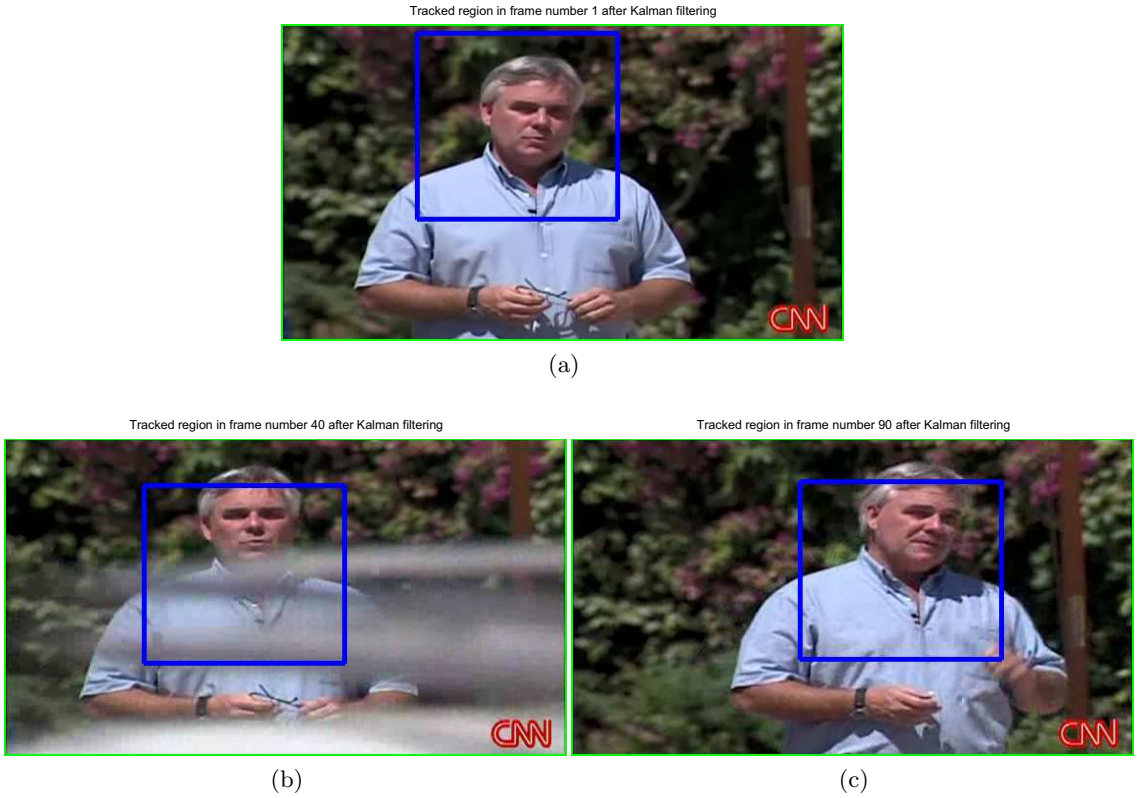


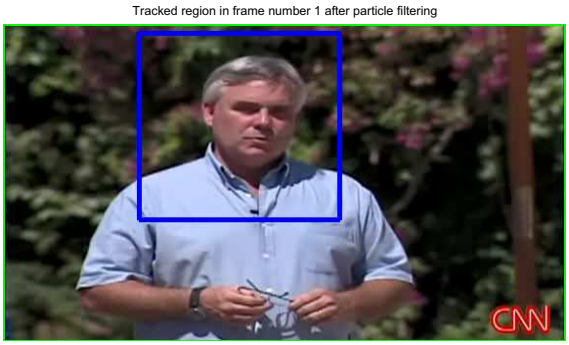
Figure 9.14: Tracked region in CNN mideast movie in frames 1, 40 and 90 after applying a Kalman filter.

The results of the tracked region after particle filtering is shown in Fig. 9.15. We can see that the tracking results are much better. The particle filter does not assume that the posterior probability density function of the state is a Gaussian distribution. In this case the distribution is a bimodal distribution. One peak is a state with location around the reporter's face and no velocity.

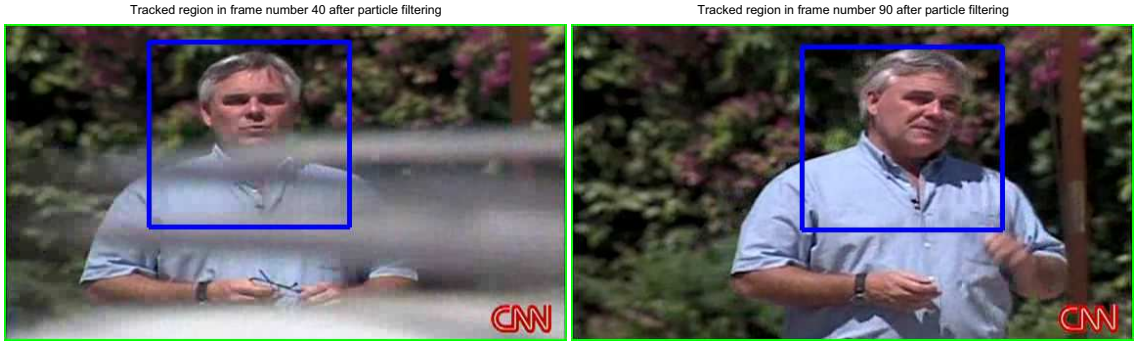
The other peak is a state that its location is moving downward with high velocity, becuase of the wrong estimated values. The use of the particle filtering enbales to better describe than the Kalman filter this posterior probability density function.

The Kalman filter state, when the car passes, changes to a state with a velocity downwards, becuase of the wrong estimated motion parameters. After the car passes, the estimated motion parameters are close to zero. The Kalman filter relates to this estimated parameters as errors since they are very different from its state. It takes a few frames till the state update to a state with no velocity.

The particles with the largest weight, at the start of the scene, are the particles with locations around the face and with no motion. When the car passes, the weights of these particles decrease. The weights of particles with high velocity downwrads increase. The output is the weighted average of these particles' states. The particles represent the posterior probability density function of the dynamic model state, and thus this weighted average is an estimation of the expectation value of the state. After the car passes the weights of particles that represent a state with no motion increase again. Therefore, as can be seen from the example, the particle filter estimates well the posterior probability density function of the state.



(a)



(b) (c)

Figure 9.15: Tracked region in CNN mideast movie in frames 1, 40 and 90 after particle filtering.

Chapter 10

Conclusion

10.1 Summary

The goal of this work is to perform an adjustment of video in SD (standard definition) resolution to the resolution used in cellular phones or other mobile devices. The naive solution requires the rescaling of each frame to the desired output size. However, this solution will make it hard to recognize important small objects. For example, faces should be shown in a sufficient size. Extracting only the region of interest (ROI) for the output display solves this problem. We focus in this work on news broadcasting and interview scenes. Our main contributions in this work are summarized in this section.

We developed a novel algorithm for tracking the ROI. We assumed a model of three parameters for the global motion casued by the camera movements, representing zoom, tilt and pan. This simple model is sufficient for news broadcasting or interview scenes, where the speakers are not approaching the camera. We showed that two-dimensional Fourier transform of the whole frames can be used for estimating the model parameters. We showed that only slices of the two-dimensional Fourier transform are enough for the estimation, enabling us to use the slice-projection theorem and thus reduce computational complexity. We then developed an efficient algorithm based on only on the horizontal and vertical projections. We developed another method for scale estimation using the Analytical Fourier-Mellin Transform (AFMT). We then reduced computational complexity by applying the one-dimensional Mellin

transform to slices in the freuqency domain, instead of applying the AFMT to the whole frames. Assuming that the changes in the motion parameters of consecutives frames are small, we used the estimated parameters from the previous frame as an initial estimation for the current frame. Then, we further reduced the computational complexity estimating the parameters by using the Mellin transform in the spatial-domain. The Mellin transform is used to convert the scale factor into a multiplicative factor, which can be estimated more efficiently.

The result of the tracking algorithm is jittery usually due to estimation errors caused by noise and interference. We reduced the jitter by using a Kalman filter. For more complicated cases including interferences (e.g., occlusion), we use a particle filter, obtaining also a more robust tracking. We showed that the particle filter improve estimating the dynamic model state of the ROI.

The ROI should not be extracted as is for vieweing for two reasons. The first reason is that the speaker might move his head while talking. Tracking this movement cause the output video to be jittery to the viewers' eyes. The second reason is that its size may be arbitrary and scaling it will cause distortion. Therfore, we define a region of display (ROD) as the region to be extracted. The ROD always contains the ROI and its location changes smoothly resulting in a stable video required for broadcasting quality.

For interview scenes, several regions can be marked as regions of interest in the first frame of every scene. Only one region should be sent to the output display. We proposed to make a decision which region to send, by finding the region with the current active speaker. Finding the current active speaker is done by finding the correlation between times of significant changes in the acceleration of visual points and times of audio onsets. We showed that the visual points with the highest correlation to the onsets in the soundtrack are likely to be on the speaker's lips area. Therefore, we can make a decision that the region with the current active speaker is the region with the visual point with the highest correlation to the audio onsets. We showed that a simple audio onsets detector based on energy increase is sufficient for that purpose.

We proposed a method for detecting the ROI in a semi-automatic manner by a single click of an operator on the speaker's face. We used a combination

of a skin-color detector and region growing to determine the bounding box around the speaker's face.

10.2 Future Directions

We suggest several issues for future research.

The first issue involves motion estimation. In our work we assumed a three parameter model for describing the global motion. The parameters are the zoom factor, tilt and pan. This model is sufficient to describe well a news scene or an interview scene. More complicated models, such as the affine or the perspective models, can describe more types of content. Also, in our work we assumed that the main object in the ROI (e.g., face) has a local motion that can be described by horizontal and vertical shifts only. This model is sufficient for cases where the object is in the same distance from the camera for the whole scene. In the case where an object is approaching the camera or is moving away from the camera, there is also local scale change in the ROI. A more complicated model is needed to describe that motion well. Introducing a model with more parameters than in our work will enable the tracking to work well for more complex scene types, but probably will increase the complexity of estimating the parameters. In our work we reduced the complexity by using the projection-slice theorem and the Mellin transform. Similar methods for reduced complexity parameters estimation should be developed for the more complicated models.

The second issue concerns the active speaker detection. In our work we proposed to use the correlation between audio onsets and significant changes in the acceleration of visual points. We showed that visual points on the speaker's lips area have good correlation with the soundtrack. For tracking the visual points we used the well known KLT-tracker. An improvement to our work can be made by introducing a model for the movement of the speaker's lips. Using such a model will lead to a more robust tracking of points on the speaker's lips. Also, in our work, we use only the time locations of audio onsets and the time locations of significant chages in acceleration of the visual points. We did not use the amount of change in the audio energy or in the acceleration. Taking these amounts into consideration when performing the

correlation might improve the active speaker detection.

The third issue concerns tracking a Region of ADvertisments (ROAD). A ROAD should be a region of 'no-interest', i.e., not containing motion, faces, etc. The tracking algorithms developed in this work can be also used to track such a region.

Another extension is the development of a transcoder that uses the information generated by our work. The location of the ROI in each frame is added as a metadata to the compressed video stream in the edtiting phase. Only when needed for transmission, the ROI is extracted and scaled to match the desired output display, and then it is compressed. The naive solution is to decode the compressed video stream, extract the ROI, rescale it and compress it again. Compression of video have high complexity. An extension for our work is the development of a transcoding scheme for the extraction, rescaling and compressing. A transcoder may re-use information from the original compressed video stream and thus reduce computational complexity. For example, the motion vectors inside the ROI from the original compressed stream may be reused when compressing only the extracted ROI for transmission.

Appendix A

Scale Estimation Using AFMT

The AFMT is defined in equation 3.10. It is written here again for clarity.

$$M_f(k,v) = \frac{1}{2\pi}\int_0^{\infty}\int_0^{2\pi} f(r,\theta)\, r^{\sigma-jv} e^{-jk\theta} d\theta \frac{dr}{r} \tag{A.1}$$

Where σ is a parameter.

The relation between frames related by scale and rotation is given in (3.11). In [6] the AFMT was used to estimate this model parameters. Our model is different since we have scale and shifts parameters. The relation between frames in our model is given in (3.1). Therefore, we change to Cartesian coordinates.

$$\begin{aligned}
M_f(k,v) &= \frac{1}{2\pi}\int_0^{\infty}\int_0^{2\pi} f(r,\theta)\, r^{\sigma-jv} e^{-jk\theta} d\theta \frac{dr}{r} \\
&= \frac{1}{2\pi}\int_0^{\infty}\int_0^{2\pi} f(r,\theta)\, r^{\sigma-jv} r^{k} (rcos\theta + jrsin\theta)^{-k} d\theta \frac{dr}{r} \\
&= \frac{1}{2\pi}\int_0^{\infty}\int_0^{2\pi} f(r,\theta)\, r^{k-1+\sigma-jv} (rcos\theta + jrsin\theta)^{-k} d\theta dr \\
&= \frac{1}{2\pi}\int_0^{\infty}\int_0^{2\pi} f(p,q) \left(p^2+q^2\right)^{\frac{k-1+\sigma-jv}{2}} (p+jq)^{-k} \frac{dpdq}{(p^2+q^2)^{0.5}} \\
&= \frac{1}{2\pi}\int_0^{\infty}\int_0^{2\pi} f(p,q) \left(p^2+q^2\right)^{\frac{k-2+\sigma-jv}{2}} (p+jq)^{-k} dpdq
\end{aligned} \tag{A.2}$$

Where

$$\begin{aligned} p &= rcos\theta \\ q &= rsin\theta \end{aligned} \tag{A.3}$$

Note, that we divided the integrand by the Jacobean r when changing to Cartesian coordinates, in the fourth equality.

In discrete form [6]:

$$M_f^d(k,v) = \frac{1}{2\pi} \sum_{q=Q_{min}}^{q=Q_{max}} \sum_{p=P_{min}}^{p=P_{max}} f(p,q) \left(p^2+q^2\right)^{\frac{k-2+\sigma-jv}{2}} (p+jq)^{-k} \tag{A.4}$$

Suppose we have two frames f_1, f_2 with a relation by the model in (3.1). The relation between their Fourier transforms is given by equation (3.2). Taking the absolute values of the Fourier transforms, we get:

$$|F_2(f_x,f_y)| = \frac{1}{\alpha^2} \left| F_2\left(\frac{f_x}{\alpha}, \frac{f_y}{\alpha}\right)\right| \tag{A.5}$$

Performing an AFMT on the absolute values of the Fourier transforms, we get M_{f_1}, M_{f_2}. The scale estimation is given by $\alpha = e^{\frac{t}{2-\sigma}}$, where t is define by:

$$\begin{aligned} t &= \frac{\sum_{v=-V}^{v=V}\sum_{k=-K}^{k=K}\left(ln|M_{f_1}^d(k,v)| - ln|M_{f_2}^d(k,v)|\right)}{(2K+1)(2V+1)} \\ &= \frac{\sum_{v=-V}^{v=V}\sum_{k=-K}^{k=K}\left(ln|M_{f_1}^d(k,v)| - ln|\alpha^{-2}\alpha^{\sigma-jv}e^{-jk\beta}M_{f_1}^d(k,v)|\right)}{(2K+1)(2V+1)} \\ &= \frac{\sum_{v=-V}^{v=V}\sum_{k=-K}^{k=K}\left(ln|M_{f_1}^d(k,v)| - ln|M_{f_1}^d(k,v)| - ln|\alpha^{-2+\sigma}|\right)}{(2K+1)(2V+1)} \\ &= \frac{\sum_{v=-V}^{v=V}\sum_{k=-K}^{k=K}\left(-ln|\alpha^{-2+\sigma}|\right)}{(2K+1)(2V+1)} = (2-\sigma)\,ln(\alpha) \end{aligned} \tag{A.6}$$

Appendix B

Computational Complexity of Scale Estimation Methods

In this section we calculate the complexity of the different scale estimation methods, the frequency-domain slice cross-correlaton method described in section 3.1 and the spatial-domain Mellin transform method described in 3.4.

For both methods we need to calculate the projections. The number of additions to be done is determined by the width W and height H. We will demonstrate the operations for $W = 720, H = 576$.

For the slice cross-correlation method we can calculate the projections for each frame only once. This means that we have $2WH$ more additions per frame for horizontal and vertical projection. For our example, the value is $2WH = 829,440$.

For the Mellin transform method, we need to calculate the projections for each frame twice because of the method for reducing the fixed-size display effects. For the method using cross-correlation of slices in the frequency domain, we do not need to use the method for reducing the effects, becuase the estimation results were sufficiently good. The first time that we need to perform projections is when calculating the global motion parameters between the previous frame and the current frame. We need to zero out parts in the current frame that do not appear in the previous frame, in order to reduce the fixed-size display effects. The second time that we need to perform projections is when calculating the global motion parameters between the current frame

and the next frame. We need to zero out parts in the current frame that do not appear in the next frame. The parts that need to be zeroed out in both cases may be different. This means that we have to calculate the projections twice and do $4WH$ additions. But, we can use the results of the projections from the first time and just subtract the pixels that we want to zero out, or add the pixels that were zeroed out and we want them for the current projection. Typically, this is quite a small number of pixels. For example, if the scale factor is 0.99, the width is 720 and the height is 576, we need to zero out 7 rows and 6 columns, which means we are left with a frame of $713 * 570$. The number of pixels for the subtraction is $8,310$. It means that we can calculate both projections by $2WH + 2 * 8,310 = 846,060$ additions.

We now summarize the number of additions and multiplications for each method. The number of operations for the slice cross-correlation method for the first pair of frames is $2WH = 829,440$ additions for the projections, $Wlog_2W + Hlog_2H = 12,116$ additions and multiplications for the Fourier transforms and $(2M-1)N_1 = 617,660$ additions and multiplications for the correlation. Per frame we have $1,459,216$ additions and $629,776$ multiplications. Assuming 30 frames per second we have $43,776,480 \approx 43.8MIPS$ for the additions and $18,893,280 \approx 18.9MIPS$ for the multiplications (MIPS - Million Instructions per Second).

The number of operations for the slice cross-correlation method for all the frames, except the first pair of frames, is $2WH = 829,440$ additions for the projections, $Wlog_2W+Hlog_2H = 12,116$ additions and multiplications for the Fourier transforms and $(2M-1)N_1 = 54,132$ additions and multiplications for the correlation. Per frame we have $895,688$ additions and $66,248$ multiplications. Assuming 30 frames per second we have $26,870,640 \approx 26.9MIPS$ for the additions and $1,987,440 \approx 2MIPS$ for the multiplications.

The number of operations for the Mellin transform is $2WH + 2W + 2H = 832,032$ additions for calculating the projections and $2 * 8,310 = 16,620$ additions and multiplications for the calculating the transform. Assuming 30 frames per second we have $25,459,560 \approx 25.5MIPS$ for the additions and $498,600 \approx 0.5MIPS$ for the multiplications.

Appendix C

Probabilty Distribution Function of Estimated Scale

In this Appendix we find the probability distribution function of the estimated scale factor $\tilde{\alpha}$, defined in section 4.2.

We want to find the distribution of the estimated scale $\tilde{\alpha}$. First we will find the distribution function of $\tilde{\alpha}^s$. Considering equations (4.11) and (4.12), we need to find the distribution of a ratio of two Gaussian random varaibles.

Define two normal distributed variables:

$$\begin{aligned} X &\sim \mathcal{N}(\mu_x, \sigma_x) \\ Y &\sim \mathcal{N}(\mu_y, \sigma_y) \end{aligned} \tag{C.1}$$

We also assume that the variables are uncorrelated. The distribution of their ratio is the distribution of a variable $Z = X/Y$ and is given by [7]:

$$p_Z(z) = \frac{b(z)\cdot c(z)}{a^3(z)} \frac{1}{\sqrt{2\pi}\sigma_x\sigma_y} \left[2\Phi\left(\frac{b(z)}{a(z)}\right) - 1\right] + \frac{1}{a^2(z)\cdot \pi\sigma_x\sigma_y} e^{-\frac{1}{2}\left(\frac{\mu_x^2}{\sigma_x^2}+\frac{\mu_y^2}{\sigma_y^2}\right)} \tag{C.2}$$

Where:

$$\begin{aligned} a(z) &= \sqrt{\frac{1}{\sigma_x^2}z^2 + \frac{1}{\sigma_y^2}} \\ b(z) &= \frac{\mu_x}{\sigma_x^2}z + \frac{\mu_y}{\sigma_y^2} \end{aligned}$$

$$
\begin{aligned}
c(z) &= e^{\frac{1}{2}\frac{b^2(z)}{a^2(z)}-\frac{1}{2}\left(\frac{\mu_x^2}{\sigma_x^2}+\frac{\mu_y^2}{\sigma_y^2}\right)} \\
\Phi(z) &= \int_{-\infty}^{z}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}u^2}\,du
\end{aligned}
\tag{C.3}
$$

Note, that $p_Z(z)$ is a standard Cauchy distribution for the case where $\mu_x = \mu_y = 0$ and $\sigma_x = \sigma_y = 1$. In our case μ_x and μ_y are positive and therefore not 0.

We define $\tilde{z} = \tilde{\alpha}^s$. Therefore, $f_{\tilde{z}}(\tilde{z})$ is given by (C.2) with expectations and variances as given in (4.12). The next step is finding the probability distribution function of $\tilde{\alpha}$ given the probability distribution function of $\tilde{z}$.

$$
\tilde{\alpha} = \tilde{z}^{\frac{1}{s}} = \left[\frac{\tilde{M}_{p_1}(s)}{\tilde{M}_{p_2}(s)}\right]^{\frac{1}{s}} = \left[\frac{M_{p_1}(s)+n_{M_1}}{M_{p_2}(s)+n_{M_2}}\right]^{\frac{1}{s}} \tag{C.4}
$$

We can define $\tilde{\alpha} = g(\tilde{z}) = \tilde{z}^{\frac{1}{s}}$. Therefore, the probability distribution function of $\tilde{\alpha}$ is given by:

$$
f_{\tilde{\alpha}}(\tilde{\alpha}) = \sum_{k}^{n(\tilde{\alpha})} \left|\frac{1}{g'\left(g_k^{-1}(\tilde{\alpha})\right)}\right| \cdot f_{\tilde{z}}\left(g_k^{-1}(\tilde{\alpha})\right) \tag{C.5}
$$

Here, g^{-1} denotes the inverse function and g' the derivative. $n(\tilde{\alpha})$ is the number of solutions for the equation $\tilde{\alpha} = g(\tilde{z})$, and $g_k^{-1}(\tilde{\alpha})$ are these solutions.

We also note that $g(\tilde{z})$ is monotonic so that we have:

$$
f_{\tilde{\alpha}}(\tilde{\alpha}) = \left|\frac{1}{g'\left(g_k^{-1}(\tilde{\alpha})\right)}\right| \cdot f_{\tilde{z}}\left(g_k^{-1}(\tilde{\alpha})\right) = \left|\frac{1}{g'(\tilde{\alpha}^s)}\right| f_z \tilde{\alpha}^s \tag{C.6}
$$

Since $g'(\tilde{\alpha}) = \frac{1}{s}\tilde{\alpha}^{\frac{1}{s}-1}$,we have that:

$$
f_{\tilde{\alpha}}(\tilde{\alpha}) = |s\tilde{\alpha}^{s-1}| f_z \tilde{\alpha}^s \tag{C.7}
$$

Equations (4.10), (C.2), (C.3), (C.4), (C.7) define the probability distribution function of $\tilde{\alpha}$.

$$
f_{\tilde{\alpha}}(\tilde{\alpha}) = |s\tilde{\alpha}^{s-1}| \frac{b(\tilde{\alpha}^s)\cdot c(\tilde{\alpha}^s)}{a^3(\tilde{\alpha}^s)} \frac{1}{\sqrt{2\pi}\sigma_{n_M}\sigma_{n_M}} \left[2\Phi\left(\frac{b(\tilde{\alpha}^s)}{a(\tilde{\alpha}^s)}\right) - 1\right]
$$

$$+ \quad \frac{1}{a^2(\tilde{\alpha}^s) \cdot \pi \sigma_{n_M} \sigma_{n_M}} e^{-\frac{1}{2}\left(\frac{M_{p_1}^2(s)}{\sigma_{n_M}^2} + \frac{M_{p_2}^2(s)}{\sigma_{n_M}^2}\right)} \tag{C.8}$$

Where

$$\begin{aligned} a\left(\tilde{\alpha}^s\right)) &= \sqrt{\frac{1}{\sigma_{n_M}^2}\tilde{\alpha}^2 s + \frac{1}{\sigma_{n_M}^2}} \\ b\left(\tilde{\alpha}^s\right) &= \frac{M_{p_1}(s)}{\sigma_{n_M}^2} z + \frac{M_{p_2}(s)}{\sigma_{n_M}^2} \\ c\left(\tilde{\alpha}^s\right) &= e^{\frac{1}{2}\frac{b^2(\tilde{\alpha}^s)}{a^2(\tilde{\alpha}^s)} - \frac{1}{2}\left(\frac{M_{p_1}^2(s)}{\sigma_{n_M}^2} + \frac{M_{p_2}^2(s)}{\sigma_{n_M}^2}\right)} \\ \Phi\left(\tilde{\alpha}^s\right) &= \int_{-\infty}^{\tilde{\alpha}^s} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}\, du \end{aligned} \tag{C.9}$$

We use $f_{\tilde{\alpha}}(\tilde{\alpha})$ for the perfomance analysis of the scale estimation.

Bibliography

[1] T. Koga, K. Iinuma, A. hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *in Proc. Nat. Telecommun. Conf.*, New Orleans, L.A., pp. G5.3.1-G5.3.5. Nov.-Dec. 1981.

[2] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans, Circuits Syst. Video Technol.*, Vol. 8, No. 4, pp. 369-377, Aug. 1998.

[3] Deigmoeller, J. Itagaki, T. Stoll, G., "An approach for an intelligent crop and scale application to adapt video for mobile TV," *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2008.

[4] B.S. Reddy, B.N. Chatterji, "An fft-based technique for translation, rotation,and scale-invariant image registratio," *IEEE trans. on Image Processing*, Vol. 5, issue 8, pp. 1266–1271, Aug. 1996.

[5] Dan E. Dudgeon, Russell M. Mersereau, "Multidimensional Digital Signal Processing," Prentice Hall, Englewood Cliffs, NJ, 1984.

[6] Xiaoxin Guo, Zhiwen Xu, Yinan Lu, Yunjie Pang, "An application of Fourier-Mellin transform in image registration," *The Fifth International Conference on Computer and Information Technology*, CIT 2005.

[7] D. V. Hinkley, "On the Ratio of Two Correlated Normal Random Variables," Biometrika, Vol. 56, No. 3, pp. 635-639. Dec. 1969.

[8] Siatras, S. Nikolaidis, N. Krinidis, M. Pitas, I. "Visual Lip Activity Detection and Speaker Detection Using Mouth Region Intensities," *IEEE trans. on Circuits and Systems for Video Technology*, pp.133-137, Jan. 2009.

[9] X. Sun, J. Foote, D. Kimber and B.S. Manjunath, "Region of interest extraction and virtual camera control based on panoramic video capturing," *IEEE Trans. On Multimedia*, vol. 7 no. 5, pp. 981-990, Oct. 2005.

[10] Rogerio Schmidt Feris, Teofilo Emidio de Campos, and Roberto Marcondes Cesar Junior, "Detection and Tracking of Facial Features in Video Sequences," MICAI, pp. 127-135, 2000.

[11] Einat Kidron, Yoav Y. Schechner, Michael Elad, "Pixels that sound," *Proc. IEEE CVPR*, Vol. 1, pp. 88-96, 2005.

[12] Zohar Barzelay, Yoav Y. Schechner, "Harmony in motion," *Proc. IEEE CVPR*, 2007.

[13] J. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler,"A tutorial on onset detection in music signals," *IEEE Trans. Speech and Audio Process*, vol. 13, pp. 1035-1047, 2005.

[14] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," *Proc. IEEE ICASSP*, vol. 6, pp. 3089-3092, 1999.

[15] J. Shi and C. Tomasi, "Good features to track," *Proc. IEEE CVPR*, pp. 593-600, 1994.

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

[17] S. Birchfield, "An implementation of the Kanade-Lucas-Tomasi feature tracker," Available at http://www.ces.clemson.edu/s̃tb/klt/.

[18] Paul Viola, Michael Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proc. IEEE CVPR*, Vol. 1, pp. 511-518, 2001.

[19] Paul Viola, Michael Jones, "Robust Real-time Object Detection," *International Journal of Computer Vision*, 2001.

[20] Yoav Freund and Robert E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Computational Learning Theory: Eurocolt '95*, pp. 23-37, Springer-Verlag, 1995.

[21] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 11 pp. 1254-1259, Nov 1998.

[22] Ki Tae Park, Young Shik Moon, "Automatic Extraction of Salient Objects Using Feature Maps," *IEEE ICASSP*, Vol. 1, pp.617-620, Apr. 2007.

[23] Remi Trichet, Bernard Merialdo, "Fast Video Object Selection for Interactive Television," *IEEE International Conference on Multimedia and Expo*, pp. 989-992, 2006.

[24] D. Comaniciu, V. Ramesh, and P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift," *Proc. IEEE CVPR*, pp. 142-151, 2000.

[25] Birchfield, S.T. Sriram Rangarajan, "Spatiograms Versus Histograms for Region-Based Tracking," *Proc. IEEE CVPR*, Vol. 2, pp. 1158-1163, 2005.

[26] Peiliang Wu, Lingfu Kong, Fengda Zhao, Xianshan Li, "Particle Filter Tracking Based on Color and SIFT Features," *IEEE ICALIP*, pp. 932-937, 2008.

[27] Ming-Chieh Chi, Mei-Juan Chen, Chia-Hung Yeh, Jyong-An Jhu, "Region-of-interest video coding based on rate and distortion variations for H.263+," *Signal Processing: Image Communication 23*, pp. 127-142, 2008.

[28] F. Ghorbel, "A complete invariant description for gray-level images by the harmonic analysis approach," *Pattern Recognition Letters*, Vol. 15 p. 1043-1051, Oct. 1994.

[29] Rudolf E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, Vol. 82, Series D, 1960.

[30] Gordon N.J., Salmond D.J. and Smith A.F.M. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings F Radar and Signal Processing*, Vol. 140, pp. 107-113, Apr. 1993.

[31] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, Vol. 50, No. 2, Feb. 2002.

[32] A. Doucet, A.M. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later," Technical report, Department of Statistics, University of British Columbia. Dec. 2008

[33] A. Doucet, "On sequential Monte Carlo methods for Bayesian filtering," Dept. Eng., Univ. Cambridge, UK, Tech. Rep., 1998.

[34] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian non-linear state space models," *Journal of Computational and Graphical Statistics*, Vol. 5, pp. 1-25. 1996.

[35] R. Douc, O. Cappe, "Comparison of resampling schemes for particle filtering," *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pp. 64-69, 2005.

[36] Hol, Jeroen D. Schon, Thomas B. Gustafsson, Fredrik, "On Resampling Algorithms for Particle Filters," *IEEE Nonlinear Statistical Signal Processing Workshop*, pp. 79-82, Sep. 2006.

[37] S.H. Choi, S.W. Lee, "Region tracking using perspective motion model," *Pattern Recognition*, Vol. 33, Issue. 12, pp. 2095-2098, Dec. 2000.

[38] M. Ritch, N. Canagarajah, "Motion-Based Video Object Tracking in the Compressed Domain," *IEEE ICIP 2007*, Vol. 6, pp. 301-304, Oct. 2007.

[39] Y.D. Guo, X.D. Gu, Z.B. Chen, Q.Q. Chen, and C. Wang, "Adaptive Video Presentation for Small Display While Maximize Visual Information," *Lecture Notes in Computer Science*, pp. 322-332, 2007.

[40] J. Yang, W. Lu and A. Waibel, "Skin-Color Modeling and Adaptation," *CMU CS Technical Report*, CMU-CS-97-146. May, 1997.

[41] Matei Mancas, Bernard Gosselin, Benoit Macq, "Segmentation using a region-growing thresholding," *Image Processing: Algorithms and Systems IV. Proceedings of the SPIE*, vol. 5672, pp. 388-398, 2005.

התאמה מבוססת איזור עניין של סרטי וידאו למכשירים ניידים

תמיר נוריאל

התאמה מבוססת איזור עניין של סרטי וידאו למכשירים ניידים

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר

מגיסטר למדעים בהנדסת חשמל

תמיר נוריאל

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

תשרי ה'תש"ע חיפה אוקטובר 2009

המחקר נעשה בהנחיית פרופ' דוד מלאך בפקולטה להנדסת חשמל

הכרת תודה

תודתי העמוקה לפרופ' מלאך על הדרכתו והנחייתו המסורה לאורך כל תקופת המחקר.
אני רוצה להודות גם לצוות המעבדה לעיבוד אותות ותמונות על העזרה והתמיכה הטכנית.
תודה מיוחדת לאשתי היקרה, טל, על הסבלנות והתמיכה. עבודה זו מוקדשת לה.

אני מודה לטכניון ולמאגד נגב על התמיכה הכספית הנדיבה בהשתלמותי.

תקציר

שידורי טלויזיה למכשירים ניידים הוא שוק גדל ומבטיח. תהליך ייצור של תוכן המתאים לצפייה במסכים קטנים הוא חלק מרכזי בצמיחה זו. המטרה של עבודה זו היא לפתח שיטה להתאמה של וידאו בפורמט רגיל לוידאו בפורמט בעל רזולוציה נמוכה יותר הנמצאת בשימוש נפוץ במכשירים סלולריים או מכשירים ניידים אחרים. הפתרון הנאיבי הוא לבצע הקטנה של כל מסגרת בנפרד לרזולוציה הרצויה במוצא. החסרון של פתרון זה הוא ההקטנה של כל העצמים בתמונה. על מנת להציג וידאו על מסך קטן, העצמים חייבים להיות בגודל מספיק על מנת שיוכלו לזהות אותם בקלות. לכן, בעבודות קודמות הוצע להציג רק את איזור העניין, כלומר איזור המכיל רק את התוכן החשוב ביותר.

בעבודה זו אנו מתמקדים בסצינות של קריינות חדשות או ראיון. אנו מגבילים את עצמנו לאיזור עניין מלבני, המקביל לצירים של הסרט המקורי. איזור העניין מוגדר להיות המלבן המכיל את פני הדובר או הדוברים. תהליך מציאת איזור העניין נעשה בשלב עריכת הוידאו, אשר בדרך כלל מבוצע באופן לא מקוון. בתחילת כל סצינה יש לסמן את איזור העניין ולעקוב אחריו בצורה חלקה לכל אורך הסצינה. לאחר מכן, יש לבצע התאמה לרזולוציה הנדרשת במוצא רק של איזור העניין.

בעבודה זו פיתחנו מספר כלים לעורך הוידאו. פיתחנו אלגוריתם חדשני לעקיבה אחר איזור העניין המסומן בתחילת סצינה עד לסופה. מיקום איזור העניין בתחילת הסצינה מסומן על ידי העורך. האלגוריתם משערך את התנועה הגלובלית הנגרמת כתוצאה מתנועת המצלמה או שינויים בפרמטרים שלה וכן את התנועה המקומית הנגרמת מתזוזות של הדובר. אנו מניחים מודל מצלמה הכוללת תנועות של הזזה לימין או שמאל, הזזה למעלה או למטה וכן פעולות זום, כלומר שינוי סקאלה. אנו מציגים תחילה אלגוריתם לשערוך פרמטרים אלו המתבסס על ביצוע התמרת פוריה דו-מימדית על כל המסגרת. אנו מראים כי בפועל נדרשים רק פרוסות מתוך ההתמרה

הדו-מימדית. היחס בין פרוסות של התמרת פוריה דו-מימדית לבין התמרת פוריה חד-מימדית של הטלות אופקיות ואנכיות נתונה על ידי משפט ידוע בשם "משפט פרוסה-הטלה". השיטה שלנו מתבססת לכן על ביצוע הטלות אופקיות ואנכיות. התמרת הטלות אלו למישור פוריה מאפשרת שערוך פרמטרי התנועה ביעילות. שימוש בהטלות אלו חוסך את הסיבוכיות החישובית הדרושה לצורך ביצוע התמרת פוריה דו-מימדית. שיטה אחרת לשערוך הפרמטרים היא על ידי הפעלת התמרת מלין ($Mellin$) דו-מימדית על כל המסגרת. התמרה זו מאפשרת הפיכה של גורם שינוי הסקאלה לגורם כפלי, שניתן לשערכו בסיבוכיות חישובית נמוכה יותר. אנו נראה שיטה נוספת לשערוך הסקאלה על ידי הפעלת התמרת מלין חד מימדית על הפרוסות במישור התדר, וכך להמנע מהשימוש בהתמרות דו-מימדיות. אנו נניח בנוסף כי שינויים בפרמטרי התנועה בין מסגרת למסגרת הם קטנים מספיק כך שהפרמטרים ממשוערכים ממסגרת מסוימת יכולים לשמש כניחוש התחלתי לפרמטרים במסגרת הבאה. בהנתן ניחוש התחלתי שכזה, אנו מראים שיטה יעילה אף יותר לשערוך הפרמטרים. בשיטה זו נשתמש בהתמרת מלין במישור המרחב ישירות על ההטלות האופקיות והאנכיות. אנו מחשבים ומשווים את הסיבוכיות החישובית של השיטות השונות. כמו כן, אנו מציגים מודל סטטיסטי אשר מאפשר לנתח את ביצועי משערך פרמטרי התנועה.

שידור סרט המכיל רק את איזור העניין עלול להראות לצופה כרוטט מדי. הסיבה לכך היא שגיאות בשערוך פרמטרי התנועה הנגרמות על ידי רעש או שגיאות מודל. לכן, על מנת להחליק את התוצאה, אנו מניחים מודל תנועה עבור איזור העניין. המודל מניח כי מרכז איזור העניין נע במהירות קבועה בין מסגרת למסגרת וכי מהירות זו משתנה על ידי תאוצה אקראית בעלת התפלגות גאוסית. באופן דומה, המודל מניח שינויים במהירות קבועה בין מסגרת למסגרת עבור האורך והרוחב של איזור העניין וכן תאוצה אקראית עבור מהירות זו בעלת התפלגות גאוסית. אנו משתמשים במסנן קלמן לשערוך מיקום ומהירות איזור העניין בכל מסגרת. עבור מודל לינארי והתפלגות גאוסית, מסנן קלמן הוא המסנן האופטימלי במובן של מינימום שגיאה ריבועית. שערוך המיקום באופן זה מביא לתוצאות חלקות יותר מבחינת הצופה. בנוסף, עבור מקרים מסובכים יותר, למשל במקרה של הפרעה הגורמת להסתרה חלקית, אנו משתמשים במסנן חלקיקים. כל חלקיק מייצג מצב אפשרי של איזור העניין מבחינת מיקום ומהירות. הרעיון המרכזי במסנן חלקיקים הוא לייצג את פונקציית צפיפות ההסתברות של מצב איזור העניין על ידי מתן משקל שונה לכל חלקיק לפי התאמתו למדידות, כלומר לפרמטרי התנועה המשוערכים. חלקיק מקבל משקל גבוה יותר ככל שהוא מתאים יותר למדידות. המיקום והמהירות של איזור העניין נקבעים לפי

ממוצע משוקלל של החלקיקים השונים, המייצג תוחלת של מצב איזור העניין. ייצוג פונקצית צפיפות ההתפלגות על ידי חלקיקים מאפשרת לייצג פונקציות התפלגות כלשהן. במקרים שבהם הפונקציה האמיתית היא לא גאוסיאנית, למשל אם היא בי-מודלית, מסנן חלקיקים צפוי להשיג שערוך טוב יותר ממסנן קלמן.

גם לאחר החלקה של תוצאות העקיבה לא רצוי לשדר רק את איזור העניין. לכך יש שתי סיבות. סיבה ראשונה היא שאנו לא רוצים להזיז את התמונה לפי תנועות הראש של הדובר. למשל, אם הדובר מהנהן אין צורך להזיז את התמונה למעלה ולמטה כדי לעקוב אחריו. סיבה שנייה היא שאיזור העניין הוא מלבן בגודל נתון. אם נבצע ישירות שינוי רזולוציה רק על איזור העניין אנחנו עלולים לקבל עיוות מאחר והיחס בין האורך לרוחב של איזור העניין שונה מהיחס בין אורך לרוחב של מסך המוצא. לכן, אנו מגדירים איזור נוסף בתור איזור התצוגה. איזור זה תמיד יכלול את איזור העניין ויהיה בעל אותו יחס אורך-רוחב של מסך המוצא. כך נמנע את עיוות התמונה.

בנוסף, מוצג אלגוריתם עזר לעורך לזיהוי איזור העניין בתחילת הסצינה. העורך יידרש ללחוץ לחיצה אחת בלבד על פני הדובר והאלגוריתם יימצא את איזור העניין המכיל את פני הדובר. על מנת למצוא את האיזור אנו משתמשים בגלאי של צבע עור וכן באלגוריתם סגמנטציה המשתמש בערכי הבהירות.

העורך יכול לסמן מספר איזורי עניין לעקיבה במשך הסצינה, אולם בזמן מסוים רק אחד מהם ישודר למוצא. כך יהיה ניתן לסמן שני אנשים בזמן ראיון כאיזורי עניין, אך לשדר רק את הדובר הפעיל בכל מסגרת. כלי נוסף שמוצג בעבודה משמש למציאת הדובר הפעיל במקרה של מספר איזורי עניין עם מספר דוברים. בחנו תחילה שיטות לזיהוי הדובר הפעיל על סמך מידע מהתמונה בלבד. למשל, שימוש בבהירות פיקסלים באיזור הפה. בהמשך גם בחנו שיטה למציאת הדובר הפעיל על ידי שילוב מידע פס הקול. מתבצעת עקיבה אחר מספר נקודות עניין על פני הדוברים השונים ונמצאים הזמנים בהם יש שינויים משמעותיים במאפייני תנועת כל נקודה, למשל, שינוי בתאוצה. במקביל מוצאים זמני אירועים משמעותיים בפס הקול, למשל, עלייה חדה באנרגיה היכולה להצביע על תחילת הברה. הקורלציה בין זמני האירועים בוידאו ובפס-הקול משמשת לזיהוי הדובר הפעיל בכל רגע.

כל הכלים שפותחו נבדקו הן בסימולציות וכן עבור סצינות אמיתיות שהוקלטו מש-ידורי טלויזיה.