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

T. Nuriel , and D. Malah, *Life Fellow, IEEE*

***Abstract*—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. 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). In this work we focus on news broadcasting and interview scenes, so the ROI contains the speaker's face. At the beginning of every scene, the ROI is marked by an 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 estimate the motion parameters using horizontal and vertical projections. The slice-projection theorem and the Mellin transform are used for reducing complexity.**

I. Introduction

In recent years Mobile TV is a growing and promising market. Most of the video content today have SD (standard definition) resolution, which is up to 720x576 pixels. On the other hand, the resolution of mobile devides displays, like cellphones or palm PCs, is usually smaller, e.g., CIF: 352x288 pixels. Therefore, production of a content suitable for watching video on small displays becomes an important task [1]. A naive adjustment method is to scale 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 drawback. It reduces the size 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 [1]. 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 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. The location of the ROI in each frame 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 scaled.

One of the methods for tracking a region is based on a color histogram (e.g., [2]). The algorithm searches in the current frame for a region with a color histogram similar to the histogram of the ROI from the previous frame. There are four parameters for ROI location: the width, the height and two for the center coordinates. The search is usually performed by the N-step [3] or diamond search [4] methods, in the four-parameter space.

In [5] 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, 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 for tracking. In [6] it is proposed to use color cues and local scale-invariant feature transform (SIFT) features with particle filtering [7]. The weight of each particle 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 model parameter estimation 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 any-way for other tasks, such as compression [8]. A combination of tracking by visual features and a motion model was proposed in [9], where a fitting of the motion vectors in a target region to the perspective motion model is suggested. The proposed method considers the motion vectors only of the feature points in the target region.

The main disadvantage of all the methods described above is the high computational complexity required for on an ordinary editor's PC. In our work we develop computationally efficient algorithms for detecting and tracking the ROI.

The paper is structured as follows. In section II we develop the tracking algorithm based on horizontal and vertical projections. In section III we reduce our algorithm complexity using the Mellin transform. In section IV we introduce the region of display for a stable video required for broadcasting quality. In section V we present the computational complexity of the tracking algorithms and in section VI we present their performance analysis. In section VII we present experimental results and then conclude in section VIII.

T. Nuriel is with the Department of Electrical Engineering Technion- Israel Institute of Technology (Email: tamir.nuriel@gmail.com)

D. Malah is with the Department of Electrical Engineering Technion- Israel Institute of Technology (Email: malah@ee.technion.ac.il)

II. Region Tracking by Horizontal and Vertical Projections

In this section 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 editor. Our goal is to track the ROI location up to the end of the scene. We 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 find the local motion of the object of interest (e.g., a face) inside the ROI, and update the ROI location accordingly.

In news and interview scenes, the camera usually only performs 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, we assume that 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:

$$f_2(x, y) = f_1(\alpha x + d_x, \alpha y + d_y) \tag{1}$$

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.

To estimate α, we need to find a transformation which is invariant to all parameter values, except for scale.

Transforming eqn.(1) into the frequency domain:

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

Since 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 2D Fourier transform. Instead, we can use the *projection-slice theorem* [10]. The theorem states that the Fourier transform of the projection of a 2D function onto a line is equal to a slice through the origin of the 2D 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:

$$P(x) = \int_{-\infty}^{\infty} f(x, y)\, dy \tag{3}$$

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

$$F(f_x, f_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\, e^{-2j\pi(xf_x + yf_y)} dxdy \tag{4}$$

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

$$S(f_x) = \int_{-\infty}^{\infty} P(x)\, e^{-2j\pi x f_x} dx \tag{5}$$

Using eqn.(2) and taking the absolute value of a slice of the transform corresponding to the line $f_y = 0$, we obtain:

$$|S_2(f_x)| = \frac{1}{\alpha^2}\left|S_1\left(\frac{f_x}{\alpha}\right)\right| \tag{6}$$

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, respectively. By converting the frequency axes in eqn.(6) to logarithmic scale we have:

$$|S_2(log f_x)| = \frac{1}{\alpha^2}|S_1(log f_x - log\alpha)| \tag{7}$$

The scale factor α can therefore be estimated by performing a cross-correlation between both sides of equation (7), and finding the shift for which the maximum is achieved. Since natural images are smooth, high frequencies do not contain much information, and hence we consider only part of the frequency band. We also don't want to work with frequencies close to zero becuase of numerical problems when changing to logarithmic scale. Hence, we selected the band edges as: $f_{min} = 0.2\pi, f_{max} = 0.8\pi$. The relation between the frequencies in logarithmic scale is:

$$\alpha = base^{-lag}; \;\; base = (f_{max}/f_{min})^{\frac{1}{N-1}}, \tag{8}$$

where N is the number of points in the logarithmic scale. Given the desired resolution we can calculate the number of points to use. We can repeat the estimation process for the vertical slice $f_x = 0$, and then average the results.

III. Region Tracking using the Mellin Transform

A. Scale Estimation using the Mellin Transform in the Frequency-domain

The cross-correlation method for estimating the scale has relatively high complexity, bacause we need to calculate the cross-correlation for a large number of lag values (typically, hundreds) in order to achieve good resolution for the scale estimation. In a typical scene there are 1000 frames. We need a good resolution becuase the estimation errors are getting accumulated. For example, if instead of $\alpha = 1$ we estimate $\alpha = 1.001$, the total scale factor is $\alpha_T = 1.001^{1000} \approx 2.7$.

The Mellin transform for scale estimation was used in [11], where a registration algorithm for the alignment of images using the 2D analytical Fourier-Mellin transform is proposed. To reduce complexity, we apply the 1D Mellin transform.

Definition - One-dimensional Mellin transform [12]:

$$M(s) = \int f(x)\, x^{s-1} dx, \tag{9}$$

where s is a parameter.

The relation between the 1D Mellin transforms of the absolute values of the Fourier transforms of the projections is given by:

$$M_2(s) = \int |S_2(x)|\, x^{s-1} dx = \alpha^{s-2} M_1(s) \tag{10}$$

Now we can estimate the scale by:

$$\alpha = \left[\frac{M_1(s)}{M_2(s)}\right]^{\frac{1}{2-s}} \tag{11}$$

B. *Effect of a Fixed-Size Display*

The experimental results for the scale estimation described in the previous subsection were not good enough. The main reason 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. Therefore, the projection is different from the theoretical one. We can solve this by zeroing the parts of the each frame that do not appear in the other frame. After this operation, we have almost no such effects, except for those that appear because we are working at a one-pixel resolution. Assuming that the change of scale and shifts in consecutives frames are small, and assuming that we have the previous frame scale and shifts estimation, we can use them as a good guess for the parameters needed for the zeroing operation.

C. *Scale Estimation using Spatial-Domain Mellin transform*

We perform the projections on the current frame after zeroing parts of the frame to reduce the effect of the fixed-size display. The relation between the projections after normalization by the number of summed pixels is:

$$P_2\left(x\right) = P_1\left(\alpha x + d_x\right) \tag{12}$$

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 substitute $P_1\left(\alpha x + d_x\right)$, so the relation becomes:

$$P_2\left(x\right) = P_1\left(\alpha x\right) \tag{13}$$

Then, we can apply the Mellin transform directly on the projections in the spatial domain to estimate the scale. By applying the Mellin transform on both sides of eqn.(13), the scale can be estimated from:

$$\alpha = \left[\frac{M_{p_1}\left(s\right)}{M_{p_2}\left(s\right)}\right]^{\frac{1}{s}} \tag{14}$$

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 II. 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 improved translation parameters as described in the next subsection.

The parameter s has a significant effect on the estimation. We want to find an empirical value of s that yields the lowest error in the scale estimation. 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. The mean value of a pixel, $MeanPixelValue$, is 128.

$$SNR = 20log_{10}\left(\frac{MeanPixelValue}{\sigma}\right) \tag{15}$$

We tested for several SNR values, with similar results. Here we demonstrate the result for $SNR = 20dB$. We found the

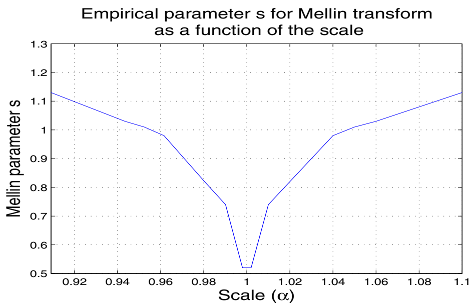


Fig. 1: Empirical parameter s for spatial-domain Mellin transform as a function of the scale α.

parameter s which yields the best result for each scale. The results are shown in Fig. 1. The solid line only connects the discrete values of s for the true α values that were tested.

D. *Global Translation Estimation*

After estimating the scale factor between any two consecutive frames, we can update the estimate of horizontal and vertical shifts between them. The relation between the projections is given by eqn.(12). 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 $P_1^r(x)$.

$$P_1^r(x) = P_1(\alpha x) \tag{16}$$

The relation between the scaled projection for the previous frame and the projection from the current frame is:

$$P_1^r(x) = P_2\left(x - \frac{d_x}{\alpha}\right) \tag{17}$$

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

E. *Local Motion Estimation*

The local motion is typically the motion of the speaker inside the ROI. The ROI window is updated according to the estimated global motion parameters. We assume there is no local scale change (e.g., no object is approaching the camera). We calculate horizontal and vertical projections, not for the whole frames, but only for the ROI in the previous frame and in the current frame. We scale 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.

IV. Region of Display Determination

The ROI is needed for viewing on a mobile device. But, we should not extract just the ROI, becuase of two reasons. First, the speaker might move while talking. Tracking this movement causes the output video to be jittery, in the viewers' eyes. Second, the ROI size may be arbitrary, therefore scaling it to a predefined output size may cause distortion. Therefore, we define a region of display (ROD) as the region to be extracted, which always contains the ROI. From our application requirements, the extracted region is limited to two different sizes only: the whole frame or a smaller pre-defined output size (e.g., CIF). When the ROD is determined to be the whole frame, we have to scale it to the output size. In the first frame of each scene we check the size of the ROI. If it is smaller then the output size, we mark the ROD as a region of this 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, so small movements of the ROI will not cause any jitter. When the ROI is getting too close to the borders of the ROD, we change the ROD center location, smoothly, for several frames, to make it look like a natural camera movement. When the ROI becomes bigger (due to a zoom-in) than the ROD, we change the ROD to be the whole frame. We apply this change in a single frame, so it looks like a scene cut. When the ROI becomes small enough (due to a zoom-out), we change the ROD size to the output size.

V. Complexity of Scale Estimation Algorithms for ROI Tracking

We compare the complexity of scale estimation by cross-correlation of slices in the frequency domain, described in section II, and the method that applies spatial-domain Mellin transofrm, described in section III-C.

In the first method we perform slice cross-correlation in a logarithmic scale. The relation between the frequencies is given by equation (8). Given the desired resolution we wish to calculate the number of points to use. Since we use a logarithmic scale the resolution is different for different values of scale. We will find the resolution around $\alpha = 1$. The closest possible value of scale to 1 is $\alpha = base^1$. Therefore the resolution near $\alpha = 1$ is:

$$\Delta\alpha = base^1 - base^0 = base - 1 \approx resolution \tag{18}$$

Assume that we want a resolution of 0.001 we have:

$$resolution = 0.001 \Rightarrow N = 1388 \tag{19}$$

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

$$M = \frac{log\left(\alpha_{max}\right)}{log\left(base\right)} = \frac{-log\left(\alpha_{min}\right)}{log\left(base\right)} \approx 223 \tag{20}$$

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 first 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 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 Mellin transform in the spatial domain method, we do not need to perform a Fourier transform. For this method the number of operations is equal to the length of the projection.

The rest of the complexity analysis is given in [13]. The results are summarized in Table I. We assume that each frame has width $W = 720$ and height $H = 576$, and a frame rate of 30 frames per second. The complexity of the slice-cross correlation method is calculated assuming the cross-correlation is performed only in the vicinity of the previous estimated scale. The complexity of the slice cross-correlation method for the first pair is calculated assuming we use M as in 20. For comparison between them, we calculate their complexity as if we use each of the methods all the frames in the scene.

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 I: Summary of scale estimation algorithms complexity. The first line is shown only for comparison with methods using 2D transforms.

Considering that we will perform the correlation method only once in a scene, we have that the average complexity is determined by the last line and its value is around $26MIPS$. It can be easily implemented for real-time applications on a Pentium with a 3GHz processor. Most consuming operations, in the scale estimation process, are the projections, requiring $\approx 24MIPS$ for additions. The projections can be implemented using parallel computing, so the algorithm can work efficiently on a single computer with multiple processors.

VI. Performance Analysis

In this section we calculate the variance of the scale estimation error using the spatial-domain Mellin transform method. We first calculate the pdf of the estimated scale. We assume 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 several different true scale values and several different signal to noise ratios (SNR). The first operation is the projection. For example, let's consider the vertical projection:

$$P\left(x\right) = \frac{1}{H}\sum_{i=1}^{H} f\left(x, y_i\right) \tag{21}$$

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

$$\begin{aligned} \tilde{P}\left(x\right) &= \frac{1}{H}\sum_{i=1}^{H} f\left(x, y_i\right) + n\left(x, y_i\right) = P\left(x\right) + n\left(x\right) \\ \sigma_n^2 &= \frac{\sigma^2}{H} \end{aligned} \tag{22}$$

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:

$$M(s) = \int_x P(x-d_x)x^{s-1}dx = \int_x P(x)(x+d_x)^{s-1}dx \tag{23}$$

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, is given by:

$$\begin{aligned}\tilde{M}(s) &= \frac{1}{W}\sum_x \tilde{P}(x)\,x^{s-1} = \frac{1}{W}\sum_x \left[P(x)+n(x)\right]x^{s-1} \\ &= M(s) + \frac{1}{W}\sum n(x)\,x^{s-1} = M(s)+n_M \end{aligned} \tag{24}$$

The variance of the noise is now given by:

$$\sigma^2_{n_M} = \frac{\sigma^2}{W^2 H}\sum_x \left(x^{s-1}\right)^2 \tag{25}$$

We know that the scale estimation is given by equation (14). Therefore, we have:

$$\tilde{\alpha}^s = \frac{\tilde{M}_{p_1}(s)}{\tilde{M}_{p_2}(s)} = \frac{M_{p_1}(s)+n_{M_1}}{M_{p_2}(s)+n_{M_2}} \tag{26}$$

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

$$\begin{aligned}\tilde{M}_{p_1}(s) &\sim \mathcal{N}\left(M_{p_1}(s), \sigma_{n_M}\right) \\ \tilde{M}_{p_1}(s) &\sim \mathcal{N}\left(M_{p_1}(s), \sigma_{n_M}\right)\end{aligned} \tag{27}$$

We want to find the distribution of the estimated scale $\tilde{\alpha}$ by equation (26). We use the pdf of the ratio of two normal random variables given by [14]. The rest of the anaylsis is given in [13] and the result is:

$$\begin{aligned} &f_{\tilde{\alpha}}(\tilde{\alpha}) = \qquad (28)\\ &|s\tilde{\alpha}^{s-1}|\frac{b(\tilde{\alpha}^s)\cdot c(\tilde{\alpha}^s)}{a^3(\tilde{\alpha}^s)}\frac{1}{\sqrt{2\pi}\sigma_{n_M}\sigma_{n_M}}\left[2\Phi\left(\frac{b(\tilde{\alpha}^s)}{a(\tilde{\alpha}^s)}\right)-1\right] \\ &+\frac{1}{a^2(\tilde{\alpha}^s)\cdot\pi\sigma_{n_M}\sigma_{n_M}}exp\left[-\frac{1}{2}\left(\frac{M^2_{p_1}(s)}{\sigma^2_{n_M}}+\frac{M^2_{p_2}(s)}{\sigma^2_{n_M}}\right)\right]\end{aligned}$$

Where,

$$\begin{aligned} a(\tilde{\alpha}^s)) &= \sqrt{\frac{1}{\sigma^2_{n_M}}\tilde{\alpha}^2 s + \frac{1}{\sigma^2_{n_M}}} \\ b(\tilde{\alpha}^s) &= \frac{M_{p_1}(s)}{\sigma^2_{n_M}}z + \frac{M_{p_2}(s)}{\sigma^2_{n_M}} \\ c(\tilde{\alpha}^s) &= exp\left[\frac{1}{2}\frac{b^2(\tilde{\alpha}^s)}{a^2(\tilde{\alpha}^s)} - \frac{1}{2}\left(\frac{M^2_{p_1}(s)}{\sigma^2_{n_M}}+\frac{M^2_{p_2}(s)}{\sigma^2_{n_M}}\right)\right] \\ \Phi(\tilde{\alpha}^s) &= \int_{-\infty}^{\tilde{\alpha}^s}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}u^2}\,du \end{aligned} \tag{29}$$

We can repeat the above estimation process for the horizontal projection. Assuming that we have scale estimations $\tilde{\alpha}_V, \tilde{\alpha}_H$, we can estimate the scale by averaging them.

$$\tilde{\alpha} = 0.5\left(\tilde{\alpha}_V + \tilde{\alpha}_H\right) \tag{30}$$

For simplicity, we will assume that the noises in the 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:

$$\begin{aligned} f_{0.5\tilde{\alpha}_V}(x) &= 2f_{\tilde{\alpha}_V}(2x) \\ f_{0.5\tilde{\alpha}_H}(x) &= 2f_{\tilde{\alpha}_H}(2x) \end{aligned} \tag{31}$$

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

$$f_{\tilde{\alpha}}(\tilde{\alpha}) = \int_{-\infty}^{\infty} f_{0.5\tilde{\alpha}_V}(y)\,f_{0.5\tilde{\alpha}_H}(x-y)\,dy \tag{32}$$

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 (22) and (24), 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.

$$\begin{aligned} SNR &= 20log_{10}\left(\frac{MeanPixelValue}{\sigma}\right) \\ \Rightarrow \sigma &= 10^{-SNR/20}MeanPixelValue \end{aligned} \tag{33}$$

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. 2.

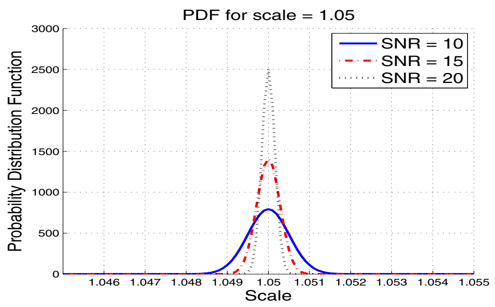


Fig. 2: Examples for probability distribution functions of scale estimation.

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 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 III-C. We calculated an histogram of the estimated values. The normalized histogram is compared to the probabilty density function in Fig. 3. We can see from the figure that the simulations results are quite similar to those expected from theory.

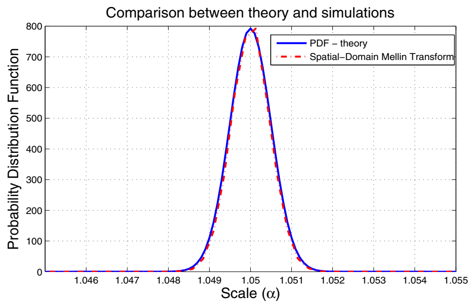


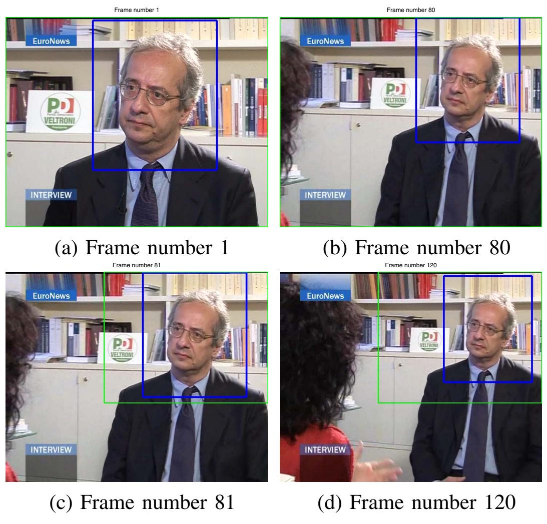
Fig. 3: Comparison between estimation methods. For example, for $\alpha = 1.05$ and $SNR = 20dB$, the estimated scale error had a standard deviation of $1.6 * 10^{-4}$

VII. Experimental Results

We tested our algorithm using several scenes. We demostrate here results for a particular scene, In the scene the camera zooms-out and also moves to the left and down. We can see that the algorithm tracks the head from the first frame in Fig. 4a till the end of the scene in Fig. 4d. We can also see that when the ROI is small enough, the ROD is changed from the whole frame in Fig. 4b, to be a region of pre-defined output size as in Fig. 4c.

VIII. Conclusion

We developed a novel algorithm for tracking a ROI for adaptation of video to mobile devices. We assumed a zoom, tilt and pan motion model. We showed that slices of the 2D Fourier transform alone are sufficient for scale estimation, enabling us to use the slice-projection theorem and reduce complexity. We developed an algorithm based on the horizontal and vertical projections. We reduced complexity further by applying the 1D Mellin transform to the slices. Assuming that the changes in the motion parameters of consecutives frames are small, we reduced even more the complexity by using the Mellin transform in the spatial-domain. We introduced the ROD as the region to be extracted. The ROD always contains the ROI and its location is changed smoothly, resulting in a stable video, as required for broadcasting quality. We calculated the computational complexity of the algorithms and did a performance analysis, using a statistical model that enabled us to calculate the probabilty distribution function of the estimated scale. We will test the model results by simulations. Future work could concentrate on improving tracking robustness, since we found that in case of external interference, such as occlusion, the global motion estimation yields poor results. Using advanced tools like particle filtering should enable overcoming this problem.



(a) Frame number 1 (b) Frame number 80
(c) Frame number 81 (d) Frame number 120

Fig. 4: Tracking Example for a zoom-out scene - the ROI marked as a blue rectangle and the ROD marked as a green rectangle. The ROD always contains the ROI. In frame 81 the ROI is small enough and the ROD changes to a smaller pre-defined size.

Acknowledgment

This research was performed at the Signal and Image Processing Lab (SIPL), Technion I.I.T and funded by the Negev consurtium of the Israeli Ministry of Industry and Trade. The authors would like to thank the devoted SIPL staff Nimrod Peleg, Yair Moshe, Ziva Avni and Avi Rosen for their technical support.

References

[1] 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.
[2] D. Comaniciu, V. Ramesh, and P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift," *Proc. IEEE CVPR*, pp. 142-151, 2000.
[3] 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. 1981.
[4] 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.
[5] Birchfield, S.T. Sriram Rangarajan, "Spatiograms Versus Histograms for Region-Based Tracking," *Proc. IEEE CVPR*, Vol.2, pp.1158-1163, 2005.
[6] Peiliang Wu, Lingfu Kong, Fengda Zhao, Xianshan Li, "Particle Filter Tracking Based on Color and SIFT Features," *IEEE ICALIP*, pp. 932-937, 2008.
[7] 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.
[8] M. Ritch, N. Canagarajah, "Motion-Based Video Object Tracking in the Compressed Domain," *IEEE ICIP 2007*, Vol. 6, pp. 301-304, Oct. 2007.
[9] S.H. Choi, S.W. Lee, "Region tracking using perspective motion model," *Pattern Recognition*, Vol. 33, Issue. 12, pp. 2095-2098, Dec. 2000.
[10] Dan E. Dudgeon, Russell M. Mersereau, "Multidimensional Digital Signal Processing," Prentice Hall, Englewood Cliffs, NJ, 1984.
[11] 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.
[12] Flajolet P., Gourdon X., Dumas P., "Mellin transforms and asymptotics: Harmonic sums," *Theoretical Computer Science*, Vol.144, pp.3-58, 1995.
[13] Nuriel T., "Region-of-Interest based Adaptation of Video to Mobile Devices," M.Sc. thesis, Technion - Israel Institute of Technology, Dec. 2009.
[14] D. V. Hinkley, "On the Ratio of Two Correlated Normal Random Variables," Biometrika, Vol. 56, No. 3, pp. 635-639. Dec. 1969.