A Segmentation System with Model Assisted Completion of Video Objects

Dirk Farin[a], Peter H.N. de With[b], Wolfgang Effelsberg[a]
[a]Dept. of Computer Science IV, University of Mannheim, Germany
[b]LogicaCMG / Eindhoven University of Technology, The Netherlands

ABSTRACT

This paper presents a new algorithm for video-object segmentation, which combines motion-based segmentation, high-level object-model detection, and spatial segmentation into a single framework. This joint approach overcomes the disadvantages of these algorithms when applied independently. These disadvantages include the low semantic accuracy of spatial segmentation and the inexact object boundaries obtained from object-model matching and motion segmentation. The now proposed algorithm alleviates three problems common to all motion-based segmentation algorithms. First, it completes object areas that cannot be clearly distinguished from the background because their color is near the background color. Second, parts of the object that are not considered to belong to the object since they are not moving, are still added to the object mask. Finally, when several objects are moving, of which only one is of interest, it is detected that the remaining regions do not belong to any object-model and these regions are removed from the foreground. This suppresses regions erroneously considered as moving or objects that are moving but that are completely irrelevant to the user.

Keywords: Video-object segmentation, motion segmentation, color segmentation, model-based object detection, inexact graph-matching, dynamic programming.

1. INTRODUCTION AND RELATED WORK

Segmentation of video sequences into separate video-objects is an important step for a large set of applications. These include object-based video-coding algorithms like MPEG-4, the analysis of video sequences, or surveillance applications. Segmentation-techniques can be coarsely classified into spatial and temporal segmentation. Spatial segmentation provides accurate region boundaries, but since it is working on the signal level, a semantically meaningful object separation cannot be found without further information. On the other hand, semantic objects can be identified rather accurately by observing areas that are moving consistently. Hence, temporal segmentation techniques can give superior results for separating different objects. Unfortunately, temporal segmentation does not provide as accurate object boundaries as can be obtained from spatial segmentation. This is why several algorithms have been proposed in the literature to combine the output of spatial and temporal segmentations into a final segmentation mask.

For combining motion information with spatial segmentation, Fablet *et al.*[1] propose to use a neighbourhood graph over spatially segmented image-regions. A Markovian labeling process is carried out on this graph, introducing motion information through the definition of clique potentials. However, this approach does only differentiate between regions moving compatible with dominant motion, and regions which do not. Hence, when only part of the object is moving, the segmentation algorithm will not cover the complete object. A comparable algorithm has been proposed be Patras *et al.*[2] The approach suggested in Alatan *et al.*[3] obtains motion information in the form of change detection masks.[4] Fusion of motion cues and a spatial color-segmentation is carried out using a system of hard-coded rules.

A weak point of the previous algorithms is the fact that fusion of motion and spatial information is still done in a heuristic way, without integrating real semantic knowledge about the object to be extracted. However, difficult segmentation problems exist which cannot possibly be solved without this high-level knowledge. For example, take a head-and-shoulder sequence where only the head is moving and the body is static. Even though every human would probably consider head and body to be a single object, the computer has no indication why the body should be assigned to the same object as the head. Consequently, the body will be treated as background.

To solve this problem, we propose a new algorithm for the fusion of temporal and spatial segmentation which uses a model of the object to enable the algorithm to find the complete object even when parts of it do

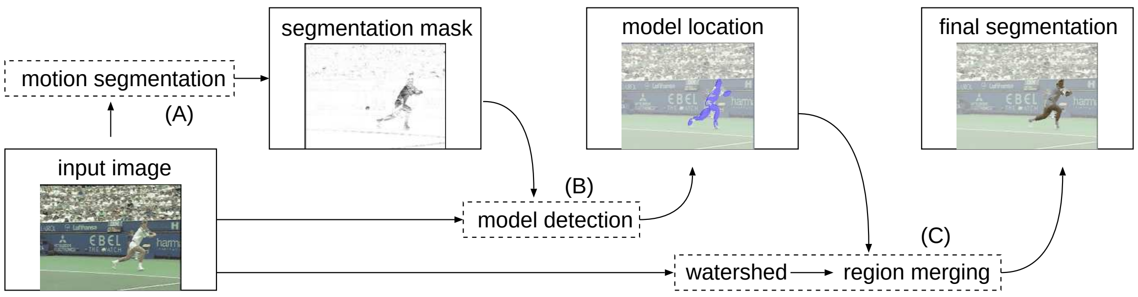


Figure 1. Overview of the segmentation system.

not move. The object-model is based on a graph representation which especially allows for articulated object motion. Our segmentation algorithm is composed of three main parts, which are described in the following sections. After a short outline of the system framework in the next section, motion segmentation is described in Section 3. Section 4 introduces the object-model used, describes the model-editor for creating the object-models, and presents the matching algorithm to detect the model in the image. Spatial segmentation and integration of the previously detected object-model is described in Section 5. The paper concludes with presenting some experiments and giving conclusions and an outlook.

2. SEGMENTATION SYSTEM ARCHITECTURE

Our segmentation system comprises three main steps, each using a different algorithm class, but all working together to combine their respective advantages for getting good results. An overview flow-graph of the algorithm is depicted in Figure 1. First (A), a motion-segmentation is employed to find the moving regions in the image, giving an approximate location of the object. This step builds a synthetic background image of the sequence by composing all available frames of the sequence. Camera motion is compensated using a robust global-motion estimation and all input frames are stitched together into a single background mosaic. Foreground objects are removed using a temporal filter over the motion-compensated input frames.

The background mosaic is used to detect objects by subtracting a camera-motion compensated version of the background from the current input image. In the ideal case, this can already provide an accurate segmentation of the moving object. However, there are several cases when this scheme will fail. If the sequence is very short or foreground objects do not move much, it may be impossible to reconstruct the background and, consequently, foreground objects will remain in the mosaic. This can cause two kinds of errors: objects may have holes in them if the object stays at the same position in the input image, or the errors in the mosaic may show as spurious objects in the difference-frame which will further lead to errors in the segmentation mask. Another problem occurs when the object color does not differ much from the background at the same position. In this case, even with a correct mosaic, part of the object will be missing in the segmentation mask. Finally, when there are several moving objects in the sequence, the segmentation mask will contain all those objects even if some of them are of no importance in a specific application. These may be small variations of the background like clocks or wiggling trees or even large objects that we want to disregard. Both cases, adding missing parts and excluding parts from the segmentation mask, cannot be accomplished without further high-level information about the object's appearance.

In a second step (B), an abstract model of the object is searched for in the input image to determine approximately the area the object covers. We are applying a graph-based object representation, where homogeneously colored regions are the graph-nodes and graph-edges indicate spatial proximity. A graph-based representation was chosen to allow for articulated object motion. Since the object-model is based on high-level knowledge about the object structure, it cannot be constructed automatically. For this reason, the object-model is generated manually using a graphical editor. The object-model is a general description that can be used throughout the sequence or even for multiple sequences when the same object appears again, thereby keeping user intervention low. A matching algorithm searches for the most likely position of the model in an input frame. The matching uses information about the region color and shape from the model, to search for a compatible location in the current input image. Additional information from the motion-based segmentation mask is integrated to help the matching lock to the moving object. Since the motion segmentation provides us with a good first guess about the object position and object shape, the model can be fitted even when the scene content is difficult. The output of this step is an indicator of object location which does not provide exact object boundaries, but which covers the whole object including parts that are non-moving.

In a last step (C), spatial color-segmentation is employed to obtain exact object boundaries. Here, a region-merging algorithm is used, starting on the regions obtained from a watershed[5] pre-segmentation. Usually, region-merging is based solely on color information. The difficulty with this is that real objects are mostly not uniformly colored. Color variations inside the object may even be much larger than color differences between the object and its background. This makes it impossible to find a single threshold when color segmentation should stop (Fig. 5a). Hence, it is important to provide an approximate indication of where the object is to the spatial segmentation. In our algorithm, this object-location indicator is taken from the object-model matching step and it is integrated into the region-merging criterion. The criterion favors segmentation of regions inside of the object and prohibits merging of regions crossing the object/background boundary. Finally, regions which are covered by the object-location indicators are combined to give the final object segmentation-mask.

3. TEMPORAL SEGMENTATION

The purpose of temporal segmentation is to distinguish static regions in the image from moving areas. This can be either done in a short-term perspective by considering two or three successive frames, and a long-term perspective by reconstructing a background which only contains the static parts of the sequence and differencing to this background. Short-term changes are detected easily by computing difference-frames between successive frames. This simple approach has the difficulty that it cannot distinguish between covered areas and uncovered areas. Moreover, since change detection masks always consider neighbouring frames, only the borders around moving object regions can be found in the change detection mask (CDM). Detection of long-term changes requires the reconstruction of the background, which is not possible in all cases. Often, sequences are too short and do not provide sufficient information for the complete background. However, the background-subtraction masks are usually more accurate than the masks obtained from short-term change detection.

Our algorithm can operate with both kinds of moving area detection. However, we prefer to use long-term motion detection, since it provides better estimates of moving regions and because possible errors are usually compensated easily in the subsequent steps.

3.1. Background Subtraction

The static scene background is reconstructed by applying a median filter on the luminance channel for every pixel in the temporal direction. This effectively results in an image of the background if each part of the image shows background most of the time. Clearly, if there is camera motion, this has to be compensated before applying the temporal filter. The amount of memory needed to store the input frames can be reduced by only considering every nth frame. Since object motion is usually not very fast, objects overlap in subsequent frames and subsequent frames will thus not provide much more background information than a single frame taken from a short time period.

The motion segmentation masks are obtained by taking the absolute difference between background and current input frame. Later, we will refer to this difference frame as $m(x, y) \in [0; 255]$. Since this difference can be used directly in the model-matching step, it is not required to binarize the motion mask.

3.2. Camera-Motion Compensation

If the input sequence was recorded with a moving camera, it is required to compensate this camera motion. Since the total field of view will now be larger than the input resolution, a background mosaic must be computed, which combines all the background parts seen throughout the sequence (Fig. 6a). For this task, we are applying a robust estimator on the perspective motion model

$$x' = \frac{a_{00}x + a_{01}y + t_x}{p_x x + p_y y + 1}, \quad y' = \frac{a_{10}x + a_{11}y + t_y}{p_x x + p_y y + 1}. \tag{1}$$

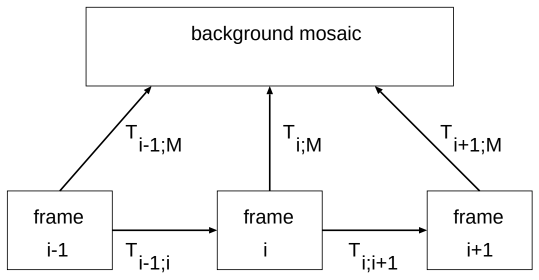


Figure 2. Illustration for predicting the motion-model $T_{i+1;M}$.

Let us denote the transformation from frame i's coordinate system to frame $i+1$ as $T_{i;i+1}((x;y)) = (x';y')$ and the transformation of frame i into the background mosaic as $T_{i;M}$ (see Fig. 2). The parameters are determined using a Levenberg-Marquardt gradient-descent algorithm, minimizing the motion-compensated frame difference. To prevent error accumulation, motion parameters are always computed for the transformation between current input and the mosaic created so far. Clearly, it is necessary to provide a good initialization to the gradient-descent algorithm to ensure convergence to the correct minimum. When using the last transformation $T_{i;M}$ as a starting point for the next frame transformation $T_{i+1;M}$, the algorithm may still converge to a local, suboptimal solution if there is fast camera motion.

For this reason, we assume that motion is continuous and start optimization with the prediction $\hat{T}_{i+1;M} = T_{i;M} \cdot T^{-1}_{i;i+1}$, where the operator ($\cdot$) denotes concatenation of transforms. The transform $T^{-1}_{i;i+1}$ is not directly known, but because of the continuous motion assumption, we have $T_{i;i+1} = T_{i-1;i}$. Together with $T_{i-1;i} = T^{-1}_{i,M} \cdot T_{i-1,M}$, we obtain the motion-model prediction

$$\hat{T}_{i+1;M} = T_{i;M} \cdot T^{-1}_{i-1;M} \cdot T_{i;M}. \tag{2}$$

When building the mosaic, new pixels should only be added to the mosaic at coordinates that are currently undefined. This reduces error accumulation that would otherwise occur if the mosaic was overwritten with the current input frame in each step. Moreover, when applying the temporal median filter, it should be considered that not all mosaic pixels will be defined for each time step.

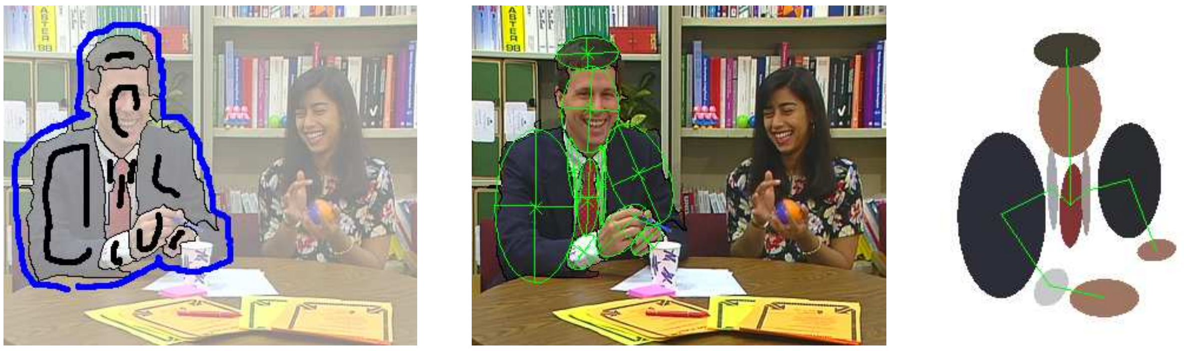
4. MODEL MATCHING

Object-models describe the appearance of articulated objects independent from a special realization in an image. They define the geometric structure of the object, specifying the colors of the main regions and the neighbourhood relationships. In our approach, we are representing object-models with graphs $G_M = (V_M, E_M)$. Each graph node $v \in V_M$ represents one object region with uniform color, while neighbouring regions are connected with a graph edge $e \in E_M$. Note that the model only contains those edges that are semantically necessary for the specific object. For a model of a human, for example, the head is directly connected to the body, but not to one of the arms even though this may be the case in a specific image. Model graph nodes are attributed with the region's mean color and an ellipse specifying the approximate size and shape of the region. Figure 3c shows an example object-model.

Since general sub-graph matching is an NP problem, we introduce the restriction that model graphs must be trees. This reduces computational cost in the model-detection process to polynomial complexity. This does not impose strong practical limitations as most real-world objects can be described by graphs without any circular structures. Even if a natural representation would suggest to use cycles, we can still create object-models by omitting an arbitrary edge, thereby breaking the cycle.

4.1. Model Editor

Defining object-models is an interactive task since this is the step where semantic knowledge is introduced. To ease the task of defining models, we are using a graphical editor to specify object-models. The editor presents



(a) Manually placed watershed markers and resulting segmentation.

(b) Ellipses, fitted to region boundaries.

(c) Final object-model.

Figure 3. Manual object-model creation in the model editor.

the image of a typical image of the object to the user. He specifies the different regions comprising the object by using a manual color-segmentation algorithm, and he also specifies edges between regions, if they must lie close to each other in order to be valid.

Manual segmentation of the object regions is based on a marker-driven watershed algorithm. The difference to the standard watershed technique is that water only starts to flow from markers supplied by the user. Hence, one region is created for each marker set into the image. Region borders will be established at the location where different water flows will join. This will be the case at the position with the largest color changes. Our implementation allows placing several markers that are considered to belong to the same image region (Fig. 3a). This makes it easy to define arbitrarily complex regions even in difficult source material.

For each region defined, an ellipse is fitted to the region border. This abstraction is a good approximation to most region shapes and it still allows easy processing. Each ellipse is further attributed with its mean color of the corresponding region. We denote the color assigned to ellipse $e^{(i)}$ as $e_r^{(i)}, e_g^{(i)}, e_b^{(i)}$ when referring to its color components in RGB space.

4.2. Region-Shape Extraction

The region border resulting from the manual segmentation process is approximated by an ellipse to simplify the subsequent matching step. Our implementation uses two different representations for ellipses in different parts of the algorithm, since each may be more suitable in a specific context. One representation is the *explicit* form, where an ellipse is specified by its center $\vec{c}$ and its two principal axes $\vec{a}_1, \vec{a}_2$. However, ellipse fitting starts with the *implicit* form

$$\begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} A & B & D \\ B & C & E \\ D & E & F \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0. \tag{3}$$

Formulas for converting a representation to the alternative form can be found in the Appendix.

The fitting process runs in two steps. The first step uses algebraic minimization to obtain a first estimate. This is a very fast approach, which on the other hand does not always yield exactly the expected solution, since a semantically meaningless, algebraic residual is minimized. Hence, in a second step, we further refine the solution using a gradient-descent approach to minimize Euclidean distances.

Fitting starts with the implicit ellipse representation where we apply the normalization $A+C=1$ to avoid the trivial solution (see [6]). To solve for the parameters, an equation system is constructed by enumerating all pixels

$(x_i; y_i)$ on the region boundary and appending one equation for each pixel. This results in the overdetermined system

$$\begin{pmatrix} x_0^2 - y_0^2 & 2x_0y_0 & 2x_0 & 2y_0 & 1 \\ x_1^2 - y_1^2 & 2x_1y_1 & 2x_1 & 2y_1 & 1 \\ x_2^2 - y_2^2 & 2x_2y_2 & 2x_2 & 2y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} A \\ B \\ D \\ E \\ F \end{pmatrix} = \begin{pmatrix} -y_0^2 \\ -y_1^2 \\ -y_2^2 \\ \vdots \end{pmatrix}, \tag{4}$$

which is solved in the least squares sense, using Singular Value Decomposition. Since this algebraic optimization may produce inexact results, the parameters are refined in a subsequent gradient-descent process minimizing an approximation to Euclidean distance. We use a symmetric distance measure that consists of the sum of both, region to ellipse distances $d_{r\to e}$, and the ellipse to region distances $d_{e\to r}$:

$$d_{eucl} = d_{r\to e} + d_{e\to r}. \tag{5}$$

The difference between both is that in the first case, the region border is sampled and the corresponding minimum distances to the ellipse are calculated. In the second case, the ellipse is sampled and nearest region-border pixels are being searched for. This symmetric distance results in better region-shape approximations.

For the region-to-ellipse distance, we use an approximation to the point-to-ellipse distance, which is the distance of the point to the ellipse, measured along the ray from the center to the point. This enables the calculation of the region to ellipse distance in closed form as

$$d_{r\to e} = \frac{1}{|border|} \sum_{\vec{p}\in border} \left\| \vec{p} - \vec{c} - \underbrace{\frac{(\vec{p}-\vec{c})}{\|\mathbf{M}^{-1}(\vec{p}-\vec{c})\|}}_{\substack{\text{distance } \vec{c} \text{ to intersection} \\ \text{of ray } \vec{p}-\vec{c} \text{ with ellipse}}} \right\|, \tag{6}$$

with $\mathbf{M} = (\vec{a}_1\ \vec{a}_2)$ being a matrix consisting of the axes of the ellipse and $\vec{c}$ being its center. The point $\vec{p}$ iterates through all pixels on the region border. Distance $d_{e\to r}$ is computed by sampling the ellipse with a sufficient number of points and each time searching for the nearest region-boundary pixel. Hence, the ellipse-to-region distance computes as

$$d_{e\to r} = \frac{1}{N} \sum_{n=0}^{N-1} \min_{\vec{p}\in border} \left\| \vec{p} - \vec{a}_1 \cos\frac{2\pi n}{N} - \vec{a}_2 \sin\frac{2\pi n}{N} \right\|. \tag{7}$$

An example result of this ellipse fitting process is shown in Fig. 3b.

4.3. Model Detection

The purpose of model detection is to find the position of the object-model in an input frame that comes closest to the model. The distance of a specific configuration to the model is defined through node and edge costs. The node cost evaluates the difference between the color of the model node and the mean color of the area covered by a possible model-ellipse position. Additionally, the node cost is reduced if the covered area contains moving image content. This property gives the algorithm a preference for locking on moving objects, which will most probably include the actual object we are searching for. Edge costs evaluate the distance between areas that are connected in the object model. The larger the distance between these areas, the higher the edge cost.

Let $e^{(i)}(x, y, \alpha)$ denote the set of pixels contained in the ellipse $e^{(i)}$, which is shifted so that its center lies at $(x; y)$ and which is additionally tilted by an angle α. We define the node matching cost for ellipse i at position x, y and angle α as

$$s^{(i)}(x,y,\alpha) = \underbrace{\left\| \frac{1}{|e^{(i)}|} \left[\sum_{(x';y')\in e^{(i)}(x,y,\alpha)} \begin{pmatrix} f_r(x',y') \\ f_g(x',y') \\ f_b(x',y') \end{pmatrix} \right] - \begin{pmatrix} e_r^{(i)} \\ e_g^{(i)} \\ e_b^{(i)} \end{pmatrix} \right\|}_{\text{color matching cost}} - \underbrace{\frac{\gamma}{|e^{(i)}|} \sum_{(x';y')\in e^{(i)}(x,y,\alpha)} m(x',y')}_{\text{motion area bonus}}, \tag{8}$$

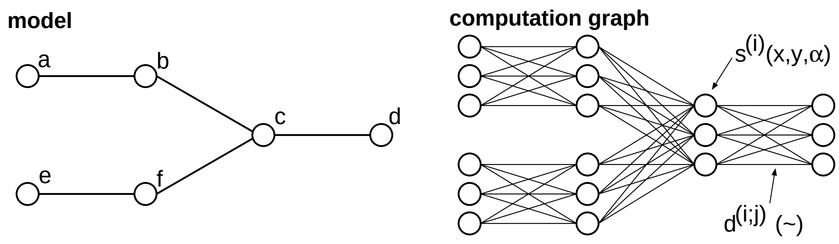


Figure 4. Dynamic-programming computation-graph example for a simple model graph.

where the first term measures the color difference and the second term reduces the cost for moving areas. The parameter γ is a fixed constant. Note that the sum over all pixels in the ellipse area can be computed very efficiently using the scheme presented in the Appendix. An example error-map $s(x, y, \alpha)$ is shown in Figure 5c.

Edge cost is defined as the distance between two ellipses $e^{(i)}, e^{(j)}$. This distance is approximated in a similar approach as in Equation (6):

$$d^{(i,j)}(x_i, y_i, \alpha_i, x_j, y_j, \alpha_j) = \left| \left\| \vec{c}^{(i)} - \vec{c}^{(j)} \right\| - \left\| \frac{(\vec{c}^{(i)} - \vec{c}^{(j)})}{\|\mathbf{M}^{(\mathbf{i})^{-1}}(\vec{c}^{(i)} - \vec{c}^{(j)})\|} \right\| - \left\| \frac{(\vec{c}^{(i)} - \vec{c}^{(j)})}{\|\mathbf{M}^{(\mathbf{j})^{-1}}(\vec{c}^{(i)} - \vec{c}^{(j)})\|} \right\| \right|. \tag{9}$$

The best matching location of the object-model in an input image is determined as the set of ellipse locations $\{x_i\}, \{y_i\}, \{\alpha_i\}$ that minimizes the following expression:

$$\min_{\{x_i\},\{y_i\},\{\alpha_i\}} \sum_{k \in V_M} s^{(k)}(x_k, y_k, \alpha_k) + \sum_{(k,l) \in E_M} d^{(k,l)}(x_k, y_k, \alpha_k, x_l, y_l, \alpha_l). \tag{10}$$

If E_M has tree structure, a dynamic programming approach can be used for an efficient computation of the optimum (see reference [7] for a detailed description of the algorithm). The basic approach is to start the calculation at the leaves of the tree and proceed to the (arbitrarily chosen) root. For each node, a set of candidate positions is evaluated where only the edge costs to the direct predecessor nodes have to be considered (Fig. 4). In order to avoid high computational complexity, not all possible positions are included in the computation graph for dynamic programming. Instead, a set of about $n = 30$ candidates is selected according to the following process. First, all positions which are no local minimum are excluded. The remaining positions are sorted according to increasing matching costs. The best n positions are selected as candidates provided that there is no position with lower cost in the vicinity ($\|(x_0, y_0)^T - (x_1, y_1)^T\| < thresh_d$ and $|\alpha_0 - \alpha_1| < thresh_\alpha$), and the cost $s(x, y, \alpha)$ does not exceed a threshold. If it does, less than n positions are selected as candidates.

5. SPATIAL SEGMENTATION

The automatic spatial segmentation stage uses two sources of information. The first source is color information from the current input frame while the second information comes from the fitted model-location in form of the ellipse parameters. This high-level knowledge about the approximate object location helps to control the spatial segmentation such that areas attributed as foreground will not be merged with background areas even if color differences are small.

5.1. Spatial Segmentation Algorithm

We apply a two-step approach for spatial segmentation. First, a watershed pre-segmentation is applied. This is a fast algorithm which usually results in heavy oversegmentation. The reason for applying this step is that it speeds up the following region-merging algorithm which can now start with the regions obtained from the watershed presegmentation instead of starting at the pixel level.

The region-merging algorithm gradually merges the two most similar, neighbouring regions together.[8] To do so efficiently, a heap data-structure is used to store all dissimilarity scores of neighbouring regions. After two regions have been merged, the dissimilarity scores to all neighbours have to be recomputed.

5.2. Merging Criterion

The merging criterion for evaluating region dissimilarity is composed of two terms. The first is the *Ward*-criterion, which minimizes the variance of the luminance component in a region. More clearly, the first term in Equation (11) equals the increase of variance if two regions r_i and r_j would be merged. This can be computed efficiently by keeping track of the mean region luminances μ_i, and μ_j and their sizes.

$$S(r_i, r_j) = \underbrace{\frac{|r_i| \cdot |r_j|}{|r_i| + |r_j|}(\mu_i - \mu_j)^2}_{\text{Ward}} \cdot \underbrace{\left(\max_k \frac{|r_i \cap e^{(k)}|}{|r_i|} \cdot \frac{|r_j \cap e^{(k)}|}{|r_j|} + \beta\right)^{-1}}_{\text{penalty for crossing object/background border}} \tag{11}$$

The second term increases merging cost if one or both regions are not covered by one of the object-model ellipses. This inhibits merging of regions outside of the object. The parameter $\beta > 0$ controls the influence of the second term. Since $r_i \cap r_j = \emptyset$ for $i \neq j$, it holds that

$$\frac{|(r_i \cup r_j) \cap e^{(k)}|}{|r_i \cup r_j|} = \frac{|r_i \cap e^{(k)}| + |r_j \cap e^{(k)}|}{|r_i| + |r_j|} \tag{12}$$

and an updated $S(r_i, r_j)$ can be computed efficiently after each merging step by keeping track of attributes $|r_i \cap e^{(k)}|$ for each region r_i. Region merging stops as soon as the minimum $S(r_i, r_j)$ exceeds a threshold.

The final object mask is obtained by joining all regions for which at least 50% of their area is covered by an object-model ellipse. Small holes in the masks that can result from non-overlapping model ellipses are closed in a final post-processing step.

6. EXPERIMENTS AND RESULTS

This section demonstrates the performance of our segmentation system on two typical input sequences. The first example is taken from the *paris* sequence. This is a head-and-shoulder sequence without any camera motion. The two main difficulties with this example are that the body of the man at the left does move much, which makes it undetectable for the motion-segmentation, and secondly, the small color difference between the hair of the man and the background. Because of this small difference, a color-based segmentation without any semantic knowledge usually results in a wrong segmentation (Fig. 5a), merging hair and background into the same region. There are more errors which cannot be seen so clearly: part of the arm at the left was merged with part of the chair, dark parts in his hand were merged with shadow, and so on.

We have edited a model of the man (see Fig. 3) and applied the algorithm to frame 20. The found model is shown superimposed to the input frame in Figure 5e. The model is placed at a sensible configuration, but the object is still not completely covered by the model, since the use of ellipses as region models does not fit exactly the region shape. Figure 5f shows the result after color-segmentation when the object-model location is considered in the merging criterion. Almost the complete object is covered in the segmentation mask. Small parts of the head are missing as these areas were not covered by the model. On the other hand, a small part of the background was added to the mask because the ellipse covering the jacket region is slightly too large.

The second example is taken from the *stefan* sequence, showing a tennis-player with strong camera-motion. Since the object is small compared to the background and the object is also moving, the background image can be reconstructed without error (Fig. 6a). Nevertheless, the long-term change detection mask (Fig. 6b) does not give a clear result. Three problems can be identified:

- not all parts of the background vanish in the difference frame, because of small background variations or minor errors in the motion compensation,
- parts of the tennis-player are lost, since they have coincidentally the same color as the background, and
- the mask contains both the tennis-player and the tennis-ball, while sometimes one of them may not be needed.

Again, we constructed a model of the tennis-player and applied the segmentation algorithm on frame 177 of the sequence. The result shows that the tennis-player is found, but it also still has inaccuracies, especially at parts where the object has some fine texture. We also supplied the algorithm with a model of the tennis-ball, which is simply a single graph-node. The result is shown in Fig. 6e. Consequently, our segmentation system allows to selectively extract objects from a video-sequence, depending on the object-model that is applied to it.

7. CONCLUSIONS

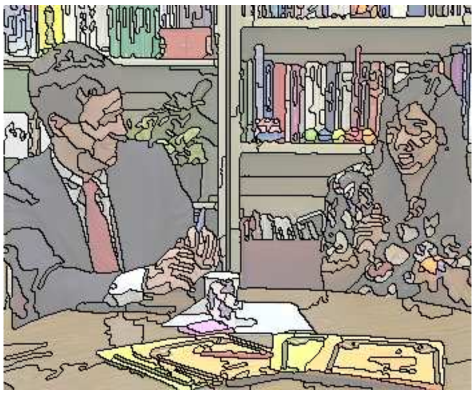
This paper has presented a segmentation system that combines motion-segmentation, spatial-segmentation, and model-based object detection into a single framework. Motion-segmentation is used for an approximate localization of the object position. The object detection fits a manually defined object-model to the current input frame to cover the complete area of the modelled object. Spatial segmentation is used to refine the object boundary and to generate accurate segmentation masks.

The algorithm has the advantage over previous techniques in the sense that it does not solely rely on motion or spatial information to decide what the object should be. Instead, the user can specify the exact object he wants to extract without doing the segmentation manually. A further advantage is that cases, in which only part of the object is moving or cases in which the object is not clearly distinguishable from the background, can be solved through the combination of several features.

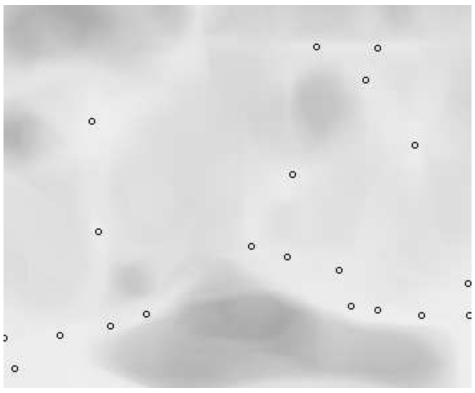
Problems with the current approach mostly become visible at areas close to the region boundary, where the region-shape cannot be approximated well using ellipses. Therefore, future research should consider other representations for region shapes. Another improvement will be the inclusion of textured regions in the object model. Finally, in some cases it would be advantageous to include ordering constraints into the matching process to disambiguate situations where the spatial ordering is known. For example, in a human model, we know that the head will be above the body. However, in most cases this would introduce additional graph-edges which will violate the tree-structure limitation of our graph matching algorithm. Thus, in the future, we would like to relieve from the limitations of special graph structures and apply a general graph matching algorithm. However, it has still to be explored, which algorithm provides a good compromise between accuracy and computational complexity.

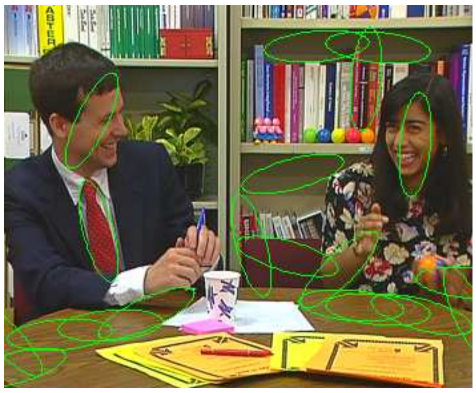
REFERENCES

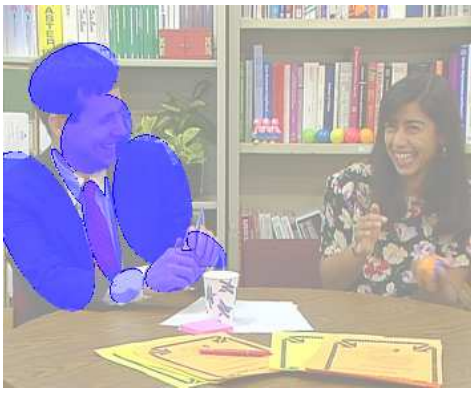
1. R. Fablet, P. Bouthemy, and M. Gelgon, “Moving object detection in color image sequences using region-level graph labeling,” in 6th *IEEE International Conference on Image Processing, ICIP*, Oct. 1999.
2. I. Patras, E. A. Hendriks, and R. L. Lagendijk, “Video segmentation by MAP labeling of watershed segments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(3), pp. 326–332, 2001.
3. A. Alatan, L. Onural, M. Wollborn, R. Mech, E. Tuncel, and T. Sikora, “Image sequence analysis for emerging interactive multimedia services - the european cost 211 framework,” *IEEE Transactions on Circuits and Systems for Video Technology* **8**, pp. 802–813, Nov. 1998.
4. R. Mech and M. Wollborn, “A noise robust method for segmentation of moving objects in video sequences,” in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP'97*, pp. 2657–2660, Apr. 1997.
5. L. Vincent and P. Soile, “Watersheds in digital spaces: an efficient algorithm based on immersion simulations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(6), pp. 583–597, 1991.
6. Z. Zhang, “Parameter estimation techniques: A tutorial with application to conic fitting,” Tech. Rep. RR-2676, INRIA, Oct. 1995.
7. D. Farin, P. H. N. de With, and W. Effelsberg, “Recognition of user-defined video object models using weighted graph homomorphisms,” in *SPIE Image and Video Communications and Processing, 5022*, Jan. 2003.
8. D. Farin and P. H. N. de With, “Towards real-time MPEG-4 segmentation: a fast implementation of region merging,” in 21st *Symposium on Information Theory in the Benelux*, pp. 173–180, 2000.
9. M. Pelillo, K. Siddiqi, and S. W. Zucker, “Continuous-based heuristics for graph and tree isomorphisms, with application to computer vision,” in *NIPS 99 Workshop on Complexity and Neural Computation*, Dec. 1999.
10. D. Eppstein, “Subgraph isomorphism in planar graphs and related problems,” tech. rep., Dept. of Information and Computer Science, University of California, May 1994.


(a) Color only segmentation. Note the undersegmentation at the head.

(b) Short-term CDM.


(c) Node matching cost for the region corresponding to the tie. Brighter means lower cost. Candidate positions are marked.


(d) Candidate configurations for the tie region.


(e) Fitted object model.

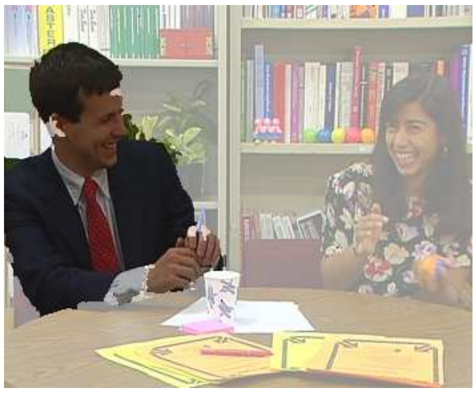
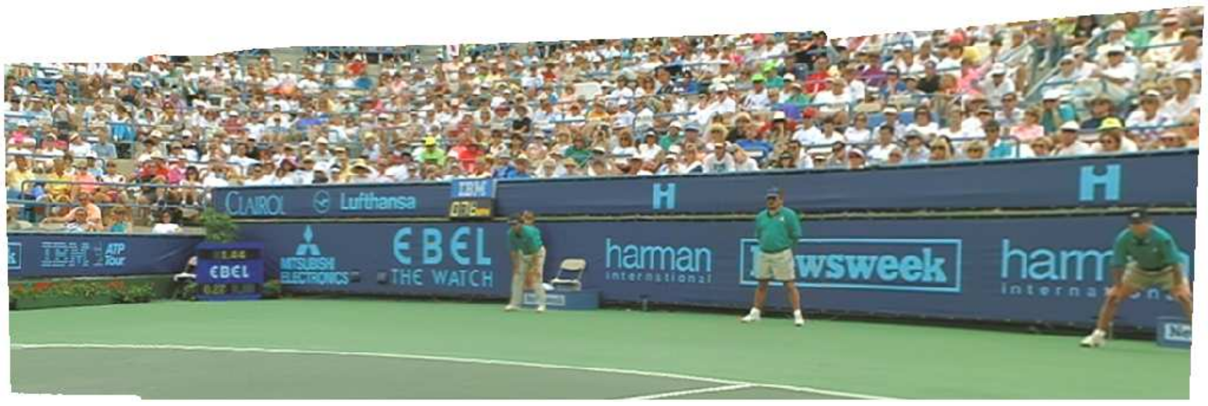
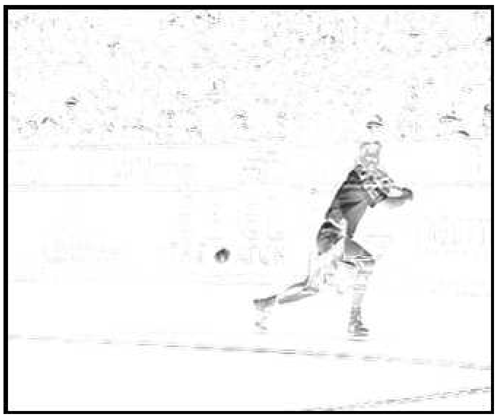

(f) Final segmentation mask.

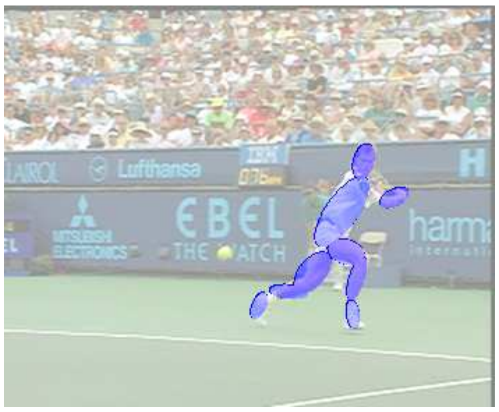
Figure 5. Frame 20 of the *paris* sequence.



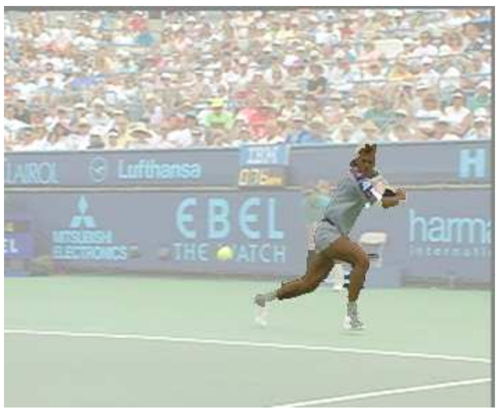
(a) Reconstructed background.



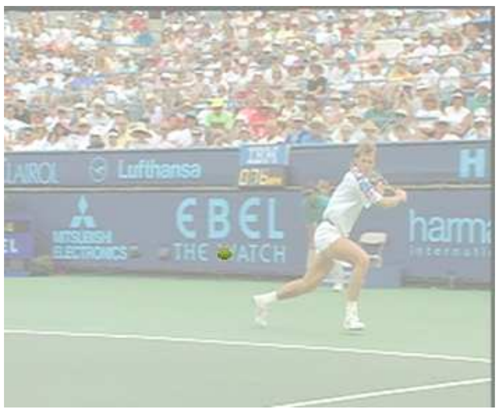
(b) Long-term CDM.



(c) Fitted object model (tennis-player).



(d) Final segmentation (tennis-player).



(e) Final segmentation (tennis-ball).

Figure 6. Frame 177 of the *stefan* sequence.

APPENDIX

Converting from implicit to explicit form

The conversion is carried out in three steps. First, we determine the conic center position. In a second step, the implicit parameters are modified such that the ellipse is shifted to the origin. Finally, the axes are obtained using Eigenvector analysis. Given a conic with parameters (A, B, C, D, E, F), its center $(c_x; c_y)$ can be determined by

$$c_y = \frac{A \cdot E - D \cdot B}{B^2 - A \cdot C}, \qquad c_x = -\frac{D + B \cdot c_y}{A}. \tag{13}$$

Now, the parameters (A', B', C', D', E', F') for a conic shifted to the origin can be obtained using

$$A' = A, \quad B' = B, \quad C' = C, \quad D' = E' = 0, \quad \text{and} \quad F' = F - (Ac_x^2 + 2Bc_xc_y + Cc_y^2). \tag{14}$$

This gives us the zero centered conic equation $\left(x\ y\right) \mathbf{Q} \left(x\ y\right)^T = 1$ with

$$\mathbf{Q} = -\frac{1}{F'} \begin{pmatrix} A & B \\ B & C \end{pmatrix}. \tag{15}$$

By determining the Eigenvectors of $\mathbf{Q}$ and scaling according to their Eigenvalues, we get the principal axes $\vec{a}_1, \vec{a}_2$. Hence, we have the explicit ellipse parameters $\vec{c} = (c_x\ c_y)^T$, $\vec{a}_1$, and $\vec{a}_2$.

Converting from explicit to implicit form

Assume that the ellipse is given in explicit form with center $(c_x; c_y)$, tilt angle α and lengths of principal axes a_1, a_2. To get the implicit form, we start with

$$\left(\frac{x'}{a_1}\right)^2 + \left(\frac{y'}{a_2}\right)^2 = 1 \tag{16}$$

for an ellipse whose axes are aligned to the coordinate axes. After translating and rotating the coordinate system by

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x - c_x \\ y - c_y \end{pmatrix}, \tag{17}$$

we finally obtain

$$\begin{aligned}
A &= a_2^2 \cos^2\alpha + a_1^2 \sin^2\alpha \\
C &= a_2^2 \sin^2\alpha + a_1^2 \cos^2\alpha \\
B &= (a_1^2 - a_2^2)\cos\alpha \sin\alpha \\
D &= a_2^2 \cos\alpha(-c_x\cos\alpha + c_y\sin\alpha) - a_1^2 \sin\alpha(c_x\sin\alpha + c_y\cos\alpha) \\
E &= a_2^2 \sin\alpha(c_x\cos\alpha - c_y\sin\alpha) - a_1^2 \cos\alpha(c_x\sin\alpha + c_y\cos\alpha) \\
F &= a_2^2(c_x\cos\alpha - c_y\sin\alpha)^2 + a_1^2(c_x\sin\alpha + c_y\cos\alpha)^2 - a_1^2a_2^2.
\end{aligned} \tag{18}$$

Efficient computation of the sum over an elliptical area

Assume that we want to calculate the sum of $f(x, y)$ over the area inside an ellipse, given in implicit form. In a pre-computation step that has only to be done once for every $f(x, y)$ and which is independent of the ellipse, we compute

$$F(x, y) = \sum_{i=0}^{x} f(i, y). \tag{19}$$

Now, the sum over part of a single line can be computed in constant time as $\sum_{x=a}^{b} f(x, y) = F(b, y) - F(a - 1, y)$. To sum over the ellipse area, we proceed line by line, computing the horizontal range $[x_{min}; x_{max}]$ on each scanline. Using Equation (3), we obtain for a fixed y

$$x_{min}, x_{max} = -\frac{B + D}{A} \mp \sqrt{\left(\frac{B + D}{A}\right)^2 - \frac{F + Cy^2 + 2Ey}{A}}. \tag{20}$$