The function of the expert is not
to be more right than other people,
but to be wrong for more sophisticated reasons.
(David Butler)

Chapter 10

Object Detection based on Graph Models II: Natural

This chapter presents an algorithm for video-object segmentation that combines motion information, a high-level object-model detection, and spatial segmentation into a single framework. This joint approach overcomes the disadvantages of these algorithms when they are applied independently. These disadvantages include the low semantic accuracy of spatial color segmentation, the inexact object boundaries obtained from object-model matching and the often incomplete motion information. The described algorithm alleviates three of the problems that we encountered in the segmentation system that was described in the first part of the thesis. First, it completes object areas that cannot be clearly distinguished from the background because their color is similar to the background color. Second, parts of the object that were not extracted because they are not moving, are now 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 fit to any object model and these regions are removed from the foreground. This suppresses regions that are considered erroneously as moving, or objects that are moving but irrelevant to the user.

10.1 Introduction

Segmentation-techniques can be coarsely classified into spatial and temporal segmentation. Spatial segmentation includes segmentation based on texture or simply color. Color segmentation provides accurate region boundaries, but since it is working at 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. The background-subtraction technique that we used as the core of the segmentation system described in the first part of the thesis, is also a temporal segmentation approach, since it detects only the moving areas. Unfortunately, temporal segmentation does not always include the complete objects. We have observed for example, that head-and-shoulder sequences like they are common in video conferencing or news report scenes, do not yield good results because large parts of the objects are not moving.

One approach to solve this problem is to combine spatial segmentation and motion information into a joint segmentation framework. Several algorithms following this approach have been proposed in the literature. Fablet *et al.* [45] propose to apply a neighbourhood graph over spatially segmented image-regions. A Markovian framework is used to label the region into foreground and background according to their motion. However, this approach only differentiates between regions moving compatible with dominant motion, and regions that do not. Hence, when only part of the object is moving, the segmentation algorithm will not cover the complete object. A comparable algorithm that labels the regions into several consistent motion classes has been proposed by Patras *et al.* [141]. The approach presented by Alatan *et al.* in [6] obtains motion information in the form of change-detection masks [124]. 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 motion and spatial information are still fused 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. Consider for example 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 that applies an additional model-based

Technique	Advantage	Disadvantage	Example
Spatial segmentation	Accurate region boundaries	Weak semantic meaning	
Temporal segmentation	Good object detection	Does not work with static objects	
Model-based	Provides high-level semantics	Inaccurate object boundaries	

Table 10.1: *Advantages and disadvantages of various segmentation techniques.*

approach to combine all three segmentation approaches (Table 10.1). The object model enables the algorithm to find the complete object even when parts of it do not move, and using the model, it can lock to a specific object even when several objects are moving at the same time. We use an object model based on the graph representation that was introduced in the previous chapter. In this chapter, we adapt this model and the matching algorithm to obtain a segmentation framework for natural images.

10.2 Segmentation system architecture

Our segmentation system comprises three main steps, corresponding to the three approaches *spatial*, *temporal*, and *model-based* segmentation. An overview flow-graph of the algorithm is depicted in Figure 10.1.

In the first step, a change-detection algorithm is employed to find the moving regions in the image, giving an approximate location of the object. The change detection can be computed as long-term change detection, relative to a globally reconstructed background image (see Chapter 7). In this case, Step 1 is basically the complete segmentation system as described in Part I. Alternatively, the change detection can also be computed as short-

term change detection between pairs of successive frames. The two most common errors observed in the change detection are incomplete objects, because of insufficient motion, and detection of uninteresting objects (like moving trees in the background). Both classes of errors, missing object parts and uninteresting objects, cannot be removed without further high-level information about the object's appearance.

In the second step, we use an object model to search for the location of the object in the input image. The object model is represented by a graph similar to that described in the previous chapter. The nodes in the graph represent homogeneously colored regions and graph-edges indicate spatial proximity. To generate the object model, we reuse an extended version of the graphical editor described in the previous chapter. The edited 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 the need for 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. Differing from the matching process described in the previous chapter, we now also integrate additional information from the change-detection mask to help the matching lock to the moving object. Since the motion information 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 that covers the whole object including parts that are non-moving.

In a third step, spatial color-segmentation is employed to obtain exact object boundaries. Here, a region-merging algorithm is used (see Appendix E), starting with the regions obtained from a watershed [189] 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 can even be larger than color differences between the object and its background. This makes it impossible to find a single threshold at which color segmentation should stop (Fig. 10.8(a)). Hence, it is important to indicate the object location 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 model are combined to give the final object segmentation-mask.

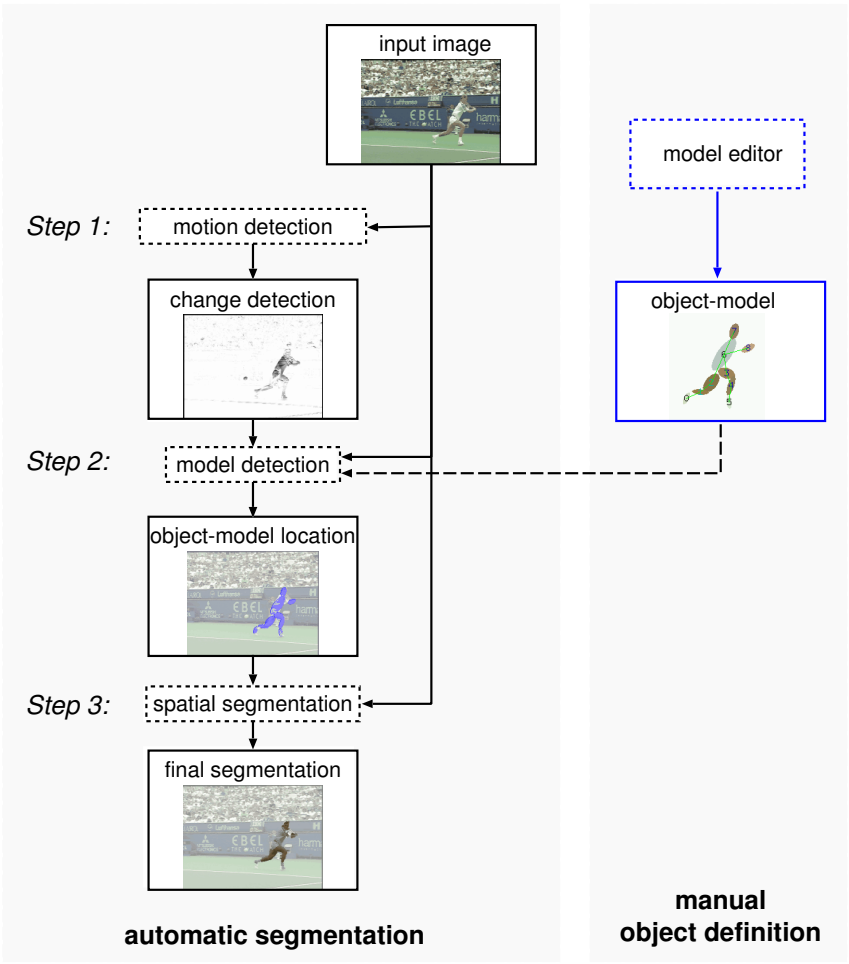


Figure 10.1: *Overview of the segmentation system incorporating object models. First an abstract object model is built manually in a graphical editor (right side). This model is used to help the automatic segmentation to identify the correct object location. The automatic segmentation starts with a change detection to obtain some indication of the object position. Subsequently, the object location is found by fitting the constructed model onto the image using color information and the change-detection mask. Finally, a color segmentation is applied to compute the accurate object boundary.*

10.3 Step 1: motion detection

The purpose of the motion detection is to distinguish static regions in the image from moving areas. This can be done either in a short-term perspective by considering two successive frames, or in a long-term perspective by reconstructing a background which only contains the static parts of the sequence and computing differences to this background.

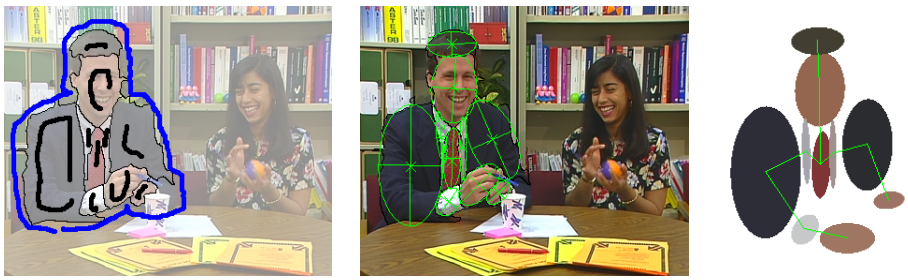
Short-term *change-detection masks* (CDMs) are obtained easily by computing difference-frames between successive frames. If applicable, camera-motion compensation can be applied prior to computing the difference image. This simple approach shows the problem that occluded areas and uncovered areas cannot be distinguished in the mask. Moreover, change-detection masks of neighbouring frames only include the borders around moving object regions.

The alternative approach is to detect long-term changes relative to a static background image. This approach has been extensively studied in the first part of the thesis. The advantage of this approach is that background-subtraction masks are usually more accurate than the masks obtained from short-term change detection. However, for some classes of scenes, the sequences are too short or do not provide sufficient information for the reconstruction of the complete background image. Our algorithm can operate with both kinds of motion-area detection. However, we prefer to use long-term motion detection, since it provides better estimates of moving regions.

10.4 Step 2: model matching

Our object model describes the appearance of articulated objects independent from a special realization in an image. The model defines the geometric structure of the object, specifying the colors of the main regions and the neighbourhood relationships. In our approach, we represent 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 $e \in E_M$. In this respect, the object models are similar to the models used in the previous chapter. However, we use a slightly modified set of features for the graph nodes. Additionally to an attribute specifying the region's mean color, we approximate the shape of a region with an ellipse[1]. Figure 10.2(c) shows an example object model. The use of ellipses was motivated by tracking algorithms that use a mixture of multi-variate Gaussians as object observation model [196]. Ellipses allow

[1]These ellipses have already been shown in the visualizations in the last chapter, but they were not actually used in the matching algorithm.



(a) Manually placed watershed markers and the resulting segmentation.

(b) Ellipses, fitted to the region boundaries.

(c) Extracted object model.

Figure 10.2: *Manual object model creation in the model editor.*

to represent compactly the key feature of a compact region such as size, shape aspect-ratio, and orientation.

10.4.1 Model editor

The graphical editor for defining object models is similar to the editor described in the previous chapter. The user places markers in the essential object regions (Fig. 10.2(a)). These markers are used in a watershed algorithm to extract the region boundaries. After each user modification, the segmentation boundaries are recomputed, which makes it very easy for the user to control the segmentation process and correct errors. Additionally to the object region, the user also defines the graph edges between connected regions.

Differing from the editor described in the previous chapter, a different set of features is extracted. First, an ellipse is fitted to the shape of each object region. 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.

Ellipse fitting to region-shape

To obtain the model parameters, the region border resulting from the manual segmentation process, has to be approximated by an ellipse. 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

$$\begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} A & B & D \\ B & C & E \\ D & E & F \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0. \tag{10.1}$$

Formulas for converting a representation to the alternative form can be found in an appendix section to this chapter (Section 10.8).

The parameter estimation to determine the ellipse parameters $A, \ldots, F$ of Eq. (10.1) proceeds in two steps. The first step uses algebraic minimization to obtain a first estimate. This is a very fast approach, but it does not always yield the expected solution, because 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.

Linear estimation

The first step applies an algebraic fitting using the implicit ellipse representation, where we apply the normalization $A + C = 1$ to avoid the trivial solution (see [200]). 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

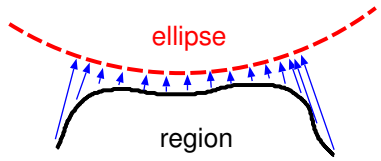
$$\begin{pmatrix} x_0^2 - y_0^2 & 2x_0y_0 & 2x_0 & 2y_0 & 1 \\ x_1^2 - y_1^2 & 2x_1y_1 & 2x_1 & 2y_1 & 1 \\ x_2^2 - y_2^2 & 2x_2y_2 & 2x_2 & 2y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} A \\ B \\ D \\ E \\ F \end{pmatrix} = \begin{pmatrix} -y_0^2 \\ -y_1^2 \\ -y_2^2 \\ \vdots \end{pmatrix}, \tag{10.2}$$

which is solved in the least-squares sense, using a Singular Value Decomposition.

Nonlinear estimation

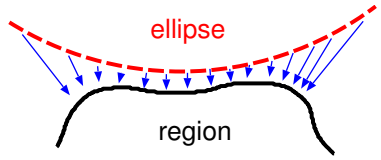
Since this algebraic optimization may produce inexact results, the parameters are refined with a subsequent gradient-descent process, minimizing the geometric distance between the region boundary and the ellipse. We use a symmetric distance measure that consists of the sum of the region-to-ellipse distances $d_{r\to e}$, and the ellipse-to-region distances $d_{e\to r}$:

$$d_{geom} = d_{r\to e} + d_{e\to r}. \tag{10.3}$$



(a) Distance $d_{r\to e}$, measured from region to ellipse.

(b) Example of degenerated fit for $d_{r\to e}$.



(c) Distance $d_{e\to r}$, measured from ellipse to region.

(d) Example of degenerated fit for $d_{e\to r}$.

Figure 10.3: *If only one of the measures $d_{r\to e}$ or $d_{e\to r}$ would be used, the result would include undesired fitting results (b), (d). This can be prevented by combining both distance measures.*

The difference between both distances 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 pixel is being searched for. The symmetric distance d_{geom} results in better region-shape approximations than an approximation with only one asymmetric distance (see Figure 10.3).

For the region-to-ellipse distance, we iterate through all point on the region border and compute the distance of that point to the ellipse. Since the computation of the distance of a point to an ellipse is computationally complex, we use an approximation to the point-to-ellipse distance. We define the approximate point-to-ellipse distance as the distance measured along the ray from the ellipse center $\vec{c}$ to the point $\vec{p}_i$ (Fig. 10.4). To determine this distance, we first compute the vector $\vec{s}-\vec{c}$. Let $\mathbf{M} = (\vec{a}_1\ \vec{a}_2)$ be a matrix consisting of the axes of the ellipse and let $\vec{c}$ be the ellipse center. By transforming the ellipse back to a unit circle using $\mathbf{M}^{-1}$, we easily see that in the back-transformed coordinate system, $\vec{s}-\vec{c}$ must have unit length. Hence, by scaling $\vec{p}_i - \vec{c}$ with the inverse of the $||\vec{p}_i - \vec{c}||$ in the back-transformed coordinate system, the vector $\vec{s} - \vec{c}$ is obtained. Consequently, the approximate distance of a point $\vec{p}_i$ to the ellipse computes as

$$d(\vec{p}_i) = ||(\vec{p}_i - \vec{c}) - (\vec{s} - \vec{c})|| = \left\| \vec{p}_i - \vec{c} - \frac{(\vec{p}_i - \vec{c})}{\|\mathbf{M}^{-1}(\vec{p}_i - \vec{c})\|} \right\|. \qquad (10.4)$$

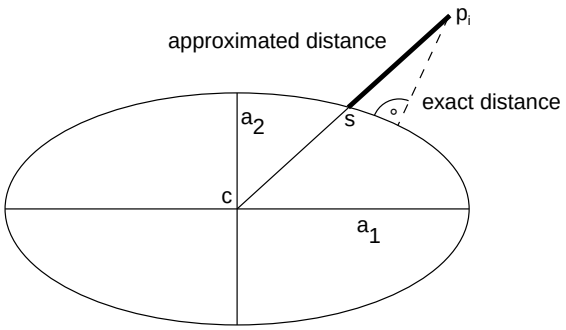


Figure 10.4: *The distance of a point p_i to an ellipse is approximated by the distance along the ray connecting the point with the ellipse center c.*

This enables the calculation of the region-to-ellipse distance in closed form as

$$d_{r\to e} = \frac{1}{|border|} \sum_{\vec{p}_i \in border} d(\vec{p}_i). \tag{10.5}$$

The point $\vec{p}$ iterates through all pixels on the region border.

The 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

$$d_{e\to r} = \frac{1}{N} \sum_{n=0}^{N-1} \min_{\vec{p}\in border} \left\| \vec{p} - \vec{a}_1 \cos\frac{2\pi n}{N} - \vec{a}_2 \sin\frac{2\pi n}{N} \right\|. \tag{10.6}$$

An example result of this ellipse-fitting process is shown in Fig. 10.2(b).

10.4.2 Model detection

The purpose of model detection is to find the position of the object model in an input frame. Remember that in the previous chapter, we applied an automatic segmentation to the input frame to obtain a graph representation similar to the object model. Model detection could then be carried out as a graph-matching problem. This was well possible, since we were concentrating on cartoon sequences for which an automatic color segmentation is easy. However, for natural sequences, we cannot rely on a good color segmentation. Hence, we search for the object model directly in the input image.

The fitting value of a specific configuration of ellipses is again defined through a combination of node and edge costs.

- **Node costs** evaluate the difference between the color of the model node and the mean color of an area in the image. Additionally, the node cost is reduced if the considered image area contains moving image content. This reduction on moving areas gives the algorithm a preference for locking on moving objects, which will most probably include the actual object we are searching for.

- **Edge costs** represent the geometric distance between areas that are connected in the object model. The larger the distance between these areas, the higher the edge cost.

Node costs

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

$$s^{(i)}(x,y,\alpha) = \underbrace{\left\| \frac{1}{|e^{(i)}|} \left[\sum_{(x',y')\in e^{(i)}(x,y,\alpha)} \begin{pmatrix} f_r(x',y') \\ f_g(x',y') \\ f_b(x',y') \end{pmatrix} \right] - \begin{pmatrix} e_r^{(i)} \\ e_g^{(i)} \\ e_b^{(i)} \end{pmatrix} \right\|}_{\text{color matching-cost}} - \underbrace{\frac{\gamma}{|e^{(i)}|} \sum_{(x';y')\in e^{(i)}(x,y,\alpha)} m(x',y')}_{\text{motion-area bonus}}, \tag{10.7}$$

where the first term measures the color difference and the second term reduces the cost for moving areas. The parameter γ is a fixed weighting factor. Note that the sum over all pixels in the ellipse area can be computed very efficiently using the scheme presented in the Section 10.8. An example error-map $s^{(i)}(x, y, \alpha)$ is shown in Figure 10.8(c).

Edge costs

Edge costs are defined as the distance between two ellipses $e^{(i)}, e^{(j)}$. This distance is approximated by the distance along the line connecting the two ellipse centers (Figure 10.5). The approximation can be computed in closed

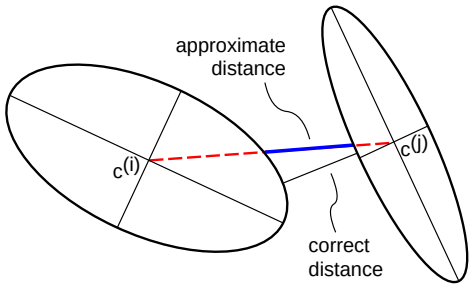


Figure 10.5: *Approximate computation of distance between two ellipses. See Eq. (10.8).*

form by

$$d^{(i,j)}(x_i,y_i,\alpha_i,x_j,y_j,\alpha_j) = \mathrm{abs}\Bigg\{ \underbrace{\left\| \vec{c}^{(i)} - \vec{c}^{(j)} \right\|}_{\text{distance between centers}} - \underbrace{\left\| \frac{(\vec{c}^{(i)} - \vec{c}^{(j)})}{\|\mathbf{M}^{(\mathbf{i})^{-1}}(\vec{c}^{(i)} - \vec{c}^{(j)})\|} \right\|}_{\text{center-to-border distance}} - \underbrace{\left\| \frac{(\vec{c}^{(i)} - \vec{c}^{(j)})}{\|\mathbf{M}^{(\mathbf{j})^{-1}}(\vec{c}^{(i)} - \vec{c}^{(j)})\|} \right\|}_{\text{center-to-border distance}} \Bigg\}. \tag{10.8}$$

Fitting process

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:

$$\min_{\{(x_i,y_i,\alpha_i)\}} \sum_{k \in V_M} \underbrace{s^{(k)}(x_k, y_k, \alpha_k)}_{\text{node cost (ellipse position)}} + \sum_{(m,l) \in E_M} \underbrace{d^{(m,l)}(x_m, y_m, \alpha_m, x_l, y_l, \alpha_l)}_{\text{edge cost (ellipse distances)}}. \tag{10.9}$$

If E_M has a tree structure, a dynamic-programming approach similar to the algorithm described in the previous chapter can be used for an efficient computation of the optimum. This is a generalization of the one-dimensional shortest-path problem to a minimum-cost tree problem (Figure 10.6). In order to avoid high computational complexity, not all possible positions are included in the computation graph for dynamic-programming. Instead, $n \approx 30$ candidate positions are selected according to the following process. First, the costs $s^{(k)}(x_k, y_k, \alpha_k)$ for placing an ellipse k at position (x_k, y_k) with angle α_k are calculated (Fig. 10.8(c)). All local minimum positions

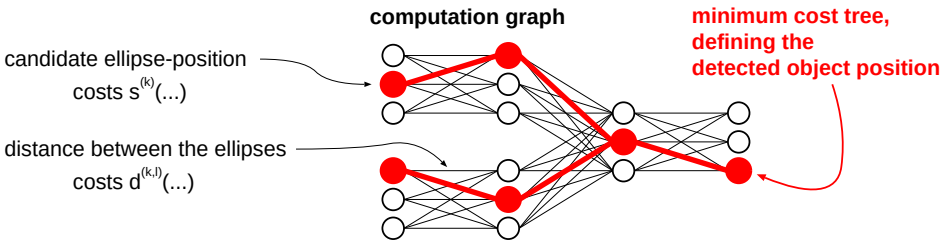


Figure 10.6: *Dynamic-programming computation-graph example for a simple model graph (Fig. 9.9(b)).*

could be potential candidate positions for the ellipse, but to decrease the complexity of the matching process, we only select the n best positions. Positions with a cost exceeding a threshold are not included, which may decrease the number of candidates further to less than n positions. Figure 10.8(d) shows all candidate positions for the ellipse corresponding to the tie in the model of the man. Using the same process, candidate locations for each model region are extracted. Finally, using the dynamic-programming algorithm, the best combination of ellipse locations is computed, resulting in the detection of the object in the input image (Figure 10.8(e)).

10.5 Step 3: spatial segmentation

The automatic spatial segmentation stage uses two sources of information to compute more accurate object boundaries. 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.

10.5.1 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 subsequent region-merging algorithm, which can now start with the regions obtained from the watershed pre-segmentation instead of starting at the pixel level. The region-merging algorithm gradually merges the two most similar, neighbouring regions together (see Appendix E for a

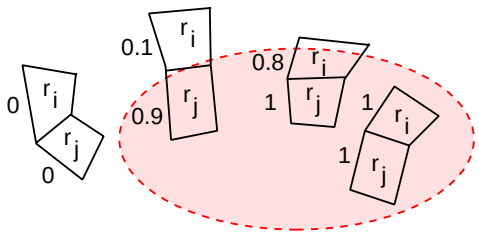


Figure 10.7: *The merging criterion also evaluates the position of the regions relative to the covered area of the model ellipse. Merging of two regions within the covered area is favoured. The numbers beneath the regions denote the value of* $|r_i \cap e^{(k)}|/|r_i|$.

description of the region-merging algorithm). We use the fast implementation of region-merging that is described in [46].

10.5.2 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 (10.10) 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 the region sizes.

$$S(r_i, r_j) = \underbrace{\frac{|r_i| \cdot |r_j|}{|r_i| + |r_j|}(\mu_i - \mu_j)^2}_{\text{Ward}} \cdot \underbrace{\left(\max_k \frac{|r_i \cap e^{(k)}|}{|r_i|} \cdot \frac{|r_j \cap e^{(k)}|}{|r_j|} + \beta \right)^{-1}}_{\text{penalty for crossing object/background border}} \tag{10.10}$$

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 inside the object with 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

$$\frac{|(r_i \cup r_j) \cap e^{(k)}|}{|r_i \cup r_j|} = \frac{|r_i \cap e^{(k)}| + |r_j \cap e^{(k)}|}{|r_i| + |r_j|} \tag{10.11}$$

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 smallest $S(r_i, r_j)$ 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 filled up in a final post-processing step.

10.6 Experiments and results

This section presents the performance of the proposed object-detection system on various input sequences. The first example is taken from the *paris* sequence. This is a head-and-shoulder sequence without camera motion. The two main difficulties with this example are that the body of the man at the left does not move much, which makes it undetectable for the motion-segmentation, and that the color difference between the hair of the man and the background is small. Because of this small difference, a color-based segmentation without any semantic knowledge usually results in a wrong segmentation (Fig. 10.8(a)), 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. 10.2) and applied the algorithm to frame 20. The detected object is superimposed onto the input frame in Figure 10.8(e). The model is placed at a sensible configuration, but the object is still not completely covered by the model, since the use of ellipses does not fit exactly to the region shape. Figure 10.8(f) shows the result after a color segmentation that integrated the detected object location. Almost the complete object is covered in the segmentation mask. Small parts of the head are missing as these areas are 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, which has 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. 6.22). Nevertheless, the long-term change-detection mask (Fig. 10.9(a)) 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 movements within the audience,
- 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 only one of them may be desired in the output.

We constructed a model of the tennis player and applied the segmentation algorithm again 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. 10.9(d).

Another example showing a news report scene is depicted Figure 10.10. The body of the foreground object is mostly static, but there is strong motion in the scene background. Still a good segmentation result is obtained.

Figure 10.11 shows example frames from the *carphone* sequence. It is visible that the detected object follows the input sequence without major problems as long as only articulated motion is present. However, in frame 180, the size of the object in the input frame is increased because the man comes closer to the camera. This change of object size is not included in our model and consequently, the object cannot be covered completely.

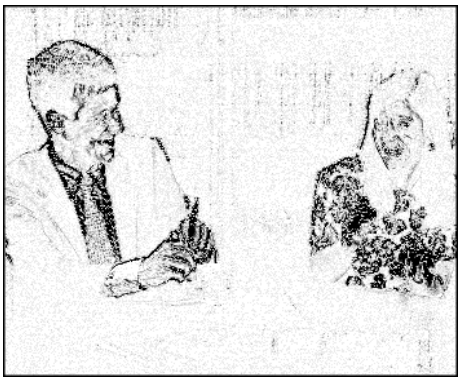
Finally, Figure 10.12 presents results for the *foreman* sequence. At the end of the sequence, the camera turns and the object leaves the visible area. During this time, the algorithm has difficulties because not all parts of the object are visible. Since the algorithm has to find all parts in the input image, it places some parts at areas beneath the object. In Figure 10.12(d), it is visible that the complete model is stretched to keep the body part inside the image as much as possible, while the head moves left.

10.7 Conclusions

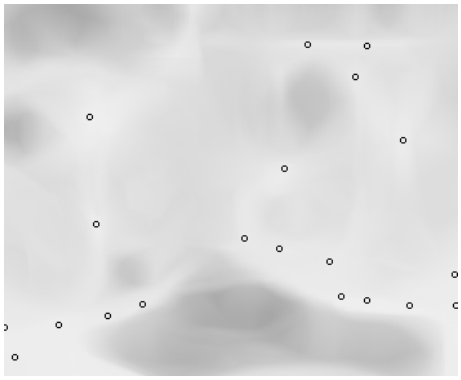
This chapter has described a segmentation system that combines motion detection, spatial segmentation, and model-based object detection into a single framework. Motion detection is used for an approximate localization of the object position. The object detection fits a manually defined object model to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

The object model uses attributed graphs to describe the main regions of the object and their spatial relationship. Because the spatial relationships are restricted to a tree structure, the matching algorithm can apply a dynamic programming approach for the efficient detection of the model. The restriction to tree-shaped graphs is no serious limitation for practice, because most natural objects have articulated limbs, but no cycles. The shape of each object region is described compactly with an ellipse, because it provides a good compromise between computational simplicity and accuracy of describing the object shape. Note that a too detailed description

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



(b) Short-term change-detection mask.



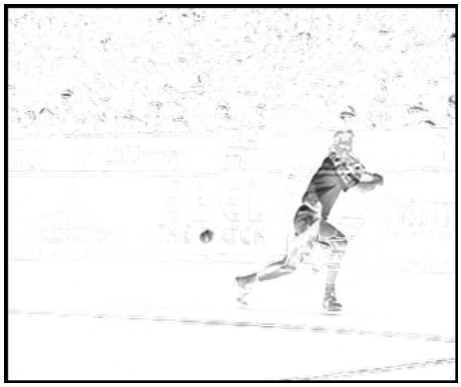
(c) Node matching cost for the region corresponding to the tie. Brighter means lower cost. Candidate positions are marked.

(d) Candidate configurations for the tie region.

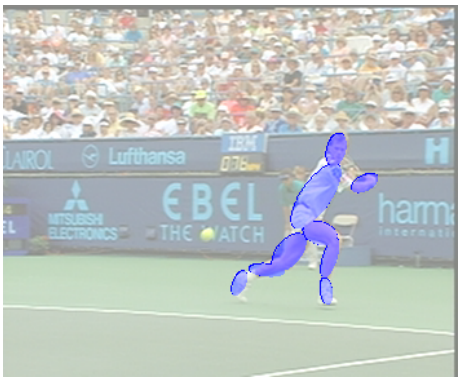
(e) Fitted object model.

(f) Final segmentation mask.

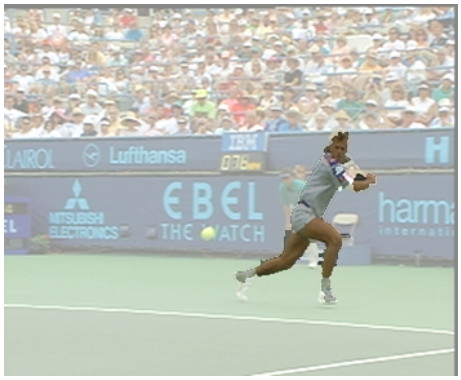
Figure 10.8: *Frame 20 of the* paris *sequence.*



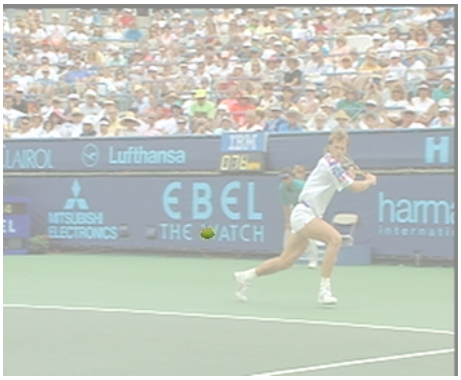
(a) Long-term change-detection mask.



(b) Fitted object model.



(c) Final segmentation (player).



(d) Final segmentation (ball).

Figure 10.9: *Frame 177 of the* stefan *sequence.*

(a) Frame 1.

(b) Frame 150.

Figure 10.10: *News report. Stong motion also occurs in the background.*

of object shape is not desired, since the object shape usually varies considerably when viewing the object from different directions.

The algorithm has the clear 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 that he wants to extract, without doing the segmentation manually. A further advantage is that in cases in which only part of the object is moving, or in cases in which the object is not clearly distinguishable from the background, the object segmentation can be solved through the combination of several features.

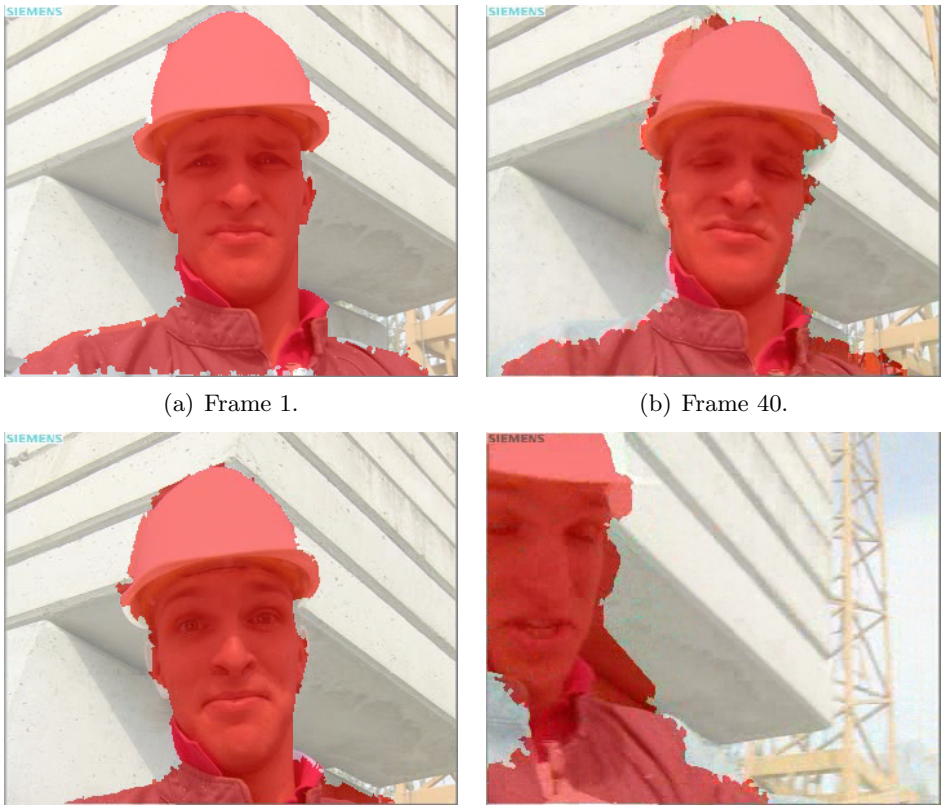
Problems with the current approach mostly become visible at areas close to the region boundary, where the region-shape cannot be approximated well using ellipses. Therefore, future research should consider other representations for region shapes. We have conducted first experiments that use deformable templates for the object regions. This approach seems promising, since it appears to provide more accurate object boundaries. However, it is more sensitive to 3-D motion and deformations of the object. Hence, finding a good model for the objects will be an interesting topic for further research.

Another possible improvement may be the inclusion of textured regions in the object model, since our current approach is still limited to uniformly-colored object regions. 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 and we may want to add this knowledge to help the matching process. However, in most cases this would introduce additional graph edges which will violate the tree-structure limitation of our graph-matching algorithm. Thus, future research should also consider object models for which fitting algorithms exist that provide a good compromise between accuracy and computational complexity.

(a) Frame 1. (b) Frame 90.

(c) Frame 180. (d) Frame 380.

Figure 10.11: *Carphone sequence. In (c), the head is not completely covered, since the size in the image is larger than the size in the model.*



(a) Frame 1. (b) Frame 40.

(c) Frame 80. (d) Frame 280.

Figure 10.12: *Foreman sequence. Around frame 280, the object leaves the visible area and the algorithm cannot localize the complete model anymore.*

10.8 Appendix: notes on ellipse processing

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. A conic at the origin is defined as

$$A'x^2 + B'xy + C'y^2 = F'. \tag{10.12}$$

Shifting the conic to the center (c_x, c_y) results in

$$A'(x - c_x)^2 + B'(x - c_x)(y - c_y) + C'(y - c_y)^2 = F'. \tag{10.13}$$

Comparing this with the general equation for conics (10.1), we obtain the two equations

$$\begin{aligned} D &= -(Ac_x + Bc_y) && (10.14)\\ E &= -(Cc_y + Bc_x). && (10.15) \end{aligned}$$

From this, we can compute the center of the conic by

$$c_y = \frac{A \cdot E - D \cdot B}{B^2 - A \cdot C}, \qquad c_x = -\frac{D + B \cdot c_y}{A}. \tag{10.16}$$

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

$$\begin{aligned} &A' = A, \quad B' = B, \quad C' = C, \quad D' = E' = 0, \quad \text{and}\\ &F' = F - (Ac_x^2 + 2Bc_xc_y + Cc_y^2). \end{aligned} \tag{10.17}$$

This gives us the zero-centered conic equation $\begin{pmatrix} x & y \end{pmatrix} \mathbf{Q} \begin{pmatrix} x & y \end{pmatrix}^T = 1$ with

$$\mathbf{Q} = -\frac{1}{F'} \begin{pmatrix} A & B \\ B & C \end{pmatrix}. \tag{10.18}$$

By determining the Eigenvectors of $\mathbf{Q}$ and scaling according to their Eigenvalues, we obtain 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

$$\left(\frac{x'}{a_1}\right)^2 + \left(\frac{y'}{a_2}\right)^2 = 1 \tag{10.19}$$

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

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x - c_x \\ y - c_y \end{pmatrix}, \tag{10.20}$$

we finally obtain

$$\begin{aligned}
A &= a_2^2 \cos^2\alpha + a_1^2 \sin^2\alpha \\
C &= a_2^2 \sin^2\alpha + a_1^2 \cos^2\alpha \\
B &= (a_1^2 - a_2^2)\cos\alpha\sin\alpha \\
D &= a_2^2 \cos\alpha(-c_x\cos\alpha + c_y\sin\alpha) - a_1^2 \sin\alpha(c_x\sin\alpha + c_y\cos\alpha) \\
E &= a_2^2 \sin\alpha(c_x\cos\alpha - c_y\sin\alpha) - a_1^2 \cos\alpha(c_x\sin\alpha + c_y\cos\alpha) \\
F &= a_2^2(c_x\cos\alpha - c_y\sin\alpha)^2 + a_1^2(c_x\sin\alpha + c_y\cos\alpha)^2 - a_1^2 a_2^2.
\end{aligned} \tag{10.21}$$

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

$$F(x, y) = \sum_{i=0}^{x} f(i, y). \tag{10.22}$$

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 (10.1), we obtain for a fixed y

$$x_{min}, x_{max} = -\frac{B+D}{A} \mp \sqrt{\left(\frac{B+D}{A}\right)^2 - \frac{F + Cy^2 + 2Ey}{A}}. \tag{10.23}$$