In theory, there is no difference between theory and practice. But in practice, there is.
(Manfred Eigen)

Chapter 7

Background Subtraction

When the background view excluding the foreground objects is available, it becomes obvious that the foreground objects can be obtained by comparing the background image with the current video frame. In practice, camera noise and regions in which the object has the same color as the background make the separation of foreground objects and background more difficult. The previous chapters showed how this background image is reconstructed from a video sequence, and how the camera motion can be compensated. This chapter discusses the background-subtraction module, which determines the segmentation masks that are the final output of the segmentation system. The chapter commences with simple independent pixel classification and then proceeds to more complex tests that include contextual information to decrease the number classification errors. Furthermore, typical problems that lead to segmentation errors are identified and a modification to the segmentation algorithm is provided to reduce these effects. Finally, a few postprocessing filters are presented that can remove obvious errors like small clutter regions.

7.1 Introduction

Chapter 5 described how a pure background image of a video scene can be synthesized from the input video itself. By combining many video frames into this background image, we are able to reconstruct the background image excluding the foreground objects. Such a background image can be elegantly used to determine the foreground objects by comparing the input frame with the background image and mark the differences as foreground objects. This technique is commonly known as *background subtraction* or *change detection*. It is the most popular approach in video surveillance applications, because it is a computationally efficient technique and it is relatively easy to obtain background images for static surveillance cameras.

Previous work

The change-detection problem has been studied for a variety of applications, where surveillance, medical diagnosis, and remote sensing are currently the most important. In all of these applications, the objective is to compare an image to a background image and identify the regions that have changed. Unfortunately, the exact definition of what is actually meant with *changed* depends on the application and cannot be generalized.

A large number of change-detection operators have been proposed and good surveys about available techniques can be found in [41, 152, 157, 19]. Apart from independent-pixel classification schemes, algorithms have been developed that integrate contextual information to improve the robustness. Prominent techniques are classifiers using statistical models of the pixel neighborhood [1, 202], Markov Random Field (MRF) based models [101, 15, 14, 3, 98], or algorithms combining intensity and texture differences [113].

Since surveillance is a major application, many algorithms are designed to be invariant to changes in global illumination. A standard technique to approach the problem of varying illumination is to model the luminance distribution of each background pixel with one or several Gaussian distributions [174]. However, since we consider only short video sequences in our application, the problem of global illumination changes is usually neglegible.

Another common difficulty is the problem of image misregistration. It has been shown in [33] that even small registration errors of less than 0.2 pixels can lead to about 10% of additional false detections. As a solution, it has been proposed in [13] to estimate the expected distribution of misregistration noise and adapt the distance measure accordingly.

Chapter outline

This chapter gradually develops a robust background subtraction algorithm in a number of steps. We start with independent classification of the input pixels, for which we compare different metrics to detect changed image content. It will be shown that classifying the pixels independently does not provide robust results, since objects often have colors similar to the background and there is also a considerable amount of noise in the input video. Both effects cause misclassification errors, namely foreground pixels that are not detected, and background pixels that are detected as changed. To enhance the robustness, we implement a classification scheme [1] that considers a neighborhood of pixels at once, using a χ^2 significance test. This approach is further refined to using a Markov Random Field model, allowing to integrate pre-knowledge about the object shape. Finally, we propose a few modifications to reduce problems caused by image misregistration and we add morphological postprocessing operations to optimize the object masks.

7.2 Pixel-based classification

Change detection can be carried out on either greyscale information only, or using the full color information. Many algorithms in the literature are described for the greyscale case, but it is obvious that the robustness can be increased by examining all three color channels (e.g., consider two differently colored objects with equal luminance).

7.2.1 Distance metrics

We denote the pixel values in the RGB color-space as triples $\mathbf{I}^{RGB} = (I^R, I^G, I^B)$ and in the YUV color-space as $\mathbf{I}^{YUV} = (I^Y, I^U, I^V)$. The background and input image are indicated with a subscript, such that I_B denotes the background image and I_t denotes the current input image.

The following distance metrics are implemented:

- the pixel difference of only the luminance channel

$$d_y = |I_t^Y - I_B^Y|, \tag{7.1}$$

- the sum of pixel differences of the three color channels

$$d_1^{RGB} = |I_t^R - I_B^R| + |I_t^G - I_B^G| + |I_t^B - I_B^B|, \tag{7.2}$$

- the sum of squared pixel differences of the three color channels

$$\left(d_2^{RGB}\right)^2 = |I_t^R - I_B^R|^2 + |I_t^G - I_B^G|^2 + |I_t^B - I_B^B|^2, \tag{7.3}$$

- the maximum pixel difference in the three color channels

$$d_\infty^{RGB} = \max\{|I_t^R - I_B^R|, |I_t^G - I_B^G|, |I_t^B - I_B^B|\}, \tag{7.4}$$

- and the Mahalanobis distance in YUV space

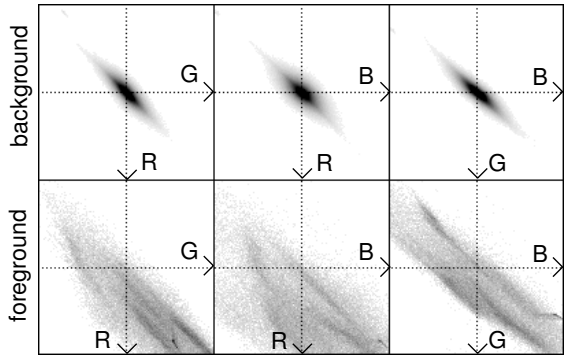
$$\left(d_M^{YUV}\right)^2 = (\mathbf{I}_t^{YUV} - \mathbf{I}_B^{YUV})\ \mathbf{S}^{-1}\ (\mathbf{I}_t^{YUV} - \mathbf{I}_B^{YUV})^\top, \tag{7.5}$$

where $\mathbf{S}$ is the 3×3 covariance matrix of the color differences. It is discussed later how the entries of this matrix are chosen. Note that the three metrics d_1, d_2, d_∞ correspond to the L_1, L_2, and L_∞ norms. Additionally to the RGB space, these three metrics are also implemented in the YUV space, where they are denoted as $d_1^{YUV}, d_2^{YUV}, d_\infty^{YUV}$. All proposed metrics have in common that their values increase when the color $\mathbf{I}_t$ in the current image deviates more from the reference color $\mathbf{I}_B$. This enables a classification in which a pixel is noted as changed (foreground) if the difference to the background exceeds a threshold τ. For example, using the greyscale difference, we obtain the decision function $d_y > \tau$.

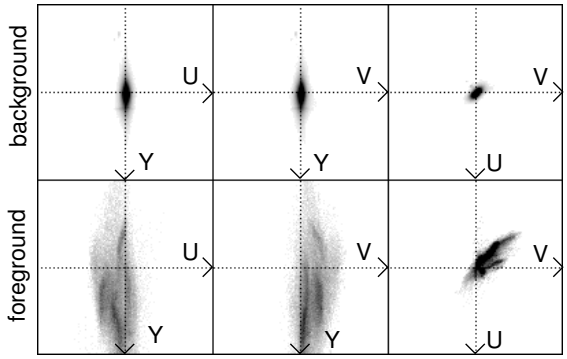
7.2.2 Influence of the color-space

The previous section has defined a number of metrics for measuring the distance between two colors. It is not obvious, which of these metrics will yield the most accurate result in combination with a certain color-space. Theoretically, the choice of the color-space has no influence on the classification accuracy. To understand this, assume that we have a best reference metric in some color-space. This metric together with a threshold defines a decision boundary in this color-space. Now we take an arbitrary different color-space. If there is a transform from the first color-space to the second, we can also transform the decision boundary into the second color-space, so that we realize a classification function of similar performance in every color-space.

In practice, the question is for which color-space a computationally simple metric can be defined that leads to a good classification. Primarily, this depends on the distribution of noise in the color-space. Let us assume that the noise is mainly a variation of luminance, while the hue of the color remains stable. Consequently, the noise distribution in the YUV space will have its largest variance along the Y-axis and it will have considerably



(a) RGB color-space.

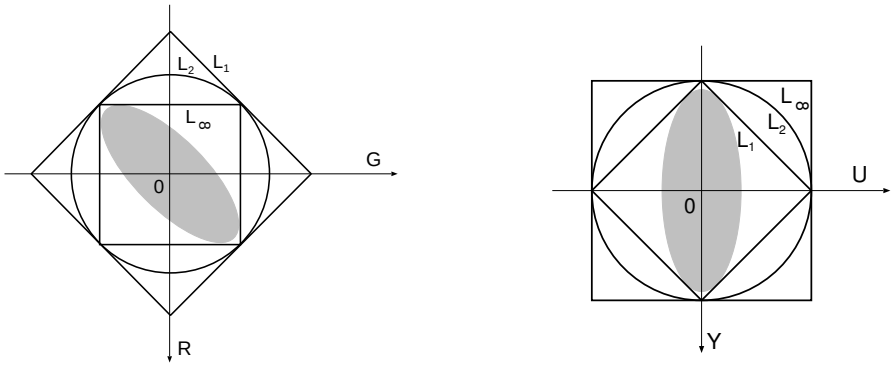


(b) YUV color-space.

Figure 7.1: *Distribution of color-differences for the* stefan *sequence, separately for changed and unchanged pixels. The images show projections of the 3-D distribution.*

smaller variance in the U and V dimension (see Figure 7.1). In the RGB color-space, the noise distribution is oriented along the diagonal vector $(1, 1, 1)$. This is of course a simplification, because the orientation of the distribution will vary for different colors. However, we can still make use of it, since the majority of the colors in natural images are usually not very saturated.

Let us now discuss how the isosurfaces of constant cost are defined by the different metrics. Since the L_2 metric equals Euclidean distance, the isosurfaces of constant cost are spheres. For L_∞, they are axis-parallel cubes and for L_1, they are cubes with corners on the axes. Selecting a specific threshold corresponds to selecting one of the isosurfaces to separate the class of background colors (inside of the closed isosurface) from the foreground-colors (outside space).



(a) RGB color-space. (b) YUV color-space.

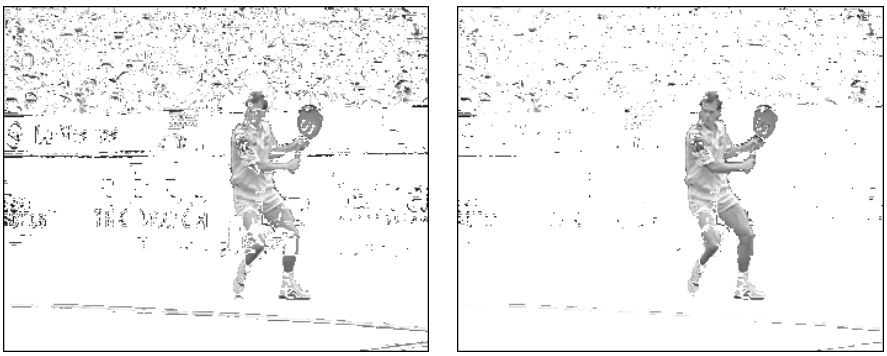
Figure 7.2: *Isolines of constant cost for L_∞ in the RGB space (a), and for L_1 in the YUV space (b). Displayed is a cut through the three-dimensional color-space such that the reference background color is at the origin. The optimal metric to separate a foreground color from the background distribution depends on the orientation of the background distribution.*

Figure 7.2 depicts typical noise distributions in the RGB and the YUV color-space. It can be concluded that the distribution along the diagonal (RGB space) can be enclosed most tightly with the L_∞ norm. It is easy to note that using the L_1 or L_2 metrics would enclose more foreground colors when the complete background distribution should still be contained. The same argument is also valid in the YUV-space, in which the L_1 norm provides a good separation.

In both color-spaces, neither of the L_1, L_2, or L_∞ metrics enclose the background distribution optimally. However, assuming that the noise distribution is a multi-variate gaussian and provided that its covariance matrix is known, the results can be improved by using the Mahalanobis distance metric. In the case of the YUV space, this is especially easy, since it can be assumed that the luminance is independent from the two chrominance dimensions. Consequently, the distribution is axis-parallel and the covariance matrix is diagonal $\mathbf{S} = \sigma^2 \mathrm{diag}(1, \beta^2, \beta^2)$ with the standard deviation σ in the luminance channel and $\beta \cdot \sigma$ in the color channels. This gives the particularly simple formulation of d_M^{YUV} as

$$\left(d_M^{YUV}\right)^2 = \frac{1}{\sigma^2}(I_t^Y - I_B^Y)^2 + \frac{1}{\sigma^2\beta^2}\left((I_t^U - I_B^U)^2 + (I_t^V - I_B^V)^2\right). \qquad (7.6)$$

Because a change of σ^2 can be compensated by adapting the threshold τ, the standard deviation can be arbitrarily set $\sigma = 1$. Note that this gives



(a) Thresholded greyscale difference. (b) Thresholded RGB error maximum.

Figure 7.3: *Background differencing using independent pixels (*stefan *sequence, frame 6).*

basically the equation for Euclidean distance with an additional scaling factor for the chrominance components. According to our experiments, good results are obtained for $\beta^2 \approx 0.1$.

7.2.3 Classes of errors

Example results of independent pixel classification are depicted in Figure 7.3. Two kinds of errors can be identified in the results.

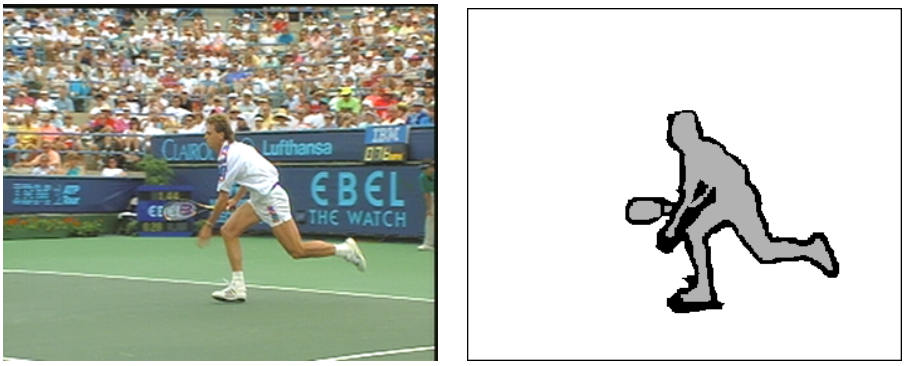
1. Pixels that actually belong to the background, but which are classified as foreground, because of camera noise or misregistration errors.

2. Pixels that belong to the foreground object, but which are classified as background, because their color is very similar to the background at the same position.

It is not possible to minimize both errors at the same time. Depending on the decision threshold, it is only possible to reduce one type of error while simultaneously increasing the other type of error.

7.2.4 Evaluation method

To be able to quantify the quality of the obtained segmentation masks, we manually created reference masks for several sequences. Because an object often has a blurred boundary or a soft shadow, it is difficult to define the correct border of an object. We therefore separated the pixels in our reference masks into three different classes: *Foreground* pixels, *background*



(a) Input image. (b) Reference mask.

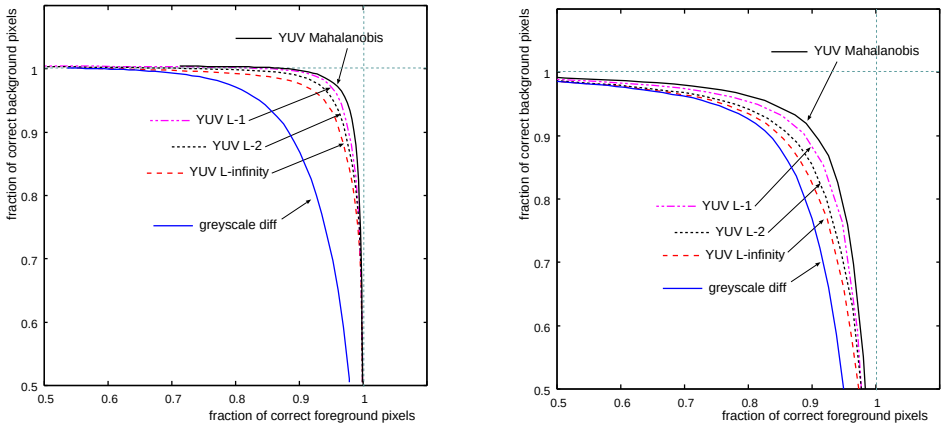
Figure 7.4: *Example of a manually generated reference mask. Grey areas indicate* foreground *regions, black areas represent* don't care *regions that are excluded from the evaluation of segmentation results.*

pixels, and *don't care* pixels that are excluded from the evaluation (see Fig. 7.4(b)).

To evaluate the quality of the segmentation algorithms, we compare the obtained segmentation masks with the reference masks by counting the percentage e_b of incorrectly classified pixels in the background region and the percentage of incorrectly classified pixels in the foreground region e_f. Note that both error percentages are depending on the chosen threshold. Lowering the threshold will lead to less pixels that are classified as foreground and hence, it will decrease e_b, but at the same time, it will increase e_f. Increasing the threshold has the opposite effect.

The dependence of the two error classes for different thresholds are collected in an ROC (*Receiver Operating Characteristic*) curve [150]. This curve depicts the relation between the two errors that the algorithm yields for different thresholds (see Fig. 7.5). An ideal segmentation algorithm would succeed to reach 100% correct foreground pixels and 100% correct background pixels at the same time. Practically, this ideal case cannot be obtained.

In general, two ROC curves for different algorithms are not necessarily better or worse along the whole curve. It is well possible that two curves intersect. To still quantify the accuracy of a segmentation algorithm, we measure its performance using the area under the ROC curve.



(a) *Stefan* sequence. (b) *Surveillance* sequence.

Figure 7.5: *ROC curves for single pixel classification. Because the* stefan *sequence comprises more saturated colors, the difference between the greyscale metric and the color metrics are larger than for the* surveillance *sequence.*

7.2.5 Results

The performance of all pixel-based classification metrics on the three sequences *stefan*, *surveillance*, and *hall-and-monitor* was measured. A description about the content of these sequences can be found in Appendix D.

Greyscale vs. color metrics

The first observation is that color-based metrics provide higher accuracy than the greyscale metric d_y (see the ROC curves in Fig. 7.5). This is not surprising, since each colored pixel comprises three channels which should all be non-changing when background is shown, whereas a change in one of the color channels is sufficient to detect a foreground object. However, the advantage of a color-based metric over a greyscale metric diminishes if the sequence shows mostly low-saturated colors. This can be observed when comparing the colorful *stefan* with the low-saturated *surveillance* sequence. While the difference between the metrics is well visible for the colorful *stefan* sequence, it is smaller for the *surveillance* scene.

Influence of the color-space

We claimed in Section 7.2.2 that from the implemented metrics, the L_∞ metric should give the most accurate results in the RGB-space. In the YUV-

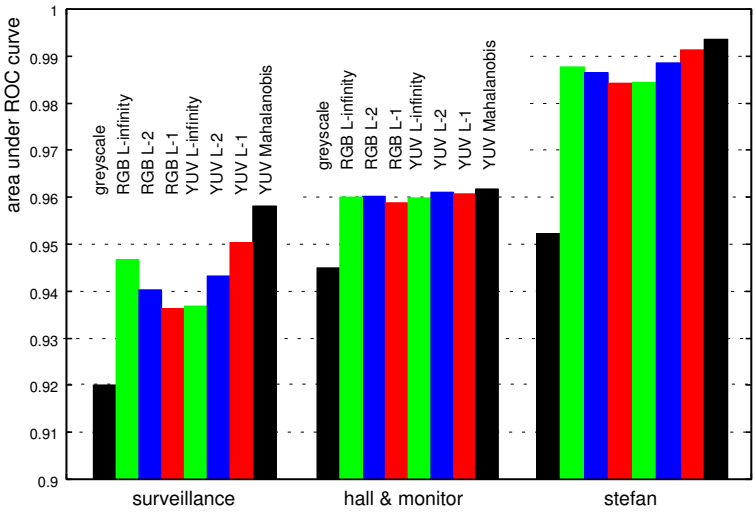


Figure 7.6: *Area under ROC curve for greyscale and color-based distance metrics.*

space, the best results should be obtained using the L_1 or, even better, the Mahalanobis distance.

This predicted behaviour can indeed be seen in Fig. 7.6. For RGB, L_∞ gives the best results, followed by L_2 and L_1 at the end. For YUV, it is just the opposite, with the Mahalanobis distance giving the best results, followed by L_1, L_2, and L_∞.

7.3 Multi-pixel based significance tests

Up to this point, the pixels in the image were considered to be independent. However, since objects are compact regions of foreground pixels, the probability that a pixel belongs to a specific class increases with the number of pixels of the same class in its neighborhood. This regularity can be exploited to increase the robustness of the segmentation algorithm. Two main approaches are possible. The first assumes that all pixels in a small neighborhood belong to the same class (either foreground or background). With this assumption, the measurements in the pixel neighborhood can be combined to deduce a more robust decision function. The second approach uses a Markov Random Field model to define *a-priori* probabilities for the pixel classification based on the class of its neighborhood pixels. Both approaches will be discussed in the current and the subsequent section, respectively, as they are both used in our final segmentation system.

7.3.1 Classification using a χ^2 test

If the change detection is based only on observations of individual pixels, it is often not possible to detect whether small color differences are the consequence of a foreground object or if they are only camera noise. To make change detection more robust, the neighborhood of a pixel should be included into the decision. Let us introduce this approach by reviewing the technique proposed in [1] for greyscale images and Gaussian noise. Subsequently, we will enhance this algorithm to color images.

Let us denote the difference between the current input greyscale image and the background as $D^Y = I_t^Y - I_B^Y$. For simplicity, we omit the superscript for D in the following. In the case that the pixel did not change (null-hypothesis H_0), we assume that it is subject to Gaussian noise with variance σ^2, so that the pixel difference satisfies the distribution

$$p(D|H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{D^2}{2\sigma^2}\right\}. \tag{7.7}$$

It is a valid assumption to use Gaussian noise, since the pixel differences are only a consequence of the camera noise. Another distribution that is frequently used in this context is the Laplace distribution. However, the Laplace distribution is usually considered as the distribution of prediction errors. In fact, we conducted experiments which showed that the real distribution is *in-between* Gaussian and Laplace. Since the variance of the noise is very small, both distributions can be used with comparable results.[1]

Let us now observe a window w centered at the considered pixel. We construct a vector $\mathbf{D}$ of all pixel differences within the window as $\mathbf{D} = (D(x,y))_{(x,y)\in w}$. Since the pixel labels have a high spatial correlation, we employ the simple assumption that the center pixel of the window is classified as unchanged only if every pixel in the window has a low difference value. This assumption is valid for most areas of the image, but obviously, it is not true along the object boundaries (Fig. 7.7). The defects that are introduced along the object boundary are eliminated in the subsequent processing stage that is described in Section 7.4.

Under the assumption that all pixels in the window are unchanged and their noise is independent, the probability distribution becomes

$$p(\mathbf{D}|H_0) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{N_w} \exp\left\{-\sum_{(x,y)\in w} \frac{D(x,y)^2}{2\sigma^2}\right\}, \tag{7.8}$$

[1] In [1] it is derived that for the Laplace distribution, a test-statistic using the sum of absolute differences should be used, while for the Gaussian distribution, the sum of squared differences is optimal.

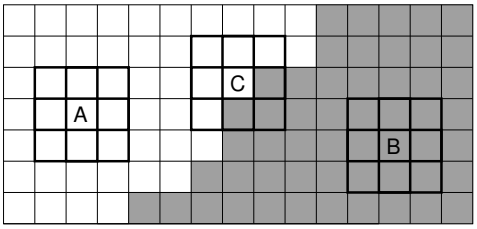


Figure 7.7: *In the multi-pixel classification test, we label a pixel as unchanged only if all pixels in the neighborhood window have small differences. This leads to the correct decision for pixels A and B (dark pixels denote object pixels with large difference values). However, pixel C will be erroneously classified as object pixel. Consequently, the detected object will be slightly larger than the true object.*

where N_w denotes the number of pixels within the window w. Because it is very difficult to deduce any property of the probability distribution for the case of the counter-hypothesis H_1 that a pixel has changed, we use a significance test based on $p(\mathbf{D}|H_0)$. Notice that this distribution can also be written as a function of the sum of squared pixel differences

$$\Delta = \sum_{(x,y)\in w} (D(x,y)/\sigma)^2. \tag{7.9}$$

Assuming that changed foreground pixels will show larger pixel differences, we will put a threshold t_α on Δ such that a pixel is detected as changed, if $\Delta > t_\alpha$. Hence, the probability of obtaining a false positive equals $P(\Delta > t_\alpha|H_0)$. Now, we can choose a significance level α defined as

$$P(\Delta > t_\alpha|H_0) = \alpha, \tag{7.10}$$

from which we can deduce t_α. Since Δ is a sum of Gaussian-distributed random variables, Δ itself is distributed according to a χ^2 distribution with N_w degrees of freedom. The complete classification process based on the χ^2 test can be summarized as follows.

- Choose an observation window w, a significance level α, e.g. 10^{-5}, and set an appropriate standard deviation of the image noise $\sigma \approx 10$.
- Use the inverse cumulative function of the χ^2 distribution to obtain a threshold t_α according to Eq. (7.10).
- Iterate through the picture and compute the sum of squared differences Δ within each neighborhood window. Set the center pixel to *changed* if $\Delta > t_\alpha$.

7.3.2 Extension to color images

The above significance test was designed for greyscale images, but Section 7.2 revealed that our results can be improved if the color information is included into the change detection algorithm. This can be achieved in a straight-forward way by adding the color channel values as extra pixels to the difference vector $\mathbf{D}$, so that it is extended to three times its original length:

$$\mathbf{D} = (\underbrace{D^Y(x,y),\dots}_{(x,y)\in w}\ ,\ \underbrace{D^U(x,y),\dots}_{(x,y)\in w}\ ,\ \underbrace{D^V(x,y),\dots}_{(x,y)\in w}). \tag{7.11}$$

Clearly, this means that the threshold has to be determined using the χ^2 distribution with $3 \cdot N_w$ degrees of freedom. Additionally, two important aspects are considered. First, we have to ensure that the differences in the two color channels are statistically independent to the luminance channel values and also between the color channels themselves. This can be considered true if we operate in the YUV color-space, but it is not valid for the RGB color-space, as we saw earlier. Therefore, we only consider the YUV color-space in the following. Second, the variances for difference signals in the luminance channels and the chrominance channels are different. This means that we should replace the constant σ in the monochrome case with σ for the luminance values and $\beta \cdot \sigma$ for the chrominance values. This leads to a modified definition of Δ, so that

$$\Delta = \sum_{(x,y)\in w} \frac{1}{\sigma^2} D^Y(x,y)^2 + \frac{1}{\sigma^2\beta^2}\left(D^U(x,y)^2 + D^V(x,y)^2\right) \tag{7.12}$$

$$= \sum_{(x,y)\in w} \left(d_M^{YUV}(x,y)\right)^2, \tag{7.13}$$

which is based on the Mahalanobis metric for the YUV color-space.

7.3.3 Fast implementation

A direct implementation of the above algorithm becomes inefficient for larger window sizes. To reduce the complexity, we propose to use the technique of cumulative sums [190]. Instead of computing the sum over a rectangular window w directly from the pixel costs $d_M^{YUV}(x,y)^2$, we first compute the cumulative costs

$$C(x,y) = \sum_{i=0}^{x}\sum_{k=0}^{y} \left(d_M^{YUV}(i,k)\right)^2. \tag{7.14}$$

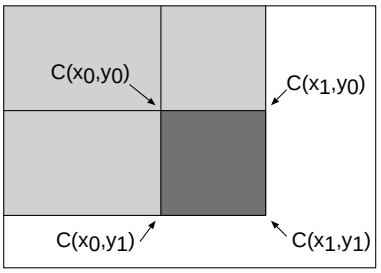


Figure 7.8: *The sum over the dark area can be computed from the cumulative sums* $C(x_1, y_1) - C(x_0 - 1, y_1) - C(x_1, y_0 - 1) + C(x_0 - 1, y_0 - 1)$.

Each $C(x, y)$ equals the sum of all pixel costs in the rectangle starting at the top-left corner and the bottom-right corner at (x, y). Note that $C(x, y)$ can be computed iteratively with two passes over the cost image. In the first pass, costs are cumulated in the horizontal direction, followed by a second pass in the vertical direction. When the cumulative costs $C(x, y)$ are available, the sum over an arbitrary rectangular area with top-left position (x_0, y_0) and bottom-right position (x_1, y_1) can be obtained in constant time with (see Fig. 7.8)

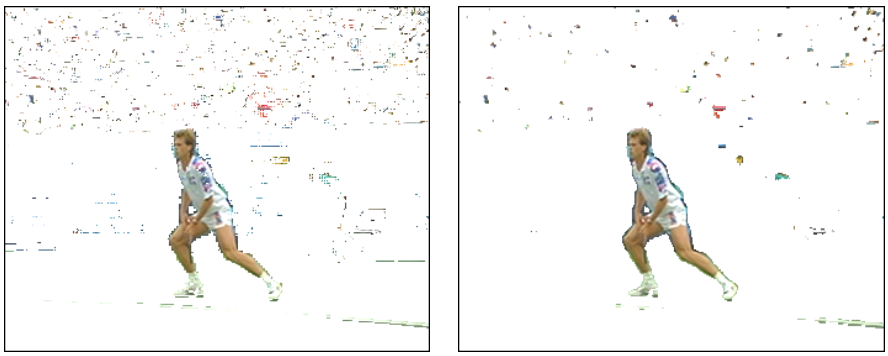
$$C(x_0, y_0, x_1, y_1) = C(x_1, y_1) - C(x_0 - 1, y_1) - C(x_1, y_0 - 1) + C(x_0 - 1, y_0 - 1). \tag{7.15}$$

7.3.4 Evaluation

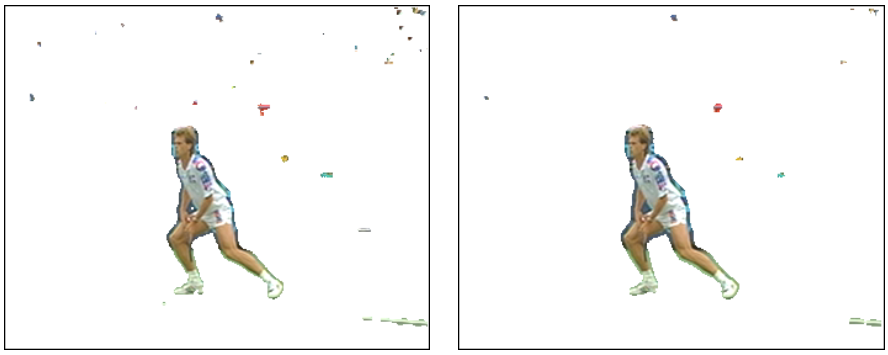
Two questions arise for the classification algorithms using the significance test. First, how does the quality of the result depend on the size of the neighborhood window, and second, does the inclusion of color information improve the segmentation result? To illustrate the typical segmentation masks that result from different window sizes, Figure 7.9 portrays the outcome of the color-based significance test.

Influence of window size

The immediate observation is that the amount of small pixel-noise is significantly reduced when using even a small 3×3 window, instead of the independent pixel classification. The amount of background errors is further reduced by applying larger window sizes, but simultaneously, a disadvantageous effect becomes more apparent. Foreground objects are surrounded by a growing *aura* of pixels that are falsely attributed to the foreground object.



(a) Independent pixel classification. (b) χ^2 test in 3×3 neighborhood.



(c) χ^2 test in 5×5 neighborhood. (d) χ^2 test in 7×7 neighborhood.

Figure 7.9: *Background differencing using significance tests on YUV color (*stefan *sequence, frame 80). Compare the reduced noise in the mask compared to an independent pixel classification with the Mahalanobis metric. But also note the increasing aura around the object larger window sizes.*

The reason for this effect is the initial assumption that a pixel is only classified as unchanged if every pixel in the window is unchanged. On the other hand, this means that the center pixel is classified as changed even if some pixels at arbitrary position in the window are changed, but not the center pixel itself. Consequently, foreground pixels at the object boundary have influence even on pixels that are further away from the object.

The same effect also leads to closing small holes in the segmentation mask. Even though closing these holes seems to be a good effect, this is not so important, since this can also be achieved with a simple post-processing step (will be described in Section 7.6.1).

We evaluated the quality of the obtained segmentation masks by comparing them to our reference masks in the same way as for the single-pixel

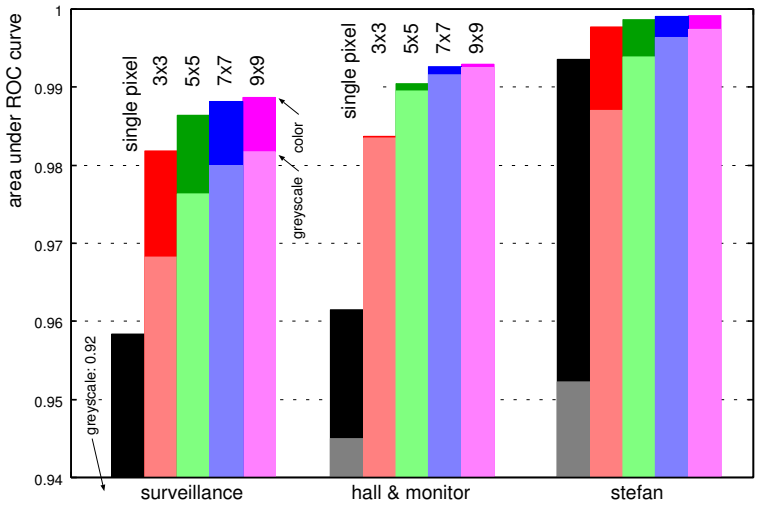


Figure 7.10: *Area under ROC curve for the color-based significance test with different window sizes. Each bar is divided into two parts. The lower (brighter) part shows the result using the greyscale distance measure, while the upper (darker) part shows the result for using the Mahalanobis color distance.*

classification case. Finally, we computed the ROC curves (see Fig. 7.11 for an example) and measured the area under the ROC curves (Fig. 7.10).

It can be noticed that the accuracy of the segmentation in fact increases with larger window sizes, where the maximum is at about 9×9 pixels. However, in this case, this evaluation with the ROC-area as a quality criterion is partly misleading because of two reasons. First, remember that the objects in the reference masks have an *aura* of don't-care pixels surrounding them, meaning that segmentation errors in these regions are not counted. When using a large neighborhood window, it appears that the masks grow larger than the objects. However, this error is not detected during the comparison with the reference masks, so that the quality measure still reports good results. For larger windows, the results start to deteriorate (Fig. 7.11). This phenomenon only occurs after the object aura extends beyond the don't-care areas of the reference masks.

Second, the background is typically larger than the objects, which means that false background pixels as in the object aura count much less than missing pixels in the object mask. This does not match the subjective quality, where errors of both classes count as equally imporant.

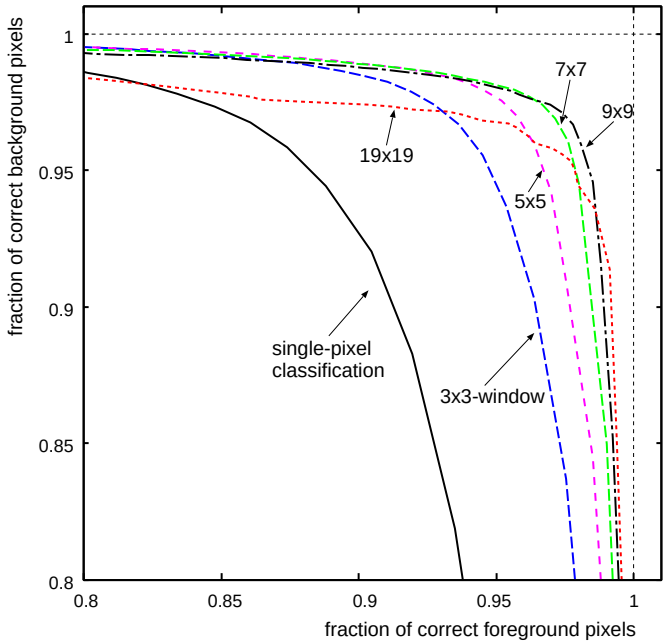


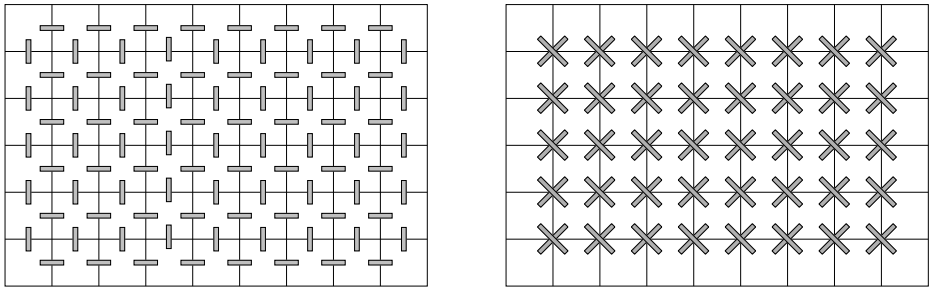
Figure 7.11: *ROC curves for χ^2 significance test with varying window size* (hall-and-monitor).

Greyscale vs. color

As expected, Fig. 7.11 shows that color information gives an additional gain to the segmentation accuracy. However, it can also be deduced that the differences are not as large as before, since good results can also be obtained using only greyscale information. Especially for the *hall-and-monitor* sequence, the color information seems to have almost no influence on the result. It is surprising that adding the color information does not make a big difference for the *stefan* sequence, because we obtained big improvements for this sequence with single-pixel decision functions. However, it should be noted that the results for *stefan* are already very close to the optimum.

7.4 Classification using Markov random fields

The results of the statistical-significance algorithm showed that the segmentation can in fact be improved by incorporating contextual information into the decision process. While reducing the noise in the segmentation mask, the previously discussed algorithm has the main disadvantage that the po-



(a) $\mathcal{C}_{hv}$ (b) $\mathcal{C}_{diag}$

Figure 7.12: *The cliques set $\mathcal{C}_{hv}$ comprises the all direct horizontal/vertial neighbors, while $\mathcal{C}_{diag}$ comprises diagonal neighbors.*

sitions of the object boundaries are not preserved. The obtained segmentation masks are extended beyond the true object boundary, resulting in the *aura* effect.

7.4.1 MRF model for segmentation masks

A change detection algorithm that is based on a Markov Random Field (MRF) model for the segmentation mask has been proposed in [2]. The idea is to determine the segmentation mask as a Maximum A-Posteriori (MAP) approximation, in which the *a-priori* probability of the segmentation masks are modeled as Gibbs/Markov Random Fields. We will review the model in this section, extend it to color images, and discuss its implementation.

Let $Q = \{q(x, y)\}$ be the segmentation mask, where $q(x, y) \in \{u, c\}$ is the label of one pixel. We label an unchanged (background) pixel as $q(x, y) = u$ and a changed pixel (foreground) as $q(x, y) = c$. We model the *a-priori* probability of a segmentation mask as a Gibbs field [114] with second-order cliques. To define the clique potentials, we further divide the second-order cliques into the set of horizontal/vertical cliques

$$\mathcal{C}_{hv} = \big\{\{(i, k), (x, y)\} \;\big|\; |i - x| + |k - y| = 1\big\} \tag{7.16}$$

and the set of diagonal cliques

$$\mathcal{C}_{diag} = \big\{\{(i, k), (x, y)\} \;\big|\; |i - x| = |k - y| = 1\big\}\,. \tag{7.17}$$

These two sets of cliques are illustrated in Fig. 7.12. Let us define the probability of a given segmentation mask Q by

$$p(Q) = \frac{1}{Z} \exp\{-U\} \tag{7.18}$$

where the constant Z normalizes the sum of probabilities to unity. However, it will be shown in the sequel that the value of Z is not required for the resulting computations. The properties of the segmentation masks are modeled with the energy function U as

$$U = \sum_{\{(x,y),(x',y')\}\in\mathcal{C}_{hv}} \gamma_{hv} \cdot V(q(x,y), q(x',y')) \quad + \tag{7.19}$$

$$\sum_{\{(x,y),(x',y')\}\in\mathcal{C}_{diag}} \gamma_{diag} \cdot V(q(x,y), q(x',y')), \tag{7.20}$$

where the function $V(q, q')$ is defined as

$$V(q,q') = 1 - \delta(q,q') = \begin{cases} 1 & \text{if } q \neq q', \\ 0 & \text{if } q = q'. \end{cases} \tag{7.21}$$

Consequently, the parameters γ_{hv} and γ_{diag} control the regularity of neighbored labels. Setting both values to zero results in equally probably segmentation masks ($p(Q) = const$), which leads to an independent pixel classification. Higher values for $\gamma_{hv}, \gamma_{diag}$ increase the probability of masks in which neighboring pixels have the same label. Consequently, the resulting segmentation masks will have smoother boundaries, but small details can be lost. See Table 7.1 for the values that we used during our experiments.

7.4.2 Obtaining a MAP estimate

Let us now consider a single pixel (x, y). We want to know which state (*changed* or *unchanged*) is more probable, given the difference image $d(x, y)$ and keeping the remaining segmentation mask fixed. In other words, we want to know if

$$p(q = u|d) \gtrless_c^u p(q = c|d), \tag{7.22}$$

where the pixel coordinates are omitted for brevity. Using the Bayes rule, we can rewrite this as

$$\frac{p(d|q = u) \cdot p(q = u)}{p(d)} \gtrless_c^u \frac{p(d|q = c) \cdot p(q = c)}{p(d)}, \tag{7.23}$$

or simply

$$p(d|q = u) \cdot p(q = u) \gtrless_c^u p(d|q = c) \cdot p(q = c). \tag{7.24}$$

To solve this, an assumption about $p(d|q)$ is needed. In the case of unchanged pixels ($q = u$), we model this as Gaussian noise with variance σ^2 according to Eq. (7.7). We cannot derive much about the case of changed

pixels so that we model this also with a Gaussian distribution, but with a much larger σ_c. Finally, we determine the probabilities $p(q)$, where Q is fixed with the exception of this one pixel at (x, y). Since we modeled the segmentation mask with a Gibbs random field, the influence of one pixel is limited. The total field probability is computed from all cliques, but the number of cliques that are influenced by the choice of one pixel is small (see Fig. 7.12). This allows us to reorder the two sums in Eq. (7.20) into two new sums, namely one sum over all cliques $\mathcal{C}^0$ that are not affected by the choice of the pixel (the majority), and the sum over cliques $\mathcal{C}^1$ that are affected by the pixel. This results in the desired probability

$$p(q) = \underbrace{\frac{1}{Z} \cdot \exp\left\{ -\sum_{\mathcal{C}^0} \cdots \right\}}_{\text{unaffected by label of } q} \cdot \underbrace{\exp\left\{ -\sum_{\mathcal{C}^1} \cdots \right\}}_{\text{depending on label of } q}. \tag{7.25}$$

The second sum can be written as

$$\sum_{\mathcal{C}^1} \cdots = n_{hv}\gamma_{hv} + n_{diag}\gamma_{diag}, \tag{7.26}$$

where n_{hv} is the number of horizontally or vertically neighboring pixels with a different label, and n_{diag} is the number of diagonally neighboring pixels with different labels (see Fig. 7.13(b)). Since we consider both cases of an *unchanged* and a *changed* center pixel, we denote the number of inhomogeneous cliques as n^u_{hv}, n^u_{diag} and n^c_{hv}, n^c_{diag}, respectively. Inserting (7.25) and (7.26) into (7.24) results in

$$\frac{p(d|q=u) \cdot \frac{1}{Z} \exp\left\{ -\sum_{\mathcal{C}^0} \cdots \right\} \exp\left\{ -n^u_{hv}\gamma_{hv} - n^u_{diag}\gamma_{diag} \right\}}{p(d|q=c) \cdot \frac{1}{Z} \exp\left\{ -\sum_{\mathcal{C}^0} \cdots \right\} \exp\left\{ -n^c_{hv}\gamma_{hv} - n^c_{diag}\gamma_{diag} \right\}} \gtrless^u_c 1, \tag{7.27}$$

which can be simplified to

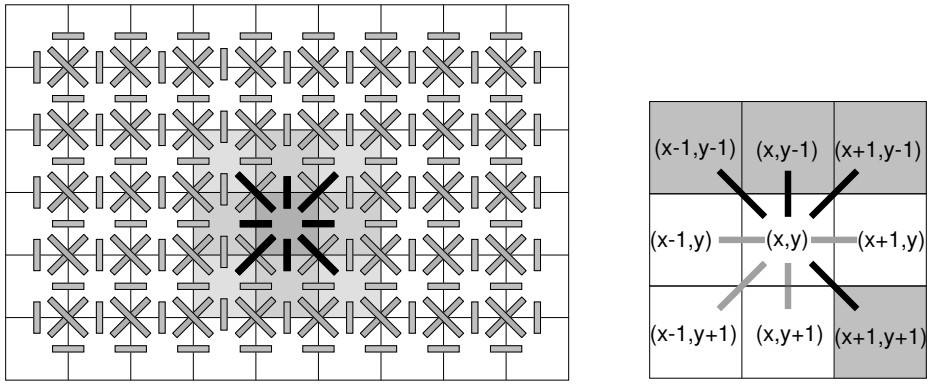
$$\frac{p(d|q=u) \cdot \exp\left\{ -n^u_{hv}\gamma_{hv} - n^u_{diag}\gamma_{diag} \right\}}{p(d|q=c) \cdot \exp\left\{ -n^c_{hv}\gamma_{hv} - n^c_{diag}\gamma_{diag} \right\}} \gtrless^u_c 1. \tag{7.28}$$

Inserting Gaussian distributions for $p(d|q)$ leads to

$$\frac{1/\sqrt{2\pi\sigma^2} \exp\left\{ -d^2/(2\sigma^2) - n^u_{hv}\gamma_{hv} - n^u_{diag}\gamma_{diag} \right\}}{1/\sqrt{2\pi\sigma_c^2} \exp\left\{ -d^2/(2\sigma_c^2) - n^c_{hv}\gamma_{hv} - n^c_{diag}\gamma_{diag} \right\}} \gtrless^u_c 1, \tag{7.29}$$

and after taking the logarithms, we obtain the final decision function

$$d^2 \gtrless^c_u 2\frac{\sigma_c^2\sigma^2}{\sigma_c^2 - \sigma^2} \left(\ln\frac{\sigma_c}{\sigma} + (n^c_{hv} - n^u_{hv})\gamma_{hv} + (n^c_{diag} - n^u_{diag})\gamma_{diag} \right). \tag{7.30}$$



(a) Local influence of a pixel.

(b) Homogeneous and inhomogeneous cliques.

Figure 7.13: *(a) The state of one pixel has only influence on the eight indicated clique potentials. (b) In this example configuration, there is one inhomogeneous vertical clique ($n_{hv} = 1$) and three inhomogeneous diagonal cliques ($n_{diag} = 3$).*

The right-hand side of this equation is effectively an adaptive threshold that depends on the local neighborhood. If the number of inhomogeneous cliques is the same, independent of the state of the center pixel ($n^c_{hv} = n^u_{hv}$ and $n^c_{diag} = n^u_{diag}$), the neighboring pixels have no influence on the decision and we obtain the special case

$$d^2 \gtrless^c_u 2\frac{\sigma_c^2\sigma^2}{\sigma_c^2 - \sigma^2} \ln\frac{\sigma_c}{\sigma} \qquad (\text{if } n^c_{hv} = n^u_{hv} \text{ and } n^c_{diag} = n^u_{diag}). \tag{7.31}$$

Otherwise, the threshold is shifted in either direction to bias the decision level. If, for example, the neighborhood has many unchanged pixels, n^u_{hv} and n^u_{diag} will be low, leading to a higher threshold. This again will bias the decision for the center pixel towards *unchanged*.

7.4.3 Extension to color images

Up to this point, the MRF-based approach was described for luminance images only. However, it is not difficult to extend this approach to color images, assuming that the color distribution is a multi-variate Gaussian distribution (as done previously when introducing the Mahalanobis distance metric in Section 7.2.2). When considering again images in the YUV color-space, we can assume that the noise variance in the luminance channel is σ^2, while it is $\beta^2\sigma^2$ in the color channels. Consequently, the covariance

matrix of color pixel-differences $\mathbf{d} = |\mathbf{I}_t^{YUV} - \mathbf{I}_B^{YUV}|$ is a diagonal matrix given as $\mathbf{S} = \sigma^2 \text{diag}(1, \beta^2, \beta^2)$. This enables to write the probability density as a multi-variate Gaussian

$$p(\mathbf{d}|q = u) = \frac{1}{\sqrt{(2\pi)^3 \sigma^3 \beta^2}} \exp\left\{ -\frac{1}{2} \mathbf{d} \mathbf{S}^{-1} \mathbf{d}^\top \right\}. \tag{7.32}$$

Inserting this into Eq. (7.28) results in

$$\frac{\sqrt{(2\pi\sigma_c)^3 \beta^2} \exp\left\{ -\frac{1}{2} \mathbf{d} \cdot \mathbf{S}^{-1} \cdot \mathbf{d}^\top - n_{hv}^u \gamma_{hv} - n_{diag}^u \gamma_{diag} \right\}}{\sqrt{(2\pi\sigma)^3 \beta^2} \exp\left\{ -\frac{1}{2} \mathbf{d} \cdot \mathbf{S}^{-1} \cdot \mathbf{d}^\top - n_{hv}^c \gamma_{hv} - n_{diag}^c \gamma_{diag} \right\}} \gtrless_c^u 1, \tag{7.33}$$

which finally leads to the decision function

$$(d_M^{YUV})^2 \gtrless_u^c 6 \frac{\sigma_c^2 \sigma^2}{\sigma_c^2 - \sigma^2} \left(\ln \frac{\sigma_c}{\sigma} + (n_{hv}^c - n_{hv}^u)\gamma_{hv} + (n_{diag}^c - n_{diag}^u)\gamma_{diag} \right), \tag{7.34}$$

where d_M^{YUV} is the Mahalanobis distance. Note that this decision function differs from Eq. (7.34) only in the changed constant at the right-hand side.

7.4.4 Optimization algorithm

The optimization problem to find the segmentation mask with the maximum a-posteriori probability (MAP) for the described Markov field model is difficult, since the number of variables equals the number of pixels in the image. A number of algorithms has been proposed to find an approximate solution to maximize the joint probability (see [114] for a comparison). We have chosen to use the *iterated conditional modes* (ICM) algorithm, which was initially described in [9]. This algorithms runs several passes over the image, where each pixel is assigned the most probable label, considering its current local neighborhood. Specifically, this means that we iterate through all pixels in the segmentation mask image and set the label of a pixel according to Eq. (7.30). When all the labels have converged, the algorithm is stopped. It is assured that the algorithm converges, because each modification of a pixel increases the joint probability of the Markov field.

Efficient implementation with a queue of boundary pixels

Fortunately, the algorithm complexity can be reduced, since most of the pixels will not change their label from one iteration to the next. As a consequence, we maintain a queue of pixels that should be checked for modification. In each iteration, we take a new pixel out of the queue until the queue is empty. If we check a pixel and it does not change its label,

(a) Frame 6. (b) Frame 80.

Figure 7.14: *Results of the MRF-based segmentation. (Compare to Figs. 7.9 and 7.3.)*

the neighboring pixels will not be affected. However, if the pixel label is changed, this may also influence the label of the neighboring pixels. Consequently, when the label of a pixel is changed, we put the coordinates of the eight neighboring pixels into the queue to check them for modification in a later iteration. This algorithm will converge, because we know that the number of label changes are limited and the queue does only grow in size as long as labels are modified.

Initialization

Up to now, the inner loop of the segmentation has been described. In each step, this iterative process improves the segmentation mask from the previous step. However, we still need a segmentation mask to start with. Taking a completely transparent or completely opaque segmentation mask does not converge to a good solution, because the shape prior effectively inhibits any change.

One solution is to start with a very weak shape prior with parameters $\gamma_{hv}, \gamma_{diag} \approx 0$, and to increase these parameters gradually. Note that if both are set to zero, we obtain the pixel-based decision function Eq. (7.31).

An easier approach is to start with the result of one of the previous segmentation algorithms. In our system, we use the χ^2 significance test on 5×5 windows with the Mahalanobis distance. After the initialization of the segmentation mask, the queue of pixels is initialized with pixels to be checked. To limit the number of pixels in the queue, we only fill in pixels along the object boundary.

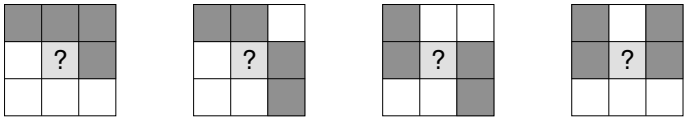


Figure 7.15: *Patterns or neighborhood pixels, for which* $n^c_{hv} = n^u_{hv} = n^c_{diag} = n^u_{diag} = 2$. *In these cases, the label for the central pixel is chosen only based on the difference value at the central pixel.*

7.4.5 Evaluation

The result of the Markov field based segmentation for two example frames of the *stefan* sequence is depicted in Fig. 7.14. When comparing these results to the results in Figs. 7.9 and 7.3 that were obtained with the previous segmentation algorithms, it can be concluded MRF-based segmentation yields more accurate results. The quality of the interior of the object is comparable to the χ^2 significance test, but the algorithm does not yield the disadvantageous aura effect along the object boundary.

Parameter selection

The MRF-based segmentation algorithm includes two parameters $\gamma_{hv}, \gamma_{diag}$ that control the influence of the shape prior. In our final system, we used $\gamma_{hv} = 2.5$ and $\gamma_{diag} = 1.25$, which are the values proposed in [2]. However, we noticed that an increase of these parameters does not have much effect on the obtained segmentation mask.

Surprisingly, we could even obtain comparable results with setting $\gamma_{hv} = \gamma_{diag} \approx \infty$. If we examine Eq. (7.34), it can be derived that setting these two parameters to very high values can force a pixel to foreground or background just because of the labels of its neighbors. Only for the case that $n^c_{hv} = n^u_{hv} = 2$ and $n^c_{diag} = n^u_{diag} = 2$, the shape prior is zero and the decision function reduces again to Eq. (7.31). The cases for which this happens are depicted in Fig. 7.15. In all other cases, the label for the center pixel can be simply derived from the pixel neighborhood. This simplified algorithm is not used in our implementation, but it can be advantageous for a hardware implementation, for which it is easier to implement.

7.5 Sources of errors and robustness improvements

In the previous section, it was assumed that scenes were recorded with a static camera, or that the camera motion was compensated by taking background pictures from the synthesized background mosaic. At first view, this should assure that corresponding pixels are co-located in the foreground and the background image. However, in practicer, three problems occur:

- **Registration errors.** When the background view is reconstructed from a synthesized panorama, small inaccuracies in the motion-model can occur. Even though this is usually much less than a pixel distance, it can cause difficulties along strong edges. If there is a large difference in brightness across the edge, a tiny inaccuracy in the motion model or aliasing in the input video can cause a large value in the difference image.

- **Interpolation errors.** In the background reconstruction process, the input images are resampled when they are warped onto the background image. This resampling involves bi-linear interpolation that results in some image blurring. To obtain the camera-motion compensated image, a second resampling is carried out to obtain the current background view from the synthesized background overview image. The consequence of these two resampling steps is that a blurred background reconstruction is compared with the sharp input image. Especially within fine texture, this can lead to false positives.

- **Motion blur.** For the case that the input sequence comprises fast camera motion, the image is not only transformed geometrically, but it can also show motion blur. The amount of blur can differ throughout the sequence, but this cannot be reflected in the synthesized background. Consequently, errors occur, e.g., when comparing a motion-blurred input image with a sharp background image (see Fig. 7.16). However, note that motion blur itself does not cause segmentation errors provided that the same motion blur occurs both in the input image and the background image.

Our general concept to approach these problems is to identify those parts of the image, that have a high error risk. We then modify the previously discussed segmentation algorithms to exclude the identified pixels of high risk from the difference image computation. We will present solutions to improve the robustness against the first two problems in the remainder of this section. First, we describe the concept of using maps of *risky pixels*

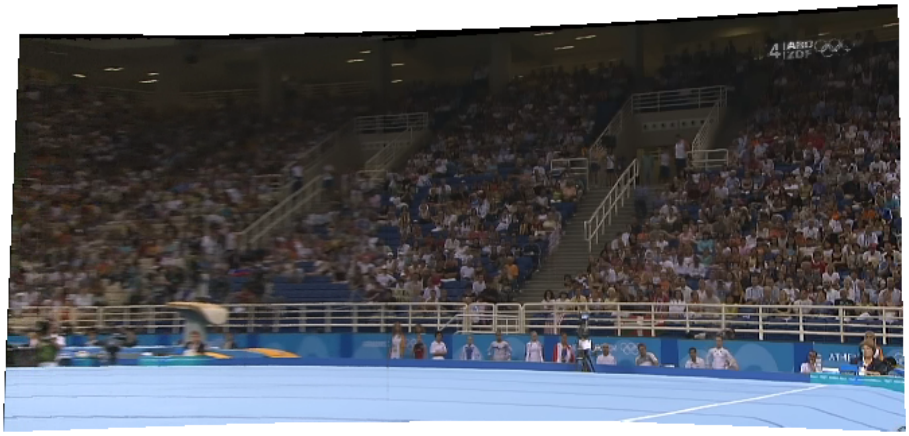


Figure 7.16: *A background image that shows different strengths of motion blur, because it combines slow and fast camera motion in a single image. At the right side, camera motion was slow, so that the image is sharp. At the left side, the camera motion is much faster, resulting in significant image blurring. Especially note the difference between the two stair rails in the center area.*

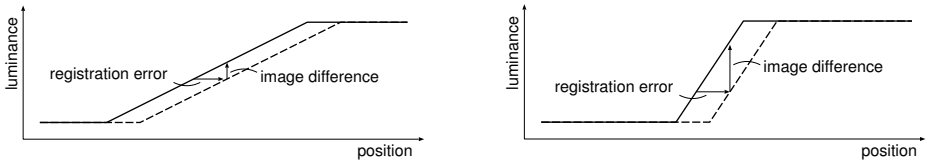
and how to obtain these maps. Afterwards, the modifications to the segmentation algorithms to consider these maps in the object segmentation are presented.

7.5.1 Map of misregistration risk

The previous segmentation algorithms assumed that the difference image comprises only camera noise and differences to foreground objects. However, the camera-motion compensation itself can lead to misregistration and resampling errors that also appear in the difference image. This kind of error depends on the local contrast in the pixel neighborhood and has the largest value at strong edges (see Fig. 7.17). This is not in accordance with the assumption that the variance of noise is independent of the position in the image.

As a first solution [47], we investigated to change the image difference measurement. Instead of using the direct luminance difference $|I_B(x,y) - I_t(x,y)|$, we compensated for the expected misregistration along edges by dividing by the luminance gradient in the background image, leading to

$$d(x,y) = \frac{|I_B(x,y) - I_t(x,y)|}{||\nabla I_B(x,y)||}. \tag{7.35}$$



(a) Low-contrast. (b) High-contrast.

Figure 7.17: *Two edge-profiles, with (a) low-contrast, (b) high-contrast edge. In the high-contrast case, a misregistration of the edges induces a larger luminance error than in low-contrast cases.*

Even though this approach helps in reducing the misregistration errors, we observed problems in cases where the background is textured and the object has uniform color. In these cases, the gradient in the background is high, which consequently leads to a lower total error. This is not optimal, because we know that the observed region cannot be background as the object shows no texture. This observation leads to a different approach, which explicitly identifies a pixel to be probably affected by misregistration errors. More specifically, we identify a pixel as *risky* if there is a high-contrast edge in the background image, and there is also a high-contrast edge in the foreground image at the same position. To detect steep edges, we simply apply a threshold onto the gradient magnitudes. Since a misregistration only results in a segmentation error when there are edges is both images, we combine the detected edges into the map $R_M(x, y)$ of misregistration risk with the definition that

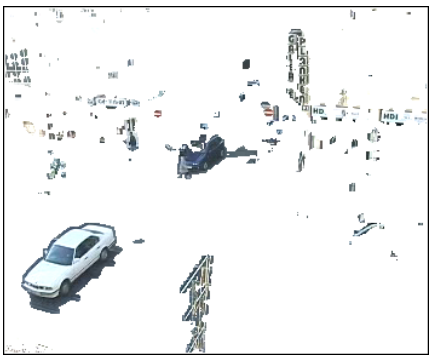
$$R_M(x,y) = \left(||\nabla I_B(x,y)||^2 > \tau_m\right) \wedge \left(||\nabla I_t(x,y)||^2 > \tau_m\right), \qquad (7.36)$$

where τ_m is a fixed threshold for edge detection. An example scene with an particularly strong misregistration effect is shown in Fig. 7.18. Note that in this example, the camera was assumed static, so that a static background image was used for the segmentation. However, the camera seems to have moved a little bit, probably because of wind. This tiny motion results in the observed misregistration.

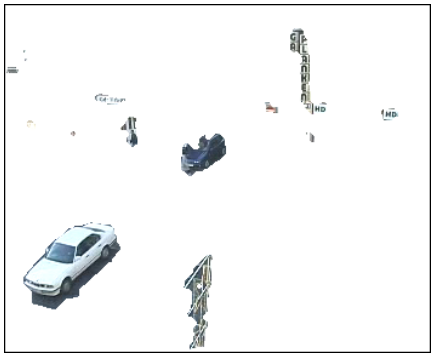
7.5.2 Map of interpolation errors

The second problem is introduced by the image warping in the camera motion compensation. Remember that during the background reconstruction, the input frames are warped into the reference coordinate system. In this process, the target pixel values are interpolated from the input pixels using bi-linear interpolation. A second interpolation step is carried out to

(a) Input image.



(b) χ^2 test without misregistration reduction.



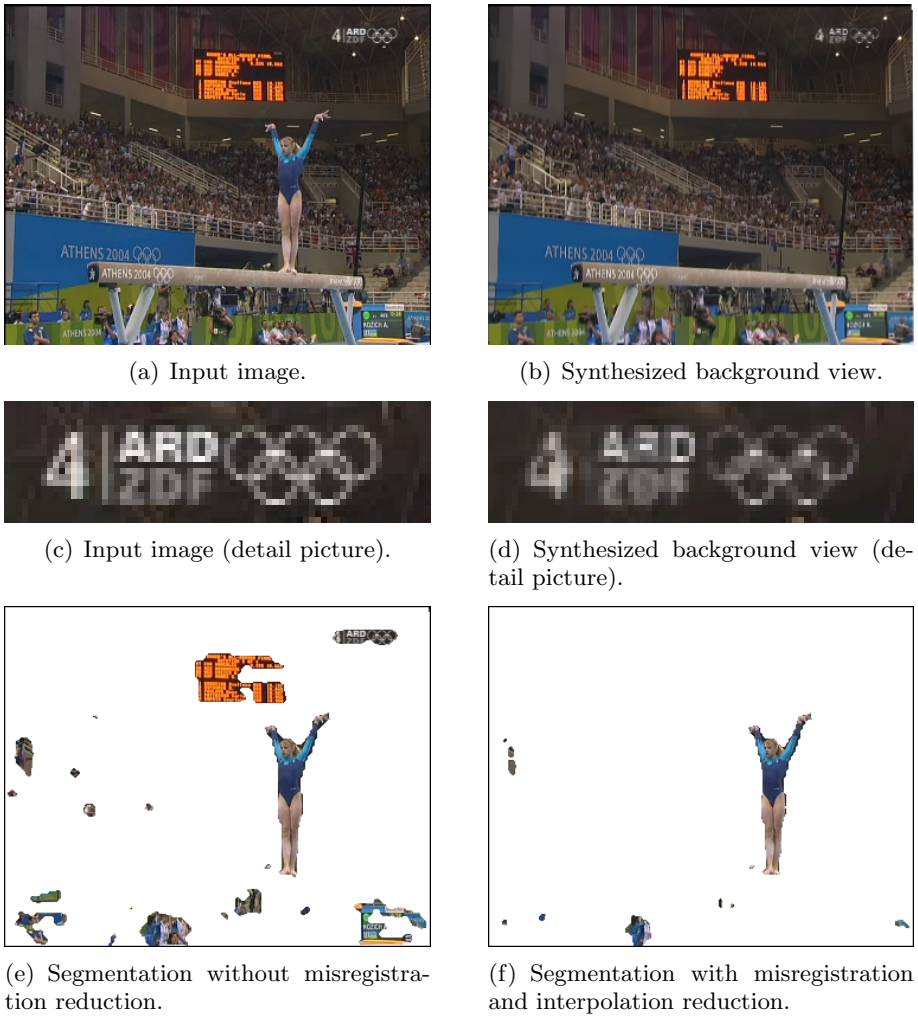
(c) MRF segmentation without misregistration reduction.

(d) MRF segmentation with misregistration reduction.

Figure 7.18: *Example for misregistration, following from a lightly moving camera caused by wind.*

reconstruct the current camera view from the background reconstruction. In total, this means that this interpolation is applied twice, such that a reconstructed background view looks blurred, compared to the original input frame. Suppose that we now take the difference to the original input, large differences can occur just because of this blurring effect (see Fig.7.19). We approach this problem in a comparable way as with the misregistration errors.

To simulate the blurring of the two interpolation steps, we apply a simple low-pass filter to the current input image. This blurred image is compared with the input image and a risk of interpolation error is detected when the difference exceeds a threshold τ_i. This defines the map of inter-



(a) Input image.

(b) Synthesized background view.

(c) Input image (detail picture).

(d) Synthesized background view (detail picture).

(e) Segmentation without misregistration reduction.

(f) Segmentation with misregistration and interpolation reduction.

Figure 7.19: *Example for interpolation error reduction. When comparing the input image (a) and the reconstructed background view (b), it is visible that the background view is slightly blurred. Since this blur is only visible at the pixel level, detail views are provided in (c) and (d). Segmentation without misregistration/interpolation reduction erroneously includes regions like the score display or the logo. These regions are removed by enhancing the segmentation algorithm with misregistration and interpolation risk maps.*

polation risk as

$$R_I(x,y) = true \quad \text{iff} \quad \left| I_t(x,y) - \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \otimes I_t(x,y) \right| > \tau_i. \quad (7.37)$$

A different approach to eliminate the interpolation problem that could be further studied in future research is to avoid the double interpolation by reorganizing the segmentation process. The current background subtraction data-flow is shown in Fig. 7.20(a). Note that the actual background subtraction is performed in the input coordinate space. However, it is also possible to carry out the background subtraction in the coordinate system of the background image as shown in Fig. 7.20(b). The advantage of this would be that the input image is transformed exactly once. Moreover, in both cases, for reconstructing the background and also for carrying out the background subtraction, exactly the same transformation is used. Consequently, both images will show the same interpolation artifacts. When comparing the two images, this means that both interpolation artifacts cancel out. However, the disadvantage of this approach is that we obtain the segmentation mask in the background coordinate system and we may require an additional step to transform the obtained mask to the image coordinate system again.

7.5.3 Integrating risk maps into the segmentation process

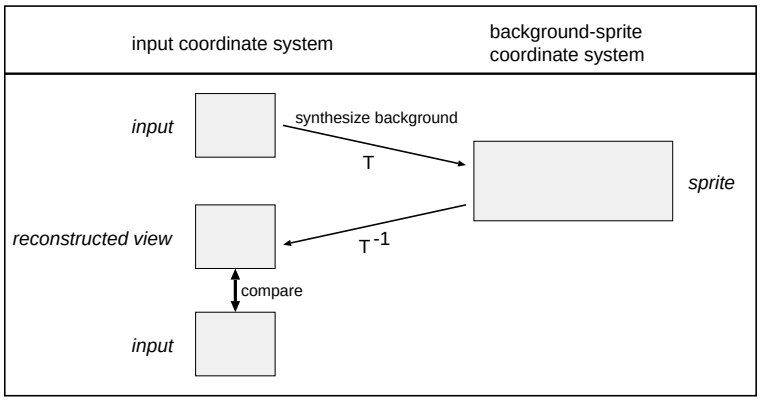
Once the pixels are obtained for which a possible unreliability is identified in the difference image, the previously discussed algorithms can be modified such that these unreliable pixels are excluded from the computation.

χ^2 significance test

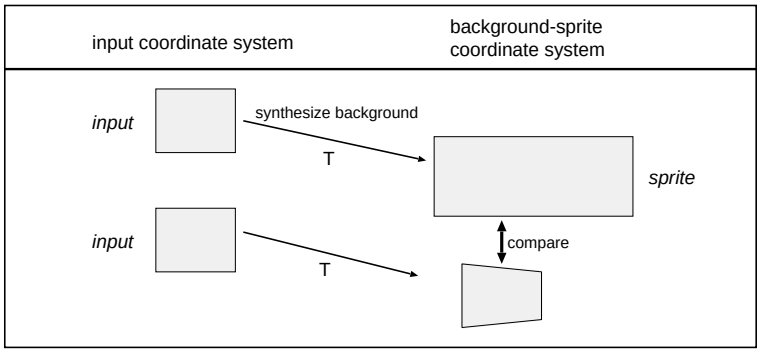
The central operation of the χ^2 significance test is to sum the squared difference values in a small neighborhood window. The change when integrating the risk maps is that pixels that we classified as risky are not included in the sum. Note that the degrees of freedom in the χ^2 test are not reduced by the number of removed pixels.

MRF-based segmentation

The MRF-based segmentation uses Eq. (7.30) as the decision function. If the center pixel is classified as risky, we base the segmentation only on the



(a) Asymmetric interpolation errors.



(b) Symmetric interpolation errors.

Figure 7.20: *Two possible data-flows for carrying out the background subtraction. (a) The background is transformed to the current camera view, background subtraction is performed in the input coordinate space. (b) The input is transformed to the background coordinate space, background subtraction is carried out in the background coordinate space.*

neighborhood information, i.e., the shape prior. This leads to the decision function

$$0 \gtrless_u^c (n_{hv}^c - n_{hv}^u)\gamma_{hv} + (n_{diag}^c - n_{diag}^u)\gamma_{diag} \tag{7.38}$$

if the considered pixel is risky, otherwise Eq. (7.30) is used without modification.

7.6 Postprocessing the object mask

When performing a subjective evaluation of the segmentation masks, we can discover several obvious errors, which on a closer look are only considered obvious because we recognize the object and can immediately judge that a specific mask cannot be correct. When the type of the observed objects are known, various heuristic rules can be derived for postprocessing the segmentation mask to remove these errors. A selection of these rules seem to be true in many practical situations, so that they can be provided as general postprocessing filters in a multi-purpose segmentation system.

7.6.1 Filling holes in the object

Most natural or man-made objects have no holes (in the sense of a torus, not in the sense of a coffee cup). The majority of the objects are either compact (like cars, balls), or they are star- or tree-shaped (like humans, animals, flowers). Consequently, small holes in objects are most likely segmentation errors and we can usually improve the segmentation by closing the holes in objects. The most frequent case in which this heuristic fails is when articulated objects like humans build loops with their limbs (see Fig. 7.21). Hence, to limit the impact of the modification, we put a small threshold τ_h on the size of the hole and close it only if the size if below this threshold.

7.6.2 Heuristics for removing clutter in the mask

Another frequent case is that we know in advance to expect only relatively large objects in the image. Small regions in the mask are likely either just errors or small changes in which we are not interested. One example of the latter is a recording of sports, where the actors are visible in the front and the audience is located at the background. In this case, not only the actor will move, but also the audience has little motion. Usually, this motion irrelevant for most applications. Hence, we also provide a filter to remove regions if their size is below a small threshold s_o. If it is known that the scene contains only one object, we can even exclude all regions except the largest one.

However, in the case that there is only one object of interest, excluding all regions but the largest one can result in incomplete segmentation masks. This is the case if parts of the object are disconnected from the main region. One example case is shown in Figure 7.21(a), where part of one leg has no connection to the body. If we would only keep the largest region, this part of the leg would be lost. To prevent this behaviour, we modify the single-object heuristic to include also regions which are close to the largest region

(a) Unprocessed segmentation mask. (b) Postprocessed mask.

Figure 7.21: *Postprocessing of segmentation masks. (a) Output from MRF-based segmentation. (b) Output of mask post-processing. Small clutter regions are removed and the disconnected object region (the leg) is added to the main object mask.*

and which are not too small. The algorithm becomes as follows.

1. Find the largest region in the segmentation mask.
2. Grow this largest region by d_o pixels. This is done with d_o dilate operations.
3. If another region is touched during the dilate operations, check if the size of this region exceed the minimum size s_d.
4. If the region size is larger than s_d, add this region to the final segmentation mask.

The result of this postprocessing heuristic is shown in Fig. 7.21(b).

7.7 Overview of the segmentation process

This section summarizes which of the previously described algorithms are used in our segmentation system and how they are connected. The data-flow of the complete background-subtraction module is shown in Figure 7.22. The input of the module is the original input video stream and a synchronous stream of reconstructed background images. These background images are obtained from the synthesized multi-sprite background model by extracting the current camera view from the multi-sprite. This means that the background sequence shows the same content as the input sequence, but without the foreground objects.

Processing step	Parameters
χ^2 significance test	$w = 5 \times 5$, $\sigma = 8$, $\alpha = 10^{-6}, \beta^w = 0.2$
MRF optimization	$\sigma = 8$, $\sigma_c = 40$, $\gamma_{hv} = 2.5, \gamma_{diag} = 1.25$
Fill object holes	$s_h = 20$
Remove clutter	$s_o = 500$ or largest only, $d_o = 15$, $s_d = 150$
Misregistration risk map	$\tau_m = 200$
Interpolation risk map	$\tau_i = 12$

Table 7.1: *Parameters used in our background-subtraction module.*

The background-subtraction module first computes the risk map, which identifies for which pixels the segmentation is possibly unreliable. Taking this risk map into account, a χ^2 significance test is carried out to obtain an initial segmentation mask. The Mahalanobis distance in the YUV color-space is employed as distance measure. This algorithm already yields a good segmentation mask, but it still shows the aura effect along the object boundary and it usually also contains small clutter. Both problems are reduced in the successive refinement based on the Markov random field model. This algorithm also takes the risk map into account.

The final two postprocessing steps comprise filling of object holes and removing small clutter regions from the segmentation mask. We consider these two steps as optional, since their integration may depend on the application. Small regions that remain after the MRF optimization step usually correspond to actual changes in the image. However, many of these small changes are often unimportant actions.

The background-subtraction module is influenced by many adjustable parameters. The parameter values selected in our system are summarized in Table 7.1. Most of these parameters are not critical and can be changed without strong effects on the output. The most sensitive parameter is σ, for which we obtained the best results for the range $\sigma = 6, \ldots, 16$, depending on the quality of the input sequence. For compressed input sequences, a higher σ of around 12 proved to be most suitable.

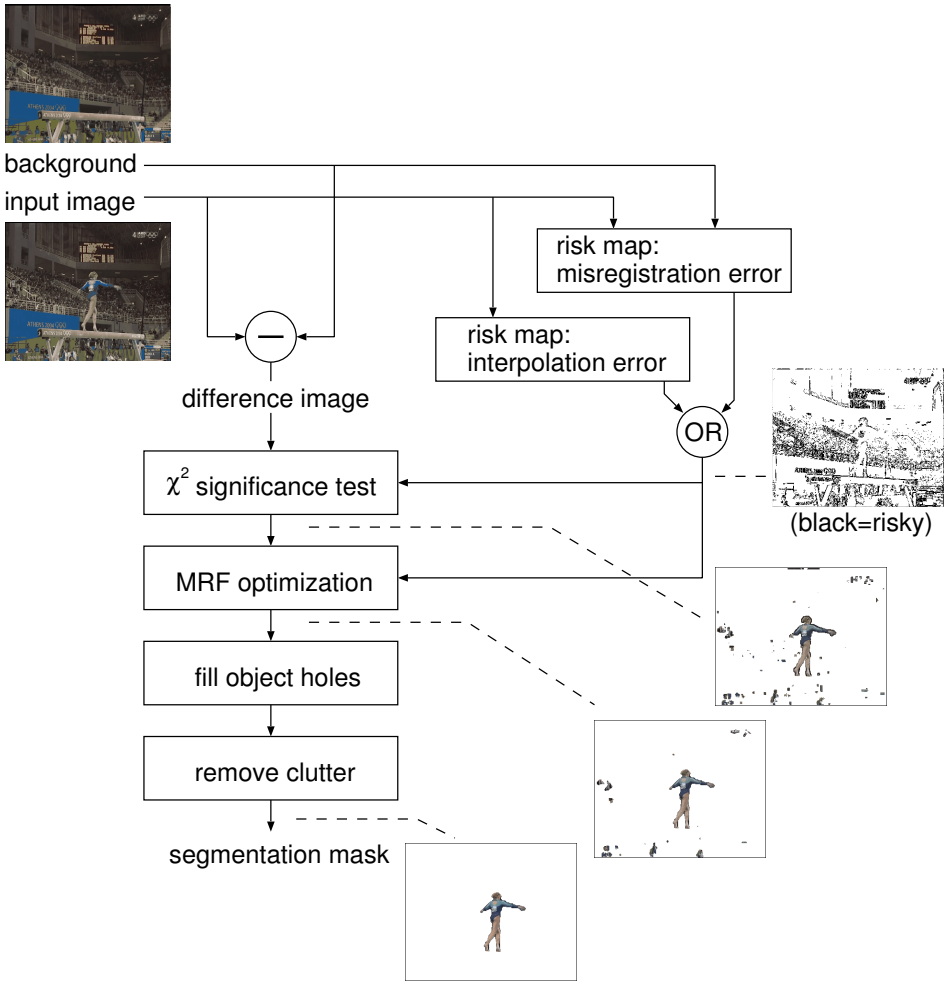


Figure 7.22: *Architecture of the background-subtraction module.*