most of the information relevant to the task-domain or application scenario. In this setting model-based interpretation of images is an *optimization problem*.

A basic problem occurs if a single static image or a volume-sequence of images is to be interpreted, Examples are single MR or CT images, or volume-equences of MR or CT images. To solve this problem processing often proceeds in three phases. At first, a segmentation of the image into segmentation objects like lines, regions, and vertices is computed. This topic is treated in Section 3.1. Then objects or in general semantically meaningful components in the image are classified as described in Section 3.4. Finally, in the third phase the meaning of configurations of objects is interpreted in the context of the task domain; this is described in Section 3.5.

If a time-sequence of images is to be interpreted, some specializations may be necessary or advisable due to the real-time requirements. In a time-equence of images new objects may become visible, visible objects may move out of the field of vision, the same object may be visible on many image frames, and an object needed to achieve some task may not yet be visible. In such cases it is not advisable to treat every image frame completely independent of the other ones. Rather, one should distinguish the *phases* of

- *initializing* the processing by making a full and detailed analysis of usually a few consecutive images, for example, using the techniques described in Section 3.4 and Section 3.5,
- *tracking* of recognized objects as long as they are visible, see e.g. Section 2.4 and Section 3.2.2,
- *detecting* new objects entering the field of vision, that is, being *attentive* to changes in the image content, see e.g. Section 2.4 or Section 7.4,
- *actively searching* for objects which are important for the fulfilment of the task as described in Section 3.6.

3.1 SEGMENTATION

D. PAULUS

Many applications of image processing and image analysis have been outlined in the previous chapters of this volume. More examples can be found in medical, industrial, military or geographical applications. We saw that simple *classification* problems can be solved with statistical methods and that analysis and understanding of complex patterns generally requires knowledge about the particular task domain.

But even knowledge-based image analysis systems tailored to a certain problem usually have some components which operate on images independently from the specific task. These components are commonly referred to as the image preprocessing stage. Typical operations are a variety of filters, geometric transformations and corrections, extraction of features, like e.g. gray-level histograms.

Preprocessing is not the only part of an image analysis system which can be designed without knowledge about the task domain. The analysis of complex scenes usually requires that the image is split into simpler components.

This process is called the *segmentation* of an image. Segmentation may in some cases be directed by the knowledge-base, but may in many cases as well be initially performed independently of the specific task.

3.1.1 Introduction

The term 'segmentation' is used in many publications with slightly different meaning. For a definition the following two statements are helpful:

Description of an image

"The description of an image is its decomposition or segmentation into simpler constituents (or pattern primitives or segmentation objects) and their relations, the identification of objects, events and situations by symbolic names, and if required, an inference about its implications for the task domain." [508, p.6]

Analysis of an image

"When analyzing (complex) patterns each pattern is given an individual symbolic description." [508, p.5]

Image segmentation thus is a process that transforms the image into a set of simpler constituents. The types of the objects detected depend highly on the algorithm used. Three major classes of algorithms are

- region-based segmentation which searches for objects fulfilling a certain homogenity criterion,
- line-based systems which detect discontinuities in the image function, and
- point-segmentation which identifies interesting points.

The objects detected in the segmentation may be represented in various ways as indicated in Table 3.1. One object in the scene may be represented by more than one representation at the same time. An example is a line which is detected as a chain code and is approximated by a spline.

An initial segmentation requires little if any knowledge about the structural properties of patterns. It results in a *segmentation object* which has certain *attributes*. A segmentation object is an initial symbolic description of an image. Typical attributes of segmentation objects are the location in two-dimensional image coordinates or in three-dimensional world coordinates, gray-level and color, texture, motion, depth, surface normal, shape and reliability or certainty of the detection. Attributes and relations between them are used to represent data driven evidence of parallelism, symmetry, curvature, and collinearity.

Since color images are now provided by most CCD cameras, the ideas of image segmentation which were introduced for gray-level images in the literature, have to be extended to multi-channel images. This is the case for line detection, region segmentation, as well as the identification if points of interest. In the following we will outline the principles of segmentation in their original form. If extensions to color images already exist, we will cite and outline them as well. If no such extension is known yet but is feasible, we will propose it.

Geometric object	Representation
Region	Contour-line
	Characteristic function
	Quad tree
	Run length code
Lines	Chain code
	Polygon
	Spline
Point	Coordinate pair

Table 3.1. Representations of results of the segmentation for 2D images

The organization of the section is as follows: Region segmentation of intensity images is introduced in [338]; we survey this topic in Section 3.1.2. Edge detection and line segmentation is formalized in [469, 106]; we describe these approaches in Section 3.1.3. Detection of corners, vertices, and interesting points is introduced in Section 3.1.4. These segmentation strategies are applied to range images in Section 3.1.5. The problem arises, how to represent these symbolic data; formalisms and data structures for this purpose and implementation issues are covered in Section 3.1.6. In Section 3.1.7 we refer to literature on performance valuation, which is crucial for image segmentation.

Many other ideas exist for image segmentation, e.g., region segmentation using texture information. Due to space limitation, we omit these subjects and refer to the literature, e.g. [190].

3.1.2 Region Segmentation

In the following we describe two region segmentation algorithms and mention a third approach. The first method extends an algorithm which was introduced in [191]. It extends and uses the 'split-and-merge' algorithm of [338], to color differences. The second is based on image morphology.

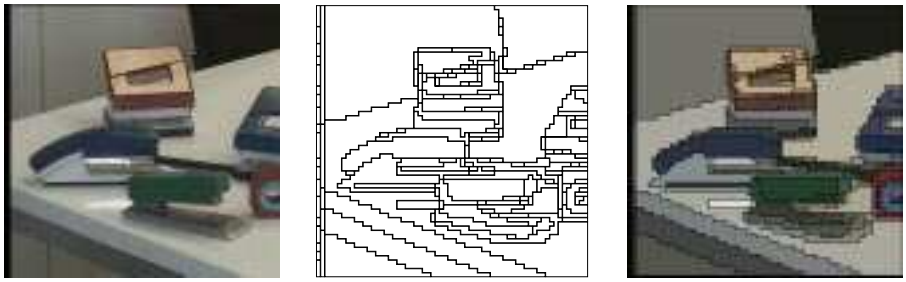
For the 'split-and-merge' algorithm we assume that we have a quadratic color image with a size which is a power of two; we organize the pixels initially as a quadtree which is a hierarchical data structure for efficient region representation.

The color mean vector $\mu_{i,j}$ in the square window

$$\boldsymbol{W}_{i,j} = [\boldsymbol{w}_{\mu\nu}]_{\mu,\nu=1,\cdots,n} = [\boldsymbol{f}_{a,b}]_{a=i,\cdots,i+n,b=j,\cdots,j+n} \tag{3.1}$$

of size $(n+1) \times (n+1)$ is computed from the color values

$$\boldsymbol{f}_{i,j} = (r_{ab}, g_{ab}, b_{ab})^T \tag{3.2}$$



(a) (b) (c)

Figure 3.1. Example of region segmentation: color image (a), contours (b), contours overlayed to regions (c).

at positions $(a, b)^T$ within the window $\boldsymbol{W}_{i,j}$ by an average operation on the color channels:

$$\mu_{i,j} = \frac{1}{n^2} \sum_{a=i}^{a=i+n} \sum_{b=j}^{b=j+n} \boldsymbol{f}_{a,b}. \tag{3.3}$$

The variance is computed as the sum of variances in the color channels in [191]. To generalize this measure to color spaces other than RGB, the variance ${\sigma^2}_{i,j}$ in the square $\boldsymbol{W}_{i,j}$ was defined in [169] as follows, where $d(\mu_1, \mu_2)$ denotes the distance between μ_1 and μ_2

$${\sigma^2}_{i,j} = \frac{1}{n^2} \sum_{a=i}^{a=i+n} \sum_{b=j}^{b=j+n} \left(d(\boldsymbol{f}_{a,b}, \mu_{i,j})\right)^2. \tag{3.4}$$

If the image inside a window $\boldsymbol{W}_{i,j}$ is sufficiently heterogeneous, i.e., if the variance ${\sigma^2}_{i,j}$ is greater than a threshold θ_{v}, then we split the square into four sub-squares. The minimal size of the squares is a parameter of the algorithm which can be used to tune the speed and the desired accuracy and noise sensitivity of the segmentation.

During the subsequent merge phase, four adjacent squares which have a common direct ancestor in the quadtree are merged if their color distance is below a threshold θ_{m}. In a final grouping stage, similar regions are merged if their color distance is below a threshold θ_{g}. The result is no longer a quadtree. A typical sequence of intermediate results of the processing steps is shown in Figure 3.1.

The so-called *watershed transform* [730] has recently gained increasing interest in computer vision. This morphological image segmentation method can in some cases considerably enhance image analysis.

The watershed transform idea can be easily explained by help of a geographical imagination. Suppose a drop of water is falling on a topographic relief—it will run down until it reaches a local minimum. The influence zones of those local minima are called catchment basins. As the water level increases and two basins are going to flow together, a dam has to be built. The dam separates the local minima against each other and can be seen as a watershed (see Figure 3.2 (right)).

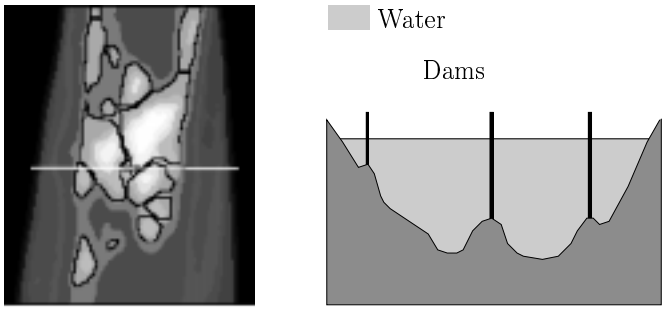


Figure 3.2. One-dimensional effect of the immersion simulation on a gray-level profile (right); cut from a real thermographic image showing the skin temperature on a human forearm (white line on the left), implementation according to Baxes [47]. Immersion depth 33.9^oC.

The two general ideas for the watershed transform are the *global* and the *local approach* to watershed segmentation. In the local approach, decision about further flow of water is based on the neighborhood of a pixel. In the global approach, all pixels are affected by the flooding. First we invert the image and—figuratively spoken—we put holes into each local minimum. Now the mountains are immersed into water which passes through the prepared holes. Whenever two basins unite, a dam is built (see Figure 3.2 (right)). The computed contour lines of these dams surround the areas searched for. During the flooding process, only the points on the border of the basin are required for computation. This contour based algorithm allows for an efficient implementation with linear complexity when these points are queued and inspected in parallel [730]. The immersion can be stopped at a certain level or can be continued until the highest mountain is covered by water and only dams remain visible. Partial flooding can be used to separate objects from the background. This strategy has been applied to segment thermographic images in a medical application [528].

Vincent [730] describes the watershed transform as a special case of a morphological operation. For segmentation purposes, the watershed transform can be treated as a region growing algorithm. Region segmentation requires the definition of a criterion for homogeneity [508]. Catchment basins are homogeneous in the sense that they contain exactly one local minimum together with the related influence zone.

3.1.3 Edge and Line Detection

Literally hundreds of edge detector schemes were proposed in the literature in the past. They can be categorized into three types:

- Gradient based approaches which compute discrete approximations of the partial derivatives f_x and f_y of the image function f.
- Edge masks which filter the image locally by a mask modeling a particular edge type and orientation.

Figure 3.3. Masks for Sobel (left) and Prewitt (right) operator. Masks on the left: $\boldsymbol{f}_x$, masks on the right: $\boldsymbol{f}_y$. Note that these masks may be flipped with respect to other literature since we choose the origin of the coordinate system on the left top.

- Local parametric models which approximate the image locally by a function modeling the edge.

We briefly introduce gradient methods and outline an optimal edge detector which uses a parametrized edge model. Edge operators create an edge image in which edges are combined to lines. These lines can be further appoximated to geometric primitives such as straight line segments or circular arcs. We present results of these processing steps.

Parametric models which approximate the image function as well as edge modeles have been proposed for edge detection [508, 529]. In the following we will briefly show methods based in the intensity gradient.

Two of the most common operators for gradient computation in gray-level images are the Sobel operator and the Prewitt operator. These linear operations can be computed by a convolution of the input image with the masks shown in Figure 3.3. In [151] it is shown that the Sobel mask is an approximation of the first derivative. The same idea can be applied to color images in a straight forward manner using color vectors instead of intensity pixels. Edge detectors result in an edge image which contain a matrix of edge elements. These edge elements may be either a representation of the intensity gradient, or—which is more common—an encoding as edge strength f_{e} and edge orientation f_{o}; edge orientation is perpendicular to the edge gradient and the strength is a measure computed from the norm of the gradient.

In many cases which are of practical relevance, color information does not considerably increase the quality of edge detection, although color region segmentation has been quite successful e.g. for the segmentation of traffic signs. This is due to the fact that most color edges for real objects are also conceivable in the gray-level equivalent; color regions can use a homogeneity predicate which is not based on intensity, e.g., only chrominance. Results of edge detection on the image Figure 3.1 (a) are shown in Figure 3.4; the result of the Sobel operator on the gray-level input image is shown on the left; the center image shows the edge strength for the color-Sobel operator. On the right we have the edge orientation image computed from the gray-level operator.

Several gradient operators are listed in [508, 529]. We now assume that the horizontal derivative $\boldsymbol{f}_x$ and the vertical derivative $\boldsymbol{f}_y$ can each be computed by a convolution of the image $\boldsymbol{f}$ with a mask:

$$\boldsymbol{f}_x = \boldsymbol{f} \star \boldsymbol{G}_x, \tag{3.5}$$

$$\boldsymbol{f}_y = \boldsymbol{f} \star \boldsymbol{G}_y. \tag{3.6}$$

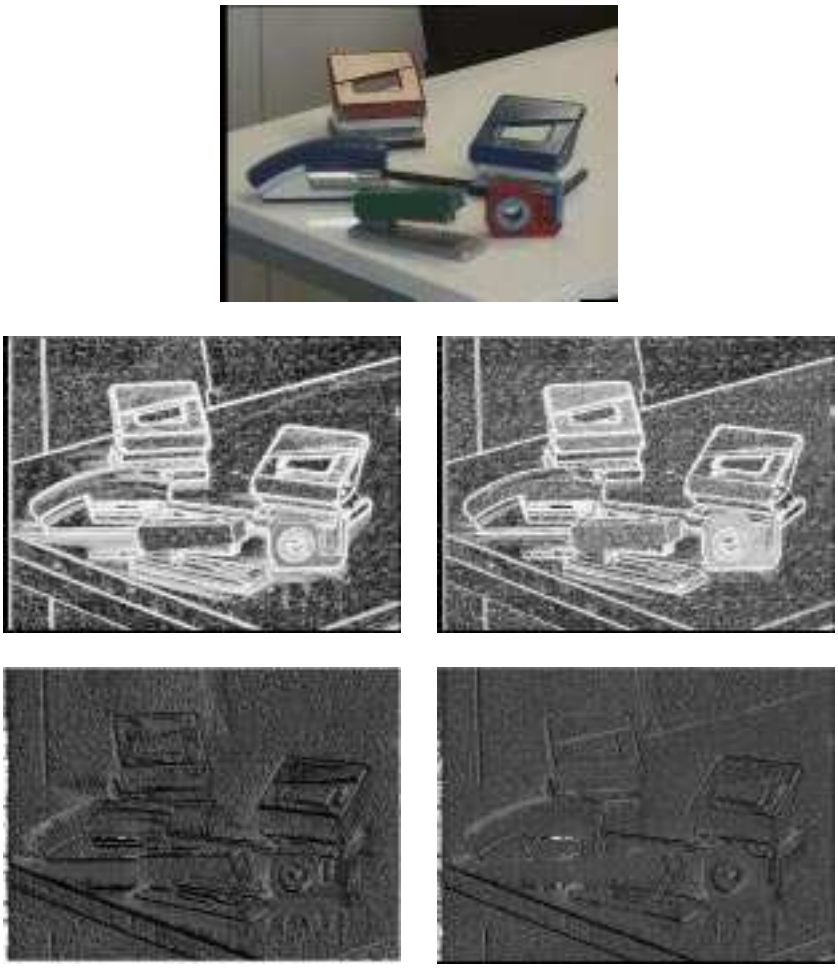


Figure 3.4. Comparison of edge detection in gray-level and color image. Top: original image, middle: edge strength, bottom: edge orientation. Left: gray-level segmentation, right: color segmentation.

Lines can now be computed from edge images tracking the edge elements with high edge strength along the direction indicated by the edge orientation. The resulting lines are often represented as chain codes (cmp. Table 3.2). These chains can be further approximated to find segments which are approximately straight or are circular arcs. Various line following and approximation schemes have been introduced in the literature, e.g. in [284]. A method that computes straight lines directly from edge images is the application of the Hough-transform on edge images. The idea is to represent the straight lines by $y = ax + b$ and to use the quantized parameter space for a, b. An accumulator which holds the quantized parameter values is initialized to 0. For each edge element $(f_{\mathrm{e}}, f_{\mathrm{o}})$ computed with a gradient operator at a position

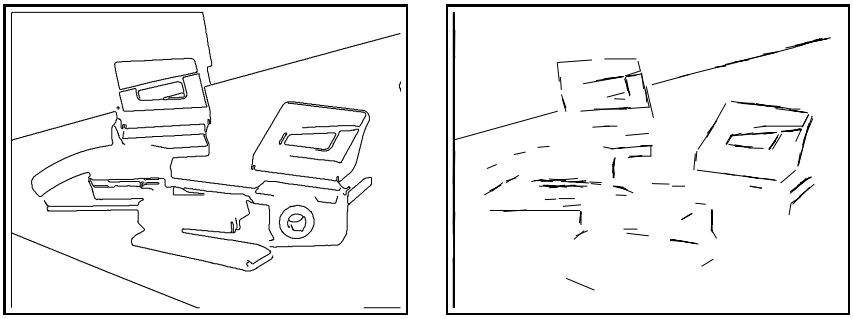


Figure 3.5. Straight lines and circular arcs from line approximation (left), and straight lines computed by the Hough transform (right).

$[i, j]$, we compute the parameter b from i and j and estimate the parameter a from f_{o}. If the slope a is finite, we increment the accumulator at a, b by f_{e}. If instead of the parametrization above, a line representation as $r = \cos \alpha x + y$ is chosen, the parameter array has a finite size but the results of edge detection cannot be used as parameters directly. Local maxima in the accumulator are used as indications for straight lines in the image.

Straight lines and circular arcs computed by the line following as described in [284], and straigth lines computed by the Hough-transform are shown in Figure 3.5.

The Canny-operator [106] was shown to be an optimal edge detector under fairly general assumptions, such as the assumption of step edges and the presence of white additive noise. It optimizes three criteria: detection, localization, and uniqueness. The final result of the fairly mathematical derivation in [106] is that the operator is computed from the convolution of the image f with $\partial\phi/\partial\boldsymbol{n}$ where $\phi\left[x, y\right] = \exp\left(-\left(x^2 + y^2\right) / \left(2\sigma^2\right)\right)$ is a two-dimensional Gaussian function and $\boldsymbol{n}$ is an estimate of a normal orthogonal to the edge orientation which can be computed form the edge derivatives. The edge orientation estimate as well as the parameters of the Gaussian have to be varied to obtain the optimal result. The result of the convolution is then analyzed to link edge elements to lines; an example is shown in Figure 3.6.

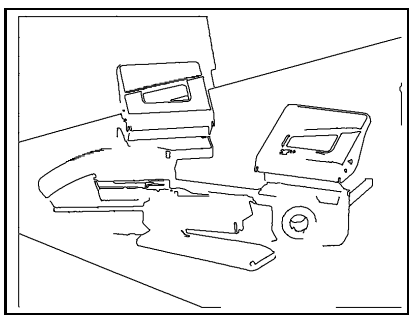


Figure 3.6. Result of Canny operator on the image in Figure 3.1.

3.1.4 Points, Vertices, Corners

A well-known method to find interesting points in an image is due to Moravec and is thus called the *Moravec interest* operator. He extended his own method in [490]; for each pixel in the input image, he uses a local window

$$\boldsymbol{W}_{ij} = [w_{ij,\mu\nu}]_{\mu,\nu=0,\ldots,7} = [f_{a,b}]_{a=i,\ldots,i+7,b=j,\ldots,j+7}\,. \tag{3.7}$$

The following features are computed from these windows:

$$m_{\mathrm{xx}ij} = \sum_{\mu=0}^{6}\sum_{\nu=0}^{6}\left(w_{ij,\mu\nu} - w_{ij,\mu\nu+1}\right)^2, \tag{3.8}$$

$$m_{\mathrm{yy}ij} = \sum_{\mu=0}^{6}\sum_{\nu=0}^{6}\left(w_{ij,\mu\nu} - w_{ij,\mu+1\nu}\right)^2, \tag{3.9}$$

$$m_{\mathrm{xy}ij} = \sum_{\mu=0}^{6}\sum_{\nu=0}^{6}\left(w_{ij,\mu\nu} - w_{ij,\mu+1\nu+1}\right)^2, \tag{3.10}$$

$$m_{\mathrm{yx}ij} = \sum_{\mu=0}^{6}\sum_{\nu=0}^{6}\left(w_{ij,\mu+1\nu} - w_{ij,\mu\nu+1}\right)^2. \tag{3.11}$$

The operator now is

$$H_{ij} = \min\left\{m_{\mathrm{xx}ij}, m_{\mathrm{xy}ij}, m_{\mathrm{yx}ij}, m_{\mathrm{yy}ij}\right\}. \tag{3.12}$$

The application of this operator on an image results in an interest map which is subsequently filtered by a Laplace-operator. The extension to color images is straight forward and uses the window

$$\boldsymbol{W}'_{ij} = [\boldsymbol{w}_{ij,\mu\nu}]_{\mu,\nu=0,\ldots,7} = [\boldsymbol{f}_{a,b}]_{a=i,\ldots,i+7,b=j,\ldots,j+7} \tag{3.13}$$

in a color image, i.e., the scalars $w_{ij,\mu\nu}$ in the Equations 3.8–3.11 are replaced by color vectors and a suitable difference operator. The square of the intensities is replaced by the scalar product of the resulting vectors. Figure 3.7 shows results of the interest operators a gray-level image and a RGB color image where the Euclidian distance of color vectors was used.

Another operator is the Harris corner detector [287]: for an image $\boldsymbol{f}$ we compute a gradient image whose horizontal and vertical components $\boldsymbol{f}_x$ and $\boldsymbol{f}_y$ are convolved with a Gaussian mask $\boldsymbol{G}_\sigma$ of standard derivation σ to obtain two smoothed images $\tilde{\boldsymbol{f}}_x$ and $\tilde{\boldsymbol{f}}_y$:

$$\tilde{\boldsymbol{f}}_x = \boldsymbol{f}_x \star \boldsymbol{G}_\sigma = \boldsymbol{f} \star \boldsymbol{G}_x \star \boldsymbol{G}_\sigma, \tag{3.14}$$

$$\tilde{\boldsymbol{f}}_y = \boldsymbol{f}_y \star \boldsymbol{G}_\sigma = \boldsymbol{f} \star \boldsymbol{G}_y \star \boldsymbol{G}_\sigma. \tag{3.15}$$

The default value for σ is $\sigma = 0.7$. At each position in the image $(i,j)^T$ we compute the matrix

$$\boldsymbol{M}_{ij} = \begin{pmatrix} M_{ij,11} & M_{ij,12} \\ M_{ij,21} & M_{ij,22} \end{pmatrix} \tag{3.16}$$

Figure 3.7. Interest map computed by Equation 3.12 for gray-level (left) and color images (right) overlayed to brightened input image from Figure 3.1.

with

$$M_{ij,11} = \tilde{f}^2_{x,ij}, \tag{3.17}$$
$$M_{ij,12} = \tilde{f}_{x,ij} \cdot \tilde{f}_{y,ij}, \tag{3.18}$$
$$M_{ij,21} = \tilde{f}_{x,ij} \cdot \tilde{f}_{y,ij}, \tag{3.19}$$
$$M_{ij,22} = \tilde{f}^2_{y,ij}. \tag{3.20}$$

The result of the Harris operator at position $(i,j)^T$ is now defined as

$$H_{ij} = \det(\boldsymbol{M}_{ij}) - k\,\mathrm{tr}(\boldsymbol{M}_{ij}), \tag{3.21}$$

where $k = 0.04$ is a fixed value for the computation. Again, the result is an interest map. In [174] it has been generalized to color in as

$$M_{ij,11} = \tilde{\boldsymbol{f}}^2_{x,ij}, \tag{3.22}$$
$$M_{ij,12} = \tilde{\boldsymbol{f}}_{x,ij} \cdot \tilde{\boldsymbol{f}}_{y,ij}, \tag{3.23}$$
$$M_{ij,21} = \tilde{\boldsymbol{f}}_{x,ij} \cdot \tilde{\boldsymbol{f}}_{y,ij}, \tag{3.24}$$
$$M_{ij,22} = \tilde{\boldsymbol{f}}^2_{y,ij}, \tag{3.25}$$

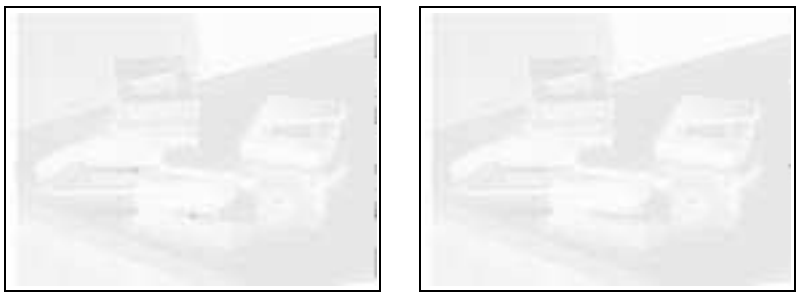


Figure 3.8. Interest map computed by the Harris operator for gray-level (left) and color images (right) overlayed to brightened input image from Figure 3.1.

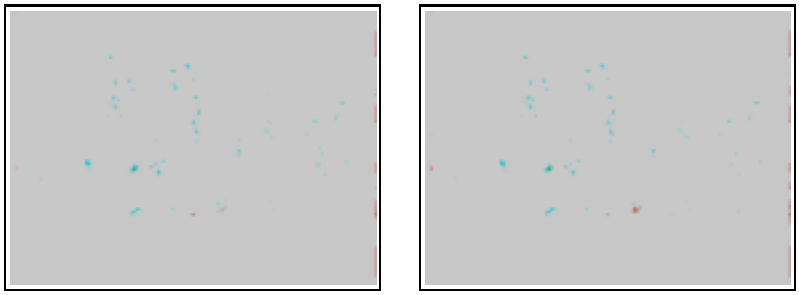


Figure 3.9. Interests map computed for gray-level (left) and color images (right).

where the products and squares are computed by scalar products of the vectors. Smoothing of the color gradient images is done componentwise.

Results of the gray-level and color version of this operator on the test image ot Figure 3.1 are shown in Figure 3.8. Figure 3.9 depicts a comparison of the three interest operators.

In addition to point operators which are mostly useful for point detection in image sequences, we also need detectors for points in single images. The demand for stability and roboustness can be satisfied when a larger context is used for the computation, as it is the case, e.g., by points which lie on lines. Points are detected by intersections or corners on these lines. Corners are identified by the analysis of the curvature of the lines. Thereby interesting points are found by post-processing lines resulting from line segmentation [284]. A result of point detections based on corners and vertices is shown in Figure 3.10.

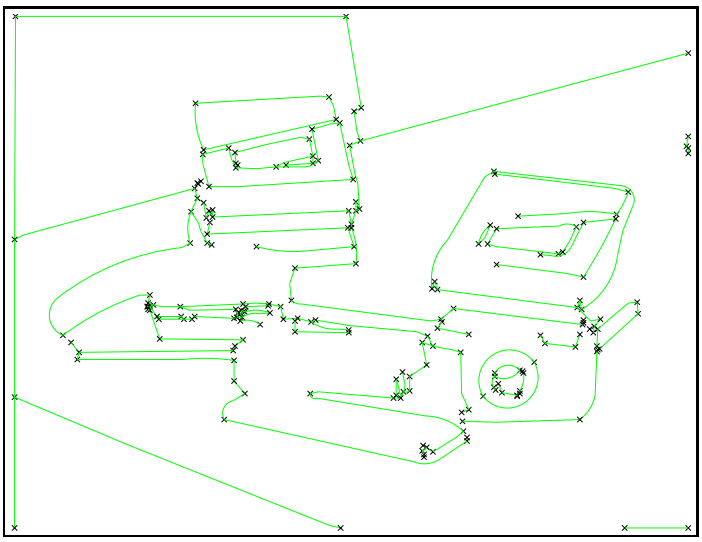


Figure 3.10. Segmented Corners and Vertices indicated by crosses on the lines.

Figure 3.11. Example of range image segmentation. Result of segmentation rendered with different colors corresponding to regions of different curvature characteristics (left). The other two images additionally show the minimal and maximal curvature coded by the brightness of the color.

3.1.5 Segmentation of Range Images

Range images as provided by several sensor types (cf. Section 1.1) are also subject to image segmentation. The principles introduced for intensity images can be applied as well, if the depth information is represented as a matrix of distance values, i.e., as a range *image*. Instead of the intensity gradient, the surface gradient can be used for detection of discontinuities. In order to be less sensitive to noise, the gradient is often computed analytically from a parametric function whose parameters are estimated by an error minimization on a surface patch.

Surface patches can also be detected by region segmentation. The criterion of homogeneity can be, e.g., the minimal and the maximal curvature of the surface or the $H - K$ map (see Section 4.6). These parameters are invariant features with respect to viewpoint changes and can thus be used for object recognition [59] (cmp. Section 3.4). Often, a combination of intensity and range segmentation provides the best results.

An example of a triangle mesh that was reconstructed from multiple range images and segmented using the curvature estimation rule of Section 4.3.1 is shown in in Figure 3.11.

3.1.6 Data Representation

As described in Section 3.5, knowledge-based image analysis consists of a sequence of transformations from the input image over an initial segmentation to a symbolic description. The results of data driven segmentation thus have to be passed to an interface which provides the transfer of the information to the symbolic processing stage.

Data structures for the representation of the results of the segmentation process were presented by various authors as summarized in Table 3.2. A general data structure for the interface must have the expressional power to represent all kinds of information resulting from any segmentation process. Surfaces, lines, regions, vertices and various structural and temporal relations between any two of these objects may be detected during segmentation and have to be stored in these data structures. Naturally, representations for lines use representations of points, representations for regions utilize lines, e.g. for contour representation. The segmentation object [508, 529] has these properties and can be used as a common interface definition. To give an example, regions

Data structure	Author
Primal Sketches	Marr
2.5D Sketches	Marr
Recursive Structure for Line Drawings	Shapiro
Iconic-Symbolic Data Structure	Tanimoto
RSE-Graph	Hanson & Riseman
Line Adjacency Graph	Pavlidis
Region Adjacency Graph	Pavlidis
Region Graph	Fisher
Spatial Data Structure	Shapiro & Haralick
Segmentation Objects	Niemann & Paulus

Table 3.2. Candidates for interfaces from segmentation to knowledge based processing according to [530].

detected with the methods of Section 3.1.2 can be represented as chain codes and are stored as a set of contour lines in a segmentation object; some of the contours may be polylines composed of straight lines; data-driven estimation of parallelism results in relations between some of these lines which are also recorded in the segmentation object. A syntactic description of segmentation objects is given in [508, p. 74–75]; it is also mapped to a syntactic description of concepts of a knowledge base. An object-oriented implementation of segmentation objects can be found in [529]. The system is further extended to contain a hierarchy of image segmentation operators in [529].

3.1.7 Performance Evaluation

Objective and general methods for the evaluation of image processing algorithms, filtes, and segmentation are not available, yet. The major problem is to define the evaluation criterion which often depends on the application, whereas segmentation algorithms were implemented independently from the the application.

In order to compare the results of different algorithms which produce the same type of segmentation results, their parameters have to be known and have to be varied systematically. To give an example, the algorithm in Section 3.1.2 requires the choice of several thresholds θ_{v}, θ_{m}, etc. In most of the cases, the notion of a 'good result' will depend on the application and the task of the overal system. In the context of active vision (Section 3.6), the 'quality' of a result may be chosen sub-optimal, since higher computation speed may be crucial instead of high precision. 'Optimal' solutions such as the Canny operator, may be not the right choice since they require too much computation time or since the underlying assumptions are not valid.

Performance evaluation of computer vision systems is a problem of ongoing research; the sub-task of evaluating segmentation algorithms will be useful for overall evaluation [283]. A survey on evaluation of segmentation methods can be found in [773].