Institut für Photogrammetrie und Kartographie
der Technischen Universität München
Lehrstuhl für Photogrammetrie und Fernerkundung

Hierarchical Real-Time Recognition of Compound Objects in Images

Dissertation

Markus Ulrich

Institut für Photogrammetrie und Kartographie
der Technischen Universität München
Lehrstuhl für Photogrammetrie und Fernerkundung

Hierarchical Real-Time Recognition of Compound Objects in Images

Markus Ulrich

Vollständiger Abdruck der von der Fakultät für Bauingenieur- und Vermessungswesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.rer.nat. E. Rank

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. H. Ebner
2. Univ.-Prof. Dr.-Ing. habil. Th. Wunderlich

Die Dissertation wurde am 30.4.2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 10.6.2003 angenommen.

Abstract

This dissertation proposes a novel approach for the recognition of compound 2D objects in images under real-time conditions. A compound object consists of a number of rigid object parts that show arbitrary relative movements. The underlying principle of the approach is based on minimizing the overall search effort, and hence the computation time. This is achieved by restricting the search according to the relative movements of the object parts. Minimizing the search effort leads to the use of a hierarchical model: only a selected root object part, which stands at the top of the hierarchy, is searched within the entire search space. In contrast, the remaining parts are searched recursively with respect to each other within very restricted search spaces. By using the hierarchical model, prior knowledge about the spatial relations, i.e., relative movements, between the object parts is exploited already in an early stage of the recognition. Thus, the computation time can be reduced considerably. Another important advantage of the hierarchical model is that it provides an inherent determination of correspondence, i.e., because of the restricted search spaces, ambiguous matches are avoided. Consequently, a complicated and computationally expensive solution of the correspondence problem is not necessary. The approach shows additional remarkable features: it is general with regard to the type of object, it shows a very high robustness, and the compound object is localized with high accuracy. Furthermore, several instances of the object in the image can be found simultaneously.

One substantial concern of this dissertation is to achieve a high degree of automation. Therefore, a method that automatically trains and creates the hierarchical model is proposed. For this, several example images that show the relative movements of the object parts are analyzed. The analysis automatically determines the rigid object parts as well as the spatial relations between the parts. This is very comfortable for the user because a complicated manual description of the compound object is avoided. The obtained hierarchical model is used to recognize the compound object in real-time.

The proposed strategy for recognizing compound objects requires an appropriate approach for recognizing rigid objects. Therefore, the performance of the generalized Hough transform, which is a voting scheme to recognize rigid objects, is further improved by applying several novel modifications. The performance of the new approach is evaluated thoroughly by comparing it to several other rigid object recognition methods. The evaluation shows that the proposed modified generalized Hough transform fulfills even stringent industrial demands.

As a by-product, a novel method for rectifying images in real-time is developed. The rectification is based on the result of a preceding camera calibration. Thus, a very fast elimination of projective distortions and radial lens distortions from images becomes possible. This is exploited to extend the object recognition approach in order to be able to recognize objects in real-time even in projectively distorted images.

Zusammenfassung

In der vorliegenden Arbeit wird ein neues Verfahren vorgestellt, mit dem zusammengesetzte 2D Objekte in Bildern unter Echtzeit-Anforderungen erkannt werden können. Ein zusammengesetztes Objekt besteht aus mehreren starren Einzelteilen, die sich relativ zueinander in beliebiger Art bewegen können. Das dem Verfahren zugrunde liegende Prinzip basiert auf der bestmöglichen Verringerung des Suchaufwandes und dient somit dem Ziel, die Berechnungszeit während der Erkennungsphase zu minimieren. Die Umsetzung dieses Zieles wird durch die Einschränkung der Suche entsprechend der relativen Bewegungen der Objektteile erreicht. Dies führt zu der Verwendung eines hierarchischen Modells: Lediglich das Objektteil, das an der Spitze der Hierarchie steht, wird innerhalb des gesamten Suchraumes gesucht. Die verbleibenden Objektteile werden hingegen innerhalb eingeschränkter Suchräume relativ zueinander unter Verwendung eines rekursiven Verfahrens gesucht. Durch den Einsatz des hierarchischen Modells kann Vorwissen über die räumlichen Beziehungen, d.h. die relativen Bewegungen, zwischen den Objektteilen bereits in einer sehr frühen Phase der Erkennung genutzt werden. Dadurch wird die Rechenzeit entscheidend reduziert. Ein weiterer großer Vorteil des hierarchischen Modells ist die inhärente Bestimmung der Zuordnung: Durch die eingeschränkten Suchräume werden Probleme, die durch auftretende Mehrdeutigkeiten hervorgerufen werden würden, vermieden. Eine komplizierte und rechenintensive Lösung des Zuordnungs-Problems während der Erkennungsphase erübrigt sich somit. Das vorgestellte Verfahren besitzt weitere bemerkenswerte Eigenschaften: Es ist nicht auf eine bestimmte Objektart beschränkt, sondern ist nahezu auf beliebige Objekte anwendbar. Das Verfahren zeichnet sich außerdem durch eine hohe Robustheit aus und ermöglicht es, das zusammengesetzte Objekt mit hoher Genauigkeit im Bild zu lokalisieren. Darüber hinaus können auch mehrere Instanzen eines Objektes im Bild simultan gefunden werden.

Ein wesentliches Anliegen dieser Arbeit ist es, einen hohen Automatisierungsgrad zu erzielen. Aus diesem Grund wird eine Methode entwickelt, die es erlaubt, das hierarchische Modell automatisch zu trainieren und aufzubauen. Hierfür werden einige Beispielbilder, in denen die relativen Bewegungen der Objektteile zu sehen sind, analysiert. Durch die Analyse können sowohl die starren Objektteile als auch die Relationen zwischen den Teilen automatisch ermittelt werden. Dieses Vorgehen ist äußerst komfortabel, da sich eine komplizierte manuelle Beschreibung des zusammengesetzten Objektes durch den Benutzer erübrigt. Das somit abgeleitete hierarchische Modell kann schließlich für die Erkennung in Echtzeit genutzt werden.

Die in dieser Arbeit vorgeschlagene Strategie zur Erkennung zusammengesetzter Objekte setzt die Nutzung eines Verfahrens zur Erkennung starrer Objekte voraus. Deshalb werden einige neue Modifikationen der generalisierten Hough-Transformation, einem Voting-Mechanismus zur Erkennung starrer Objekte, vorgestellt, die die Leistungsfähigkeit der generalisierten Hough-Transformation verbessern. Die erzielte Leistungsfähigkeit wird durch einen Vergleich mit weiteren Erkennungsverfahren für starre Objekte eingehend evaluiert. Es zeigt sich, dass die modifizierte generalisierte Hough-Transformation strengen industriellen Anforderungen genügt.

Gleichsam als ein Nebenprodukt der vorliegenden Arbeit wird eine neue Methode zur Rektifizierung von Bildern in Echtzeit vorgestellt. Die Rektifizierung basiert auf dem Ergebnis einer zuvor durchgeführten Kamerakalibrierung. Dadurch ist es möglich, sowohl projektive Verzerrungen als auch radiale Verzeichnungen des Kameraobjektives in Bildern sehr effizient zu eliminieren. Die Rektifizierung kann dann genutzt werden, um das Objekterkennungsverfahren dahingehend zu erweitern, Objekte auch in projektiv verzerrten Bildern in Echtzeit zu erkennen.

Contents

Chapter 1

Introduction

Using a hierarchical model for the recognition of compound objects provides higher efficiency and inherent determination of correspondence in contrast to standard methods, and hence facilitates real-time applications. This is the thesis of this dissertation.

The high relevance of the increasing automation process in the field of industrial production is undisputed. The already available high potential of automation can be attributed, amongst other things, to the progress in computer vision in general and in machine vision in particular. One of the most important topics in machine vision, and hence in industrial automation, is object recognition, i.e., objects of the real world must be automatically recognized and localized in digital images by a computer.

The thesis refers to the recognition of compound objects in real-time. To emphasize the novel aspects of this dissertation and to explain the basic idea behind it, definitions of the two decisive terms "real-time" and "compound objects" are given:

The term "real-time" is used in many applications with different semantics. A definition of real-time from a computer science point of view is given in (SearchSolaris.com 2002):

> "*Real-time* is a level of computer responsiveness that a user senses as sufficiently immediate or that enables the computer to keep up with some external process (for example, to present visualizations of the weather as it constantly changes). . . . Real-time describes a human rather than a machine sense of time."

Based upon this definition it is obvious that the upper boundary for the length of the processing time interval that makes a process real-time capable is application dependent (Russ 2000). Thus, operating in real-time is not about being "real fast" because the time interval may range from microseconds to megaseconds (Jensen 2002). In the field of video processing, for example, often the video frame rate (about 30 ms) is decisive, whereas, in remote sensing one would rather speak of online processing instead of real-time. This is because the image sequences that are dealt with in remote sensing are based on arbitrary time patterns and are not necessarily equidistant in time. Hence, it is not unusual that the real-time or online analysis of remotely sensed data takes several minutes or even hours.

In this dissertation "real-time" primarily demands from the object recognition process a computation time that enables the computer to keep up with an external process. The object recognition approach, however, should not be related to any specific application. I.e., the time constraint must be derived from an external process that is application independent. Since the process of image acquisition is an indispensable step in every application, it is reasonable to take the video frame rate of common off-the-shelf cameras as reference, which typically is 1/30th of a second. In a multitude of applications new information is available not in each frame but only in each third or fifth frame, for example. With this it is possible to give at least a coarse definition of what "real-time" means in this dissertation: the computation time of the object recognition process should be in the range of a few hundredths of a second to a few tenths of a second using common standard hardware. This requirement considerably complicates the development of an appropriate object recognition method. By using a hierarchical model, as it is proposed in this dissertation, the gain in efficiency facilitates real-time applications.

In contrast, the definition of “compound object” is considerably simpler. First of all, it should be pointed out that in this dissertation 2D objects are considered because the recognition of 3D objects, as it is performed in the field of robotics, for example, is not necessary for most applications in industry. The term “compound object” implies that the object consists of a number of object parts. Furthermore, the object parts are allowed to move with respect to each other in an arbitrary way. The term “movement”, in a mathematical sense, describes a translation and a rotation. Following this definition, objects can be classified into the two classes: *compound objects* and non-compound or *rigid objects*. Rigid objects may also consist of several object parts, but the constellation of the parts is fixed, i.e., the parts do not move with respect to each other. In contrast, compound objects consist of several object parts that are rigid objects. Additionally, the constellation of the object parts is variable. For instance, a wheel of a car can be seen as a rigid object consisting of the two parts, the rim and the tire. The car itself can be seen as a compound object consisting of the body and the four moving wheels: the wheels rotate and change their distance to the body because of the shock absorbers. Because the movements of the object parts, and hence the appearance of the compound object, is not known à priori, an efficient recognition of compound objects in images is complicated dramatically in contrast to the recognition of rigid objects. Furthermore, a correspondence problem arises when dealing with compound objects that additionally hampers the recognition: even if the wheels of the car have been recognized, it is not immediately clear which of the four wheels is the front left wheel, for example. Therefore, this correspondence problem must be solved in a subsequent step taking into account the constellation of all object parts. For example, one is unable to assign the label “front left” to one of the four wheels until the body of the car is recognized. Unfortunately, solving this correspondence problem is complicated and computationally expensive, especially for compound objects that consist of a large number of similar object parts. Consequently, real-time computation would be impossible. By using a hierarchical model, however, additionally to the gain in efficiency an inherent determination of the correspondence is ensured, and hence the correspondence problem becomes dispensable.

To summarize, the main novel aspect described in this dissertation is the development of an approach that combines the ability to recognize compound objects with the ability to perform the recognition in real-time.

Chapter 2

Scope

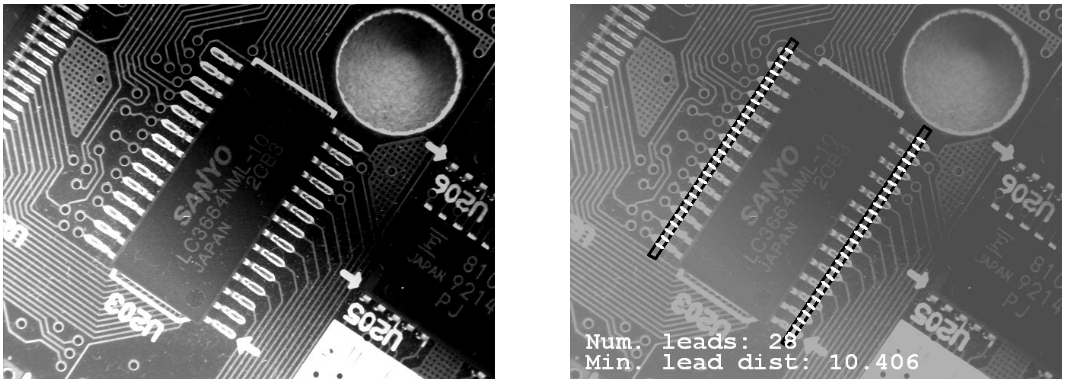
In this chapter the scope of this dissertation is introduced. The conceptual formulation for the work is illustrated by giving several example applications that are discussed in detail (Section 2.1). At first, the requirements for the object recognition approach are derived from the example applications, and are completed by additional constraints (Section 2.2). The concept of the object recognition approach that is described in this dissertation is subsequently introduced (Section 2.3). After that, the background of the work, which considers the general conditions under which the dissertation has originated, is explained (Section 2.4). The chapter is concluded by a short overview in which the structure of this dissertation is described. This may help the reader to arrange the single sections of this work into an entire framework and to understand the interrelationship between individual working steps without losing touch with the central theme (Section 2.5).

2.1 Example Applications and Motivation

2D object recognition is used in many computer vision applications. It is particularly useful for machine vision, where often an image of an object must be aligned with a (well-defined) *model* of the object. In general, the model contains a certain description of the object that can be used for recognition. For instance, a model can be represented by a CAD model, a gray scale image, extracted features like points, lines, or elliptic arcs, or any other description. In most cases, the result obtained by the object recognition approach directly represents the transformation of the model to the image of the object. Object recognition delivers the transformation parameters of a predefined class of transformations, e.g., translation, rigid transformations, similarity transformations, or general 2D affine transformations (which are usually taken as an approximation of the true perspective transformations an object may undergo). This definition implies that object recognition not only means recognizing an object, i.e., deciding whether the object is present in the image or not, but additionally means localizing it, i.e., getting its transformation parameters. The transformation refers to an arbitrary reference point of the model and is often referred to as *pose* in the literature (Rucklidge 1997). In the remainder of this dissertation no distinction will be made between the two separate processes of recognition and localization: recognition will always include the process of localization.

The pose that is returned by the object recognition approach can then be used for various tasks, ranging from alignment, quality control, inspection tasks over character recognition to complex robot vision applications like pick and place operations. In the following, several example applications are introduced in order to elaborate the conceptual formulation for this dissertation and to derive the most important requirements that should be fulfilled.

A typical inspection application is illustrated in Figure 2.1. The task is to count the number of leads of the integrated circuit (IC) and additionally check the distances between neighboring leads to ensure that short circuits are avoided. Before these measurements can be performed the pose of the IC must be determined in



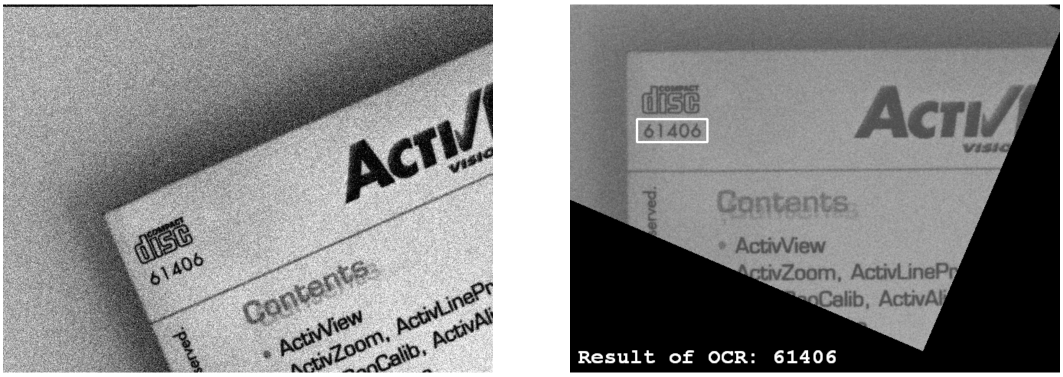
(a) Input image

(b) Inspected leads of the IC

Figure 2.1: Example that illustrates the role of object recognition in inspection tasks. The leads of the integrated circuit (IC) in (a) are to be inspected. The measurement windows (black), the extracted edges of the single leads (white), and the results of the measurement are shown in (b).

the image by using an object recognition approach. In this case, the print on the IC is an obvious distinct object that can be used to build a model for the recognition process. A single image of the object should be sufficient to automatically build the model in order to keep the model creation as simple as possible. Because the relative position of the leads with respect to the print is approximately constant and known à priori, two measurement windows can be opened, which include the leads on both sides of the IC. This can be done after the pose of the print has been determined by the recognition approach. Within the measurement windows subpixel precise edges are computed and used to count the leads and to measure the distances between neighboring leads (see Figure 2.1(b)). If one takes a closer look at Figure 2.1(a), a non-uniform illumination can be observed in the image, which is due to a light source that was not perfectly mounted, leading to a stronger illumination of the lower left corner of the image. A uniform illumination that additionally is constant over time is highly desirable in most applications. Unfortunately, sometimes a controlled illumination is hard to achieve if one refrains from using an expensive set-up. Thus, it becomes obvious that the object recognition method must be robust against these kind of illumination conditions. For visualization purposes only, the contrast of the image in Figure 2.1(b) is lowered. This auxiliary visualization step is performed whenever additional information is plotted within a gray scale image and the original image contrast makes it necessary. Therefore, this must not be confused with a meaningful image processing operation under any circumstances.

Figure 2.2 illustrates one possible role of object recognition in the field of optical character recognition (OCR). Here, the task is to read the digits below the “disc” label. In many implementations, object recognition is not directly applied to recognize the characters. Instead, OCR is performed as a classification process, in which sample characters are trained and used to derive a set of classification parameters for each character. Often, these parameters are not rotationally invariant. Hence, it is only possible to read characters that have the same orientation as the characters used for training. In general, this assumption regarding the orientation is not valid. A brute-force solution is to train the characters in all possible orientations. However, the computation time for training and recognizing the characters increases. Additionally, the recognition rate decreases since the risk of confusion is higher. For example, it is not immediately possible to distinguish the letters “d” and “p” if they may appear in arbitrary orientation. A more sophisticated approach uses object recognition in a preliminary stage. In the example of Figure 2.2, the parameters are trained using characters that have been horizontally aligned. The CD label shown in Figure 2.2(a), however, may appear in arbitrary orientation. Therefore, the image must be rotated back to a horizontal orientation before the OCR can be applied. This process is often called *normalization*. Object recognition can be used to obtain the orientation angle by which the image must be rotated. Because the digits below the “disc” label are not known, but must be determined, they cannot serve as object for the recognition process. In contrast, the appearance of the “disc” label itself is constant and is an ideal pattern that can be searched in the image. As can be seen from this example, the



(a) Input image

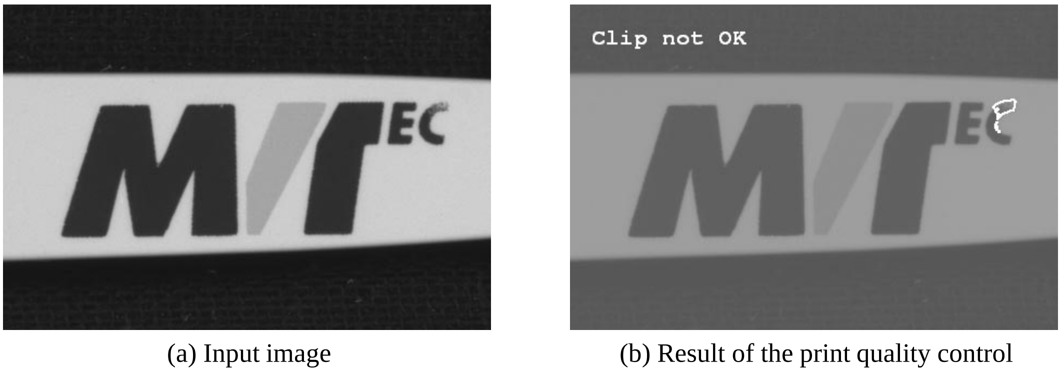
(b) Result of the OCR

Figure 2.2: Example that illustrates the role of object recognition in optical character recognition (OCR). The digits below the “disc” label in (a) are to be read. To simplify the classification of the characters, the image is horizontally aligned according to the orientation of the recognized “disc” label (b).

recognition approach should be robust against a moderate degree of image noise. After the label has been recognized, the image is normalized, i.e., horizontally aligned by rotating it by the negative orientation of the found label. The result is shown in Figure 2.2(b). Although in this case the entire image is rotated for demonstration purposes, normally, it is sufficient to only rotate the part below the disc label to speed up the process. Finally, the region of interest, i.e., the part of the image, in which the OCR is to be performed, can be restricted to the image region directly below the label. Based on these two examples, it can be postulated that the recognition approach must be invariant to object orientation.

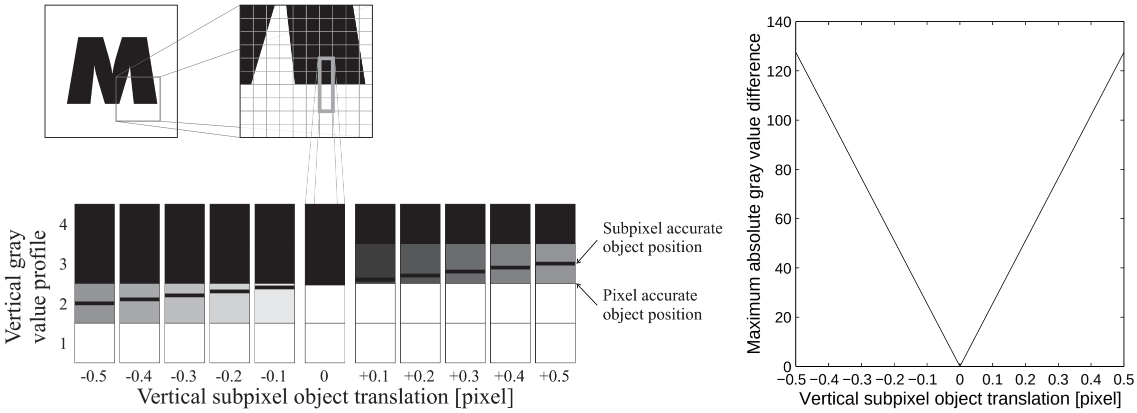
Another frequently arising problem is to check the quality of various kinds of prints. For example, it is established by law that food must have an appropriate durability indication, e.g., “Best Before:”, “Best Before End:”, or “Use By:”, followed by the corresponding date. Therefore, it is important that the date on food packagings is easy to read, and hence the corresponding print must not have severe quality faults. To mention another example, companies are very intent on handing out their products only with a perfectly printed company logo, because otherwise the imperfections of the logo are directly attributed to possible imperfection of the company by the potential customer. Figure 2.3(a) shows the print on a pen clip that represents the company logo “MVTec”. In this example, the rightmost character “c” shows a substandard print quality in the upper part of the character. A typical way to examine the print quality is to compare the gray values of the print that is to be checked with the gray values of an ideal template, which holds a perfect instance of the print (Tobin et al. 1999). Absolute gray value differences that exceed a predefined threshold are interpreted as severe quality faults and returned by the program. The alignment of the ideal template over the print that is to be checked can be achieved using object recognition by selecting, for example, the entire print as the object to be found. From this it can be reasoned that even if parts of the object are missing, as is the case when dealing with print faults, the recognition method must still be able to find the object. This is a hard but important requirement, since the case of missing parts is anything but rare, especially in the field of industrial quality control. Furthermore, especially in the field of quality control the colors of the object may vary, for example, depending on the used pressure during the print, on the amount of ink on the stamps, or on the color mixture. Thus, not only a non-uniform illumination but also the change of the object itself affects the gray values of the object in the image. Therefore, the object recognition approach should be robust against general changes in brightness of the object. Finally, the returned pose of the object can then be used to transform the ideal template to the desired position and orientation. Especially in this application the real-time aspect becomes important since the operational capacity in the pen production is very high, and hence fast computation for the object recognition is demanded.

Based on this example, another demand on the recognition method can be derived which deals with subpixel object translations. The principle of the effect of subpixel translation is shown in Figure 2.4(a) using



(a) Input image (b) Result of the print quality control

Figure 2.3: Example that illustrates the role of object recognition in quality control. The quality of the print on the label of the pen clip in (a) is to be checked. An ideal template of the print is transformed according to the result of the object recognition and compared to the input image. Gray value differences that exceed a predefined threshold are returned as errors (b).

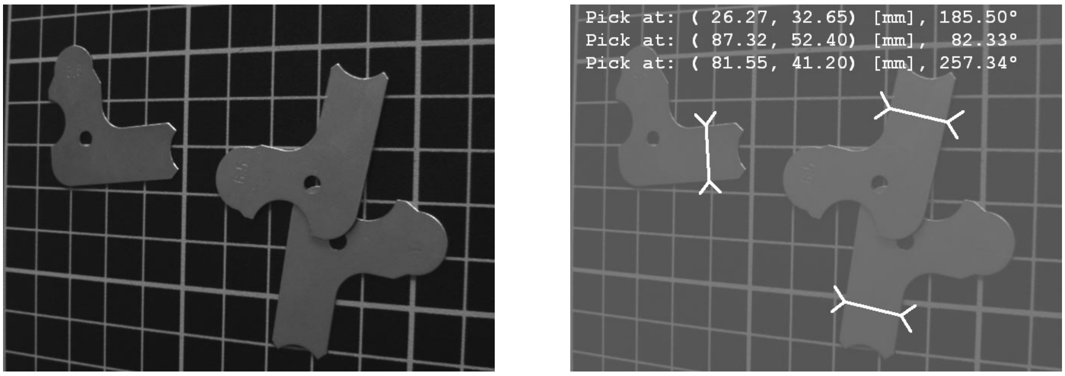


(a) Effect of subpixel translation (b) Induced error

Figure 2.4: The effect of subpixel translation on the gray values is shown in (a). Pixel precise object recognition methods induce errors in the case of subpixel translations (b).

a synthetic example, where a horizontal edge of the letter “M” is considered. For the ideal template a white background (gray value 255) and a black foreground (gray value 0) are assumed. Let the horizontal edge of the letter exactly fall on the border between two neighboring vertically arranged pixels. Then a sharp horizontal edge with a gray value jump from 0 to 255 arises. If the letter is translated in a vertical direction by 1/2 pixel in both directions using a step width of 1/10 pixel, the gray value of the corresponding pixel smoothly changes. Consequently, the originally sharp horizontal edge becomes more and more blurred. When using a pixel precise object recognition method, the subpixel translation would be undetectable, leading to a maximum difference of 1/2 pixel between the true vertical location and the vertical location that is returned by the recognition method. The resulting absolute gray value difference between the print and the incorrectly transformed ideal template are plotted in Figure 2.4(b). The gray value differences, in this case, reach amplitudes of 127, which make a reliable detection of defects in the print almost impossible. In contrast, such effects are avoided when using a subpixel precise object recognition method. Further examples that show the need for subpixel precise object recognition can be found in image registration and feature location measurements in photogrammetry, remote sensing, image sequence analysis, or nondestructive evaluation (Tian and Huhns 1986).

The example application illustrated in Figure 2.5 introduces further important aspects to be considered in object recognition. Here, the three metal parts shown in Figure 2.5(a) must be picked by a robot. From this example it follows that the object recognition method should also be able to recognize several instances of


Pick at: (26.27, 32.65) [mm], 185.50°
Pick at: (87.32, 52.40) [mm], 82.33°
Pick at: (81.55, 41.20) [mm], 257.34°

(a) Input image

(b) Pick points for the robot

Figure 2.5: Example that illustrates the role of object recognition in pick and place applications. The metal parts shown in (a) are to be picked by a robot. The pick points are marked in (b). It is important to note that the recognition approach must cope with projective distortions and overlapping objects.

the object in the image at the same time. Additionally, the different metal parts may overlap each other, and hence the recognition approach must also be able to handle occlusions up to a certain degree. This problem is equivalent to the situation where parts of the object are missing, as occurred in the example application of Figure 2.3. Furthermore, the image plane of the camera is not parallel to the plane in which the objects lie during image acquisition. This deviation from the nadir view leads to projective image distortions that consequently influence the appearance of the objects in the image and make the recognition much more difficult. After the metal parts have been localized by the recognition method, the world coordinates of the pick points (see Figure 2.5(b)) are transmitted to the robot. More common pick and place applications can be found in the semiconductor industry where circuit boards are automatically equipped using robots.

Up to now, only examples with non-compound objects have been introduced. In the following, the motivation for recognizing compound objects will be elaborated based upon further example applications. These examples are also useful to elaborate the definition of compound objects that was given in Chapter 1. Because in the following rigid objects must be distinguished from compound objects, the model representation of a rigid object is referred to as *rigid model* and the model of a compound object as *compound model* in the remainder of this dissertation.

To give a first example, the application of quality control shown in Figure 2.3 is used. However, in contrast to the previously discussed example, now the considerations are extended to multiple occurrences of the pen clip (see Figure 2.6). Because the printing process of the logo was performed in two steps by applying two independent stamps, one for each color, misalignments within the print may occur between the dark gray letters “M Tec” and the light gray letter “V”. Keeping the application of quality control in mind, it is necessary to perfectly align the ideal template to the print. The misalignment within the print, however, causes a discrepancy between the appearance of the print in the image and the object description in the model that is used to recognize the object. This discrepancy cannot be described by one global 2D transformation — which is typically used in the recognition process — because different parts of the object are transformed individually. This leads to difficulties during object recognition and during the detection of print quality faults. One solution is to split the object, i.e., the entire print, into two separate objects, one representing the dark gray letters and the other the light gray letter, respectively. The object recognition approach is then started twice (once for each object), resulting in two independent poses for the two objects in the image. The drawback of this solution is that available information regarding the relations between the two objects is not exploited. In this example, such information could be, e.g., that the letter “V” is somewhere in between “M” and “Tec”. The consequence of ignoring this information is a loss of efficiency, since both objects must be searched in the image without prior knowledge. This loss in most cases is already important when dealing with objects that consist of two separate object parts — as in this example. Considering the real-time requirement, the

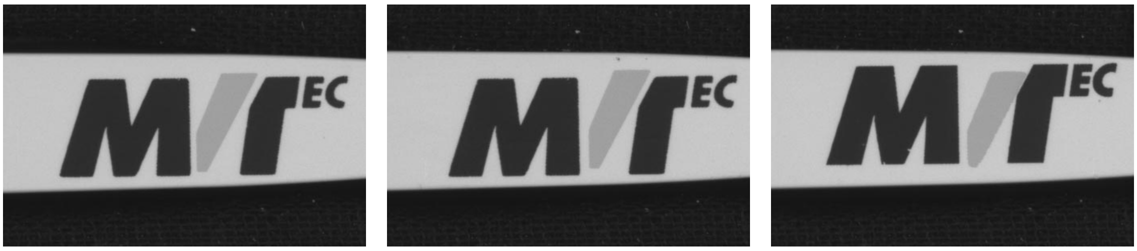


Figure 2.6: The logo “MVTec” is an example of a compound object that consists of the two object parts “M Tec” and “V”.

more object parts that are involved, the more important it becomes. As a consequence, the object recognition approach should be able to handle compound objects that consist of several object parts. The relations between the object parts should be explicitly modeled and taken into account during the recognition process as prior knowledge in order to obtain a high efficiency and to be able to fulfill the real-time requirement even for compound objects.

To get an idea of a more complex compound object, an example is presented in which the object consists of more than two object parts. In Figure 2.7 several prints of a label are shown that are used to mark the minimum durability on food packaging. The readability of the print can be checked by using a similar method as explained in the application of quality control shown in Figure 2.3. When taking a look at the images given in Figure 2.7 one can discern that the label, which represents the object, can be decomposed into five object parts: the rectangular border, the string “BEST BEFORE END:”, and the three two-digit numbers of which the last two are supplemented by a preceding slash. Obviously, a few images are already sufficient for a human being to identify the object parts into which the label decomposes. The number of required images depends on the relative movements that are shown in the images. The relative movement between two object parts must be shown in at least one image. For example, if the movements between all object parts are already included in two images then these two images are sufficient to detect the object parts. The object recognition approach should be able to automatically identify the object parts of compound objects using a sufficient number of example images — as shown in Figure 2.7. Furthermore, the relations between the single object parts and a search strategy should also be derived automatically by using the same example images. Based on this information, the compound model should be created. The compound model can then be used to recognize the compound object in an image. To give an example, one possible search strategy is to search for the rectangular border at first, and then restrict the search for the remaining parts to the image region lying inside the border.

In Figure 2.8, a last example of a compound object is introduced. It shows a circuit board equipped with five electronic modules, which are visualized in the upper left image by enclosing white ellipses. A typical application within the production process is to check whether all modules are present on the board and whether they are in the correct position and orientation in order to guarantee the perfect operation of the board. Because the positions and orientations of the electronic modules vary slightly from board to board, the five modules do not describe one rigid object, but can be put together into one compound object. Hence, in this example the compound object cannot be described by one physical object in the real world, but instead can be understood as a virtual object containing the five electronic modules. Thus, a compound object does not necessarily correspond to a real world object but can be seen on a more abstract level. Furthermore, in this example the background is strongly textured, which additionally complicates the object recognition.

The presented examples give an insight into the broad spectrum of applications that can be automated to a high degree using object recognition or that at least profit from object recognition in one of various ways. In order to make these advantages available to a large number of users, special knowledge of the user about image processing or computer vision must not be required. Furthermore, the degree of automation should be as high as possible to limit the user interactions to a minimum. Consequently, the motivation from a practical point of view, upon which this dissertation is based, is to develop an object recognition approach that is easy to use.

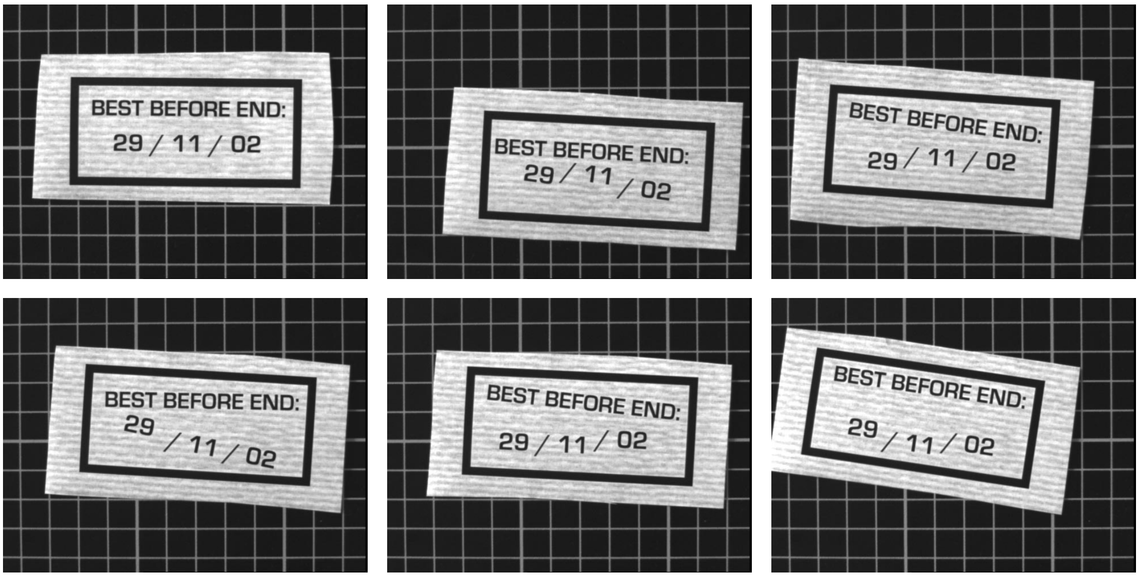


Figure 2.7: The compound object decomposes into five object parts: the rectangular border, the string “BEST BEFORE END:”, and the three two-digit numbers, of which the last two are supplemented by a preceding slash.

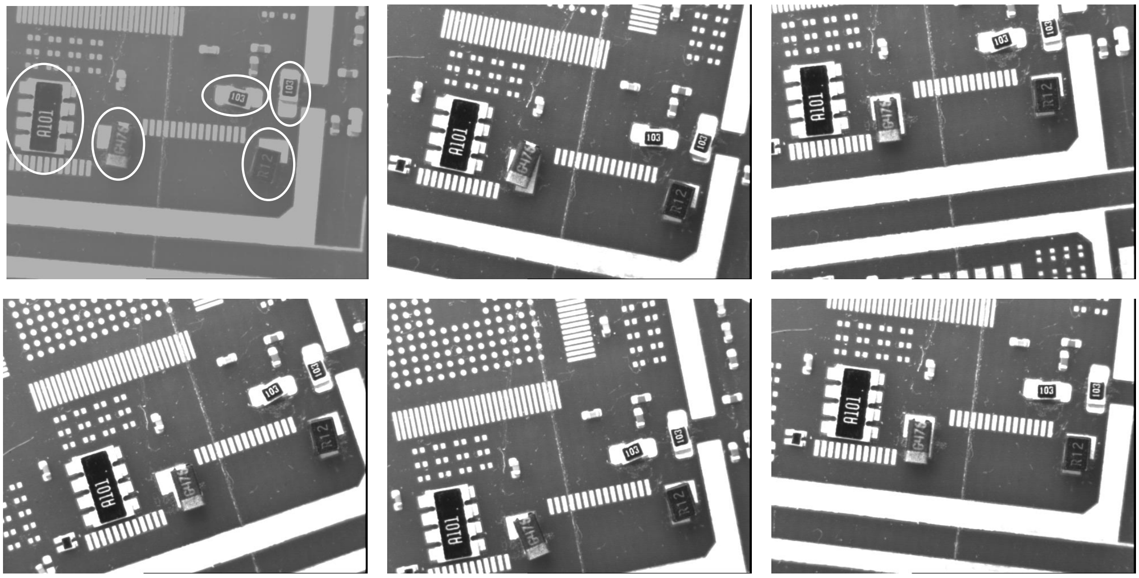
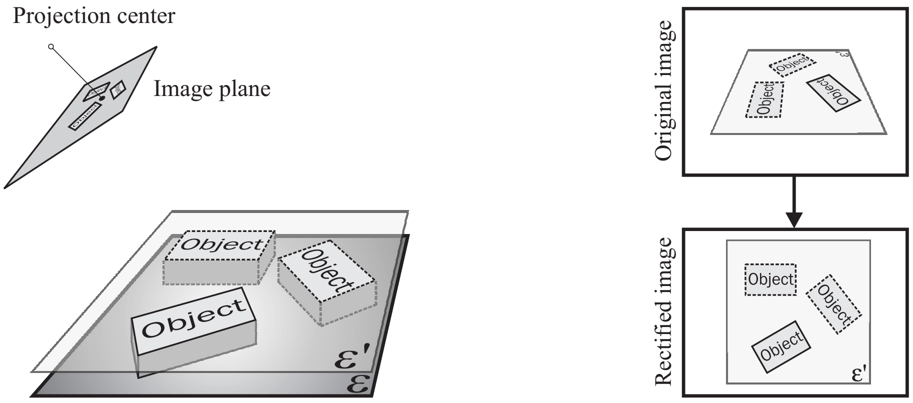


Figure 2.8: The five electronic modules, which are visualized in the upper left image (white ellipses), slightly vary their position and orientation on the circuit boards. They can be represented by one compound object.

2.2 Requirements

Following the discussion of the example applications (Section 2.1), the requirements that an object recognition approach should fulfill will now be summarized. They are completed by additional requirements that have to be considered in industry.

However, before listing the demands some general remarks must be mentioned. Firstly, one of the aims of this dissertation is to develop an object recognition approach for a broad spectrum of applications. Consequently, there must be no special requirements on the necessary hardware in order to maximize the field of possible applications. Usually, only three hardware components should be necessary for real-time object recognition: a camera, a computer, and a frame grabber. Starting with the first component, it should be sufficient to use



(a) Image plane and object plane ε' are not parallel

(b) Rectification

Figure 2.9: The object recognition is restricted to planar objects. Projective distortions are caused by deviations from the nadir view (a). By rectifying the image the projective distortions can be eliminated (b).

off-the-shelf cameras. In a majority of cases monochrome cameras are used to deliver the video signal in one of the two most prevalent analog video formats *RS-170* (monochrome equivalent to *NTSC*) with an image resolution of 640×480 pixels and *CCIR* (monochrome equivalent to *PAL*) with a resolution of 768×576 pixels. On the one hand, these cameras do not demand high financial investments and are therefore best qualified to satisfy the condition of a broad applicability. On the other hand, the use of these cameras prohibits object recognition approaches that are based on color information. As the second component, standard personal computers systems are already available in most companies and deliver high performance for low cost. No special image processing hardware should be needed. The frame grabber, as the last component, simply acts as an interface between camera and computer. It takes the video signal, which can be understood as a continuous stream of video frames, and grabs one or more images out of the sequence, whenever triggered to do so. In the case of analog cameras, the frame grabber additionally converts the analog signal into a digital signal that can be processed by the computer. Common frame grabbers use an 8-bit quantization. Thus, in the case of monochrome cameras, gray scale images with a maximum of 256 different gray values are obtained.

Secondly, it is sufficient in many applications — especially in industry — to recognize planar objects (Steger 2001). Therefore, this dissertation only deals with the recognition of 2D objects. Since in the real world no 2D objects exist, the meaning of “2D” in the context of this dissertation is discussed in the following. In general, the mapping of a moving object into an image can be described by two separate transformations. The first describes the transformation of the object in the real world (like 3D translation, 3D rotation, 3D scaling, etc.). The second describes the mapping of the object from the real world (3D) into the image plane of the camera (2D). The two transformations are abstracted in Figure 2.9(a). The 3D object is symbolized as a box that may be transformed in 3D space to different positions and orientations. Assume that the object is planar, i.e., its thickness is small relative to its distance from the camera. Assume furthermore that the transformation in the real world can be described by a 2D transformation (like 2D translation, 2D rotation, 2D scaling, etc.) within the plane that is spanned by the planar object. Consequently, also all possible appearances of the object are restricted to lie within that plane. This plane will be called *object plane* in the following. In Figure 2.9(a) the planar object is represented by the upper surface of the box containing the string “Object”. Since the box moves on a plane ε the upper surface moves on the object plane ε' that is parallel to ε at a distance that corresponds to the height of the box. Consequently, the mapping from the real world into the image plane is a homography and can be described by a projective transformation between two planes (ignoring any lens distortions for the moment).

Using camera calibration, the projective distortions of the object plane in the image can be eliminated by transforming the image plane back into the object plane ε' (see Figure 2.9(b)). This process will be referred to as *rectification* in the following. Subsequently, the object recognition approach only needs to cope with the remaining 2D transformation of the planar object in the real world. In the example applications presented so far, the 2D transformation can be described by a rigid motion (translation and rotation). In practice, it is sufficient that the 3D object has at least an approximately planar surface: although a minor unevenness introduces additional perspective distortions that cannot be eliminated by the rectification these distortions are negligible as long as the deviation from the nadir view is also sufficiently small. What is important is that all transformations the 3D object may undergo must lead to a 2D transformation of the planar object surface. In the following, the object will be equated with its planar surface since the 3D object as a whole is irrelevant for further considerations in this work. To give some examples, in Figure 2.1 the IC represents the 3D object with the print on the IC as the planar object surface, in Figure 2.2 the CD cover represents the 3D object with the "disc" label as the planar object surface, in Figure 2.3 the pen clip represents the 3D object, with the logo as the planar object surface, and in Figure 2.5 a metal part represents both the 3D object and the (approximately) planar object surface.

Now, after the general conditions have been stated, the requirements for an object recognition approach are given:

- *The object recognition approach should be able to handle compound objects.* Compound objects should not be treated as a set of independent objects that ignore the relations between them but should be explicitly modeled leading to an increased computational efficiency. Furthermore, the correct correspondence of the object parts should be given by the approach.

- *Objects should be recognized in real-time.* This is strongly connected with the previous requirement because without modeling the relations between object parts, real-time computation is hard to achieve when dealing with compound objects. Nevertheless, this requirement additionally implies the existence of an object recognition approach that is able to recognize *rigid* objects in real-time since a rigid object can be seen as a degenerated compound object with only one object part. Because the computational complexity of object recognition approaches depends on the image size, the real-time demand must be related to a maximum occurring image size. Bearing the above considered hardware requirements in mind, RS-170 or CCIR images are assumed in this dissertation. Hence, objects should be recognized in real-time when using images that have a size of not substantially larger than 768 $\times$ 576 pixels.

- *The model representation of a rigid object should be computed from an example image of the object.* Keeping in mind the claim that the object recognition approach should be easy to use, the computation of the rigid model should only ask for a single model image of the object. This is the most comfortable way because usually it is too costly or time consuming to compute a more complicated model, e.g, a CAD model, or to transform a given CAD model into a model representation that can be used for object recognition.

- *The model representation of a compound object should be computed from several example images of the compound object.* In contrast to the previous requirement, the model representation of compound objects is more complicated to compute since movements between object parts cannot be detected from a single example image. Nevertheless, in order to keep the model computation as simple as possible for the user, it should be sufficient to make several example images available. The object recognition approach should then be able to automatically derive the relations between the object parts from the given example images and to derive the compound model.

- *The object recognition approach should be general with regard to the type of object.* The approach should not be restricted to a special type of object. Thus, the model, which represents the object, should be able to describe arbitrary objects. For example, if straight lines or corner points were chosen as features to describe the object it would be impossible to recognize ellipse-shaped objects.

- *The object recognition approach should be robust against occlusions up to a certain degree.* This is often highly desirable in cases where several objects may overlap each other or in cases where object parts are missing.

- *The object recognition approach should be robust against changes in brightness of an arbitrary type up to a certain degree.* Illumination changes often cannot be avoided and are, for instance, caused by non-uniform illumination over the entire field of view, changing light (position, direction, intensity), objects with non-Lambertian surfaces, etc. Furthermore, changes in the color of the object itself also lead to changes in brightness in the image.

- *The object recognition approach should be robust against clutter.* Clutter in this context means any additional information in the image, aside from the object that is to be recognized. This information can, for example, be a strongly textured background or additional objects that are visible in the image, and which are possibly similar to the object of interest.

- *The object recognition approach should be robust against image noise.* Since noise cannot be avoided in the image, the approach should be robust against noise up to a certain degree.

- *Objects under rigid motion should be recognized.* This is closely related to the requirement of real-time computation. In general, the more degrees of freedom the transformation of an object includes the higher the complexity of the recognition approach and therefore the higher the computation time to recognize the object. Hence, the real-time demand is coupled with the allowable degrees of freedom. In this dissertation rigid motion (translation and rotation) is considered, i.e., the object recognition approach should be able to find the object at arbitrary position and orientation. This does not imply that the approach cannot be extended to more general transformations like similarity transformations or affine transformations. However, there is a trade-off between the real-time demand and the transformation class.

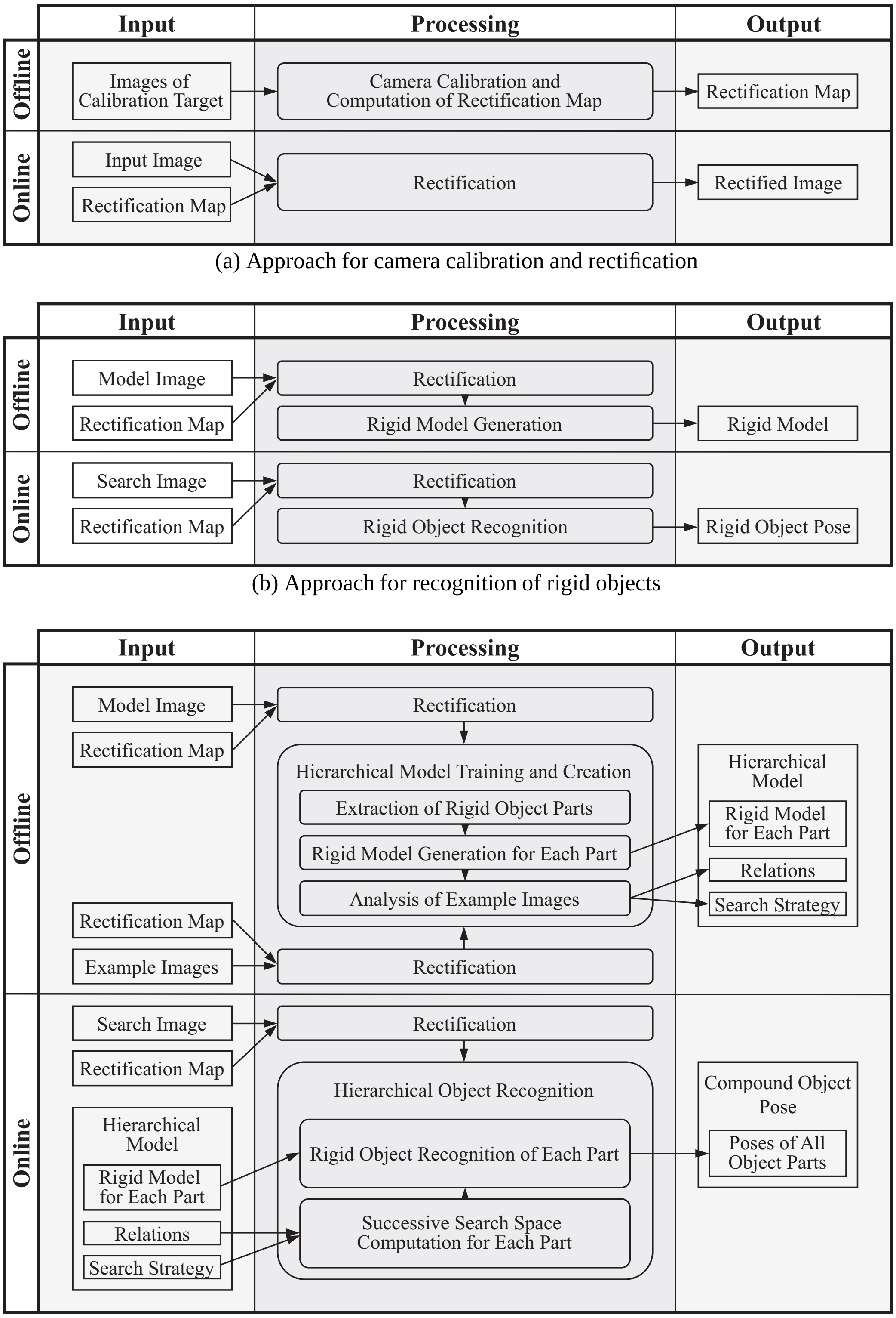
- *The approach should cope with deviations from the nadir view.* Often, it is not possible to mount the camera with a viewing direction perpendicular to the plane in which the object appears. The resulting projective distortions should be managed by the recognition approach.

- *The returned pose parameters should be of high accuracy.* This means that the pose parameters should not be restricted to discretely sampled values but go beyond any quantization resolution. For example, the position parameters of the object should not be restricted to the pixel grid but should be subpixel precise. The same holds for the object's orientation.

- *Finally, all instances of an object should be found in the image.* The approach should not only find the "best" instance of an object in an image but return all instances that fulfill a predefined criterion. In the remainder of this dissertation found object instances in an image will be referred to as *matches*.

2.3 Concept

In this section, the concept of the proposed object recognition scheme is introduced. The basic idea is that the recognition of compound objects can be seen as a framework in which the recognition of rigid objects is one important component. In Figure 2.10, an overview of the concept at a generalized level is given. The concept is split into three blocks representing three approaches that can be characterized as *camera calibration and rectification* (see Figure 2.10(a)), *recognition of rigid objects* (see Figure 2.10(b)), and *recognition of compound objects* (see Figure 2.10(c)). The type of the graphical representation is chosen so that the input data, the processing steps, and the output data of the three blocks are distinguished. Furthermore, the *offline phase* is visually separated from the *online phase*. In the offline phase, computations that can be done in a preliminary step and have to be carried out only once for a specific object are performed, e.g., creating the model description of the object. Therefore, these computations are not time-critical. In contrast, computations that are performed in the online phase have to be executed whenever the model is used to find the object in the image. Thus, these computations must be performed in real-time. In the following, the three main blocks are introduced and the relations between the blocks are indicated.



(a) Approach for camera calibration and rectification

(b) Approach for recognition of rigid objects

(c) Approach for recognition of compound objects

Figure 2.10: The concept of the object recognition described in this dissertation

The first block represents the camera calibration and the rectification (see Figure 2.10(a)). It is only relevant if the camera was not mounted perpendicular to the plane in which the objects lie or the camera exhibits severe radial distortions. Otherwise this block can be omitted. The idea behind the calibration is to eliminate projective distortions by rectifying distorted images before the images are passed to further processing steps (see Figure 2.10(b) and Figure 2.10(c)). This has the considerable advantage that all further processing steps do not need to concern themselves with projective distortions at all. The disadvantage is that an additional image transformation and a re-sampling step have to be performed, which are, in general, very time consuming. In order to reduce this additional computation time, this process of rectification is split into an offline phase and an online phase. In the offline phase, the camera calibration is computed using several images of a known calibration target and a *rectification map* is derived from the calibration data. This is a time consuming step, but it has to be performed only once for a specific camera pose and a specific object plane. The rectification map can be seen as a kind of look-up table that facilitates a fast rectification of an input image in the online phase. The resulting rectified image is free of radial and projective distortions.

In the second block the general design of an approach for recognizing rigid objects is described (see Figure 2.10(b)). Here, in the offline phase, the rigid model is derived from an image of the object. The image part that shows the object is referred to as *model image* and — if necessary — has been rectified in a preceding step using the rectification map. The rigid model can then be used in the online phase to recognize the object in one or more (rectified) *search images*. While the rectification of the model image in the offline phase is not time-critical the rectification of the search images in the online phase must be performed in real-time.

The third block describes the concept of the approach for recognizing compound objects (see Figure 2.10(c)). Generally, the model of a compound object is referred to as compound model. In the proposed approach the compound model shows a hierarchical structure, which is also indicated by the thesis "Using a *hierarchical model* for the recognition of compound objects provides higher efficiency and inherent determination of correspondence in contrast to standard methods, and hence facilitates real-time applications". Therefore, the compound model that is generated during the offline phase will also be referred to as *hierarchical model*. The hierarchical model generation comprises the extraction of rigid object parts on the basis of the model image and several example images. The most important thing to note is that for each rigid object part a rigid model is generated by employing the offline phase of the recognition of rigid objects (see Figure 2.10(b)). Hence, the offline phase of recognizing rigid objects is embedded in the offline phase of recognizing compound objects. Consequently, the resulting hierarchical model holds a rigid model for each part of the compound object. The relations between the parts and the search strategy for the online phase are automatically derived by analyzing the example images and complete the hierarchical model. Analogous to the offline phase, the online phase of recognizing rigid objects is embedded in the online phase of recognizing compound objects. An important characteristic of the online phase for compound objects is, however, the computation of an individual search space for each object part in order to minimize the search effort. This computation is based on the hierarchical model using the relations between the parts and the derived search strategy.

Consequently, the concept of recognizing compound objects represents a framework in which an approach for recognizing rigid objects is embedded as a substantial part. This modularity facilitates the interchangeability of the latter approach without affecting the concept of recognizing compound objects. Thus, the concept of recognizing compound objects is independent from the chosen embedded approach. As another consequence, the requirements listed in Section 2.2 that do not explicitly refer to compound objects have to be fulfilled, not only by the approach for recognizing compound objects, but also by the approach for recognizing rigid objects.

2.4 Background

In this section, the background and the general external conditions from which the dissertation has originated and under which it was developed are explained. This is essential because these conditions influence several aspects of the work.

The author's work has been supported by the software company *MVTec Software GmbH* (Munich, Germany). Their main product, *HALCON*, represents a machine vision tool that is based on a large library of image processing operators (MVTec 2002). The implementation of the presented approach is partly based on image processing operations that are provided by the HALCON library. The motivation for MVTec Software GmbH in supporting the author's work was, on the one hand, to extend their existing knowledge in the field of object recognition in general. On the other hand, a new approach for the recognition of compound objects that can be directly included in the HALCON library should be developed and implemented. HALCON is mainly applied to specific tasks that arise in industry. A selection of the typical example applications are demonstrated in Section 2.1. Thus, the requirements listed in Section 2.2, and hence the derived concept of this work introduced in Section 2.3, are indirectly influenced by industrial demands.

Two approaches for recognizing rigid objects have been developed approximately simultaneously with the aim of fulfilling the established requirements: on the commercial side, the *shape-based matching* (Steger 2002) has been developed at MVTec Software GmbH, and on the scientific side, the author has developed the *modified generalized Hough transform* in the context of this dissertation (Ulrich et al. 2001b). Because of these close relationships, the developments have not been completely independent of each other but have overlapped in a few areas. Both approaches are introduced in the dissertation, where the main focus is on the modified generalized Hough transform. The overlapping points will only be explained once. However, the approach for recognizing compound objects is then built on the basis of the shape-based matching because the latter has already been thoroughly tested and included in the HALCON library.

2.5 Overview

In the following, a brief overview of the dissertation is given. According to to the concept outlined in Figure 2.10 the next three chapters correspond to the three main tasks. Chapter 3 describes the camera calibration and the rectification. It comprises the introduction of the used camera model, the calibration, as well as the novel rectification process. This chapter is then concluded with a small example. Chapter 4 addresses the recognition of rigid objects. An extensive review of recognition methods is carried out and the generalized Hough transform (Ballard 1981) as a promising candidate is selected and further examined. The drawbacks of the generalized Hough transform are elaborated and analyzed. In the following sections, several novel modifications are introduced to eliminate the drawbacks. The respective modifications are applied, resulting in a modified generalized Hough transform. Finally, after the shape-based matching is introduced, an extensive performance evaluation compares the modified generalized Hough transform and the shape-based matching with several other approaches for the recognition of rigid objects. In Chapter 5 the new approach for recognizing compound objects is explained. A review of the respective literature is followed by an overview that broadly describes the pursued strategy. A more detailed description of the single processing steps is subsequently given focusing on the main novel aspects of this work. This chapter is then concluded with several examples that show the advantages of the new approach. Finally, in Chapter 6 some conclusions are given.

Chapter 3

Camera Calibration and Rectification

Geometric camera calibration is a prerequisite for the extraction of precise 3D information from imagery in computer vision, robotics, photogrammetry, and other areas.

Since in this dissertation only 2D objects are considered, the benefit of using 3D camera calibration for the purpose of 2D object recognition should be addressed first. The first point has already been discussed in Chapter 2 and must be considered when the image plane is not parallel to the plane in which the objects occur, which results in an homographic mapping between the two planes. In order to eliminate the resulting projective distortions in the image, one has to know the 3D poses of both planes in the real world. The second point addresses the problem of lens distortions, i.e., the physical reality of a camera geometrically deviates from the ideal perspective geometry. Therefore, whenever precise measurements must be derived from the image data, these deviations must be considered. In the case of compound objects, quantitative statements about the relative poses of the object parts in the real world must be made. This is important in order to facilitate a correct automatic computation of the hierarchical model. Hence, it is essential to perform a camera calibration in a preceding step. The remainder of this chapter is organized as follows: In Section 3.1, a short review of camera calibration techniques is given in order to select the appropriate method for the task of recognizing compound objects. Section 3.2 describes the applied camera model and the involved parameters and in Section 3.3 the calibration process is briefly explained. In Section 3.4, a novel way to rectify images based on the calibration result that facilitates real-time computation is introduced. The rectified images are free of lens distortions and free of projective distortions of the object plane. Finally, Section 3.5 concludes with an example.

3.1 Short Review of Camera Calibration Techniques

One aspect of camera calibration is to estimate the interior parameters of the camera. These parameters determine how the image coordinates of a 3D object point are derived, given the spatial position of the point with respect to the camera. The estimation of the geometrical relation between the camera and the scene is also an important aspect of calibration. The corresponding parameters that characterize such a geometrical relation are called exterior parameters or camera pose. Thus, the camera parameters describe the interior and exterior orientation of the camera. In this work, camera calibration means to determine all camera parameters. It should be noted that sometimes camera calibration only comprises the determination of the interior camera parameters, as in the field of photogrammetry and remote sensing. Literature provides several methods of camera calibration. In photogrammetry two basic approaches can be distinguished: laboratory methods and field methods (Heipke et al. 1991). The interior orientation of metric cameras is usually determined under laboratory conditions. The interior orientation of metric cameras is constant and the image coordinate system is defined by special fiducial marks within the camera. Field methods can be further subdivided into testfield calibration, simultaneous self calibration, and system calibration. Testfield calibration is carried out for non-

and semi-metric cameras prior to image acquisition. The object coordinates of several control points within the testfield are known and used to derive the orientation of the camera within a photogrammetric block adjustment. In (Ebner 1976), a simultaneous self calibration is presented where the interior orientation parameters are determined simultaneously with the desired object space information. Finally, system calibration combines testfield and simultaneous self calibration where images are acquired that show both the testfield and the object and that are evaluated in one step (Kupfer 1987).

In machine vision, mainly non-metric digital cameras (e.g., off-the-shelf CCD cameras) come into operation because of their lower prices, higher flexibility, and manageable size in contrast to metric and semi-metric cameras. Because their interior orientation is not known à priori and cannot be assumed to be constant, the requirement for laboratory calibration methods is not fulfilled. Hence, in most cases, cameras are calibrated using field methods. The advantages of simultaneous self calibration are its high accuracy and that no control point coordinates in object space need to be known à priori (Wester-Ebbinghaus 1983). However, several images taken from different camera poses must be acquired in order to perform the calibration. This contradicts the claim of the object recognition approach to be simple and easy to use. Camera calibration should be possible in industry, even during system operation without unmounting the camera. Another limitation of simultaneous self calibration is that it requires a high number of corresponding points in the images, which are not available in all cases. The requirements for testfield calibration are less stringent than the requirements for simultaneous self calibration. Nevertheless, high accuracies are also possible. The geometric quality of solid-state imaging sensors was already verified in (Gruen and Beyer 1987), where an accuracy of 1/10th of the pixel spacing was achieved with a planar testfield. In (Heipke et al. 1991), it was shown that the calibration using a 3D testfield even fulfills the stringent accuracy requirements of photogrammetric tasks. Accuracies up to 1/50th of the pixel spacing could be verified with a 3D testfield in (Beyer 1987). Also in (Beyer 1992) and (Godding 1993) 3D testfields are applied for camera calibration. Unfortunately, the construction of the 3D testfield and the precise determination of the object coordinates of the control points within the testfield are very time consuming and costly. Generally, the assignment of the measured image coordinates to the control points must be done manually because the correspondence problem in 3D is difficult to solve. The uncomfortable handling of such targets is another drawback that rules out the use of 3D testfields in this work. Although in general, the accuracy achieved by a planar 2D testfield is lower in comparison to 3D testfields, there are several arguments for preferring the use of a 2D testfield for calibration: it is more robust, much easier to produce, less expensive to gauge, and simpler to transport. Furthermore, the extraction and assignment of the control points in the image can be done automatically since the correspondence problem in 2D is much easier to solve.

In order to perform the camera calibration, one has to select an appropriate camera model where the implemented parameters describe the physical mapping process with sufficient accuracy. According to the literature, different approaches for camera calibration are suitable for different camera models. In (Weng et al. 1992), the existing techniques are classified into three categories. *Direct non-linear minimization* relates the parameters to be estimated with the 3D coordinates of control points and their image plane projections and minimizes the residual errors using an iterative algorithm (Brown 1966, Faig 1975, Wong 1975). The advantages of this type of technique are that the camera model can be very general to cover many types of distortions, and that a high accuracy can be achieved. However, because of the non-linearity of the resulting equations, a good initial guess is required for the iteration. In *closed-form solutions*, parameter values are computed directly through a non-iterative algorithm by defining intermediate parameters that can be determined by solving linear equations. The final parameters are subsequently computed from the intermediate parameters (Abdel-Aziz and Karara 1971, Wong 1975, Faugeras and Toscani 1986). On the one hand this enables a fast computation, on the other hand, in general, distortion parameters cannot be considered and poor results are obtained in the presence of noise. In (Abdel-Aziz and Karara 1971), the direct linear transformation (DLT) has been extended to incorporate distortion parameters. However, the corresponding formulation is not exact. Finally, *two-step methods* involve a direct solution for most of the calibration parameters and some iterative solution for the remaining parameters. In (Tsai and Lenz 1988), a two-step method is presented that is able to handle radial distortions. In (Weng et al. 1992), the two-step approach is extended to also take more general distortions into account and in the second step not only the remaining but all parameters are estimated iteratively.

The radial lens distortions of spherical lenses often cannot be eliminated even when using a lens design that comprises a system of lenses aligned on the optical axis. Radial lens distortions cause concentric circles that are centered at the optical axis to be mapped as circles with distorted radius. The influence of radial symmetric distortions is about one magnitude higher than the influence of other distortions, e.g., decentering distortions or thin prism distortions (Weng et al. 1992). Furthermore, the overall influence of distortions on small images in comparison to photogrammetric images is much lower. Because of these two reasons, it is essential, but also sufficient, for the camera calibration within the object recognition approach to model the radial distortions. Nevertheless, the camera lens should be of a reasonable quality and should not show severe distortions in order to keep the non-modeled part of the distortions as small as possible.

The approach proposed in (Lanser et al. 1995) is able to handle radial distortions and combines the advantages of a 2D testfield with the higher accuracy of the 3D testfield. This is achieved by simultaneously evaluating several images that show the 2D testfield in different distances and poses. The used iterative approach, which is one representative of the direct non-linear minimization technique, allows high accuracies for the desired camera parameters. Because the term “testfield” is unusual in computer vision, the more common term “calibration target” will be used instead. The requirement for several images does not contradict the demand of ease of use, because it is not difficult for a user to move an appropriate 2D calibration target to different poses and take several *calibration images* of it while the camera position remains unchanged. Because the relative pose of the calibration target in the images does not need to be known à priori, the camera calibration can be easily performed in practice, even in mobile or autonomous systems. Based on the above mentioned arguments this approach is chosen to perform the camera calibration within the object recognition approach developed in this dissertation. A detailed description of this approach is given in the following section.

3.2 Camera Model and Parameters

In order to calibrate a camera, a model for the mapping of the 3D points of the world to the 2D image generated by the camera, lens, and frame grabber is necessary. In the approach described in (Lanser et al. 1995) the camera model of (Lenz 1987) is used, where a pinhole camera with radial distortions is assumed. The camera model describes the perspective projection of a 3D world point $\boldsymbol{p}^w$ into the pixel with *r*ow and *c*olumn coordinate $(r, c)^\top$ of the image (see Figure 3.1).

The perspective projection can be divided into three steps. In the first step the 3D world point $\boldsymbol{p}^w$ is transformed from the world coordinate system (WCS) into a 3D point $\boldsymbol{p}^{cam} = (x, y, z)^\top$ of the camera coordinate system (CCS). The transformation is described by the exterior camera parameters, which comprise a rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$:

$$\boldsymbol{p}^{cam} = \boldsymbol{R} \cdot \boldsymbol{p}^w + \boldsymbol{t} \ . \tag{3.1}$$

In the second step, the 3D point $\boldsymbol{p}^{cam}$ in the CCS is projected into the image plane by assuming a pinhole camera and using the equations of perspective transformation:

$$u = f\frac{x}{z}, \quad v = f\frac{y}{z} \ , \tag{3.2}$$

where the effective focal length f is the distance between the image plane and the optical center. Equation (3.2) idealizes the perspective transformation by ignoring any lens distortions. In Figure 3.1 the world point $\boldsymbol{p}^w$ is projected through the optical center of the lens to the point $\boldsymbol{p}$ in the image plane, which is located at a distance of f behind the optical center. If no lens distortions were present, $\boldsymbol{p}$ would lie on a straight line from $\boldsymbol{p}^w$ through the optical center (see Figure 3.1). Lens distortions cause the point $\boldsymbol{p}$ to lie at a different position. Because in the present camera model radial distortions are considered, the “real” point $(\tilde{u}, \tilde{v})^\top$ in the image coordinate system can be calculated by:

$$\tilde{u} = \frac{2u}{1 + \sqrt{1 - 4\kappa(u^2 + v^2)}}, \quad \tilde{v} = \frac{2v}{1 + \sqrt{1 - 4\kappa(u^2 + v^2)}} \ . \tag{3.3}$$

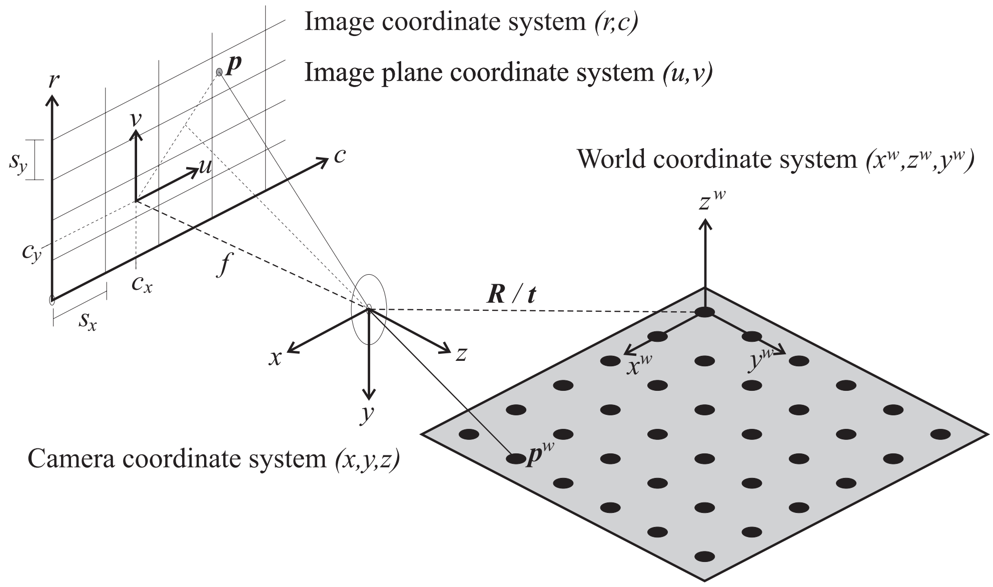


Image coordinate system (r,c)
Image plane coordinate system (u,v)
World coordinate system (x^w,z^w,y^w)
Camera coordinate system (x,y,z)
r
c
v
u
p
s_y
c_y
c_x
s_x
f
R / t
x
y
z
z^w
x^w
y^w
p^w

Figure 3.1: Perspective projection of the world point $\boldsymbol{p}^w$ to the point $\boldsymbol{p}$ in the image plane when using a pinhole camera according to (Lenz 1987)

Here, the radial distortions are described by the parameter κ. If κ is negative, the distortions are barrel-shaped, while for positive κ they are pincushion-shaped. This model for the lens distortions has the advantage that it can be easily inverted in order to calculate the correction of the distortions analytically:

$$u = \frac{\tilde{u}}{1+\kappa(\tilde{u}^2+\tilde{v}^2)}, \quad v = \frac{\tilde{v}}{1+\kappa(\tilde{u}^2+\tilde{v}^2)} \quad . \tag{3.4}$$

Finally, in the third step, the 2D image point $(\tilde{u},\tilde{v})^\top$ is transformed into the pixel $\boldsymbol{p} = (r,c)^\top$ of the image coordinate system:

$$r = \frac{\tilde{v}}{s_y} + c_y, \quad c = \frac{\tilde{u}}{s_x} + c_x \quad , \tag{3.5}$$

where s_x and s_y are scaling factors that describe the horizontal and vertical spacing of the CCD elements on the sensor. The point $(c_x, c_y)^\top$ is the principal point of the image, which also defines the center of the radial distortions. The parameters f, κ, s_x, s_y, c_x, and c_y are independent of the exterior orientation of the camera, and hence represent the interior camera parameters. They describe the intrinsic mapping characteristics of the camera. The parameters f, s_x, and s_y are not independent from each other since a change in f can be compensated by an appropriate change in s_x and s_y. Thus, they cannot be determined simultaneously. Since s_y is usually known because the video signal is sampled line-synchronously by the frame grabber, this datum defect can be eliminated by keeping s_y fixed within the minimization process described in the following section.

Completing this section, it should be mentioned that in some applications cameras with telecentric lenses are used instead of pinhole cameras. In contrast to pinhole cameras, telecentric lenses show no perspective projection but parallel projection, i.e., no perspective distortions occur. The camera model for those kind of lenses can be easily adapted changing equation (3.2) to

$$u = x, \quad v = y \quad . \tag{3.6}$$

As can be seen, the effective focal length is no longer valid for telecentric lenses. Furthermore, the distance z of the object to the image plane has no influence on the image coordinates. Consequently, when using telecentric lenses in object recognition the restriction that objects always must appear in the same 3D plane is dispensable. Even 3D objects can be recognized at arbitrary position if their rotation in 3D object space is restricted to rotations around the viewing direction of the camera, because then the resulting mapping in the image can be described by a simple rigid motion. The diameter of telecentric lenses, however, must have at least the same size as the parallel projection of the extent of the object. The following considerations refer to pinhole cameras but can be easily adapted for the use of telecentric lenses.

3.3 Camera Calibration

In this section, the approach for camera calibration proposed in (Lanser et al. 1995), which uses the model introduced in the previous section, is briefly explained. This approach extends the single image calibration (SIC) to a multi image calibration (MIC) and is able to handle numerous images of the calibration target. At first, the calibration using single images is introduced, and afterwards, the necessary modifications for the multi image calibration are explained.

In the SIC a single image of a 2D calibration target with N circular black colored marks is used (see Figure 3.2). The centers of the circular marks represent the 3D control points. The 2D coordinates of the control points in the planar calibration target are known with high accuracy. The coordinate system of the control points also defines the coordinate system, in which the pose of the camera, i.e., the exterior orientation, is computed. The square border allows the inner part of the calibration target to be found automatically. Circular marks are used because their center point can be determined with high accuracy. The circular marks are mapped to ellipses under projective transformations. It should be noted that, in general, the extracted centers of the ellipses do not coincide with the projected centers of the circular marks. Therefore, an analytical correction must be applied in order to increase the accuracy (Kager 1981). Finally, the array layout of the rows and columns of the circles facilitates the determination of the correspondence of the control points and their projections in the image.

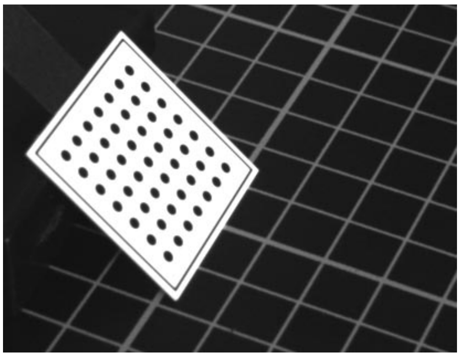


Figure 3.2: 2D calibration target with 49 circular marks

The approach is based on minimizing the sum of squared distances e between the projected 3D centers of the marks and the extracted 2D centers in the image by using a parameter adjustment. Let $\boldsymbol{m}_i$ be the 3D centers of the marks and $\tilde{\boldsymbol{m}}_i$ the extracted 2D centers in the image. Each $\boldsymbol{m}_i$ is projected into a (sub)pixel point in the image using the projection $\pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$, which is a function of $\boldsymbol{x} = (\boldsymbol{R}, \boldsymbol{t}, f, \kappa, s_x, s_y, c_x, c_y)^\top$ and comprises equations (3.1), (3.2), (3.3), and (3.5):

$$e(\boldsymbol{x}) = \sum_{i=1}^{N} \|\tilde{\boldsymbol{m}}_i - \pi(\boldsymbol{m}_i, \boldsymbol{x})\|^2 \longrightarrow \min \quad . \tag{3.7}$$

Because of the given non-linear relationship, suitable starting values for the unknown camera parameters must be known. Starting values for the interior camera parameters can be obtained from the specifications of the

CCD sensor and the lens. For the calculation of the starting values for the exterior camera parameters the interested reader should refer to (Lanser et al. 1995).

The exact camera parameters can then be calculated using standard linear least-squares optimization routines (Koch 1987) or preferably non-linear methods, e.g., based on the Levenberg-Marquardt algorithm (Levenberg 1944, Marquardt 1963, Press et al. 1992), because of their better convergence behavior, and hence higher tolerance to unprecise starting values.

During the minimization process within the SIC, flat minima often occur because of correlations between the camera parameters. This problem occurs, for example, if the 3D plane of the calibration target is approximately parallel to the image plane. In order to stabilize the result and to improve the accuracy, the SIC is extended to evaluate several calibration images of the calibration target simultaneously resulting in the MIC. A similar approach can also be found in (Beardsley et al. 1992).

To get an accurate result, several images (in practice, 10–15 images are sufficient) should be taken, in which the calibration target is moved so that all degrees of freedom are used while the camera remains fixed. The detection of the marks in the image and the calculation of the starting values of the exterior camera parameters is performed independently for each calibration image. It is important to note that, although the camera remains fixed, the exterior camera parameters are specific for each image. This is because the world coordinate system, in which the pose of the camera is described, is defined by the calibration target, which moves from image to image. During the adjustment within the MIC, the different exterior camera parameters of all images and the constant interior camera parameters are estimated simultaneously using one of the above mentioned optimization routines. The experiments in (Lanser et al. 1995) showed that the maximum distance between the projected 3D centers of the marks and the corresponding image points was about 1/10th of a pixel after the adjustment. A camera with a 2/3 inch sensor, a 12.5 mm lens, and images of size 768 $\times$ 576 was used in these experiments.

3.4 Rectification

In this section, the novel method for rectifying images in real-time is introduced. The rectification is based on the previously determined camera parameters and eliminates projective as well as radial lens distortions. To achieve real-time computation, the projection is split into two phases. In the offline phase, the mapping of the projection is computed based upon the camera parameters, and stored within a look-up table, which is called *rectification map*. In the online phase, the rectification map can be applied to an image (*original image*) in order to rectify the image in a very short period of time.

3.4.1 Computation of the Rectification Map

The rectification process corresponds to a projection of the image onto a 3D projection plane in the WCS.

The projection plane, which corresponds to the rectified image, is defined in the WCS with respect to the 3D plane of the calibration target as shown in Figure 3.3. Thus, the calibration target should be placed on the object plane in at least one of the images that are used for calibration. The origin of the WCS can be translated by a vector $\boldsymbol{o} = (o_x, o_y, o_z)^\top$ to an arbitrary position that defines the position of the upper left pixel of the rectified image. If an appropriate value for o_z is chosen, planes parallel to the calibration target can be rectified. Therefore, it is, for example, possible to take the thickness t of the calibration target into account: in order to rectify the object plane and not the plane described by the calibration target, o_z must be set to $-t$. In order to complete the definition of the rectified image, one has to specify the pixel size d in world units and the size of the rectified image by the number of rows R and the number of columns C. In general, the value for d should be chosen according to the information content of the original image, i.e., the pixels in the rectified image should approximately have the same size (in object space) as in the original image on average. This

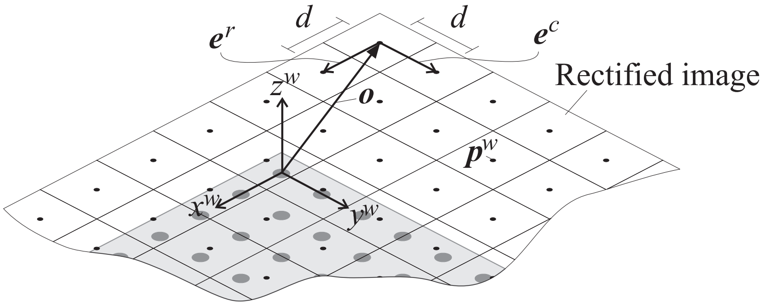


Figure 3.3: The plane of the calibration target is used to define the plane of the rectified image. Each pixel of the rectified image can be computed in 3D world coordinates ($\boldsymbol{p}^w$) as a linear combination of the two 3D basis vectors $\boldsymbol{e}^r$ and $\boldsymbol{e}^c$. The parameter d corresponds to the length of $\boldsymbol{e}^r$ and $\boldsymbol{e}^c$, respectively, and defines the pixel size of the rectified image in object space in world units, e.g., meters.

minimizes the loss of information while keeping the rectified image of manageable size. The pixel $\boldsymbol{p}^w(r', c')$, $r' = 0, \ldots, R-1$, $c' = 0, \ldots, C-1$ of the rectified image can be calculated in the WCS:

$$\boldsymbol{p}^w(r', c') = \boldsymbol{o} + r' \cdot \boldsymbol{e}^r + c' \cdot \boldsymbol{e}^c \ , \tag{3.8}$$

where $\boldsymbol{e}^r$ and $\boldsymbol{e}^c$ denote the 3D basis vectors of the spanning rectification plane with length d:

$$\boldsymbol{e}^r = (0, d, 0)^\top, \quad \boldsymbol{e}^c = (d, 0, 0)^\top \ . \tag{3.9}$$

The 3D point $\boldsymbol{p}^w(r', c')$ can then be transformed into the 2D image point $\boldsymbol{p} = (r, c)^\top$ of the original image using equations (3.1), (3.2), (3.3), and (3.5). Since the transformation of the WCS into the CCS is a 3D rigid transformation, angles are preserved by the transformation. Therefore, it is possible to speed up the computation by pre-calculating the translated origin $\boldsymbol{o}$ and the two 3D basis vectors $\boldsymbol{e}^r$ and $\boldsymbol{e}^c$ in the CCS using equation (3.1):

$$\boldsymbol{o}^{cam} = \boldsymbol{R} \cdot \boldsymbol{o} + \boldsymbol{t}, \quad \boldsymbol{e}^{r,cam} = \boldsymbol{R} \cdot \boldsymbol{e}^r, \quad \boldsymbol{e}^{c,cam} = \boldsymbol{R} \cdot \boldsymbol{e}^c \ . \tag{3.10}$$

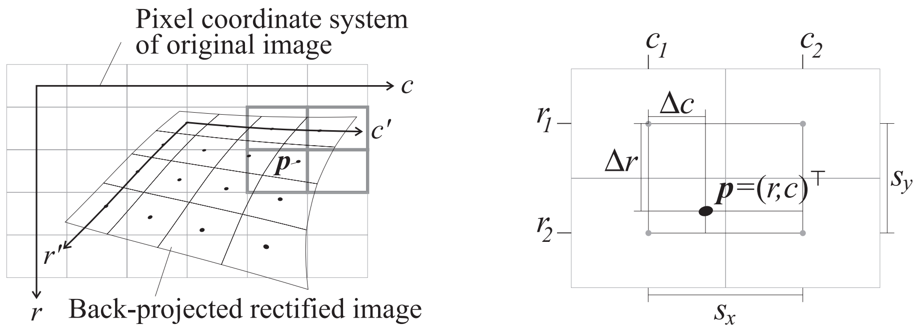
Thus, the pixels of the rectified image can be directly obtained in the CCS

$$\boldsymbol{p}^{cam}(r', c') = \boldsymbol{o}^{cam} + r' \cdot \boldsymbol{e}^{r,cam} + c' \cdot \boldsymbol{e}^{c,cam} \tag{3.11}$$

and transformed into the 2D image point $\boldsymbol{p} = (r, c)^\top$ of the original image by using only equations (3.2), (3.3), and (3.5). This procedure is applied to all pixels in the rectified image yielding a back-projected rectified image within the original image (see Figure 3.4(a)). This *backward projection* is essential, because in the *forward projection*, where the original image is projected into the rectification plane, gray values of pixels in the rectified image may remain undefined.

In general, the resulting row and column coordinates (r, c) of the 2D image points $\boldsymbol{p}$ are not integer values. Therefore, the gray value of the rectified image must be re-sampled from the original image using the gray values in the neighborhood of $\boldsymbol{p}$. Bilinear interpolation is a common re-sampling method. It represents a compromise between computation time and accuracy in comparison to other methods like nearest-neighbor or higher order polynomial interpolation (Mikhail et al. 2001). The method uses the four neighboring pixels of $\boldsymbol{p}$ as sampling points: $\boldsymbol{p}_1 = (r_1, c_1)^\top$, $\boldsymbol{p}_2 = (r_1, c_2)^\top$, $\boldsymbol{p}_3 = (r_2, c_1)^\top$, and $\boldsymbol{p}_4 = (r_2, c_2)^\top$, where $r_1 = \lfloor r \rfloor$, $r_2 = \lceil r \rceil$, $c_1 = \lfloor c \rfloor$, and $c_2 = \lceil c \rceil$. Figure 3.4(b) illustrates the principle of bilinear interpolation. Let $g(r, c)$ and $g'(r', c')$ be the gray value of the original image and the rectified image, respectively. Then $g'(r', c')$ can be finally obtained applying bilinear interpolation:

$$g'(r', c') = w_1 g(r_1, c_1) + w_2 g(r_1, c_2) + w_3 g(r_2, c_1) + w_4 g(r_2, c_2) \ . \tag{3.12}$$



(a) Rectified image projected back into image plane (b) Bilinear interpolation

Figure 3.4: The rectified image is projected back into the original image (a) and re-sampled (b). It should be noted that because of radial distortions, lines are not necessarily projected to lines.

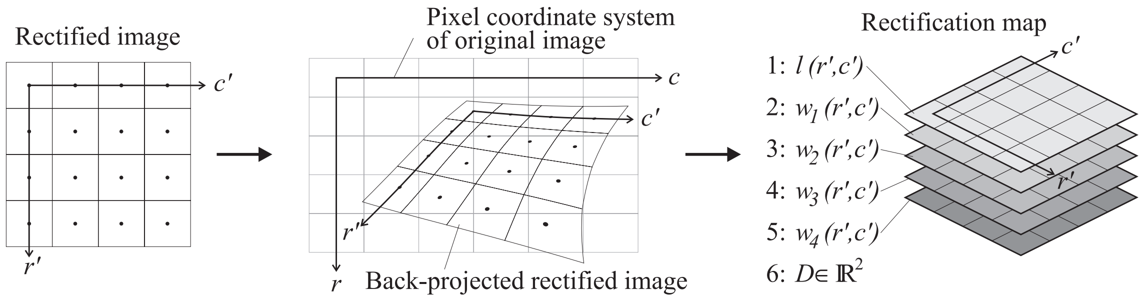


Figure 3.5: The process of computing the rectification map is summarized in two steps. In the first step, the pixels of the rectified image are projected back into the original image. In the second step, the linearized coordinates l of the projected pixels, the four corresponding interpolation weights w_i, and the domain D of the rectified image are stored within the rectification map.

The weights w_i can be computed as

$$w_1 = (1 - \Delta r)(1 - \Delta c), \quad w_2 = (1 - \Delta r)\Delta c, \quad w_3 = \Delta r(1 - \Delta c), \quad w_4 = \Delta r \Delta c \ , \tag{3.13}$$

where $\Delta r = r - r_1$ and $\Delta c = c - c_1$. As one can see, the rectification of an image is a computationally expensive procedure starting with the projection and ending with the computation of the interpolation weights. Hence, real-time computation is impossible to achieve when using this straightforward method. A more sophisticated technique exploits that the camera pose is constant during the acquisition of images for object recognition in most cases. Therefore, the back-projection of the pixels of the rectified image into the original image remains unchanged, and hence the interpolation weights are constant for each pixel of the rectified image. Consequently, $r(r', c')$, $c(r', c')$, and the corresponding weights $w_i(r', c')$ only depend on the pixel coordinates of the rectified image and can be stored in the rectification map. The rectification map, which is displayed in Figure 3.5, consists of five 2D arrays that are of the same size as the rectified image itself ($R \times C$). In the first array, the *linearized coordinate* l of $\boldsymbol{p}_1$ is stored, which can be calculated as $l(r', c') = r_1(r', c') \cdot C + c_1(r', c')$. The linearized coordinate is used because images are stored as linearized arrays, which has the advantage of a higher efficiency. From the linearized coordinate and the known image width C, the row and column coordinates can be recalculated. The remaining four arrays hold the interpolation weights $w_i(r', c')$.

Because, in general, the forward projection of the rectangular outline of the original image does not result in a rectangular outline in the rectified image, additionally, the image domain $D \subset \mathbb{R}^2$ of the rectified image must be stored within the rectification map. The image domain only contains those pixels of the rectified image that are mapped into the (rectangular) domain of the original image, and can be efficiently stored using run-length encoding (Haberäcker 1995, Jähne 2002). The processes described so far can all be attributed to

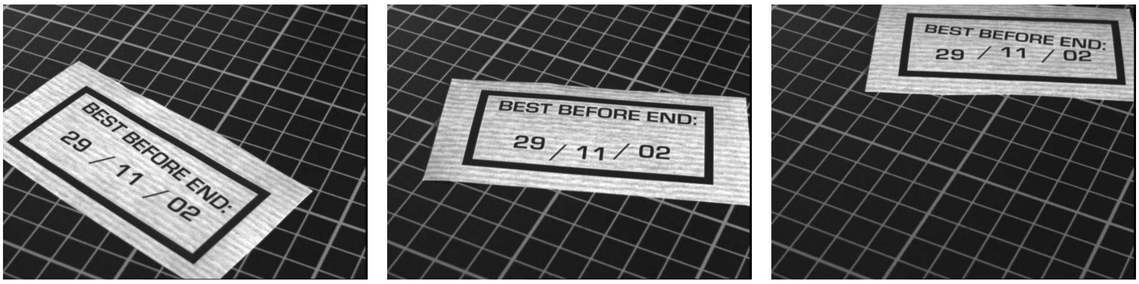


Figure 3.6: Deviations from the nadir view cause projective distortions in the object plane.

the offline phase and only need to be executed once for a specific camera and object plane configuration. In the following, the online phase of the rectification is described.

3.4.2 Rectification Process

The input data of the online phase, i.e., in the rectification process, are the image that must be rectified and the rectification map that holds all necessary information for the rectification. The derivation of the rectification map already implies the process of computing the output data, i.e., the rectified image. Therefore, only some short remarks are given. At first, a new image is created where the image domain is set to the image domain of the rectification map D. Then, for each pixel $(r', c')^\top$ within the image domain the gray value $g'(r', c')$ is computed only from the four corresponding gray values of the original image $g(r, c)$, $g(r, c+1)$, $g(r+1, c)$, and $g(r+1, c+1)$ multiplied by the respective weights $w_i(r', c')$, $i = 1, \ldots, 4$, using equation (3.12). The coordinates r and c are obtained from the linearized coordinate $l(r', c')$ and the image width. Thus, this extremely lean computation comprises the entire process of rectification in a compressed form and facilitates real-time computation. The rectified image is not only free of radial and projective distortions in the object plane but — as an additional feature — also exhibits square pixels since the pixel spacing is estimated independently during camera calibration in each direction. Non-square pixels would also lead to deformations of an object in the image. This deformation is not constant but dependents on the object orientation, and hence is critical for object recognition.

3.5 Example

In this section, an example is given, not only to prove the high efficiency of the rectification method, but also to accentuate the need for considering the radial distortions within the camera model.

The label of the example application of Figure 2.7 in Section 2.1 is taken as the test object. Here, it is assumed that it is not possible to mount the camera perpendicular to the object plane. This causes projective distortions of the object plane as can be seen from Figure 3.6, where the “deformation” of the label depends on the pose of the label in the image.

In order to rectify the images, the camera must be calibrated. Here, a calibration target of size 3 cm × 3 cm was used and images that show the target in 15 different poses were taken. Figure 3.7 shows three of the 15 calibration images (640 × 482 pixels). The 3D coordinates of the center points of the black circles are known and used as control points within the camera calibration as described in Section 3.4. The computation time for the whole camera calibration was about 2 s on a 2 GHz Pentium 4, including the segmentation of the calibration target, the calculation of the starting values in all images, and the final parameter estimation using the Levenberg-Marquardt method (Press et al. 1992). The resulting interior camera parameters in this example are $f = 13\,\text{mm}$, $\kappa = -1095.2\,[1/\text{m}^2]$ (i.e, slightly barrel-shaped distortions), $s_x = 7.3\,\mu\text{m}$, $s_y =$

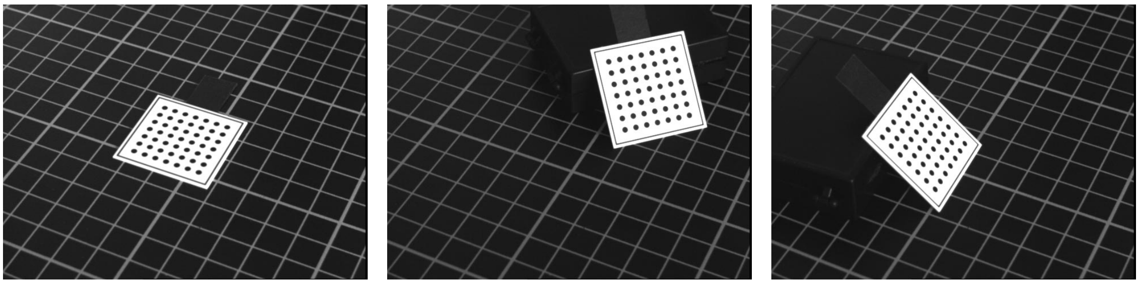


Figure 3.7: Three examples of the 15 calibration images that were taken from the calibration target. The extracted ellipse centers of the projected marks are assigned to the given 3D circle centers of the calibration target and used to estimate the camera parameters. The calibration target in the left image is used to define the object plane to be rectified.

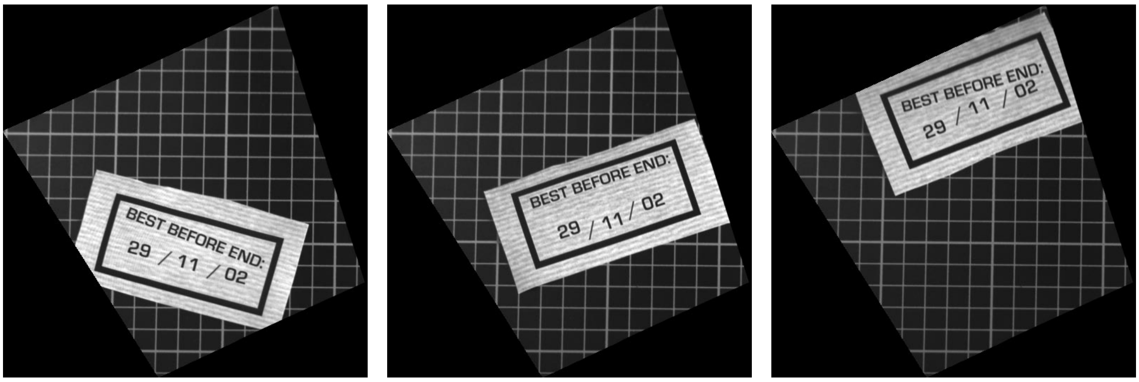


Figure 3.8: Rectified versions of the images shown in Figure 3.6. The average rectification time was 8.3 ms on a 2 GHz Pentium 4.

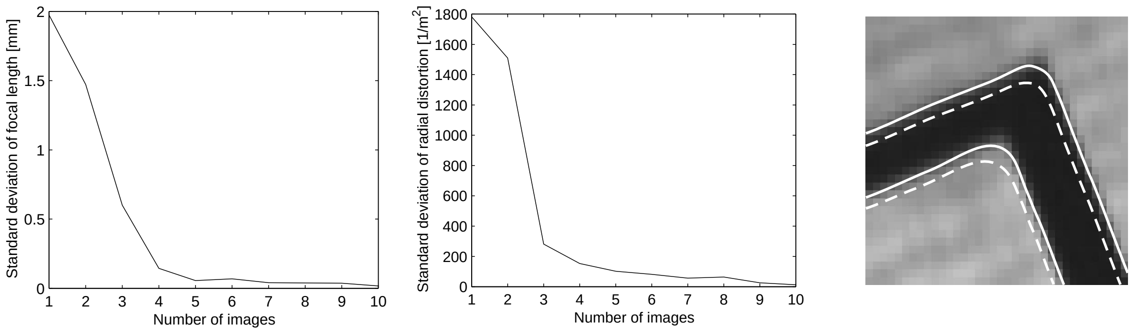
7.3 μm, $c_x = 321.422$ pixels, and $c_y = 236.303$ pixels. To get an impression of the achieved accuracy of the estimation, the 3D control points are projected back into the image using the estimated parameters and compared to the extracted ellipse centers resulting in a mean distance of 0.027 pixels.

Now, the rectified image can be defined. The pose of the calibration target in the left image of Figure 3.7 defines a plane that is offset from the object plane by the thickness $t = 0.63$ mm from the calibration target. Therefore, the z component of the origin o_z must be set to -0.63 mm in order to correctly rectify the object plane. Additionally, the pixel size d is chosen to be 0.32 mm to obtain a suitable size of the rectified image. To ensure that the upper left pixel of the original image is contained in the image domain D of the rectified image, o_x and o_y are set to -9.4 cm each. Finally, the height R and width C are chosen appropriately ($R = 527$, $C = 515$). Now, the computation of the rectification map can be performed using the interior camera parameters, the exterior camera parameters according to the left image of Figure 3.7, and the chosen values for $\boldsymbol{o}$, d, R, and C. In the example it took approximately 120 ms to compute the rectification map.

Since all time-consuming computations of the offline phase have been completed, now, in the online phase images can be rectified in real-time. The three example images of Figure 3.6 are rectified by applying the computed rectification map in an average time of only 8.3 ms per image. The result is shown in Figure 3.8.

Finally, two experiments are carried out to show that firstly, the achievable accuracy of the camera parameters depends on the number of acquired calibration images and that secondly, it is essential to include the radial distortions within the camera model.

For the first experiment, a basic set of 10 selected calibration images out of the entirety of 15 images is formed. Then the number of images n^I is varied from 1 to 10 and for each n^I, 10 sub-sets containing n^I images are taken from the basis set. It should be noted that only one sub-set of 10 images can be formed if $n^I = 10$. The camera calibration is then performed for the different number of images for each of the 10 sub-sets. The



(a) Standard deviation of f (b) Standard deviation of κ (c) Effect of radial distortions

Figure 3.9: Standard deviation of the estimated effective focal length (a) and the radial distortions (b) versus the number of images. In (c) subpixel precise edges are shown, once taking the radial distortions into account (white solid), and once ignoring the radial distortions (white dashed).

standard deviation of the estimated interior camera parameters are then calculated for each n^I. As an example of the result, the standard deviation of the effective focal length f and of the parameter κ, which describes the radial distortions, are plotted in Figure 3.9(a) and Figure 3.9(b), respectively. From both plots it appears that even for a small number of calibration images, the accuracy of the camera calibration increases considerably in comparison to the SIC. In this example, the calibration using only a single image is not meaningful: the standard deviation of κ is $\sigma_\kappa = 1782.0\,1/\mathrm{m}^2$, which is too high for precise measurements. This is because the used calibration target is relatively small and the radial distortions within a small part of the image are approximately constant, and hence hard to determine. Therefore, it is essential to base the calibration on several images where the calibration target should appear in different parts of the image. The same holds for the effective focal length ($f = 13\,\mathrm{mm}$), the standard deviation of which is $\sigma_f = 2\,\mathrm{mm}$. Generally, the achievable accuracy depends on the number of used images and on the accuracy of the extracted ellipse centers in the calibration images. However, it is also apparent from the plots, that continuously increasing the number of images does not necessarily increase the accuracy much. The maximum achievable accuracy is restricted by the distortions that are not considered in the camera model.

For the second experiment, the rectified image of the label that is displayed in Figure 3.8 on the right is used to demonstrate the effect of radial distortions. In Figure 3.9(c), the enlarged detail of the image containing the upper right corner of the label is shown. This image part is far enough away from the principal point to exhibit significant radial distortions. Subpixel precise edges are extracted in the image using the approach presented in (Steger 1998) and visualized as white solid lines. In a second step, the image is rectified with κ set to 0 when computing the rectification map. In the resulting rectified image, subpixel precise edges are again extracted and overlaid on the image of Figure 3.9(c) as white dashed lines. From the displacement of the edges, it is obvious that ignoring the radial distortions in this example would lead to position errors of up to 2 pixels. In general, the position errors induced by radial distortions are in the range of 1 to 4 pixels using off-the-shelf lenses. Thus, it is absolutely essential to model and eliminate the radial distortions in the images.

Chapter 4

Recognition of Rigid Objects

This chapter deals with the recognition of rigid objects. An extensive review of the respective literature is given (Section 4.1). The novel approach for recognizing rigid objects, the modified generalized Hough transform, is explained in detail (Section 4.2). Furthermore, a second new approach, the shape-based matching, which is proposed in (Steger 2002), is introduced (Section 4.3). The shape-based matching is not included in the review because of its exceptional position within this dissertation: firstly, some aspects of its development are closely related to the modified generalized Hough transform and secondly, it is used as module within the implementation of the proposed approach for recognizing compound objects. Finally, an extensive performance evaluation compares the modified generalized Hough transform and the shape-based matching with several existing approaches (Section 4.4).

4.1 Previous Work

Visual object recognition cannot be formulated as a single problem. Too many different paths lead to visual object recognition, resulting in a diversity of approaches to the problem. In (Ullman 1989), some of these paths are listed: the recognition of an object can be done either on the basis of its characteristic shape, color and texture, location relative to other objects, or characteristic motion, for example. These primarily visual stimulations can be supplemented by other sources, e.g., by the use of prior knowledge, expectations, and temporal continuity, or other reasoning. In this section, the diversity of object recognition approaches is reviewed. The focus is set on approaches that are based on shape, color, or texture. These are the most common and important aspects of visual recognition, and therefore of object recognition.

4.1.1 Classification of Object Recognition Approaches

Although object recognition is quite natural, it is difficult to define the term “object recognition” in a simple, precise, and uncontroversial manner. Sometimes object recognition means identifying individual specific objects. For example, in the industrial applications introduced in Chapter 2, individual objects like a specific IC, the “MVTec” logo, or a specific metal part are to be recognized. In other cases, recognition means identifying the object as a member of a certain class. For instance, in photogrammetry and remote sensing, houses, roads, cars, vegetation, or other classes of objects must be recognized in aerial or satellite imagery. A global frame including a detailed review and evaluation of automatic object extraction from aerial imagery is given in (Mayer 1998). This kind of recognition requires the existence of object properties that are, on the one hand, specific enough to distinguish between different classes of objects but are, on the other hand, also general enough to comprise the object variety within one class. Another frequently arising task in object recognition is to decide which object or which class of objects is present in the image. This can be seen as a classification process that is based on a large database of objects or object classes. In (Cohen and Guibas 1997), for instance, Chinese characters are recognized in images by comparing them to a database of images comprising 500 possible characters.

In the scope of the present work, object recognition means identifying individual specific objects and not classes of objects and, furthermore, is not interpreted as a classification process. Nevertheless, even in this restricted domain, several terms are commonly used in the literature. The terms imply different applications, but can all be summarized under “object recognition”. In Table 4.1, some examples are listed where the terms “object” and “recognition” are treated separately: each of the two terms can be replaced by one of various expressions. For example, “object” is often substituted by “image”, “pattern”, or “target”, while “recognition” is often substituted by “alignment”, “matching”, or “detection”. These expressions can be combined arbitrarily within the same row of Table 4.1, resulting in various paraphrases for the term “object recognition”. The table further discriminates between terms that imply a 2D approach (first row) and terms that are used in both, 2D and 3D approaches (second row). For example, “image alignment” assumes a 2D approach, while the term “target recognition” is used in 2D and 3D approaches.

	object	*recognition*
2D	image	alignment / registration / retrieval / comparison / matching
2D / 3D	object / pattern / template / target / shape	recognition / matching / localization / detection / alignment

Table 4.1: “Object recognition” — a term including various semantics

Apart from the classification of object recognition approaches that distinguishes between 2D and 3D approaches, several other classification schemes are possible. In (Ullman 1989), object recognition approaches are divided into *invariant properties methods*, *object decomposition methods*, and *alignment methods*, where also combinations between the methods are allowed. In invariant properties methods, the overall recognition process is broken down into the extraction of a number of different properties (that are invariant to a particular kind of transformation), followed by a final decision based on these properties. Object decomposition methods rely on the decomposition of objects into constituent parts, where in a first stage the parts are recognized and in a second stage the initial classification of the individual parts is used to recognize the object itself by including knowledge about the relations between the parts. For example, in the first stage, straight lines are searched in the image, which can be used in the second stage to recognize rectangles by exploiting knowledge about the relations between the lines. Finally, alignment methods try to find a particular transformation that will maximize some *similarity measure*. The similarity measure is often called *similarity metric* if it fulfills the requirements for a metric, i.e., non-negativity, symmetry, and triangle inequality (Bronstein et al. 2001). The similarity measure describes the similarity between the transformed model and the image. The model is systematically compared to the image using all degrees of freedom of the chosen class of transformations. The maxima of the similarity measure are used to decide whether an object is present in the image and to determine its pose.

Another way to classify object recognition methods in general, and alignment methods in particular, is to use the criteria *feature space*, *search space*, *search strategy*, and *similarity metric* (Brown 1992). Feature space describes the information that is used for object recognition (e.g., gray values, points, lines, elliptic arcs, etc.). Search space is the class of transformation the object is allowed to undergo in the image. Search strategy describes how to find the correct transformation within the search space and, finally, similarity metric is used to evaluate the transformations.

In the following, the criterion *feature space* is used to classify the reviewed approaches into *intensity-based*, *low level feature-based*, and *high level feature-based*. In intensity-based approaches, the raw gray values are used as feature, low level features are, e.g., edge pixels, points, or gray value gradients, and high level features comprise lines, polygons, elliptic arcs, or object parts for instance. The approaches that use intensity or low level features are further subdivided into those that use area-based and non-area-based strategies, respectively. In *area-based strategies* each pixel in the model image and in the search image is involved in the matching process, whereas in *non-area-based strategies*, objects are represented in a more efficient way. The advantage of this classification is that, in general, the algorithm's complexity and the computational complexity of the object recognition approach can be directly deduced from its corresponding class. While intensity-based approaches often use simple algorithms, low and especially high level features often demand a more complex class of object recognition approaches. Furthermore, approaches that involve area-based strategies are often connected with a high computational complexity, while non-area-based strategies in general facilitate faster computations. Since in this dissertation it is differentiated between the recognition of rigid objects and the recognition of compound objects there are two sections about previous approaches. In the following, only approaches dealing with the recognition of rigid objects will be reviewed, while approaches dealing with compound objects will be reviewed in Chapter 5. Because of the almost unlimited number of different approaches, in the following subsections only the most important approaches for recognizing rigid objects will be included in this review.

4.1.1.1 Approaches Using Intensity Information

The simplest class of object recognition methods is based on the raw intensity information of the model image and the search image. In the case of 8-bit gray scale images, the intensity information is given by the gray values in the image. These methods are also often called *direct methods* because they recover the unknown pose parameters directly from the measurable image quantities at each pixel in the images (Irani and Anandan 1999). This is in contrast to the feature-based methods, which first extract a set of distinct features from the images, and then recover and analyze their correspondences.

Alignment methods are probably the most popular methods that are used for object recognition. They can be used in both intensity-based approaches as well as in feature-based approaches. A comprehensive survey is given in (Brown 1992). The principle of alignment methods will be explained in the following. Let $\hat{I}^m$ be the model image with area of definition $\hat{D}^m \subset \mathbb{R}^2$ and I^s the search image with area of definition $D^s \subset \mathbb{R}^2$, respectively[1]. The alignment method transforms $\hat{I}^m$ and $\hat{D}^m$ to all discrete transformations $T_i \in \mathcal{T}$ within a specific transformation class $\mathcal{T}$ (e.g., translations, rigid, similarity, affine, or perspective transformations) and computes the similarity measure $M(T_i)$ between the transformed model images $I^m = T_i(\hat{I}^m)$ within $D^m = T_i(\hat{D}^m)$ and the search image for each of the discrete transformations. Thus, a high similarity results in a peak in M. The transformations T_i that result in local maxima of $M(T_i)$ and that exceed a certain threshold are treated as matches and represent the poses of the object instances in the image. Consequently, the run time complexity of alignment methods can be formalized as $O(|T_i|o^m)$, where $|T_i|$ denotes the number of discrete transformations and o^m the number of operations performed in the respective similarity measure. What also follows is that the accuracy of the obtained pose parameters is restricted to the resolution of the sampling that is applied to $\mathcal{T}$ in order to get the discrete transformations T_i. This is insufficient for many applications (cf. Section 2.2). Unfortunately, the resolution of the sampling cannot be chosen arbitrarily fine because the finer the chosen resolution the more discrete transformations must be computed and checked for similarity, which leads to an increasing computation time. There are several more practical methods to refine the pose parameters. Some of them will be introduced in Section 4.1.2. Furthermore, it is often useful to separate the translation part $\mathcal{T}^t$ of $\mathcal{T}$ from the remaining transformations $\mathcal{T} \setminus \mathcal{T}^t$ when dealing with alignment methods. The model image that has been transformed by the remaining transformations is translated according to the pixel grid of the search image. This avoids unfavorable gray value re-sampling. In order to simplify further discussions, $\mathcal{T}$ is assumed to only represent translations. Consequently, M only depends on the translation parameters $(x, y)^\top \in D^s$ that translate the model image within the search image.

[1] Throughout the remainder of this dissertation the area of definition will also be referred to as *domain* or *region of interest* (ROI)

Area-Based Strategies. One of the most frequently applied intensity-based similarity measures in alignment methods is the *cross correlation* (Brown 1992). It is calculated using the following expression:

$$CC(x,y) = \sum_{(u,v)\in D^m} I^m(u,v) I^s(x+u,y+v) \ . \tag{4.1}$$

This corresponds to an image convolution of I^s with a non-mirrored convolution mask I^m and approximately describes the squared Euclidian gray value distance (Cha 2000). The run time complexity of this kind of alignment methods is $O(n^s n^m)$, where $n^m = |D^m|$ and $n^s = |D^s|$ denote the number of image points (pixels) in the domain of the model image and the search image, respectively ($|T_i| = n^s$, $o^m = n^m$). On the one hand, this similarity measure always returns a high value for bright search image regions, no matter how similar the images are. On the other hand, it is not invariant to changes in brightness. Therefore, in most applications its normalized version (NCC) is applied: the mean μ and the standard deviation σ of the gray values in the model image and in the search image, respectively, are incorporated to achieve invariance to linear changes in brightness, i.e., $NCC(x,y) = 1 \Rightarrow I^s(x+u,y+v) = aI^m(u,v) + b$, $(a > 0)$:

$$NCC(x,y) = \frac{1}{n^m} \sum_{(u,v)\in D^m} \left(\frac{I^m(u,v) - \mu^m}{\sigma^m} \cdot \frac{I^s(x+u,y+v) - \mu^s(x,y)}{\sigma^s(x,y)} \right) \ . \tag{4.2}$$

This zero-mean normalized cross correlation, which corresponds to the well-known *correlation coefficient*, has the additional advantage that it maps the similarity between images to an interval of $[-1,+1]$, where 1 denotes a perfect match. It will be simply referred to as *normalized cross correlation* in the following. Other forms of normalizing the cross correlation can also be found in literature, e.g., (Aschwanden and Guggenbühl 1992, Martin and Crowley 1995), each with its own characteristics.

The computation of the normalized cross correlation is relatively expensive. Because of its convolution character, the computation of the unnormalized cross correlation can be accelerated by transforming I^m and I^s into the Fourier domain (e.g., by using the FFT (Fast Fourier Transformation) (Ballard and Brown 1982)) and multiplying the resulting Fourier transforms (Anuta 1970): $CC = \mathcal{F}^{-1}\{\mathcal{F}(I^s)\mathcal{F}^*(I^m)\}$, where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transform and the inverse Fourier transform, respectively. $\mathcal{F}^*$ symbolizes the conjugate complex and accomplishes the reversal of I^m to get a mirrored filter mask. Unfortunately, the normalized form of cross correlation does not have a simple and efficient frequency expression and is therefore much more difficult to compute. Several methods that try to overcome this problem have been developed, e.g., *high-pass filtering* (Lewis 1995, Jähne 2002), *phase correlation* (Kuglin and Hines 1975, Foroosh et al. 2002), or a subsequent normalization after the convolution (Lewis 1995). Because of the problems that occur when dealing with correlation in the frequency domain, the normalized cross correlation is still computed in the spatial domain in most cases (Gonzalez and Woods 1992). Thus, several methods have been investigated to speed up the computation in the spatial domain, e.g., by appropriately rearranging (4.2) or by using a *pseudo-correlation* (Radcliffe et al. 1993). However, the major drawback of approaches using the normalized cross correlation is its limited robustness against occlusions and clutter as well as against non-linear changes in brightness. Attempts have been made to improve the alignment reliability by utilizing image preprocessing prior to the execution of the normalized cross correlation (Bacon 1997). Unfortunately, there is no suitable preprocessing strategy that is appropriate for the general case. Furthermore, the robustness against clutter cannot be improved by such methods.

Another class of alignment methods uses the *sum of absolute or squared gray value differences* (Brown 1992) or one of its various derivatives. For instance, the sum of absolute gray value differences SAD is computed as follows:

$$SAD(x,y) = \frac{1}{n^m} \sum_{(u,v)\in D^m} |I^m(u,v) - I^s(x+u,y+v)| \ . \tag{4.3}$$

Thus, SAD indicates the average difference of the gray values and therefore represents a measure that maps the dissimilarity between two images to a positive value that is not limited by an upper boundary (apart from the maximum gray value that can occur). This complicates making statements about the quality of matches.

In contrast to the correlation it is a measure of dissimilarity. Nevertheless, it will subsequently be referred to as similarity measure because simple negation or inversion can be applied to straighten out this dilemma. Like the normalized cross correlation, the sum of gray value differences is also sensitive to occlusions and clutter. However, it is faster to compute. Additionally, some speed-ups can be incorporated. For example, in (Cha 2000), a method that quickly eliminates most transformations from consideration is proposed. Hence, the computation of the sum of gray value differences can be restricted to the remaining transformations.

In contrast to cross correlation, the sum of gray value differences can be made invariant only either to additive changes in brightness (e.g., $\sum |(I^m(u,v) - \mu^m) - (I^s(x+u, y+v) - \mu^s)|$) or to local multiplicative changes (e.g., $\sum |I^m(u,v) - \mu^m / \mu^s I^s(x+u, y+v)|$). However, linear changes in brightness cannot be taken into account in a simple way. In (Lai and Fang 1999c), the sum of gray value differences is made robust against linear changes in brightness by modeling the varying brightness with low-order polynomials. Robustness against a moderate amount of occlusion and clutter is obtained by computing the similarity measure in a statistically robust manner.

Several experimental results of a comparative study regarding the normalized cross correlation and the sum of gray value differences can be found in (Aschwanden and Guggenbühl 1992). Also different modifications and normalization techniques are included in the study. They analyze the robustness of the similarity measures by simulating different types of image distortions. It is shown that similarity measures that are based on the normalized cross correlation are more robust against all types of tested distortions than similarity measures that are based on gray value differences. In particular, changes in lighting conditions cause problems when using difference-based measures. An alleviation to this problem is found in the use of more complicated normalization techniques. However, under certain assumptions the computational requirements to implement these functions are only slightly better than for the normalized cross correlation.

Non-Area-Based Strategies. Intensity-based approaches that are combined with non-area-based strategies can lead to less computational effort and higher robustness. The principle is that not all pixels within the model image and the search image are used to calculate the intensity based similarity measure. Instead, only a small selected fraction is used. For example, in (Edwards and Murase 1998), the region of overlap between modeled objects is determined and excluded from the calculation to increase the robustness against occlusions. However, an expensive iterative processing is needed since the region of overlap is not known à priori. Furthermore, occlusions that occur because of missing object parts or because of an overlap with non-modeled objects cannot be taken into account. In (Li et al. 1997), the computation of the cross correlation is restricted to only a few pixels dependent on the image contrast in these pixels. This not only decreases the computation time but also increases the robustness against occlusions and clutter. However, applying the FFT for efficiently computing the correlation is no longer possible.

4.1.1.2 Approaches Using Low Level Features

There are two main motivations for using features in object recognition approaches, which can be directly concluded from the weak points of most intensity-based approaches. The first motivation arises from the relatively high sensitivity to occlusions and clutter of most intensity-based approaches. Several features can be found that enable higher robustness against such variations. The second motivation can be attributed to the fact that features combined with non-area-based strategies are able to represent an object in a more compressed, and hence efficient form. Recognition approaches can take advantage of this property in order to achieve less computational effort.

When regarding alignment methods, a difference in the approaches that use features and approaches that use intensity information should be mentioned. In feature-based approaches, there are, in general, two ways for an alignment method to obtain the features of the transformed model image using the discrete transformations $\mathcal{T}_i$. The first possibility is to transform the model image and to calculate the features in the transformed model image. In the second possibility, features are only calculated in the untransformed model image and merely the features themselves are transformed. While the second method is faster, it usually suffers from quantization effects or unpleasant properties of the feature extractor (e.g, inaccuracy or non-isotropy).

Area-Based Strategies. A first class of feature-based alignment methods concentrates on gray value statistics. For example, gray value histograms (Ballard and Brown 1982) that are derived from images can be interpreted as features that are used to compute the similarity between the images, since the raw gray value information is not directly used within the similarity measure. A similarity measure that is based on the difference between two histograms is invariant to rotations. Histograms of angular measurements are used in (Cha and Srihari 2000) to recognize handwritten characters. In (Bhat and Nayar 1998), a similarity measure is proposed that uses an ordinal measure as feature, where images are represented by their gray value ranks. For illustration purposes, the gray values of an image I are written sequentially, e.g., $I = (5, 10, 50, 40)$. Then the gray values are sorted and the image is represented by its rankings $\pi = (1, 2, 4, 3)$. This representation can be used in alignment methods, where, not the gray values themselves, but the rankings of the model image and the respective part of the search image are compared by applying a rank correlation coefficient (Gideon and Hollister 1987). Similarity measures that are based on gray value rankings have the advantage that they are invariant to changes in brightness that do not affect the ranking such as linear changes. Another advantage is that the ranking is less sensitive to outliers, which leads to a higher robustness against occlusions and clutter in comparison to the normalized cross correlation or the sum of gray value differences. However, in general, the mapping of an image to its gray value statistics is not an injective function: for example, when using histograms, the spatial arrangement of the pixels is lost. This may lead to a large number of false positive matches, i.e., a high similarity value between obviously dissimilar images, especially in the case of image clutter.

In (Kaneko et al. 2002), a similarity measure for alignment methods is proposed that is based on the *increment sign correlation*. In the first step, the two images to be compared are encoded into corresponding binary images. For this, a procedure is applied that maps local gray value changes into 1 if the gray value of a neighboring pixel (e.g., the right neighbor) is higher than the gray value of the current pixel, and into 0 otherwise. This increment sign is used as the feature to represent the images. In the second step, a binary correlation coefficient is computed based on the binary representations of both images. It is shown that this measure is robust against occlusions and changes in brightness. The disadvantage of this method is the high reduction of information from (in general) 8 bit to 1 bit. This results in an increased number of false positive matches, especially in the case of small model images, where the discriminative power of the increment sign correlation is poor. Furthermore, in the case of model images showing regions of approximately constant gray values, the increment sign is more or less random. This reduces the correlation coefficient even if the images are similar.

Another class of approaches performs an intensity-based similarity measure not directly on the raw gray values but on derivatives of the gray values. For example, in (Martin and Crowley 1995), the cross correlation and the sum of gray value differences are computed on the first derivative magnitude (gradient magnitude) and the sum of second derivatives (Laplacian). It is shown that the decision whether one should use the raw gray values, the gradient magnitude, or the Laplacian for applying the cross correlation depends on the requirements of the task. In the frequency domain, an ideal first derivative grows linearly with increasing frequency, while a second derivative grows quadratically. Therefore, a correlation of first derivatives has a more precise peak than a correlation of raw intensity images, but is more sensitive to high frequency noise. A second derivative doubles the effect. Experiments in (Martin and Crowley 1995) showed that using the gradient magnitude usually provides more stable results in comparison to the use of raw intensity values or of the Laplacian. The approaches presented in (Scharstein 1994, Crouzil et al. 1996, Fitsch et al. 2002) extend the principle of these ideas by using the gradient direction as feature. For example, in (Fitsch et al. 2002), angles between gradient directions are used as similarity measure. This results in invariance to changes in brightness, since, the gradient directions are unaffected by changing brightness. It also shows robustness against occlusions and clutter. In this approach, however, the object representation by the chosen feature is not very efficient since for each pixel in the model image the orientation is computed and used in the similarity measure. Thus, the number of features is equal to the number of pixels. Therefore, there is no real improvement regarding the computation time in comparison to the intensity-based approaches.

Non-Area-Based Strategies. The number of features that are involved in the matching process of non-area-based strategies is less than the number of pixels. Several classes of feature-based object recognition methods that are reviewed in this section use the object's edges as geometric feature, which will also be referred to as the object *shape* in the following discussions. A review of edge extraction algorithms is not given here. Instead, the reader should refer to standard text books (Ballard and Brown 1982, Gonzalez and Woods 1992, Haberäcker 1995, Jähne 2002). Usually, the edge pixels are defined as pixels in the image where the magnitude of the gradient is maximum in direction of the gradient. In most cases, edges are extracted in two steps. At first, the image is convolved using an edge filter that provides the first partial derivatives of the gray values in row and column direction. An edge filter responds to gray value changes in the image by taking the neighboring gray values into account, e.g., the *Roberts* (Gonzalez and Woods 1992), *Sobel* (Gonzalez and Woods 1992), *Canny* (Canny 1983, Canny 1986), *Deriche* (Deriche 1987), or *Lanser* (Lanser and Eckstein 1992) filters. The edge magnitude γ can be computed from the first partial derivatives when using a gradient-based edge detection:

$$\gamma = \sqrt{\left(\frac{\partial I(x,y)}{\partial x}\right)^2 + \left(\frac{\partial I(x,y)}{\partial y}\right)^2} \; . \tag{4.4}$$

In the second step, a threshold is applied on the edge magnitude, which is a measure of the contrast in the image. This results in the segmented edge regions of the image.

Other recognition methods use points as geometric features. They can also be extracted from an image in various ways, e.g., using so-called *interest operators*. For a comprehensive overview and evaluation of different interest operators the interested reader should refer to (Heipke 1995, Schmid et al. 2000).

The first class of non-area-based strategies does not use the shape of the object, but is based on global image transforms. Both the model image and the search image are transformed into the frequency domain, e.g., using a wavelet-based transformation (Bronstein et al. 2001, Wang 2001): the original images can be represented as a linear combination of the respective wavelet functions. By truncating the wavelet coefficients below a certain magnitude, image data can be sparsely represented. However, a loss of detail must be expected. A set of such coefficients can be used as feature vector for object recognition. Approaches that use wavelet techniques can be found in (Jacobs et al. 1995) and (Wang et al. 1995), for example. The major drawback of these methods is that because of their global nature, it is difficult to compare the model image to only a part of the search image. Consequently, robustness against occlusions or clutter is hard to achieve when using global image transforms.

The second class of approaches works on an object as a whole, i.e., on a complete object area or shape. Therefore, these methods are often called global object methods. The use of geometric moments is a very popular representative of this class (Teh and Chin 1988). Geometric moments are used in several object recognition applications as features, e.g., (Liao and Pawlak 1996). By combining moments of different orders, one can find features that are invariant to rotation, scaling, or other transformations of the objects. Some examples are area, circularity, eccentricity, compactness, major axis orientation, Euler number (Veltkamp and Hagedorn 1999). These invariant features can be computed in the model image as well as in the search image and can be represented in a feature vector. The feature vector of both images can then be used to compute the similarity between both images using an appropriate distance measure. The main advantage of object recognition based on moments is that the class of transformations $\mathcal{T}$ can be reduced by transformations that are covered by the invariants of the selected moments themselves. Thus, the computational effort can be reduced considerably. Unfortunately, the computation of the moments itself is very time consuming in general. This often annihilates the advantage of the reduced parameter space. A closely related method uses the principle component analysis, which decomposes the object shape into an ordered set of eigenvectors (also called eigenshapes). The eigenvectors of the object in the model image and in the search image can be used to compute a similarity measure to recognize the object. Finally, in another global object method, the shape of the object is represented by its contour parameterized by the arc-length (Mokhtarian et al. 1996). The contour is successively smoothed using a Gaussian kernel. The characteristic behavior of the contour while applying the smoothing is exploited and used as an object-specific feature. This feature is invariant to orientation and moderate scale changes of the object and robust to noise. However, an important drawback of all global object methods is that the complete object to be found in the search image must be clearly segmented, which is in itself an ill-posed problem. Consequently, most of these methods fail in the case of occlusions and clutter.

Another class of approaches performs object recognition using alignment methods. The most elementary similarity measure that can be applied to alignment methods based on image edges is binary correlation. Here, the intersection of edge pixels in the transformed model image and the search image is a measure of similarity. The advantage of a simple computation is overshadowed by a high sensitivity to small edge displacements. I.e., a high similarity measure is only obtained if the edges of model and search image are almost perfectly identical. A method that relaxes this stringent requirement, and hence is less sensitive to small edge displacements, is presented in (Borgefors 1988). The algorithm matches points in the transformed model image and the search image by minimizing a generalized distance between them. Although the algorithm is designed to cope with arbitrary binary images, in most cases edges are used as features. The result of the edge extraction are two sets of points $\mathcal{P}^m \ni \boldsymbol{p}_i^m, i = 1, \ldots, n^m$ and $\mathcal{P}^s \ni \boldsymbol{p}_j^s, j = 1, \ldots, n^s$, representing the edge pixels in the transformed model image (*model edges*) and the search image (*search edges*), respectively, where n^m and n^s are the number of edge pixels in the corresponding images. The average distance between the two sets of pixels $\mathcal{P}^m$ and $\mathcal{P}^s$ is then used as similarity measure to find the pose of the object in the image. Unfortunately, the distance computation between two point sets is computationally expensive. Therefore, in (Borgefors 1988) a more efficient solution is applied that exploits the distance transform (Jähne 2002) for the matching: in the search image, each non-edge pixel is assigned a value that is a measure of the distance to the nearest edge pixel. The edge pixels have a value of zero. Fast algorithms using iterative local operations are available for computing the distance transform (Soille 1999). Since the true Euclidean distance is expensive to compute, it is approximated by using integer values (Danielsson 1980, Borgefors 1984). Assuming that the horizontal and vertical distance of neighboring pixels is d and the diagonal distance is d^d, then integer values for d and d^d are chosen appropriately in order to approximate the Euclidean distance, i.e., $d^d/d \approx \sqrt{2}$. Well known combinations are, for instance, $d = 1$, $d^d = 2$ (city block distance), $d = 1$, $d^d = 1$ (chess board distance), and $d = 3$, $d^d = 4$ (chamfer distance). To compute the average distance between $\mathcal{P}^m$ and $\mathcal{P}^s$, the edge pixels $\mathcal{P}^m$ of the transformed model image are superimposed on the distance-transformed search image and the distance values of the pixels in the distance image that are hit by $\mathcal{P}^m$ are added. In (Borgefors 1988), the distance measure B between two sets of points is computed using the chamfer distance and the root mean square average:

$$B(\mathcal{P}^m, \mathcal{P}^s) = \frac{1}{3}\sqrt{\frac{1}{n^m}\sum_{i=1}^{n^m} v_i^2} \quad , \tag{4.5}$$

where $v_i = \min_{\boldsymbol{p}_j^s \in \mathcal{P}^s} \|\boldsymbol{p}_i^m - \boldsymbol{p}_j^s\|$ are the distance values hit by the model edges, and $\|\cdot\|$ is the underlying norm of the chosen distance metric. To compensate the unit distance of 3 in the chamfer distance, the average is divided by 3. For speed reasons, the implementation of (Borgefors 1988) uses a hierarchical structure by applying image pyramids. The principle of image pyramids will be explained in Section 4.1.2. There are some major drawbacks, which are inherently connected with this similarity measure. It is not a symmetric measure. I.e., a different similarity value is obtained depending on which point set is used to compute the distance transform and which point set is superimposed on the distance-transformed image. If the distance transform is computed on the search image, then the distance measure is not robust against partial occlusions. The reason for this is that some missing edge pixels in the search image cause the corresponding edge pixels of the model to get a high distance value. This increases the root mean square average in a non-proportional way. Additionally, the distance measure is not sensitive to even severe clutter, which would be desirable. E.g., if all pixels in the search image would represent edge pixels, then the distances of all model edge pixels would be zero. Assume now that the distance transform is computed on the model image. Then, on the one hand, the distance measure is not robust against moderate clutter. On the other hand, it is not sensitive to even severe occlusions, which would also be desirable when considering the case that no edge pixels are present in the search image. Concluding, a good distance measure (similarity measure) should be, on the one hand, sensitive to occlusions and clutter, i.e., the distance measure (similarity measure) should increase (decrease) when occlusions and clutter increases. On the other hand, it should be robust against occlusion and clutter, i.e., the distance measure (similarity measure) should not increase (decrease) in a non-proportional way. The *Hausdorff distance* proposed in (Huttenlocher et al. 1993) and (Rucklidge 1997) tries to remedy the above mentioned shortcomings. In (Huttenlocher et al. 1993) the Hausdorff distance H is defined as

$$H(\mathcal{P}^m, \mathcal{P}^s) = \max(h(\mathcal{P}^m, \mathcal{P}^s), h(\mathcal{P}^s, \mathcal{P}^m)) \ , \tag{4.6}$$

where

$$h(\mathcal{P}^m, \mathcal{P}^s) = \max_{\boldsymbol{p}_i^m \in \mathcal{P}^m} \min_{\boldsymbol{p}_j^s \in \mathcal{P}^s} \|\boldsymbol{p}_i^m - \boldsymbol{p}_j^s\| \tag{4.7}$$

and $\min_{\boldsymbol{p}_j^s \in \mathcal{P}^s} \|\boldsymbol{p}_i^m - \boldsymbol{p}_j^s\|$ again can be efficiently obtained by computing the distance transform on $\mathcal{P}^s$. The function $h(\mathcal{P}^m, \mathcal{P}^s)$ is called the *directed* Hausdorff distance from $\mathcal{P}^m$ to $\mathcal{P}^s$. It identifies the point $\boldsymbol{p}_i^m \in \mathcal{P}^m$ that is farthest from any point of $\mathcal{P}^s$ and measures the distance from $\boldsymbol{p}_i^m$ to its nearest neighbor in $\mathcal{P}^s$. By computing the directed Hausdorff distance in both directions and taking the maximum of both, the Hausdorff distance is a symmetric measure. Furthermore, it is sensitive to both occlusions and clutter in the search image because one of both directed Hausdorff distances is affected and the maximum of both is taken. However, since the maximum of all edge distances is taken in (4.7) it still shows no robustness against occlusions and clutter. Therefore, in (Rucklidge 1997) the *partial* directed Hausdorff distance is proposed:

$$h^f(\mathcal{P}^m, \mathcal{P}^s) = \underset{\boldsymbol{p}_i^m \in \mathcal{P}^m}{f\text{th}} \min_{\boldsymbol{p}_j^s \in \mathcal{P}^s} \|\boldsymbol{p}_i^m - \boldsymbol{p}_j^s\| \ , \tag{4.8}$$

where $f\text{th}_{x \in \mathcal{X}}\, g(x)$ denotes the f-th quantile value of $g(x)$ over the set $\mathcal{X}$, for values of f between zero and one. Hence, f denotes the fraction of points that are used to compute the partial directed Hausdorff distance. For example, the 1-th quantile value is the maximum and the 1/2-th quantile value is the median. Thus, when $f = 1$, the partial directed Hausdorff distance corresponds to the unmodified directed Hausdorff distance. Consequently, the partial undirected Hausdorff distance is defined as

$$H^{f_F f_R}(\mathcal{P}^m, \mathcal{P}^s) = \max(h^{f_F}(\mathcal{P}^m, \mathcal{P}^s), h^{f_R}(\mathcal{P}^s, \mathcal{P}^m)) \ . \tag{4.9}$$

Here, f_F and f_R are the *forward fraction* and *reverse fraction*, respectively, and define the fractions for the evaluation of the directed distances. This measure is robust against $100(1 - f_F)$% occlusions and $100(1 - f_R)$% clutter in the image.

The Hausdorff distance has undergone several further improvements and extensions, including, for example, sophisticated search strategies, computational shortcuts, and enhanced robustness (Huttenlocher et al. 1993, Olson and Huttenlocher 1996, Paumard 1997, Rucklidge 1997, Huttenlocher et al. 1999, Kwon et al. 2001, Sim and Park 2001). To enhance the robustness against clutter, attempts have been made to also include the angle difference between the model edges and the search edges into the Hausdorff distance (Olson and Huttenlocher 1995, Olson and Huttenlocher 1996, Olson and Huttenlocher 1997, Sim and Park 2001). Unfortunately, the computation is based on several distance transforms, and hence is too computationally expensive for real-time object recognition.

Another class of feature-based object recognition methods are summarized under the term *voting schemes*. One of the most important representatives of this class is *pose clustering*, also known as the *generalized Hough transform* (GHT) (Ballard 1981), which uses the edge position and direction as features. Its principle is based on the well-known *Hough transform* (Hough 1962), which is a voting scheme to detect analytical curves in images. Comprehensive surveys of different Hough transform techniques are given in (Illingworth and Kittler 1988) and (Leavers 1993). Because an analytical description of objects is not always available, or not even possible, the conventional Hough transform is only of limited use for object recognition. Therefore, Ballard (1981) generalizes the Hough transform to detect arbitrary shapes. Here, the parameters that describe the analytical curve in the classical Hough transform are replaced by parameters that define the class of allowed transformations $\mathcal{T}$. By taking the edge direction into account, not only the number of false positives is reduced, but also a speed-up is obtained. Strictly speaking, the gradient direction is computed instead of the (tangential) edge direction. Let θ_i^m and θ_j^s be the associated gradient directions at the model edge point $\boldsymbol{p}_i^m$ and the search edges point $\boldsymbol{p}_j^s$, respectively. Similar to the edge magnitude (4.4) the gradient direction of the edge can be computed, for example, from the partial derivatives obtained from an arbitrary gradient-based edge filter:

$$\theta = \arctan \frac{\partial I(x,y)/\partial y}{\partial I(x,y)/\partial x} \quad . \tag{4.10}$$

To perform the GHT, in the offline phase a look-up table (*R*-table) is computed by using information about the edge positions and the corresponding gradient directions in the model image. The *R*-table is generated as follows: At first, an arbitrary reference point $\boldsymbol{o}^m$, e.g., the centroid of all model edge points, is chosen. Then the *displacement vectors* $\boldsymbol{r}_i = \boldsymbol{o}^m - \boldsymbol{p}_i^m$ are calculated for all edge points $\boldsymbol{p}_i^m, i = 1, \ldots, n^m$. Finally, $\boldsymbol{r}_i$ is stored as a function of θ_i^m in the *R*-table. Informally speaking, the *R*-table contains the position of all edge points in the model image with respect to a reference point sorted by their corresponding gradient direction.

For the online phase a two dimensional accumulator array A is set up over the domain of translations. Thus, A represents the sampled parameter space of $\mathcal{T}$. In general, each cell of this array corresponds to a certain pixel position of the reference point $\boldsymbol{o}^s$ in the search image. For each edge pixel $\boldsymbol{p}_j^s$ in the search image the displacement vectors $\boldsymbol{r}_i$ that are stored under the corresponding gradient direction $\theta_i^m = \theta_j^s$ are selected from the *R*-table. For the selected vectors, the cells $\boldsymbol{p}_j^s + \boldsymbol{r}_i$ in A receive a vote, i.e., they are incremented by 1. Thus, at each edge pixel in the search image all possible candidates for the reference point are calculated. This is repeated for all edge pixels in the search image. Finally, each cell in A has a value that specifies the likelihood that the reference point is located in the cell. Thus, local maxima in A that exceed a certain threshold represent object instances found in the search image.

The advantage of the GHT is the high robustness against occlusions, clutter, and against changes in brightness of an arbitrary type. The GHT is more efficient in comparison to conventional alignment methods because it does not explicitly compute all translations of the model image to test the similarity. In contrast, it restricts the search to the relevant information in the search image, i.e., the edge information. This is achieved by using the relative position of the model edges with respect to the reference point (i.e., the displacement vectors) as translation invariant feature. Thus, the transformation class $\mathcal{T}$ of the alignment method can be reduced by the sub-class of translations. For example, if the class of rigid transformations is chosen then the model image needs to be only rotated to all possible object orientations. Nevertheless, in the conventional form the GHT requires large amounts of memory for the accumulator array and long computation times to recognize the object if more general transformation classes than translations are considered.

Many algorithms have been suggested to reduce the computational load associated with the GHT. Davis (1982) proposes a hierarchical Hough transform in which sub-patterns, i.e., simple geometric objects, such as line segments, instead of edge points are used as basic units. In a similar way, a local classification of the instances of detected contours is performed in (Cantoni and Carrioli 1987). The implementations of these approaches are complicated since local classifications of sub-patterns are required. It is also difficult to find the desired sub-patterns in a search image accurately, especially in the presence of noise. In (Ballard and Sabbah 1983), a two-level approach is proposed that takes similarity transformations into account, in which the factors of scaling and rotation are estimated first from the lengths and directions of the line segments in the search image before the GHT is applied. However, accurate extraction of line segment data from the image is a difficult task. A fast GHT is described in (Jeng and Tsai 1990), where the basic GHT is performed on a sub-sampled version of the original image and a subsequent inverse GHT operation is used to finally determine the pose in the original image. In this approach the edge direction is ignored during the inverse GHT. Thus the robustness against clutter is reduced. The method proposed in (Thomas 1992) uses displacement vectors relative to the gradient direction to achieve orientation invariance. Hence, besides the invariance of the conventional GHT to translations, invariance to rigid motion is obtained because one more degree of freedom within $\mathcal{T}$ is eliminated. This method is extended to scale invariance in (Kassim et al. 1999) without adding an extra dimension in parameter space. This is obtained by incrementing a line of cells in the accumulator array that correspond to a range of defined scales instead of incrementing a single cell. In (Lo and Tsai 1997), even a perspective transformation invariant GHT is proposed using only a 2D parameter space. However, the solution is connected with long computation times for the recognition of perspectively distorted planar objects. This prohibits a use for real-time applications. Furthermore, when using one of these methods, which are based on the projection of the parameter space to fewer dimensions, information about the projected dimensions, e.g., orientation and scale of the object, is lost and must be reconstructed using a subsequent computation step. In (Ma and Chen 1988), analytical features that consist of local curvature and slope are used to reduce the 4D parameter space of similarity transformations to two 2D parameter spaces. This approach reduces the computational complexity but also has some limitations. The memory requirement is as high as in the case of the conventional GHT, the accuracy of the curvature estimator and the gradient operator can adversely affect the performance, additional computations in the image space are required, and the algorithm fails for shapes that are composed mainly of straight lines (i.e., zero curvature).

Another voting scheme that is also often applied is *geometric hashing* (Wolfson 1990, Cohen and Guibas 1997). Here, the object is represented as a set of geometric features, such as edges or points, and their geometric relations are encoded using minimal sets of such features under the allowed transformation class. The geometric hashing method described in (Wolfson 1990) is illustrated for a 2D object using affine transformation as transformation class. The object is described by interest points, which are invariant under affine transformation, e.g., corners, points of sharp concavities and convexities, or points of inflection. Thus, two sets of interest points are obtained, one in the (untransformed) model image (*model points*) and one in the search image (*search points*). In the offline phase, a model description is constructed from the model points by choosing any triplet of non-collinear points $\boldsymbol{e}_{00}$, $\boldsymbol{e}_{10}$, and $\boldsymbol{e}_{01}$. The point triplet defines an affine basis, into which all other model points can be transformed by representing each model point $\boldsymbol{p}_i^m$ as a linear combination of the affine basis:

$$\boldsymbol{p}_i^m = \alpha_i(\boldsymbol{e}_{10} - \boldsymbol{e}_{00}) + \beta_i(\boldsymbol{e}_{01} - \boldsymbol{e}_{00}) + \boldsymbol{e}_{00} \quad . \tag{4.11}$$

The obtained coordinate pair (α_i, β_i) is invariant under affine transformations. The (α, β)-plane is quantized into a two-dimensional hash-table and the chosen point triplet is recorded in all cells of the table that are associated with (α_i, β_i). To achieve robustness against occlusions, the calculation must be repeated for several different affine basis triplets while using the same hash-table. In the online phase, an arbitrary triplet of non-collinear search points $(\boldsymbol{e}'_{00}, \boldsymbol{e}'_{10}, \boldsymbol{e}'_{01})$ is chosen and used to express the other search points as linear combination of this affine basis. Thus, for each search point $\boldsymbol{p}_j^s$ a coordinate pair (α_j, β_j) is obtained. Each triplet $(\boldsymbol{e}_{00}, \boldsymbol{e}_{10}, \boldsymbol{e}_{01})$ in the cell of the hash-table that is associated with (α_j, β_j) receives a vote. The affine transformation that maps the triplet $(\boldsymbol{e}'_{00}, \boldsymbol{e}'_{10}, \boldsymbol{e}'_{01})$ to the triplet $(\boldsymbol{e}_{00}, \boldsymbol{e}_{10}, \boldsymbol{e}_{01})$ that received the most votes is assumed to be the transformation between the model points and the search points. The advantage of such methods is that several different objects can be searched simultaneously without affecting the computation time of the online phase. However, in the case of clutter there is a high probability for choosing a point triplet in the search image that is not represented in the model image. Thus, to achieve a higher robustness, also in the online phase several point triplet must be selected subsequently. This increases the computational effort.

Another approach that is closely related to geometric hashing but is not a real voting scheme is described in (Hartley and Zisserman 2000). It is based on a robust estimator RANSAC (RANdom SAmple Consensus). Continuing the example of affine transformations, a sample point triplet is selected randomly in the model image and in the search image and the affine transformation is computed between these two triplets. The support for this candidate transformation is measured by the number of points in the search image that lie within some distance threshold to the transformed model points. This random selection is repeated a number of times and the transformation with most support is deemed the searched transformation between the model and the search points. Hartley and Zisserman (2000) showed that RANSAC can cope with a high rate of outliers, e.g., clutter in the search image, even for a relatively small number of randomly selected samples.

4.1.1.3 Approaches Using High Level Features

A prominent class of object recognition methods represents an object not only by isolated features but also takes the relations between the features into account. The relations between features can be seen as self-contained high level features. In (Li 1999), for example, invariants under a certain class of transformations serve as features and relations. The approach further distinguishes between invariants of order one, (representing a single feature, i.e., an invariant property or invariant unary relation), invariants of order two (representing relations between two features, i.e., an invariant binary relation), invariants of order three (representing relations between three features, i.e., an invariant ternary relation), and so on. For each class of transformations, invariant relations of different order can be found. Under rigid motion, the curvature (e.g., of a curve or a surface) is an invariant unary relation, whereas the distance between two points is an invariant binary relation. Under similarity transformations, the length ratio between two lines or the angle between two lines are preserved and are therefore invariant binary relations, whereas the three angles in a triangle constitute an invariant ternary relation. The cross-ratio for four points on a line is an invariant quaternary relation that is preserved under perspective transformation, and so on. This information can be represented in a graph, where the nodes represent the unary relations and the edges between the nodes represent the relations of higher order. Thus, the object recognition problem is transformed into the problem of determining the similarity of graphs, which is also known as *graph matching*. In (Bunke 2000), it is shown that graphs are a versatile and flexible representation formalism suitable for a wide range of object recognition tasks. Unfortunately, the complexity of graph matching is NP-complete, i.e., it cannot be solved in polynomial time. Therefore, several algorithms have been developed that try to minimize the computational effort by finding either optimal solutions, which in the worst-case take exponential time (Ullmann 1976, Messmer and Bunke 1998), or approximate solutions, which are not guaranteed to find the optimum solution (Christmas et al. 1995). To give an example application, in (Kroupnova and Korsten 1997) an algorithm is proposed to recognize electronic components on printed circuit boards using graph matching, where the nodes in the respective graph represent region attributes (color, shape) and the edges in the graph represent spatial relations between the regions (adjacent to, surrounds). However, because graph matching algorithms either are extremely slow or fail to find the optimum solution, they are not really well suited for a robust real-time object recognition system.

Some of the approaches that use high level features can be interpreted as a natural extension of the approaches that deal with low level features as introduced in the previous section. Geometric hashing, for instance, can be extended to cope with higher level features like lines, for example. In (Procter and Illingworth 1997), 3D objects are recognized using edge-triple features. 3D polyhedral objects can be decomposed into triples of connected straight edges, which are projected as three straight, connected lines into the image. The two angles formed by these three consecutive lines are invariant under 2D similarity transformations. Images of the object are taken from different viewing angles to build the hash table. For each edge-triple the two angles are used to index the hash table. In the corresponding cell of the hash table the current viewing angle of the model is stored. In the search image connected line triples are extracted and used to vote for the viewing angles stored in the associated cell of the hash table. The viewing angle that receives most votes can be used to compute the transformation of the object in the search image. By using high level features, namely edge-triples, instead of points, objects can be represented by a smaller number of more meaningful structures. This reduces the computational effort of the voting process and also the sensitivity to noise. However, the preprocessing, i.e., the extraction of the lines, is computationally more expensive than the use of low level features. Furthermore, the method is restricted to polyhedral or partially polyhedral objects.

To climb the next rung on the ladder of feature hierarchy, in (Vosselman and Tangelder 2000) complete 3D CAD models are used to describe the object. An approach is designed to recognize parts of industrial installations like straight pipes, curved pipes, T-junctions, and boxes. CAD models of these parts can be composed from simpler CAD model primitives like cylinders, boxes, cones, and spheres. The CAD models are projected into the image using a hidden line algorithm and fitted to the extracted edges in the search image using a constrained least-squares adjustment, resulting in accurate pose parameters. However, the requirement for the existence of a CAD model contradicts the requirement for an easy model generation.

4.1.2 Methods for Pose Refinement

For many applications it is insufficient to determine the pose of an object with an accuracy that is limited to the chosen quantization of the transformation class. Therefore, several methods have been developed to refine the discrete pose parameters of alignment methods in a subsequent step. Often the pose refinement is restricted to the transformation class of translations. In general, the sampling of translations is done according to the pixel grid, and hence methods to refine the translation parameters are referred to as *subpixel* refinement methods. In (Tian and Huhns 1986), four different subpixel refinement methods are distinguished: correlation interpolation, intensity interpolation, differential methods, and phase correlation.

In correlation interpolation the similarity measures at the sampled pixel positions are used to locally fit an interpolation surface. Often a second-order interpolation function can provide an accurate representation (Tian and Huhns 1986). In the refinement step the maximum of the surface is analytically computed, yielding a subpixel precise object position.

Intensity interpolation locally adapts the parameter quantization. If a position accuracy of 0.1 pixel is desired, then the model image is successively translated by 0.1 pixel steps in the neighborhood of a found match. Because of the subpixel translation, the gray values of the translated model image must be computed by re-sampling the original gray values.

The idea behind differential methods can be also interpreted as exploiting the well-known *optical flow* (cf., e.g., (Jähne 2002)) for the use of object recognition. The optical flow was originally applied in the analysis of image sequences in order to detect object motion. It is based on the *brightness constancy assumption*

$$I_0(x+\Delta x, y+\Delta y) = I_1(x,y) \ , \tag{4.12}$$

where I_0 and I_1 are two images taken at time t_0 and t_1 and $(\Delta x, \Delta y)^\top$ is the displacement vector, which describes the object movement during the time $t_1 - t_0$ at a location $(x,y)^\top$. This equation assumes that the gray values in image I_1 are identical to those in I_0 but occur at a different position in the image as it happens when projecting object movement into the image plane. Transferring this relationship to object recognition, I_0 corresponds to I^m and I_1 to I^s, respectively, and the displacement vector represents the desired pose parameters. In order to incorporate changes in brightness, a generalized brightness model is used in (Szeliski and Coughlan 1997): $I_0(x+\Delta x, y+\Delta y) = aI_1(x,y) + b$, where a and b model the global linear change in brightness. However, this model cannot account for spatially varying brightness variations. To overcome this restriction, a dynamic model is applied in (Negahdaripour 1998), which describes the parameters $a(x,y)$ and $b(x,y)$ as a function of location. This model is, for example, applied in (Lai and Fang 1999a, Lai and Fang 1999b) for object recognition, using affine transformations as transformation class. This is achieved by writing the displacement vector $(\Delta x, \Delta y)^\top$ as a function of the affine transformation parameters and substituting this, together with $a(x,y)$ and $b(x,y)$, into (4.12). The transformation parameters, and hence the object pose, can then be accurately calculated by using a robust estimation. Good starting values for the parameters are needed, because there is a non-linear relationship between the observed gray values in both images and the unknown transformation parameters. This requirement is less problematic when dealing with image sequences where the object pose in two consecutive image frames changes only slightly. However, it is of more importance in object recognition, because in general no such prior information is available. Therefore, optical flow is an ideal candidate for pose refinement when starting values for the parameters are obtained from any other pixel precise recognition method.

Phase correlation can be used to detect subpixel shifts with a higher accuracy than it is possible with the original normalized cross correlation when using correlation interpolation methods. It exploits the fact that a shift in the spatial domain is transformed in the Fourier domain into linear phase differences (cf., e.g., (Stöcker 1995)). In (Foroosh et al. 2002), a more sophisticated method is proposed that is based on the idea that in down-sampled images phase correlation does not concentrate in a single peak but rather in several coherent peaks mostly adjacent to each other and centered at the (subpixel) object position. The position and the magnitude of the peaks can then be used to determine the exact object pose.

Another class of refinement approaches tries to minimize the geometrical distance between the features in the model and the search image by using robust parameter estimation (Wallack and Manocha 1998). As for the above described minimization of gray value differences (Lai and Fang 1999c) also here starting parameters are needed, which can be obtained by a preceding pixel precise recognition method. The advantage of minimizing the geometrical distance between features is the inherent robustness against changes in brightness. Thus, it is dispensable to model the changes in brightness as required in approaches that use intensity information, as in (Szeliski and Coughlan 1997), for example.

4.1.3 General Methods for Speed-Up

Some general improvements that can be used to speed up object recognition approaches and that are not restricted to a special approach are introduced in the following.

In alignment methods, for example, most similarity measures can be computed using a recursive implementation when dealing with rectangular model image domains D^m, which makes the computational complexity independent of the size of the model image.

Another way to speed up the computation is to introduce stopping criteria (Barnea and Silverman 1972): the computation of the similarity measure for a certain model image transformation can be immediately stopped as soon as it is certain that the predefined threshold for the similarity measure cannot be reached for this transformation. In this way, the computational cost is decreased considerably.

Reducing the number of discrete model image transformations is also often desirable, especially when dealing with transformation classes that are more general than translations. Therefore, it can be reasonable to combine similar model transformations into one common representative transformation. The similarity measure is then only computed for the representative instead of for all transformations separately. The combination of similar transformed model images can be achieved, e.g., applying the Karhunen-Loeve decomposition proposed in (Uenohara and Kanade 1997), or any other principal component analysis.

Furthermore, the search for objects is usually done in a coarse-to-fine manner, e.g., by using image pyramids. Often objects can be more easily recognized in images that have a very low sampling rate. There are two main reasons for this (Ballard and Brown 1982). Firstly, the computations are fewer because of the reduced image size. Secondly, confusing details, which are present in the high resolution version of the image, may disappear at the reduced resolution. However, to be able to accurately determine the pose of an object, detailed image information is required, which is only revealed at the higher resolutions. This naturally leads to the use of image pyramids in object recognition, where the search for objects is started at a low resolution with small image size, and refined at increasing resolutions until the highest resolution at the original image size is reached. The basic idea to compute an image pyramid is based on the Nyquist theorem. This theorem states that the highest frequency that can be represented in a sampled version of a signal is less than one-half of the sampling rate. Thus, in the original image only frequencies below 1/2 [1/pixel] are represented. In a first step the original image is smoothed using a low pass filter that eliminates frequencies exceeding 1/4 [1/pixel], e.g., by applying a Gaussian filter kernel (Jähne 2002). In a second step the smoothed image can be sub-sampled using a sampling interval of 2 pixels, without violating the Nyquist theorem, i.e., without created aliasing artifacts. These two steps are repeated iteratively, which results in a series of images with an image area successively reduced to one fourth. This series is called a *Gaussian Pyramid* (Jähne 2002) and can be seen as a multi-scale representation of the original image (Lindeberg 1994). As a matter of course, also other smoothing filters can be applied. Because in real-time object recognition the computation time plays a decisive role, often the mean filter is used instead of the Gaussian filter because it is less computationally expensive and serves as a sufficiently accurate approximation to the Gaussian filter. The few artifacts coming from the deviation of the mean filter from the optimum low-pass filter are accepted for the gain in speed.

In (Tanimoto 1981), some possible techniques are introduced to apply image pyramids to alignment methods. In general the object is searched in an image pyramid using *recursive descent*: the search is started in a coarse (high) level, and continued in a local area at the next finer (lower) level where the similarity measure in the coarse level is promising. Depending on the application, the pyramid search strategy can be to obtain either the first match where a certain threshold for the similarity measure is exceeded, the best match with globally maximum similarity, or all matches that exceed the threshold. Furthermore, the pyramid search strategy can make use of standard search techniques like depth-first search, breadth-first search, or forward-pruned search. Finally, in (Tanimoto 1981) *descent policies* and *sensitivity policies* are distinguished. A descent policy specifies which descendants of a pixel are to be tracked down the pyramid. Since one pixel on a coarse resolution corresponds to four pixels on the next finer resolution one can track, for example, all four pixels. However, this would lead to an exploding number of matches to track. Another more efficient possibility would be to only track the pixel that yields the best match of the four pixels while disregarding the others. To increase the probability of locating the sought object location correctly, the similarity measure is additionally computed for pixels that are adjacent to the four directly descending pixels. Sensitivity policies specify the threshold for the similarity measure at each pyramid level. The threshold can be chosen to be constant for all levels, adaptive, regarding the similarity of the matches on the previous higher level, or as a function of the pyramid level. The last strategy requires the balancing of conflicting requirements. On the one hand, it seems desirable that thresholds increase as the search goes deeper, because it can be expected that objects become sharper and return higher similarity values. On the other hand, when image noise is present, zooming in on objects reveals high-frequency noise, which lowers similarity values. Thus, the threshold should decrease at the fine pyramid levels.

4.1.4 Conclusions

It becomes obvious that the power of object recognition approaches that are based on intensity information lies in their simple and straightforward implementation as well as in their robustness against certain intensity transformations. Furthermore, no feature correspondence problem arises as in the case of several feature-based approaches. However, intensity-based approaches are computationally expensive and are not robust against occlusions or clutter, especially when area-based strategies are involved. In contrast, approaches based on low level features often combine a higher robustness with efficient computations, especially in the case of non-area-based strategies. High level features are often not of a general nature but only applicable to a specific type of object. They often imply complex and computationally expensive algorithms to compute the features or to solve the correspondence problem. Some object recognition approaches inherently return the pose of the object with high precision, while other approaches have to be complemented by an additional refinement method. Furthermore, the often inefficient brute-force computation of several object recognition approaches can be speeded up by using approach-specific sophisticated short-cuts during the online phase. Additionally, some general methods for speed-up, like the use of a coarse-to-fine approach, are applicable to most object recognition approaches.

The GHT as one representative of the class of non-area-based object recognition methods that uses low level features is particularly efficient because of its inherent translation invariance. Only the edge position and the edge direction are involved as features, and not higher level features. Therefore, the feature extraction time should be manageable even for real-time applications. The GHT is also robust against occlusions, clutter, and changes in brightness up to a certain degree. Furthermore, a low rate of false positives can be expected from this method because the edge direction is taken into account. Obviously, the GHT as a rigid object recognition method shows a very high potential to serve as core module within the approach for real-time recognition of compound objects. Therefore, in the following section the GHT will be analyzed in more detail and several novel modifications will be proposed to enhance its performance. In the subsequent section the shape-based matching, which was excluded from the review in this section because of its special role within this dissertation, will be introduced. To assess the performance of the modified generalized Hough transform and of the shape-based matching, their performances will be evaluated and compared to the performances of several other approaches in the last section of this chapter. For this, three of the reviewed standard approaches are selected. In spite of some known drawbacks of these approaches, it is highly desirable to include them in the evaluation because they are widespread methods and used in the majority of industrial object recognition applications. By this, a common denominator is created that facilitates the comparability of the presented evaluation with other evaluations that also include one of the three approaches. Additionally, three commercial high-end recognition tools will be evaluated. They have not been included in the review since no detailed information about their principle is available. Nevertheless, their performance helps to emphasize the high potential of the new approaches.

4.2 Modified Generalized Hough Transform

The principle of the GHT as well as some inherent advantages have already been addressed in Section 4.1.1.2. However, the discussion has been done on a very coarse level of detail. On the one hand, this level of detail should be sufficient to understand the general principle and to compare the GHT to other object recognition approaches. On the other hand, it is not detailed enough to enable the reader to obtain a comprehensive impression of the GHT. However, a comprehensive impression is essential to understand the modifications that have been applied within the scope of this dissertation in order to improve the GHT. The goal of the modifications is to make the GHT fulfill the requirements concerning the recognition of rigid objects introduced in Section 2.2. Therefore, in the following the principle of the GHT is explained in more detail. Furthermore, the inherent advantages and drawbacks of the GHT are elaborated and used as basis for the proposed modifications. Some modifications are based on the ideas of the previously introduced methods concerning pose refinement (cf. Section 4.1.2) and speed-up (cf. Section 4.1.3). These ideas are picked up and are further extended.

4.2.1 Generalized Hough Transform

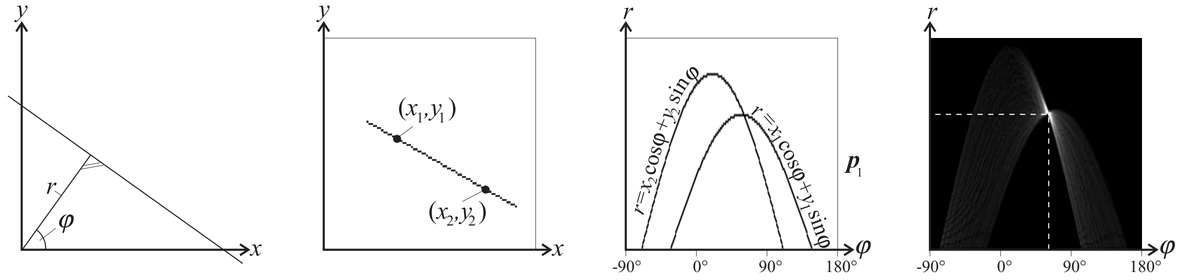
4.2.1.1 Principle

The conventional Hough transform (HT) is a standard method to efficiently detect analytical curves (e.g., lines, circles, ellipses) in images. Although the GHT is built on the idea of the HT, they both are independent methods. Therefore, in this dissertation only the idea of the HT and its relations to the GHT are introduced. For further details, the interested reader should refer to (Hough 1962) or to standard text books like (Ballard and Brown 1982, Jähne 2002).

To introduce the HT, the problem of detecting straight lines in images is considered. A straight line can be described by the points $(x, y)^\top$ that fulfill

$$r = x \cdot \cos\varphi + y \cdot \sin\varphi \quad . \tag{4.13}$$

Thus, the line is represented by its distance to the origin r and its orientation φ (see Figure 4.1(a)). Figure 4.1(b) shows an image after edge segmentation in which lines should be detected. Each edge pixel in the (x, y) image space describes a sinusoidal curve in the (r, φ) parameter space, which is also referred to as *Hough space* in the literature. This can be seen from (4.13) by treating x and y as fixed and letting r and φ vary. Thus, the corresponding sinusoidal curve in the parameter space represents all lines in image space that meet in the same image point $(x, y)^\top$. In Figure 4.1(c) the sinusoidal curves of the two example edge pixels shown in Figure 4.1(b) are displayed. All image points on the same line intersect at the same point in the parameter space. This relation between image space and parameter space is exploited in the HT. For this, the parameter space is divided into rectangular cells and represented by an accumulator array. In the first stage, each edge pixel is transformed into the Hough space and the corresponding cells are incremented. The second stage is an exhaustive search for maxima in the accumulator array. The maxima represent the parameters of the straight lines in the image. Figure 4.1(d) shows the resulting accumulator array, where higher values of the cells are visualized by brighter gray values. The maximum represents the found straight line. The extension of

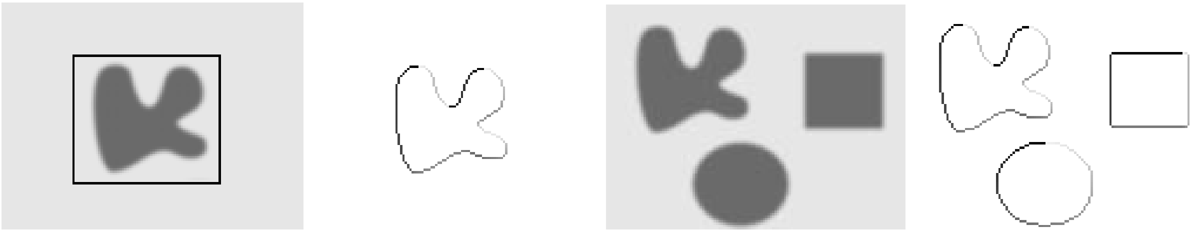

y
x
r
φ
y
x
(x_1,y_1)
(x_2,y_2)
r
$r=x_2\cos\varphi+y_2\sin\varphi$
$r=x_1\cos\varphi+y_1\sin\varphi$
p_1
φ
-90°
0°
90°
180°
r
φ
-90°
0°
90°
180°

(a) Line parameterization (b) Segmented image (c) HT of 2 points (d) Accumulator array

Figure 4.1: Example that illustrates the detection of straight lines using the HT. Lines are parameterized using a polar coordinate representation (a). The edge pixels of the segmented image (b) are transformed into the Hough space (c). The corresponding cells in an accumulator array are incremented and the maximum is extracted (d).

this technique to detect curves other than straight lines is straightforward. By using the gradient direction as additional information fewer cells need to be incremented, which results in faster computations (Ballard and Brown 1982). For example, when detecting straight lines, points in image space can be transformed to points in parameter space. The main advantages of the HT are that it is relatively unaffected by gaps in the curves and by noise (Ballard and Brown 1982).

In contrast to the HT, the GHT is not restricted to analytical curves. The parameters that describe an analytical curve in the HT correspond to the pose parameters of the object in the GHT. In the offline phase of the GHT, the *R*-table is built from a model image that shows the object to be recognized. Thus, in the special case of the GHT the *R*-table represents the model that will be later used to recognize the object in the search image during the online phase. In the following, a simple example will illustrate the principle of the GHT. In the example, the transformation class is restricted to translations to simplify the explanations. In Figure 4.2(a) a model image is given, in which the object is defined by a ROI, which in this example is the inner part of the black rectangle. In a preprocessing step of the offline phase, edge filtering is performed on the model image, resulting in the edge magnitude and gradient direction. By applying a threshold on the edge magnitude, the model edges (pixels or points) are obtained, which are shown in Figure 4.2(b). Additionally, the gradient direction is visualized by using different gray values. This is all the information that is needed to compute the *R*-table.



(a) Model image and ROI (b) Model edges (c) Search image (d) Search edges

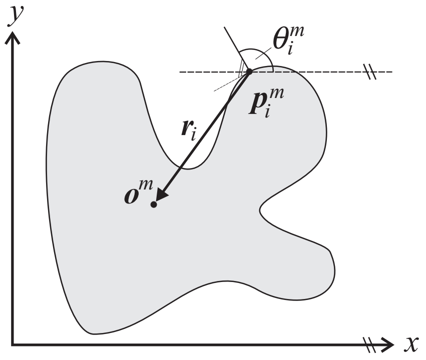
Figure 4.2: In the offline phase, based on a model image (a) edges are extracted and their gradient directions are computed (b). For visualization purposes the gradient direction is encoded with different gray values. In the online phase, also for the search image (c) edges and the corresponding gradient directions are computed (d).

The principle is illustrated in Figure 4.3. At first, an arbitrary reference point $\boldsymbol{o}^m = (o_x^m, o_y^m)^\top$ is chosen. In general, $\boldsymbol{o}^m$ is selected to be the centroid of all model edge points, i.e., $\boldsymbol{o}^m = 1/n^m \sum \boldsymbol{p}_i^m$. For each model edge point the displacement vector

$$\boldsymbol{r}_i = \boldsymbol{o}^m - \boldsymbol{p}_i^m, \ \ \forall i = 1, \ldots, n^m \tag{4.14}$$

is calculated. The displacement vectors are then stored in the *R*-table as a function of θ_i^m, where θ_i^m denotes the gradient direction at the model edge point $\boldsymbol{p}_i^m$. For this purpose, the range of possible gradient directions must be quantized using quantization intervals of size $\Delta\theta$. In general, the gradients occur in arbitrary direction, and hence the range of possible gradient directions corresponds to the interval of $[0°, 360°[$. Each row k of the *R*-table is then assigned one quantization interval Θ_k, $k = 0, \ldots, n^\theta - 1$, leading to an overall number of $n^\theta = \frac{2\pi}{\Delta\theta}$ rows. Finally, the displacement vector $\boldsymbol{r}_i$ of the model edge point $\boldsymbol{p}_i^m$ is recorded in the row of the *R*-table that contains the associated gradient direction θ_i^m.

The example *R*-table in Figure 4.4 is built by using four selected model edge points. Here, the quantization interval for the gradient directions was set to $\Delta\theta = 60°$. Since the gradient directions at the two points $\boldsymbol{p}_2^m$ and $\boldsymbol{p}_3^m$ are identical, both associated displacement vectors are recorded in the same row within the *R*-table.

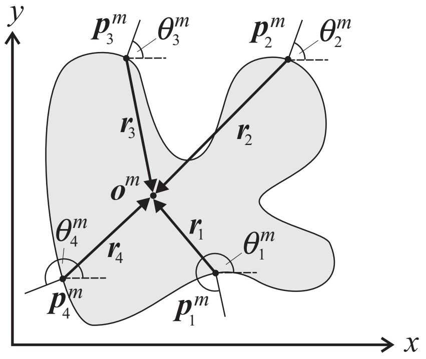


(a) Model edges

k	Θ_k	$\boldsymbol{r}_i$
0	$[\ \ 0,\ \ \Delta\theta[$	$\{\boldsymbol{r}_i \mid \theta_i^m \in \Theta_0\}$
1	$[\ \Delta\theta, 2\Delta\theta[$	$\{\boldsymbol{r}_i \mid \theta_i^m \in \Theta_1\}$
2	$[2\Delta\theta, 3\Delta\theta[$	$\{\boldsymbol{r}_i \mid \theta_i^m \in \Theta_2\}$
⋮	⋮	⋮

(b) *R*-table

Figure 4.3: Information about the model edges (a) is used to build the *R*-table (b).



(a) Four selected example model edge points

k	Θ_k	$\boldsymbol{r}_i$
0	$[\ \ 0°,\ \ 60°[$	
1	$[\ 60°, 120°[$	$\boldsymbol{r}_2, \boldsymbol{r}_3$
2	$[120°, 180°[$	
3	$[180°, 240°[$	$\boldsymbol{r}_4$
4	$[240°, 300°[$	$\boldsymbol{r}_1$
5	$[300°, 360°[$	

(b) Example *R*-table with $\Delta\theta = 60°$

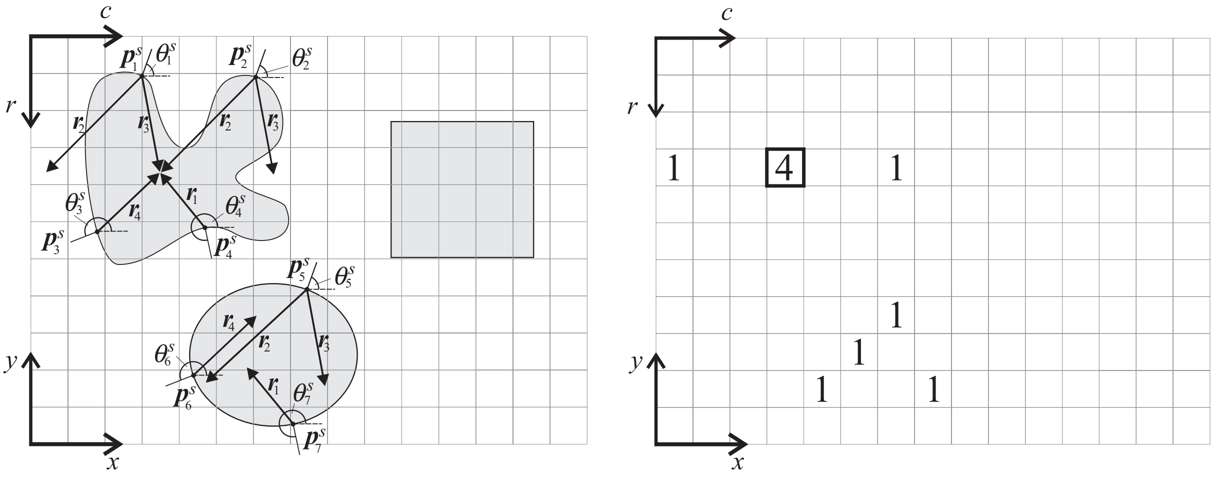
Figure 4.4: Example that illustrates the offline phase of the GHT. The gradient directions at the four selected example model edge points are $\theta_1^m = 280°$, $\theta_2^m = 65°$, $\theta_3^m = 65°$, $\theta_4^m = 200°$. The associated displacement vectors $\boldsymbol{r}_i$ are recorded in the appropriate row of the *R*-table according to their gradient direction.

For the online phase a two dimensional accumulator array A is set up over the domain of translations, representing the sampled parameter space or Hough space of $\mathcal{T}$. Figure 4.5 shows the principle of an accumulator array for the example search image of Figure 4.2. This is similar to the HT, however, each cell of this array now corresponds to a certain range of positions of the reference point $\boldsymbol{o}^s = (o_x^s, o_y^s)^\top$ in the search image. In general, the size of the cells is adapted to the pixel grid, i.e., each cell represents one pixel. The accumulator array is initialized by setting the values of all cells to 0. For each edge pixel $\boldsymbol{p}_j^s$ in the search image the row k in the *R*-table that corresponds to the gradient direction θ_j^s is selected. Each displacement vector that is recorded within the selected row represents the position of one reference point candidate $\breve{\boldsymbol{o}} = (\breve{o}_x, \breve{o}_y)^\top$ relative to $\boldsymbol{p}_j^s$

in the search image. Formally, the displacement vectors are added to $\boldsymbol{p}^s_j$ in order to obtain the reference point candidates:

$$\breve{\boldsymbol{o}}_{i,j} = \boldsymbol{p}^s_j + \boldsymbol{r}_i, \ \forall j = 1, \dots, n^s, \ \forall \{i | \theta^m_i \in \Theta_k\}, \ k | (\theta^s_j \in \Theta_k) \ . \tag{4.15}$$

Finally, each cell in the accumulator array A that is hit by one reference point candidate receives a vote, i.e., its value is incremented by one. After the voting process, each cell in A has a value that describes the likelihood that the reference point is located in this cell. Thus, local maxima in A that exceed a certain threshold represent found object instances in the search image. Figure 4.5 shows the principle of the online phase by means of seven selected search edge points. In Figure 4.5(a) the cells of the accumulator array are overlaid on the search image of Figure 4.2. For illustration purposes, in this case one cell covers several pixels. For example, the edge direction θ^s_1 at point $\boldsymbol{p}^s_1$ is 65°. In the *R*-table of Figure 4.4(b) the respective gradient interval is $\Theta_1 = [60°, 120°[$, and hence $k = 1$. The two displacement vectors $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$ that are recorded in row $k = 1$ are added to the point position $\boldsymbol{p}^s_1$ and the two obtained reference point candidates are used to increment the two corresponding cells. Figure 4.5(b) shows the final accumulator array after the voting process, where the number of votes are entered in each cell. The cell with maximum number of votes represents the position of the reference point, and hence the found object instance in the search image.



(a) Accumulator array and voting process

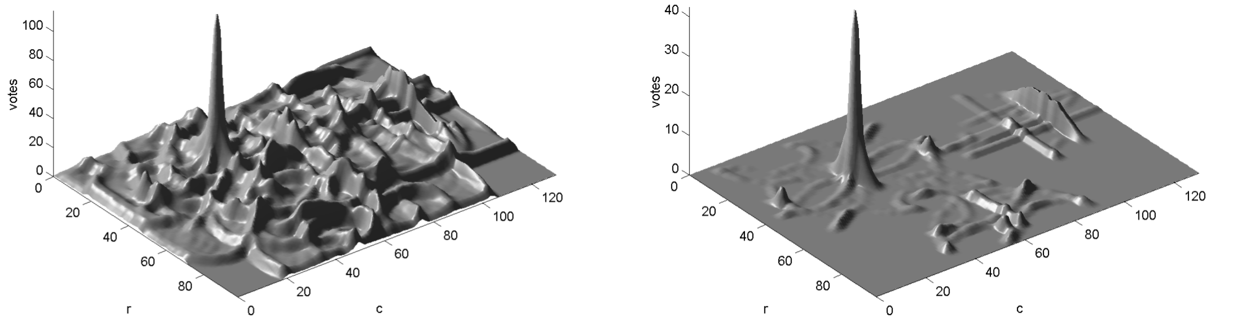
(b) Final votes (empty cells have zero votes)

Figure 4.5: Example that illustrates the online phase of the GHT. The gradient directions at seven selected example search edge points are $\theta^s_1 = 65°$, $\theta^s_2 = 65°$, $\theta^s_3 = 200°$,$\theta^s_4 = 280°$, $\theta^s_5 = 65°$, $\theta^s_6 = 200°$, and $\theta^s_7 = 280°$ (a). According to the gradient direction the displacement vectors in the respective row of the *R*-table are added to the point positions, and the corresponding cells receive a vote (b).

4.2.1.2 Advantages

The GHT shows several inherent advantages already in its original form. In contrast to alignment methods that use an arbitrary similarity measure, it is not necessary to translate the model image to all possible positions of the object in the search image. Instead, the translation is regarded implicitly since the model edges are stored relative to the reference point. This saves significant computation time.

Further savings in computation time are achieved by taking the gradient direction information into account. Thus, for one edge point in the search image only the model edge points with similar gradient direction instead of all model edge points are used for incrementing the respective cells. Considering the gradient direction leads to another important advantage. The robustness against clutter is increased considerably, i.e., the probability of false positive matches is decreased. This effect is illustrated in Figure 4.6, which shows the accumulator array after the voting process. In contrast to Figure 4.5, now one cell corresponds to one pixel. The number of votes in each cell is visualized as height in the 3D plots. Figure 4.6(a) shows the result of the voting process in the case that no gradient direction information is taken into account. This can be simulated by creating an R-table that consists of only a single row, i.e., $\Delta\theta = 360°$. Consequently, for each edge pixel in the search image all displacement vector are used for voting, independent of their associated gradient direction. Although the artificially created search image is free of noise and the two additional objects differ from the searched object, the accumulator array shows noisy regions and some additional (albeit small) peaks. The signal-to-noise-ratio (SNR), calculated as the peak-signal-to-peak-noise in this case is 3.3. In Figure 4.6(b) the same situation is applied, however, the gradient direction was quantized using an interval size of $\Delta\theta = 5°$. One can see that the noise in the accumulator array has diminished significantly. The almost doubled SNR of 6.5 confirms the visual impression. In this example, an acceptable SNR is obtained even if no gradient direction is considered, and hence the intrinsic peak can be easily distinguished from the noise in the accumulator array. However, one can imagine the problems when dealing with noisy, highly textured images, which contain additional objects that are similar to the object to be found. In such cases the gradient direction is an absolutely essential information.



(a) Neglecting gradient direction

(b) Considering gradient direction

Figure 4.6: The accumulator array shows several peaks after the voting process. The signal-to-noise ratio (SNR) can be significantly increased by considering the gradient direction information. In this example, the SNR almost doubles from 3.3 (a) to 6.5 (b).

Another advantage of the GHT is its high robustness against occlusions. At first glance, one may think that occlusions in the search image affect the position of the reference point (e.g., the centroid), and hence cause the voting process to fail. However, this is not the case. In contrast to moment-based approaches, for example, which require the object as a whole to be present in the image, the GHT is tolerant against a certain degree of occlusions. Looking at the principle of the GHT again, the reference point in the search image is not computed as the centroid of all edge points but is the result of the maximum search after the voting process. If some of the edge pixels in the search image are missing because of partial occlusions, the peak height is merely reduced in proportion to the fraction of occluded edge pixels. This is a rather intuitive behavior.

Furthermore, the GHT is robust against changes in brightness of an arbitrary type up to a certain degree. Since the recognition is based on the image edges instead of the raw gray value information, changes in brightness do not affect the recognition unless the edge magnitude falls below the threshold that is applied to edge segmentation. The features that are used in the GHT are edge position and gradient direction. Both features do not change dramatically when the brightness or the contrast in the image changes. However, if the contrast is too low in some object parts of the search image fewer edges will be segmented. This leads to the same effect as partial occlusions. Nevertheless, if the contrast is too low in the entire image the GHT fails to recognize the object, unless the segmentation threshold is set to a lower value.

4.2.1.3 Drawbacks

In this dissertation objects must be found under the transformation class of rigid motion. Unfortunately, the GHT was originally designed to cope with object translations only. If one wants to recognize objects that may appear in an arbitrary orientation the GHT must be extended accordingly: now for each possible orientation φ^s of the object in the search image a separate *R*-table is generated by using a certain orientation step $\Delta\varphi$ to sample the range of possible orientations. Depending on $\Delta\varphi$ and the orientation range in which the object may appear, a number of *R*-tables is generated at discrete orientations φ_r, $r = 0, \ldots, n^\varphi - 1$, where n^φ denotes the number of discrete orientations, and hence the number of *R*-tables. Finally, each *R*-table is labeled with its associated orientation φ_r. In the online phase, n^φ separate 2D accumulator arrays are built, one for each *R*-table. All single 2D arrays can be put together into one 3D accumulator array. The first two dimensions represent the position of the reference point $\boldsymbol{o}^s$ and the third dimension represents the orientation of the object φ^s in the search image. The voting process is then performed for each *R*-table separately, where the votes are entered in the respective 2D hyperplane of the accumulator array that is associated with the current *R*-table. The position of local maxima in the 3D array that exceed a threshold describe the position and orientation of found object instances. Obviously, the computational effort in the online phase increases linearly with the number of *R*-tables, and hence with the number of rotation steps.

Thus, one weakness of the GHT algorithm is the — in general — huge parameter space represented by the 3D accumulator array A when allowing rigid motion. Assume an image of size 768 × 576, a rotation step of $\Delta\varphi = 1°$, and an object that may appear at an arbitrary orientation. Hence, the number of discerned orientation steps n^φ is 360. Consequently, the accumulator array would contain $768 \times 576 \times 360 = 159 \cdot 10^6$ cells. This involves large amounts of memory to store the accumulator array as well as high computational costs in the online phase for the initialization of the array and the search for local maxima after the voting process.

Another crucial part is the large number of voting events n^{vote} that have to be performed. The average number of voting events is

$$n^{vote} = n^\varphi n^s \frac{n^m}{n^\theta} \quad , \tag{4.16}$$

where $n^\theta = \frac{2\pi}{\Delta\theta}$ again is the number of different gradient direction intervals, and hence the number of rows in the *R*-tables. Assuming a non-cluttered search image, i.e., no additional edges are present, and thus $n^s = n^m$, the effort for incrementing the respective cells in the accumulator array increases quadratically with the number of model edge pixels n^m. This behavior hinders real-time performance, especially for large or highly textured objects.

Additionally, the conventional GHT is sensitive to image noise. Although the edge position is relatively unaffected by a moderate degree of noise, the gradient direction is much more sensitive. Since the gradient direction is directly involved as feature within the GHT, image noise also has an impact on the performance of GHT. Especially, when the size of the gradient direction intervals $\Delta\theta$ is chosen small, the robustness may significantly decrease.

Finally, the properties of the GHT lead to the fact that the accuracies achieved for the returned pose parameters depend on the quantization of the transformation class of rigid motion. Thus, in the mentioned example the accuracy of object position is limited to the pixel size and the accuracy of object orientation is limited to 1°. Unfortunately, in practice the quantization cannot be chosen arbitrarily fine because of memory requirements and computation time. Furthermore, the robustness against clutter would decrease considerably when increasing the resolution of the accumulator array. This is because the peak would be dispersed over an increasing number of cells in the accumulator array.

In the following, the mentioned problems will be tackled, resulting in a modified Generalized Hough Transform (MGHT) (Ulrich et al. 2001b, Ulrich et al. 2001a): a multi-resolution model and search strategy in combination with an effective limitation of the search space is introduced in order to increase the efficiency, and hence meet the real-time requirements. Furthermore, a technique for refining the returned pose parameters without noticeably decelerating the online phase is presented. In addition, some crucial quantization problems and their solutions to increase the robustness of the GHT are discussed.

4.2.2 Computation of the *R*-tables

First of all, some general comments on the computation of the *R*-tables for the different object orientations should be made. Feature extraction is a prerequisite for applying the principle of the GHT for object recognition. In this dissertation the edge magnitude and the gradient direction are computed using the Sobel filter. This filter represents a good compromise between computation time and accuracy: it can be computed in less time than, for instance, the Lanser or Canny filters. Its anisotropic response and its worse accuracy can be balanced by the proposed approach: it will be shown in this section that the effect of the anisotropy can be reduced considerably by selecting an appropriate method for the computation of the *R*-tables. The residual anisotropy and the worse accuracy can be balanced by choosing an adequate quantization $\Delta\theta$ of the gradient directions, which will be explained in Section 4.2.5. Both the edge magnitude γ and the gradient direction θ are computed from the first partial gray value derivatives in x and y direction, which are returned by the Sobel filter. The edge pixels are segmented by applying a threshold γ^{min} on the edge magnitude. This threshold must be set by the user because it is highly correlated with the task of the application and with the type of object to be recognized. A subsequent non-maximum suppression eliminates the edge pixels that are no local maxima in the direction of the gradient. For this, the current edge pixel is compared to the two adjacent pixels in the gradient direction. However, because of the discrete structure of the image, still two pixel wide edges may occur. Therefore, the number of edge pixels can be further reduced by computing the skeleton based on a medial axis transform (Ballard and Brown 1982). Thus, by using only the most representative edge pixels in the *R*-table the computational effort during the online phase can be kept small.

To allow object rotation, for each object orientation φ a separate *R*-table must be built. In general, this can be accomplished by following one of four possibilities. The four possibilities arise by combining one of two methods to build the *R*-tables with the decision whether the *R*-tables are built within the offline phase or within the online phase.

The first method to build the *R*-tables is to compute the *R*-table for the model image as described before, yielding a prototype *R*-table. The *R*-tables of all discrete orientations φ_r are then obtained by applying simple transformations on the prototype *R*-table, as proposed in (Ballard 1981). For this, the limits of all gradient intervals $\Theta_k,\ 0 \leq k < n^\theta$ of the prototype *R*-table are just incremented by the current rotation angle φ_r. Furthermore, all displacement vectors within the prototype *R*-table are rotated by φ_r. The second method to build the *R*-tables is to rotate the model image itself and generate an *R*-table based on each of the rotated model images. When using the Sobel filter the latter method has a crucial advantage. By deriving the *R*-tables from the rotated model images the error caused by the anisotropy of the Sobel filter can be eliminated since the anisotropy is the same in the rotated model image and in the search image, and hence the error cancels out. In contrast, the advantage of the first method is its lower computational effort, since only a “small” number of displacement vectors must be rotated but not the entire image.

Furthermore, both methods can be applied either in the offline phase or in the online phase leading to four different combinations. Pre-computing the *R*-tables in the offline phase is more memory-intensive since all rotated *R*-tables must be stored in memory. However, it facilitates a substantially more efficient object recognition during the online phase since no rotations must be performed online. In this dissertation the *R*-tables are derived from the rotated model image in the offline phase. This, on the one hand, optimizes the computation time of the online phase and, on the other hand, eliminates the anisotropy error. Hence, the second computation method is combined with computing the *R*-tables offline.

Table 4.2 summarizes the four possibilities and additionally points out the respective advantages and drawbacks. The combination used in this work is emphasized in bold type.

It should be noted that in some applications it might be less important to achieve real-time computation but to keep the required memory amount as small as possible. Then it would be preferable to compute the required *R*-tables online. Furthermore, in such cases it might be desirable to use the first computation method to allow a reasonable computation time of the online phase even though the anisotropy error must be accepted.

	Computing R-tables offline	Computing R-tables online
First computation method 1. computing prototype R-table from model image 2. transforming prototype R-table for all orientations	• fast offline phase • high memory for model • very fast online phase • anisotropy error	• very fast offline phase • low memory for model • slow online phase • anisotropy error
Second computation method 1. rotating model image 2. computing R-tables from rotated images	• **slow offline phase** • **high memory for model** • **very fast online phase** • **no anisotropy error**	• very fast offline phase • low memory for model • very slow online phase • no anisotropy error

Table 4.2: Four different possibilities to compute the R-tables when allowing object rotations

4.2.3 Increasing the Efficiency

4.2.3.1 Multi-Resolution Model

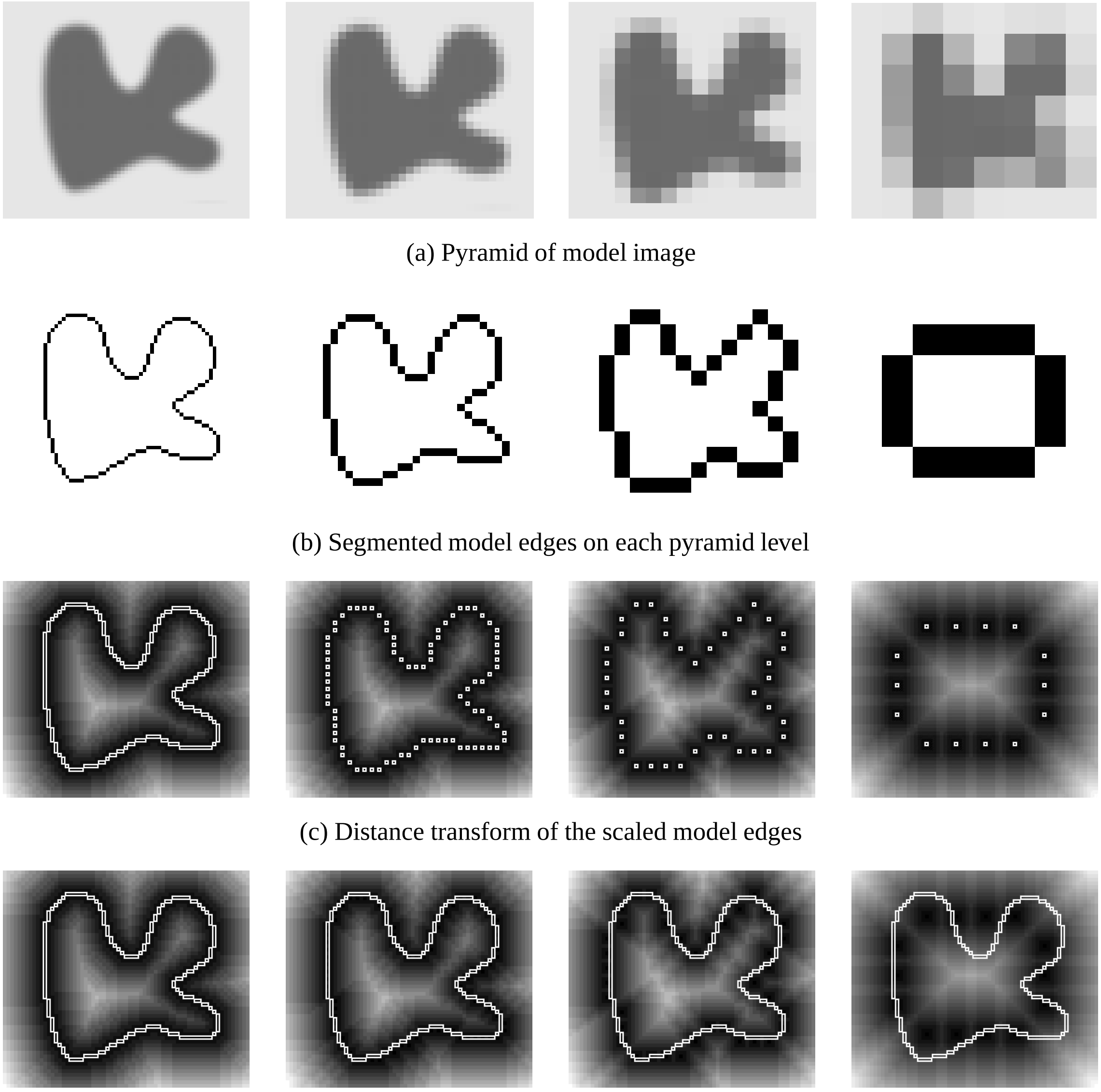
To reduce the size of the accumulator array and to speed up the online phase, the original GHT is embedded in a coarse-to-fine framework using image pyramids as described in Section 4.1.3. This coarse-to-fine approach affects both the offline phase and the online phase. In the offline phase, it leads to the generation of a multi-resolution model. The construction of this multi-resolution model will be described below.

At first, an image pyramid of the model image is generated. Let I_l^m, be the model image at pyramid level l, $l = 0, \ldots, n^l - 1$, where n^l denotes the number of involved pyramid levels. I_0^m represents the model image I^m at original resolution. For increasing values of l the resolution, and hence the image dimensions, are successively halved. To obtain the image I_l^m at pyramid level l, the image I_{l-1}^m is smoothed using a mean filter of size 2×2 in order to meet the Nyquist theorem, and sub-sampled using a sampling interval of 2 pixels, as described in Section 4.1.3.

When determining the optimum value for n^l two conflicting objectives must be balanced. On the one hand, the number of pyramid levels should be chosen as high as possible to obtain a high potential for speeding up the recognition process. On the other hand, the object on the top pyramid level, which has the lowest resolution, must still be recognizable. I.e., the object must still exhibit significant characteristics that keep it distinguishable from other objects in the image. Formally, the number of pyramid levels must be maximized under the constraint that object characteristics are preserved. To avoid burdening the user with an additional input parameter and to ensure a high degree of automation, in the following a method that automatically computes n^l will be introduced. Obviously a meaningful description of the object is impossible if the number of model edge pixels on the current level falls below a certain threshold. This represents the first criterion that must be fulfilled. Several practical tests have shown that pyramid levels containing less than ten model points can be discarded.

A minimum number of edge pixels is a necessary but in no way a sufficient requirement. Therefore, a more sophisticated approach must be applied when evaluating the requirement for preserved object characteristics. The principle of this second criterion is illustrated in Figure 4.7.

At first, an image pyramid is computed on the model image using the maximum number of levels, i.e., the top pyramid level is only one pixel wide in at least one dimension. Figure 4.7(a) shows the first four pyramid levels of the example model image. On all pyramid levels edges are segmented (see Figure 4.7(b)). For each pyramid level that fulfills the criterion of a minimum number of model edges pixels, the edges are scaled back to the original resolution and a distance transform (Jähne 2002) is computed on the scaled edges using the chamfer distance (cf. Section 4.1.1.2) for high accuracy. Figure 4.7(c) shows the scaled edges and the associated distance transforms, where brighter gray values represent higher distances. Finally, the model



(a) Pyramid of model image

(b) Segmented model edges on each pyramid level

(c) Distance transform of the scaled model edges

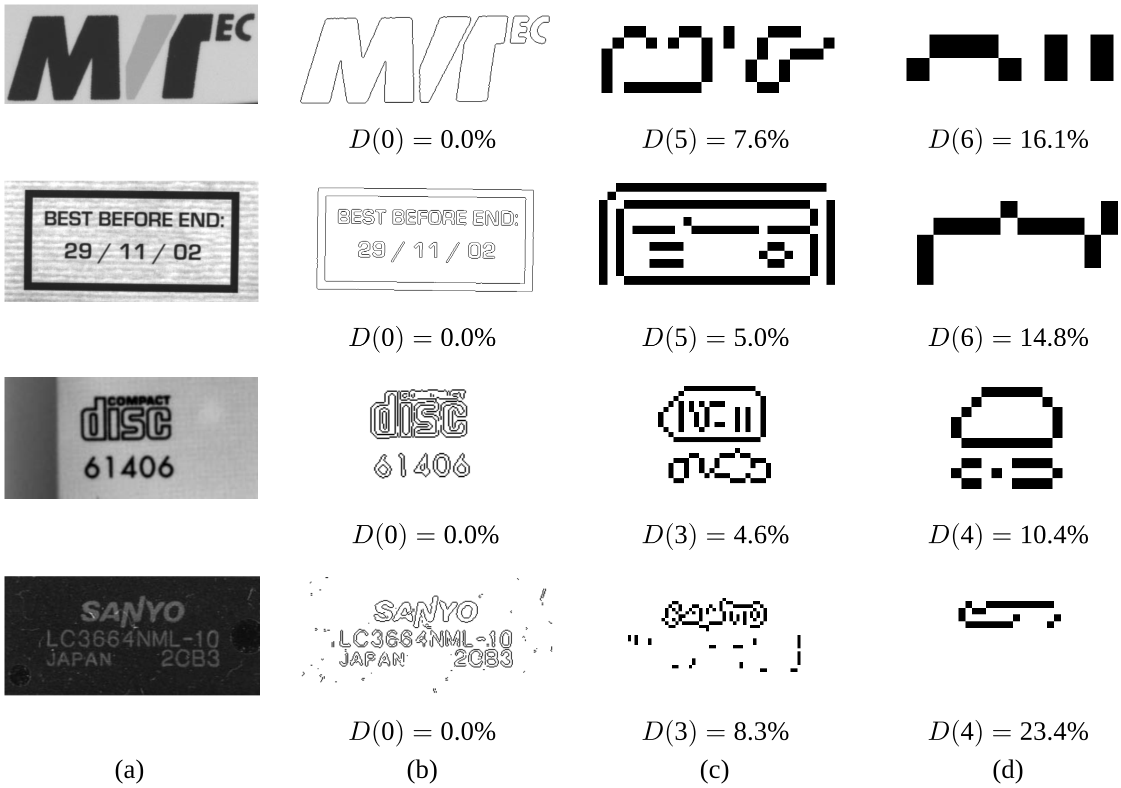
(d) Model edges at original resolution superimposed on the distance transform

Figure 4.7: The number of pyramid levels is computed automatically by measuring the deformations of the model edges on the respective pyramid levels ($D(0) = 0\%$, $D(1) = 1\%$, $D(2) = 3\%$, $D(3) = 9\%$).

edges at the original resolution are superimposed on the distance-transformed image and the mean distance of the original model edges to the (scaled) edges at the current level is calculated by summing up the underlying gray values (see Figure 4.7(d)). Hence, this is a measure of how much the model edges are deformed by the smoothing that comes with the image pyramid. The average distance is normalized by dividing it by the size of the object. This takes into account that small objects are already less distinctive and therefore allow only small deformations while bigger objects can cope with higher deformations without loosing their distinctive characteristics. The object size is represented by the radius r_0 of a circle that has the same area as the ellipse that has the same moments as the model edges at original resolution. Thus, the normalized average distance is a measure of deformation $D(l)$ that describes how much the original shape is degenerated on pyramid level l. $D(l)$ is computed as:

$$D(l) = \frac{\sum_{i=1}^{n^m} \min_{j=1,\ldots,n_l^m} \|\boldsymbol{p}_i^m - \Lambda(\boldsymbol{p}_{j,l}^m, l)\|}{3 n^m r_0} \quad , \tag{4.17}$$

where $\boldsymbol{p}_{j,l}^m$, $j = 1, \ldots, n_l^m$ are the model edge points at pyramid level l, $\|\cdot\|$ is the chamfer distance, and



$D(0) = 0.0\%$ $D(5) = 7.6\%$ $D(6) = 16.1\%$

$D(0) = 0.0\%$ $D(5) = 5.0\%$ $D(6) = 14.8\%$

$D(0) = 0.0\%$ $D(3) = 4.6\%$ $D(4) = 10.4\%$

$D(0) = 0.0\%$ $D(3) = 8.3\%$ $D(4) = 23.4\%$

(a) (b) (c) (d)

Figure 4.8: The number of pyramid levels are calculated for four examples. From the model image (a) edges are extracted (b). The edges of the highest accepted pyramid level that fulfills the criterion $D(l) < 10\%$ (c) and the edges of the lowest non-accepted level (d) are shown.

$\Lambda(\boldsymbol{p}^m_{j,l}, l)$ is the scaling of the model edge points from the current level l back to the original resolution:

$$\Lambda(\boldsymbol{x}, l) = 2^l \boldsymbol{x} \ . \tag{4.18}$$

The division by three in (4.17) again compensates the unit length of the chamfer distance. Finally, $D(l)$ must not exceed a certain threshold for all involved pyramid levels. This threshold is generic and independent from the object and can be determined empirically. Several experiments involving different types of objects with various sizes have shown that $D(l)$ should not exceed 10% in order to avoid strong deformations.

In order to visualize the theoretical results, four practical examples are presented in Figure 4.8. Figure 4.8(c) shows the edges of the top pyramid level that has just been accepted by the algorithm. One can see that in most cases the result of the automatic determination of the number of pyramid levels is very intuitive, except maybe for the example in the third row, where one might chose the fourth instead of the third level as top pyramid level. However, the deformation measure $D(4)$ of 10.4% indicates a narrow decision.

After the pyramid of the model image has been derived, the generation of the multi-resolution model can be started. While generating the model, one has to distinguish between the top level $I^m_{n^l-1}$ and the lower levels. In the online phase, also for the search image an image pyramid is derived by computing the same number of pyramid levels as for the model image. A breadth-first search is then applied: the recognition process starts at the top pyramid level of the search image without any prior information about the transformation parameters $\boldsymbol{o}^s$ and φ^s available. Therefore, the conventional GHT is applied to the top pyramid level. As *top level strategy*, all cells in A that are local maxima and exceed a certain threshold are stored as *match candidates* and used to initialize approximate values on the next lower level. Thus, the coarse values on the top level are subsequently refined by tracking the match candidates down through the pyramid to the highest resolution of the original search image. Using the breadth-first strategy, all match candidates are refined at the current level before the candidates are tracked to the next lower level. The breadth-first strategy is preferable for various reasons, most notably because a heuristic for a best-first strategy is hard to define, and because depth-first search results in higher recognition times if all matches should be found (Steger 2002).

Unfortunately, the GHT in its conventional form is not very well suited for the use of image pyramids because the prior information cannot be used in a straightforward way, as it is the case when using alignment methods, for example. This is the reason why only for the top level the *R*-table is built in its conventional form as described in Section 4.2.2. Whereas on the lower levels $I^m_{n^l-2}$ to I^m_0 a modified strategy is employed to efficiently take advantage of the prior information, i.e., approximate transformation parameters, obtained from the next higher level. As *lower levels strategy*, during the refinement only the respective best match within a local neighborhood of the approximate transformation parameters in parameter space is further tracked (see the description of the descent policy of Section 4.1.3). This strategy combined with the top level strategy facilitates finding all matches in the image while keeping the computational effort low. The problems of using image pyramids within the GHT will be discussed in the following section. Additionally, the proposed solutions will be introduced.

4.2.3.2 Domain Restriction

The use of image pyramids in alignment methods naturally comes along with a significantly increased efficiency of the online phase. This is because the parameter space on the lower levels can be restricted to a local neighborhood around the approximate transformation parameters. The similarity measure needs then to be computed only for a small number of object poses specified by this local neighborhood. In contrast, the GHT only profits from the restriction of the parameter space in one of the three dimensions: the approximate value for the object's orientation can be used to apply only *R*-tables that are labeled with an orientation that is close to the approximate value. However, the two dimensions describing the object's position cannot be restricted in this straightforward manner. This is because of the inherent property of the GHT that it eliminates the two degrees of freedom by the position invariant description of the edge points with respect to the reference point. Thus, there is no explicit translation of the model image over the search image that could be restricted, which is the case when using alignment methods. Therefore, a more sophisticated approach must be applied to take advantage of the approximate pose parameters in all three dimensions.

Figure 4.9 may help to understand the principle that is illustrated in the following. Let $\tilde{\boldsymbol{o}}^s = (\tilde{o}^s_x, \tilde{o}^s_y)^\top$ and $\tilde{\varphi}^s$ be the approximate à priori values of the pose parameters at the current pyramid level, which are obtained from the upper level for a given match candidate. To refine the parameters in the current level, it is unnecessary to take the edge pixels in the entire image into account. Instead, $\tilde{\boldsymbol{o}}^s$ and $\tilde{\varphi}^s$ can be used together with the knowledge about the position of the edge points with respect to the reference point to optimally restrict the image domain: the approximate position of the edge points that belong to the match candidate can be computed in the current level. In this way the domain for edge extraction on the current level can be restricted to the neighborhood of the approximate edge position of all match candidates, where the neighborhood is defined by the propagated uncertainties of the approximate pose parameters. Thus, the restriction is not directly applied to the parameter space but to the image space by restricting the image domain. The uncertainty of the approximate parameter values are expressed by the associated standard deviations $\sigma_{\tilde{o}^s_x}$, $\sigma_{\tilde{o}^s_y}$, and $\sigma_{\tilde{\varphi}^s}$. These values can be derived from the corresponding maximum errors δx, δy, and $\delta\varphi$ that are to be expected when tracking the match candidate down one level. The maximum errors of position and orientation are visualized in Figure 4.9(a) by a gray square and a gray sector, respectively. Assuming an approximately equal distribution of the tracked parameters within the interval of $[-\delta, +\delta]$, respectively, the standard deviation (Bronstein et al. 2001) can be computed by

$$\sigma^2_x = \int\limits_{-\infty}^{+\infty} (x-\mu_x)^2 f(x)\,dx = \frac{1}{2\delta}\int\limits_{-\delta}^{+\delta} x^2\,dx = \frac{\delta^2}{3} \quad , \tag{4.19}$$

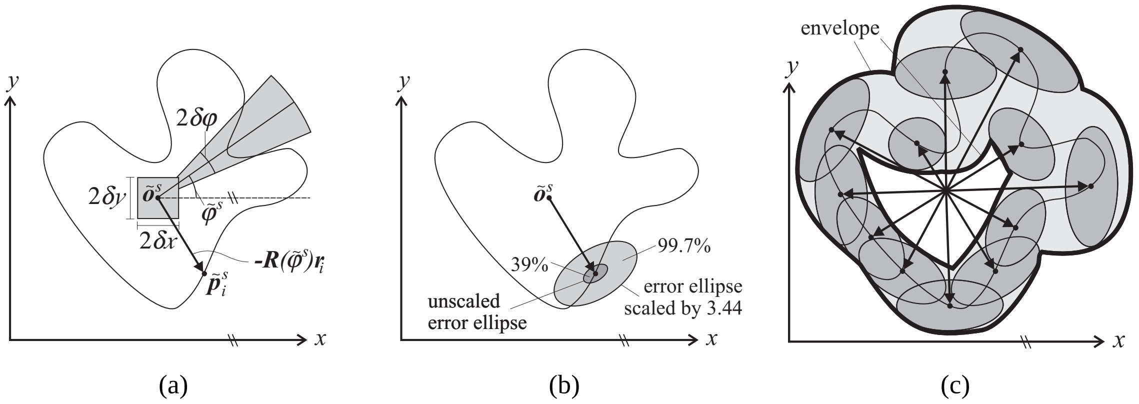


Figure 4.9: Approximate pose parameters are obtained from the upper pyramid level and can be used to calculate approximate edge positions (a). The uncertainty of the pose propagates to the edge position (b). The envelope of the error ellipses thus obtained for all edge points describes a region in which edge points lie with a certain probability (c).

finally leading to

$$\sigma_{\tilde{o}^s_x} = \frac{\delta x}{\sqrt{3}} \ , \ \sigma_{\tilde{o}^s_y} = \frac{\delta y}{\sqrt{3}} \ , \ \sigma_{\tilde{\varphi}^s} = \frac{\delta \varphi}{\sqrt{3}} \ . \tag{4.20}$$

The values for the maximum errors have been derived from several tests. During the tests it turned out that it is sufficient to set δx and δy to 3 pixels and $\delta\varphi$ to $3\Delta\varphi$. The approximate position of each model edge pixel in the search image $\tilde{\boldsymbol{p}}^s_i = (\tilde{x}^s_i, \tilde{y}^s_i)^\top$ can be obtained by transforming the associated negated displacement vector according to the approximate pose parameters (see Figure 4.9(a)):

$$\tilde{\boldsymbol{p}}^s_i = \tilde{\boldsymbol{o}}^s - \boldsymbol{R}(\tilde{\varphi}^s)\boldsymbol{r}_i \ . \tag{4.21}$$

Here, $\boldsymbol{r}_i$ represents the displacement vectors obtained from the non-rotated model image. The covariance matrix for the two coordinates of $\tilde{\boldsymbol{p}}^s_i$ is easily computed by applying the law of error propagation. The corresponding error ellipse (Gotthardt 1968) is obtained from the eigenvectors and eigenvalues of the covariance matrix (see Figure 4.9(b)). The error ellipse represents the boundary of a confidence region in which $\tilde{\boldsymbol{p}}^s_i$ lies with a probability of 39%. A more appropriate probability can be obtained by scaling the axis of the ellipse by a factor of 3.44 leading to a confidence of $1 - e^{-(3.44^2/2)} = 99.7\%$ (Gotthardt 1968). Finally, the envelope of all single confidence regions describes the image domain that the edge segmentation can be restricted to for the current match candidate, where on average only 0.3% of the edge pixels are missed (see Figure 4.9(c)).

Unfortunately, the exact computation of the envelope region, as is was described above, is extremely time consuming. Furthermore, the assumption of the equal error distribution is not entirely correct. Therefore, it is advisable and meaningful to simplify the computations. The strategy is illustrated in Figure 4.10. Again the approximate position of the model edge pixels in the search image is given (see Figure 4.10(a)). The idea is based on blurring the model edges according to the maximum errors of position and orientation. The maximum errors in position δx, δy are taken into account by dilating the edges using a structuring element of size $(2\delta x + 1) \times (2\delta y + 1)$ (see Figure 4.10(b)). A dilation is a morphological region operation that efficiently expands the input region using a certain structuring element (Soille 1999). Assuming that $\delta = \delta x = \delta y$ the dilation provides a region that is expanded by δ in both directions. The resulting dilated region is then successively rotated in both directions until the two extremes of the maximum error in orientation $\pm\delta\varphi$ are reached. The final region is obtained by merging all rotated regions (see Figure 4.10(c)). Since the sequence of the blurring affects the result, the blurring is done a second time by starting with the rotation and dilating in the second step. The two resulting blurred regions are merged by computing the union of the two regions.

The blurred region can be pre-computed in the offline phase, where for each pyramid level different from the top level and for each discerned orientation one region is computed. For the top pyramid level no blurred regions need to be computed because no prior information will be available. In the online phase, the respective blurred region is selected according to the pyramid level and the approximate object orientation $\tilde{\varphi}^s$ and is

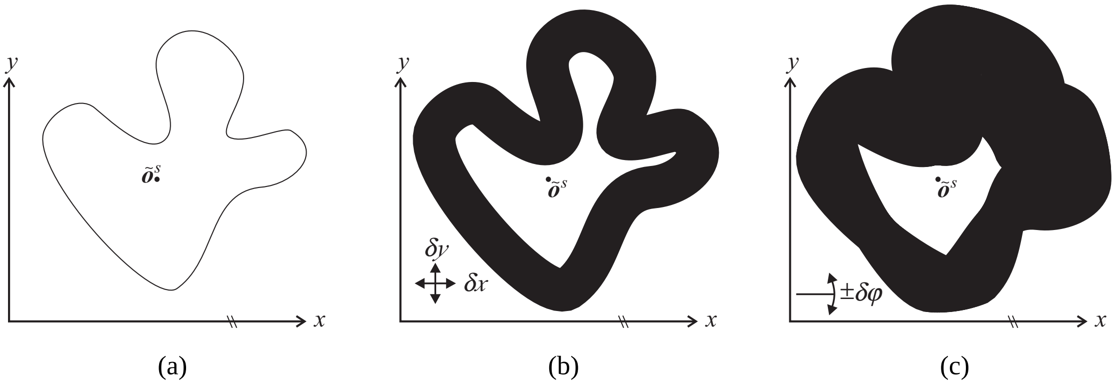


Figure 4.10: The computation of the blurred region is done in two steps. The model edges (a) are dilated (b) and successively rotated (c) according to the maximum errors of the approximate pose parameters.

superimposed on the current level of the search image at the position of $\tilde{\boldsymbol{o}}^s$. If several candidates are to be tracked, for each candidate the blurred region is superimposed. Edge extraction is then applied to the union of all superimposed regions. This increases the efficiency of the online phase dramatically, because not only the edge extraction is speeded up but also the voting process itself, since fewer edge pixels are involved. In addition, the size of the accumulator array A can be narrowed down according to the maximum errors of the à priori parameters. This decreases the memory amount considerably. For each candidate that has to be refined in the current level, an accumulator array merely of size $(2\delta x + 1) \times (2\delta y + 1) \times (2\delta\varphi + 1)$ is necessary. Using the values proposed above, the array now has dimensions of 7 × 7 × 7 for each candidate instead of using one single array of size 768 × 576 × 360, for example.

Figure 4.11 illustrates the use of the blurred region during the online phase. In this example, three image pyramids are used. On the top pyramid level the conventional GHT is performed, which results in two match candidates. The position and orientation of the two candidates are visualized in Figure 4.11(a) by black circles and arrows, respectively. By tracing the two candidates down one level, approximate pose parameters are obtained in level 1. To optimally restrict the image domain, the two corresponding blurred regions are superimposed based on the approximate poses (see Figure 4.11(b)). On this level the bottom match does not receive enough votes, and hence is discarded. Thus, on level 0 only one match candidate remains, for which the blurred region again is used to restrict the image domain accordingly (see Figure 4.11(c)).

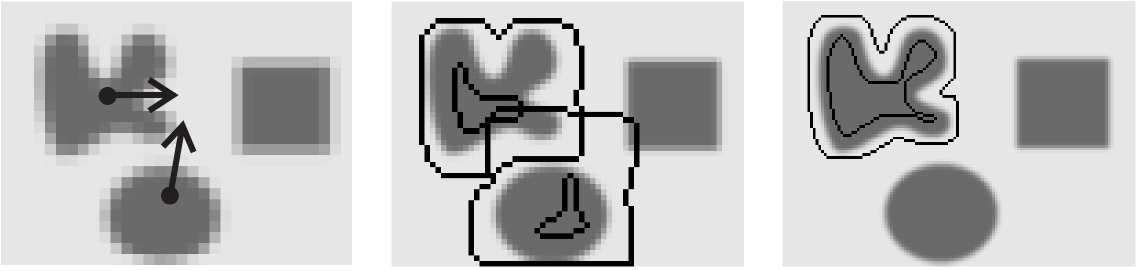
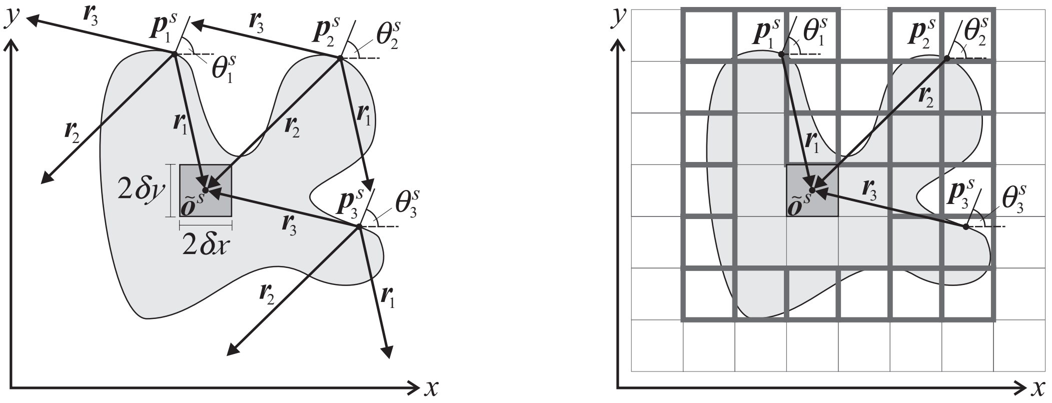

(a) Level 2 (top level)
(b) Level 1
(c) Level 0

Figure 4.11: The blurred region is superimposed at the approximate pose of the match candidates at lower pyramid levels to restrict the image domain for further processing.

While already in this artificial example the increased efficiency becomes clear, the restriction of the image domain becomes even more effective when dealing with highly cluttered search images or with objects that are small in relation to the image size. Furthermore, the use of the blurred region not only speeds up the online phase but also increases the robustness against clutter on lower pyramid levels. The reason for the increased robustness is that edges that are not in the neighborhood of the expected object edges are completely ignored. To avoid that already on the top pyramid level candidate matches are falsely eliminated, the threshold for the minimum number of votes should be slightly decreased for the top level.

4.2.3.3 Tiling the Model

A second improvement of the GHT leads to a further increase of efficiency. The principle of the conventional GHT combined with the multi-resolution approach results in redundant voting events on lower pyramid levels. This can be avoided by exploiting the prior information once more. The problem is shown in Figure 4.12(a). Again, the prior information is displayed as the gray square, which represents the maximum error of the approximate position parameters from the level above. This area will be referred to as the *approximate zone*. The three edge pixels $\boldsymbol{p}_1^s$, $\boldsymbol{p}_2^s$, and $\boldsymbol{p}_2^s$ have identical gradient directions. Thus, if any of those edge pixels is processed in the online phase of the conventional GHT each of the three associated displacement vectors $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$ is added and the corresponding three cells of the accumulator array receive a vote. Consequently, it would be impossible to narrow the accumulator array to the approximate zone. One possible solution is to check during the voting process whether the added displacement vectors fall in the approximate zone or not. This query and the summation of the vectors would take too much time to ensure real-time performance. Therefore, the opposite approach is taken: already in the offline phase the information about the rough location of the edge pixels relative to the reference point is calculated and stored within the model. This is done by overlaying a grid structure over the model image and splitting the image into tiles. In Figure 4.12(b) the tiles are displayed as squares. For each tile that is occupied by at least one edge pixel a separate R-table is generated (the squares with bold border in Figure 4.12(b)). The displacement vector $\boldsymbol{r}_i$ is then stored in the associated R-table. Thus, unnecessary voting steps in the online phase are already avoided in the preliminary stage of the offline phase. Analogously to the blurred regions, the tiles are computed for all pyramid levels except for the top pyramid level and for all discrete orientations. Accordingly, the final model that is used for object recognition consists of a multitude of R-tables: on the top pyramid level of the model image for each quantized orientation one R-table is built. Whereas, on the lower pyramid levels for each quantized orientation and for each occupied tile a separate R-table is created.



(a) Disregarding prior information

(b) Taking prior information into account

Figure 4.12: The GHT in its conventional form cannot take prior information about the transformation parameters into account. Therefore, many unnecessary voting steps are executed (a). By using a tile structure, unnecessary voting steps are avoided (b).

In the online phase, for the current edge pixel that is to be processed the associated tile is calculated using the relative position of the current edge pixel to the approximate position of the reference point. Finally, only the displacement vectors in the R-table of the respective tile (and with appropriate gradient direction) are used to calculate the cells that receive a vote. Besides the decreased computational load in the online phase the tiling further increases the robustness against clutter because unnecessary voting events are suppressed. In the current example (see Figure 4.2), a speed-up of 78% is achieved when using tiles of size 7 × 7 pixels. An even higher gain in efficiency can be expected for objects that show a high number of edge pixels with identical gradient direction.

Summarizing the most important points, the conventional GHT is not suited for the use of image pyramids in a straightforward way. However, by applying a multi-resolution model that includes the domain restriction with the blurred region and the avoidance of unnecessary voting steps with the tile structure, the advantages of image pyramids also become accessible to the GHT. The multi-resolution model reduces the memory requirements in the online phase drastically and facilitates real-time object recognition. Several tests that show the correctness of the latter statement will be described in the performance evaluation, which is presented in Section 4.4. Furthermore, another important advantage of the proposed multi-resolution model is the increased robustness against clutter.

4.2.4 Pose Refinement

In this section, a method for pose refinement that breaks the limits on the achievable accuracy that are induced by the quantization of the parameter space is described. After the voting process on the lowest pyramid level, matches that correspond to cells in the accumulator array and that are local maxima exceeding a certain threshold of votes are obtained. Therefore, the accuracy of the position parameters o_x^s and o_y^s is limited to the cell size used in the accumulator array on the lowest pyramid level, which in general corresponds to the pixel grid. Analogously, the accuracy of the orientation parameter φ^s is limited to the step size $\Delta\varphi$ that is used to rotate the model image on the lowest pyramid level. The proposed refinement method is based on the idea of correlation interpolation (Tian and Huhns 1986) that was introduced in Section 4.1.2. However, the proposed method is not restricted to refine the object position, as it is the case in (Tian and Huhns 1986), but is able to simultaneously refine position and orientation. Furthermore, not a correlation measure but the votes in the accumulator array are used for interpolation. To refine the position and orientation, the *facet model principle* (Haralick and Shapiro 1992) is applied. It states that an image can be thought of as a continuous gray level intensity surface, where the acquired digital image is a noisy sampling of a distorted version of this surface. To apply this principle to pose refinement, the 3D parameter space is assumed to be a 3D continuous intensity surface $f(x, y, \varphi)$, where the intensities f describe the likelihood that the object is present in the image at pose $(x, y, \varphi)^\top$. The accumulator array is the sampled version of f, in which the intensities correspond to the number of votes in the cells. To represent the discrete accumulator array by a continuous function, a second order polynomial is locally fitted to the accumulator array at the position of the discrete local maximum, which also defines the origin of the local coordinate system:

$$f(x, y, \varphi) = k_0 + k_1 x + k_2 y + k_3 \varphi + k_4 x^2 + k_5 xy + k_6 x\varphi + k_7 y^2 + k_8 y\varphi + k_9 \varphi^2 \quad . \tag{4.22}$$

The coefficients $k_0, \ldots, k_9$ can be determined efficiently using the 3D facet model masks presented in (Steger 1998), where for each coefficient a $3\times3\times3$ filter mask is designed. Thus, the votes in a $3\times3\times3$ neighborhood of the local maximum in A are used to estimate the ten parameters (see Figure 4.13(a)). The pose refinement can finally be obtained by analytically computing the maximum of the continuous function (4.22). For this, f is rewritten as:

$$f(x, y, \varphi) = (x, y, \varphi) \begin{pmatrix} k_4 & \frac{1}{2}k_5 & \frac{1}{2}k_6 \\ \frac{1}{2}k_5 & k_7 & \frac{1}{2}k_8 \\ \frac{1}{2}k_6 & \frac{1}{2}k_8 & k_9 \end{pmatrix} \begin{pmatrix} x \\ y \\ \varphi \end{pmatrix} + (x, y, \varphi) \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix} + k_0 \quad . \tag{4.23}$$

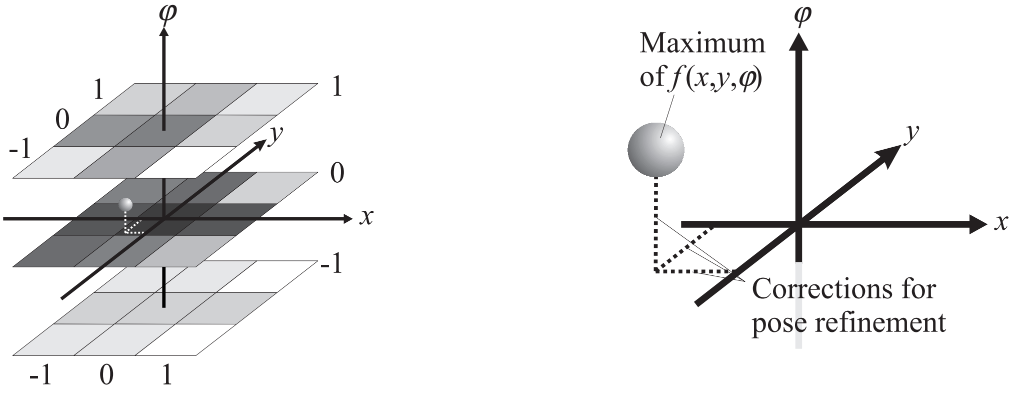
The maximum is defined as the point where the gradient of f vanishes:

$$\nabla f(x, y, \varphi) = \begin{pmatrix} 2k_4 & k_5 & k_6 \\ k_5 & 2k_7 & k_8 \\ k_6 & k_8 & 2k_9 \end{pmatrix} \begin{pmatrix} x \\ y \\ \varphi \end{pmatrix} + \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix} = \mathbf{0} \quad . \tag{4.24}$$

Finally, the subpixel precise position of the maximum in the local coordinate system is obtained by solving (4.24) with respect to (x, y, φ). This corresponds to the problem of solving a 3×3 linear equation system:

$$\begin{pmatrix} x \\ y \\ \varphi \end{pmatrix} = \begin{pmatrix} 2k_4 & k_5 & k_6 \\ k_5 & 2k_7 & k_8 \\ k_6 & k_8 & 2k_9 \end{pmatrix}^{-1} \begin{pmatrix} -k_1 \\ -k_2 \\ -k_3 \end{pmatrix} \quad . \tag{4.25}$$

The obtained values for (x, y, φ) should lie in the range of $[-\frac{1}{2}, +\frac{1}{2}] \times [-\frac{1}{2}, +\frac{1}{2}] \times [-\frac{1}{2}, +\frac{1}{2}]$ to be accepted as a meaningful result. The final refined pose is then obtained by adding the coordinates of the maximum in the local coordinate system to the discrete pose parameters (see Figure 4.13(b)). The reason for choosing a second order polynomial is that it represents the shape of the maximum in the accumulator array sufficiently enough. Furthermore, the existence of a unique extremum, in which one is interested in, is ensured. Because the above computations are only applied to local maxima in the discrete accumulator array, it is ensured that the fitted polynomial always exhibits a maximum rather than a minimum. Only the final matches on the lowest pyramid level need to be refined. This is the reason why the additional computation time for the pose refinement is almost negligible.



(a) $3 \times 3 \times 3$ sub-cube of A

(b) Extracted maximum

Figure 4.13: Applying a 3D facet model mask on the $3 \times 3 \times 3$ neighborhood of a local maximum in the discrete parameter space A (a). The object pose is refined by adding the coordinates of the maximum to the discrete pose parameters (b).

4.2.5 Quantization Effects

When applying the principle of the GHT several problems are caused by the quantization of the parameters in the accumulator array and of the gradient directions in the *R*-tables. A similar difficulty occurs when using the tile structure described in Section 4.2.3.3. In this section, these problems will be analyzed and the respective solutions will be proposed.

4.2.5.1 Rotation

Two contrary objectives must be balanced for the determination of the orientation step size $\Delta\varphi$ when rotating the model image during model creation. On the one hand, $\Delta\varphi$ should be chosen large to reduce the number of *R*-tables, and hence the computational load in the online phase. On the other hand, if $\Delta\varphi$ is chosen too large, objects that appear between two discrete orientation steps may be missed in the online phase. Informally speaking, the peak height in the parameter space must not drop down considerably if the object appears in the middle of two sampled orientation steps. The optimum value for $\Delta\varphi$ depends on the object, especially on its size. More precisely, $\Delta\varphi$ can be increased for decreasing object size. Formalizing the statements, in the optimum case all displacement vectors $\boldsymbol{r}_i$ of the model should hit the same cell of A in the online phase, independent of the object orientation. For this, the positions of the reference point candidates $\breve{\boldsymbol{o}}$, which are obtained by adding the displacement vectors, may only vary within the range of one cell. Assuming that one cell corresponds to one pixel, the maximum allowable distance ε of $\breve{\boldsymbol{o}}$ from the center of the cell is 1/2 pixel in x and y, respectively. The maximum allowable value for $\Delta\varphi$ can be written as a function of ε (see Figure 4.14):

$$\Delta\varphi \leq 2 \arcsin \frac{\varepsilon}{r^{max}} \approx 2 \frac{\varepsilon}{r^{max}} \quad , \tag{4.26}$$

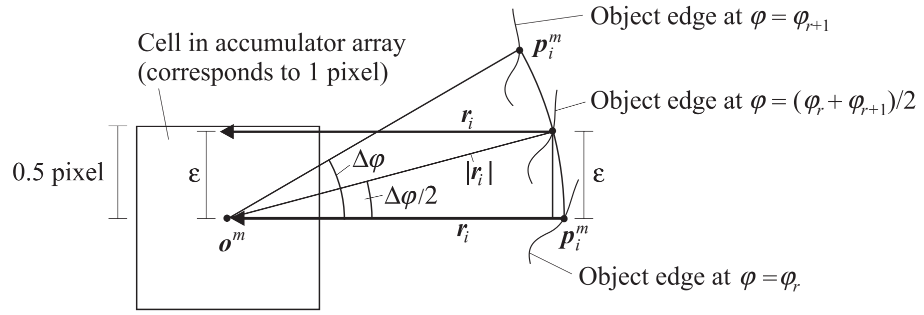
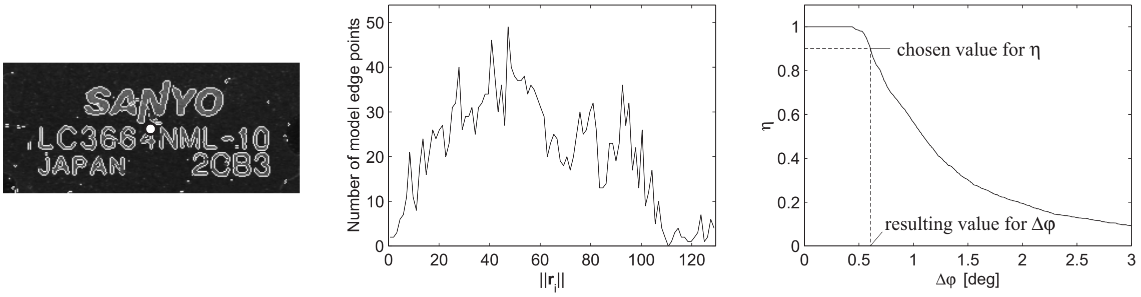

Object edge at φ = φ_{r+1}
Cell in accumulator array
(corresponds to 1 pixel)
p_i^m
Object edge at φ = (φ_r + φ_{r+1})/2
r_i
0.5 pixel
ε
Δφ
|r_i|
ε
Δφ/2
o^m
r_i
p_i^m
Object edge at φ = φ_r

Figure 4.14: The orientation step $\Delta\varphi$ depends on the distance of the model edge points $\boldsymbol{p}_i^m$ from the reference point $\boldsymbol{o}^m$.



(a) Model edges (b) Histogram of $\|\boldsymbol{r}_i\|$ (c) Weakening factor η

Figure 4.15: In (a), the model image, the reference point (white dot), and the extracted model edges (white) are shown. The optimum orientation step for an object is computed based on the histogram of the displacement vector lengths (b). For a chosen allowable weakening factor η the associated orientation step can be computed (c).

with

$$r^{max} = \max_{i=1,\dots,n^m} \|\boldsymbol{r}_i\| \quad . \tag{4.27}$$

Thus, r^{max} is the farthest distance of all model edge points to the reference point $\boldsymbol{o}^m$, which serves as the fixed point of the rotation.

The drawback of equation (4.26) is that the worst-case of only one isolated model edge point with a high distance to the reference point is already sufficient to cause a very fine quantization of the orientation. However, a coarser quantization would only decrease the peak height by one vote if the distance of the remaining edge pixels is significantly smaller. Because of the resulting large memory amounts and time consuming computations caused by a very fine quantization, a more tolerant computation of $\Delta\varphi$ is required. Here, the distances of all model edge pixels are taken into account instead of using only r^{max} as representative for the whole object. As previously mentioned, if $\Delta\varphi$ is chosen too large, the peak in the corresponding cell in A will be weakened because some of the reference point candidates will miss the cell. By solving (4.26) with respect to r^{max} one can compute for a given orientation step $\Delta\varphi$ the maximum allowable distance of the edge points from the reference point in order not to weaken the peak. Consequently, the weakening factor η can be formalized as the fraction of displacement vectors that have a length that does not exceed the maximum allowable distance:

$$\eta(\Delta\varphi) = \frac{\left|\left\{\boldsymbol{r}_i \,\middle|\, \|\boldsymbol{r}_i\| \le \frac{2\varepsilon}{\Delta\varphi}\right\}\right|}{n^m} \quad , \tag{4.28}$$

where $\varepsilon = 0.5$. The maximum possible peak height Γ^{max} that corresponds to the number of model points and that is achieved when (4.26) is fulfilled, will then be reduced to

$$\Gamma(\Delta\varphi) = \eta(\Delta\varphi) \cdot \Gamma^{max} \quad . \tag{4.29}$$

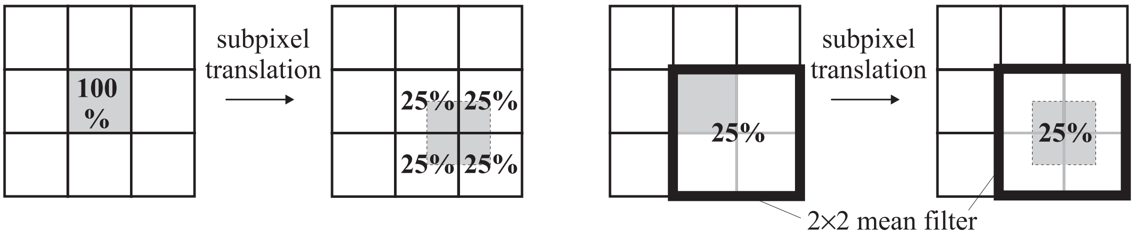
By evaluating $\eta(\Delta\varphi)$ or, optionally, $\Gamma(\Delta\varphi)$ for different values of the orientation step $\Delta\varphi$, a characteristic curve for the current object is obtained, as shown in Figure 4.15. One possibility to choose an appropriate value for $\Delta\varphi$ is to specify a minimum value for η or Γ that is still tolerable in the online phase. Because η is a fractional value, and hence is limited to the range of $[0, 1]$, it is more intuitive for the user to specify a value for η rather than for Γ. Setting η to 1 means that the same value for $\Delta\varphi$ as when using (4.26) is obtained. However, the use of (4.28) instead of (4.26) facilitates a more intuitive and flexible calculation of the optimum orientation step for a specific application.

The proposed computation is either executed for all pyramid levels separately, or as an approximation, only for the original resolution, where $\Delta\varphi$ is successively doubled for the next higher pyramid level.

4.2.5.2 Translation

Unfortunately, the height of the peak in A also depends on subpixel translations of the object. Under ideal conditions the peak in the parameter space is equal to n^m. If the object in the search image is translated by subpixel values in x and y relative to its position in the model image the peak height decreases because the votes are dispersed over more than one cell. This effect reaches its maximum at a subpixel translation of 1/2 pixel in each direction, which diminishes the peak to 25% of its original height. Figure 4.16(a) illustrates this behavior.

Under the assumption that the neighborhood of the peak is rotationally symmetric the peak height can be made independent of subpixel translations by smoothing the 2D hyperplanes of the accumulator array, i.e., $\varphi =$ const., by a mean filter of size 2×2. In Figure 4.16(b) the effect is illustrated. Although, in general, the number of votes per cell is reduced because of the smoothing, the peak height is independent of the subpixel translation of the object. However, the threshold for the extraction of local maxima must be adapted accordingly.



(a) Without mean filtering

(b) Applying a 2×2 mean filter

Figure 4.16: Subpixel translations of 1/2 pixel in both directions disperse the original peak (100%) over four adjacent cells, leading to a diminished peak height of 25% in each of the four cells (a). This effect is eliminated by applying a 2×2 mean filter to the votes in the 2D accumulator arrays, which results in peak heights that are independent of subpixel translations (b).

4.2.5.3 Gradient Direction

The optimum quantization of the gradient direction within the *R*-tables depends on various factors. The size $\Delta\theta$ of the gradient direction intervals defines the range of gradient directions that are treated as equal during the voting process. The smaller the interval, the faster the computation, because, on average, fewer displacement vectors are contained in the same row of the *R*-table. Hence, fewer voting events must be performed for a given gradient direction in the search image. However, an interval that is chosen too small leads to instable results. This problem will be discussed in the following. The discussion is based on the computation of the maximum gradient direction error that is expected to occur in the search image. From this error conclusions about the optimum value of $\Delta\theta$ can be drawn.

The first point to consider is the error of the gradient directions due to noise in the image. The gradient directions are computed from the first partial directional derivatives in x and y direction returned by the Sobel filter:

$$\theta = \arctan \frac{\partial I(x,y)/\partial y}{\partial I(x,y)/\partial x} \ , \tag{4.30}$$

where

$$\frac{\partial I(x,y)}{\partial x} =: I_x = \frac{1}{N^S} I(x,y) * \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \tag{4.31}$$

and

$$\frac{\partial I(x,y)}{\partial y} =: I_y = \frac{1}{N^S} I(x,y) * \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \ . \tag{4.32}$$

N^S is the normalization factor for the Sobel filter. In the scope of this dissertation N^S is set to 4, without any restrictions, in order to obtain intuitive edge magnitudes. The derivative in x, for example, is explicitly computed as follows (it is important to note that the filter masks are mirrored during convolution):

$$\begin{aligned} I_x \ = \ & \frac{1}{N^S}(-I(x-1,y+1) + I(x+1,y+1) - \\ & -2I(x-1,y) + 2I(x+1,y) - \\ & -I(x-1,y-1) + I(x+1,y-1)) \ . \end{aligned} \tag{4.33}$$

The derivative in y is computed accordingly. Now, assume that the gray values are independent from each other and show a constant standard deviation σ_I in the image. Then the standard deviation of I_x can be determined by applying the law of error propagation to (4.33). Finally, $\sigma^2_{I_x}$ and $\sigma^2_{I_y}$ are obtained as:

$$\sigma^2_{I_x} = \sigma^2_{I_y} = \frac{12}{(N^S)^2}\sigma^2_I \ . \tag{4.34}$$

In order to derive the standard deviation of the gradient direction θ, the partial derivatives of (4.30) with respect to I_x and I_y must be computed:

$$\frac{\partial \theta}{\partial I_x} = -\frac{I_y}{{I_x}^2 + {I_y}^2} \ , \quad \frac{\partial \theta}{\partial I_y} = \frac{I_x}{{I_x}^2 + {I_y}^2} \ . \tag{4.35}$$

Applying the law of error propagation to (4.30) results in:

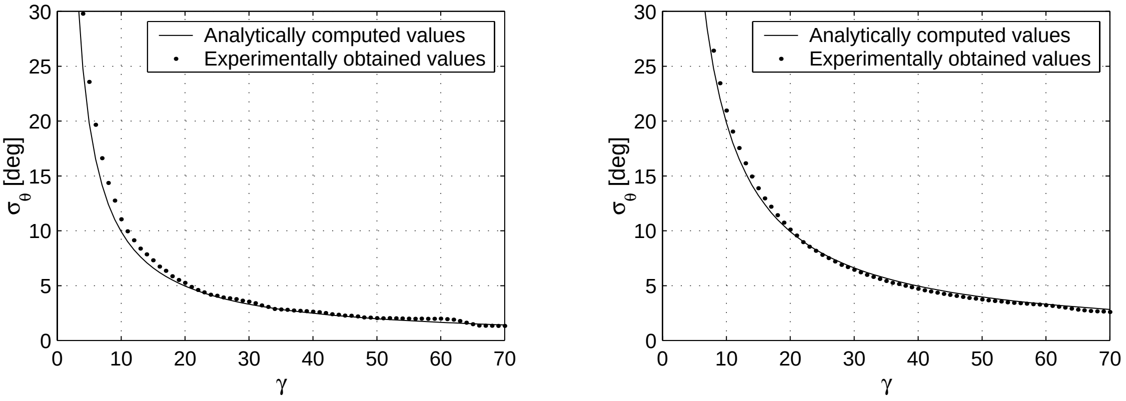
$$\sigma^2_\theta = \left(\frac{\partial \theta}{\partial I_x}\right)^2 \sigma^2_{I_x} + \left(\frac{\partial \theta}{\partial I_y}\right)^2 \sigma^2_{I_y} \ . \tag{4.36}$$

By plugging (4.34) and (4.35) into (4.36) and applying some simplification steps, finally, the standard deviation of the gradient direction is obtained:

$$\sigma_\theta = \frac{2\sqrt{3}}{N^S\sqrt{I_x^2 + I_y^2}}\,\sigma_I \ . \tag{4.37}$$

It is obvious that σ_θ increases with lower edge magnitudes γ ($= \sqrt{I_x^2 + I_y^2}$). Fortunately, during edge segmentation only pixels with an edge magnitude exceeding the threshold γ^{min} are selected for further processing. Therefore, an upper boundary for σ_θ can be computed (assuming $N^S = 4$):

$$\sigma_\theta \leq \frac{\sqrt{3}}{2\gamma^{min}}\,\sigma_I \ . \tag{4.38}$$



(a) $\sigma_I = 2.0$

(b) $\sigma_I = 4.0$

Figure 4.17: The standard deviation σ_θ of the gradient direction obtained from the Sobel filter depends on the edge magnitude γ. The experimentally obtained values show a very good correspondence with the values computed analytically from (4.38).

The standard deviation σ_I depends on the utilized camera and can be determined experimentally. For this, several images (e.g., 50) of the same scene under identical exterior circumstances are acquired. For each pixel the gray value standard deviation can be computed over the collection of images. An empirical value for σ_I can then be obtained by computing the average standard deviation over all pixels. This procedure needs to be executed only once for a specific camera (assuming a constant integration time) and therefore does not hamper the applicability of the proposed approach. Off-the-shelf cameras of higher quality show typical values for σ_I in the range of $[1.5, 2]$. The practical correctness of the analytically derived values for σ_θ was experimentally confirmed. An artificial image showing a gray value edge with a length of 1000 pixels was created. Random noise with Gaussian distribution was added to the image using σ_I as standard deviation. The gradient direction was computed using the Sobel filter and the standard deviation of the gradient direction was derived over the 1000 edge pixels. This procedure was repeated 100 times and a mean standard deviation was computed. The experiment was executed for edge magnitudes γ in the range of $[1, 70]$. Figure 4.17 displays the mean standard deviations obtained from the experiments for two selected values of σ_I in dependence on the edge magnitude. Additionally, the values obtained analytical from (4.38) are plotted. Evidently, theory and practice show a very good correspondence.

After having derived the standard deviation of the gradient direction, the maximum error induced by noise in the image can be determined. By assuming an approximately normal distribution of the computed gradient direction, one can specify the desired percentage of gradient directions that should lie within a certain interval. For example, if at least 95 percent should fall inside the interval then its boundaries are $[-2\sigma_\theta, +2\sigma_\theta]$, i.e., the maximum gradient direction error ξ^n induced by noise in the image is

$$\xi^n = 2\sigma_\theta \ . \tag{4.39}$$

Practical experiments have confirmed that assuming a confidence interval of 95 percent was sufficient for all tested examples.

The second influence that must be taken into account is the inherent absolute accuracy of the Sobel filter, i.e., the difference between the real partial derivatives and the Sobel response. Since the Sobel filter is an anisotropic filter, its absolute accuracy depends on the current gradient direction. The anisotropy error is eliminated to a high degree by rotating the model image instead of the displacement vectors when deriving the *R*-tables for the different object orientations, as proposed in Section 4.2.2. However, the anisotropy error within the range of one orientation step $\Delta\varphi$ still remains. The error of the gradient direction e_θ depends on the frequency in the image and on the actual gradient direction itself, and can be computed with the following formula (Jähne 2002):

$$e_\theta(\bar{f},\theta) \approx -\frac{(\pi\bar{f})^2}{48}\sin 4\theta + O(\bar{f}^4) \ , \tag{4.40}$$

where $\bar{f} \in \,]-1,1[$ is the frequency normalized to the maximum frequency that can be sampled (Nyquist frequency). Therefore, errors with highest magnitude of about 11.8 degrees are obtained. This is the case when applying the Sobel filter to image structures with frequencies close to the Nyquist frequency, i.e., $\bar{f} \rightarrow 1$, and with gradient directions of $\theta = \frac{\pi}{4}(z+\frac{1}{2})$, $z \in \mathbb{Z}$. Although the case $\bar{f} = 1$ cannot occur in practice because the Sobel filter would return zero edge magnitude, the assumption $\bar{f} \rightarrow 1$ is reasonable in order to represent the asymptotically worst-case scenario. The anisotropy errors are eliminated when calculating the *R*-tables in the proposed way. Thus, if the object appears in the search image in exactly the same orientation from which an *R*-table was computed, using a sampled orientation φ_r, the errors cancel out. In general, objects do not care about the sampled orientations but appear in arbitrary orientations. Consequently, each sampled orientation φ_r must represent a range of object orientations $[\varphi_r - \frac{\Delta\varphi}{2}, \varphi_r + \frac{\Delta\varphi}{2}]$. Therefore, it is of interest, how much the error of the gradient direction changes within this range of orientations. For this, (4.40) is differentiated with respect to θ. Ignoring higher order terms and assuming the worst-case by setting $\bar{f}$ to 1 one obtains:

$$e'_\theta(\theta) = \frac{\pi^2}{12}\cos 4\theta \ . \tag{4.41}$$

Accordingly, the maximum change of the gradient error is $\frac{\pi^2}{12}$, obtained at $\theta = z\frac{\pi}{4}$, $z \in \mathbb{Z}$. Assuming an orientation step of $\Delta\varphi$, the maximum change of the anisotropy error ξ^a that may occur within one range of orientations $[\varphi_r - \frac{\Delta\varphi}{2}, \varphi_r + \frac{\Delta\varphi}{2}]$ with respect to the reference orientation φ_r is

$$\xi^a = \frac{\Delta\varphi}{2}\max_\theta(e'_\theta(\theta)) = \Delta\varphi\frac{\pi^2}{24} \ . \tag{4.42}$$

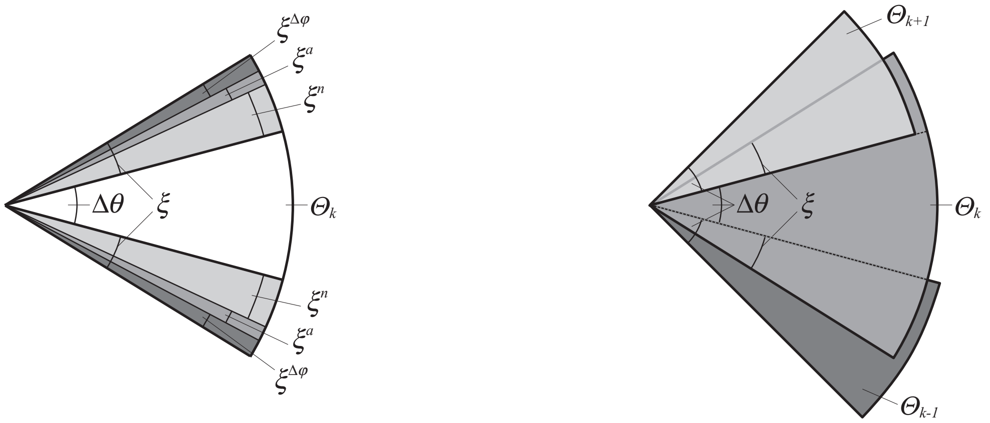
The third and most evident influence on the gradient direction directly arises from the orientation step $\Delta\varphi$ itself. Assume that an *R*-table was generated at the discrete orientation φ_r. If the object appears at orientation $\varphi_r \pm \frac{\Delta\varphi}{2}$ in the search image all gradient directions at corresponding edge points also change by the same value $\pm\frac{\Delta\varphi}{2}$ in comparison to the gradient directions that are stored in the *R*-table. Therefore, the maximum error on the gradient direction that is caused by the quantization of the object orientation is given by

$$\xi^{\Delta\varphi} = \frac{\Delta\varphi}{2} \ . \tag{4.43}$$

Finally, the resulting maximum error ξ of the gradient direction is the sum of all single errors (see Figure 4.18(a)):

$$\begin{aligned} \xi &= \xi^n + \xi^a + \xi^{\Delta\varphi} \\ &= 2\sigma_\theta + \left(\frac{1}{2}+\frac{\pi^2}{24}\right)\Delta\varphi \approx 2\sigma_\theta + 0.91\Delta\varphi \ . \end{aligned} \tag{4.44}$$

Imagine that in the offline phase the displacement vector $\boldsymbol{r}_i$ is stored in row k of the *R*-table, and hence the associated gradient direction θ_i^m is in the interval Θ_k. Furthermore, assume that θ_i^m exactly lies at the upper boundary of the interval Θ_k. Because of the error of the gradient directions, in the online phase the gradient direction at the corresponding edge point in the search image may change to $\theta_i^m + \xi$ in the worst-case and therefore leave the interval Θ_k. Consequently, in the conventional GHT the displacement vectors of the wrong row in the *R*-table are used to increment the cells. In contrast, the correct displacement vector $\boldsymbol{r}_i$ would remain unconsidered. Thus, whenever a certain gradient direction occurs in the search image, it is not known whether it is distorted by an error or not. This means that it is impossible to reliably compute the correct row of the *R*-table. The optimum solution to this problem would be to consider all those displacement vectors for voting where the associated gradient direction at most differs by $\pm\xi$ from the computed gradient direction in the search image. Unfortunately, this solution would slow down the computation in the online phase considerably because additional comparisons would have to be performed. A more efficient solution is to generate overlapping gradient direction intervals, as shown in Figure 4.18(b). For a correct computation, the overlap size must be chosen to be ξ in both directions of the interval. This ensures that in spite of potential errors in the gradient direction the right displacement vectors are chosen. To realize the overlap, the displacement vector with gradient direction θ_i^m that is within the interval Θ_k, and hence stored in row k of the *R*-table, is additionally stored in neighboring rows of the table. The neighboring rows are chosen so that they completely cover all possible gradient directions, which are given by the interval $\theta \in [\theta_i^m - \xi, \theta_i^m + \xi]$. This ensures that in the online phase the computed *R*-table row contains the correct displacement vector with a very high probability (95%).



(a) Gradient direction error

(b) Overlapping intervals

Figure 4.18: The associated gradient directions of the displacement vectors that are stored in interval Θ_k of the *R*-table may be distorted by the maximum error ξ in the search image and therefore exceed the border of the interval (a). This can be avoided using overlapping intervals, where the correct overlap is ξ in both directions (b).

Once the overlap size has been correctly computed, the interval size $\Delta\theta$ itself can be chosen arbitrarily without risking the loss of any displacement vectors in the online phase. However, the computation time of the online phase directly depends on $\Delta\theta$. The smaller the interval size, the faster the computation is. For the reason of simplifying further considerations, in the following the tile structure will be disregarded without any restrictions on the generality. Then, the number of voting events that must be performed on average for one edge pixel in the search image (for one specific object orientation, i.e., for one specific *R*-table) can be quantified (cf. (4.16)):

$$n^{vote} = n^m \frac{\Delta\theta}{2\pi}\left(1 + \frac{2\xi}{\Delta\theta}\right) = n^m \frac{\Delta\theta + 2\xi}{2\pi} \quad . \tag{4.45}$$

From this, it is clear that the number of voting events, and hence the computation time in the online phase, increases linearly with $\Delta\theta$, reaching its minimum for $\Delta\theta = 0$. Unfortunately, the memory requirement for one *R*-table increases with decreasing $\Delta\theta$. Since n^{vote} in (4.45) represents the average number of displacement vectors that are stored in one row of the *R*-table, and since there are $\frac{2\pi}{\Delta\theta}$ rows altogether, the number of (multiply) stored displacement vectors n^r in one single *R*-table is

$$n^r = n^{vote}\frac{2\pi}{\Delta\theta} = n^m\left(1 + \frac{2\xi}{\Delta\theta}\right) \quad . \tag{4.46}$$

It is essential to note that n^r is proportional to the memory that is needed to store one *R*-table. This means that there is a trade-off between computation time (4.45) and memory requirement (4.46) when choosing an appropriate value for $\Delta\theta$. Setting $\Delta\theta = 2\xi$ is an empirically determined suitable compromise. This means that each displacement vector is stored twice in each *R*-table.

Another factor that affects the gradient direction is subpixel translations. Taking the edges as 2D curves in the image, the magnitude of the gradient direction variation that is caused by subpixel translation mainly depends on the curvature of the edges. In Figure 4.19 an example of subpixel translation in the y direction shows this effect. Here, the gradients of the corner pixel and the pixel below significantly change because of the translation. One possible solution for this problem is to introduce only “stable” edge points into the model. I.e., those pixels are introduced whose gradient directions at most vary in a small range. The stable points can be found by translating the model image by 1/2 pixel in each direction and comparing the computed gradient directions θ^t with those of the untranslated image θ^0. The edge pixels with small differences, i.e., $|\theta^0 - \theta^t| \leq \xi$, form the model. This ensures that the errors that are induced by subpixel translations are already covered by the appropriately chosen overlap size of the gradient direction intervals. All other edge pixels are disregarded when computing the *R*-tables. A more pragmatic approach of finding the stable points is to directly threshold the curvature of the edge pixels. However, a suitable value for the threshold is difficult to find. It should be noted that if the fraction of edge pixels that are to be eliminated by the above criterion is too high then the threshold for the maximum differences should be relaxed. This is important in order to be still able to handle arbitrary objects (especially objects that exhibit high curvature in most edge pixels).

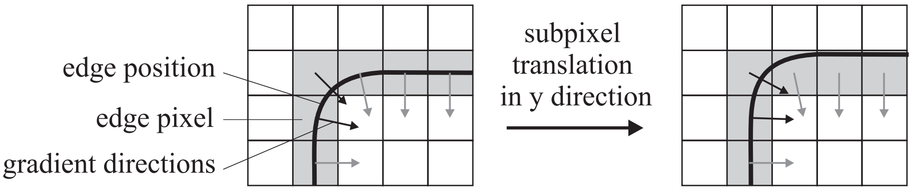
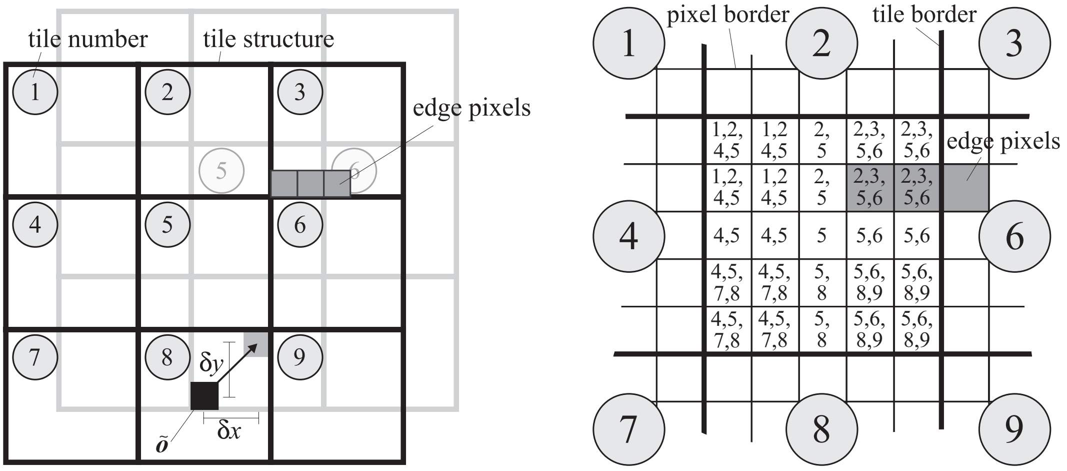


Figure 4.19: Subpixel translations affect the gradient directions, particularly in regions with high edge curvature.

4.2.5.4 Tile Structure

A problem similar to the quantization of the gradient directions occurs when using the tile structure described in Section 4.2.3.3. To simplify further considerations, the size of the tiles should be chosen so that the uncertainty of the approximate position is taken into account. I.e., the dimension of the tiles in the x and y direction should be $2\delta x$ and $2\delta y$. Furthermore, it must be ensured that an error of δx and δy of the approximate position $\tilde{\boldsymbol{o}}$ does not result in omitting the relevant edge pixels as a consequence of considering the wrong tile. This problem is illustrated in Figure 4.20(a). In this example, the model consists of three edge pixels that are stored in tile 3 within the model (for illustration purposes only, the reference point in this example differs from the centroid). However, in the online phase, the approximate position of $\tilde{\boldsymbol{o}}$ may vary within the range of δx and δy. Now, assume that the approximate position is not computed at its true location but displaced by $+\delta x$ and $+\delta y$. To compute the respective tile numbers for the voting process, the tile structure is then centered at the displaced approximate position, leading to the fact that the calculated tile number for the three edge pixels would now be 5 and 6 instead of 3. However, in these two tiles no displacement vectors are stored, and thus no voting event would be executed. Consequently, the match candidate would be deleted.

The solution is illustrated in Figure 4.20(b). To avoid omitting relevant tiles (tile 3 in this example), certain neighboring tiles of the calculated tile must be considered additionally during voting. Since not all of the neighbors need to be taken into account a look-up table that holds the relevant tiles to be checked is constructed. For example, the two edge pixels on the left of Figure 4.20(a) fall into tile 5. It is easy to see that edge pixels occurring at this position may only belong to tiles 2, 3, 5, or 6. Therefore, these four tiles must be taken into account during the voting process. The associated tiles can be calculated for each edge pixel in each tile in the offline phase and are stored together with the tile structure. This look-up table is computed for all lower pyramid levels. Finally, in the online phase, for each edge pixel on lower pyramid levels the corresponding tiles are investigated by just reading the entry in the look-up table. This facilitates a fast computation while keeping the memory requirement low.



1,2, 4,5	1,2 4,5	2, 5	2,3, 5,6	2,3, 5,6
1,2, 4,5	1,2 4,5	2, 5	2,3, 5,6	2,3, 5,6
4,5	4,5	5	5,6	5,6
4,5, 7,8	4,5, 7,8	5, 8	5,6, 8,9	5,6, 8,9
4,5, 7,8	4,5, 7,8	5, 8	5,6, 8,9	5,6, 8,9

(a) Edge pixels changing their tiles

(b) Tile number look-up table

Figure 4.20: In (a) the three edge pixels are originally contained in tile 3. Because of the maximum error δx and δy of the approximate position $\tilde{o}$, the edge pixels might move to neighboring tiles. With this, the tiles 5 and 6 are occupied and should be taken into account during voting. The problem can be solved using overlapping tiles, which are realized by creating a look-up table, where for each pixel in each tile the tile numbers that must be taken into account during voting are stored (b).

4.2.6 Implementation Details

To complete the description of the MGHT, some remarks that mainly concern the practical implementation of the proposed approach will be made. At first, some additional points that have not been explained so far will be discussed. Afterwards, a short summary of all necessary and possible user interactions provided by the implementation is given.

The first point to discuss is the threshold for the number of votes that a local maximum must achieve in the accumulator array in order to be treated as a match candidate. This threshold strongly depends on the maximum degree of occlusion that may occur in the search image because the number of votes decreases proportionally to the percentage of object occlusion. Therefore, the user must specify this threshold in order to balance between computation time and allowable occlusions. If the threshold is set to a low value, then on the one hand objects that are occluded to a high degree can be recognized, but on the other hand in general more match candidates must be tracked through the pyramid. The most intuitive way for the user to determine the threshold value is to specify the ratio $s^{min} \in [0, 1]$ to which the object must be visible in the image. With this, the number of votes can be transformed into a score value s that reflects the visibility of the object. Unfortunately, because the accumulator array is smoothed after the voting process, the values of the cells do not represent the number of votes any more. Consequently, the threshold cannot be computed by simply multiplying the number of model edge pixels (or twice the number if $\Delta\theta = 2\xi$) with s^{min}. Therefore, a method to experimentally specify the peak height in the smoothed accumulator array is applied: already in the offline phase the object is recognized in the model image and the smoothed peak height is stored within the model as the reference peak height. In the online phase, the smoothed values of the accumulator array are then normalized by the reference peak height, yielding the score value s. This value can be directly compared to the user-specified threshold s^{min}. It should be noted that s may exceed the value of 1 in cases of heavy clutter in the image since randomly distributed votes that are caused by the clutter may falsely increase the actual peaks of the match candidates. Furthermore, on higher pyramid levels the score value of the match candidates may decrease although the object is completely visible. This effect can be attributed to the fact that image pyramids are not invariant to translations (Lindeberg 1994). Hence, the user-specified threshold s^{min} must be slightly reduced on higher pyramid levels in order to avoid that matches are missed.

Because a breadth-first strategy is applied, the computation time when searching for several instances simultaneously does not dramatically increase in contrast to searching only for one single instance. Consequently, a second point that should be remarked concerns the number of matches that should be found during object recognition. Here, the user can choose between two options. The first option is to return all matches that have a score exceeding the specified threshold s^{min}. The second option allows the user to introduce additional information about the maximum number n^{match} of objects instances that may appear in the search image. This information can be used in the breadth-first search to eliminate a high number of match candidates during the tracking through the pyramid. This results in a high gain of computational efficiency. One way to eliminate match candidates is to count the number of candidates that exceed the threshold s^{min} on the current pyramid level. If this number is higher than the user-defined maximum number of object instances, extra match candidates with lower score values are excluded from further processing. The score values of the candidates do not behave in a predictable manner during the tracking through the pyramid. Thus, a candidate with lower score on a higher pyramid level can turn into a candidate with higher score on a lower pyramid level. For this reason, a more tolerant heuristic is applied. A match candidate is only eliminated if it additionally fulfills the condition that its score is significantly lower than the lowest score of non-eliminated match candidates. Finally, on the lowest pyramid level only the n^{match} best match candidates that additionally fulfill the requirement for the minimum score are returned as matches.

The last point to mention is the mutual overlapping of matches. In practice, sometimes one instance of an object in the search image returns more than one match. For example, when dealing with objects that exhibit symmetries, several matches are returned at similar positions, but at different orientations. In order to avoid this behavior, the user can specify a fraction by which two returned matches are allowed to overlap each other at most. The actual overlap fraction between two matches is approximated by the overlap between the two smallest enclosing rectangles of the two objects at the pose of the respective matches. If the computed overlap fraction exceeds the user-specified maximum overlap, the match with lower score value is eliminated.

In the following, all necessary and possible user interactions or parameter settings of the MGHT are summarized.

- Input data passed to the offline phase (model generation):
  - The model image in which the object is defined by an arbitrary region of interest *must* be provided. Usually, the ROI is specified by the user. Another possibility would be to generate the ROI automatically using suitable segmentation techniques.
  - The user *must* specify the threshold for the minimum edge magnitude γ^{min} that is used for edge segmentation. As previously mentioned, this value depends on the application and cannot be determined automatically.
  - The camera-specific noise σ_I of the gray values in the image *must* also be provided in order to automatically compute the optimum quantization of the gradient direction intervals used in the *R*-tables.
  - Optionally, the domain of the transformation class *may* be restricted by the user, i.e., the range of possible object orientations can be restricted to a certain interval to avoid unnecessary *R*-table generations. Thus, the memory requirement of the model and the computational effort in the online phase is reduced.
  - In general, the automatically determined values for the reference point $\boldsymbol{o}^m$, the number of pyramid levels n^l, the orientation step $\Delta\varphi$, and the size of the gradient quantization intervals $\Delta\theta$ are suitable for most applications. Nevertheless, the user *may* optionally overwrite these values in order to receive a higher flexibility.
- Input data passed to the online phase (object recognition):
  - The search image *must* be provided, where an arbitrary ROI *may* restrict the image domain for edge extraction. Thus, the user may introduce additional prior knowledge about the object position as

well as the object orientation, which can be specified during model generation, to speed up the recognition.

- The minimum score s^{min} *must* be provided to specify the minimum peak height for match candidates.
- Also in the search image edge extraction is performed. Therefore, the minimum edge magnitude *can* be optionally specified if it differs from the value set in the online phase. It is important to note that the automatic computation of $\Delta\theta$ is based on γ^{min}, which was specified in the offline phase.
- The maximum number of matches n^{match} *can* be specified in order to speed up the computation.
- The maximum allowable overlap between different matches *can* be specified to eliminate multiply found object instances.

Finally, for each match the object pose and the associated score value is returned. The pose is represented by the object position o_x^s, o_y^s, and the object orientation φ^s, where φ^s is 0 if the orientation of the object in the search image is identical to its orientation in the model image.

4.2.7 Conclusions

To give an idea of the improvement of the MGHT in comparison to the conventional GHT, the memory requirement and the computational complexity of both methods are computed for a typical practical example. For this, a search image size of 600 $\times$ 600 pixels, a tile size of 7 $\times$ 7 pixels, 3000 model points at the original resolution, and four image pyramid levels are assumed. The result is summarized in Table 4.3. For the MGHT the memory requirement of the model increases from 4.3 MB to 30.8 MB in comparison to the conventional GHT, which is mainly caused by the tile structure. However, in the online phase, the memory requirement shrinks from 480 MB to 0.5 MB because of the use of image pyramids. Since image pyramids are used and the computation time is optimized by the proposed approach the computational complexity of the online phase is reduced considerably: theoretically, the MGHT is about 16.000 times faster than the conventional GHT in this example. However, in practice the recognition time is reduced from 45.60 s to 0.07 s on a 400 MHz Pentium II, which corresponds to a factor of still 650. The difference to the theoretical value can be explained by the increased memory access within the MGHT in contrast to the conventional GHT. Nevertheless, a tremendous improvement is still observable.

	GHT	MGHT
Memory Requirement [MB] (Offline)	4.3	30.8
Memory Requirement [MB] (Online)	480	0.5
Computation Time [s]	45.60	0.07

Table 4.3: Improvement of the modified generalized Hough transform (MGHT) in comparison to the conventional generalized Hough transform (GHT) according to memory requirement and computation time

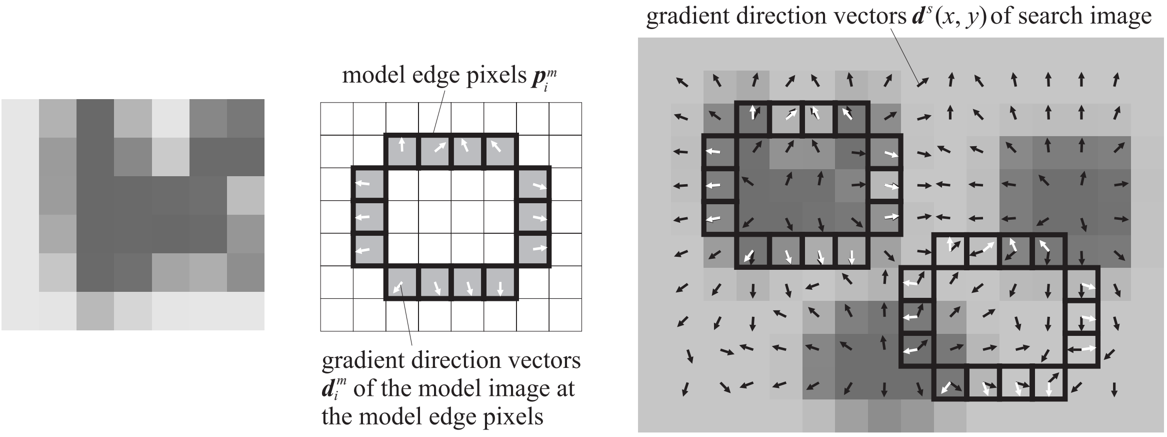
In summary, the MGHT that has been presented in this section eliminates the major drawbacks of the conventional GHT while keeping its inherent advantages and fulfills the stated requirements concerning the recognition of rigid objects listed in Section 2.2. A comprehensive performance evaluation, including a comparison with other rigid object recognition methods, will be presented in Section 4.4 in order to support the theoretically derived improvements. Although the descriptions in this section have been restricted to deal with rigid motion as the transformation class, the MGHT can be easily extended in a straight forward way to be able to cope with more general transformations, like similarity or affine transformations. Therefore, also applications that require parameter spaces of higher dimensions can benefit from the proposed modifications.

4.3 Shape-Based Matching

As mentioned in Section 2.4, simultaneously to the development of the MGHT, a new similarity measure has been developed. This similarity measure is already included in the HALCON library and is referred to as *shape-based matching* (SBM). In this section, the new similarity measure, an improved method for pose refinement, and some implementation details are described. Further details can be found in (Steger 2001, Ulrich and Steger 2002, Steger 2002).

4.3.1 Similarity Measure

The model of the SBM consists of a set of model points $\boldsymbol{p}_i^m = (x_i^m, y_i^m)^\top$ as in the case of the MGHT. Additionally, the model is complemented by the associated gradient direction vectors $\boldsymbol{d}_i^m = (t_i, u_i)^\top$, $i = 1, \dots, n^m$, where $t_i = I_x^m \big|_{\boldsymbol{p}_i^m}$ and $u_i = I_y^m \big|_{\boldsymbol{p}_i^m}$. Figure 4.21 illustrates the principle of this similarity measure by means of the example that has already been introduced in Section 4.2.1. Here, the third pyramid level is displayed (see Figures 4.2, 4.7(a), and 4.7(b)). In the offline phase, edges are extracted from the model image, which is shown in Figure 4.21(a), and their associated gradient direction vectors are computed. The resulting edges and the normalized gradient direction vectors (white arrows) are shown in Figure 4.21(b).


gradient direction vectors $\boldsymbol{d}^s(x,y)$ of search image
model edge pixels $\boldsymbol{p}_i^m$
gradient direction vectors $\boldsymbol{d}_i^m$ of the model image at the model edge pixels

(a) Model image (b) Model edges (c) Search image

Figure 4.21: In the shape-based matching edges are extracted and the associated gradient direction vectors are computed (b) from the model image (a). The search image is edge filtered, resulting in a gradient direction vector at each model edge pixel (c).

The search image, which is shown in Figure 4.21(c), can be transformed into a representation in which a gradient direction vector $\boldsymbol{d}^s(x,y) = (v(x,y), w(x,y))^\top$ is obtained for each image point $(x,y)^\top$. In the matching process, the transformed model must be compared to the search image at a particular location using a certain similarity measure. The basic idea behind the similarity measure within the SBM is to sum the normalized dot products of the gradient direction vectors of the transformed model and the search image over all model points. From this, a matching score s is obtained at each point $(x,y)^\top$ in the search image. In Figure 4.21(c) the normalized gradient direction vectors are displayed as black arrows at each pixel in the search image. A threshold on the similarity measure has to be specified by the user. For this, a similarity measure with a well-defined range of values is desirable. The following similarity measure achieves this goal:

$$\begin{aligned} s(x,y) &= \frac{1}{n^m} \sum_{i=1}^{n^m} \frac{\langle \boldsymbol{d}_i^m, \boldsymbol{d}^s(x+x_i^m, y+y_i^m) \rangle}{\|\boldsymbol{d}_i^m\| \cdot \|\boldsymbol{d}^s(x+x_i^m, y+y_i^m)\|} \qquad (4.47) \\ &= \frac{1}{n^m} \sum_{i=1}^{n^m} \frac{t_i v(x+x_i^m, y+y_i^m) + u_i w(x+x_i^m, y+y_i^m)}{\sqrt{t_i^2+u_i^2} \cdot \sqrt{v(x+x_i^m, y+y_i^m)^2 + w(x+x_i^m, y+y_i^m)^2}} \ , \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ represents the dot product between two vectors. Therefore, $s(x, y)$ represents the averaged cosine of the angle differences between the gradient directions of the transformed model points and the respective gradient directions in the search image. The cosine takes into account that the gradient direction is influenced by image noise. The effect of the resulting small angle differences on the similarity measure is reduced by the low sensitivity of the cosine to small arguments, which yields a high robustness against noise. Figure 4.21(c) also shows the calculation of the matching score $s(x, y)$ for two example positions of the object in the search image. At the true position (upper left) the angle differences between the direction vectors are small, and thus the cosine will be close to 1 at each model edge pixel. The reason for the small differences can be attributed to artificially added noise in the search image. At the second example position (lower right) the corresponding gradient directions show significantly higher differences, leading to cosine values that are much smaller than 1.

This similarity measure shows several inherent advantages. Because of the normalization of the gradient direction vectors, the similarity measure is invariant to arbitrary changes in brightness since all vectors are scaled to length one. It is essential to note that edge filtering is applied to the search image, but no segmentation is performed, i.e., no threshold on the edge magnitude is applied. This has the advantage that the similarity measure not only exhibits robustness but true invariance against changes in brightness of an arbitrary type: there is no risk that any edge pixel falls below an edge threshold if the contrast is too low. This constitutes the major advantage in comparison to all the object recognition methods that rely on segmented features in the search image, like, for instance, the Hausdorff distance or the GHT. Furthermore, this measure is robust against occlusion and clutter because in case of missing features, either in the model or in the search image, noise will lead to random gradient direction vectors, which, on average, will contribute nothing to the sum (see (Steger 2002) for further discussions concerning this topic).

Focusing on (4.47) again, the similarity measure will return a high score if all the direction vectors of the model and the search image align, i.e., have the same direction. A score of 1 indicates a perfect match between the transformed model and the search image. Furthermore, the score roughly corresponds to the portion of the object that is visible in the search image.

4.3.2 Implementation Details

An important feature of this similarity measure is that it does not need to be evaluated completely when object recognition is based upon a threshold s^{min} for the similarity measure. Let s_j denote the partial sum of the dot products up to the j-th model point:

$$s_j(x, y) = \frac{1}{n^m} \sum_{i=1}^{j} \frac{\langle \boldsymbol{d}_i^m, \boldsymbol{d}^s(x + x_i^m, y + y_i^m) \rangle}{\|\boldsymbol{d}_i^m\| \cdot \|\boldsymbol{d}^s(x + x_i^m, y + y_i^m)\|} \quad . \tag{4.48}$$

Obviously, all the remaining terms of the sum are all smaller or equal to 1. Therefore, the partial score can never reach the required score s^{min} if $s_j < s^{min} - 1 + j/n^m$, and hence the evaluation of the sum can be discontinued after the j-th element whenever this condition is fulfilled. This criterion speeds up the recognition process considerably.

Nevertheless, further speed-ups are highly desirable. Another criterion is to set the condition that at each partial sum the score is better than s^{min}, i.e., $s_j \geq s^{min} j/n^m$. When this criterion is used, the search will be very fast, but it can no longer be ensured that the object recognition finds the correct instances of the object. This is because if missing parts of the object are checked first, then the partial score will be below the required score. To speed up the recognition process with a very low probability of missing the object, the following heuristic can be used: the first part of the model points is examined with a relatively safe stopping criterion, while the remaining part of the model points is examined with the hard threshold $s^{min} j/n^m$. The user can specify a parameter g, which represents the fraction of the model points that are examined with the hard threshold. This parameter will be referred to as *greediness* below. If $g = 1$, all points are examined with the hard threshold, while for $g = 0$, all points are examined with the safe stopping criterion. With this, the evaluation of the partial sums is stopped whenever $s_j < \min(s^{min} - 1 + \zeta j/n^m, s^{min} j/n^m)$, where $\zeta = (1 - gs^{min})/(1 - g)$. The term greediness is chosen because it appropriately reflects the intention of the parameter: if g is set to 1 the search is very greedy, i.e., the search strongly tends to early abort the evaluation of less promising transformations in order to reduce the computation time. However, the risk of missing a match is high.

To further speed up the recognition process, the model is generated in multiple resolution levels, which are constructed by generating an image pyramid from the original image — in the same way as in the MGHT. Because of runtime considerations, again the Sobel filter is used for feature extraction. To take rigid transformations into account the model is also created for different object orientations. For this, $2/r^{max}$ has proven to be an appropriate value for the orientation step size $\Delta\varphi$ (cf. Section 4.2.5.1). In the online phase, an image pyramid is constructed for the search image. To identify potential matches, an exhaustive search is performed for the top level of the pyramid. With the termination criteria using the threshold s^{min}, this seemingly brute-force strategy actually becomes extremely efficient. After the potential matches have been identified, they are tracked through the resolution hierarchy until they are found at the lowest level of the image pyramid.

Furthermore, the same method for pose refinement that is applied to the MGHT can be used in the SBM (cf. Section 4.2.4): once the object has been recognized on the lowest level of the image pyramid, its pose parameters position and orientation are extracted to a resolution better than the discretization of the search space by fitting again a second order polynomial (in the three pose variables) to the similarity measure values in a $3 \times 3 \times 3$ neighborhood around the maximum score.

4.3.3 Least-Squares Pose Refinement

If an even higher accuracy is desirable than the accuracy that can be achieved by using the pose refinement described in Section 4.2.4, a least-squares adjustment can be applied. Here, the sum of squared distances between the transformed model points and the points in the search image is minimized. For this, it is necessary to extract the model points as well as the corresponding points in the search image with subpixel accuracy. However, an algorithm that performs a least-squares adjustment based on closest point distances would not improve the accuracy much since the points would still have an average distance significantly larger than 0 tangentially because the model and search points are not necessarily samples at the same points and distances. Therefore, for each transformed model point the closest point in the search image is found and the squared distance of that point to a line defined by the corresponding model point and the corresponding tangent to the model point is minimized. I.e., the directions of the model points are taken to be correct and are assumed to describe the direction of the object's border. The tangents are defined by the edge points $\boldsymbol{p}_i^m = (x_i^m, y_i^m)^\top$ in the model image and the perpendicular to the corresponding normalized gradient direction vectors $\boldsymbol{d}_i^m/\|\boldsymbol{d}_i^m\| = (\bar{t}_i, \bar{u}_i)^\top$. Hence, the tangents are given by the points $(x, y)^\top$ that satisfy $\bar{t}_i(x - x_i^m) + \bar{u}_i(y - y_i^m) = 0$. The Hessian normal form is used to compute the distance between each tangent and the potential corresponding edge point $\boldsymbol{p}_j^s = (x_j^s, y_j^s)^\top$ in the search image. Finally, the sum of squared distances is minimized:

$$d^2(\boldsymbol{a}) = \sum_{i=1}^{n^m} [\bar{t}_i(x_j^s(\boldsymbol{a}) - x_i^m) + \bar{u}_i(y_j^s(\boldsymbol{a}) - y_i^m)]^2 \rightarrow \min \quad . \tag{4.49}$$

Here, $\boldsymbol{a}$ represents the vector of the unknown (inverse) pose parameters and $x_j^s(\boldsymbol{a})$, $y_j^s(\boldsymbol{a})$ the transformed corresponding edge point of the search image. It should be noted that instead of transforming the tangents at the model points by the pose parameters, the points in the search image are transformed back using the inverse pose parameters $\boldsymbol{a}$ in order to simplify the calculations.

The potential corresponding edge points $\boldsymbol{p}_j^s$ in the search image are obtained by a non-maximum suppression of the edge magnitude in gradient direction and are further extrapolated to subpixel accuracy (Steger 2000). Thus, a segmentation of the search image is further avoided in order to maintain invariance against changes in brightness. The model points are transformed according to the pose parameters that have been obtained from the polynomial fitting (cf. Section 4.2.4). Because the points in the search image are not segmented and correspondence is based on smallest euclidian distance, spurious points in the search image may be brought into correspondence with model points. Therefore, to make the adjustment robust, only those edge points in the search image perpendicular to the local tangent direction are taken into account. Furthermore, corresponding edge points that have a distance larger than a statistically computed threshold are ignored in the adjustment. Since the point correspondences may change by the newly estimated pose, an even higher accuracy can be gained by iterating the correspondence search and the parameter adjustment. Typically, after three iterations the accuracy of the pose no longer improves.

4.4 Performance Evaluation of the MGHT and the SBM

In this section, an extensive empirical performance evaluation of the two developed approaches, the MGHT and SBM, is presented.

To assess the performance of the two approaches, they are compared to six different 2D object recognition techniques that are already available. For this purpose, three standard similarity measures are chosen because they are frequently used methods in industry and therefore are rather well known: the sum of absolute differences (SAD) and the normalized cross correlation (NCC) as two standard intensity-based approaches and the Hausdorff distance (HD) as a standard feature-based approach. They constitute a common basis that facilitates the comparability of the subsequent evaluation with other evaluations that also include one of these standard approaches. Furthermore, including the three standard approaches in the performance evaluation is scientifically interesting because no direct comparison of the three approaches could be found in literature. Additionally, to be able to evaluate the potential of the two new approaches, three commercial high-end recognition tools have been included in the evaluation: the *Geometric Model Finder* (GMF) developed by *Matrox Electronic Systems Ltd.* (Matrox 2002) as well as *PatMax*® (PM) and *PatQuick*® (PQ), both developed by Cognex (Cognex 2000). The analysis of the performance characteristics of rigid object recognition methods is an important issue (cf. (Ulrich and Steger 2001)). Firstly, it helps to identify breakdown points of the algorithms, i.e., areas where the algorithms cannot be used because some of their underlying assumptions are violated. Secondly, it makes an algorithm comparable to other algorithms, thus helping users to select the appropriate method for the task they have to solve. In the special case of this dissertation a performance evaluation of rigid object recognition methods is essential in order to select the approach that is best suited to serve as a module within the approach to recognize compound objects.

The remainder of the chapter is organized as follows. After a short introduction of the respective methods, several criteria that allow an objective evaluation of object recognition approaches are introduced. Three main criteria are used to evaluate the performance and to build a common basis that facilitates an objective comparison (cf. (Ulrich and Steger 2002)): the robustness, the accuracy of the returned pose parameters, and the computation time. Thereby, it is distinguished between the robustness against occlusions and clutter and the robustness against arbitrary changes in brightness. Experiments on real images are used to apply the proposed criteria. For this, in a first step, the experimental set-up for the evaluation measurements is explained in detail. In a second step, the results are illustrated and analyzed.

4.4.1 Additionally Evaluated Object Recognition Methods

4.4.1.1 Sum of Absolute Differences

The special implementation that has been investigated to evaluate the sum of absolute gray value differences considers rigid motion and makes use of image pyramids to speed up the recognition process. The average distance is used as dissimilarity measure using the formula given in (4.3). Since the SAD is a measure of dissimilarity, the resulting average difference is also denoted as error e in the following. Hence, in contrast to all other methods this measure is not normalized to a certain interval, but can take arbitrary values, where an error of 0 denotes a perfect match. The position and orientation of the best match, i.e., the match with smallest error, is returned. Additionally, it is possible to specify the maximum error e^{max} of the match to be tolerated, similar to s^{min} when using a similarity measure. The lower e^{max} is chosen, the faster the recognition is performed, since fewer matches must be tracked down the pyramid. Subpixel accuracy of position and the refinement of the discrete orientation are calculated by interpolating the minimum of e.

4.4.1.2 Normalized Cross Correlation

For the purpose of evaluating the performance of the normalized cross correlation the implementation of the *Matrox Imaging Library* (MIL) — as one typical representative — was used. The MIL is a software development toolkit of *Matrox Electronic Systems Ltd.* (Matrox 2001). Some specific implementation characteristics should be explained to ensure the correct appraisal of the evaluation results: The algorithm is able to find a predefined object under rigid motion. A hierarchical search strategy using image pyramids is used to speed up the recognition. The quality of the match is returned by calculating a score value as $s = max(NCC, 0)^2 \in [0, 1]$, where NCC is the value of the NCC as it is obtained when using (4.2). The subpixel accuracy of the object position is achieved by a surface fit to the match scores around the peak. The refinement of the object orientation is not comprehensively explained in the documentation. However, it is supposed that the refinement of the obtained discrete object orientation is realized by a finer resampling of the orientation in the angle neighborhood of the maximum score and recalculating the NCC at each refined orientation.

4.4.1.3 Hausdorff Distance

The original implementation by Rucklidge (Rucklidge 1997) was utilized to rate the performance of the Hausdorff distance. The core of the implementation uses the partial HD (cf. (4.8) and (4.9)). The program expects the forward and the reverse fraction as well as the thresholds for the forward and the reverse distance as input parameter. Both, the model image and the search image must be passed in binary form to the algorithm. Therefore, edges are extracted in advance using a minimum edge magnitude γ^{min} by applying the same preprocessing steps as in the case of the MGHT and the SBM to ensure a high comparability of the approaches. Since the method of (Rucklidge 1997) returns all matches that fulfill the given score and distance criteria, the best match was selected based on the minimum forward distance. If more than one match had the same minimum forward distance, the match with the maximum forward fraction was selected as the best match. The forward fraction was interpreted as score value $s \in [0, 1]$ during the evaluation in order to measure the quality of the match. A score of 1 denotes a perfect match in the sense that all model pixels fulfill the selected forward distance. The implementation is not able to recognize rotated objects but only allows to recognize translated objects with fixed orientation. Furthermore, no subpixel refinement is included. Generally, it seems to be very difficult to compute a refinement of the returned parameters directly based upon the forward or reverse fraction. Although the parameter space is treated in a hierarchical way there is no use of image pyramids, which makes the algorithm very slow.

4.4.1.4 Geometric Model Finder

The Geometric Model Finder uses edge-based geometric features to find objects in images. Unfortunately, no detailed information about the principle of the GMF is available from its documentation (Matrox 2002). The default threshold that is used for edge extraction in the model image as well as in the search image can be influenced by a parameter that allows to chose the detail level, and hence whether a medium (default), a high, or a very high number of edges are to be extracted. This parameter of the GMF is similar to the threshold for the minimum edge magnitude γ^{min} used in the MGHT, the SBM, and the HD. To measure the quality of a match, a score value $s \in [0, 1]$ is computed and returned. The minimum score that must be reached by a match candidate can be set by choosing s^{min}. Finally, the object pose is returned with subpixel accuracy. However, as a last important parameter to mention, the accuracy level can be controlled choosing between a medium and a high accuracy. The high accuracy is gained at the expense of a slower computation.

4.4.1.5 PatMax and PatQuick

As described in their documentation (Cognex 2000), PatMax® and PatQuick® use geometric information. The main difference between the two approaches is that PQ is a faster but less accurate version of PM. Both methods apply a three-step geometric measurement process to an object: At first, it identifies and isolates the key individual features within the model image and measures characteristics such as shape, dimensions, angle, arcs, and shading. Then, it matches the key features of the model and the search image, encompassing both distance and relative angle. By analyzing the geometric information from both the features and spatial relationships, PM and PQ are able to determine the object's position and orientation precisely, i.e., the accuracy is not limited to a quantized sampling grid.

The model representation, however, which can be visualized by PM and PQ, apparently consists of subpixel precise edge points and respective edge directions. From this, one can be led to conclude that the two approaches use similar features as the SBM, in contrast to their documentation.

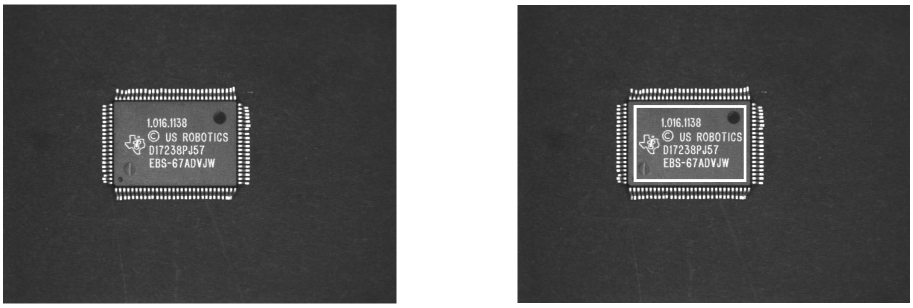
To speed up the search, a coarse-to-fine approach is implemented. Indeed, there is no parameter that enables users to explicitly specify the number of image pyramids to be used. Instead, the parameter *coarse grain limit* can be used to control the size of the features to be used during recognition and, therefore, the depth of the hierarchical search. This parameter has a similar meaning as the number of pyramid levels, but it cannot be equated with. To indicate the quality of the match, a score $s \in [0, 1]$ is computed, where 1 again represents a perfect match.

4.4.2 Robustness

The first criterion to be considered is the *robustness* of the approaches. This includes the robustness against occlusions and clutter as well as the robustness against arbitrary changes in brightness.

Experimental Set-Up. For all subsequent experiments an IC (see Figure 4.22(a)) was chosen as the object to be found. Only the part within the bounding box of the print on the IC is used as ROI, from which the models of the different recognition approaches are created (see Figure 4.22(b)). For the recognition methods that segment edges during model creation (MGHT, SBM, HD), the threshold for the minimum edge magnitude γ^{min} in the model image was set to 30 for all experiments. The detail level of the GMF was set to medium, which is the default value and results in approximately the same edges that are obtained when applying a threshold of 30 to the edge magnitude. All images that were used for the evaluation are of size 652 × 494 pixels. The experiments were performed on a 400 MHz Pentium II. For those recognition methods that use image pyramids (MGHT, SBM, SAD, NCC), four pyramid levels were used to speed up the search. The algorithm presented in Section 4.2.3.1 found this number of levels to be the optimum for the IC. This number also agrees with human intuition. For PQ and PM the automatically determined value for the parameter *coarse grain limit* was assumed to be the optimum one, and hence no manual setting was applied.

To apply the criterion of robustness, two image sequences were taken, one for testing the robustness against occlusions and clutter, the other for testing the robustness against changes in brightness. The recognition rate was defined as the number of images in which the object was recognized at the correct pose divided by the total number of images, and hence is an indicator for robustness against occlusions. The false alarm rate was defined as the number of images, in which the object was recognized at an incorrect pose divided by the total number of images, and thus is an indicator for robustness against clutter. Such matches are called false positives.



(a) Model image

(b) ROI

Figure 4.22: An IC is used as the object to be recognized (a). The model is created from the print on the IC using a rectangular ROI (b).

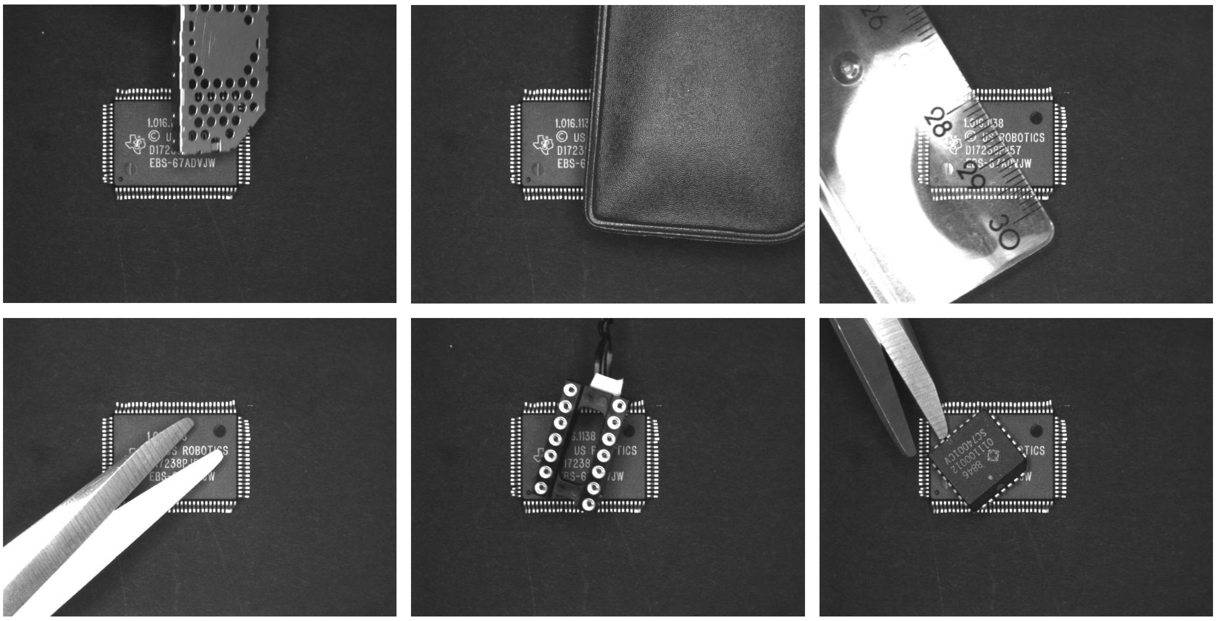


Figure 4.23: Six of the 500 images that were used to test the robustness against occlusions

The first sequence contains 500 images of the IC, in which the IC was kept at a fixed position and orientation and was occluded to various degrees with various objects, so that in addition to occlusion, clutter of various degrees was created in the images. Figure 4.23 shows six of the 500 images. In the corresponding model image of this sequence, the size of the bounding box that defines the ROI is 180 × 120 pixels at the lowest pyramid level and contains 2127 edge pixels extracted by the Sobel filter. For the approaches that segment edges in the search image (MGHT, HD, and GMF) the parameter to control the edge extraction was set to the same value as in the model image.

Additionally, the relation between the actual degree of occlusion and the returned score value is examined, because the correlation between the visibility v of the object and the score s can also be seen as an indicator for robustness. If, for example, only half of the object is visible in the image then, intuitively, also the score should be 0.5, i.e., one would expect a very high correlation in the ideal case. For this purpose, an effort was made to keep the IC in exactly the same position in the image in order to be able to measure the degree of occlusion. The true amount of occlusion was determined by extracting edges from the search images and intersecting the edge region with the edges within the ROI in the model image. Since the objects that occlude the IC generate clutter edges, this actually underestimates the occlusion.

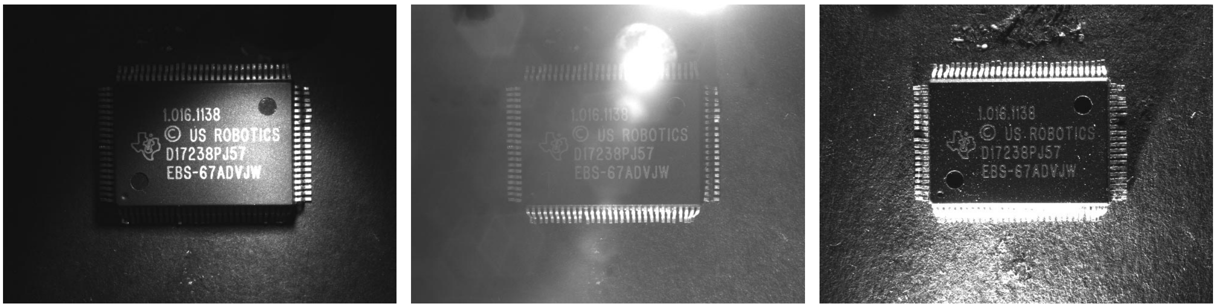


Figure 4.24: Three of the 200 images that were used to test the robustness against arbitrary changes in brightness

To test the robustness, the transformation class was restricted to translations, in order to reduce the time required to execute the experiment. However, the allowable range of the translation parameters was not restricted, i.e., the object was searched in the entire image. It should be noted that the recognition rate would be lower and the false alarm rate would be higher when allowing rigid motion instead of translations only. This is because the probability for an arbitrarily rotated model to match a clutter object is higher than for the model at a fixed orientation. Nevertheless, restricting the experiment to translations is legitimate because it can be assumed that the resulting percentage change in both rates compared to rigid motion is approximately the same for all approaches. Consequently, a qualitative comparison is ensured. For the MGHT, the SBM, the NCC, the HD, PQ, and PM different values for the parameter of the minimum score were applied. As previously mentioned, in the case of the HD the forward fraction was interpreted as score value. Initial tests with the forward and reverse fractions set to 0.3 resulted in run times of more than three hours per image. Therefore, the reverse fraction was set to a constant value of 0.5 and the forward fraction was successively increased from 0.5 to 0.9 using an increment of 0.1. The parameters for the maximum forward and reverse distance were set to 1 pixel. For the other three approaches the minimum score s^{min} was varied from 0.1 to 0.9. In the case of the SAD the maximum mean error e^{max} instead of the minimum score was varied. Since the mean error e is not limited to an interval, e^{max} was varied from 10 to 50 using an increment of 10. Tolerating higher values for e would result in hardly meaningful matches, i.e., an occluded object instance could not be distinguished from clutter in the search image. Furthermore, extremely expensive computations would be the consequence, which would make the algorithm unsuitable for practical use. Since the robustness of the SBM depends on the parameter greediness, additionally the value for greediness was varied in the range of 0 to 1 using increments of 0.2.

To test the robustness against arbitrary changes in brightness, a second image sequence of the IC was taken. The sequence contains 200 images including various illumination situations, e.g., spot lights, reflections, non-uniform illumination, different ambient light intensity, etc. Three example images are displayed in Figure 4.24. Because of the smaller distance between the IC and the camera, the ROI is now 255 × 140 pixels and contains 3381 model points on the lowest pyramid level. The parameter settings of all methods are equivalent to the settings for testing the robustness against occlusions. However, since the MGHT segments the search image, additionally the threshold for the minimum edge magnitude in the online phase is varied from 5 to 30 using an increment of 5. The same holds for the GMF, where the detail level was set to medium, high, and very high, respectively. Furthermore, in the case of the SAD the range of values for the maximum mean error e^{max} was limited from 10 to 30 since higher values showed no significant improvements.

Results. At first, the sequence of the occluded IC was tested. A complete comparison of all approaches concerning the robustness against occlusion is shown in Figure 4.25. The recognition rate is plotted versus the minimum score s^{min} and the maximum error e^{max}, respectively. For the SBM the greediness was set to 0 at first, in order to receive the best obtainable recognition rate. As one would expect, the number of correctly recognized objects decreases with increasing minimum score for all approaches. Thus, the higher the degree

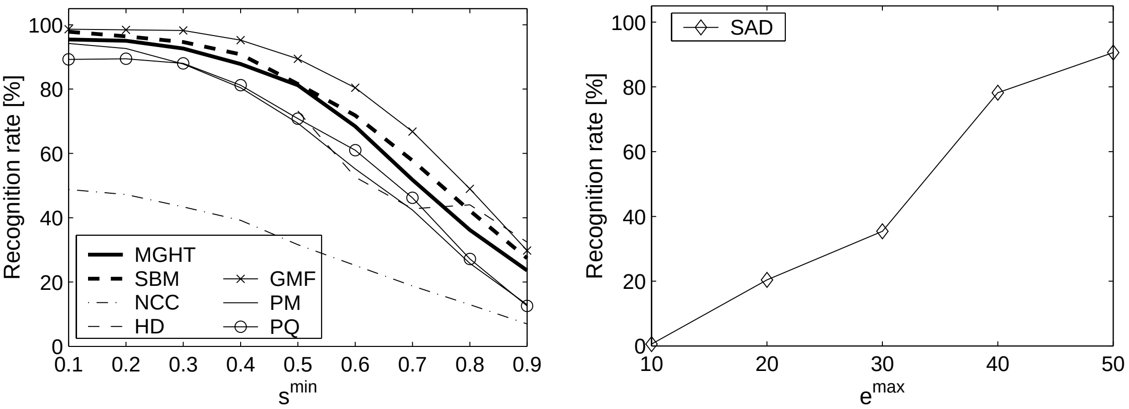


Figure 4.25: The recognition rate indicates the robustness against occlusions. The recognition rate depends on the chosen value for s^{min} and e^{max}, respectively.

of occlusions the smaller the parameter of the minimum score must be chosen in order to correctly recognize the occluded objects. The opposite holds for the maximum error in the case of the SAD. In Figure 4.25 the inferiority of the intensity-based approaches (SAD, NCC) to the feature-based approaches becomes clear. The NCC does not reach a recognition rate of 50% at all, even if the minimum score is chosen small. The approach using the SAD as similarity measure also shows a poor behavior: although the expectation is fulfilled that the robustness increases when the maximum error is set to a higher value, even relatively high values for the mean maximum error (e.g., 30) only lead to a small recognition rate of about 35%. Admittedly, a further increase of the maximum error results in higher recognition rates. However, no real improvement is achieved since meaningful results of matches exhibiting maximum mean errors of 50 gray values are hardly imaginable. This suspicion will be confirmed later. The HD, which incorporates the standard feature-based approach in the evaluation, shows significantly better results, especially for high values of s^{min}. This can be explained by the fact that in the case of severe occlusions, clutter edges in the search image reduce the otherwise high distance values of the forward distance and therefore lead to higher values for the forward fraction. The MGHT and the SBM both show very high recognition rates. They are only beaten by the GMF, which achieves the best result in this particular test. It is worth noting that the robustness of the MGHT hardly differs from the robustness achieved by the SBM even when using a greediness of 0. Comparable results are obtained by PM and PQ, which both, however, are significantly inferior to the MGHT, the SBM, and the GMF.

The robustness against occlusions of the SBM depends on the greediness parameter g. Figure 4.26 shows the recognition rate for different values of g. Apparently, the greediness parameter must be adjusted carefully when dealing with occluded objects. For a given minimum score of 0.5, for example, the recognition rate varies in the range between 48% and 82%, corresponding to the two extreme greediness values of 1 and 0. However, already greediness values of 0.8 and 0.6, improve the recognition rate significantly to 64% and 70%, respectively.

Up to now, only the robustness against occlusions was analyzed. This constitutes only one component of two inherently associated attributes. Imagine a degenerated recognition method that simply returns matches at each possible object pose within the class of transformations. In this case, the robustness against occlusions would be perfect, because even if the object is not present in the image, i.e., the object is occluded by 100%, it still would be found. Consequently, the second component that must be considered is the robustness against clutter, which on its own is also insufficient because a degenerated recognition method that never returns any match shows a perfect robustness against clutter. Hence, a high quality recognition approach combines robustness against occlusions as well as against clutter. Analogously to the recognition rate the robustness against clutter can be quantified by the false alarm rate. The *receiver operating characteristic* is a perfect feature to simultaneously evaluate the robustness against occlusions as well as against clutter, since it plots the false alarm rate versus the recognition rate.

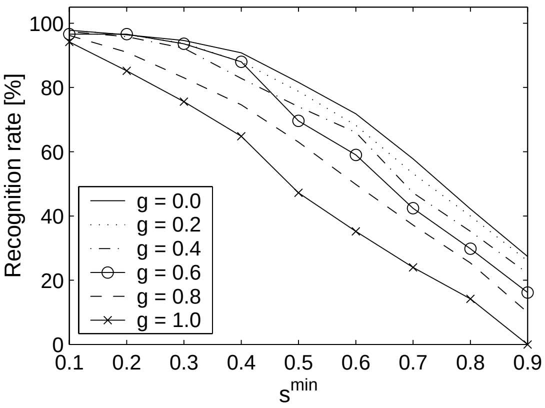


Figure 4.26: Recognition rate of the SBM additionally depends on the greediness parameter g in the case of occlusions. The “greedier” the search, the more matches are missed.

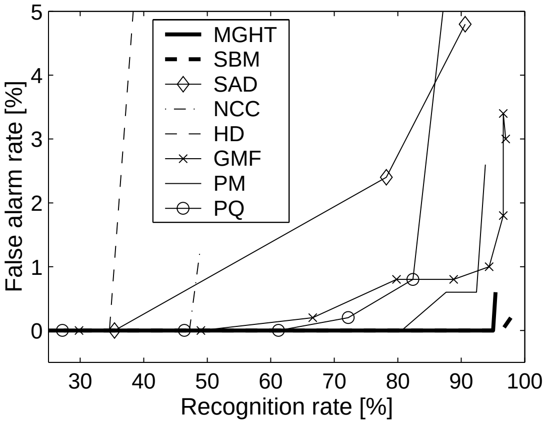
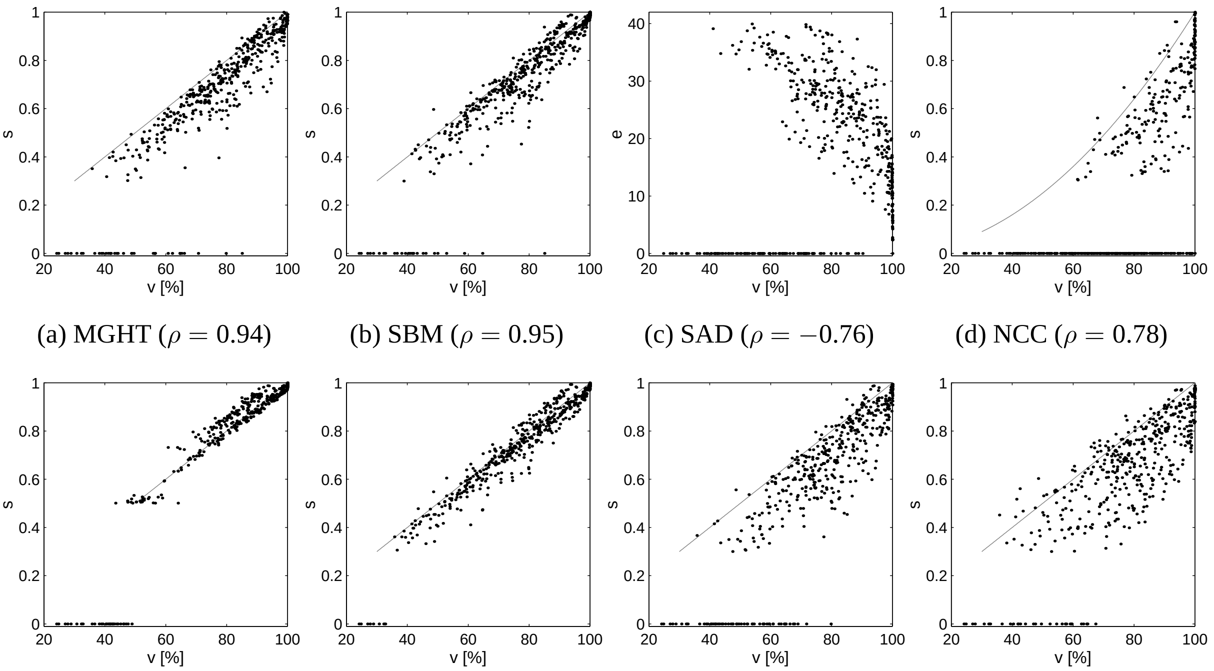


Figure 4.27: The receiver operating characteristic simultaneously evaluates the robustness against occlusions and clutter.

In Figure 4.27 the receiver operating characteristic curves of the respective approaches are shown. Here, the transformation class was restricted to translations. For arbitrary object orientations an even higher false alarm rate must be expected. This is because the probability of a clutter object fitting the arbitrarily rotated object is higher than fitting the object at one specific orientation.

The HD shows a very poor behavior because already for a low recognition rate of about 35%, false positives are returned. The false alarm rate reaches its maximum of about 32% at a recognition rate of 73% (not visible in the plot because of axis scaling). Also the NCC returns false positives even for low recognition rates. The SAD does not return any false positive for recognition rates less than 35%, which corresponds to a maximum mean error of 30 (see Figure 4.25). However, as already suspected in the previous analysis, the false alarm rate increases considerably if higher maximum mean errors are tolerated. PQ on the one hand returns stable results for recognition rates up to approximately 70%. On the other hand the false alarm rate dramatically increases for higher recognition rates, culminating in 11% false positives for the maximum achieved recognition rate of 89% (also not visible in the plot). Better results are obtained by PM, which only returns a few false positives (2.6%) even when high recognition rates (93%) are achieved. In comparison, the GMF performs worse for lower recognition rates since the false alarm rate starts to increase already for a recognition rate of 50% and reaches its maximum of 3.4% false positives. This depreciates the high recognition rates, which are obtainable with this approach, considerably (see Figure 4.25). Finally, the two developed approaches, MGHT and SBM, exhibit the highest robustness against occlusions and clutter of the evaluated object recognition methods. Even for very high recognition rates of 95% and 98% the false alarm rates remain below 0.6% (three images) and 0.2% (one image), respectively.



(a) MGHT ($\rho = 0.94$) (b) SBM ($\rho = 0.95$) (c) SAD ($\rho = -0.76$) (d) NCC ($\rho = 0.78$)

(e) HD ($\rho = 0.95$) (f) GMF ($\rho = 0.97$) (g) PM ($\rho = 0.88$) (h) PQ ($\rho = 0.79$)

Figure 4.28: Returned values for score s and error e, respectively, plotted versus the visibility v of the object. In the ideal case the correlation coefficient ρ should be 1 (in the case of the score values) and -1 (in the case of the error values), respectively.

Figure 4.28 displays the returned values for the score s and mean error e, respectively, plotted versus the estimated visibility v of the object. I.e., the correlation between these two quantities is visualized. For the plots, s^{min} was set to 30 (50 in the case of the HD) and e^{max} was set to 40. False positive matches are not visualized in the plots. To facilitate the visual evaluation, additionally, for the approaches returning a score value, the ideal curves representing 100% correlation are plotted, i.e., $s^{ideal} = v/100$. In the case of the NCC the score value is computed as $s = max(NCC, 0)^2$, and hence the associated ideal curve is $s^{ideal} = (v/100)^2$ (cf. Section 4.4.1.2). Images in which no match was found are denoted by a score or error value of 0, i.e., the corresponding points lie on the x axis of the plots. To precisely measure the correlation, additionally the correlation coefficient ρ (Bronstein et al. 2001) is computed from corresponding values of correctly found objects.

It can be seen from the plot regarding the SAD that the error is negatively correlated with the visibility — as expected. The corresponding value for $\rho = -0.76$ proves the visual impression. Nevertheless, the points are widely spread and far from an ideal virtual line with negative gradient. In addition, despite of a very high degree of visibility many objects were not recognized. One possible reason for this behavior could be that in some images the clutter object does not occlude the IC yet, but casts its shadow onto the IC, which strongly influences this metric. The magnitude of the correlation coefficient obtained for the NCC is comparable to that obtained for the SAD. Furthermore, also here the points in the plot are widely spread and many objects with high visibility were not recognized.

Most of the remaining approaches show a significantly higher positive correlation. This again confirms the higher robustness of the feature-based approaches compared to the area-based approaches. As the rules exception, the correlation coefficient obtained for PQ has a relatively low value. The plot of PM shows a better behavior, leading to a higher value for ρ. However, also here the points are not close to the ideal line but spread by a higher amount in comparison to the MGHT, the SBM, the HD, and the GMF. These approaches all show similar results and a point distribution that is much closer to the ideal one. In addition, objects with high visibility are recognized with a high probability. However, again it becomes clear that in a few cases

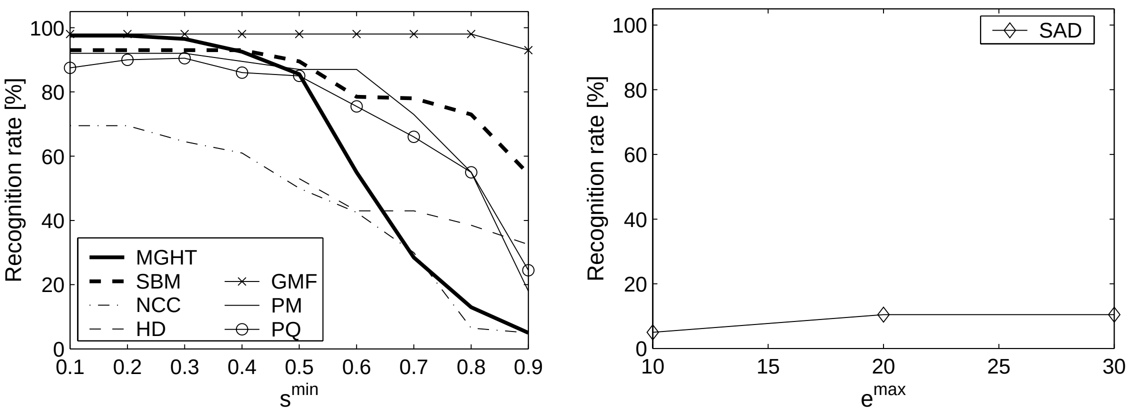


Figure 4.29: The recognition rate indicates the robustness against arbitrary changes in brightness. The recognition rate depends on the chosen value for s^{min} and e^{max}, respectively.

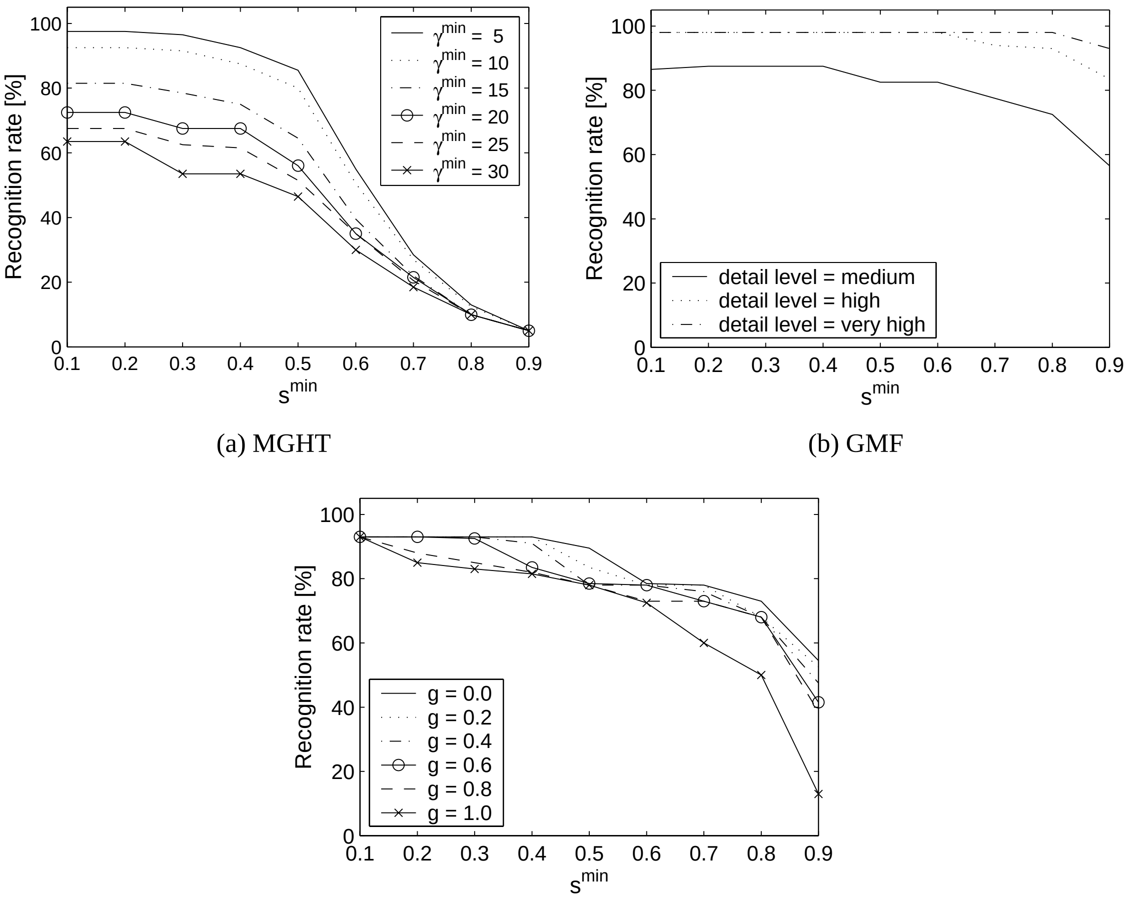
of higher object visibility the MGHT and the SBM cannot recognize the object. Conversely, the HD and the GMF can recognize the objects. But one has to keep in mind the corresponding false alarm rates.

In the following, the robustness against arbitrary changes in brightness is analyzed. Figure 4.29 shows a comparison of the recognition rates of the respective approaches. For the MGHT, the SBM, and the GMF, the respective best parameter settings are applied, i.e., the threshold for the edge extraction for the MGHT was set to the smallest value of 5, the greediness of the SBM was set to 0, and the highest detail level was used for the GMF.

The SAD shows even worse recognition rates than in the case of occlusions: now, the best recognition rate that could be obtained using a maximum error of 30 was only 11%. By comparing this value to the result obtained for a maximum error of 20, which is also 11%, it is obvious that even by further increasing the maximum allowable error no meaningful improvement can be reached. In contrast, the recognition rate of the NCC is substantially better. Obviously, the robustness of the NCC against changes in brightness is higher than against occlusions. This can be attributed to its normalization, which compensates at least global changes in brightness. The result obtained by the HD is superior to that of the NCC. Especially, in the case of large values for the minimum score it shows good results. However, for lower values it cannot reach the performance of the remaining approaches. If the minimum score is set low enough, the recognition rate of the MGHT even surpasses that of the SBM, PQ, and PM, reaching a result comparable to the GMF. For higher values its recognition rate decreases rapidly. PM and PQ show approximately equivalent results, both of which are inferior to the SBM for almost all values of s^{min}. Also here, the GMF achieves a very high and approximately constant recognition rate even for large values of s^{min}.

In the case of the MGHT and the GMF the recognition rate additionally depends on the chosen threshold for the edge extraction in the search image. As in the case of occlusions the recognition rate of the SBM additionally is influenced by the greediness parameter. Therefore, Figure 4.30 shows the recognition rates of the three approaches for different parameter settings.

The MGHT (see Figure 4.30(a)) allows to specify the minimum edge magnitude γ^{min}. The recognition rate of the MGHT strongly depends on the chosen threshold for edge extraction in the search image. As expected, higher recognition rates are obtained for lower values of the minimum edge magnitude, because fewer edge pixels fall below the threshold γ^{min}. The higher the minimum edge magnitude, the more edge pixels are missed, because dimming the light as well as stronger ambient illumination reduces the contrast. Thus, this effect is comparable to the effect of higher occlusion. Therefore, a high recognition rate can be obtained by setting the minimum score to a lower value or by choosing a lower threshold for the edge magnitude. For example, a minimum score of 0.5 and an edge threshold of 10 leads to a recognition rate of 84%. Nevertheless, the true invariance of the SBM against changes in brightness could not be reached by the MGHT. In the case of



(a) MGHT

(b) GMF

(c) SBM

Figure 4.30: The recognition rate in case of changes in brightness additionally depends on the threshold for edge extraction in the search image. The threshold can be set by γ^{min} for the MGHT (a) and by the detail level for the GMF (b). For the SBM (c) the greediness g influences the recognition rate.

the GMF (see Figure 4.30(b)) the influence of the edge extraction is less distinct. Nevertheless, if the medium detail level is chosen, the recognition rate decreases by more than 15%. The recognition rate of the SBM (see Figure 4.30(c)) in the case of changing brightness is less sensitive to the chosen greediness parameter g than in the case of occlusions. Only, when setting g to 1 and choosing high values for s^{min} a significant deterioration in the recognition rate is observable. Disregarding the result obtained with greediness set to 1, the discrepancy is smaller than 10%.

4.4.3 Accuracy

The second evaluation criterion is the *accuracy* of the approaches. The accuracy is determined by comparing the exact (known) position and orientation of the object with the returned pose parameters of the different candidates. Since the HD does not return the object position with subpixel accuracy this criterion is only applied to the remaining candidates. Additionally, the least-squares pose refinement (LSPR), which has been presented in Section 4.3.3, is taken into account in order to evaluate the gain in accuracy that is achieved by this method. Therefore, in the following the results denoted by SBM refer to the pose refinement that is obtained by the polynomial fitting, whereas LSPR implies the additional improvement of the pose parameters that is obtained by the least-squares adjustment in the SBM.

Experimental Set-Up. To evaluate the accuracy, the IC was mounted onto a table that can be shifted with an accuracy of $1\,\mu$m and can be rotated with an accuracy of 0.7’ (0.011667°). Figure 4.31 illustrates the experimental set-up. Three image sequences were acquired: In the first sequence, the IC was shifted in $10\,\mu$m increments to the left in the horizontal direction, which resulted in shifts of about 1/7 pixel in the image. A total of 40 shifts were performed, while 10 images were taken for each position of the object. The IC was not occluded in this experiment and the illumination was approximately constant. In the second sequence, the IC was shifted in the vertical direction with upward movement in the same way. In this case, a total of 50 shifts were performed. The intention of the third sequence was to test the accuracy of the returned object orientation. For this purpose, the IC was rotated 50 times for a total of 5.83°. Again, 10 images were taken at each orientation.

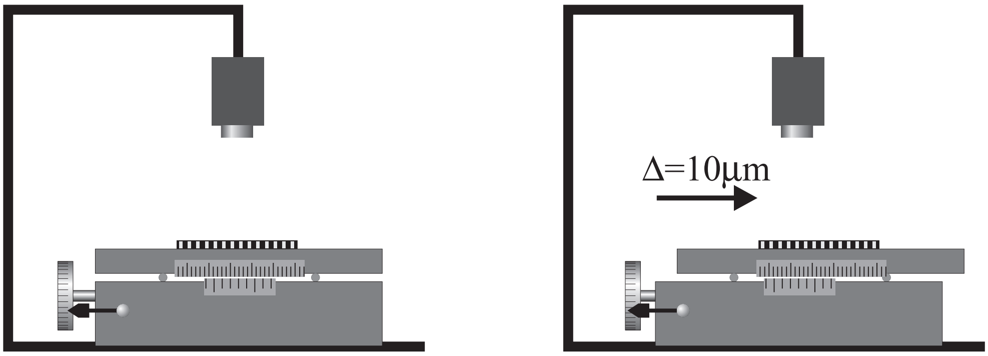


Figure 4.31: To test the accuracy, the IC was mounted on a precise $xy\varphi$-stage, which can be shifted with an accuracy of $1\,\mu$m and can be rotated with an accuracy of 0.7’ (0.011667°).

The search angle for the object orientation was restricted to the range of $[-30°;+30°]$ for all approaches, whereas the object position again was not restricted. Since no occlusions were present s^{min} could be uniformly set to 0.8 for all approaches. For the SAD e^{max} was specified to be 25. The greediness parameter of the SBM was set to 0.5, which represents a good compromise between recognition rate and computation time. For the GMF, the accuracy level was varied from medium to high.

Results. To assess the accuracy of the extracted object position, a straight line was fitted to the mean extracted coordinates of position. This is legitimated by the linear variation of the position and orientation of the IC in the world, which can be assumed when using the precise $xy\varphi$-stage. Because the IC is shifted in world units (μm) while the recognition approaches return the position of the IC in pixel coordinates, the exact position of the IC is only known in world units but not in pixel coordinates. Because this scaling is unknown, the slope of the straight line cannot be set to 1 but must be estimated during the line fit. In contrast, to assess the accuracy of the extracted object orientation the straight line does not need to be estimated but can be computed directly. This is because the unit in which the IC is rotated on the stage and the unit in which the recognition approaches return the object orientation are identical, and hence no scaling needs to be considered. The residual errors, i.e., the differences of the extracted position and orientation to the straight lines, shown in the Figures 4.32 and 4.33, are a well suited indication of the achievable accuracies.

As can be seen from Figure 4.32, the position accuracy of the MGHT, the SBM, the LSPR, the NCC, the GMF (both accuracy levels), and PM are very similar. The corresponding errors are in most cases smaller than 1/20 pixel. The conspicuous peaks in both error plots of Figure 4.32 occur for all these approaches with similar magnitude. Therefore, and because of the nearly identical lines, it is probable that the IC was not shifted accurately enough, and hence the error must be attributed to a deficient acquisition. Nevertheless, it can be concluded that the error in position in most cases must at least be smaller than 1/20 pixel, which is sufficient for most applications. However, the high and oscillating error of about 1/10 pixel when using the SAD cannot be attributed to this deficient acquisition. This error occurs because of subpixel translations that

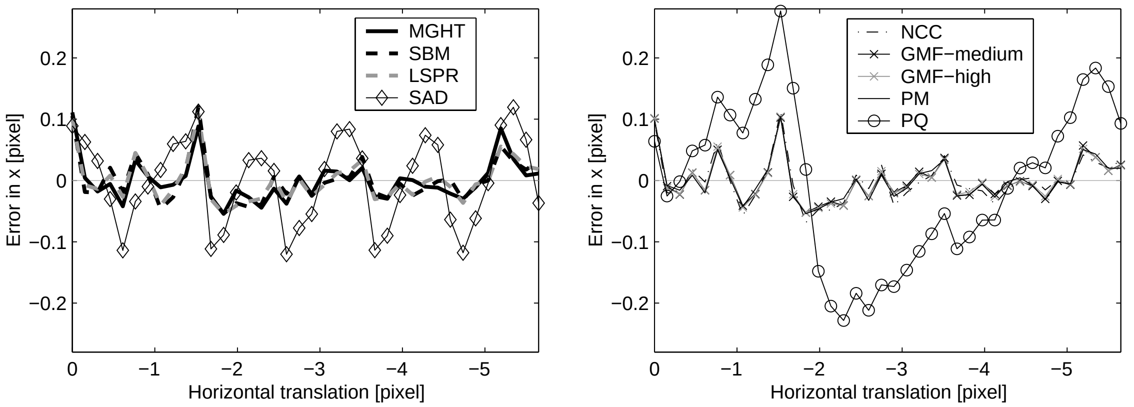


Figure 4.32: Position accuracy plotted as the difference between the actual x coordinate of the IC and the x coordinate returned by the recognition approach while shifting the IC successively by 1/7 pixel to the left

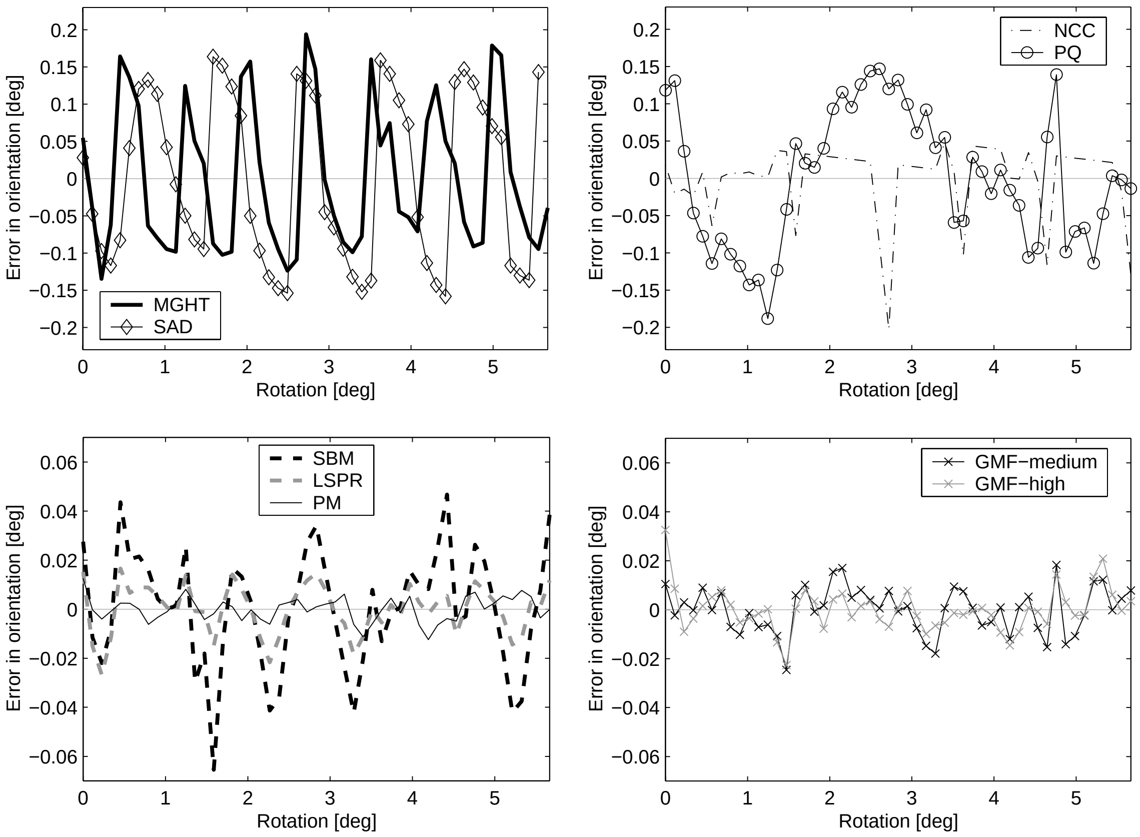


Figure 4.33: Orientation accuracy plotted as the difference between the actual object orientation of the IC and the returned angle by the recognition approach while rotating the IC successively by approximately 1/9° counterclockwise. The different scalings of the plots should be noted.

influence the gray values especially in high contrast areas of the image. Finally, PQ shows the highest errors in the x coordinate of all approaches in the test, reaching a maximum error of 1/5 pixel. Since the errors in y approximately have the same magnitude as in x they are not presented.

Figure 4.33 shows the corresponding errors of the returned object orientation. Here, the SBM complemented by the LSPR, the GMF, and PM are superior to all other candidates. They reach maximum errors between 1/50° and 1/100° in this example. Furthermore, the improvement of both the LSPR in comparison to the SBM and the high accuracy level of the GMF in comparison to the medium accuracy level becomes visible. The error of the SBM is reduced to 42% when using the LSPR. Similarly, the high accuracy level of the GMF results in orientation errors that are about 84% in magnitude of the errors when using the medium accuracy level. The remaining approaches return a less accurate object orientation. The corresponding maximum errors of the MGHT, the SAD, the NCC, and PQ are about 1/6° (10') in this example. However, it should be remarked that the accuracy of the MGHT can be easily improved by the LSPR in a similar way as the SBM because the same features (edge position and orientation) are used in both methods. Thus, a similar accuracy level can be expected when applying the LSPR to the MGHT.

4.4.4 Computation Time

The *computation time* represents the third evaluation criterion. Indeed, it is very hard to compare different recognition methods with this criterion because the computation time strongly depends on the individual implementation of the recognition methods. Nevertheless, in conjunction with the achievable robustness and accuracy the associated computation times at least allow a qualitative comparison. The HD was excluded from this test because the implementation does not use image pyramids, which results in unreasonably long recognition times. This would lead to an unfair comparison.

Experimental Set-Up. In order to test the third criterion, the configurations that were used for testing the robustness against occlusions (cf. Section 4.4.2) and for testing the accuracy (cf. Section 4.4.3) are used. The computation time of the recognition processes was measured on a 400 MHz Pentium II for each image of the sequences and for each recognition method. Therefore, one has to keep in mind that the measured recognition times would reduce to a fraction on current computers (e.g., a 2.8 GHz Pentium 4). However, it can be assumed that the relation of the recognition times between different approaches approximately remains unchanged.

The computation time of the SBM was measured for two greediness values of 0 and 1 with using the sequence for testing the robustness in order to be able to estimate the increase in computational cost for a gain in robustness. Furthermore, in order to assess the correlation between the size of parameter space and computation time, the two sequences for testing the accuracy (horizontal shift and rotation) are used a second time without restricting the angle interval to $[-30°, 30°]$, but searching the object in the full range of orientations ($[0°, 360°[$).

In this context it should be noted that the MGHT and the GMF are the only candidates whose implementations are able to recognize the object even if it partially lies outside the search image. The other approaches automatically restrict the range of possible object positions to those at which the object completely lies within the search image. Therefore, particularly in the case of large objects both methods are disadvantaged when comparing their computation times to those of the remaining approaches. This should be kept in mind when analyzing the results.

Results. At first, the computation time of the respective approaches during the robustness test is analyzed. In Figure 4.34 the mean computation time T over the 500 images of the sequence is plotted for three different values of e^{max} (SAD) and s^{min} (others), respectively. This facilitates an easy assignment of the computation times to the recognition rates displayed in Figure 4.25. Because for real-time applications the maximum time to find the object is often of interest, additionally, the maximum computation time is plotted. One can see that the approaches can be divided into two groups, where the approaches within the same group exhibit similar recognition times. In the first group (MGHT, SBM, NCC, PM, PQ) the mean recognition time varies in the range from 18 to 59 ms. In general, the lower s^{min} is chosen, the higher the robustness against occlusions (see Figure 4.25), but the higher the computation time. It also becomes evident that PQ is faster than PM. Furthermore, a greediness value of 1 speeds up the SBM in comparison to a greediness value of 0, especially

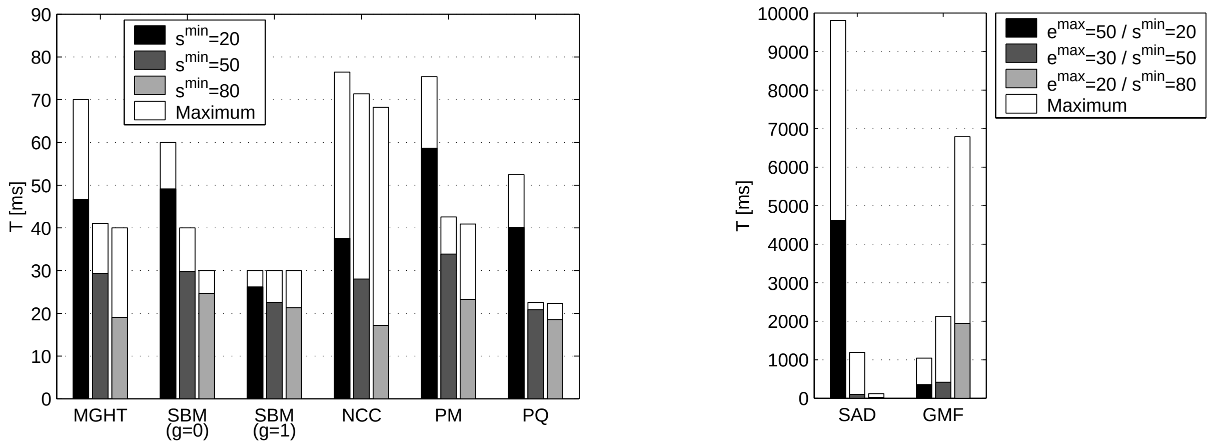
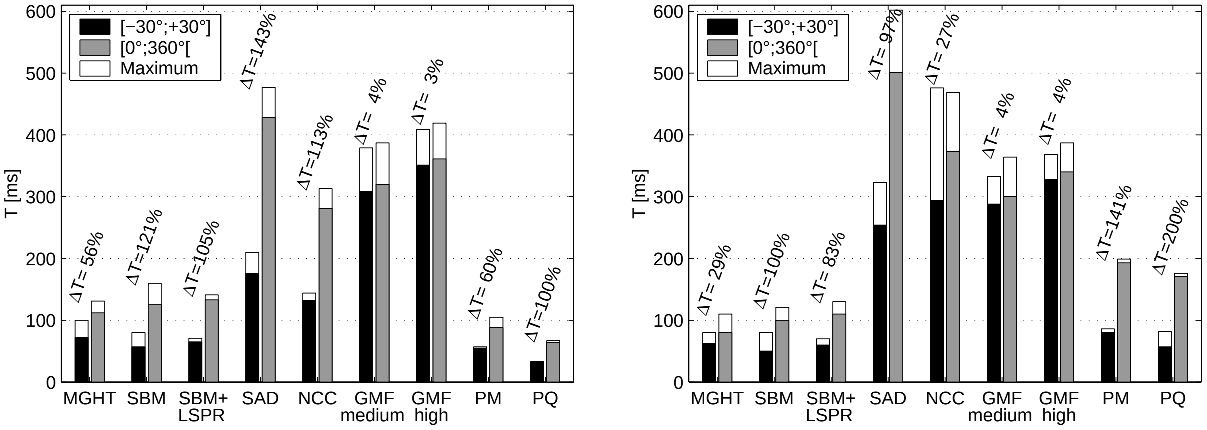


Figure 4.34: Mean and maximum recognition times of the respective approaches when applying the sequence for testing the robustness against occlusions. The computation time of each approach is represented by three bars, each associated with a certain value for e^{max} (SAD) and s^{min} (others), respectively. The different scalings of the two plots should be noted.

if s^{min} is chosen small, i.e., a high number of match candidates must be tracked through the pyramid. In the second group (SAD, GMF) the mean recognition time is much higher. Even if e^{max} of the SAD is set to the relatively small value of 30, where the associated recognition rate is only 34%, the maximum computation time already exceeds one second. If higher recognition rates should be obtained (at the expense of higher false alarm rates) the computation time may even reach several seconds. The high recognition rates of the GMF have already been depreciated by the associated high false alarm rates. Additionally, its computation time reaches several seconds. Hence, the performance of the GMF suffers from a further depreciation. However, in contrast to all other approaches the computation time increases if s^{min} is set to a higher value. During evaluation it became obvious that the GMF is slowed down considerably if the object cannot be found or if the score s is near s^{min}. From this, it can be deduced that the implementation of the SAD and the GMF are not suited for real-time object recognition.

Figure 4.35 shows the respective mean and maximum computation times when applying the accuracy test. Again, the MGHT, the SBM, PM, and PQ show fast computations. Furthermore, the mean computation time of the SBM extended by the LSPR only marginally increases in comparison to the SBM. The computation time of the LSPR, which is represented as the difference between the computation time of "SBM+LSPR" and "SBM" in the plots, does not depend on the size of the parameter space: it is constant for a given object, and is 8 ms in the case of the translation sequence and 10 ms using the rotation sequence. Therefore, the larger the parameter space the smaller the influence of this constant part becomes. In contrast, again the SAD and the GMF are substantially slower. However, because the object does not show partial occlusions in the two sequences, the GMF is not slowed down to a high degree as in the occlusion case. This is because s^{min} was set to 0.8 and the score hardly differs from 1. Because e^{max} was set to 25, the robustness of the SAD would be very poor. Thus, also the SAD would show much longer recognition times if higher robustness was desired. When looking at the result of the NCC an increase of computation time is noticeable when searching the rotated object. During evaluation it became evident that the more the IC is rotated relative to the reference orientation in the model image the longer the computation time of the NCC became. Obviously, the implementation of (Matrox 2001) does not scan the whole orientation range at the highest pyramid level before the matches are tracked through the pyramid, but starts with a narrow angle range close to the reference orientation. Hence, the computation time of the NCC is not directly comparable to the other approaches, because the orientation range of $[-30°;+30°]$ and $[0°;360°[$ is not really scanned. Hence, a comparable computation time would be still higher.

From the time increase ΔT when extending the angle search range from $[-30°;+30°]$ to $[0°;360°[$ conclusions can be drawn about the ability of the approaches to deal with more general transformation classes.



(a) Sequence of horizontally shifted IC

(b) Sequence of rotated IC

Figure 4.35: Mean and maximum recognition times of the respective approaches when applying the sequence of the shifted IC (a) and that of the rotated IC (b). The computation time of each approach is represented by two bars, where the left bar and the right bar correspond to the restricted and the unrestricted orientation search range, respectively. The associated time increase ΔT is printed in percent.

The percentage increase of the mean computation time is printed in Figure 4.35. As can be seen from Figure 4.35(a) the computation time of the GMF merely increases by 4% and 3%, respectively. Obviously, the GMF ignores the restriction of the orientation search range and always recognizes the object within the full range of orientations. This makes the use of prior knowledge about the object orientation more difficult. With this discovery the computation time of the GMF that is plotted in Figure 4.34 is overestimated because the GMF cannot profit from the parameter space restriction as the remaining approaches. Nevertheless, the computation is much too slow for real-time applications. Because of the implementation characteristics of the GMF and the NCC, only the MGHT, the SBM, the LSPR, the SAD, PM, and PQ can be compared objectively when using ΔT as criterion. Here, the MGHT shows the smallest time increase ($\Delta T = 56\%$), which indicates an advantage of the MGHT over the remaining approaches if the parameter space increases. The time increase of the SBM extended by the LSPR is lower than that of the SBM because of the constant part of the LSPR. Also, the computation time of PM only increases moderately. For most methods, a similar behavior is obtained when searching for the rotated IC (see Figure 4.35(b)). Also here, the MGHT seems to be the method that is most suited when dealing with large parameter spaces. The corresponding time increase is only 29% in this case. The computation times of PM and PQ and the associated values for ΔT are significantly higher than in the case of the shifted IC. The reason for the totally different computation times when using the two sequences is the automatic computation of the coarse grain limits (cf. Section 4.4.1.5). During the first sequence that uses the shifted IC the grain limit of both methods was automatically set to 3.72, while during the second sequence that uses the rotated IC the grain limit was automatically set to only 2.92. This results in an increased complexity. There is no obvious reason for this difference, because the object was the same in both cases. Experiments have shown that the automatic computation of the grain limit may result in a completely different value if the ROI of the model image is shifted by just one pixel without changing the number of edge points within the region.

4.4.5 Conclusions

The aim of the performance evaluation in the framework of this dissertation was to select the approach that is best suited to serve as a module within the approach for recognizing compound objects and that fulfills the stated requirements listed in Section 2.3. Therefore, in the following for each evaluated recognition approach the most important results are summarized.

There are two main reasons why the HD is unsuited regarding the stated criteria. Its strong trend to return false positive matches leads to a very low robustness against clutter. Furthermore, there is no easy way to refine the pose of the HD directly from the distance values by interpolation. No fair statements about the computation time can be made because of the mentioned implementation characteristics. As a positive property, the acceptable robustness against occlusions should be mentioned, which, however, is depreciated by the bad receiver operating characteristic.

The SAD also exhibits several drawbacks. A reasonable robustness against occlusions is only obtained in combination with increasing false alarm rates and dramatically long computation times, which make real-time applications impossible. Furthermore, it is in no way robust against changes in brightness. This could already be derived from the corresponding formula (4.3). Finally, the accuracy obtained by interpolating the score values is only mediocre.

In contrast, the robustness against changes in brightness of the NCC is higher because of its normalization. However, the low robustness against occlusions as well as the trend to false positive matches limit its applicability decisively. While the position accuracy is very high the returned object orientation is less accurate.

The GMF achieves the highest robustness of all approaches against both occlusions and changes in brightness. Unfortunately, the associated high false alarm rates and a computation time that reaches several seconds are substantial arguments against the use of the GMF.

PM shows a good compromise between robustness, accuracy, and computation time. However, its receiver operating characteristic is significantly worse than that of the MGHT and the SBM. The obtained accuracy is very high and the computation time is also satisfactory.

Although PQ is faster than PM, its receiver operating characteristic is worse. This must be attributed to its tendency to return false positive matches. Furthermore, the speed-up is won at the cost of a reduced accuracy. Especially, the position accuracy is the worst in the test.

Finally, the two developed approaches MGHT and SBM both show a very well balanced behavior regarding robustness, accuracy, and computation time. The receiver operating characteristics of both are the best of all approaches in the test. They combine highest robustness against occlusions with highest robustness against clutter. Also, the robustness against changes in brightness of the SBM is only beaten by the GMF. If smaller score values are tolerated the MGHT also shows high robustness against changes in brightness, which, however, cannot keep up with the true invariance of the similarity measure used within the SBM. Both methods show highest position accuracy in the test. The accuracy in orientation of the SBM is already high and can be further increased by the LSPR. The MGHT shows lower orientation accuracy, which, however can keep up with the SAD, NCC, and PQ, and could be further improved by using the LSPR. Finally, the computation time of the MGHT and the SBM on average is the fastest of all approaches in the test. Here, the MGHT seems to be best suited for extending the parameter space to further dimensions.

Aside from these conclusions, it should be pointed out that some of the results might change if, for example, other implementations of the approaches, other parameter constellations, or other image sequences are chosen. Therefore, the presented evaluation is more of a qualitative nature than of a quantitative one. Nevertheless, the results are very objective and help to find the optimum approach for a specific application.

The approach for recognizing compound objects, which will be introduced in the following chapter, has a modular design and is therefore independent of the used module for recognizing rigid objects. For the implementation of the approach for compound objects the SBM was selected because, apart from the argument that the SBM is already part of a commercial software and thoroughly tested, its true invariance against changes in brightness is a second argument to prefer the SBM to the MGHT. The advantage of the MGHT when dealing with higher dimensional parameter spaces is less important since only rigid motion is considered in this dissertation.

Chapter 5

Recognition of Compound Objects

This chapter describes the novel approach for recognizing compound objects. At first, a review of the respective literature is given (Section 5.1). After a coarse description of the approach (Section 5.2), the single steps are described in detail (Sections 5.3–5.5). Finally, several examples show the high performance of the new approach (Section 5.6).

5.1 Previous Work

Approaches dealing with the recognition of compound objects are rarer to find in literature than those dealing with rigid objects. In the following, the most important approaches will be described.

A prominent class of object recognition methods deals with constrained objects in general and articulated objects in particular. Although approaches of this kind mainly deal with the recognition of 3D objects it is worth to include them in the present review because some of the proposed ideas might be also useful in 2D. A constraint object is an object that is composed of a set of rigid object parts. The constellation of the parts is restricted by constraints of an arbitrary type. In articulated objects these constraints are special kinematic constraints, e.g., rotational or translational joints.

Most methods that deal with the recognition of articulated objects like (Grimson 1989, Lowe 1991, Li and Lee 2002) are too restrictive for the recognition of compound objects because the presence of joints in compound objects cannot be assumed in general. Several methods try to recognize articulated objects by decomposing the object into its parts and estimating the pose of each part separately. In a subsequent step the constraints between the parts are checked (Grimson 1987, Grimson 1989, Kratchounova et al. 1996). Although these approaches are attractive because of their simplicity, the performance suffers: the information about the constraints is not exploited during the recognition process. Also, solving the correspondences would be computationally expensive because of its combinatorial character. In (Hel-Or and Werman 1994a, Hel-Or and Werman 1994b), an approach is presented that covers articulated or other more general constrained models. Here, the process of solving the correspondence problem is fused with the process of checking the constraints. This is done in a recursive process where the pose of the current object part is predicted using a Kalman filter. The prediction is based on the poses of the parts for which the correspondence problem has already been solved. The predicted pose is then compared to all matches of the current part by computing a distance measure. The match with minimum distance is selected. The whole process is repeated for all object parts. By successively selecting the best match for all object parts, the computational effort that is associated with the correspondence problem is reduced considerably. However, in some cases it might be dangerous to fix the pose of the current part in an early stage. Especially, if the prediction relies on the poses of only a few parts this may cause problems. The major drawback of these methods is that the recognition process itself ignores the information about the constraints between the parts. Thus, approaches of this kind assume that all possible matches of all object parts are already given as input data.

The approach presented in (Li and Lee 2002) is able to recognize articulated objects. Each rigid object part is represented by an attributed graph. Also from the search image one attributed graph is generated and automatically partitioned into small subgraphs. In general, the subgraphs do not coincide with the object parts. Graph matching is then performed between one of the subgraphs and the graphs that represent the object parts. The graph matching is performed using a Hopfield network. Alternatively, other optimization techniques can be applied to the problem of graph matching, e.g., genetic algorithms (Suganthan 2002). The matching results are stored and a different subgraph is selected. Another matching is performed and the result is added to the previous ones and so on. A decision on the final result is made by interpreting all accumulated results. The obtained poses are clustered to eliminate spurious matches. By accumulating the results of several matches the robustness against noise, occlusions, and ambiguities is increased. Unfortunately, the graph matching process is very time consuming, and hence unsuitable for real-time object recognition. Furthermore, information about the constraints is not exploited during the recognition process but only considered afterwards.

In (Felzenszwalb and Huttenlocher 2000), an object is represented by a collection of parts arranged in a deformable configuration. The deformable configuration is represented by spring-like connections between pairs of parts. The globally best match in an image is found. This is done by minimizing an energy function that takes into account both the “spring” forces between the parts and the match quality for each part. Thus, the approach is able to solve the correspondence problem efficiently. However, information about the relations between the parts is ignored during the recognition process itself. Hence, there is no speed-up in comparison to simply matching each part separately: it takes several seconds to find the object, which is too slow for real-time applications. Furthermore, the model must be set up manually, which prevents the practical use. Another drawback is that only the best match can be found, and hence the approach fails if more than one instance of the object is present in the image.

A hierarchical recognition of articulated 3D objects is presented in (Hauck et al. 1997). They assume that the pose of a static part is given and determine the poses of the remaining parts recursively. The relations between the object parts are represented by rotational or translational joints. By making use of the relations and already obtained information the efficiency of the recognition is increased. For this, the possible 3D poses of the remaining parts are successively predicted and projected into the image using a hierarchical representation of the object parts. Thus, self-occlusions of the object parts can be taken into account by eliminating possibly occluded image features from the recognition process. Additionally, the search space is restricted to the predicted poses, which increases the efficiency. However, the manual generation of the 3D model is complex and time consuming. Furthermore, only articulated objects can be handled by the approach. Moreover, the 3D pose of the static part must be known à priori in the camera coordinate system in order to correctly project the 3D poses of the remaining parts. Finally, the approach fails if object parts are undetectable because then the pose prediction is impossible. Nevertheless, the idea of the hierarchical representation together with the recursive search promises to increase the efficiency also of 2D object recognition approaches.

Another category of approaches deals with the recognition of elastic, flexible, or deformable objects (Jain et al. 1996, Pilu and Fisher 1997, Lee and Street 2000, Duta et al. 2001, Sclaroff and Liu 2001, Belongie et al. 2002). These approaches are mainly used to recognize natural objects that slightly change their appearance. Often this change in appearance cannot be modeled by a global transformation but requires to take into account local deformations. For example, in medical imaging these approaches can be applied to the registration of MRI (magnetic resonance imaging), CT (computed tomography), PET (positron emission tomography), FMRI (functional magnetic resonance imaging), ultrasound imaging, etc. Apart from human organs, a recognition of plants or animals is facilitated by these methods. However, these methods fail to model compound objects, which do not show real deformations because their object parts themselves are rigid.

Another approach for recognizing deformable objects is given in (Gavrila and Philomin 1999). Here, a detection method for objects with varying shape is described. The method uses a shape hierarchy to capture the variety of object shapes. It is based on the idea that similar object shapes can be grouped together and represented by a prototype shape. Thus, in the offline phase a hierarchy is computed from a set of training shapes using stochastic optimization techniques. In the online phase, matching is performed with this prototype, rather than with the individual shapes. This is done by involving a simultaneous coarse-to-fine approach over the shape hierarchy and over the transformation parameters. To increase the performance, the existing set of training shapes can be extended with generated “virtual shapes” (Gavrila and Giebel 2001). This improves the representational capability of the prototype shapes. Approaches of this kind are suitable for compound objects that consist of only a few parts with only small relative movements. The relative movements could be seen as shape variations of the compound object. However, more object parts with even moderate movements would lead to a combinatorial explosion of the number of required models.

5.2 Strategy

The approach for recognizing compound objects proposed in this dissertation is based on a hierarchical model representation of the compound object (Ulrich et al. 2002). The strategy behind the approach can be split into three stages. Figure 5.1 displays a flowchart for each stage in a very condensed form. In the first stage the hierarchical model is trained based on several example images (see Figure 5.1(a)). Here, the rigid object parts of the compound object are extracted and the relations between single object parts are derived. It should be noted that the result of training the hierarchical model is not the hierarchical model itself: in fact, only the relations between the object parts are trained. The relations represent one essential component of the hierarchical model. This should be kept in mind during further discussions. The second stage incorporates the actual creation of the hierarchical model based upon the trained relations (see Figure 5.1(b)). Here, a distinguished *root part* is selected from all object parts using certain criteria. Furthermore, the optimum hierarchical search strategy is found, with the root part at the top of this hierarchy. Finally, in the third stage the hierarchical model is used to find the compound object in the search image (see Figure 5.1(c)): In order to achieve real-time capability, only the root part is searched within the full parameter space. Whereas, the remaining parts are searched with respect to each other only within a restricted parameter space according to the extracted relations. Consequently, the offline phase includes the first two stages, while the third stage represents the online phase. In this section, a coarse description of the three stages will be given in order to introduce the underlying strategy. The description will then be elaborated on a finer level of detail in the following three sections.

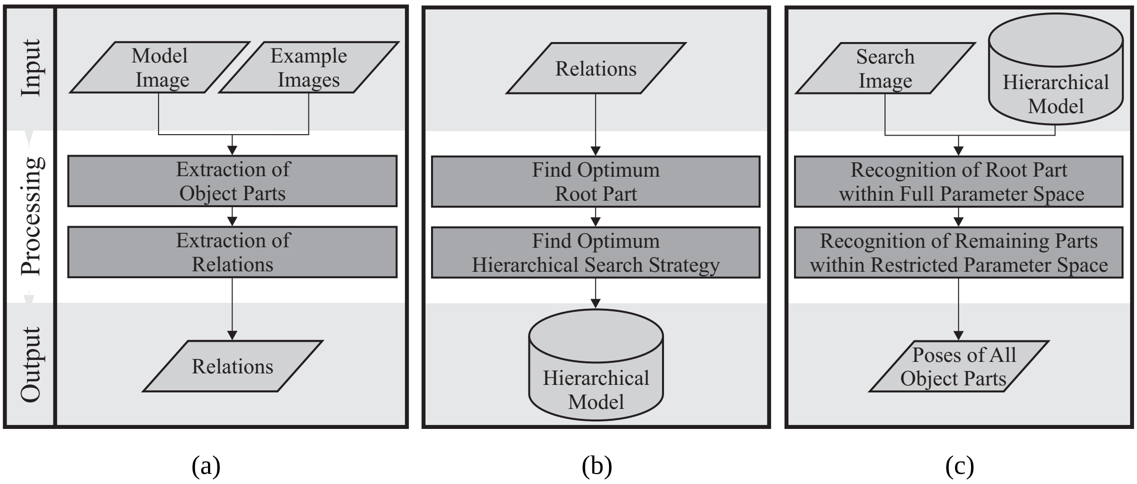

Input
Processing
Output
Model Image
Example Images
Extraction of Object Parts
Extraction of Relations
Relations
(a)
Relations
Find Optimum Root Part
Find Optimum Hierarchical Search Strategy
Hierarchical Model
(b)
Search Image
Hierarchical Model
Recognition of Root Part within Full Parameter Space
Recognition of Remaining Parts within Restricted Parameter Space
Poses of All Object Parts
(c)

Figure 5.1: Condensed flow charts represent the three stages of the proposed approach for recognizing compound objects. The offline phase includes training (a) and creating (b) the hierarchical model. The object recognition (c) represents the online phase.

In Figure 5.2 the detailed flowchart of the algorithm to train the hierarchical model is presented. A model image I^m, in which the compound object is defined by a ROI, and several example images I_i^e, $i = 1, \dots n^e$

represent the major input data to the algorithm. To enhance the illustrative power of the following explanations, an artificial example is used, which further clarifies the intermediate processing steps and results. Figure 5.3 shows the input data of the example. All images are of size 512 × 512. Taking a closer look at the example images two inherently connected observations can be immediately made. Firstly, the compound object (man) obviously consists of several rigid object parts (head, upper body, two arms, two hands, two legs, and two feet). Secondly, the parts move with respect to each other to different degrees. The objective of the first offline phase, i.e., the training of the hierarchical model, exactly is to simulate the human visual perception. This process includes extracting the rigid object parts together with the relations that describe the mutual movements.

Therefore, analyzing the process of visual perception may help to reproduce the human observations. For the following discussions it is essential to strictly distinguish between components and parts: One object part consists of one or more components. The components that belong to the same object part do not show any relative movements, while different parts move with respect to each other. Assume that there is no prior knowledge about the compound object. Consequently, if one focuses on the model image and disregards the example images, one is unable to determine the rigid object parts since no information about the movement is available. However, it is possible to decompose the object into small components. In this example, the following components are perceptible: hat, face, two arms, two hands, two legs, two feet, square margin of the upper body, and six components, one for each letter printed on the upper body. This means, that in this example the decomposition is done on the basis of image regions that exhibit a homogenous gray value. When extending the field of view to the example images, a human tries to match the corresponding components of the model image in the example images. Finally, the components that do not move with respect to each other in all example images are unconsciously and immediately merged into rigid object parts by the human brain.

With this knowledge it is possible to model the extraction of rigid object parts as shown in Figure 5.2. At first, the domain of the model image defined by the ROI is initially decomposed into small components. The resulting n^c components are described by n^c ROIs that refer to the model image. For each of the components a rigid model is generated using an arbitrary suitable object recognition approach. The pose of each component is then determined by the recognition approach in each example image and stored in the *component pose matrix* of size $n^e \times n^c$. From the component pose matrix the components that do not show any relative movement in all example images are determined. The rigid object parts can then be extracted by merging the ROIs of the respective components. Also, for the resulting $n^p (\leq n^c)$ object parts, rigid models are generated and used to determine the pose of each part in each example image in an analogous way as for the components. This results in the *part pose matrix* of size $n^e \times n^p$. Finally, the relations between the parts can be extracted by analyzing the part pose matrix. The relations are stored in the square *relation matrix* of size $n^p \times n^p$. I.e., for an arbitrary pair of object parts (p, q) the relative movement of part q with respect to part p is stored in row p and column q. The relation matrix together with the ROIs of the extracted model parts represent the output data of the training process.

As an example, Figure 5.4 shows the relations of the left arm and the upper body, respectively, to all other object parts, i.e., the two corresponding rows of the relation matrix are visualized. Hence, the relative movements of the object parts with respect to the left arm and the upper body are displayed. For visualization purposes, these movements are projected back into the model image. The object parts are symbolized by their reference points. The relative positions of the part's reference points is symbolized by enclosing rectangles, and the relative orientations by circle sectors. A relative orientation of 0° is localized at the "3" of a clock's dial and the center of the clock's dial is visualized at the mean position of the respective part. For example, when looking at Figure 5.4(a) one can see that the relative movement of the left hand with respect to the left arm is smaller than the relative movements of the other parts. Furthermore, the relative movements with respect to the upper body displayed in Figure 5.4(b) on average are smaller than the movements with respect to the left arm.

In the second stage of the offline phase the information trained in the first stage is used to create the hierarchical model. The process is illustrated in the flowchart of Figure 5.5 in a generalized form. The model image and

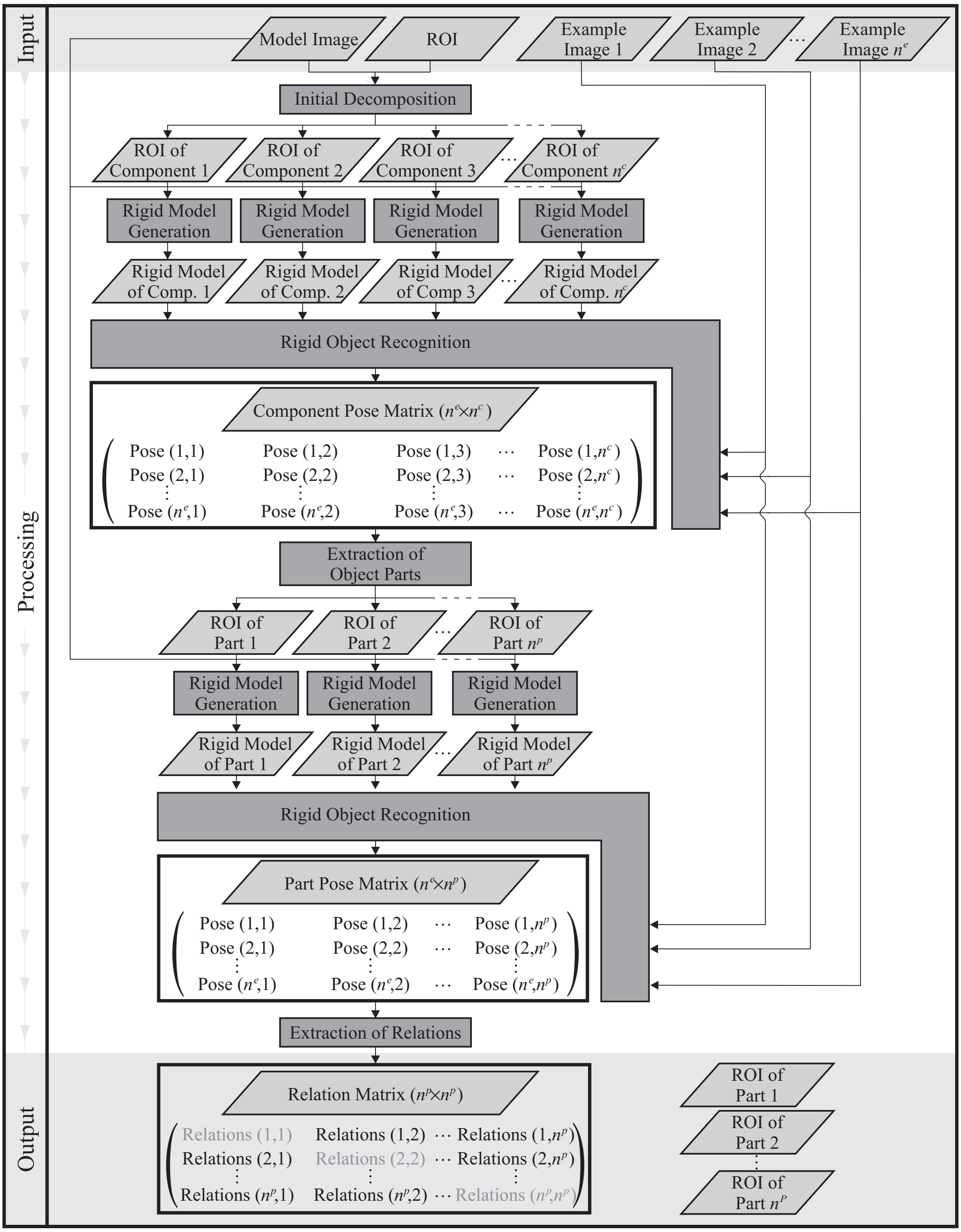
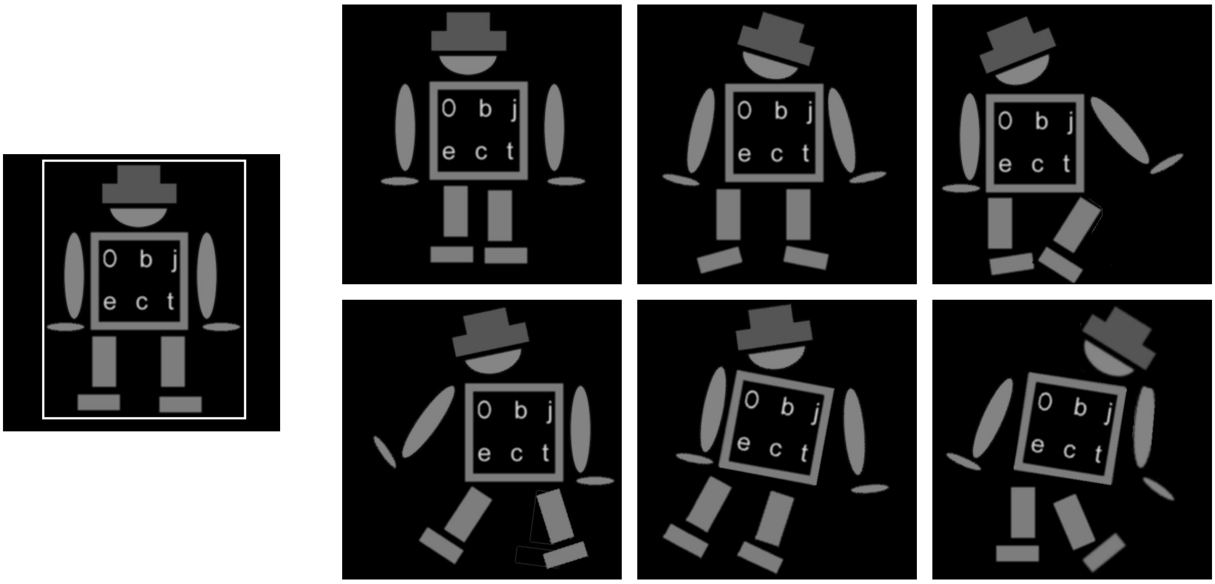

Input
Model Image
ROI
Example Image 1
Example Image 2
Example Image n^e
Initial Decomposition
ROI of Component 1
ROI of Component 2
ROI of Component 3
ROI of Component n^c
Rigid Model Generation
Rigid Model of Comp. 1
Rigid Model of Comp. 2
Rigid Model of Comp 3
Rigid Model of Comp. n^c
Rigid Object Recognition
Component Pose Matrix (n^e×n^c)
Pose (1,1) Pose (1,2) Pose (1,3) ⋯ Pose (1,n^c)
Pose (2,1) Pose (2,2) Pose (2,3) ⋯ Pose (2,n^c)
Pose (n^e,1) Pose (n^e,2) Pose (n^e,3) ⋯ Pose (n^e,n^c)
Processing
Extraction of Object Parts
ROI of Part 1
ROI of Part 2
ROI of Part n^p
Rigid Model Generation
Rigid Model of Part 1
Rigid Model of Part 2
Rigid Model of Part n^p
Rigid Object Recognition
Part Pose Matrix (n^e×n^p)
Pose (1,1) Pose (1,2) ⋯ Pose (1,n^p)
Pose (2,1) Pose (2,2) ⋯ Pose (2,n^p)
Pose (n^e,1) Pose (n^e,2) ⋯ Pose (n^e,n^p)
Extraction of Relations
Output
Relation Matrix (n^p×n^p)
Relations (1,1) Relations (1,2) ⋯ Relations (1,n^p)
Relations (2,1) Relations (2,2) ⋯ Relations (2,n^p)
Relations (n^p,1) Relations (n^p,2) ⋯ Relations (n^p,n^p)
ROI of Part 1
ROI of Part 2
ROI of Part n^P

Figure 5.2: Training the hierarchical model (first stage of the offline phase)

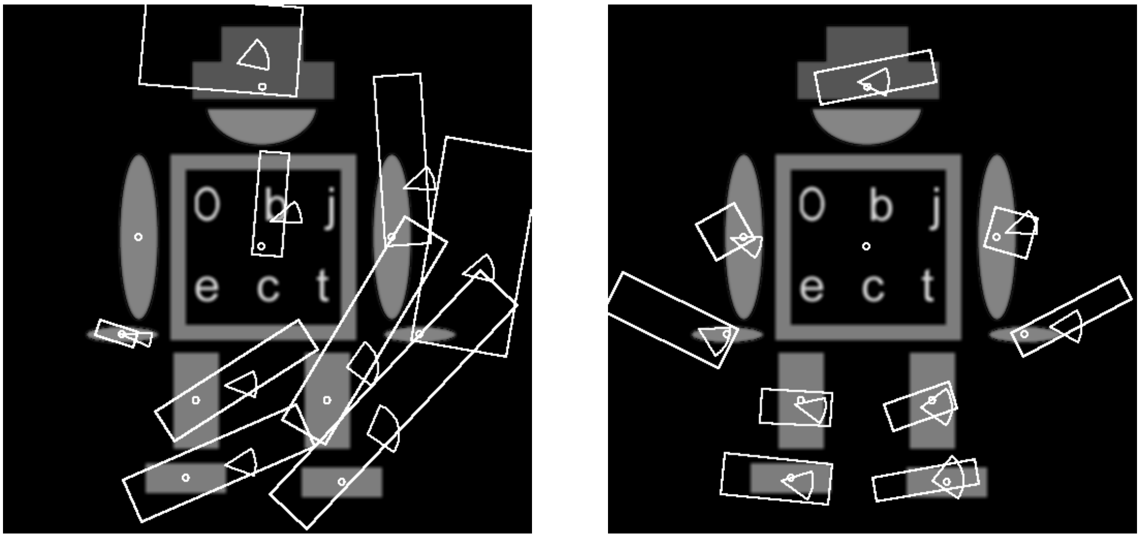
the output data of the training are passed as input data to the process. Because the orientation ranges of the object parts in the search image do not need to coincide with the orientation ranges used during training, again rigid models that cover the desired orientation range are generated. In order to find an appropriate root part, the rigid models of all parts are analyzed using certain criteria, which will be introduced later in this work. Based on the root part and the relations, an optimum hierarchical search strategy can be found by minimizing the search effort in the online phase. Here, it is assumed that in the online phase the extent of the relative part



(a) Model image

(b) Example images

Figure 5.3: Input data of the artificial example. A rectangular ROI defines the compound object in the model image (a). Six example images show the movements of the respective object parts (b).



(a)

(b)

Figure 5.4: Relations visualized for the left arm (a) and the upper body (b) in the model image. The object parts are symbolized by their reference points (small circles). The relations are visualized as rectangles (relative positions) and circle sectors (relative orientations).

movements is less or equal than the extent of the relative part movements represented in the example images. If this assumption fails, the automatically derived relations must be extended manually by appropriate tolerance values. Then, the relations between parts p and q represent the search effort that must be spent to search part q relative to part p under the assumption that the pose of part p is already known. For example, if the poses of the left arm and of the upper body in the search image are known it would be more efficient to search the left hand relative to the left arm instead of searching it relative to the upper body (see Figure 5.4). Finally, the hierarchical model comprises the rigid models of all object parts, the relation matrix, and the optimum hierarchical search strategy.

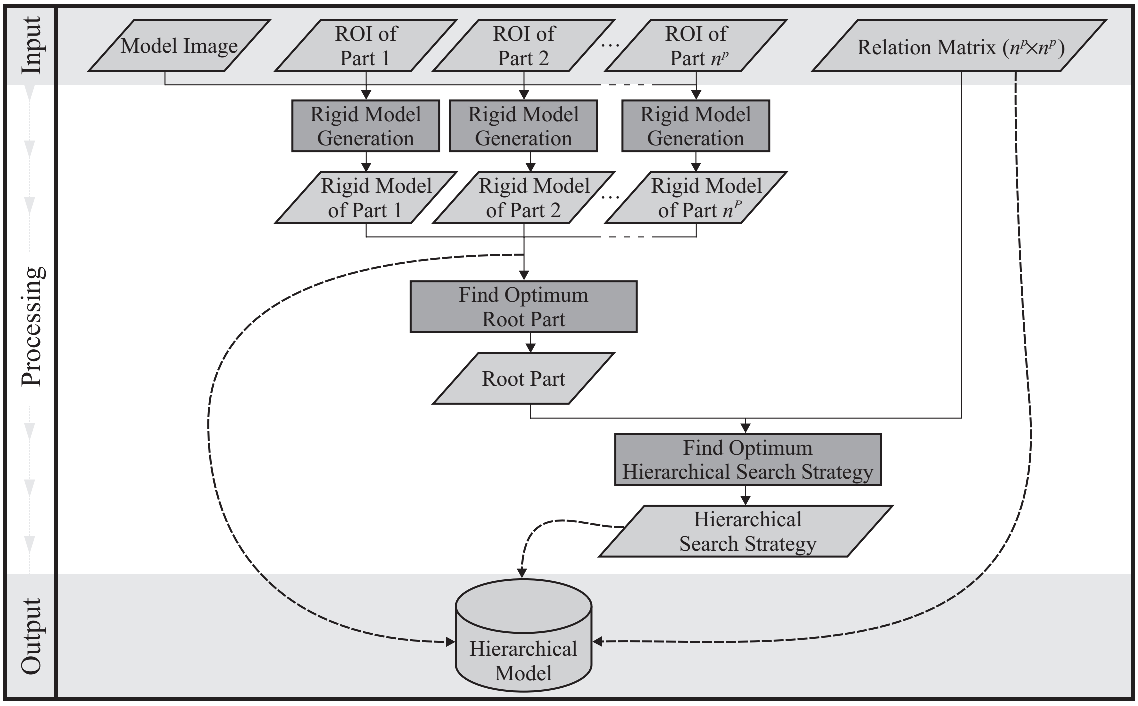
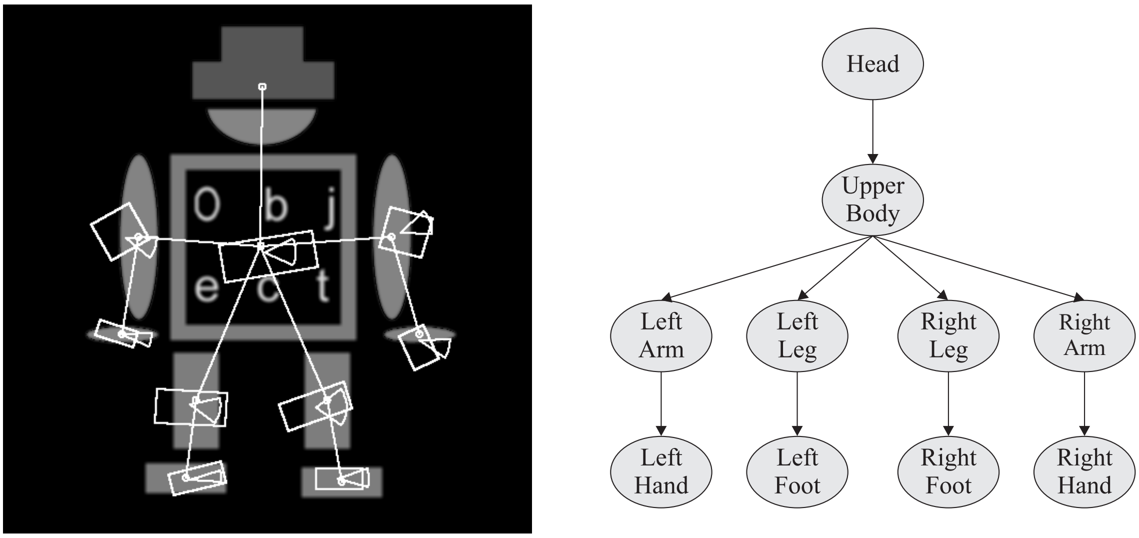


Figure 5.5: Creating the hierarchical model (second stage of the offline phase)



(a) Hierarchical model

(b) Search Tree

Figure 5.6: The hierarchical model comprises the rigid models of all object parts, the relations between the parts, and the hierarchical search strategy (a), which is represented by a hierarchical search tree (b).

Figure 5.6 visualizes the resulting hierarchical model for the example case. It uses the head as the root part. Assuming a minimum search effort in the future search image, it further searches the upper body relative to the head, searches the two arms, and the two legs relative to the upper body, and searches the hands and the feet relative to the arms and the legs, respectively. To valuate the overall search effort, the relations between the parts that are adjacent in the search tree are visualized, i.e., they are connected by a edge in the tree. Thus, during the online phase the reference points of the respective object parts must only be searched within the

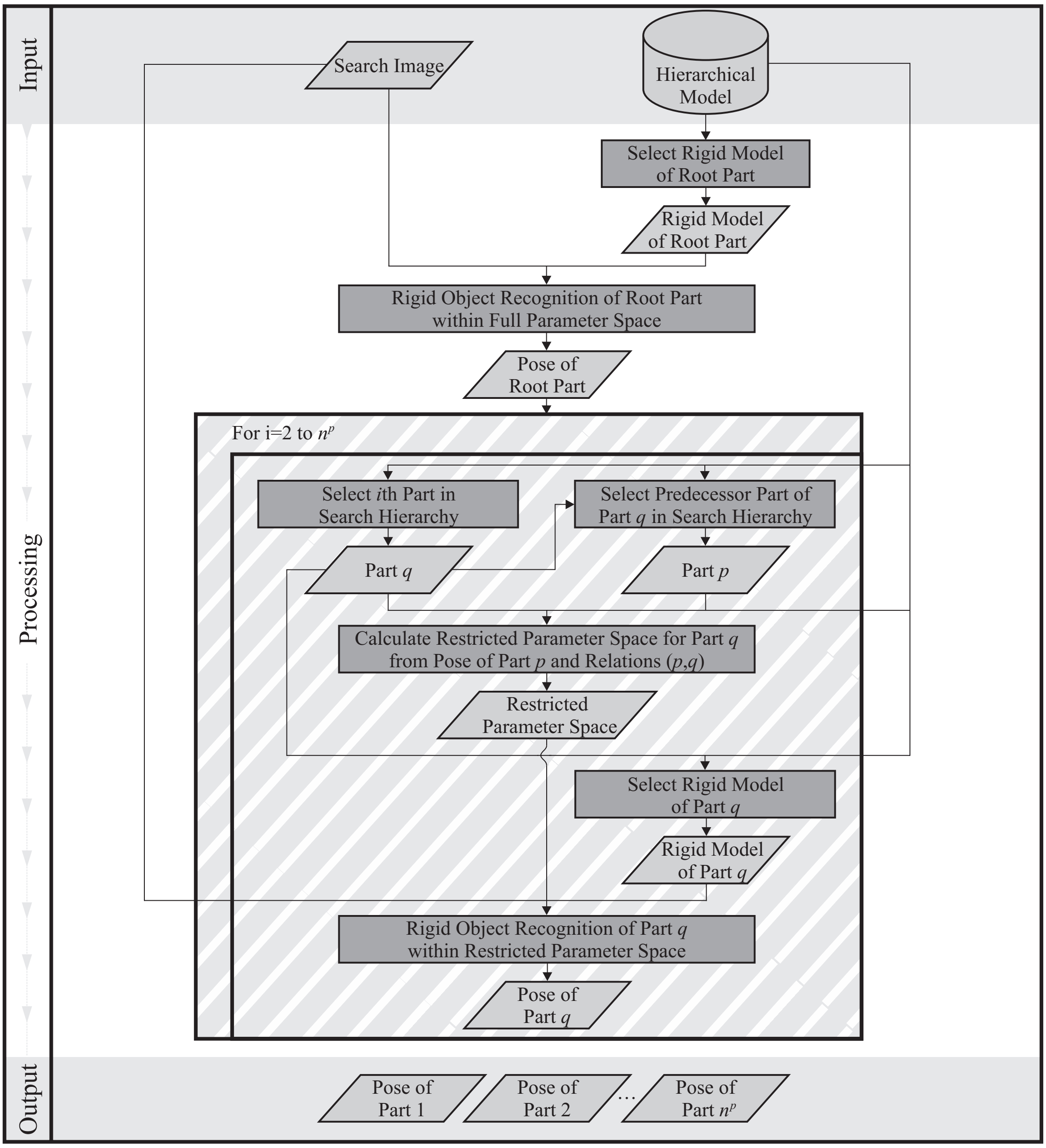
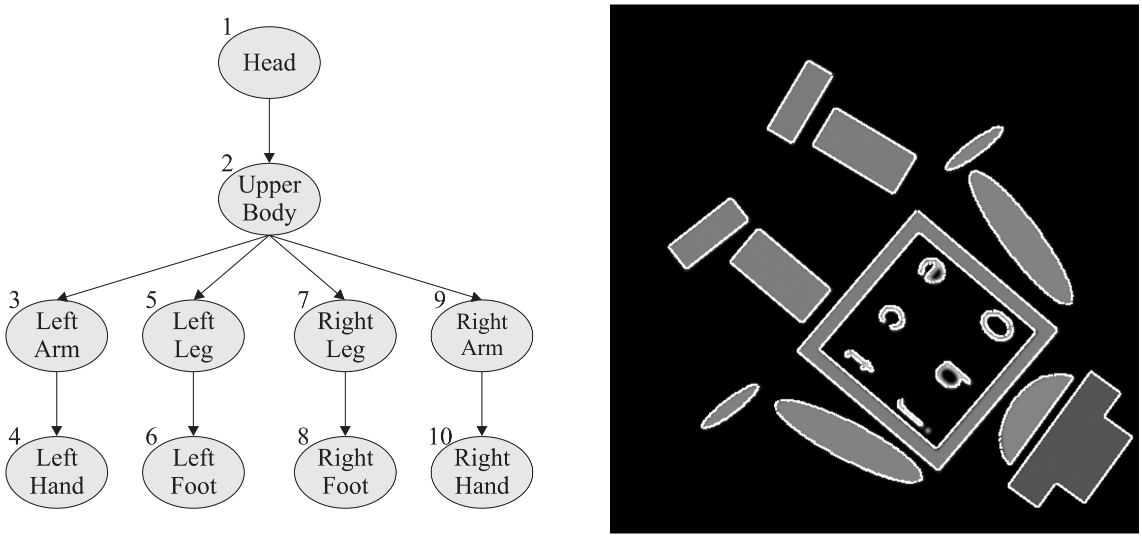

Input
Processing
Output
Search Image
Hierarchical Model
Select Rigid Model of Root Part
Rigid Model of Root Part
Rigid Object Recognition of Root Part within Full Parameter Space
Pose of Root Part
For i=2 to n^p
Select ith Part in Search Hierarchy
Select Predecessor Part of Part q in Search Hierarchy
Part q
Part p
Calculate Restricted Parameter Space for Part q from Pose of Part p and Relations (p,q)
Restricted Parameter Space
Select Rigid Model of Part q
Rigid Model of Part q
Rigid Object Recognition of Part q within Restricted Parameter Space
Pose of Part q
Pose of Part 1
Pose of Part 2
...
Pose of Part n^p

Figure 5.7: Object recognition (online phase)

small rectangular regions and within an orientation range visualized by the circle sectors.

Finally, the process of object recognition is displayed in the flowchart of Figure 5.7. Analogously to the online phase of rigid object recognition, the search image and the model — which now is a hierarchical model — are passed as input data to the algorithm. At first, the rigid model of the root part is selected from the hierarchical model and used to determine the pose of the root part in the search image. Since no prior knowledge about the pose is available the rigid object recognition approach must search the root part by scanning the full parameter space of positions and orientations.

Once the root part is found, the remaining $n^p - 1$ parts can be searched within a restricted relative search space. Thus, for each part q the predecessor part p in the search tree is selected. Assume, for example, that a depth-first search is applied to the search tree presented in Figure 5.8(a). After the pose of the head is determined, the next part to search ($i = 2$) would be the upper body, i.e., q = “Upper Body”. The associated predecessor



(a) Depth-first search

(b) Search image and found object instance

Figure 5.8: A depth-first search is applied to the search tree. In (a) the search order is indicated by numbers. In (b) an example search image and the corresponding found object instance is displayed. The poses of the individual object parts are visualized by superimposing the edges of the parts at the returned pose in white.

part in the search tree is the head, and hence p = “Head”. The parameter space that must be scanned by the recognition approach to search part q is defined by the pose of part p and the relation between the parts p and q. The rigid model of part q is selected from the hierarchical model and used to determine the pose of part q within the restricted parameter space. The whole process is repeated for each part, finally obtaining the poses of all object parts. An example search image of size 512 × 512 and the corresponding found object instance is displayed in Figure 5.8(b). It should be noted that it is not necessary that the absolute orientation of the object in the search image is covered within the example images, since only relative movements between object parts are trained.

To give an impression of the advantage when using the proposed hierarchical model, the recognition time for this example was 20 ms on a 2 GHz Pentium 4. In contrast, the brute-force method that would search all parts in the entire search space independently from each other would take 310 ms (using the SBM in both cases). The second obvious advantage should also be pointed out here: because of the inherently determined correspondence, which is provided by the hierarchical model, the returned match of the compound object implicitly covers a topologically sound representation. In contrast, when searching the parts independently, it is not immediately possible to distinguish between the matches of the left and the right arm, for example. Furthermore, if several object instances are present in the image, it is hard to assign a match of a certain object part to the correct instance of the compound object.

Although the basic idea of the approach seems to be very simple, several difficulties that are not obvious at first glance occur. They will be discussed in the following sections together with the detailed explanation of the previously introduced steps.

5.3 Training the Hierarchical Model

5.3.1 Initial Decomposition

In the first step, the compound object must initially be broken up into small components. The condition the initial decomposition must fulfill is that each object part must be represented by at least one component.

Otherwise the algorithm is unable to find the rigid object parts automatically since only a merging of components but no further splitting is provided by the algorithm. Therefore, an over-segmentation, which leads to a high number of small components, is strongly desirable. Unfortunately, very small components may fail the property of being unique or even seldom, which is a generally demanded quality for a feature (cf. (Förstner and Gülch 1987)). This makes it difficult to determine the pose unambiguously during the training phase. However, this problem can be solved by the proposed approach, as will be shown later.

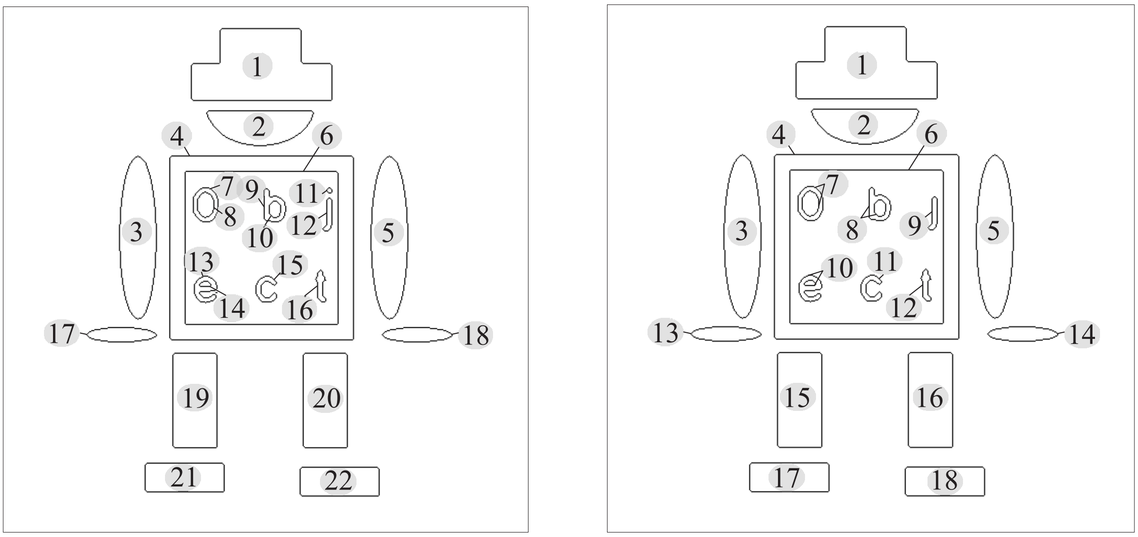
Several decomposition and grouping methods that can be found in literature are suitable for the task of initial decomposition. In general, grouping means the search for closely related primitives in the image. Gestalt psychology has uncovered a set of principles that guide the grouping process in the visual domain (Koffka 1935, Wertheimer 1938, Rock and Palmer 1990). These principles are based on different criteria that must be satisfied by related primitives. Such criteria can require that related primitives exhibit similar properties (e.g., size, color, shape) or that certain relations between the primitives are fulfilled (e.g., proximity, connectivity, parallelism, symmetry, good continuity). Although it has been proven that these principles indeed work, a satisfactory understanding of how they operate still needs to be found. Computer vision has taken advantage of these principles, e.g., in the field of perceptual organization and grouping (Ullman 1979, Marr 1982, Witkin and Tenenbaum 1983, Lowe 1985). In photogrammetry, grouping is applied in the extraction of various objects from aerial imagery. Only to mention a few examples, grouping is utilized for runway detection in airport scenes (Huertas et al. 1990), building detection (Lin et al. 1994), and road detection (Steger et al. 1997, Baumgartner 1998, Wiedemann and Hinz 1999). Unfortunately, an optimum selection of criteria is not possible in general, but is highly correlated with the application task. However, the approach for recognizing compound objects must fulfill the demand to be general with regard to the type of object. Consequently, without additional knowledge about the object no optimum grouping criteria or combination of criteria that could be used to extract components from the model image are available.

Nevertheless, for the purpose of decomposition a method is proposed that is on the one hand general enough to work with most objects and on the other hand still results in meaningful components. This method solely uses the connectivity and proximity criteria, where the edges in the model image serve as primitives. For this, edges are extracted within the ROI in the same way as it is done during the model generation of the MGHT or the SBM (cf. Section 4.2.2). The connected regions of the resulting edges are computed using a standard image processing algorithm (Jähne 2002). They represent the component hypotheses. Connected regions that are smaller than a certain threshold (20 edge pixels has proven to be a suitable value) are eliminated in order to avoid meaningless components that must be attributed to image noise or that are generally hard to identify. To apply the second criterion of proximity, for a certain component hypothesis the fraction of edge pixels that only have a small distance (e.g., 5 pixels) to another component hypothesis is computed. If this fraction is high enough (e.g., 50%) the two component hypotheses are merged. The attentive reader may see the correspondence between this fraction and the forward fraction used in the Hausdorff distance (cf. equation (4.9) in Section 4.1.1.2). This step is repeated iteratively until no more components fulfill the criterion for merging.

In Figure 5.9(a) the result of the edge extraction with the subsequent computation of connected regions is shown. In this case 22 hypotheses are generated. Since some pairs of components are closely neighbored they are assumed to form one component. Furthermore, the dot on the letter “j” is too small to be able to represent a meaningful component and is therefore eliminated. The final result, which contains 18 components, is shown in Figure 5.9(b).

The proposed strategy assumes that the edges in the model image of two different rigid object parts are neither connected nor close to each other in the above specified sense. While this assumption is true for a multitude of compound objects, its validity cannot be guaranteed in general. This is why the result of the above described method must be at least validated by user interaction and possibly substituted by a completely manual definition of the components. A manual definition can be done by passing several ROIs to the training algorithm, one for each component.

Another possibility is to allow the user to introduce prior knowledge. E.g., the user could choose among



(a) Component hypotheses (b) Final Components

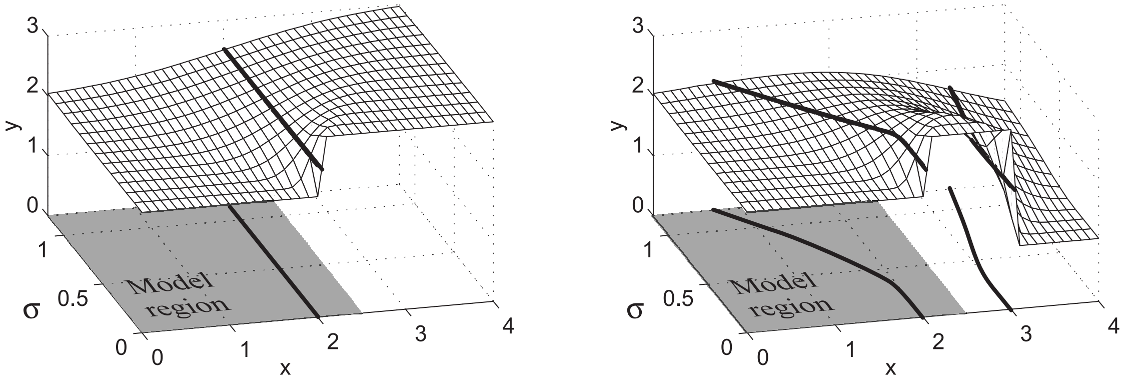
Figure 5.9: Component hypotheses are obtained by edge extraction and computation of connected edge regions (a). The hypotheses are numbered consecutively. The final components are obtained by eliminating small (11) and merging closely neighbored (7+8, 9+10, 13+14) hypotheses (b).

several offered decomposition and grouping methods, which can be easily incorporated in the approach. For example, in (Shapiro and Haralick 1979, Abe et al. 1996, Latecki and Lakämper 1999, Rosin 2000) 2D object shapes are partitioned into subparts by using the shape convexity as criterion. They base on the assumption that meaningful subparts exhibit a high convexity. Other partitioning schemes approximate the shape as the best combination of primitives such as polygons or constant curvature segments (Wuescher and Boyer 1991). Another class of approaches uses the curvature as splitting criterion (Hoffman and Richards 1984, Siddiqi and Kimia 1995, Singh et al. 1999). Here, the object shape is broken up into smaller parts at the points of minimum negative curvature. Approaches of this kind are able to split connected shapes. Hence, the requirement that the edges in the model image of two different rigid object parts must not be connected can be discarded.

5.3.2 Rigid Models for the Components

In the second step, for each component a rigid model is generated. The use of image pyramids in object recognition — so used in the SBM — is a desirable feature in order to achieve real-time performance. However, one has to take care of unfavorable scale-space effects that occur when dealing with image pyramids. A scale-space representation can be seen as a generalized form of image pyramids. It comprises a continuous scale parameter and preserves the same spatial sampling at all scales, i.e., no sub-sampling is performed. Furthermore, only the "optimum" Gaussian kernel is applied for low-pass filtering (Lindeberg 1994).

In scale-space the edges of the components are influenced by neighboring edges. This is uncritical in most cases when dealing with large objects, as is usually the case in rigid object recognition. In large objects there are still enough edges left that are not influenced by neighboring edges, and hence still provide a good match. However, some problems occur when dealing with small objects, like the components: the ratio of edge pixels of a component that is influenced by neighboring edge pixels can become high, i.e., the influence of neighboring edges increases. The principle of this scale-space effect is illustrated in Figure 5.10. The scale-space behavior of the 1D profile of an ideal step edge is shown. Here, x represents the profile direction, y the gray values, and σ the size of the Gaussian smoothing kernel, and thus the degree of smoothing. In the original image, i.e., for $\sigma = 0$, the subpixel precise edge position is at $x = 2.0$. Assuming that no further gray value changes appear in the neighborhood of the edge (see Figure 5.10(a)), the edge position remains unchanged even for successively stronger smoothing. Accordingly, when transferring this to the case of rigid model generation, the edge position would be the same on each pyramid level. Unfortunately, this assumption is not valid in general. If, for example, a rigid model is to be created from the component that represents the left arm (see component number 3 in Figure 5.9(b)) the outer square of the upper body (component number 4) influences the scale-space behavior, and hence the model creation on higher pyramid levels. In Figure 5.10(b) a second step edge is added to the profile and the edge position in scale-space is computed again. It can be seen that the higher the smoothing is, the larger the displacement of the original edge position becomes. This is uncritical if the left arm would always appear at the same relative pose with respect to the upper body because the edge displacement in scale-space would be the same in the model image and in all example images. However, in some example images the left arm moves with respect to the upper body, and hence the representation of the left arm in the pyramid of the example image differs from the representation in the model image. Especially, in the case of small components this may lead to a severe dissimilarity on higher pyramid levels between the information stored in the model and that represented in the search image. This possibly causes the recognition method not to find the component.



(a) Undisturbed model edge

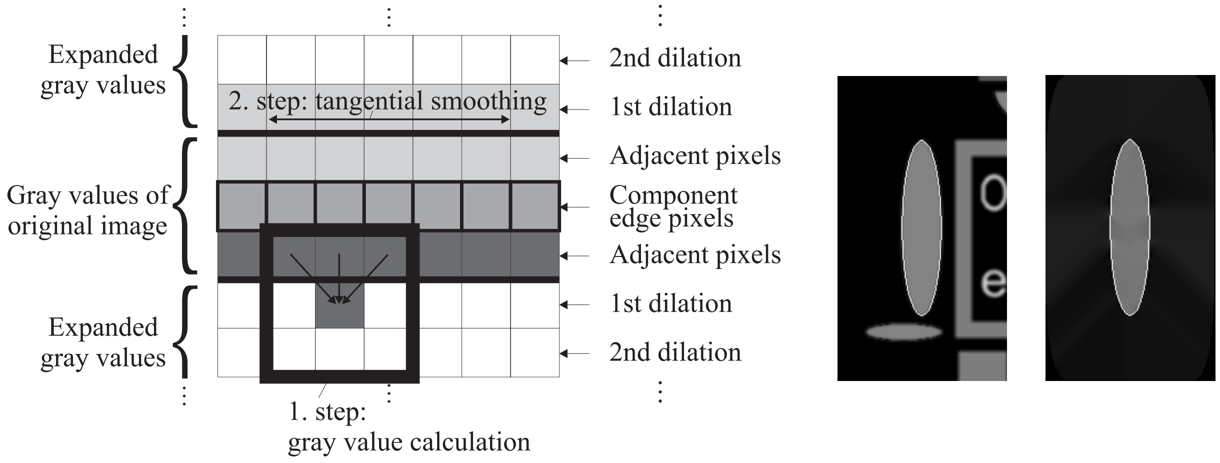
(b) Disturbed model edge

Figure 5.10: Scale-space behavior of an ideal 1D edge profile. For homogeneous gray values in the neighborhood, the edge position remains constant in scale-space (a). In contrast, the edge position is disturbed if gray value changes appear that are close to the edge, e.g., caused by a neighboring edge (b).

The problem could be avoided if no image pyramids were applied in the recognition method. However, the long computation times involved are definitely unsuitable for the practical use, even for the offline phase. Therefore, an alternative solution is proposed: the disturbing gray value inhomogeneities in the neighborhood of the model edges are eliminated. This is achieved by successively expanding the gray values of the model image that are adjacent to a component edge pixel in the direction of the edge gradient.

Figure 5.11(a) shows the principle of gray value expansion. The component edge region is dilated by one pixel, obtaining the pixels that are adjacent to the edge region. The gray values of the adjacent pixels are used as seed values that are expanded to the neighborhood. Thus, the gray values of the edge pixels and those of the adjacent pixels remain unchanged to ensure an extraction of the component edges that is the same as in the original image when applying the Sobel filter. For the gray value expansion, the already dilated edge region is successively dilated by one pixel. After each dilation, the gray values of the pixels that are added by the dilation are calculated from the already available gray values of the preceding dilation step. For this purpose, the gray values of the preceding dilation that are within a 3×3 neighborhood are averaged. Furthermore, to avoid artifacts that would lead to pseudo edges during model generation, a 1D smoothing of the newly calculated gray values is applied in a second step. In Figure 5.11(b) the part of the model image showing the left arm is displayed. The edges of the upper body and of the left hand would influence the model generation for the component representing the left arm. Figure 5.11(c) shows the result after applying the gray value expansion. The neighboring image structure, and hence the disturbing influences are eliminated.

To guarantee that all disturbing influences are eliminated, the size of the expansion must be chosen appropri-



(a) Principle of gray value expansion (b) Original image (c) Result

Figure 5.11: Successive gray value expansion is applied to the component edges in order to eliminate disturbing gray value inhomogeneities of the neighborhood (a). In (b) the left arm in the model image is shown. The extracted component edges are superimposed in white. After applying the gray value expansion to the component edges, the neighborhood is free of disturbing influences (c).

ately. If, for example, n^l pyramid levels are used in the proposed way, the gray values at both sides of the component edges must be expanded by at least 2^{n^l} pixels to fully eliminate the influence of all neighboring image structures.

The gray value expansion provides acceptable results in almost all cases. However, in rare cases pseudo edges are introduced by the algorithm. This may happen whenever two progressing gray value fronts of strongly different intensities collide. This often negligible effect can be weakened by stronger smoothing at the line of contact. Another problem arises in the case of strongly blurred edges. The expanded gray values would lead to lower edge magnitudes on higher pyramid levels in comparison to the original gray values. Both problems can be avoided by, alternatively, using other, more sophisticated approaches (Elder and Zucker 1998, Elder 1999). They explicitly model the edges by determining the edge location, magnitude, and blur scale of the underlying intensity change. After modifying the modeled edges, e.g., by deleting disturbing edges in the neighborhood, the image can be reconstructed from the remaining modeled edges.

Finally, two rigid models are built for each of the components. The first one is created based on the image in which the disturbing edges have been eliminated, the second one is based on the original image. This is because it is not known in advance whether the neighboring edges of a component belong to the same or to another rigid object part. However, eliminating edges that belong to the same rigid object part is critical because this again would lead to differences in the model and the example images. Using two models covers both cases, and hence leads to a higher probability that all components are found later in the example images. The consequence that the search for the components leads to duplicate matches is uncritical since it is compensated by the algorithm described in the following sections. In general, false positive or double matches are preferred during the training of the hierarchical model in comparison to missing a match.

5.3.3 Pose Determination of the Components

The rigid models of the components are used to search the components in each example image using the rigid object recognition approach. Because for each component two models are used, most components are at least found twice. Furthermore, one component may be found several times in an example image because of three reasons. Firstly, if a component exhibits rotation symmetry, like the components 3, 4, 5, 6, 7, 13, 14, 15, 16, 17, and 18 in Figure 5.9(b), it is found several times at similar positions but at different orientations. Secondly, if two or more components are similar to each other, like component pairs (3,5), (13,14), (15,16), and (17,18), in general each component is found at its correct pose as well as at the pose of the similar component(s). This also includes partial similarities, i.e., one component is similar to a fraction of another component. The component representing the left leg (15), for example, will be found four times in each example image because of its rotation symmetry of 180° and its similarity to the right leg (16). Thirdly, a component may also show similarities to image clutter, where, in general, smaller components are more likely to show similarities to clutter than larger components. Consequently, the result of the component search is not unique.

In the following section, an approach that solves the ambiguities by selecting the most likely pose for each component in each example image is presented. Here, the component poses that are able to represent the compound object in a least "deformed" way in comparison to the compound object in the model image are supposed to be more likely. The deformation is caused by the relative movement of the components. In (Ullman 1979), this problem is called the correspondence problem, where correspondence is the process that identifies elements in different views as representing the same object (component) at different times (in different images), thereby maintaining the perceptual identity of objects in motion or change. Thus, the movement of the components that is introduced when comparing their poses in the model image and in an example image is interpreted as apparent motion. Ullman (1979) proposes to solve the correspondence problem by minimizing the overall apparent motion. This is the basis for further considerations.

In the following, the exact solution for solving the ambiguities is presented. Upon closer examination, the problem of finding the most likely poses turns out to be a graph matching problem between a model graph G^m and a search graph G^s. The model graph represents the compound object in the model image, where the nodes in the graph represent the components, which are labeled with the associated component number. Furthermore, each node is linked to all other nodes by arcs, where an arc between two nodes is labeled by the relative position and orientation between the two associated components in the model image. Accordingly, G^s represents the poses of the component matches in an example image. In general, the number of nodes in G^s is higher than in G^m because of ambiguous matches. Furthermore, only the nodes in G^s that correspond to poses of distinct components are linked by arcs. Again the arcs are labeled by the relative poses of the components. Finally, the task of solving the ambiguities corresponds to finding G^m within G^s. The special class of graph matching that can be applied to this problem is often called *inexact subgraph isomorphism* (Kroupnova and Korsten 1997) or *error-correcting subgraph isomorphism* (Messmer 1996) in the literature. A comprehensive introduction to graph matching is given in (Messmer 1996). A *graph isomorphism* is a bijection between the nodes of two graphs. A *subgraph isomorphism* of G^m and G^s tries to find a subgraph $G^{s\prime}$ of G^s and a graph isomorphism from G^m to $G^{s\prime}$. An efficient algorithm is presented in (Ullmann 1976). Finally, an *error-correcting subgraph isomorphism* is more tolerant and is able to find an optimum subgraph isomorphism even if the two graphs are different to each other. The solution is often realized by introducing graph edit operations with associated costs. The goal is to find a sequence of edit operations with minimum cost that must be applied to G^m for a subgraph isomorphism to exist.

In the present case, because the number of nodes in G^s is higher than in G^m, a subgraph isomorphism must be found. Furthermore, the subgraph to find in G^s differs from the model graph G^m because the components may move with respect to each other. Hence, an error-correcting subgraph isomorphism is the right choice. The graph edit operations comprise modifying the relative poses between the components and deleting nodes for the case that a component is missing in the example image. Unfortunately, finding an error-correcting subgraph isomorphism is known to be NP-complete, and thus its time complexity is exponential in the number of nodes. In (Messmer and Bunke 1998), an algorithm is presented that reduces the computational load at the cost of exponentially increasing memory requirement. Both alternatives are not suitable for a practical solution of the ambiguities. Therefore, in the following an approach is presented that on the one hand shows only polynomial time complexity and on the other hand provides acceptable results. The idea is based on rating the single component matches, where matches that lead to less plausible configurations are penalized. The problem of solving the ambiguities can then be seen as uniquely assigning a match to each component such that the overall rating is maximized, while simultaneously considering certain constraints. For this, the component matches are represented as a bipartite graph (i.e., a graph with two subsets of nodes). The solution is then obtained by applying bipartite graph matching using linear programming.

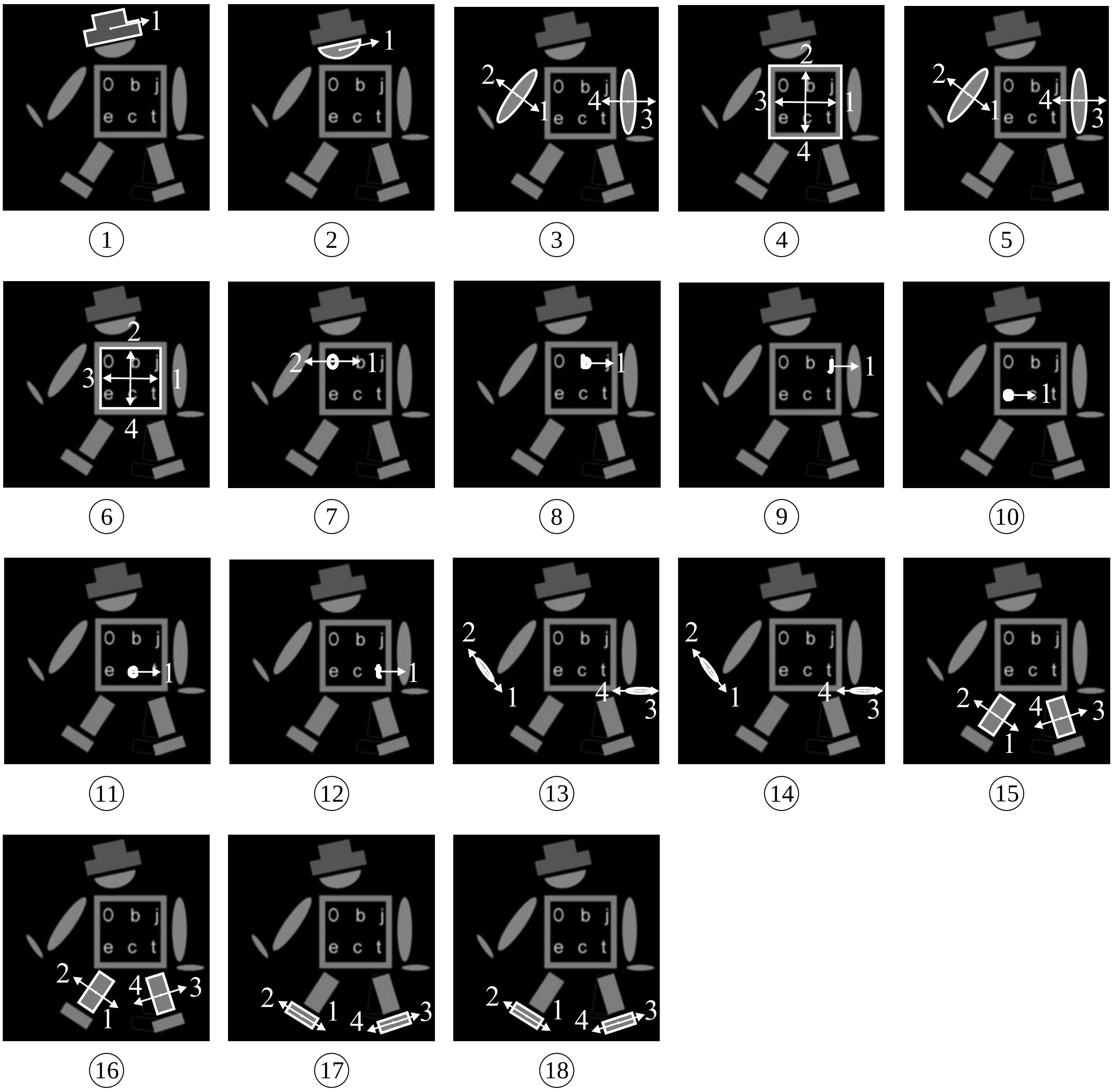


Figure 5.12: Matches of each of the 18 components in the fourth example image indicated by superimposed white edges. The orientations of the matches are indicated by white arrows. The match numbers are displayed by white numerals.

5.3.3.1 Rating of Matches

Let n^c be the number of components and $M_i = (\boldsymbol{o}_i^m, \varphi_i^m)$ the pose of component i in the model image, with position $\boldsymbol{o}_i^m$, orientation φ_i^m, and $i = 1, \ldots, n^c$. In the implementation using the SBM the orientation φ_i^m of all components is 0 since the orientation in the model image is taken as the reference. The component i may be found n_i times in the current example image because of symmetries, similarities, or clutter. The poses of the associated matches are $E_{i,k} = (\boldsymbol{o}_{i,k}^e, \varphi_{i,k}^e)$, with $k = 1, \ldots, n_i$. Figure 5.12 shows the matches $E_{i,k}$ of all 18 components in the fourth example image. Here, the duplicate matches that arise from the use of two models per component are neglected for illustrative purposes. However, theses matches can be treated in the same way like any other ambiguous matches that are introduced because of symmetries or similarities. For instance, component 15 (left leg) is found four times: at the correct pose (match 1), at the same position, but rotated by 180° (match 2), and at the position of the right leg at the two respective orientations (match 3, match 4).

The decisive point of the idea behind the approach for finding the most likely match of each component in the example image is based on rating the associated poses of all matches. The rating is performed by penalizing the matches that are less plausible by using a cost value. For this, the (unambiguous) poses of the components in the model image are taken as the reference, forming the reference component configuration. In the example image, a match receives a cost value that describes the quantified change in the component configuration that would be introduced when this match is assumed to be correct. This follows the above described principle of minimizing the overall apparent motion, where the apparent motion corresponds to the overall configuration change.

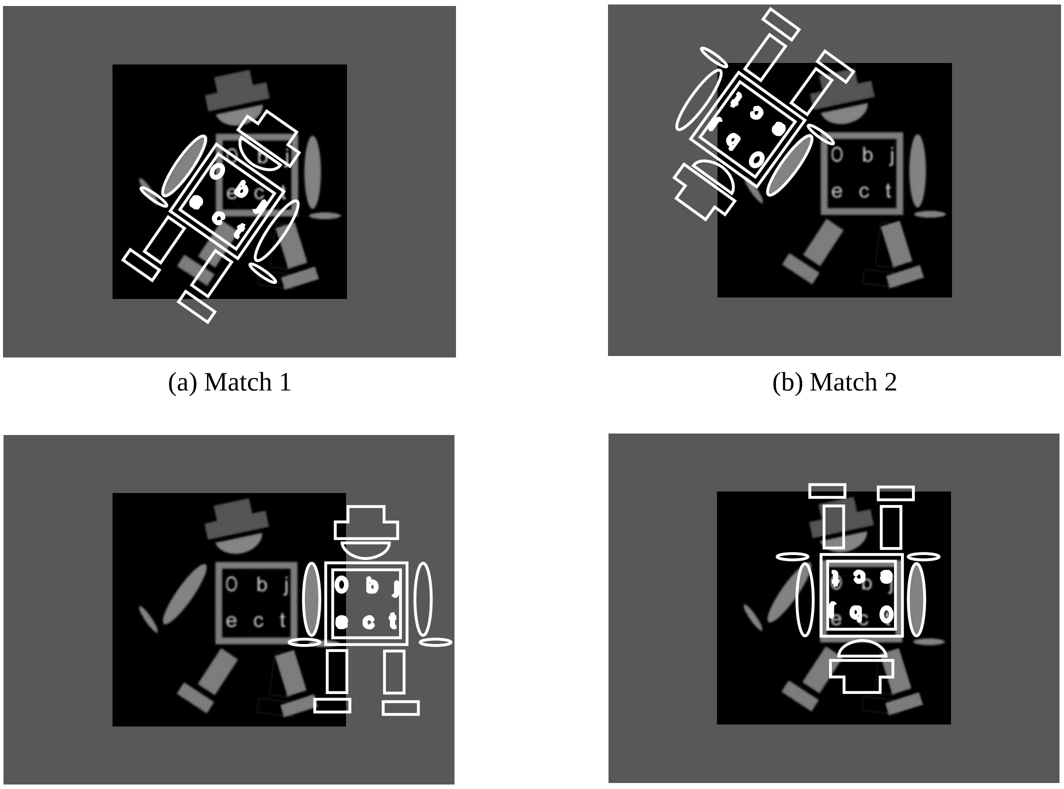
In the following, the single steps that are used to compute the cost value Ψ_{i_0,k_0} of the k_0-th match of component i_0 will be explained. At first, the parameters of the 2D rigid transformation that transform pose M_{i_0} into E_{i_0,k_0} are computed, resulting in a rotation matrix $\boldsymbol{R}$ with rotation angle α and a translation $\boldsymbol{t}$:

$$\begin{aligned} \alpha &= \varphi^e_{i_0,k_0} - \varphi^m_{i_0} && (5.1) \\ \boldsymbol{t} &= \boldsymbol{o}^e_{i_0,k_0} - \boldsymbol{R}(\alpha) \cdot \boldsymbol{o}^m_{i_0} \ . && (5.2) \end{aligned}$$

In the second step, the reference component configuration is projected into the example image by transforming the poses of all components i, providing the projected poses $M'_i = (\boldsymbol{o}^{m\prime}_i, \varphi^{m\prime}_i)$:

$$\begin{aligned} \boldsymbol{o}^{m\prime}_i &= \boldsymbol{R}(\alpha) \cdot \boldsymbol{o}^m_i + \boldsymbol{t} && (5.3) \\ \varphi^{m\prime}_i &= \varphi^m_i + \alpha \ . && (5.4) \end{aligned}$$

Figure 5.13 shows the projection of the reference component configuration into the example image for each of the four matches $k_0 = 1, \ldots, 4$ of component $i_0 = 3$.



(a) Match 1

(b) Match 2

(c) Match 3

(d) Match 4

Figure 5.13: Projection of the reference component configuration into the example image for each of the four matches of component 3 (left arm)

The projected pose of component i can then be compared to a certain match k ($k = 1, \ldots, n_i$) of the component i in the example image by computing a distance measure ψ:

$$\psi(M'_i, E_{i,k}) = \sqrt{\|\boldsymbol{o}_i^{m\prime} - \boldsymbol{o}_{i,k}^e\|^2 + w^2(\varphi_i^{m\prime} - \varphi_{i,k}^e)^2} \quad , \tag{5.5}$$

where $\|\cdot\|$ denotes the euclidian vector length and w is a weighting factor to balance the difference in position and orientation. A suitable value for w can be obtained, for example, based on the quantization of the object orientation $\Delta\varphi$. Assuming a position quantization of 1 pixel, w could be set to $1/\Delta\varphi$, for example. Because the correct match of component i is not known à priori, the distance measure $\psi(M'_i, E_{i,k})$ is computed for all matches k. The match with minimum distance is assumed to be correct and is used to compute the cost value Ψ_{i_0,k_0}. This assumption is the only difference to the exact computation using the previously explained error-correcting subgraph isomorphism, and is uncritical unless the true component configuration in the current example image does not differ extremely from the reference component configuration. Finally, the associated distance value is used in the computation of the cost value Ψ_{i_0,k_0} of match k_0 of component i_0:

$$\Psi_{i_0,k_0} = \sum_{i=1}^{n^c} \min_{k=1,\ldots,n_i} \psi(M'_i, E_{i,k}) \quad . \tag{5.6}$$

The computation of the cost value is repeated for each match k_0 and each component i_0 by applying (5.1)–(5.6). In some applications it might be desirable to apply a threshold on the cost value and eliminate the corresponding matches in order to reduce the sensitivity to outliers. The cost values are then used in the following algorithm to find the most likely match of each component using a global optimization.

5.3.3.2 Identification of Physical Instances

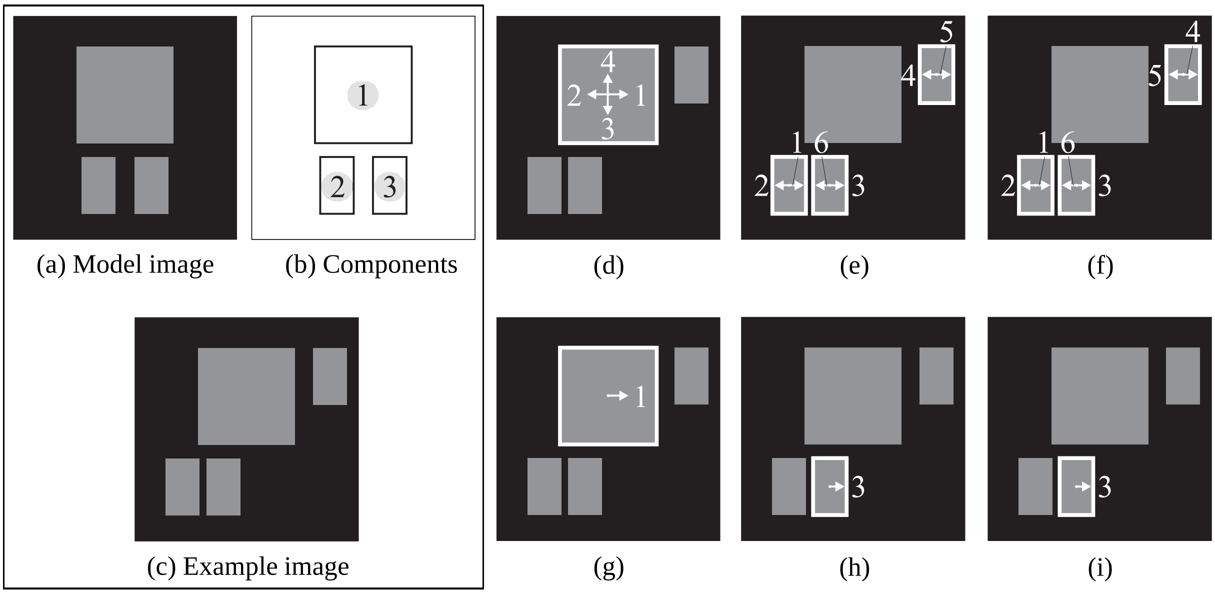
A simple way to get the most likely match of component i would be to select the match k, where $\Psi_{i,k}$ is minimal for $k = 1, \ldots, n_i$. Although this *local optimization* would assign one unique match to each component, one important condition would be neglected. It would be still possible that several matches of different components are assigned to the same *physical instance* in the example image. A physical instance represents an arbitrary structure in the example image to which components are matched. The problem is illustrated in Figure 5.14, where the compound object from the previous examples is reduced to an upper body and two legs in order to keep the explanations as clear and simple as possible. Furthermore, a clutter object has been added to the example image. Thus, four physical instances are present in the example image: the upper body, the two legs, and the clutter object.

If the match with the lowest cost value would be selected for each component, components 2 and 3 match the same physical instance (the actual instance of the right leg) in the example image. However, a desirable result would assign component 2 to match 1 and component 3 to match 3. Therefore, the task of a *global optimization* is to find for each component i the match k_i such that

$$\sum_{i=1}^{n^c} \Psi_{i,k_i} \longrightarrow \min \tag{5.7}$$

subject to the constraint that multiple matches are avoided, i.e., that at most one component is matched to a certain physical instance.

A prerequisite to solve the proposed minimization is to check whether two or more matches are assigned to the same physical instance. For this purpose, the similarities and symmetries of the components are analyzed in a preliminary stage. This is achieved by a pairwise matching of the single components to each other. All ordered pairs of components are selected. From the edges of the first component an artificial gray value image is generated by applying the algorithm of gray value expansion explained in Section 5.3.2. The second component is searched within the artificially generated image. If there are any matches then the relative pose of the second component with respect to the first component is computed and stored as the result of the analysis. After the relative poses of all pairs have been computed, the matches in the example image can be examined. If the relative pose of two components in the example image is identical to the relative pose that has been obtained as the result of the previous analysis (within a certain tolerance), it is known that the matches are assigned to the same physical instance. For example, if the right leg is searched within the artificial image created from the left leg two matches are returned. Because in this example the two components are identical, the two resulting relative poses are $((0,0)^\top, 0°)$ and $((0,0)^\top, 180°)$. With this information the respective six matches of each of the both components in the example image can be assigned to three physical instances.



(a) Model image (b) Components (d) (e) (f)

(c) Example image (g) (h) (i)

Figure 5.14: Example to illustrate the problem of multiple matches. From the model image (a) components are extracted (b) and searched in the example image (c). The matches of the components 1–3 are shown in (d)–(f), respectively, by white numerals. The match with respective minimum cost is selected for each component and displayed in (g)–(i). Components 2 and 3 match the same physical instance in the image.

5.3.3.3 Building the Bipartite Graph

Based on the previous considerations, a bipartite graph with the set of nodes V can be generated, which represents the problem in a structured form. A bipartite graph is distinguished by the property that V satisfies $V = V_1 \cup V_2$, where $V_1 \cap V_2 = \emptyset$, and each arc connects a node in V_1 to a node in V_2. Hence, no pair of nodes that are within the same set V_1 or V_2 are directly connected. The graph representing the current example is shown in Figure 5.15. Here, three set of nodes are displayed. The first two sets V^c and V^{mat} represent the components and the matches of the components in the example image, respectively. Each arc is weighted by an *affinity value* $\overline{\Psi}_{i,k} = -\Psi_{i,k}$, i.e., the higher the affinity value the more likely the match. Thus, minimizing the overall cost is equivalent to maximizing the overall affinities. Furthermore, the matches are grouped according to their associated physical instance, leading to the third set of nodes V^{phys}, where $n^{phys} \le n^{mat}$. Here, $n^{phys} = |V^{phys}|$ is the number of physical instances and $n^{mat} = |V^{mat}| = \sum_{i=1}^{n^c} n_i$ is the total number of matches in the example image. Finally, the bipartite graph is formed by the two sets of nodes V^c and V^{phys}, and is displayed in Figure 5.16(a). Consequently, each node in V^{phys} may be the head of several arcs.

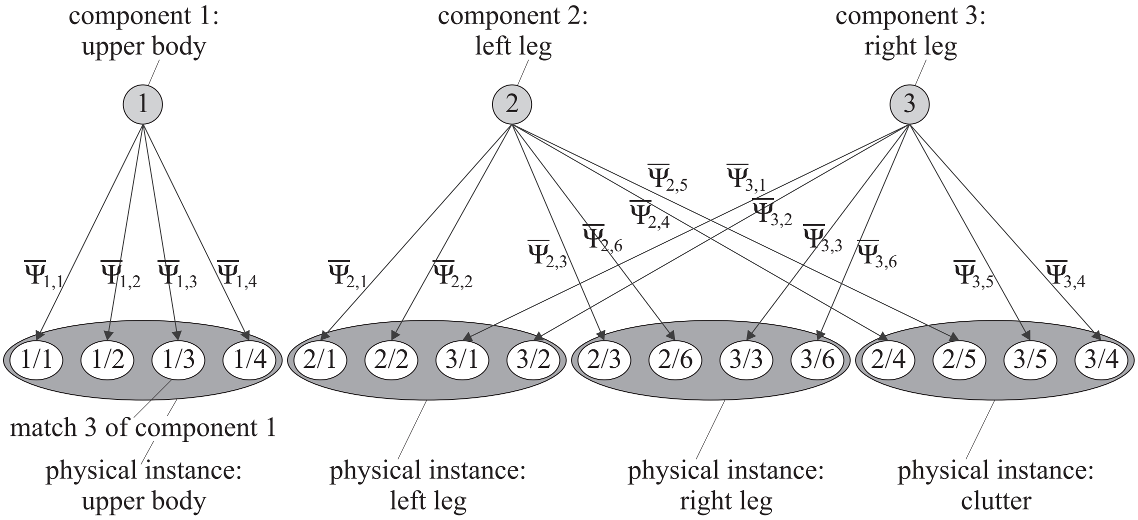


Figure 5.15: A bipartite graph with two sets of nodes V^c and V^{phys} represents the ambiguous matches in the example image (see Figure 5.14(d)–(f)). V^c is the set of nodes representing the components (three circular nodes in the upper row). The 16 small oval nodes in the lower row represent the single matches of the components. The matches are grouped according to their associated physical instance, leading to the set of nodes V^{phys} representing the physical instances (four big oval nodes in the lower row).

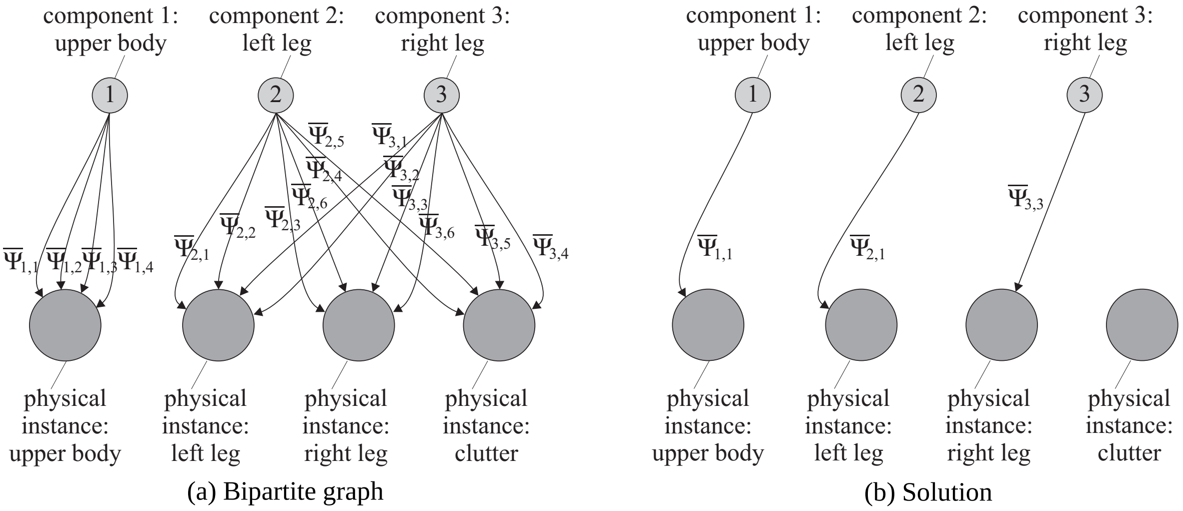


Figure 5.16: In (a) the bipartite graph of Figure 5.15 is displayed in a condensed form. The result of the bipartite graph matching contains the most likely component configuration (b) in the example image.

5.3.3.4 Bipartite Graph Matching using Linear Programming

Now, the problem of solving the ambiguities can be formulated as a bipartite graph matching problem from V^c to V^{phys}. Informally speaking, in graph theory a matching is a set of arcs, where a node is the head of at most one arc. This constraint exactly takes the original desire into account that a physical instance is assigned to at most one component. It should be noted that some physical instance, which may, for example, be caused by image clutter, may not have a corresponding component. When solving the ambiguities, a second constraint must be considered: a node is the tail of at most one arc. This ensures that to each component at most one match, and hence at most one physical component, is assigned. Because of possibly missing components in the example image, some components may not have a physical instance. Finally, since it is desirable that as many components as possible are found in the example images, the task is to find a set of arcs with maximum size and with maximum overall affinity that simultaneously fulfills the two stated constraints. The result of this special class of bipartite graph matching is displayed in Figure 5.16(b). Here, the solution includes three arcs that represent match 1 of component 1, match 1 of component 2, and match 3 of component 3. To convince oneself of the intuitive correctness of the solution, one can take a look at the respective matches displayed in Figure 5.12(d)–(f). Now, also component 2 is found at the correct pose.

The graph matching problem can be solved applying methods of linear programming. Linear programming is concerned with maximizing a linear objective function of continuous real variables, subject to linear constraints. Formally speaking, the task is to find a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_{n^x})^\top$ of n^x variables that maximizes the function

$$\boldsymbol{a}^\top \boldsymbol{x} \longrightarrow \max \tag{5.8}$$

subject to the primary constraints

$$x_i \geq 0 \quad , \forall i = 1, \ldots, n^x \tag{5.9}$$

and simultaneously subject to n^b additional constraints of the form

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \ , \tag{5.10}$$

with $\boldsymbol{a} = (a_1, a_2, \ldots, a_{n^x})^\top$, $\boldsymbol{b} = (b_1, b_2, \ldots, b_{n^b})^\top$, and the $n^b \times n^x$ matrix $\boldsymbol{A}$ (Bronstein et al. 2001).

In order to transfer the bipartite graph matching problem into the form of linear programming, a variable $x_{i,k}$ is assigned to each arc in the graph, where $i = 1, \ldots, n^c$ and $k = 1, \ldots, n_i$. The $x_{i,k}$ are the unknowns to be estimated within the linear programming. If the arc that represents match k of component i is part of the solution, $x_{i,k}$ will be 1, otherwise $x_{i,k}$ will be 0. Thus, the objective function (5.8) is

$$\sum_{i=1}^{n^c} \sum_{k=1}^{n_i} \overline{\Psi}_{i,k} \cdot x_{i,k} \longrightarrow \max \ , \tag{5.11}$$

with the aim of a maximum number of matches with maximum overall affinity. Several additional constraints must be taken into account. Because the primary constraint only considers that $x_{i,k} \geq 0$, additionally n^{mat} constraints

$$x_{i,k} \leq 1 \quad , \forall i = 1, \ldots, n^c \ , \forall k = 1, \ldots, n_i \tag{5.12}$$

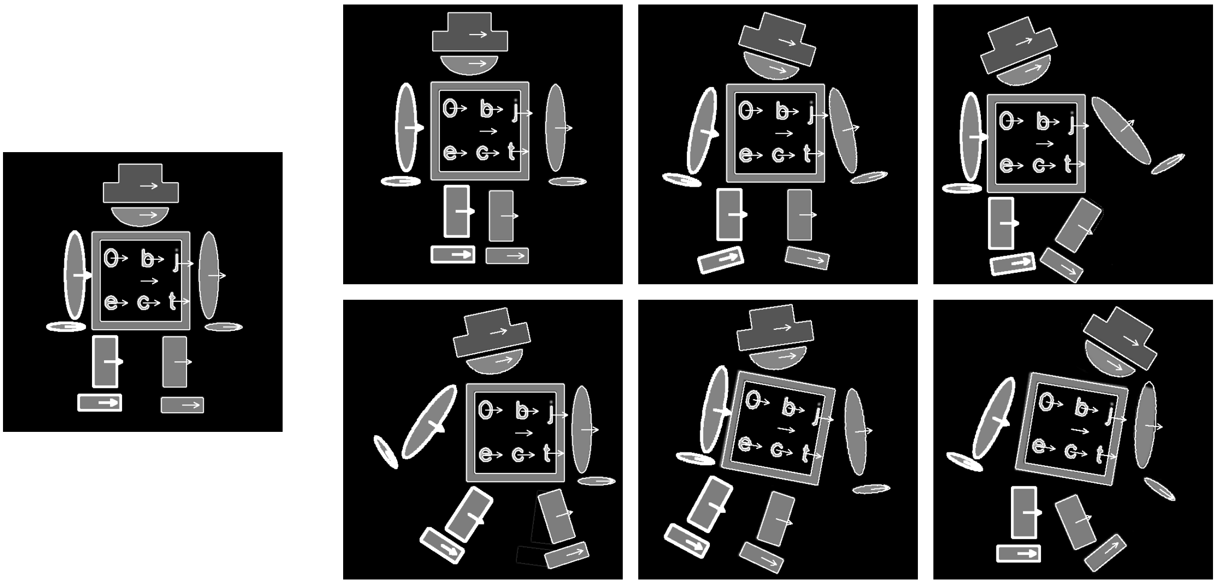
must be formalized explicitly (cf. (5.10)). This ensures that $x_{i,k} \in [0,1]$, but a meaningful solution requires $x_{i,k}$ to take binary values (i.e., $x_{i,k} \in \{0,1\}$) only. Fortunately, this is ensured by a theorem from *integer programming* (Garfinkel and Nemhauser 1972). This also becomes immediately evident when recalling the linearity of the objective function: the inequality constraints can be geometrically interpreted as a convex polyhedron in the n^{mat}-dimensional parameter space. Consequently, the position of the maximum is restricted to lie at a vertex of the polyhedron, in which $x_{i,k}$ is always either 0 or 1 (ignoring the special case in which the level lines of the objective function are parallel to an edge of the polyhedron).

Next, the constraint that to each component at most one match is assigned is introduced. Thus, the sum of all $x_{i,k}$ that are associated with the arcs leaving the same component node must be smaller or equal to 1 (cf. (5.10)):

$$\sum_{k=1}^{n_i} x_{i,k} \leq 1 \quad , \forall i = 1, \ldots, n^c \ . \tag{5.13}$$

This results in n^c additional inequality constraints. However, in the case of $n_i \leq 1$ the constraint for component i can be omitted since it is already represented in (5.12).

The final constraints ensure that each physical instance is assigned to at most one component. This can be formalized by restricting the sum of all $x_{i,k}$ that are associated with the arcs ending in the same physical instance to be smaller or equal to 1. Let n_j^{mat} be the number of matches that are assigned to the physical



(a) Model image

(b) Example images

Figure 5.17: Ambiguities solved by linear programming. The model image and the reference component configuration is shown in (a). Similar components are distinguished by different line widths for visualization purposes. For each example image the result of the linear programming assigns an unambiguous pose to each component (b).

instance j and let $x_{i,k}(j,l)$ be the variable $x_{i,k}$ that represents the l-th match that is assigned to the physical instance j. Then the constraint can be formalized as (cf. (5.10)):

$$\sum_{l=1}^{n_j^{mat}} x_{i,k}(j,l) \leq 1 \quad , \forall j = 1, \ldots, n^{phys} \quad . \tag{5.14}$$

Thus, the resulting linear programming problem is described by the objective function (5.11), which must be maximized subject to the constraints described by (5.12)–(5.14). Several efficient standard algorithms are available in the literature for linear programming. One of the most popular representatives is the *simplex method* (Press et al. 1992, Bronstein et al. 2001). Although it has been proven that its theoretically worst case runtime complexity is exponential, it merely shows polynomial time complexity on average for practical problems. Nevertheless, several “true” polynomial-time algorithms have been developed, e.g., (Karmarkar 1984). Since the description of one of these algorithms would go beyond the scope of this dissertation the reader is referred to the literature. Finally, the result of the linear programming provides a value for each $x_{i,k}$ that is either 0 or 1. In the present example, all $x_{i,k}$ are returned as 0 except $x_{1,1}$, $x_{2,1}$, and $x_{3,3}$, which are returned as 1, as one would expect (see Figure 5.16(b)).

It should be noted that the algorithm is able to handle missing components by choosing the constraints in the proposed way. However, it requires that at most one instance of the compound object is present in each example image since otherwise the algorithm would pick out the best component matches from different instances.

Returning to the original example, the ambiguities are solved for each example image individually according to the above described method. Hence, a unique pose for each component in each example image is obtained. The final result for all example images is shown in Figure 5.17. The unique poses are stored within the $n^e \times n^c$ component pose matrix.

By solving the ambiguities during the training of the hierarchical model the correspondence problem that would arise during the online phase when searching the object parts independently from each other is already implicitly solved within the hierarchical model. Thus, one can say that the correspondence problem is shifted from the online to the offline phase with the considerable advantage that a real-time recognition of compound objects is made possible.

5.3.4 Extraction of Object Parts

The rigid parts of the compound object may be represented by several single components because of the over-segmentation during the initial decomposition. Therefore, the initial components that belong to the same rigid object part, and hence exhibit identical apparent movement over all example images, can be merged by analyzing the unique poses obtained in the previous section. The merged components represent the rigid object parts. The strategy comprises two steps. In the first step, for each pair of components i_1 and i_2 the probability that the two components belong to the same object part is computed in a statistically founded manner, resulting in a square probability matrix of size $n^c \times n^c$. In the second step, the probability matrix is clustered using a pairwise clustering algorithm. The obtained clusters represent the desired object parts.

Let $M_{i_1} = (\boldsymbol{o}^m_{i_1}, \varphi^m_{i_1})$ with position $\boldsymbol{o}^m_{i_1} = (x^m_{i_1}, y^m_{i_1})^\top$ be the pose of component i_1 in the model image and $E_{i_1} = (\boldsymbol{o}^e_{i_1}, \varphi^e_{i_1})$ with position $\boldsymbol{o}^e_{i_1} = (x^e_{i_1}, y^e_{i_1})^\top$ the corresponding unique pose in the example image. Accordingly, the poses of a second component i_2 are M_{i_2} and E_{i_2}, respectively. Furthermore, assume that accuracy information is available for each pose. The accuracies are represented by the 3 × 3 covariance matrices $\boldsymbol{K}^e_{i_1}$ and $\boldsymbol{K}^e_{i_2}$, which contain the variances and covariances of x, y, and φ. If the used recognition method does not return any accuracy information for the pose parameters, the accuracy must be specified empirically (e.g., by applying tests with various objects of different size and shape). The reference position and orientation of the components in the model image are assumed to be error-free. Starting with this information, the probability that the two components belong to the same object part can be computed.

At first, the parameters α and $\boldsymbol{t}$ of the rigid transformation that transform M_{i_1} into E_{i_1} are computed (cf. (5.1) and (5.2) in Section 5.3.3.1). By applying the transformation to the pose M_{i_2} the projected pose E'_{i_2} of component i_2 in the example image is obtained (cf. (5.3) and (5.4) in Section 5.3.3.1). The assumption that both components belong to the same object part would require that component i_1 and i_2 have moved identically with respect to the model image, and hence E'_{i_2} = E_{i_2}. In general, this requirement is not fulfilled, even for components of the same object part because of the limited accuracy of the object recognition method. One method to get a kind of probability value for the current pair of components is to compute a distance measure between E'_{i_2} and E_{i_2}. The drawback of this method is that it is hard to decide up to which distance the components can be treated as belonging to the same part. A better result can be obtained by computing a real probability value $p_{i_1,i_2} \in [0, 1]$. This is achieved by applying methods of hypothesis testing. Stating the hypothesis E'_{i_2} = E_{i_2} requires

$$\begin{aligned} x^{e\,\prime}_{i_2} - x^e_{i_2} &= 0 && (5.15)\\ y^{e\,\prime}_{i_2} - y^e_{i_2} &= 0 && (5.16)\\ \varphi^{e\,\prime}_{i_2} - \varphi^e_{i_2} &= 0\ . && (5.17)\end{aligned}$$

The hypothesis can be rewritten in matrix form $\boldsymbol{Hx} = \boldsymbol{w}$, where

$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}, \boldsymbol{x} = (x^{e\,\prime}_{i_2}, y^{e\,\prime}_{i_2}, \varphi^{e\,\prime}_{i_2}, x^e_{i_2}, y^e_{i_2}, \varphi^e_{i_2})^\top, \boldsymbol{w} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \qquad (5.18)$$

For further processing, the covariance matrix $\boldsymbol{K}^{e\,\prime}_{i_2}$ of the projected pose E'_{i_2} is needed. It can be obtained by applying the law of error propagation to the covariance matrix $\boldsymbol{K}^e_{i_1}$ with respect to equations (5.1)–(5.4): $\boldsymbol{K}^{e\,\prime}_{i_2} = \boldsymbol{A}\boldsymbol{K}^e_{i_1}\boldsymbol{A}^\top$. Here, $\boldsymbol{A}$ is the 3 × 3 Jacobian matrix, which contains the partial derivatives of the pose parameters in E'_{i_2} with respect to the pose parameters in E_{i_1}. After some simplifications, $\boldsymbol{A}$ reads as follows:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & \Delta x_{1,2} \sin \Delta\varphi^{m,e} + \Delta y_{1,2} \cos \Delta\varphi^{m,e} \\ 0 & 1 & -\Delta x_{1,2} \cos \Delta\varphi^{m,e} + \Delta y_{1,2} \sin \Delta\varphi^{m,e} \\ 0 & 0 & 1 \end{pmatrix} , \qquad (5.19)$$

with $\Delta x_{1,2} = x^m_{i_2} - x^m_{i_1}$, $\Delta y_{1,2} = y^m_{i_2} - y^m_{i_1}$, and $\Delta\varphi^{m,e} = \varphi^e_{i_1} - \varphi^m_{i_1}$. Now, the associated covariance matrix of $\boldsymbol{x}$ can be composed:

$$\boldsymbol{K}_{xx} = \begin{pmatrix} \boldsymbol{K}^{e\,\prime}_{i_2} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{K}^e_{i_2} \end{pmatrix} , \qquad (5.20)$$

where $\mathbf{0}$ represents a 3 × 3 zero matrix.

With the results obtained by the previous calculations all information to perform the actual hypothesis testing is available. At first, a test value T is computed (Koch 1987):

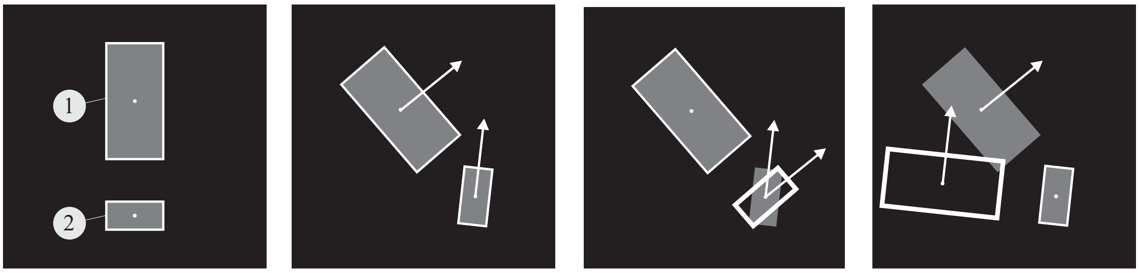
$$T = \frac{1}{r} (\boldsymbol{H}\boldsymbol{x})^\top (\boldsymbol{H}\boldsymbol{K}_{xx}\boldsymbol{H}^\top)^{-1} \boldsymbol{H}\boldsymbol{x} , \qquad (5.21)$$

where $r = 3$ denotes the number of equations in the hypothesis (5.15)–(5.17). The test value $T \sim F_{m,n}$ has a (Fisher) F-distribution with the parameters m and n denoting the degrees of freedom. The parameter m corresponds to the number of equations r and the parameter n to the redundancy involved in computing the accuracies of the pose parameters. If a value for n is not available, e.g., because the accuracy information has been obtained by empirically tests instead of a preceding parameter adjustment, it is assumed that the accuracies have been determined by using an infinite large set of samples. Consequently, $n \to \infty$ and the F distribution degenerates to the χ^2-distribution with $T \cdot r \sim \chi^2_r$. Hence, the probability p_{i_1,i_2} that the two components belong to the same object part can be written as

$$p_{i_1,i_2} = 1 - \int_{-\infty}^{T} F_{r,n}(t)\, dt \overset{n\to\infty}{=} 1 - \int_{-\infty}^{T\cdot r} \chi^2_r(t)\, dt , \qquad (5.22)$$

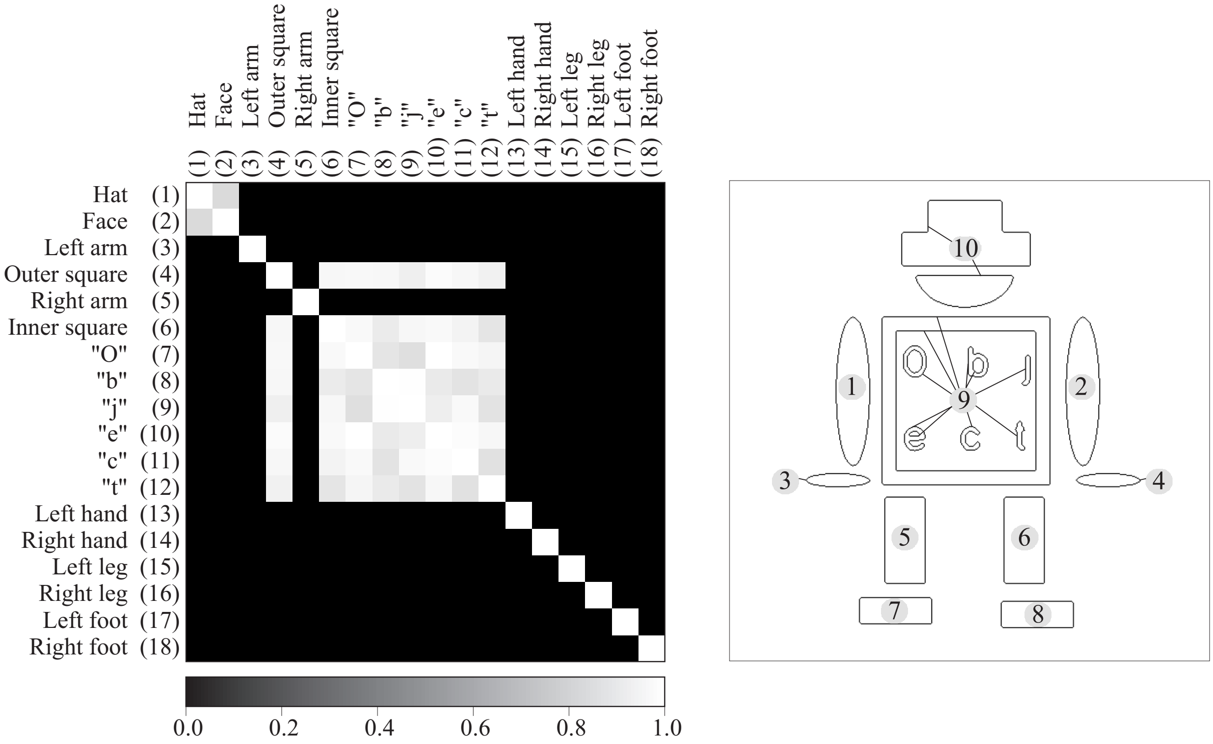
with $F_{r,n}(t)$ and $\chi^2_r(t)$ representing the probability density function of the respective distributions. For practical considerations, the evaluation of (5.22) can be reduced to the calculation of the associated incomplete gamma function (Press et al. 1992).

The probability matrix that is obtained by repeating the computations for each directed pair of components is not symmetric, i.e., $p_{i_1,i_2} \neq p_{i_2,i_1}$. This at first glance non-intuitive observation becomes evident when examining the transformation described by (5.1)–(5.4) more closely. The small example in Figure 5.18 facilitates the discussion. Assume that the pose of the two components shown in Figure 5.18(a) are determined in the example image shown in Figure 5.18(b). In the first step, the poses of component 1 in the model image and the example image, respectively, are used to compute the rigid transformation parameters (i.e., $i_1 = 1$). In the second step, component 2 is projected into the example image using the calculated transformation (i.e., $i_2 = 2$). The projected pose of component 2 only differs in orientation from its true pose (see Figure 5.18(c)). A different observation can be made if component 2 is used to compute the transformation parameters and component 1 is projected accordingly (i.e., $i_1 = 2$, $i_2 = 1$). The projected pose of component 1 not only differs in orientation but additionally differs in position from its true pose. Hence, in the second case the associated *directed* probability value is significantly lower. In order to receive a symmetric *undirected* probability measure, the minimum of both corresponding directed probabilities is taken since a rigid object simultaneously requires that both directed probability values are small. Finally, because a high probability value is required for components of the same object part in all images, either the minimum value or a more robust quantile value over all example images is computed. Consequently, another demand on the example images can be derived. Assume that the minimum probability value is decisive. If in all example images two object parts accidently move in the same manner, then the algorithm will mistakenly assume that the two parts can



(a) Model image (b) Example image (c) $i_1 = 1$ (d) $i_1 = 2$

Figure 5.18: Non-symmetry of relative movement. The poses of the two components that are extracted from the model image (a) are uniquely determined in the example image (b). In (c) component 2 is projected according to the pose of component 1. The true and projected pose only differ in orientation. However, when projecting component 1 according to the pose of component 2, additionally a translation difference occurs (d).



(a) Probability matrix (b) Object parts

Figure 5.19: The symmetric probability matrix contains information about the probability that a pair of components belongs to the same rigid object part (a). After clustering the matrix, ten rigid object parts are obtained (b). It should be noted that the numbers in (a) representing the components and the numbers in (b) representing the obtained object parts must not be confused.

be combined in one rigid part. Therefore, different object parts must show a relative movement in at least one example image in order to be detected as two separate object parts.

In Figure 5.19(a) the finally obtained symmetric probability matrix is shown after computing the minimum over all example images. One can see that the pairwise probability for belonging to the same object part is high for the hat and the face as well as for the components forming the upper body. In contrast, the remaining probabilities are approximately zero.

In the second step the components are partitioned into groups, or *clusters*, such that the probability between components in the same cluster is high and the probability between components in different clusters is small.

The clusters finally represent the rigid object parts. In the following, the computed probability matrix is also referred to as a similarity matrix, where the similarity expresses the rigidity between components. Many different clustering algorithms for similarity matrices, or dissimilarity matrices, respectively, have become available. A comprehensive overview is given in (Jain et al. 1999). To find the rigid object parts, a hierarchical agglomerative clustering algorithm is applied to the similarity matrix (actually, it is sufficient to take the upper triangular matrix into account):

1. Initialize n^c clusters, each containing one component.
2. Find the maximum entry in similarity matrix. If the maximum similarity is less than a threshold p^{min}, return clusters and stop calculation.
3. Merge associated pair of clusters into one cluster.
4. Update similarity matrix to reflect the merge operation.
5. If at least two clusters are left, go to step 2, else return clusters.

The update of the similarity matrix stated in step 3 means recalculating the similarity values between the new cluster and the remaining clusters using a certain *linkage metric*. The linkage metric characterizes the similarity between a pair of clusters. The most popular methods either use the *single-link*, *average-link*, or *complete-link* algorithm (Berkhin 2002). In the single-link algorithm, the similarity between two clusters is the maximum of the similarities between all pairs of components drawn from the two clusters. Accordingly, the complete-link algorithm takes the minimum similarity and the average-link algorithm the average similarity. On the one hand, the complete-link algorithm is too stringent, and hence often fails to link components belonging to the same object part. On the other hand, the single-link algorithm suffers from a chaining effect: in spite of a very low probability value the two associated components may be merged into the same cluster if the low probability is bridged by other components. In contrast, the average-link algorithm has proven to be a suitable compromise, and hence is applied to cluster the components. Finally, the number of returned clusters specifies the number of rigid object parts n^p within the compound object. The result is shown in Figure 5.19(b), where the probability matrix of Figure 5.19(a) has been clustered using a threshold p^{min} for the probability of 0.5 (cf. step 2 of the clustering algorithm). The 18 components have been clustered into 10 object parts. Now the hat and the face form one object part. Furthermore, the components of the upper body belong to the same object part.

Finally, for the object parts that consist of exactly one component the pose of the component is adopted by the object part in each example image. For those object parts that consist of more than one component the poses must be explicitly determined. Simply taking the average pose over all involved components within a cluster would introduce errors. To avoid these errors, a new rigid model is created for the corresponding object parts from the model image and used to search the object parts in all example images. This can be realized in a similar manner as for the components. However, the search can be focused on a very restricted parameter space since an approximate pose is known. Therefore, the computational effort is negligible and no ambiguities must be solved. After this step, for each rigid object part the pose parameters in each image are available and stored within the $n^e \times n^p$ part pose matrix for further analysis.

Some concluding remarks concerning the image rectification should be mentioned. In order to ensure a correct computation of the probability values, it is necessary that the model image and all example images are free of distortions. Therefore, it is essential that all images are rectified with the approach presented in Chapter 3 in order to eliminate radial and projective distortions. Otherwise the distortions would lead to pseudo-movements between components that belong to the same rigid object part, and hence would result in small probability values. Consequently, also the search images in the subsequent online phase must be rectified because the extraction of the relations, which will be described in the following section, is also based on the (rectified) example images. Nevertheless, in some very time-critical practical cases it may be desirable to refrain from image rectification. Therefore, if no significant projective distortions are present one can compensate existing radial distortions using one of two possibilities. The first possibility is to appropriately reduce the threshold p^{min} for the probability value. The second possibility is to appropriately increase the standard deviations of the pose parameters, which are used in the hypothesis test. Consequently, small relative movements between object parts cannot be distinguished from effects caused by the radial distortions any longer. However, if no small movements must be expected and detected this is a suitable possibility in practice to avoid the rectification and to further speed up the online phase.

5.3.5 Analysis of Relations between Object Parts

Now that the poses of the object parts are available in each example image, information about the relations or the relative movements between the single object parts can be extracted. To compute the relation between object parts i_1 and i_2, the pose of object part i_2 is computed in the local coordinate system of object part i_1. Let $(\boldsymbol{o}_{i_1}, \varphi_{i_1})$ and $(\boldsymbol{o}_{i_2}, \varphi_{i_2})$ be the poses of the two object parts in an example image or in the model image. The transformation into the local coordinate system can be described by a rotation matrix $\boldsymbol{R}$ with rotation angle α and a translation $\boldsymbol{t}$:

$$\alpha = -\varphi_{i_1} \tag{5.23}$$

$$\boldsymbol{t} = -\boldsymbol{R}(\alpha) \cdot \boldsymbol{o}_{i_1} \ . \tag{5.24}$$

The pose $({\boldsymbol{o}_{i_2}}', {\varphi_{i_2}}')$ of object part i_2 in the local coordinate system is obtained as:

$${\boldsymbol{o}_{i_2}}' = \boldsymbol{R}(\alpha) \cdot \boldsymbol{o}_{i_2} + \boldsymbol{t} \tag{5.25}$$

$${\varphi_{i_2}}' = \varphi_{i_2} + \alpha \ . \tag{5.26}$$

If equations (5.23)–(5.26) are applied in the model image and all example images the relative movement of part i_2 with respect to i_1 becomes available.

The extraction of the relations between two object parts is exemplified in Figure 5.20. Here, the pose of the right leg defines a local coordinate system into which the pose of the left arm is transformed for all images. The reference point of the transformed left arm describes a movement within the local coordinate system. To quantify the position relation, the convex hull of the transformed reference points could be used. Hence, the polygon of the convex hull would describe the border of the area within which the reference point of the left arm is assumed to appear in the local coordinate system. Furthermore, the enclosing angle interval of the orientation of the transformed left arm is assumed to describe the orientation relation. On the one hand, these assumptions imply that the example images cover the extrema of all possible relative movements. If it is not guaranteed that this requirement is fulfilled it is advisable to add appropriate tolerance values manually to the automatically computed relations. Otherwise object parts could be missed during the search in the online phase. On the other hand, even one outlier could unnecessarily inflate the convex hull and the enclosing angle interval, respectively. This would result in a sub-optimal online phase because of an increased computation time. Although this is less critical in comparison to missing object parts, it sometimes is desirable to avoid outliers by applying a statistically robust elimination of single extreme poses.

In the later online phase the left arm only needs to be searched within the parameter range described by the relations if the right leg has been found before. The number of polygon points in the convex hull may take values that correspond to the number of used images. Thus, the use of the convex hull would be accompanied by a high computational load within the online phase. This is because all polygon points would have to be transformed appropriately in order to compute the ROI in which the reference point is to be searched. Therefore, the smallest enclosing rectangle of arbitrary orientation is used instead to describe the position relation. Although the convex hull would be able to describe the relations in a less wasteful way, the rectangle is preferable also because the negligible loss in efficiency is justified by relaxed demands on the example images.

Finally, the computations are repeated for each directed object pair and the extracted smallest rectangle and the orientation angle interval are stored within the $n^p \times n^p$ relation matrix. The smallest rectangle is represented by its centroid $\boldsymbol{c}^r_{i_1,i_2}$, its semi axes $a^r_{i_1,i_2}$ and $b^r_{i_1,i_2}$, and its orientation $\beta^r_{i_1,i_2}$. The orientation angle interval

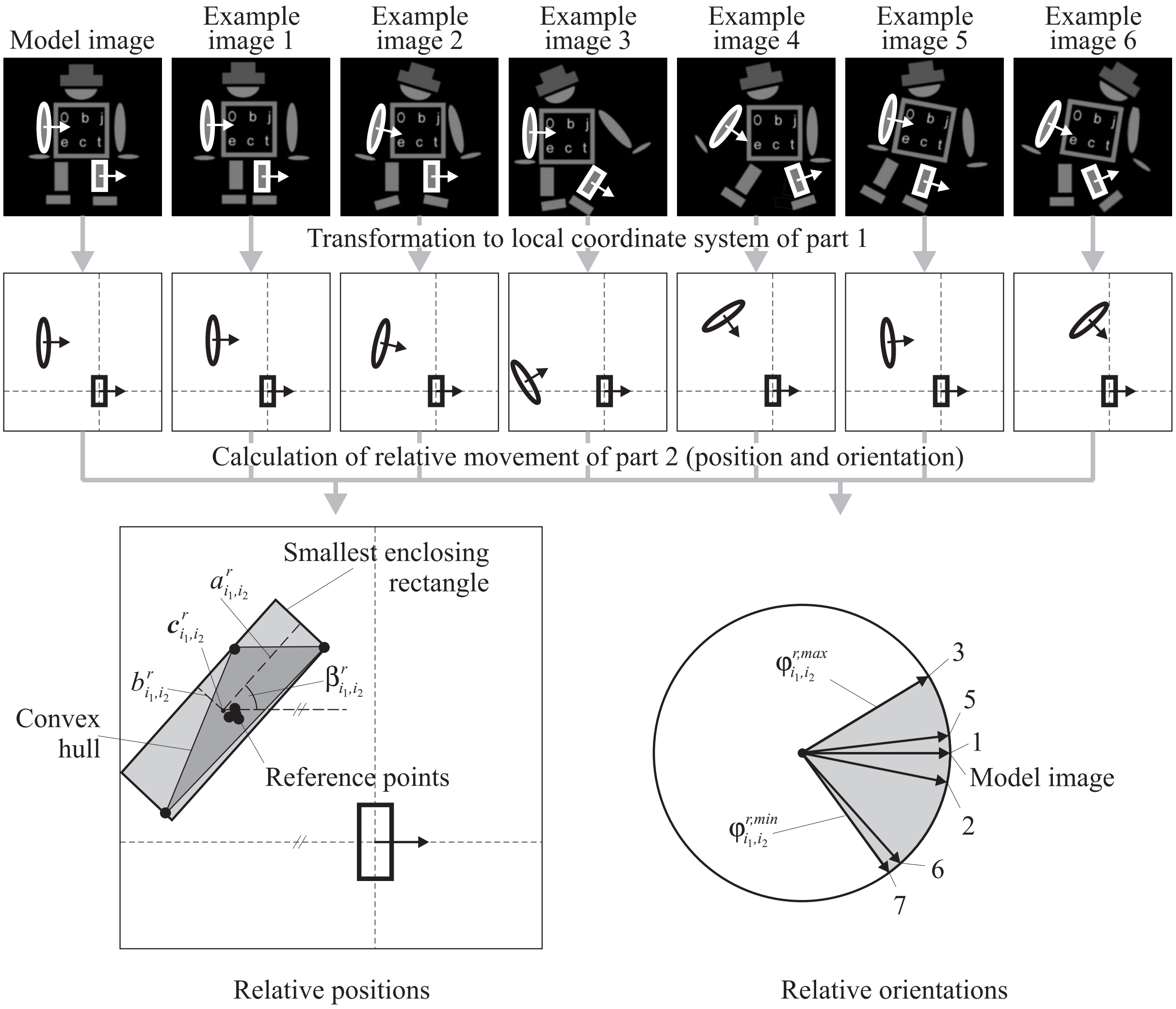


Figure 5.20: Example showing the extraction of the relative movement of the left arm (i_2) with respect to the right leg (i_1). The position relation is described by the smallest enclosing rectangle of the reference points, the orientation relation by the smallest enclosing angle interval.

is represented by its boundaries $\varphi^{r,min}_{i_1,i_2}$ and $\varphi^{r,max}_{i_1,i_2}$. Apart from the position and the orientation relation additional information about the mean and the standard deviation of the relative movements are stored. The result is shown in Figure 5.21.

After this step, the training of the hierarchical model is completed. The result comprises the relation matrix as well as the ROIs of the object parts. Each ROI refers to the associated edge region of one object part in the model image. The result represents the input data for creating the hierarchical model.

5.4 Creating the Hierarchical Model

5.4.1 Rigid Models for the Object Parts

The first step to create the hierarchical model is to generate a rigid model for each object part. Although already during the training models have been created, the orientation range for which the models have been built does not necessarily coincide with the desired orientation range during the online phase. For example, the user may introduce prior knowledge about the possible orientation of the compound object in the search

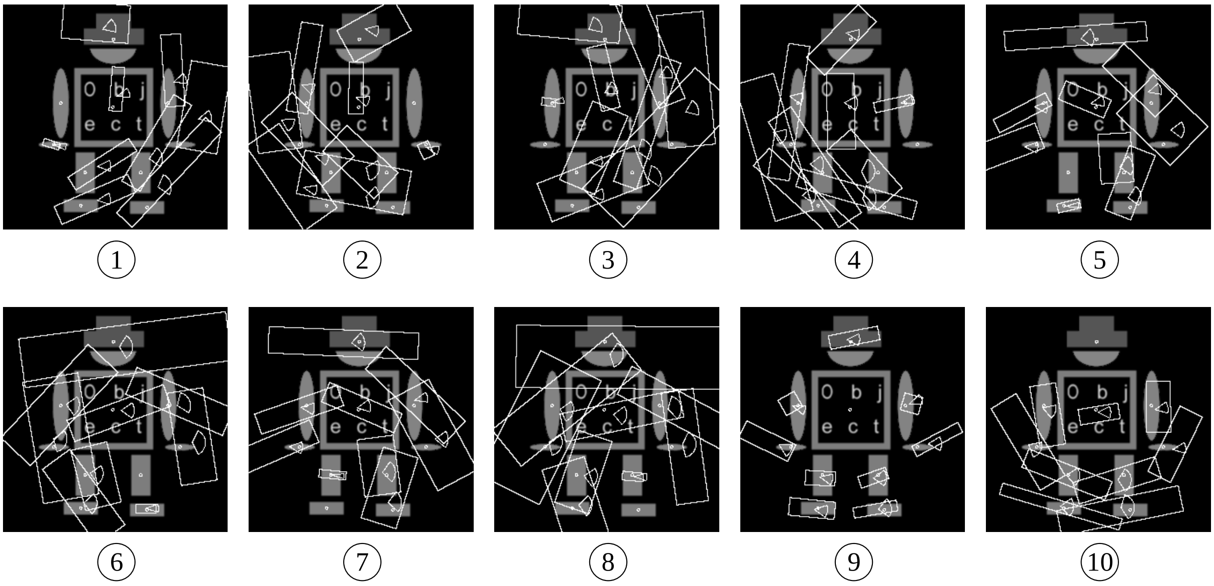


Figure 5.21: The relations of all pairs of object parts are visualized. Each picture represents an object part i_1 for which the relative movements of the remaining parts i_2 are displayed.

image. This is similar to the model creation for rigid object recognition (cf. Section 4.2.6) and may help to keep the memory requirement of the model small. However, when dealing with compound objects the question arises how the orientation of a compound object is defined. In this dissertation the orientation of the compound object is equated with the orientation of a user-specified reference object part. For the reference part the user may specify the orientation range for which the associated rigid model should be created. The orientation ranges of the remaining object parts are then automatically defined by the relations between the parts. Because the search order is unknown à priori, a worst case estimation is used to get the orientation ranges of the remaining object parts.

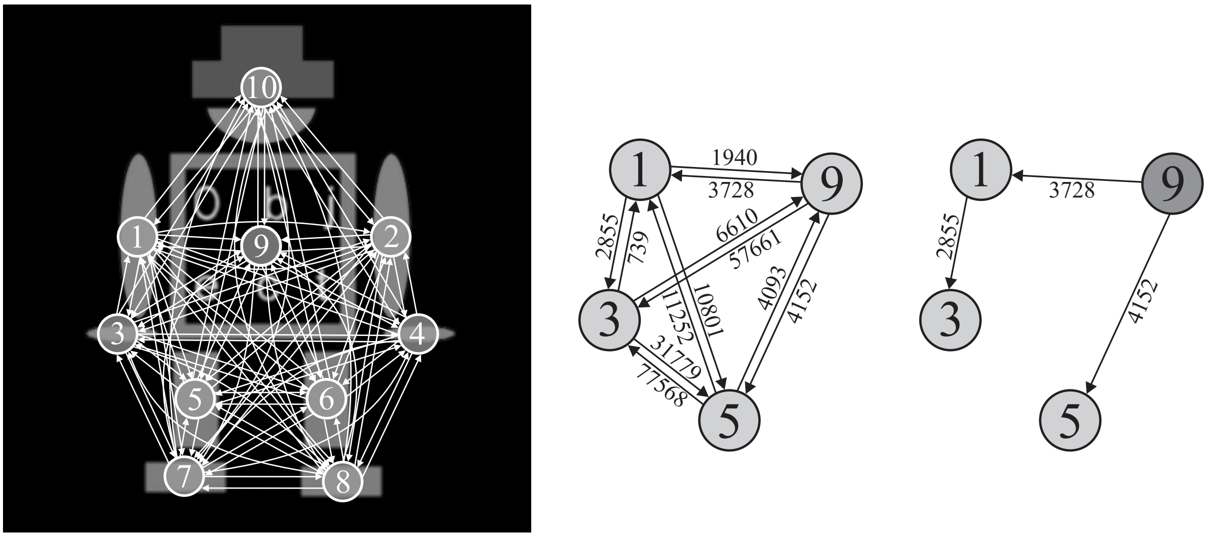
5.4.2 Optimum Search Trees

Based on the relations, an optimum hierarchical search strategy can be found by minimizing the computational search effort in the online phase. One major part of the optimum hierarchical search strategy is represented by the optimum search trees. In this section, the definition and computation of the search trees is explained.

The relation between part i_1 and i_2 is used to compute the relative search effort $\Omega^r_{i_1,i_2}$ that must be spent in order to search part i_2 relative to part i_1. The computation of the search effort depends on the kind of applied rigid object recognition method. If either the MGHT or the SBM is used the search effort is approximately given by:

$$\Omega^r_{i_1,i_2} = 2a^r_{i_1,i_2} \cdot 2b^r_{i_1,i_2} \cdot (\varphi^{r,max}_{i_1,i_2} - \varphi^{r,min}_{i_1,i_2}) \cdot \frac{n^{m,top}_{i_2}}{4^{n^l_{i_2}-1} \cdot \Delta\varphi^{top}_{i_2}} \quad . \tag{5.27}$$

The search effort $\Omega^r_{i_1,i_2}$ is proportional to the area of the rectangle multiplied by the size of the orientation angle interval, both given by the relations. Hence, this product represents the size of the continuous 3D parameter space to be scanned. Because of the quantization of the parameter space and the use of image pyramids, the search effort of part i_2 is reduced by a factor that depends on the number of pyramid levels $n^l_{i_2}$ and the quantization of the orientation on the top pyramid level given by the orientation step $\Delta\varphi^{top}_{i_2}$. Finally, the search effort increases linearly with the number of model edge points $n^{m,top}_{i_2}$ that remain at the top pyramid level. It should be noted that the search effort is not symmetric, i.e., $\Omega^r_{i_1,i_2} \neq \Omega^r_{i_2,i_1}$ (for details see Figures 5.18 and Figure 5.21 of Sections 5.3.4 and 5.3.5, respectively).



(a) Directed graph (b) Subgraph of (a) (c) Optimum search tree

Figure 5.22: The object parts and the relations between them are represented in a directed graph (a). For illustration purposes a small subgraph is selected showing four nodes and the corresponding weights (b). The minimum spanning arborescence of the graph represents the optimum search tree (c). Here, part 9 was selected as the root part.

Assume that part j serves as root part, i.e., part j is the only part that will be searched within the entire search space during the online phase. Then, the task for finding the optimum hierarchical search tree for the preselected root part j can be equated with minimizing the overall relative search effort $\bar{\Omega}^r_j$:

$$\bar{\Omega}^r_j = \sum_{i=1, i\neq j}^{n^p} \Omega^r_{p(i),i} \longrightarrow \min \ , \tag{5.28}$$

where $\Omega^r_{p(i),i}$ denotes the effort to search object part i relative to its predecessor part $p(i)$. By definition, the root part does not have a predecessor part, and hence is excluded from the calculation. Informally speaking, this optimization problem means to find a predecessor part for each object part such that $\bar{\Omega}^r_j$ is minimized. Furthermore, it must be ensured that in the online phase the pose of $p(i)$ has already been determined before searching for part i. Consequently, the search in the online phase can be represented by a tree, where the root node represents the root part, which is searched within the entire search space, and the other nodes represent the parts that are searched relative to their associated predecessor part.

To solve this optimization problem, one may think of the object parts and the relations between them as a complete directed graph $G(V, E)$, where V denotes the set of nodes with $|V| = n^p$ and E is the set of arcs with $|E| = n^p(n^p - 1)$. A complete directed graph is a directed graph where each two nodes i_1 and i_2 are connected by the two arcs (i_1, i_2) and (i_2, i_1). The nodes in the graph represent the object parts, the arcs represent the relations. The arc (i_1, i_2) is weighted by the search effort $\Omega^r_{i_1,i_2}$ and the arc (i_2, i_1) is weighted by $\Omega^r_{i_2,i_1}$. Figure 5.22(a) shows the corresponding graph of the example.

The optimum search tree can now be obtained by computing the minimum spanning arborescence of G with respect to a certain root node j. In the relevant literature, the term *arborescence* is used synonymously with the term *tree*, however, it implies that the tree has directed arcs. The minimum spanning arborescence in a directed graph is defined as a directed spanning tree $H(V, E')$, where E' is a subset of E such that the sum of $\Omega^r_{i_1,i_2}$ for all (i_1, i_2) in E' is minimized. The directed spanning tree is defined as a graph that connects all nodes with $n^p - 1$ arcs, i.e., each node, except the root node, has exactly one incoming arc. To illustrate this definition, in Figure 5.22(b) a detailed view of a small subgraph with four nodes is shown. After selecting part 9 as the root part, the minimum spanning arborescence is calculated. The result is shown in Figure 5.22(c). It is easy to convince oneself that in the result three arcs are contained, where each non-root node has exactly one incoming

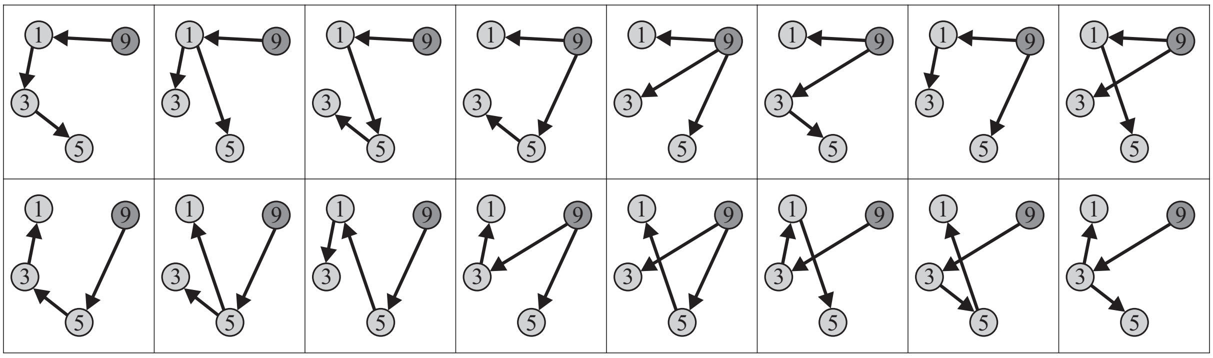
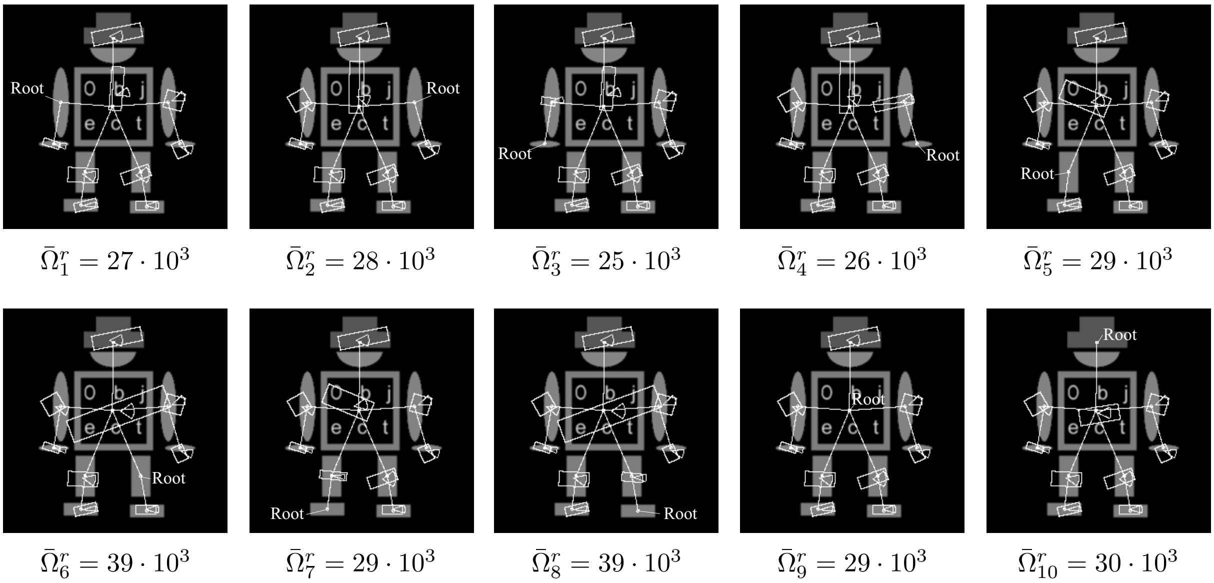


Figure 5.23: All possible spanning arborescences of the subgraph shown in Figure 5.22(b) with part 9 as root part

arc and that the overall sum of weights associated with the result is 10735. This is the minimum weight among all possible directed spanning arborescences rooted at part 9. For illustrative purposes Figure 5.23 shows all possible spanning arborescences of the respective subgraph. As can be seen, there are 16 different ways to connect the three nodes to the root node. The result of Figure 5.22(c) suggests that after the upper body (part 9) is found in the image, it is most efficient to search the left arm (part 1) and the left leg (part 5) relative to the upper body and to search the left hand (part 3) relative to the left arm.

The minimum spanning arborescence can be seen as the equivalent to the well-known minimum spanning tree in an undirected graph (if $\Omega^r_{i_2,i_1}$ would be symmetric one would obtain a undirected graph). The two most prominent algorithms to efficiently compute the minimum spanning tree are the Kruskal and the Prim algorithm (Graham and Hell 1985, Clark and Holton 1994). Unfortunately, these algorithms cannot be used or even extended to cope with directed graphs. Solving the problem of finding the minimum spanning arborescence in a directed graph is much more complicated in comparison to solving the equivalent undirected problem. A polynomial algorithm for the minimum spanning arborescence was independently proposed in (Chu and Tseng-Hong 1965), (Edmonds 1967), and (Bock 1971). In (Tarjan 1977) and (Gabow et al. 1986), efficient implementations of the algorithm are presented. The implementation used in this dissertation is presented in (Fischetti and Toth 1993). It makes use of simple data structures leading to a run time complexity of only $O(n^2)$, where n is the number of nodes in the graph. For a detailed description of the algorithm or of the implementation the interested reader should refer to the cited literature.



$\bar{\Omega}^r_1 = 27 \cdot 10^3$ $\bar{\Omega}^r_2 = 28 \cdot 10^3$ $\bar{\Omega}^r_3 = 25 \cdot 10^3$ $\bar{\Omega}^r_4 = 26 \cdot 10^3$ $\bar{\Omega}^r_5 = 29 \cdot 10^3$

$\bar{\Omega}^r_6 = 39 \cdot 10^3$ $\bar{\Omega}^r_7 = 29 \cdot 10^3$ $\bar{\Omega}^r_8 = 39 \cdot 10^3$ $\bar{\Omega}^r_9 = 29 \cdot 10^3$ $\bar{\Omega}^r_{10} = 30 \cdot 10^3$

Figure 5.24: The minimum spanning arborescences for each of the ten object parts serving as root part and the associated overall search efforts are shown. Additionally, the relations between two adjacent nodes are superimposed.

It is obvious that for different root parts j different minimum spanning arborescences are obtained with different overall weights $\bar{\Omega}_j^r$. Therefore, for each root part the associated minimum spanning arborescence is computed. Figure 5.24 shows the result for each of the ten object parts serving as root part. It should be noted that although in this example the same two object parts are directly connected in all minimum spanning arborescences it is not necessarily the case in general.

5.4.3 Root Part Ranking

To complete the optimum hierarchical search strategy, the question remains, which part to choose as the root part. One criterion for a suitable root part is a small overall search effort $\bar{\Omega}_j^r$ of the associated minimum spanning arborescences.

However, $\bar{\Omega}_j^r$ only describes the search effort that must be spent during the relative search. Therefore, the effort Ω_j^{root} that must be spent to search the root part j itself must be considered as a second criterion:

$$\Omega_j^{root} = R \cdot C \cdot (\varphi_j^{max} - \varphi_j^{min}) \cdot \frac{n_j^{m,top}}{4^{n_j^l - 1} \cdot \Delta\varphi_j^{top}} \quad , \tag{5.29}$$

where R and C are the number of rows and columns in the search image, and hence describe the position search range for the root object part, while $(\varphi_j^{max} - \varphi_j^{min})$ describes the orientation angle search range.

As a third criterion the uniqueness of the root part must be considered. The root part should exhibit as few symmetries and as few similarities to other object parts as possible. Assume that the left leg (part 5) serves as root part in the online phase and is searched within the full orientation search range. Because of its rotation symmetry and its similarity to the right leg, it would be found at four different poses (ignoring possible clutter in the search image). When looking at the associated search tree the upper body and the left foot must be searched relative to each of the four poses of the left leg. Consequently, the search effort in the online phase increases with the number of symmetries and similarities. Therefore, the symmetries and similarities of all object parts are determined using the analysis described in Section 5.3.3.2, which matches object part j to itself and to all other object parts. Assume that object part j has been found n_j^{sym} times on itself and n_j^{sim} times on other object parts during the analysis. Then, the search effort of the relative search $\bar{\Omega}_j^r$ must be multiplied by $(n_j^{sym} + n_j^{sim})$ in order to approximately estimate the influence of the non-uniqueness of the root part on the search effort. One could argue that it is sufficient to only multiply the relative search effort of the parts that are adjacent to the root part since the search can be aborted if the adjacent parts are not found. However, the multiplication of the overall relative search effort is legitimated since the object recognition, which will be described in Section 5.5, should be able to cope with occlusions. Thus, the search cannot be stopped if one object part is missing.

Finally, the search effort Ω_j that is associated with the root part j is obtained:

$$\Omega_j = \Omega_j^{root} + (n_j^{sym} + n_j^{sim})\bar{\Omega}_j^r \quad . \tag{5.30}$$

By sorting the possible root parts with respect to Ω_j in ascending order one obtains a root part ranking that expresses the suitability of all object parts to serve as the root part. In Table 5.1 the respective ranking of the example is presented. It can be seen that the head (part 10) and the upper body (part 9) are best suited to serve as the root part. This is because they both do not show any rotation symmetry or similarity to other object parts and because five pyramid levels can be used during the search. In contrast, taking one of the two hands (part 3 or part 4) as root part would result in the highest search effort: they both exhibit symmetries and mutual similarities. Furthermore, only three pyramid levels can be used because of their small size.

After this step, the creation of the hierarchical model is completed. However, a manual selection of the root part by the user is still reasonable. This is because the selection of a suitable root part also depends on the application (cf. Section 5.5.1). Therefore, the root part ranking is returned in order to help the user to select the appropriate root part for his specific application. Consequently, the search trees for all root parts are stored in the hierarchical model. In the online phase, the search tree that is associated with the user-specified root part is selected from the hierarchical model and used to search the object. Summing up, the hierarchical model consists of the rigid models of the object parts, the relations between the parts, and the hierarchical search strategy. The hierarchical search strategy is represented by the optimum search trees, which are given by the minimum spanning arborescences, and the root part ranking.

Rank	1	2	3	4	5	6	7	8	9	10
Root part j	10	9	7	8	5	6	1	2	3	4
$\Omega_j \cdot 10^{-5}$	4	5	21	21	26	27	32	32	140	140

Table 5.1: The root part ranking expresses the suitability of the object parts to serve as the root part. Parts 10 and 9 are best suited, whereas parts 3 and 4 would result in the highest search effort.

As a last point, a special case should be discussed that, however, is rather rare in practice: assuming a compound object that consists of identical object parts, then no distinct root part can be determined. For example, a chain consists of several identical links. In this case, the presented approach cannot be used because solving the ambiguities during the training would fail. This is because there is no preferable configuration of the links since the overall configuration itself is ambiguous. However, even if a hierarchical model is available its use would not be advisable because the root part would be found as many times as links are contained in the chain. Then, for each found instance of the root part the hierarchical search would be started, which leads to a high computational effort. Fortunately, in this case the use of the hierarchical model is not necessary. In contrast, it is sufficient to search only one single link in the image. The search already returns all instances of links, and thus the matches of all object parts. Hence, no further (relative) searches need to be performed. However, the correspondence problem still needs to be solved. It should be noted that in most cases when dealing with objects that consist of identical parts, it is possible to determine an auxiliary root part. For example, if several identical modules on a circuit board must be recognized, it is desirable to use the hierarchical model, and hence profit from the inherent determination of the correspondence. This can be achieved by including an additional object part in the compound object, e.g., a different module, a corner of the circuit board, a fiducial mark, or any other print on the board. The additional object part can then serve as the root part.

5.5 Object Recognition

After the hierarchical model has been generated in the offline phase, it is used to efficiently search the compound object in an arbitrary search image during the online phase. In addition to the high efficiency, the inherent determination of the correspondence of the object parts within the hierarchical model makes a solution of ambiguities during the online phase unnecessary. Section 5.5.1 outlines the principle of the hierarchical object recognition. Section 5.5.2 describes extensions for the handling of special problems that can occur during recognition.

5.5.1 Principle of Hierarchical Object Recognition

The recognition of the compound object in a given search image is based on the hierarchical model. As additional input data the user may specify the root part that should be used during the search if it differs from the first entry in the root part ranking. Furthermore, the user may restrict the search space within the search image. The recognition of the compound object is then split into two steps. In the first step, the root part is searched within the user defined search space. In the second step, starting from the pose of the root part, the poses of the remaining parts are determined during the relative search according to the search tree that is associated with the specified root part. As another parameter, the search order that is applied to the search tree can be chosen to be, e.g., breadth-first search or depth-first search (see Figure 5.8(a) in Section 5.2). The search order assigns each object part a number that describes its position within the search order.

Despite the automatically computed root part ranking, a manual selection of the root part may still be reasonable. This is because the computation of the root part ranking is only based on the endeavor to minimize the search effort. However, the selection of the root part not only affects the computation time, but the root part plays another decisive role in the case of occlusions or missing object parts: it is important that the root part of the compound object instance in the search image is recognized. If the root part cannot be recognized, e.g., because of occlusions or because it is missing in the search image, then the recognition of the remaining parts becomes more difficult. Therefore, in case severe occlusions or missing parts must be expected, the user should ensure to select a root part that is not only accompanied by a small search effort but also by a high possibility to be recognized. Because this selection depends on the object and the application, and thus requires background information, it cannot be automated. Nevertheless, for the case that no such prominent object part can be specified, e.g., because all object parts are approximately equal, in Section 5.5.2.4 a solution that is able to cope with the case of a missing root part is proposed.

In the following, the principle of the hierarchical recognition is explained in more detail. Let q_i, $i = 1, \ldots, n^p$ be the object part at position i within the search order and p_i, $i = 2, \ldots, n^p$ be the associated predecessor part within the search tree. For example, the root part is at the first position within the search order, and hence is represented by q_1. It should be noted that the root part does not have a predecessor part in the search tree. The aim of the recognition is to return one hierarchical match for each instance of the compound object in the search image. A hierarchical match comprises the matches of all object parts that belong to the same object instance. Therefore, it contains at most one pose for each object part. Each match of the root part instantiates one hierarchical match, which at this time only contains the respective pose of the root part. Then, according to the search order the next object part q_i is successively selected. The search space for part q_i is calculated based on the pose of the associated predecessor part p_i. The search space is split into the position search space for the reference point and the orientation search space. The position search space is described by the rectangle that is given by the position relation between part p_i and q_i transformed according to the pose of part p_i. Accordingly, the orientation search space is given by the transformed orientation relation. Let $(\boldsymbol{o}^s_{p_i}, \varphi^s_{p_i})$ be the pose of the predecessor part in the search image. Then the position search space for part q_i is represented by the rectangle that is described by the parameters $\boldsymbol{c}_{q_i}$, a_{q_i}, b_{q_i}, and β_{q_i}:

$$\begin{aligned} \boldsymbol{c}_{q_i} &= \boldsymbol{o}^s_{p_i} + \boldsymbol{R}(\varphi^s_{p_i})\boldsymbol{c}^r_{p_i,q_i} && (5.31)\\ a_{q_i} &= a^r_{p_i,q_i} && (5.32)\\ b_{q_i} &= b^r_{p_i,q_i} && (5.33)\\ \beta_{q_i} &= \varphi^s_{p_i} + \beta^r_{p_i,q_i} \quad . && (5.34) \end{aligned}$$

The orientation search space is described by the angle interval $[\varphi^{min}_{q_i}, \varphi^{max}_{q_i}]$:

$$\begin{aligned} \varphi^{min}_{q_i} &= \varphi^s_{p_i} + \varphi^{r,min}_{p_i,q_i} && (5.35)\\ \varphi^{max}_{q_i} &= \varphi^s_{p_i} + \varphi^{r,max}_{p_i,q_i} \quad . && (5.36) \end{aligned}$$

Finally, part q_i is searched by scanning the computed search space. The obtained pose is stored within the hierarchical match. If there are several hierarchical matches (e.g., because multiple instances of the root part have been found) the search space is computed for each hierarchical match separately. This process is successively repeated for all object parts.

To rate the quality of the hierarchical match, a score value s is computed by weighting the returned score values s_{q_i} of the single object parts:

$$s = \sum_{i=1}^{n^p} f_{q_i} s_{q_i} \quad , \tag{5.37}$$

where $f_{q_i} = w_{q_i} / \sum_{i=1}^{n^p} w_{q_i}$ represents the weighting factor of object part q_i and w_{q_i} is the respective weight. If an object part could not be found its score value is set to 0. Because the score values of the single parts approximately indicate the fraction of occluded edge pixels, it is reasonable to set w_{q_i} to the number of model edge pixels of the associated object part. Thus, the contribution of a single part to the score is proportional to the number of its model edge pixels. According to the recognition of rigid objects a minimum score value s^{min} can be set by the user. Consequently, a hierarchical match can be discarded after j object parts have been searched whenever the following condition holds:

$$\tilde{s}_j + \bar{s}_j < s^{min} \quad , \tag{5.38}$$

where $\tilde{s}_j = \sum_{i=1}^{j} f_{q_i} s_{q_i}$ denotes the score that is obtained from the parts that have been searched already. The score that is at most reachable by the remaining parts is denoted as $\bar{s}_j = \sum_{i=j+1}^{n^p} f_{q_i}$. For this, perfect matches, i.e., $s_{q_i} = 1$, are assumed for the parts that still have to be searched. By evaluating (5.38) after each searched object part, unnecessary computations can be avoided, which leads to an increased efficiency.

Finally, for each found instance of the compound object the precise poses of all found object parts that belong to the associated hierarchical match are returned. Furthermore, the score value of the compound object as well as the score values of all object parts are returned.

A difference between the SBM and the MGHT should be noted when dealing with restricted search spaces as in the case of the hierarchical search. Because the MGHT shows inherent translation invariance, it is difficult to benefit from the prior information about the object position given by the position search space. This problem is similar to the problem that occurs when tracking matches through the pyramid (cf. Section 4.2.3.2). Unfortunately, the principle of the blurred region cannot be applied in the proposed way because the position search spaces of the object parts vary in dependence on the corresponding predecessor part. This would require an exploding number of blurred regions to store within the hierarchical model. In order to avoid huge amounts of required memory, the domain restriction must be either computed in the online phase or completely neglected. However, both alternatives increase the computation time. Hence, using the SBM for the recognition of compound objects is more efficient than using the MGHT. Apart from that, similar results can be expected for both approaches, however, a lower robustness against changes in brightness must be accepted when the MGHT is used.

5.5.2 Practical Extensions

5.5.2.1 Missed Object Parts

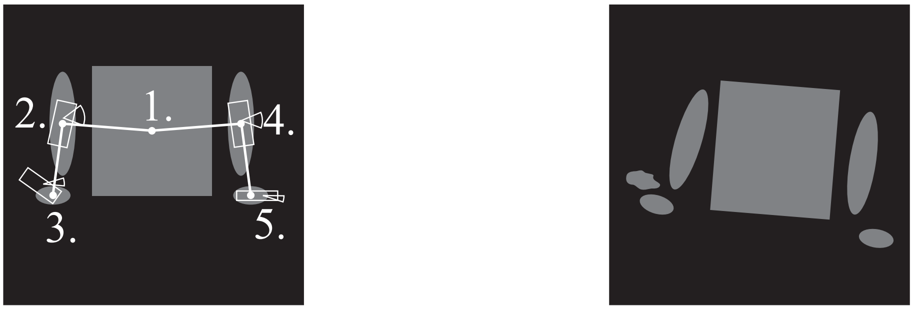
Sometimes it may happen that one object part cannot be found. Consequently, its pose is not available to restrict the search space for the object parts that reside directly below the missed object part in the search tree. On the one hand, it should be avoided to search the respective parts within an unrestricted search space. This would lead to an increased computation effort and to possibly ambiguous matches. On the other hand, the approach should be robust against occlusions. Therefore, three different strategies that make the search for an object part possible even if the pose of the predecessor is unavailable have been implemented: The first strategy is to step back in the search tree until a found object part is available. The search is then performed relative to the pose of this object part. Thus, in the worst case the search is performed relative to the pose of the root part. The second strategy is to perform the relative search from the pose of the (already found) object part from which the search effort of the relative search is minimal. In a third (trivial) strategy all object parts that reside below the missed object part in the search tree are not searched at all but are also treated as missing. The second strategy is applied as default. However, the user may select the appropriate strategy for his specific task.

In some applications it is desirable that a pose is obtained even for the object parts that could not be found. Based on the poses of all found object parts within the hierarchical match the most likely pose of a missed object part can be calculated. For this, a weighted mean pose is calculated using the mean and standard deviation of the relative movements that have been computed in Section 5.3.5. From the pose of one found object part and the mean value of position and orientation relation to the missed part, the mean pose of the missed part can be calculated. This calculation can be done with respect to each found object part. The most likely pose is obtained by computing the weighted mean of all obtained mean poses, where the weight is proportional to the inverse variance of the relative movement.

5.5.2.2 Multiple Matches

In some cases it may happen that multiple instances of one object part are found despite the restricted search space. In this case the current hierarchical match is duplicated according to the number of found matches. Each match of the object part is then assigned to a different hierarchical match. The search is continued for all hierarchical matches.

Figure 5.25 shows a small example to illustrate the search for a compound object. It should be noted that the root part (upper body) is symmetric, i.e., its pose is ambiguous, and that in the search image shown in Figure 5.25(b) a clutter object is present that is similar to the left hand.



(a) Hierarchical model

(b) Search image

Figure 5.25: A compound object consisting of five object parts. In (a) the hierarchical model is visualized. Additionally, the position in the search order of each part is displayed. In the search image (b) a clutter object is present.

The progress of the search is shown in Figure 5.26. At first, the root part, which is the upper body in this example, is searched within the entire search space. Because the root part is rotationally symmetric, it is found twice. Thus, two hierarchical matches are initialized. They are shown in the first row of Figure 5.26. Because the contribution of the upper body to the edge pixels in the entire compound object f_{q_1} is 0.6, the score $\tilde{s}_1$ is 0.6.

The next part in the search order is the left arm. The respective search space in both hierarchical matches is additionally visualized in the first row. Because of the symmetric constellation of the arms with respect to the upper body, the left arm is found in both hierarchical matches (see second row of Figure 5.26). However, the search for the left hand leads to different results in both hierarchical matches. The search using the first hierarchical match results in two matches for the left hand, because of the additionally present clutter object in the image. Consequently, the first hierarchical match is duplicated once, yielding a third hierarchical match. The two matches of the left hand are then assigned to the first and the third hierarchical match, respectively. Because the score returned for the clutter object is less than 1, the score of the third hierarchical match is only $\tilde{s}_3 = 0.78$ in contrast to $\tilde{s}_3 = 0.8$ of the first hierarchical match. The search for the left hand using the second hierarchical match remains unsuccessful. The right arm, which is the next part in the search order must be searched relative to the pose of the upper body. However, the pose of the upper body is identical in the first and the third hierarchical match. Therefore, the search needs to be performed only once for both hierarchical matches. This prevents an increase in computation time when dealing with multiple matches. In contrast, the search for the right arm in the second hierarchical match must be performed. The last step is the search

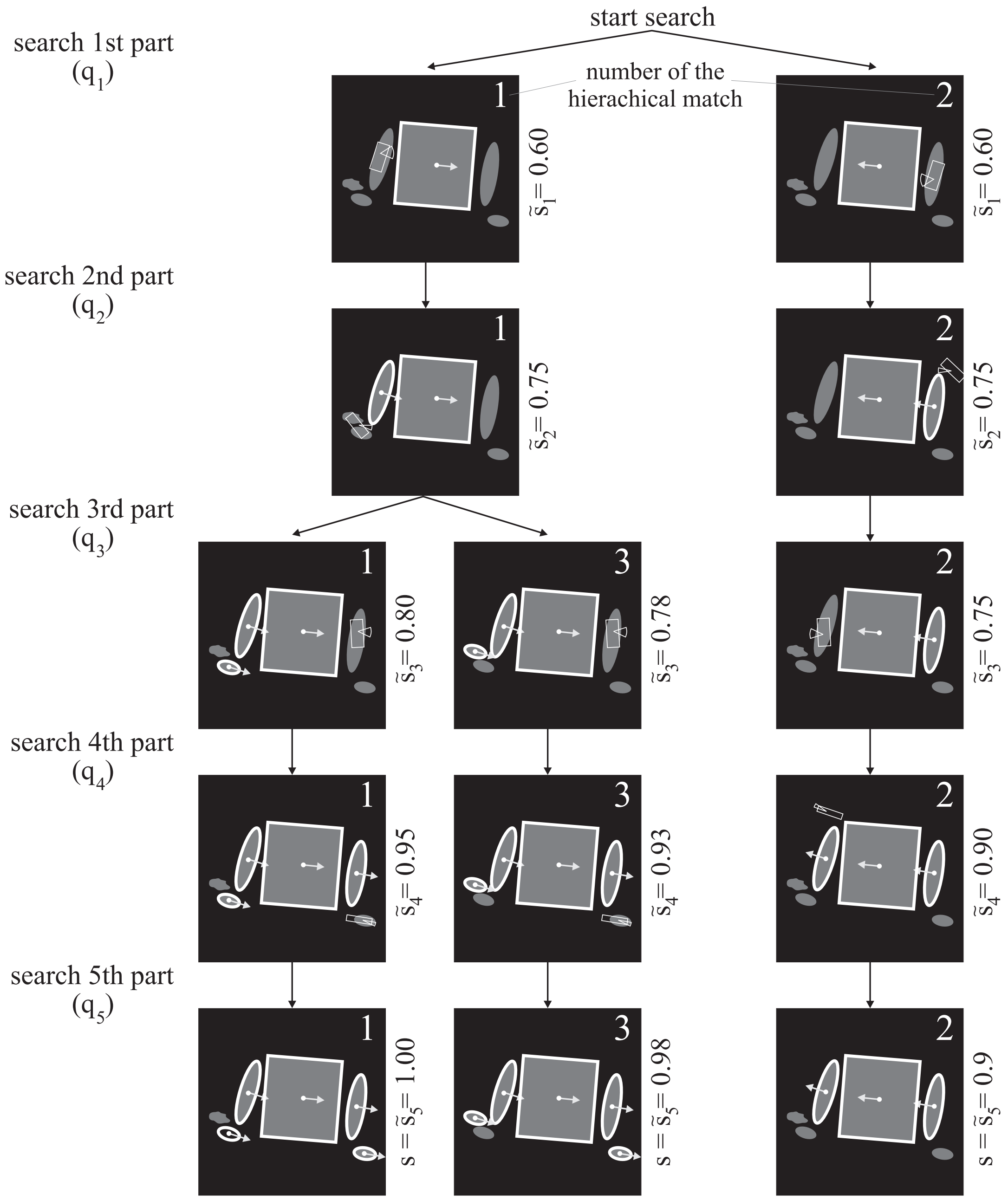


Figure 5.26: Search for the compound object using the hierarchical model. Found instances of object parts are represented by white edges. The orientation of the match is displayed as a white arrow. The search space is displayed using white rectangles and circle sectors. After the j-th part has been searched, the score $\tilde{s}_j$ is computed ($f_{q_1} = 0.60$, $f_{q_2} = 0.15$,$f_{q_3} = 0.05$, $f_{q_4} = 0.15$, and $f_{q_5} = 0.05$).

for the right hand. Finally, three hierarchical matches are returned. The first and the third hierarchical match only differ in the match of the left hand. Therefore, the two corresponding scores (s = 1.00 and s = 0.98) differ only slightly. The second hierarchical match represents a rotated instance of the compound object with occluded hands. Consequently, the corresponding score is lower (s = 0.90).

5.5.2.3 Elimination of Overlapping Matches

Sometimes it is desirable that hierarchical matches that represent the same instance of the object are eliminated. In the previous example (cf. Section 5.5.2.2), only the first hierarchical match should be returned.

Therefore, in a subsequent step after the search, the hierarchical matches are checked for mutual overlap. If the overlap fraction between two hierarchical matches exceeds a user-specified threshold the hierarchical match with the lower score is eliminated. To compute the overlap fraction, the object part matches are represented by their smallest enclosing rectangle. For each hierarchical match the union region of the smallest enclosing rectangles of all object part matches is computed. The overlap is computed by intersecting the union regions of two hierarchical matches. The overlap fraction is then obtained by dividing the area of the intersection region by the area of the smaller of both union regions. In order to save computation time, one can take profit of the fact that the overlap fraction for duplicated matches is 1 at the time of duplication. Therefore, it is sufficient to only check those object part matches for overlap that are different. Furthermore, the smallest enclosing rectangles for each object part can be computed in the offline phase. For the overlap check in the online phase they only need to be transformed according to the pose parameters of the object part matches. This facilitates an efficient computation of the overlap fraction.

5.5.2.4 Missed Root Part

An important point to discuss is the treatment of a missed root part. In some applications it cannot be ensured à priori that the root part of each instance of the compound object is found. Thus, the hierarchical search cannot be started. Consequently, the compound object cannot be found by the approach even if all object parts except the root part are visible.

Therefore, if the user specifies that the root part may be missing a special extension of the approach is applied. In this extension the search is not restricted to the use of a single root part. In contrast, the search is performed by successively selecting different root parts in accordance with the root part ranking. For each selected root part the associated search tree is used to search the remaining parts. The number of root parts that must be used can either be selected by the user or determined automatically based on the minimum score. If, for example, k root parts have been searched it is still possible that some instances of the compound object have not been detected yet. This is possible for object instances with exactly these k object parts occluded. Consequently, the score that can maximally be achieved for such object instances corresponds to the sum of weighting factors of the remaining $n^p - k$ root parts. Thus, no further root part needs to be searched when this sum is smaller than the user-specified minimum score.

Some particularities that arise when using several root parts should be discussed in the following. Firstly, the increasing computational effort must be mentioned. Fortunately, some matches of the current root part can be immediately eliminated without instantiating a new hierarchical match. This is done by checking whether a match of the current root part is identical to an already found match during the relative search of a previously used root part. The respective matches of the current root part can then be eliminated without the risk of missing an instance of the compound object. Thus, the computational effort can be reduced considerably. Nevertheless, the effort is still higher in comparison to the use of only a single root part. Therefore, the computation time is compared to the brute-force method that searches all object parts in the entire search space: Let Ω be the average computational effort of searching an object part in the entire search space. Accordingly, let Ω^r be the average computational effort of searching an object part relative to another part in a restricted search space, and hence $\Omega^r \ll \Omega$, in general. Furthermore, let n^{inst} be the number of object instances in the search image. The computational effort using the proposed hierarchical model with the extension of missing root parts can then be estimated as:

$$(1 - s^{min})n^p\Omega + n^{inst}(n^p - 1)\Omega^r \quad . \tag{5.39}$$

Here, $(1 - s^{min})n^p$ is the number of root parts that must be searched within the entire search space to ensure that all object instances with a score exceeding s^{min} are found. For each found instance the relative search must be performed for $n^p - 1$ object parts. In contrast, the computational effort using the brute-force method can be estimated as:

$$n^p\Omega \quad . \tag{5.40}$$

Consequently, the search using the hierarchical model is more efficient than the brute-force method if the following condition holds (assuming the worst case of $n^p \rightarrow \infty$):

$$s^{min} > n^{inst} \frac{\Omega^r}{\Omega} \quad . \tag{5.41}$$

This condition is not very restrictive, and hence fulfilled in most applications. For example, assuming that at least 50% of the compound object is visible ($s^{min} = 0.5$) and $\Omega^r/\Omega = 0.05$, which is still a high ratio, the search using the hierarchical model is faster than the brute-force method if fewer than ten instances are present in the image. Apart from this it should be kept in mind that the substantial advantage of the inherently determined correspondence of the object parts still remains even when using several root parts.

A last point that must be taken into account when dealing with several root parts is the possibility to introduce prior knowledge by the user about the pose of the first root part in the search image. This knowledge is used to restrict the search for the first root part. To take advantage of this prior knowledge when performing the search for other root parts, the search space for the other root parts must be explicitly determined. For this, the restricted search space of the first root part is propagated through the search tree that is associated with the first root part. The propagation is performed by successively accumulating the relative search spaces to the user-specified search space over the path in the search tree that starts at the first root part and ends at the current root part. The orientation search space is trivially computed by successively adding the orientation search spaces. Figure 5.27 shows the more complex calculation of the position search space of the second root part based on the user-specified search space of the first root part. In Figures 5.27(a)–(d) the exact calculation is shown in detail. The user-specified orientation search space $[\varphi_1^{min}, \varphi_1^{max}]$ for the first root part is propagated to the range of reference positions of the second root part.

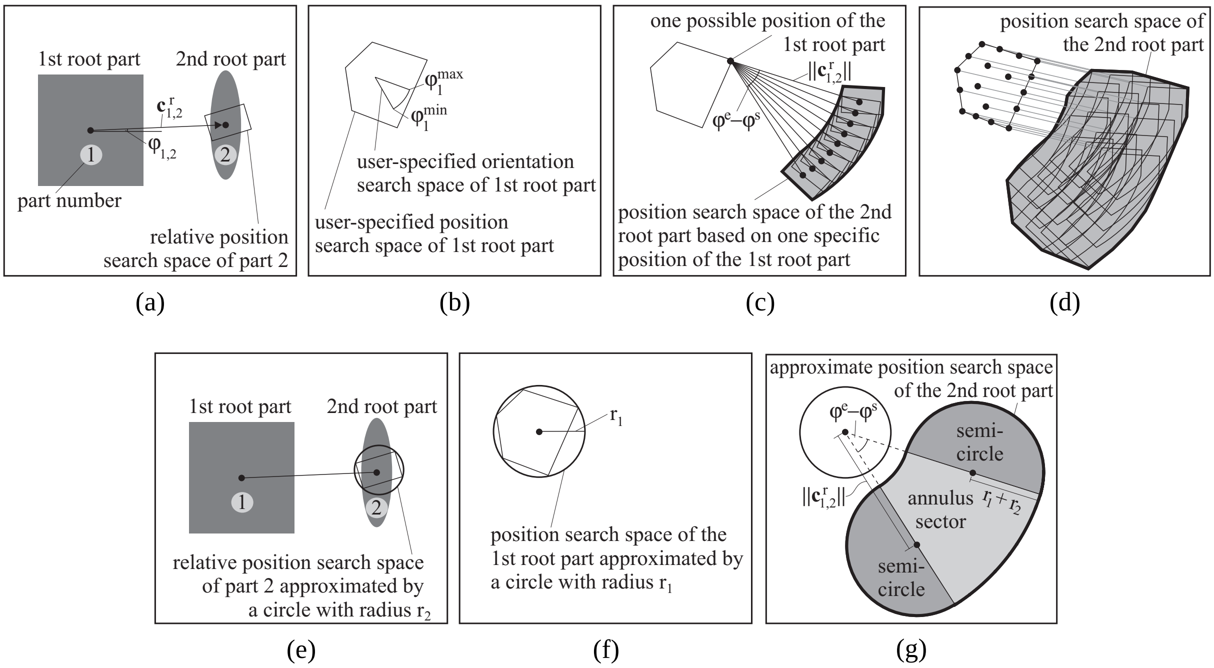

1st root part
2nd root part
$\mathbf{c}^r_{1,2}$
$\varphi_{1,2}$
1
2
part number
relative position search space of part 2
(a)
φ_1^{max}
φ_1^{min}
user-specified orientation search space of 1st root part
user-specified position search space of 1st root part
(b)
one possible position of the 1st root part
$\|\mathbf{c}^r_{1,2}\|$
$\varphi^e - \varphi^s$
position search space of the 2nd root part based on one specific position of the 1st root part
(c)
position search space of the 2nd root part
(d)
1st root part
2nd root part
1
2
relative position search space of part 2 approximated by a circle with radius r_2
(e)
r_1
position search space of the 1st root part approximated by a circle with radius r_1
(f)
approximate position search space of the 2nd root part
$\varphi^e - \varphi^s$
semi-circle
$\|\mathbf{c}^r_{1,2}\|$
annulus sector
$r_1 + r_2$
semi-circle
(g)

Figure 5.27: Computation of the position search space for the second root part based on the user-specified search space for the first root part. The exact computation is shown in (a)–(d). Because of the expensive computation, an approximate solution is shown (e)–(g).

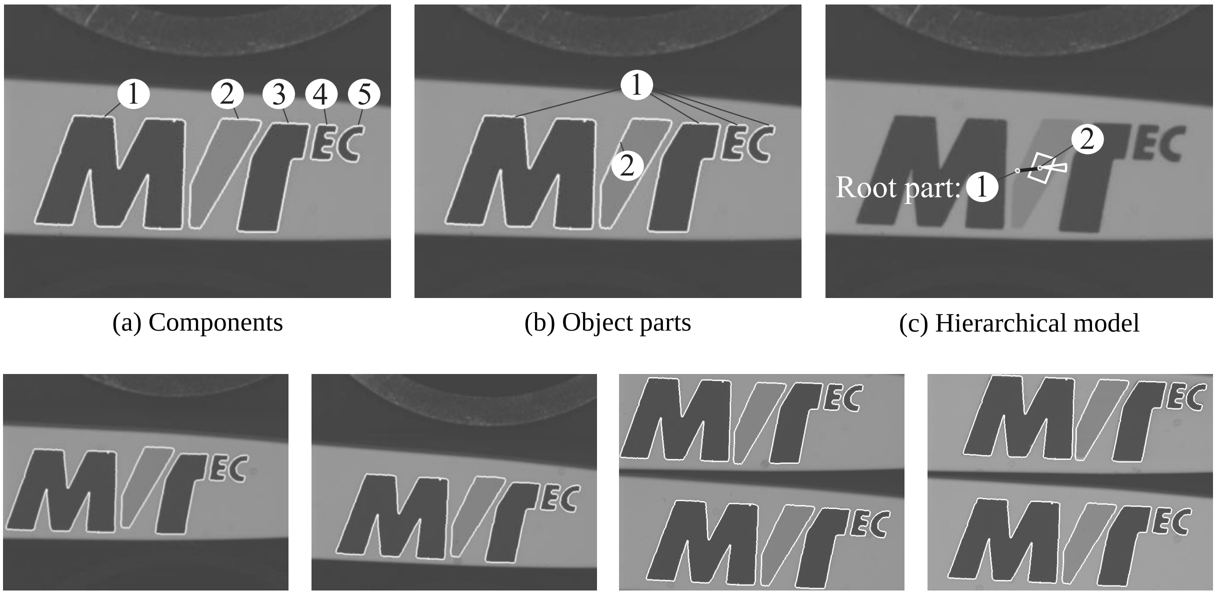
For one specific position of the first root part within the user-specified position search space the circular arc on which the centroids of all rectangles must fall that describe the position of part 2 can be calculated. Let $\varphi_{1,2} = \arctan(y_{1,2}/x_{1,2})$, where $x_{1,2}$ and $y_{1,2}$ are the coordinates of the vector $\boldsymbol{c}^r_{1,2}$ that describes the relative position of part 2 with respect to part 1. Then the circular arc is defined by the radius $\|\boldsymbol{c}^r_{1,2}\|$ and the angle interval $[\varphi^s, \varphi^e]$, with the start angle $\varphi^s = \varphi_{1,2} + \varphi_1^{min}$ and the end angle $\varphi^e = \varphi_{1,2} + \varphi_1^{max}$. At each point on the circular arc the rectangle that describes the relative position search range of part 2 is superimposed. The envelope of all rectangles describes the position search space of the second root part based on one specific position of the first root part (see Figure 5.27(c)). To take all possible positions of the first root part into account, the resulting image region must be enlarged by using the Minkowski addition (Pratt 2001) with the user-specified position search space of the first root part as structuring element. The reference point of the structuring element must correspond to the previously specified position of the first part. The result is shown in Figure 5.27(d). One can see that the exact computation of the position search space is rather expensive. Therefore, an approximate solution is proposed. The single steps are shown in Figures 5.27(e)–(g). The user-specified search space for the root part as well as the relative search spaces are approximated by the smallest enclosing circles with radius r_1 and r_2. Consequently, the search region for the second root part can be represented by an annulus sector with an inner radius of $\|\boldsymbol{c}_{1,2}^r\| - r_1 - r_2$, an outer radius of $\|\boldsymbol{c}_{1,2}^r\| + r_1 + r_2$ and an angle interval of $[\varphi^s, \varphi^e]$. Finally, two semi-circles with radius $r_1 + r_2$ must be appended at both ends of the annulus sector. Although the resulting approximate position search space is larger than the exact solution, it can be computed much more efficiently. Therefore, one can take profit from the user-specified prior information about the position and orientation of the first root part even when using other root parts.

5.6 Examples

Because of the novelty of the proposed approach for recognizing compound objects, there exist no comparable recognition methods, and hence a comparison with the performance of other approaches is not possible. Fortunately, many properties of the compound object recognition approach are inherited from the underlying rigid object recognition approach. I.e., in principle the results of the evaluation in Section 4.4 can be transferred. However, a remark on the robustness should be annotated. The calculation of the score value of the compound object differs from the calculation in the rigid case. This influences the robustness of the approach since the score value is used to decide on the presence of an object instance. Furthermore, the user-specified parameters influence the robustness of the approach: depending on the selected treatment of missed object parts, and depending on the handling of missing root parts, the robustness against occlusions changes. This should be kept in mind by the user when specifying the corresponding parameters. Nevertheless, it is possible to evaluate the computation time in comparison to the brute-force method that searches all object parts in the entire search space independently from each other. In this section, selected examples that emphasize the considerable advantages of the proposed approach are presented. The image size in all examples is 640×482. All computations are performed on a 2 GHz Pentium 4.

In the first example (see Figure 5.28), the company logo that was already introduced in Figure 2.6 is used as the compound object. The model image, a ROI that contains the entire print on the pen clip, and 10 example images are passed as input data to the process of training the hierarchical model. The example images show the relative movement of the light gray letter with respect to the dark gray letters. As the first intermediate result the automatically found components are visualized in Figure 5.28(a) by superimposing their edges on the model image in white. For each letter a separate component was detected. During the pose determination of the components in the example images no ambiguities occurred because the components are free of any similarities or symmetries. In Figure 5.28(b) the extracted object parts that are obtained from clustering the similarity matrix are shown. The threshold for the minimum probability p^{min} was set to 0.5. As one would expect, the dark gray letters are clustered into one rigid object part. Although no rectification was applied to the images in this example, the relative movement between both parts was large enough to be easily separated from pseudo movements that are caused by the radial distortions. The complete training process took 25 s.

In the next step, the hierarchical model was created. Because the logo is expected to appear only in a very limited orientation range, the model was created in a small angle interval. This was done by restricting the expected orientation angle of the part that represents the dark gray letters to $[-20°, +20°]$. In Figure 5.28(c) the object parts are represented by their reference points. The part that represents the dark gray letters was



(a) Components (b) Object parts (c) Hierarchical model

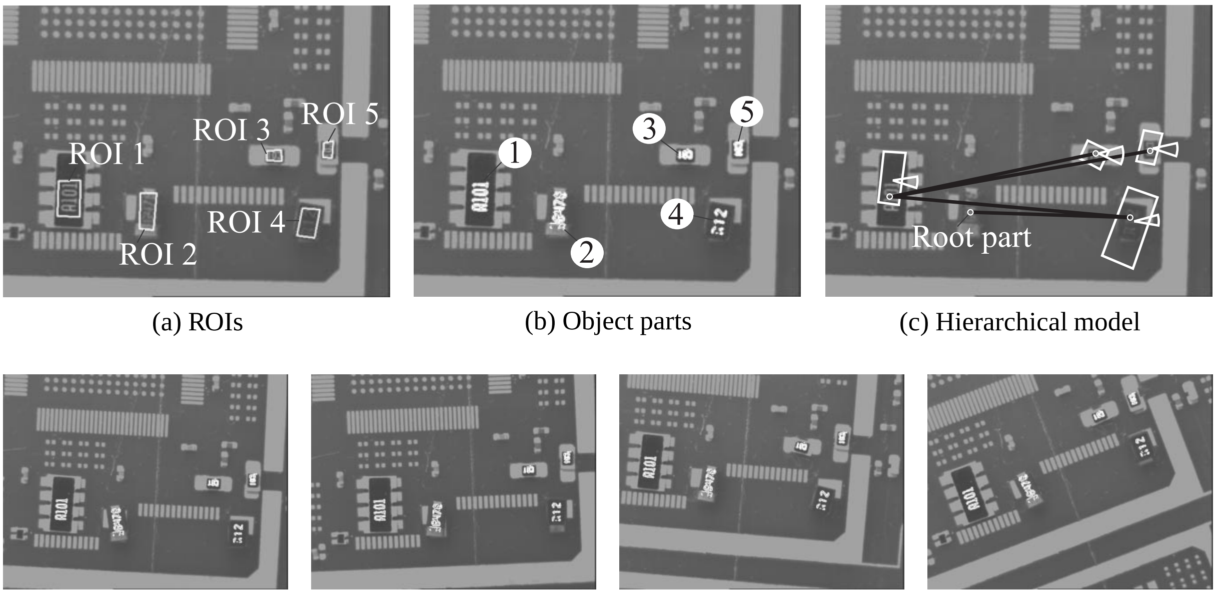
(d) Search image 1 (e) Search image 2 (f) Search image 3 (g) Search image 4

Figure 5.28: Recognition of the print on a pen clip. The pose of the light gray letter varies with respect to the dark gray letters.

selected as root part by the approach. One can see that the relative movement of the light gray letter is very small in position as well as in orientation. Because only two object parts are involved in this example, the search tree is degenerated to only two nodes with one single connection (displayed as a bold black line). Generating the hierarchical model took 6 s.

Finally, the hierarchical model was used to search the object in 20 different search images that are distinct from the example images that were used to train the hierarchical model. Figures 5.28(d)–(g) show four examples, in which the returned poses of the parts are visualized by their edges superimposed in white. The average computation time of the recognition process was 21 ms if one object instance was present in the image and 27 ms if two instances were present (as, e.g., in Figures 5.28(f) and (g)). The recognition of the root part took 15 ms, the recognition of the second part for each instance about 6 ms. In contrast, the independent recognition of both parts without using the hierarchical model would take about 30 ms (one instance) and 37 ms (two instances), respectively. Therefore, a speed-up of about 40% is received when using the hierarchical model. Although in this case the improvement in computation time that is obtained by the use of the hierarchical model is not enormous, there is still the additional advantage of the inherently determined correspondence. Thus, whenever more than one object is present in the image, the correspondence between the found object part instances and the correct compound object is implicitly given.

The second example (see Figure 5.29) deals with the circuit board that was introduced in Figure 2.8. In this example, the object parts are specified manually by the user because obviously an automatic extraction of the components using the proposed approach would fail. Therefore, instead of one ROI, now five ROIs are passed to the algorithm, each representing one object part (see Figure 5.29(a)). The associated object parts' edges are shown in Figure 5.29(b). To train the hierarchical model, 12 example images were provided. Also in this example no rectification was needed. Because two of the five electronic modules are identical (part 3 and part 5), the approach had to solve the occurring ambiguities during the training. It took 24 s to train the hierarchical model. The creation of the hierarchical model was restricted to an orientation angle interval of $[-45°, +45°]$. The final hierarchical model is displayed in Figure 5.29(c). Part 2 was recommended by the approach to serve as the root part, from which part 4 is searched. The pose of part 4 is then exploited to restrict the search space of part 1, from which finally part 3 and part 5 are searched. Because of the relatively small size of the object parts, the model creation only took 3 s.



(a) ROIs (b) Object parts (c) Hierarchical model

(d) Search image 1 (e) Search image 2 (f) Search image 3 (g) Search image 4

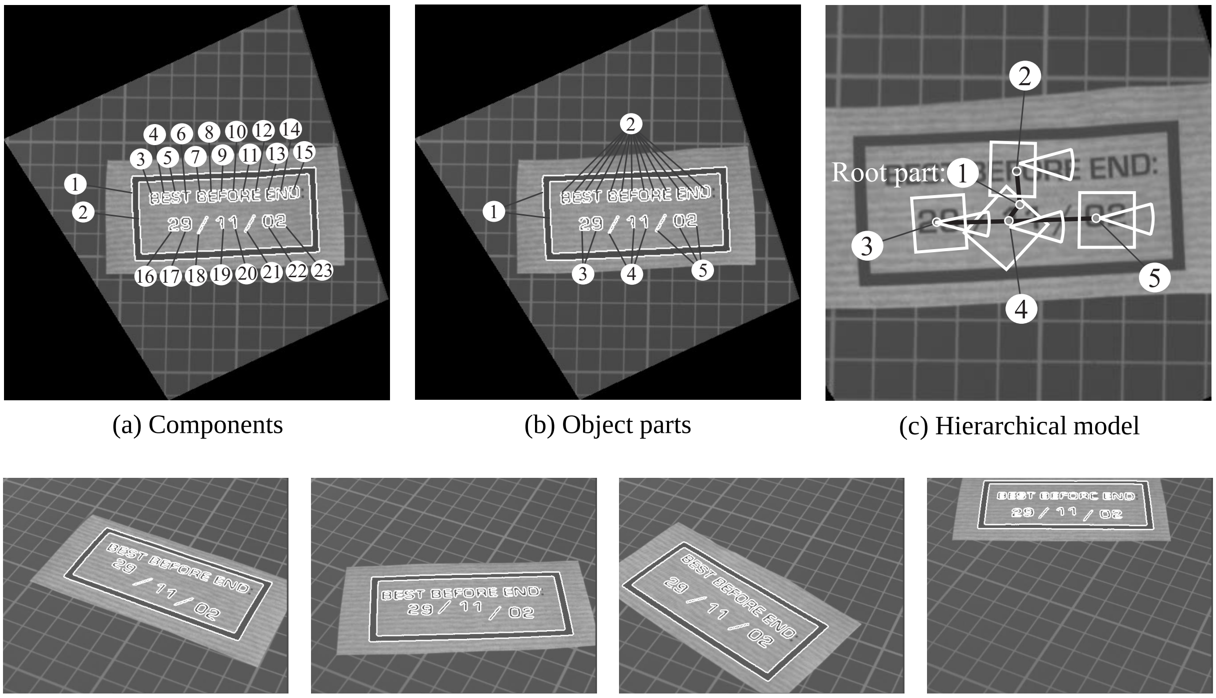
Figure 5.29: Recognition of electronic modules on a circuit board. The relative position and orientation of the modules on the board vary slightly.

In the online phase, the compound object was recognized in several search images. On average, it only took 20 ms to find the object, of which 14 ms must be attributed to the recognition of the root part and only 6 ms to the recognition of all remaining parts. In comparison, without using the hierarchical model the search would take 240 ms and the correspondence would remain unsolved. Thus, an impressive speed-up of 1100% is obtained. This example demonstrates the obvious substantial advantages of the hierarchical recognition.

In the third example (see Figure 5.30), the print on the label that was already shown in Figure 2.7 is used as the compound object. However, in this example the camera was not mounted perpendicular to the object plane, resulting in severe projective distortions (see Figure 3.6). Therefore, a camera calibration was necessary in order to rectify the images. As described in Section 3.5, 15 images of a calibration target were taken to calculate the rectification map (see Figure 3.7). The rectified images were defined by choosing an image size of 515 × 527 with a pixel size of 0.32 mm. The camera calibration took 2 s and the computation of the associated rectification map 120 ms.

Based on the rectified model image and the rectified example images the training was started. The compound object was defined to be the print on the label, and hence a rectangular ROI that encloses the entire print in the model image was passed to the training algorithm. The rectified model image together with the result of the initial decomposition is shown in Figure 5.30(a). One can see that each letter of the string “BEST BEFORE END” and each digit of the date “29/11/02” represents one component. Additionally, the inner and the outer rectangle of the black border were found to constitute two separate components. 18 rectified example images were made available and were passed to the training.

The object parts that were returned after analyzing the example images are displayed in Figure 5.30(b). Again, the threshold for the minimum probability was set to 0.5. Because the letters of the string do not exhibit any relative movement, they were clustered into one rigid object part. The same holds for the inner and the outer rectangle of the border. Furthermore, the date was grouped into three rigid object parts. In this example, several ambiguities were successfully solved by the algorithm: both rectangles, the letters “S”, “N”, and “O” as well as the digit “0” and the slash “/” show rotation symmetry, and hence are found at least twice in each example image. Furthermore, there are several mutual similarities between different components: the letters “B”,“E”, as well as the digits “1”, “2”, and the slash “/” appear more than once. Additionally, the letter



(a) Components (b) Object parts (c) Hierarchical model

(d) Search image 1 (e) Search image 2 (f) Search image 3 (g) Search image 4

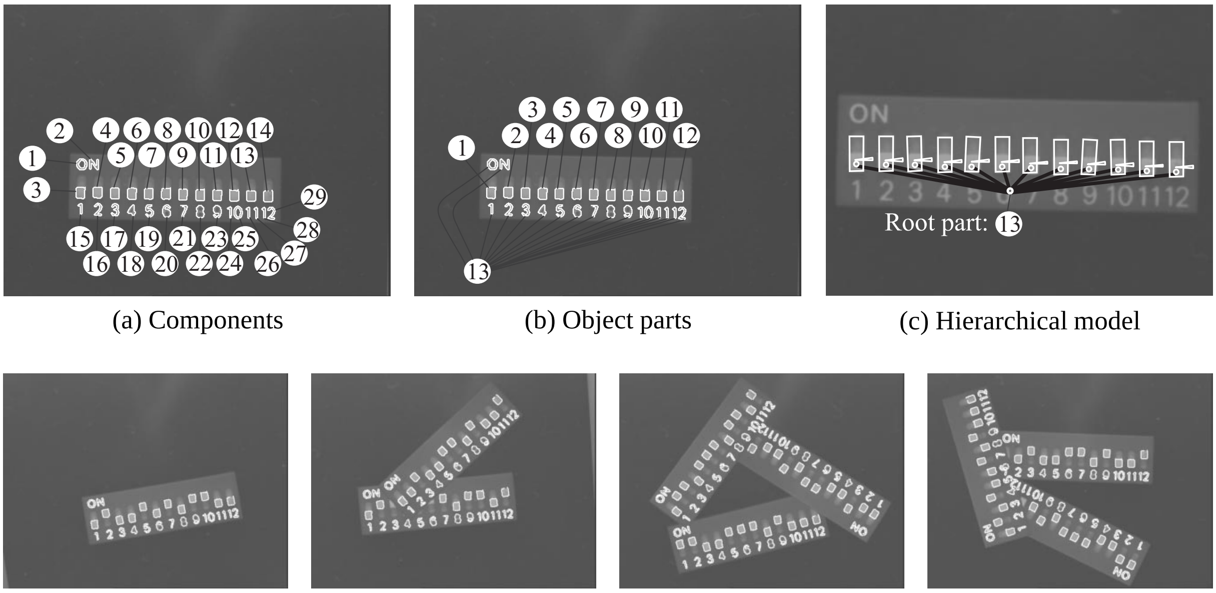
Figure 5.30: Recognition of the print on a label under projective distortions. The rectangular border, the string, and three parts of the date move with respect to each other.

“O” and the digit “0” show high similarity. Because of the large number of components, it took 12 minutes to train the hierarchical model. One possible way to speed up the training is to restrict the search space of the components. Here, the example images have been acquired such that the orientation of the components varied only slightly. Thus, the orientation search space during the training could be restricted to the interval $[-20^\circ, +20^\circ]$, which resulted in a computation time of 3 minutes. Furthermore, ambiguities due to orientation symmetries are avoided.

The subsequent creation of the hierarchical model was not restricted to an orientation range, but performed within the full orientation range of 360°. The resulting search tree and the associated relations are displayed in Figure 5.30(c). The rectangular border of the label was recommended by the approach to serve as the root part despite its rotation symmetry. Indeed, the search would be slower if the string would be chosen as root part because the number of associated pyramid levels of the string is one less than that of the rectangular border. The suggested search tree implies to search the string and the middle part of the date relative to the pose of the border. Finally, based on the pose of the middle part of the date the two remaining parts of the date are searched. The computation time to create the model was 16 s.

The online phase in this example consists of the rectification of the search image and the subsequent search with the hierarchical model. In the Figures 5.30(d)–(g), four search images are shown. To validate the resulting matches, the edges of the found object parts are projected back from the world coordinate system into the original search image and displayed in white. One can see that despite the severe distortions and the relative movements of the object parts the compound object was correctly recognized in all search images. The complete online phase took only 51 ms on average: 8 ms for the rectification, 33 ms for the search for the root part, and 10 ms for the relative search for the remaining parts. Without the hierarchical model the search would take 512 ms. Consequently, the speed-up that is achieved in this example is higher than 900%.

In a last example (see Figure 5.31), a DIP switch module containing 12 switches represents the compound object. Because each switch can be toggled either on or off, the appearance of the entire module changes. Therefore, in order to train the relations between the single object parts it is sufficient to use 12 example



(a) Components (b) Object parts (c) Hierarchical model

(d) Search image 1 (e) Search image 2 (f) Search image 3 (g) Search image 4

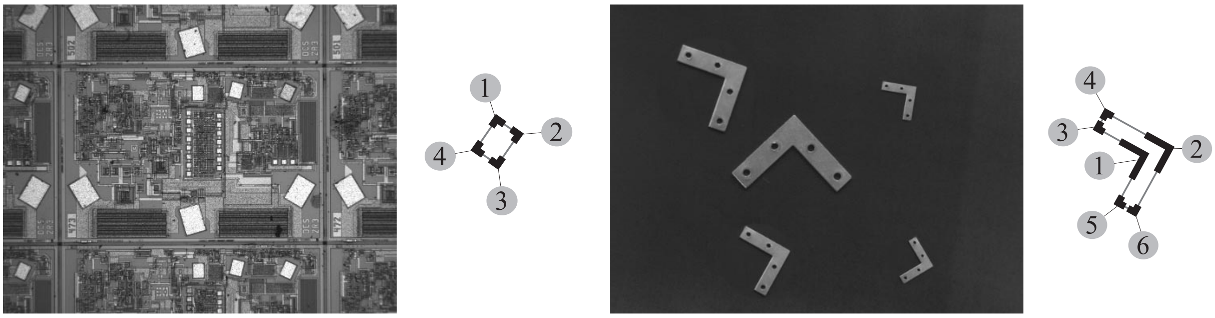
Figure 5.31: Recognition of a DIP switch module. Each switch can be toggled either on or off. Thus, the print on the module and all switches move with respect to each other.

images. In each example image another switch is toggled on, while the remaining switches are all toggled off. A rectangular ROI that encloses the print and all switches on the module is passed to the training algorithm. Also, in this example no rectification was necessary. In Figure 5.31(a) the automatically detected components are superimposed on the model image. Here, matches must be expected that are highly ambiguous: because all switches show identical square shapes, each switch was found at 48 different poses in each example image. Nevertheless, the algorithm was able to solve all ambiguities correctly. In Figure 5.31(b) the final extracted object parts, which have been determined by setting the threshold for the minimum probability to 0.5, are shown. The entire print is combined in one object part, while each switch is represented by a separate object part. The computation time for training the hierarchical model was 11 minutes. When restricting the search for the components to the interval of $[-20°, +20°]$ the computation time can be reduced to 31 s.

Again, the hierarchical model is created without restricting the orientation range. The result is shown in Figure 5.31(c). As one would expect the print on the module is best suited to serve as the root part. The search tree suggests to search each switch relative to the pose of the root part. The time to create the hierarchical model was 17 s.

Finally, several search images that show different numbers of objects have been acquired. Four examples are shown in the Figures 5.31(d)–(g). Up to three objects appear simultaneously in the image. As a matter of course, also modules with arbitrary switch configuration that deviate from the configuration in the example images can be found. Because occlusions must be expected, the threshold for the minimum score of the root part was set to a low value of 0.6. In contrast the minimum score of the switches was set to 0.8. Lower values would lead to false positive matches because the switches differ only slightly from their white background. With these parameter values, all instances were found correctly. Of course, occluded switches could not be found. The recognition of the object took 22 ms (root: 14 ms, others: 8 ms), 38 ms (root: 22 ms, others: 16 ms), or 45 ms (root: 23 ms, others: 22 ms), depending on whether one, two, or three instances were found. The times for recognizing the parts independently without the hierarchical model would be 166 ms, 346 ms, and 682 ms. Furthermore, additional time would be necessary to solve the ambiguities. The gain in computation time can be expressed by the associated speed-ups of 650%–1400%.

Another advantage of the approach is that some objects can be recognized even if their size changes. Normally,



(a) White pads on a die (b) Parts (c) Metal angles (d) Parts

Figure 5.32: Recognition of objects with varying size. The white pads on the die in (a) can be represented by four corners that move with respect to each other (b). Analogously, the metal angles in (c) can be represented by six corners (d).

this would require a recognition approach that is able to handle similarity transformations. However, the proposed approach can model the changes in scale as relative movements. Figure 5.32 shows two examples. In the first example the white pads on the die must be recognized. The pads may occur at different sizes. Four appropriate model parts can be defined by passing four ROIs to the training. Each ROI contains one corner of the rectangle. Hence, by choosing the corners of the rectangle as model parts exploits the fact that angles are preserved under similarity transformations. It is sufficient to train the model with two example images one showing the smallest occurring pad and one showing the largest occurring pad. Thus, the trained relations between the parts cover all possible object scales. Analogously, in the second example metal angles must be recognized. Here, six object parts that represent the six corners are involved. Again, two example images are enough to train the model. Actually, it would be sufficient to recognize only two object parts in both examples to determine the pose of the object. The scale of the object can be determined from the distance between the two object parts. If more than two object parts are used the computation of the scale becomes ambiguous. In this case the scale can be determined in a least-squares adjustment by minimizing the distances between the scaled model and the returned poses of the object parts. In a similar way objects that are transformed by more general transformation classes can be recognized. However, instead of angles other appropriate geometric invariants must be found.

Chapter 6

Conclusions

In this dissertation, a novel approach for recognizing compound objects in real-time is proposed. A compound object is defined as a number of rigid object parts that show arbitrary relative movements. The basic idea behind the approach is to minimize the search effort, and hence the computation time, by restricting the search in accordance with the relative movements of the object parts. This naturally leads to the use of a hierarchical model: only the root object part, which stands at the top of the hierarchy, is searched within the entire search space. In contrast, the remaining parts are searched recursively with respect to each other within very restricted search spaces. By using the hierarchical model, prior knowledge about the spatial relations, i.e., relative movements, between the object parts is exploited already in an early stage of the recognition. Thus, the computation time is reduced dramatically. Furthermore, the hierarchical model provides an inherent determination of the correspondence, i.e., because of the restricted search spaces, ambiguous matches are avoided. Therefore, a complicated and expensive solution of the correspondence problem is not necessary.

The proposed strategy for recognizing compound objects requires an appropriate approach for recognizing rigid objects. In an extensive review of rigid object recognition methods, the generalized Hough transform proves to be one of the most promising candidates. Its inherent translation invariance, as well as the high robustness, are the most important advantages. Nevertheless, it is shown that there are still several modifications necessary to fulfill industrial demands. The method is extended to recognize objects at arbitrary orientations. This leads to high computation times and large amounts of required memory. Therefore, several effective extensions to increase the efficiency are proposed. The use of image pyramids, which leads to a multi-resolution model with an associated coarse-to-fine search, is a major improvement. It is shown that the benefit achieved by the use of the multi-resolution model can be further augmented: a method for optimally restricting the image domain that is processed during the coarse-to-fine search is proposed. By splitting the model into tiles, redundant processing steps are avoided and the gain in efficiency is further increased. Additionally, several new methods to enhance the degree of automation and robustness are proposed. Finally, the obtained discrete values for the object position and orientation are analytically refined to achieve a high accuracy. It is shown that this modified generalized Hough transform is about 650 times faster than the conventional generalized Hough transform in a standard example. The performance of the new approach is evaluated thoroughly by comparing it to three standard approaches and three high-end recognition tools. Furthermore, a second new approach, the shape-based matching (Steger 2002), which was developed simultaneously to the modified generalized Hough transform, is introduced and included in the evaluation. The evaluation shows that both new approaches are considerably superior to existing standard approaches. Their behavior with respect to robustness, accuracy, and computation time is better balanced in comparison to all other approaches, except for one high-end recognition tool, which shows comparable results. From this discussion it can be seen that both approaches fulfill the industrial requirements discussed in Section 2.2. Furthermore, it follows that the modified generalized Hough transform is more than simply a by-product of this dissertation. In contrast, it can be seen as one of the best stand-alone recognition approaches for rigid objects. The field of applications that can benefit from this new approach is almost unlimited. Not only applications that use the conventional generalized Hough transform can be improved, but most applications that require rigid object recognition can achieve a high performance with this approach.

The shape-based matching is chosen to serve as a module within the approach for recognizing compound objects because it has already been thoroughly tested and included in a commercial software library. Furthermore, in contrast to the modified generalized Hough transform, it shows true invariance against changes in brightness. To achieve a high degree of automation, the hierarchical model is automatically trained. For this, some example images that show the relative movements of the object parts are automatically analyzed and used to determine the rigid object parts as well as the spatial relations between the parts. This is very comfortable for the user because a complicated manual description of the compound object is avoided. During the subsequent creation of the hierarchical model, the optimum hierarchical search strategy is automatically derived. The strategy includes a rating of the ability of each object part to serve as the root part: parts that facilitate a fast search when used as root part receive a good rating. Additionally, for each part that might be selected as the root part, an associated search tree, which represents the hierarchical search, is automatically computed. The hierarchical model that is finally obtained is used to recognize the compound object in real-time. By default, the part with the best rating is selected as root part. However, in order to exploit prior knowledge about possibly occluded object parts, the root part may be selected by the user and passed as input parameter to the search. In this case, the ratings of the root parts may assist the user while specifying the desired root part. The search is then performed in accordance with the search tree that is associated with the specified root part.

The training of the hierarchical model is performed in several steps by following the principle of human visual perception. At first, the compound object is split into several small components. Then, the components are recognized in the example images. Components that do not exhibit any relative movements are merged into rigid object parts. Finally, the relations between the parts are determined. It is shown that the high degree of automation during the training of the hierarchical model is accompanied by several problems that must be solved. One major problem is the non-uniqueness of the components, which, e.g., can be caused by rotation symmetries or mutual similarities of the components. Thus, one component may be recognized several times in the same example image. To solve this correspondence problem, a new method that uses a global criterion is proposed to estimate the likelihood of the found instances. Finally, it is shown that the correspondence problem can be transformed into a bipartite graph matching problem, which can be solved efficiently using linear programming. Thus, for each component, the most likely instance is obtained with respect to the global criterion. Because the correspondence problem is already solved during the training, the resulting hierarchical model provides an inherent determination of the correspondence. Consequently, solving the correspondence during the object recognition is unnecessary, which is a considerable advantage of the proposed approach. To obtain the rigid object parts, the probability that two components belong to the same object part is computed. This computation is performed in a statistically sound manner by using hypotheses testing. The resulting square probability matrix is clustered and the corresponding components are merged into object parts.

The creation of the hierarchical model includes the derivation of the search trees that minimize the search effort. It is shown that this problem can be translated into the problem of finding the minimum spanning arborescence in a directed graph. This guarantees an exact and efficient solution. Finally, several practical extensions that must be considered during the hierarchical search conclude the approach for recognizing compound objects.

Furthermore, as a by-product a method for rectifying images in real-time is proposed. By combining this method with camera calibration, a very fast elimination of projective distortions and radial lens distortions from images becomes possible. Thus, the recognition of compound objects is extended to deal with projective transformations of the object plane. It is shown that the rectification is performed in less than 10 ms on standard hardware using RS-170 or CCIR-sized images. Thus, it facilitates the real-time recognition of objects even under severe projective distortions. The new method is not restricted to object recognition but could also be used in several other applications that require fast computations. Whenever more than one image must be rectified with the same mapping, a gain in computation time can be achieved by the proposed method.

Moreover, the method can be used to eliminate arbitrary distortions that are not necessarily caused by lens distortions or projective distortions. For example, distortions that are caused by non-planar object surfaces can be eliminated. Once the rectification map is built the image of the curved surface can be unwrapped into a plane in real-time. Consequently, the further processing needs to focus only on “planar” algorithms, and hence can be simplified significantly.

Several examples show that the proposed approach for recognizing compound objects fulfills the real-time requirement. The computation time varies between 20 ms and 51 ms in the presented examples, which corresponds to an improvement of up to 1400% in comparison to standard recognition methods.

To summarize, the approach is able to recognize compound objects, to perform the recognition in real-time, and to provide an inherent determination of the correspondence between object parts. Furthermore, it exhibits a very high degree of automation. The approach is general with regard to the type of object, and shows a very high robustness against occlusions, clutter, and changes in brightness. The pose parameters of all object parts are returned with high accuracy. Even objects under projective distortions can be recognized. Finally, several instances of the object in the image can be found simultaneously. There is no other approach available that demonstrates comparable features.

Nevertheless, the proposed approach shows room for some promising extensions that are worth mentioning. They are discussed in the following.

The first point concerns the accuracy of the pose parameters that are returned by the rigid object recognition approach. Although the accuracy is not precisely known, the approach for training the hierarchical model requires such accuracy information. For the current implementation, the accuracy values have been determined empirically and set to a constant value. Unfortunately, the accuracy depends on several factors that were not taken into account. For example, the accuracy depends on object characteristics like the object size, the number of model points, the object shape, the histogram of the gradient directions, or object symmetries. One way to make accuracy information available would be to perform several empirical tests while using objects with different characteristics, determining the achieved accuracy, and model the accuracy as a function of the characteristics. Other influences that are not correlated with object characteristics cannot be considered by this method. Another possibility would be to derive the accuracy based on the score values in parameter space. For example, the curvature of the local maximum could be an indicator of the achievable accuracy. Unfortunately, initial tests have shown that this is a very weak indicator.

A second possible extension would be to merge object parts that only show a small relative movement into one rigid object part on higher pyramid levels. This is because the relative movement decreases for higher pyramid levels with respect to the quantization of the parameter space. The resulting larger object parts would increase the robustness because, in general, larger objects can be found with higher reliability. Furthermore, it would enhance the efficiency because larger object parts allow the use of more pyramid levels.

Another improvement deals with the rectification. When using recognition approaches that perform a segmentation of the search image during the online phase (like the modified generalized Hough transform) a further speed-up could be achieved when dealing with image distortions: the rectification in the online phase can be restricted to the features, like edge position or edge orientation, in order to avoid the complete rectification of the entire image. This would, however, result in only a moderate speed-up of the entire recognition process since the contribution of the proposed rectification to the overall computation time is small. Unfortunately, the shape-based matching is not suited to take advantage of this improvement because no segmentation of the search image is performed, and hence no speed-up can be achieved.

Finally, in some applications it is desirable to recognize an object under more general transformations than rigid transformations. For example, if the distance between camera and object is variable, or if the object itself occurs in different sizes, an additional scaling must be considered, which results in similarity transformations. The recognition of rigid objects and compound objects can be extended in a straightforward way to take scaling into account. However, extending the automatic training of the hierarchical model turns out to be more complex. The rigidity assumption of object parts must be relaxed. Consequently, the rating of the matches during the solution of the ambiguities would have to be redesigned.

In conclusion, it should be pointed out that the principle of the proposed approach for recognizing compound 2D objects can be applied to the recognition of 3D objects from 2D images as well. There are different approaches for recognizing 3D rigid objects, e.g., (Procter and Illingworth 1997, Vosselman and Tangelder 2000, Pope and Lowe 2000, Blaskó and Fua 2001). By using one of these approaches, the training and the creation of the hierarchical model can be performed in the same way as in the 2D case. Consequently, also the recognition of compound 3D objects can benefit from the use of the proposed hierarchical recognition. The computation time is reduced considerably. This is especially important in 3D because, in general, more complex and time consuming algorithms are involved than in 2D. Furthermore, the correspondence problem, which is much more complicated in 3D, must be only solved once during the training but not during the recognition process itself. Thus, real-time applications in 3D, e.g., in the field of autonomous mobile systems (Lanser et al. 1997), augmented reality (Blaskó and Fua 2001), or 3D object tracking (Torre et al. 2000) can profit from the proposed approach.

Bibliography

Abdel-Aziz, Y. I. and Karara, H. M. (1971): Direct linear transformation into object space coordinates in close-range photogrammetry, *Symposium on Close-Range Photogrammetry*, 1–19.

Abe, K., Arcelli, C., Hisajima, T. and Ibaraki, T. (1996): Parts of planar shapes, *Pattern Recognition* 29(10): 1703–1711.

Anuta, P. E. (1970): Spatial registration of multispectral and multitemporal digital imagery using fast Fourier transform techniques, *IEEE Transactions on Geoscience Electronics* 8(4): 353–368.

Aschwanden, P. and Guggenbühl, W. (1992): Experimental results from a comparative study on correlation-type registration algorithms, *in* W. Förstner and S. Ruwiedel (editors), *Robust Computer Vision: Quality of Vision Algorithms*, Wichmann, Karlsruhe, 268–289.

Bacon, D. (1997): Image preprocessing to improve the reliability of normalized correlation, *in* A. R. Rao and N. Chang (editors), *Machine Vision Applications in Industrial Inspection V*, Proc. SPIE 3029, 195–199.

Ballard, D. H. (1981): Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition* 13(2): 111–122.

Ballard, D. H. and Brown, C. M. (1982): *Computer Vision*, Prentice-Hall International, Inc., Englewood Cliffs, NJ.

Ballard, D. H. and Sabbah, D. (1983): Viewer independent shape recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5(6): 653–659.

Barnea, D. I. and Silverman, H. F. (1972): A class of algorithms for fast digital image registration, *IEEE Transactions on Computers* 21: 179–186.

Baumgartner, A. (1998): Extraction of roads from aerial imagery based on grouping and local context, *International Archives of Photogrammetry and Remote Sensing*, volume XXXII part 3/1, 196–201.

Beardsley, P., Murray, D. and Zisserman, A. (1992): Camera calibration using multiple images, *in* G. Sandini (editor), *European Conference on Computer Vision*, volume 588 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 312–320.

Belongie, S., Malik, J. and Puzicha, J. (2002): Shape matching and object recognition using shape contexts, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(4): 509–522.

Berkhin, P. (2002): Survey of clustering data mining techniques, *Technical report*, Accrue Software, San Jose, CA.

Beyer, H. A. (1987): Some aspects of the geometric calibration of CCD-cameras, *ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, Interlaken, Switzerland, 68–81.

Beyer, H. A. (1992): Accurate calibration of CCD-cameras, *Computer Vision and Pattern Recognition*, 96–101.

Bhat, D. N. and Nayar, S. K. (1998): Ordinal measures for image correspondence, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(4): 415–423.

Blaskó, G. and Fua, P. (2001): Real-time 3D object recognition for automatic tracker initialization, *4th International Symposium on Augmented Reality*.

Bock, F. (1971): An algorithm to construct a minimum directed spanning tree in a directed network, *in* B. Avi-Itzhack (editor), *Developments in Operations Research*, Gordon and Breach, New York, NY, 29–44.

Borgefors, G. (1984): Distance transformation in arbitrary dimensions, *Computer Vision, Graphics, and Image Processing* 27: 321–345.

Borgefors, G. (1988): Hierarchical chamfer matching: A parametric edge matching algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10(6): 849–865.

Bronstein, I. N., Semendjajew, K. A., Musiol, G. and Mühlig, H. (2001): *Taschenbuch der Mathematik*, 5th edn, Harri Deutsch, Thun, Frankfurt am Main.

Brown, D. C. (1966): Decentering distortion of lenses, *Photogrammetric Engineering & Remote Sensing* 32(3): 444–462.

Brown, L. G. (1992): A survey of image registration techniques, *ACM Computing Surveys* 24(4): 325–376.

Bunke, H. (2000): Graph matching for visual object recognition, *Spatial Vision* 13(2/3): 335–340.

Canny, J. (1983): Finding edges and lines in images, *Technical Report 720*, MIT Artificial Intelligence Laboratory, Cambridge, MA.

Canny, J. (1986): A computational approach to edge detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8(6): 679–698.

Cantoni, V. and Carrioli, L. (1987): Structural shape recognition in a multiresolution environment, *Signal Processing* 12: 267–276.

Cha, S.-H. (2000): Efficient algorithms for image template and dictionary matching, *Journal of Mathematical Imaging and Vision* 12: 81–90.

Cha, S.-H. and Srihari, S. N. (2000): Distance between histograms of angular measurements and its application to handwritten character similarity, *15th International Conference on Pattern Recognition*, volume 2, 21–24.

Christmas, W. J., Kittler, J. and Petrou, M. (1995): Structural matching in computer vision using probabilistic relaxation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(8): 749–764.

Chu, Y. J. and Tseng-Hong, L. (1965): On the shortest arborescence of a directed graph, *Scientia Sinica* 14(10): 1396–1400.

Clark, J. and Holton, D. A. (1994): *Graphentheorie: Grundlagen und Anwendungen*, Spektrum Akademischer Verlag, Heidelberg.

Cognex (2000): *Cognex MVS-8000 Series - CVL Vision Tool Guide*, Cognex Corporation: CVL 5.5.1.

Cohen, S. D. and Guibas, L. J. (1997): Shape-based image retrieval using geometric hashing, *ARPA Image Understanding Workshop*.

Crouzil, A., Massip-Pailhes, L. and Castan, S. (1996): A new correlation criterion based on gradient fields similarity, *13th International Conference on Pattern Recognition*, volume 1, 632–691.

Danielsson, P. E. (1980): Euclidean distance mapping, *Computer Graphics and Image Processing* 14: 227–248.

Davis, L. S. (1982): Hierarchical generalized Hough transforms and line-segment based generalized Hough transforms, *Pattern Recognition* 15(4): 277–285.

Deriche, R. (1987): Using Canny's criteria to derive a recursively implemented optimal edge detector, *International Journal of Computer Vision* 1: 167–187.

Duta, N., Jain, A. K. and Dubuisson-Jolly, M.-P. (2001): Automatic construction of 2D shape models, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(5): 433–446.

Ebner, H. (1976): Self calibrating block adjustment, *International Archives of Photogrammetry* 21(3): 1–17.

Edmonds, J. (1967): Optimum branchings, *Journal of Research of the National Bureau of Standards* 71B(4): 233–240.

Edwards, J. L. and Murase, H. (1998): Coarse-to-fine adaptive masks for appearance matching of occluded scenes, *Machine Vision and Applications* 10: 232–242.

Elder, J. H. and Zucker, S. W. (1998): Local scale control for edge detection and blur estimation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(7): 699–716.

Elder, J. (1999): Are edges incomplete?, *International Journal of Computer Vision* 34(2): 97–122.

Faig, W. (1975): Calibration of close-range photogrammetric systems: Mathematical formulation, *Photogrammetric Engineering & Remote Sensing* 41(12): 1479–1486.

Faugeras, O. D. and Toscani, G. (1986): Calibration problems for stereo, *Computer Vision and Pattern Recognition*, 15–20.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2000): Efficient matching of pictorial structures, *Computer Vision and Pattern Recognition*, volume 2, 66–73.

Fischetti, M. and Toth, P. (1993): An efficient algorithm for the min-sum arborescence problem on complete digraphs, *ORSA Journal of Computing* 5(4): 426–434.

Fitsch, A. J., Kadyrov, A., Christmas, W. J. and Kittler, J. (2002): Orientation correlation, *British Machine Vision Conference*, volume 1, 133–142.

Foroosh, H., Zerubia, J. B. and Berthod, M. (2002): Extension of phase correlation to subpixel registration, *IEEE Transactions on Image Processing* 11(3): 188–200.

Förstner, W. and Gülch, E. (1987): A fast operator for detection and precise location of distinct points, corners and centres of circular features, *ISPRS Intercommission Workshop*, Interlaken, Switzerland, 281–305.

Gabow, H. N., Galil, Z., Spencer, T. and Tarjan, R. E. (1986): Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, *Combinatorica* 6(2): 109–122.

Garfinkel, R. S. and Nemhauser, G. L. (1972): *Integer Programming*, John Wiley & Sons, Inc., New York, NY.

Gavrila, D. M. and Giebel, J. (2001): Virtual sample generation for template-based shape matching, *Computer Vision and Pattern Recognition*, 676–681.

Gavrila, D. M. and Philomin, V. (1999): Real-time object detection for "smart" vehicles, *7th International Conference on Computer Vision*, volume I, 87–93.

Gideon, R. A. and Hollister, R. A. (1987): A rank correlation coefficient, *Journal of the American Statistical Association* 82(398): 656–666.

Godding, R. (1993): Ein photogrammetrisches Verfahren zur Überprüfung und Kalibrierung digitaler Bildaufnahmesysteme, *Zeitschrift für Photogrammetrie und Fernerkundung* 2: 82–90.

Gonzalez, R. C. and Woods, R. E. (1992): *Digital Image Processing*, Addison-Wesley Publishing Company.

Gotthardt, E. (1968): *Einführung in die Ausgleichungsrechnung*, Wichmann, Karlsruhe.

Graham, R. L. and Hell, P. (1985): On the history of the minimum spanning tree problem, *Annals of the History of Computing* 7(1): 43–57.

Grimson, W. E. L. (1987): Recognition of object families using parameterized models, *International Conference on Computer Vision*, 93–101.

Grimson, W. E. L. (1989): On the recognition of parametrized 2-D objects, *International Journal of Computer Vision* 2(4): 353–372.

Gruen, A. W. and Beyer, H. A. (1987): Real-time photogrammetry at the digital photogrammetric station (DIPS) of ETH Zürich, *The Canadian Surveyor* 41(2): 181–199.

Haberäcker, P. (1995): *Praxis der Digitalen Bildverarbeitung und Mustererkennung*, Carl Hanser Verlag München Wien.

Haralick, R. M. and Shapiro, L. G. (1992): *Computer and Robot Vision*, volume I, Addison-Wesley Publishing Company, Reading, MA.

Hartley, R. I. and Zisserman, A. (2000): *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge.

Hauck, A., Lanser, S. and Zierl, C. (1997): Hierarchical recognition of articulated objects from single perspective views, *Computer Vision and Pattern Recognition*, 870–876.

Heipke, C. (1995): *Digitale photogrammetrische Auswertestationen*, Habilitationsschrift, Fakultät für Bauingenieur- und Vermessungswesen, Technische Universität München: Deutsche Geodätische Kommision, Reihe C, Heft Nr. 450, München.

Heipke, C., Stephani, M., Strunz, G. and Lenz, R. (1991): Accuracy potential of a digital CCD camera for photogrammetric applications, *in* B. Radig (editor), *Mustererkennung*, Informatik Fachberichte 290, Springer-Verlag, Berlin, 320–327.

Hel-Or, Y. and Werman, M. (1994a): Constraint-fusion for interpretation of articulated objects, *Computer Vision and Pattern Recognition*, 39–45.

Hel-Or, Y. and Werman, M. (1994b): Model based pose estimation of articulated and constraint objects, *Third European Conference on Computer Vision*, 199–204.

Hoffman, D. and Richards, W. (1984): Parts of recognition, *Cognition* 18: 65–96.

Hough, P. V. C. (1962): Method and means for recognizing complex patterns, *U.S. Patent 3,069,654*.

Huertas, A., Cole, W. and Nevatia, R. (1990): Detecting runways in complex airport scenes, *Computer Vision, Graphics, and Image Processing* 51: 107–145.

Huttenlocher, D. P., Klanderman, G. A. and Rucklidge, W. J. (1993): Comparing images using the Hausdorff distance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(9): 850–863.

Huttenlocher, D. P., Lilien, R. H. and Olson, C. F. (1999): View-based recognition using an eigenspace approximation to the Hausdorff measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(9): 951–955.

Illingworth, J. and Kittler, J. (1988): A survey of the Hough transform, *Computer Vision, Graphics, and Image Processing* 44: 87–116.

Irani, M. and Anandan, P. (1999): All about direct methods, *in* W. Triggs, A. Zisserman and R. Szeliski (editors), *International Workshop on Vision Algorithms*, 267–277.

Jacobs, C. E., Finkelstein, A. and Salesin, D. H. (1995): Fast multiresolution image querying, *Proc. SIGGRAPH '95*, 277–286.

Jähne, B. (2002): *Digital Image Processing*, 5th edn, Springer-Verlag.

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999): Data clustering: A review, *ACM Computing Surveys* 31(3): 264–323.

Jain, A. K., Zhong, Y. and Lakshmanan, S. (1996): Object matching using deformable templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(3): 267–278.

Jeng, S.-C. and Tsai, W.-H. (1990): Fast generalized Hough transform, *Pattern Recognition Letters* 11: 725–733.

Jensen, D. (2002): Real-time for the real world: URL accessed October 2002.
*http://www.real-time.org/

Kager, H. (1981): *Bündeltriangulation mit indirekt beobachteten Kreiszentren*, Dissertation, Institut für Photogrammetrie, Technische Universität Wien: Geowissenschaftliche Mitteilungen, Heft Nr. 19, Wien.

Kaneko, S., Murase, I. and Igarashi, S. (2002): Robust image registration by increment sign correlation, *Pattern Recognition* 35(10): 2223–2234.

Karmarkar, N. (1984): A new polynomial-time algorithm for linear programming, *Combinatorica* 4(4): 373–395.

Kassim, A. A., Tan, T. and Tan, K. H. (1999): A comparative study of efficient generalised Hough transform techniques, *Image and Vision Computing* 17: 737–748.

Koch, K. R. (1987): *Parameterschätzung und Hypothesentests in linearen Modellen*, Dümmler, Bonn.

Koffka, K. (1935): *Principles of Gestalt Psychology*, Harcourt Brace, New York.

Kratchounova, T., Krebs, B. and Korn, B. (1996): Erkennung und Bestimmung der aktuellen Konstellation von Objekten mit Scharniergelenken, *in* B. Jähne, P. Geißler, H. Haußecker and F. Hering (editors), *Mustererkennung*, Informatik aktuell, Springer-Verlag, Berlin, 502–509.

Kroupnova, N. H. and Korsten, M. J. (1997): Object recognition algorithm based on inexact graphs matching and its application in a color vision system for recognition of electronic components on PCB's, *in* A. R. Rao and N. Chang (editors), *Machine Vision Applications in Industrial Inspection V*, Proc. SPIE 3029, 37–48.

Kuglin, C. and Hines, D. (1975): The phase correlation image alignment method, *International Conference of Cybernetics and Society*, 163–165.

Kupfer, G. (1987): Volle geometrische Systemkalibrierung metrischer Luftbildkammern - Das Testfeld Brecherspitze, *Bildmessung und Luftbildwesen* 55: 151–154.

Kwon, O.-K., Sim, D.-G. and Park, R.-H. (2001): Robust Hausdorff distance matching algorithms using pyramidal structures, *Pattern Recognition* 34(10): 2005–2013.

Lai, S.-H. and Fang, M. (1999a): Accurate and fast pattern localization algorithm for automated visual inspection, *Real-Time Imaging* 5: 3–14.

Lai, S.-H. and Fang, M. (1999b): A FLASH system for fast and accurate pattern localization, *in* K. W. Tobin and N. S. Chang (editors), *Machine Vision Applications in Industrial Inspection VII*, Proc. SPIE 3652, 164–173.

Lai, S.-H. and Fang, M. (1999c): Robust and efficient image alignment with spatially varying illumination models, *Computer Vision and Pattern Recognition*, volume II, 167–172.

Lanser, S. and Eckstein, W. (1992): A modification of Deriche's approach to edge detection, *11th International Conference on Pattern Recognition*, volume III, 633–637.

Lanser, S., Zierl, C. and Beutlhauser, R. (1995): Multibildkalibrierung einer CCD-Kamera, *in* G. Sagerer, S. Posch and F. Kummert (editors), *Mustererkennung*, Informatik aktuell, Springer-Verlag, Berlin, 481–491.

Lanser, S., Zierl, C., Munkelt, O. and Radig, B. (1997): MORAL — a vision-based object recogntion system for autonomous mobile systems, *Computer Analysis of Images and Patterns '97*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 33–41.

Latecki, L. J. and Lakämper, R. (1999): Convexity rule for shape decomposition based on discrete contour evolution, *Computer Vision and Image Understanding* 73(3): 441–454.

Leavers, V. F. (1993): Which Hough transform?, *Computer Vision, Graphics, and Image Processing: Image Understanding* 58(2): 250–264.

Lee, K.-M. and Street, W. N. (2000): Generalized Hough transforms with flexible templates, *International Conference on Artificial Intelligence*, 1133–1139.

Lenz, R. (1987): Linsenfehlerkorrigierte Eichung von Halbleiterkameras mit Standardobjektiven für hochgenaue 3D-Messungen in Echtzeit, *in* E. Paulus (editor), *Mustererkennung*, Informatik Fachberichte 149, Springer-Verlag, Berlin, 212–216.

Levenberg, K. (1944): A method for the solution of certain non-linear problems in least squares, *Quarterly Journal of Applied Mathematics* 2(2): 164–168.

Lewis, J. P. (1995): Fast template matching, *Vision Interface*, 120–123.

Li, B. C., Zhao, D., Vilallabos, R. and Cabrera, S. (1997): Robust statistic-based template matching, *in* A. R. Rao and N. Chang (editors), *Machine Vision Applications in Industrial Inspection V*, Proc. SPIE 3029, 195–199.

Li, S. Z. (1999): Shape matching based on invariants, *in* O. Omidvar (editor), *Shape Analysis*, volume 6 of *Progress in Neural Networks*, Ablex, 203–228.

Li, W.-J. and Lee, T. (2002): Object recognition and articulated object learning by accumulative Hopfield matching, *Pattern Recognition* 35(9): 1933–1948.

Liao, S. X. and Pawlak, M. (1996): On image analysis by moments, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(3): 254–266.

Lin, C., Huertas, A. and Nevatia, R. (1994): Detection of buildings using perceptual grouping and shadows, *Computer Vision and Pattern Recognition*, 62–69.

Lindeberg, T. (1994): *Scale-Space Theory in Computer Vision*, Kluwer Academic Publishers, Dordrecht.

Lo, R.-C. and Tsai, W.-H. (1997): Perspective-transformation-invariant generalized Hough transform for perspective planar shape detection and matching, *Pattern Recognition* 30(3): 383–396.

Lowe, D. G. (1985): *Perceptual Organization and Visual Recognition*, Kluwer Academics, Boston.

Lowe, D. G. (1991): Fitting parametrized 3-D models to images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(5): 441–450.

Ma, D. and Chen, X. (1988): Hough transform using slope and curvature as local properties to detect arbitrary 2D shapes, *9th International Conference on Pattern Recognition*, 511–513.

Marquardt, D. W. (1963): An algorithm for least-squares estimation of nonlinear parameters, *SIAM Journal of Applied Mathematics* 11(2): 431–441.

Marr, D. (1982): *Vision*, W. H. Freeman and Company, San Francisco, CA.

Martin, J. and Crowley, J. (1995): Experimental comparison of correlation techniques, *IAS-4, International Conference on Intelligent Autonomous Systems*, Karlsruhe.

Matrox (2001): *Matrox Imaging Library - User Guide*, Matrox Electonic Systems Ltd: Version 6.1.

Matrox (2002): *Matrox Imaging Library - User Guide*, Matrox Electonic Systems Ltd: Version 7.0.

Mayer, H. (1998): *Automatische Objektextraktion aus digitalen Luftbildern*, Habilitationsschrift, Fakultät für Bauingenieur- und Vermessungswesen, Technische Universität München: Deutsche Geodätische Kommision, Reihe C, Heft Nr. 494, München.

Messmer, B. T. (1996): *Efficient Graph Matching Algorithms for Preprocessed Model Graphs*, PhD thesis, University of Bern, Switzerland.

Messmer, B. T. and Bunke, H. (1998): A new algorithm for error-tolerant subgraph isomorphism detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(5): 493–504.

Mikhail, E. M., Bethel, J. S. and McGlone, J. C. (2001): *Introduction to Modern Photogrammetry*, John Wiley & Sons, Inc.

Mokhtarian, F., Abbasi, S. and Kittler, J. (1996): Efficient and robust retrieval by shape content through curvature scale space, *International Workshop on Image Databases and Multi-Media Search*, Amsterdam, 35–42.

MVTec (2002): *HALCON Version 6.1 - Programmer's Manual*, MVTec Software GmbH.

Negahdaripour, S. (1998): Revised definition of optical flow: integration of radiometric and geometric cues for dynamic scene analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(9): 961–979.

Olson, C. F. and Huttenlocher, D. P. (1995): Recognition by matching dense, oriented edge pixels, *International Symposium on Computer Vision*, 91–96.

Olson, C. F. and Huttenlocher, D. P. (1996): Recognition by matching with edge location and orientation, *Image Understanding Workshop*, 1167–1173.

Olson, C. F. and Huttenlocher, D. P. (1997): Automatic target recognition by matching oriented edge pixels, *IEEE Transactions on Image Processing* 6(1): 103–113.

Paumard, J. (1997): Robust comparison of binary images, *Pattern Recognition Letters* 18(10): 1057–1063.

Pilu, M. and Fisher, R. B. (1997): Model-driven grouping and recognition of generic object parts from single images, *Journal of Robotics and Autonomous Systems* 21: 107–122.

Pope, A. R. and Lowe, D. G. (2000): Probabilistic models of appearance for 3-D object recognition, *International Journal of Computer Vision* 40(2): 149–167.

Pratt, W. K. (2001): *Digitial Image Processing*, 3rd edn, John Wiley & Sons, Inc., New York, NY.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992): *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edn, Cambridge University Press, Cambridge.

Procter, S. and Illingworth, J. (1997): Foresight: Fast object recognition using geometric hashing with edge-triple features, *International Conference on Image Processing*, volume 1, Washington, D.C., 889–892.

Radcliffe, T., Rajapakshe, R. and Shalev, S. (1993): Pseudo-correlation: A fast, robust, absolute grey level image alignment algorithm, *in* M. H. Loew (editor), *Medical Imaging 1993: Image Processing*, Proc. SPIE 1898, 134–145.

Rock, I. and Palmer, S. (1990): The legacy of Gestalt psychology, *Scientific American* 263: 84–90.

Rosin, P. L. (2000): Shape partitioning by convexity, *IEEE Transactions on Systems, Man, and Cybernetics* 30(2): 202–210.

Rucklidge, W. J. (1997): Efficiently locating objects using the Hausdorff distance, *International Journal of Computer Vision* 24(3): 251–270.

Russ, J. C. (2000): *The Image Processing Handbook*, 3rd edn, Springer-Verlag, Heidelberg.

Scharstein, D. (1994): Matching images by comparing their gradient fields, *12th International Conference on Pattern Recognition*, volume 1, 572–575.

Schmid, C., Mohr, R. and Bauckhage, C. (2000): Evaluation of interest point detectors, *International Journal of Computer Vision* 37(2): 151–172.

Sclaroff, S. and Liu, L. (2001): Deformable shape detection and description via model-based region grouping, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(5): 475–489.

SearchSolaris.com (2002): Definitions - real time: URL accessed October 2002.
*http://searchsolaris.techtarget.com/sDefinition/0,,sid12_gci214344,00.html

Shapiro, L. G. and Haralick, R. M. (1979): Decomposition of two-dimensional shapes by graph-theoretic clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1(1): 10–20.

Siddiqi, K. and Kimia, B. B. (1995): Parts of visual form: Computational aspects, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(3): 239–251.

Sim, D.-G. and Park, R.-H. (2001): Two-dimensional object alignment based on the robust oriented Hausdorff similarity measure, *IEEE Transactions on Image Processing* 10(3): 475–483.

Singh, M., Seyranian, G. D. and Hoffman, D. D. (1999): Parsing silhouettes: the short-cut rule, *Perception & Psychophysics* 61: 636–660.

Soille, P. (1999): *Morphological Image Analysis, Principles and Applications*, Springer-Verlag, Heidelberg.

Steger, C. (1998): *Unbiased Extraction of Curvilinear Structures from 2D and 3D Images*, Dissertation, Fakultät für Informatik, Technische Universität München: Herbert Utz Verlag, München.

Steger, C. (2000): Subpixel-precise extraction of lines and edges, *International Archives of Photogrammetry and Remote Sensing*, volume XXXIII, part B3, 141–156.

Steger, C. (2001): Similarity measures for occlusion, clutter, and illumination invariant object recognition, *in* B. Radig and S. Florczyk (editors), *Mustererkennung 2001*, Springer, München, 148–154.

Steger, C. (2002): Occlusion, clutter, and illumination invariant object recognition, *in* R. Kalliany and F. Leberl (editors), *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, volume XXXIV, part 3A, Graz, 345–350.

Steger, C., Mayer, H. and Radig, B. (1997): The role of grouping for road extraction, *in* A. Gruen, E. Baltsavias and O. Henricsson (editors), *Automatic Extraction of Man-Made Objects from Aerial and Space Images (II)*, Birkhäuser Verlag, Basel, 245–256.

Stöcker, H. (1995): *Taschenbuch Mathematischer Formeln und moderner Verfahren*, 3rd edn, Harri Deutsch, Thun, Frankfurt am Main.

Suganthan, P. N. (2002): Structural pattern recognition using genetic algorithms, *Pattern Recognition* 35(9): 1883–1893.

Szeliski, R. and Coughlan, J. (1997): Spline-based image registration, *International Journal of Computer Vision* 22(3): 199–218.

Tanimoto, S. L. (1981): Template matching in pyramids, *Computer Graphics and Image Processing* 16: 356–369.

Tarjan, R. E. (1977): Finding optimum branchings, *Networks* 7: 25–35.

Teh, C.-H. and Chin, R. T. (1988): On image analysis by methods of moments, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10(4): 496–513.

Thomas, A. D. H. (1992): Compressing the parameter space of the generalized Hough transform, *Pattern Recognition Letters* 13(2): 107–112.

Tian, Q. and Huhns, M. N. (1986): Algorithms for subpixel registration, *Computer Vision, Graphics, and Image Processing* 35: 220–223.

Tobin, K. W., Karnowski, T. P. and Ferrell, R. K. (1999): Image retrieval in the industrial environment, *IP&T/SPIE's 11th International Symposium on Electronic Imaging: Science and Technology*, Proc. SPIE 3652, San Jose Convention Center, 508–518.

Torre, R., Balcisoy, S., Fua, P. and Thalmann, D. (2000): Interaction between real and virtual humans: Playing checkers, *Eurographics Workshop on Virtual Environments*.

Tsai, R. Y. and Lenz, R. K. (1988): Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10(5): 713–720.

Uenohara, M. and Kanade, T. (1997): Use of Fourier and Karhunen-Loeve decomposition for fast pattern matching with a large set of templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(8): 891–898.

Ullman, S. (1979): *The Interpretation of Visual Motion*, MIT Press, Cambridge, MA.

Ullman, S. (1989): Aligning pictorial descriptions: An approach to object recognition, *Cognition* 32: 193–254.

Ullmann, J. R. (1976): An algorithm for subgraph isomorphism, *Journal of Association for Computing Machinery* 23(1): 31–42.

Ulrich, M. and Steger, C. (2001): Empirical performance evaluation of object recognition methods, *in* H. I. Christensen and P. J. Phillips (editors), *Empirical Evaluation Methods in Computer Vision*, IEEE Computer Society Press, Los Alamitos, CA, 62–76.

Ulrich, M. and Steger, C. (2002): Performance evaluation of 2D object recognition techiques, *Technical Report PF-2002-01*, Lehrstuhl für Photogrammetrie und Fernerkundung, Technische Universität München.

Ulrich, M., Baumgartner, A. and Steger, C. (2002): Automatic hierarchical object decomposition for object recognition, *International Archives of Photogrammetry and Remote Sensing*, volume XXXIV, part 5, 99–104.

Ulrich, M., Steger, C., Baumgartner, A. and Ebner, H. (2001a): Real-time object recognition in digital images for industrial applications, *5th Conference on Optical 3-D Measurement Techniques*, Vienna, 308–318.

Ulrich, M., Steger, C., Baumgartner, A. and Ebner, H. (2001b): Real-time object recognition using a modified generalized Hough transform, *21. Wissenschaftlich-Technische Jahrestagung der DGPF*, volume 10, Konstanz, 571–578.

Veltkamp, R. C. and Hagedorn, M. (1999): State-of-the-art in shape matching, *Technical Report UU–CS–1999–27*, Department of Computing Science, Utrecht University, Utrecht, The Netherlands.

Vosselman, G. and Tangelder, J. W. H. (2000): 3D reconstruction of industrial installations by constrained fitting of CAD models to images, *in* G. Sommer, N. Krüger and C. Perwass (editors), *Mustererkennung*, Informatik aktuell, Springer-Verlag, Berlin, 285–292.

Wallack, A. and Manocha, D. (1998): Robust algorithms for object localization, *International Journal of Computer Vision* 27(3): 243–262.

Wang, J. Z. (2001): Wavelets and imaging informatics: A review of the literature, 34(2): 129–141.

Wang, J. Z., Wiederhold, G., Firschein, O. and Wei, S. X. (1995): Wavelet-based image indexing techniques with partial sketch retrieval capability, *Proceedings of the Fourth Forum on Research and Technology Advances in Digital Libraries*, 13–24.

Weng, J., Cohen, P. and Herniou, M. (1992): Camera calibration with distortion models and accuracy evaluation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(10): 965–980.

Wertheimer, M. (1938): Laws of organization in perceptual forms, *in* W. Ellis (editor), *A source book of Gestalt psychology*, Routledge & Kegan Paul, 71–88.

Wester-Ebbinghaus, W. (1983): *Einzelstandpunkt-Selbstkalibrierung — ein Beitrag zur Feldkalibrierung von Aufnahmekammern*, Habilitationsschrift, Hohe Landwirtschaftliche Fakultät, Rheinische Friedrich-Wilhelms-Universität: Deutsche Geodätische Kommision, Reihe C, Heft Nr. 289, München.

Wiedemann, C. and Hinz, S. (1999): Automatic extraction and evaluation of road networks, *International Archives of Photogrammetry and Remote Sensing*, volume 32, part 3–2W5, 95–100.

Witkin, A. and Tenenbaum, J. M. (1983): On the role of structure in vision, *in* A. Rosenfeld (editor), *Human and Machine Vision*, Academic Press, Inc.

Wolfson, H. J. (1990): Model-based object recognition by geometric hashing, *in* O. D. Faugeras (editor), *European Conference on Computer Vision*, volume 427 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 526–536.

Wong, K. W. (1975): Mathematical formulation and digital analysis in close-range photogrammetry, *Photogrammetric Engineering & Remote Sensing* 41(11): 1355–1373.

Wuescher, D. M. and Boyer, K. L. (1991): Robust contour decomposition using a constant curvature criterion, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(1): 41–51.

Acknowledgements

First of all, I would like to cordially thank my thesis advisor, Prof. Dr.-Ing. Heinrich Ebner, head of the Chair for Photogrammetry and Remote Sensing at the Technische Universität München, for encouraging me and giving me the opportunity to work on this dissertation. I also appreciate his assistance during valuable discussions, his ability to impart knowledge about working scientifically, and his support for keeping me on schedule during the preparation of the dissertation.

Furthermore, I would like to express my thanks to Prof. Dr.-Ing. Thomas Alexander Wunderlich, head of the Chair for Geodesy at the Technische Universität München, for being the second reporter on this dissertation. He also significantly improved my work by giving helpful comments.

I owe a great debt to the management of the MVTec Software GmbH, Dr. rer. nat. Wolfgang Eckstein, Dr. rer. nat. Olaf Munkelt, and Dr. rer. nat. Carsten Steger, who provided financial and technical support for this dissertation and gave me the creative as well as the temporal freedom for my scientific work. I am very grateful to Dr. rer. nat. Wolfgang Eckstein and most notably to Dr. rer. nat. Carsten Steger for their stimulating discussions and the exchange of ideas. They also took the time to proofread this dissertation, even though their schedules were extremely tight. Furthermore, I extend many thanks to all my colleagues at the MVTec Software GmbH for the extremely inspiring and friendly atmosphere. I would especially like to thank Thorsten Bandlow, Dr. rer. nat. Gerhard Blahusch, Dr.-Ing. Alexa Zierl, and Christoph Zierl, who were always prepared to assist me in various ways.

Another substantial contribution to the successful preparation of this work can be attributed to the creative, motivating, and warm atmosphere at the Chair for Photogrammetry and Remote Sensing. I am very indebted in particular to Dr.-Ing. Albert Baumgartner, who always found the time for helpful discussions, and hence is responsible for the success of the work to a great extent. He also helped me greatly by proofreading this dissertation. Furthermore, I want to express my thanks to my other colleagues Konrad Eder, Christine Elmauer, Stefan Hinz, Heinz Kozlowski, Franz Kurz, Franz Meyer, Hermann Rentsch, Michael Spiegel, Dr.-Ing. Manfred Stephani, Dr.-Ing. Christian Wiedemann, and Birgit Wessel who contributed to this work by discussions, ideas, or by simply cheering me up. I am especially grateful to Franz Meyer, who also proofread this dissertation.

Finally, I want to express my sincere gratitude to Dorith for her patience and support and for being simply wonderful. Last, but not least, I am also very grateful to my parents for laying the foundation of my scientific education by arousing my interest on geodetic engineering, and by supporting my studies. Their love and constant encouragement have been more important than anything else.

Biography

Personal Data

Name:	Markus Eric Ulrich
Date of Birth:	16th December 1974
Place of Birth:	Kronach, Germany
Nationality:	German

Education

Sept. 1981 – Jul. 1985	Volksschule Weißenbrunn, Germany
Sept. 1985 – Jun. 1994	Frankenwald Gymnasium Kronach, Germany
Jun. 1994	Graduated with Abitur

Basic Military Service

Jul. 1994 – Jun. 1995	Panzerartilleriebataillon 395, Erfurt, Germany

Studies

Nov. 1995 – Mar. 2000	Technische Universität München, Germany: Vermessungswesen (Geodesy and Geoinformation)
Nov. 1999 – Mar. 2000	Diploma thesis at the Deutsches Geodätisches Forschungsinstitut, Munich, Germany: Vorhersage der Erdrotationsparameter mit Hilfe Neuronaler Netze (Prediction of the Earth Orientation Parameters by Neural Networks)
Mar. 2000	Graduated with a Dipl.-Ing. degree (M.Sc.) with honours

Professional Experience

Jun. 2000 – present	Scientific collaborator at the Chair for Photogrammetry and Remote Sensing of the Technische Universität München Software engineer at MVTec Software GmbH, Munich, Germany

Award

2000	Harbert-Buchspende Deutscher Verein für Vermessungswesen