Doctoral Thesis

Contributions to a 3-D robot vision system silhouette analysis and evidential reasoning

Author(s):
Peter, Martin

Publication Date:
1993

Permanent Link:
https://doi.org/10.3929/ethz-a-000897459

Rights / License:
In Copyright - Non-Commercial Use Permitted
Contributions to a 3-D Robot Vision System: Silhouette Analysis and Evidential Reasoning

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Technical Sciences

presented by
MARTIN PETER
Dipl. Inf. Ing. ETH Zürich
born February 27, 1957
citizen of Thalwil Kanton Zürich

accepted on the recommendation of
Prof. Dr. O. Kübler, examiner
Prof. Dr. H. Bunke, co-examiner

1993
I would first like to thank Dr. Frank Ade, who not only was my supervisor for this thesis and the manager of the COR vision system project, but who also very actively participated in the project. Without his sound knowledge of computer vision and his skills in formulating English texts, this project and this thesis would never have been successful.

I am greatly indebted to Prof. Dr. Olaf Kübler for providing the stimulating research atmosphere at the computer vision lab of the “Institut für Kommunikationstechnik”, for accepting this thesis and for reading it carefully.

I am grateful to Prof. Dr. Horst Bunke for accepting to co-examine the thesis and for reading it.

Special thanks are due to the students William Cabrera, Walter Hohl, Roland Levrand and Patrick Meyer who contributed with their semester term projects to this thesis, to my working colleagues Marjan Trobina and Antti Ylä-Jääsky who worked on other aspects of the COR vision system and above all to Martin Rutishauser, who did a great job in implementing and organizing a great deal of the whole software.
Leer - Vide - Empty
Contents

Acknowledgements ii

Contents iii

Abstract ix

Zusammenfassung xi

1 Introduction 1

1.1 Organization of the thesis 3

2 Survey of related work 5

2.1 Flynn and Jain 5

2.2 Dickinson and Pentland 7

2.3 Fan and Medioni 8

2.4 Jain and Hoffman 9

2.5 Kuono and Okamoto 10

2.6 Summary 11
3  Segmentation of contours  

3.1 Extracting contours ........................................ 13

3.2 Detecting significant points and segments in digital curves ........ 15
   3.2.1 Finding significant points in digital curves .......... 16
   3.2.2 A hybrid approach to detect significant points in digital curves ........................................ 17
   3.2.3 Seed corner point detection .................................. 18
   3.2.4 Algorithms for computing \( \theta(s) \) and curvature ...... 19
   3.2.5 Delimiting separator domains ................................ 23
   3.2.6 Approximation of segments .................................. 23
   3.2.7 Fit primitives for approximation ........................... 26
   3.2.8 Elimination of oversegmentation ............................ 30

3.3 Relations between segments .................................... 32
   3.3.1 Results ............................................................ 36
   3.3.2 Conclusion .......................................................... 37

3.4 Symmetries in contours ........................................ 38
   3.4.1 Methods working on coordinate points ................. 38
   3.4.2 Methods working on \( \theta(s) \) .............................. 40
   3.4.3 Structural matching for symmetry detection ........... 43
   3.4.4 Conclusion of symmetry axis computation ............... 48

4  Object recognition ................................................. 51

4.1 Recognition based on graph matching .......................... 52
   4.1.1 Model representation ........................................ 52
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  Additional results relative to segmentation</td>
<td>111</td>
</tr>
<tr>
<td>B  Sequence of a clearing of a tray</td>
<td>117</td>
</tr>
<tr>
<td>C  Correspondence table for names in feature base files</td>
<td>125</td>
</tr>
<tr>
<td>D  Simple example rulebase</td>
<td>127</td>
</tr>
<tr>
<td>E  Complete rulebase for the COR vision system</td>
<td>129</td>
</tr>
<tr>
<td>List of Figures</td>
<td>133</td>
</tr>
<tr>
<td>List of Tables</td>
<td>137</td>
</tr>
<tr>
<td>Curriculum Vitae</td>
<td>143</td>
</tr>
</tbody>
</table>
Abstract

This contribution originates in the framework of the interdisciplinary, application oriented project “Cooperating Robot” (COR). It contributes to the realization of a robot vision system for the task defined in the project: to build a robot capable of recognizing and clearing objects placed on a tray in a cafeteria. It addresses problems of acquiring 3-D information of transparent and specularly reflecting objects, robust and fast multi-knowledge source recognition scheme and precise object position determination.

Silhouettes as data source are used not only because of the transparent and specularly reflecting materials the objects are made of, but also because of the desired speed of operation of the system. It is demonstrated, how a gray level image of a silhouette can be segmented into a list of approximating contour segments. With the help of relations between these segments, a graph structure is obtained as symbolic representation of a silhouette.

Subsequently the method of graph matching for object recognition well known for range data is presented and applied to graphs obtained from silhouettes. This leads to an implementation of a small recognition system capable of recognizing objects in a scene composed of multiple objects from a known set of 11 objects like cup, glass, plate etc. Ambiguities in segmentation but also the small structural difference between some of the graphs of the objects from the dish set diminish the usability of this method. Therefore, the entirely different method of evidence accumulation for object recognition is introduced. It is based on a completely unbiased use of features like an observed height or a found diameter of a circle. For each feature an independent rule is formulated that states how much the given feature votes for or against a certain object hypothesis. Each object is represented in this rule base by a vector of evidence weights. For recognition, all rules are applied to the features detected
in the scene. This results in a scene vector with each element containing the evaluation result of the corresponding rule. A similarity measure between the scene vector and all object vectors is then computed to identify the object in the scene. This scheme is very efficient and well suited to integrate features from multiple sources. Specially for a 3-D vision system this is of importance since there is no way to compute a 3-D position of an object just from one view. For that reason the COR vision system is built around two silhouettes and one view from above. It is shown, how features from these three views can be integrated to an evidence accumulations scheme and how identity and position of objects can be computed.

The complete system with robot and vision system has been realized and its functioning has been proved. Results of the recognition process obtained during the operational phase of the system are presented.
Zusammenfassung


Anschließend wird die von Tiefendaten her bekannte Methode des Graph matching zur Erkennung von Objekten vorgestellt und auf Graphen, die aus Silhouetten erhalten werden, angewendet. Die Implementierung eines einfachen Erkennungssystems für 11 Objekte wie Teller, Tasse, Trinkglas etc. wird beschrieben. Mehrdeutigkeiten der Segmentierung, aber auch die geringen Strukturunterschiede zwischen manchen der Graphen, die von Objekten der Mensatabellen erhalten wurden, schränken die Leistungsfähigkeit dieser Methode jedoch stark ein. Als eine grundsätzlich verschiedene Alternative wird daher das Verfahren der Evidenzakkumulation zur Objekterkennung eingeführt. Es beruht auf einem völlig gleichartigen Gebrauch der Merkmale,

1

Introduction

The work presented in this thesis belongs to the general field of Computer Vision. In particular it focuses on model based object recognition for industrial applications. Whereas recognition of 2-D objects can be considered a mature discipline and applications in industrial environments are already in use, this is not the case for the recognition of 3-D objects. The difficulty stems from the fact that generally only a small part of a 3-D object is seen from any viewpoint. For that reason various sensing techniques are under development which use multiple views instead of one. Range data sensors that do a full 360° scan for example exist. But the problem of incomplete data comes up again when several mutually occluding objects are observed. Range sensors also use visible light for sensing the objects surfaces and this causes problems when the surfaces are specularly reflecting or transparent. This thesis therefore addresses sensing techniques for 3-D objects and in more depth the handling of incomplete geometric information. All work was done with the goal of building a functioning prototype of a complete (sensing to grasping) robot vision system for a concrete real world application. The framework for this work was given by the project “Cooperating Robot with visual and tactile capabilities, COR”\textsuperscript{1}. It defined the concrete and challenging task of building a complete system for clearing cafeteria trays. The vision system was to analyze the constellation of dishes and cutlery pieces on a tray to such an extent that it could communicate to the robot the identity of a piece and the information

\textsuperscript{1} Kooperierender Roboter mit visuellen und taktilen Fähigkeiten
necessary to correctly grasp it and to store it away to a predefined location. When a sequence of these operations had cleared a tray, it was moved away, and the next tray to be cleared would move into position. The definition of that project implied many demands on the system. Among the ones that affected the work presented here we find:

- The vision system must be capable of recognizing specularly reflecting objects like metallic cutlery pieces and transparent objects like drinking glasses.

- The complete system should operate at a speed comparable to the speed of a human doing the same work, who removes about 1 piece/second.

As consequence of the first requirement, range sensors are practically not usable for the given task. Even if ultraviolet light instead of visible light had been used, glass would have been seen as a black surface which would make it hard to detect the projected light. Besides, range data often require time-consuming 3-D geometric operations, which makes their use difficult for a system operating at high speed. To fulfill both demands, it was found that silhouettes of cafeteria trays in front of an illuminated background have good contrast, good spatial definition and contain enough information for a recognition task. Through the simplicity of these images, fast processing is possible. The silhouette image is segmented into a contour string of successive silhouette points with the help of an edge detector and a contour tracer. With the help of a newly developed algorithm, this contour string is approximated piecewise by straight lines and circular arcs. Relations like "collinear", "parallel" and "symmetric" between the segments are computed which leads to a symbolic description of the contour in form of a graph. Two well known approaches to object recognition, graph matching and evidence accumulation, are then applied to the segmented contours. While the first approach is designed to analyze a complete contour at once, the second one uses an iterative technique which, at each iteration, just analyses a small part of the contour. This approach is favored because it is much easier to recognize just one object instead of understanding a complete scene and a tray clearing task then only requires recognizing and removing one object at time. The evidence accumulation scheme is also preferred because of its potential for multi-knowledge-source integration. Information of multiple silhouettes or other gray-level images can easily be combined in a rather unstructured way. It is demonstrated how this can be done for the COR vision system which is built around two silhouette images and one gray-level image.
1.1 Organization of the thesis

- In order to situate the domain of this thesis relative to known work, chapter 2 presents a survey of some well known recognition systems.

- Chapter 3 describes segmentation techniques for digital curves. They are demonstrated on example contours of a cafeteria tray.

- Chapter 4 presents and compares two major paradigms for object recognition: attributed graph matching and evidence accumulation. It is shown how both techniques can be used to recognize objects in segmented contours described in chapter 3.

- In chapter 5 the application “cooperation robot” is presented with special focus on the vision system of that application. Extensions to the segmentation and recognition techniques from chapter 3 and 4 and their integration into the concrete and working application is shown.

- In chapter 6 final conclusions are drawn and future research directions are indicated.

- In the appendix A... E a number of additional results, descriptions and specifications to all chapters are listed.
Leer - Vide - Empty
Survey of related work

In this section, some well known present-day 3-D object recognition systems are presented. The discussion will be focused mainly on the following aspects:

- What primitives are used to describe scene and models?
- What matching algorithm is used?
- What results were achieved?
- Major advantage of the system.
- Major disadvantage of the system.

2.1 Flynn and Jain

Flynn and Jain describe in [FJ91] the model-based 3-D object recognition system “BONSAI” capable of interpreting range images. The object models are generated manually with the CAD system “GEOMOD”. Currently there are 20 models stored in the database. From the data format “IGES” of the CAD system, “BONSAI” automatically generates object models for the vision system in the form of a relational graph with nodes representing geometric primitives. Relations between them are represented by the edges of the graph.
Primitives are piecewise planar, cylindrical or spherical surfaces. In addition to this symbolic (view independent) attributes, patch areas from 320 different view points are calculated as view dependent attributes. The viewpoints are uniformly distributed over a sphere around each model object.

Input images are obtained from a laser range finder. They are segmented into surface patches with a data-driven, region based normal clustering analysis algorithm. First, a domain-independent merging algorithm merges adjacent similar patches. These are classified as planar, cylindrical or spherical using a regression and curvature-based technique. For the final representation, a second, model-based merging process, iteratively aggregates similar adjacent patches into bigger entities as long as there are patches to merge.

Object recognition is done with a graph matching algorithm that finds correspondences between the scene graph and the model graphs. To avoid a combinatorial explosion, unary and binary constraints prune the search tree. Constraints between nodes of the scene and nodes of the models (unary constraint) are the area, the type and the radius (only for cylindrical and spherical patches) of the patches. Between pairs of scene nodes and pairs of model nodes, 4 binary constraints are used. These are the rotation, the orientation, the visibility and the parallelism between the two pairs. To avoid a search of the complete model base with its 320 views of each of the 20 models, a list of good model candidates is built. It is dependent on the patches present in the scene. From the found matches between the scene nodes and a model, the rotation and translation from model to scene can be computed. The interpretation algorithm may generate multiple matches, so a verification is used to select the correct one. It is done by computing a synthetic range image from each matched model which is compared with the scene. The one with the smallest difference in patch areas is selected as the correct interpretation of the scene.

- **Primitives**: Planar, cylindrical and spherical surface patches.
- **Matching**: Constrained graph matching.
- **Results**: A correct recognition rate of 93%, a reject rate of 1 % and an incorrect recognition rate of 6 % is reported for a test series containing 5 range images of each of the 20 objects. With two-object scenes, the correct recognition rate decreases to approximately 70 %.
- **Advantage**: The combinatorial explosion is reduced.
2.2 Dickinson and Pentland

Disadvantage: The unary constraints "area" and "radius" are scale dependent. If the image acquisition geometry is not fixed as in the "BONSAI" lab the area constraint may lose value. Probably for the same reason, occlusion is barely allowed. Only one example with a scene containing two only slightly overlapping objects was shown. There is no mechanism to build patch groups as object hypotheses.

2.2 Dickinson and Pentland

[DPR92] presents a matching algorithm for recovering 3-D object information from single 2-D gray-level images. Instead of using simple 2-D structures as features more complex 3-D primitives are chosen. Recognition is done by matching aspects of these primitives to observed features. Probabilistic information is used to capture ambiguities in the mapping between 3-D primitives and the observed 2-D structures. Unlike the traditional approach which represents the entire object by a set of aspects, this approach uses aspects of 3-D primitives to represent parts of the objects instead. Ten 3-D primitives are used to build objects. These are block, truncated pyramid, pyramid, bent block, cylinder, truncated cone, cone, truncated ellipsoid, ellipsoid and bent cylinder. From these primitives, qualitative aspects in view centered coordinates are computed. This leads to a hierarchy of 2-D features. Relations and probabilities between these features are assessed for all viewpoints. All this processing can be done off-line. Recognition is done by segmenting the gray-level input image into regions which are then labeled according to the faces in the aspect hierarchy. With these regions, groups corresponding to aspects of the 3-D primitives are built and a mapping between them and the 2-D aspects leads to a set of recovered 3-D volumetric primitives. The probabilities within the hierarchy of aspects helps avoid a combinatorial explosion. The recovered primitives are used for recognition by indexing into a model database.

- **Primitives**: 3-D volumetric. 2-D aspects of them are used to construct the object model base.

- **Matching**: Segmented regions of the input are matched to pre-computed aspects. Found aspects are then mapped to the 3-D primitives.

- **Results**: Reported are results with synthetic images of scenes containing one or two-objects and with real scenes of one object. There is a small amount of occlusion in the two object scene.
• **Advantage:** Elaborated scheme for aspect matching with a fixed aspect set that is independent of the model object base.

• **Disadvantage:** The method has been proven to work only with fairly simple scenes (one or two objects). The images used were either synthetic or taken under ideal lab conditions. Objects with free-form surfaces cannot be represented.

### 2.3 Fan and Medioni

In [FMN88] a system for recognizing 3-D objects in range images is presented. A laser light plane projected onto a turning table is used as a range sensor. Objects are placed on that table and observed by a camera. With the help of triangulation, range images are computed from the images seen by the camera. They are segmented into 3-D faces of objects by detecting significant curvature features of the surface. Groups of surface points are aggregated to surface patches of simple surface models. In that way “planar”, “cylindrical” and “general quadric” surface patches are obtained. They are described with a variety of attributes like “visible area”, “orientation”, “average principal curvatures” and “estimated ratio of occlusion”. Boundary relations between these patches are then computed and classified into “jump boundary”, “convex crease”, “concave crease” and “limb”. Objects are described by graphs built using the patches and the relations. In these graphs, nodes represent surface patches and edges represent relations between them. 2 to 6 views are taken from each model object and the obtained graph is stored in the model database. For a reliable recognition it has to be ensured that each object face is seen at least in one view. Scenes may be composed of multiple mutual occluding objects. Data acquisition, segmentation and building of the attributed graph is done in the same way as it is done for the models. Recognition is done through a constrained graph matching. Three types of constraints, unary, binary and geometric are used to speed up the match. Multiple object scenes are split into subgraphs for recognition. To do so, for each edge a probability that its adjacent patches belong to the same object is computed. Edges with weak values are removed.

• **Primitives:** surface patches, planar, cylindrical, quadric.

• **Matching:** constrained graph matching.
• **Results:** Toy objects in a lab environment were used. The method was successfully demonstrated with 8 object models with a total of 28 views and scenes containing up to 4 objects.

• **Advantage:** moderately complex objects with occlusion can be described and matched.

• **Disadvantage:** the primitives are not sufficient for representing general objects. Ambiguities can happen for example on cylinders, which can be approximated through planar pieces as well as through cylindrical patches. This situation can result in no match or false matches.

### 2.4 Jain and Hoffman

In [JH88] Jain and Hoffman present a segmentation and recognition scheme for range images. In the low level processing steps, first the noise of the images and the background is filtered out. Pixels are then grouped into surface patches with a surface normal clustering algorithm. With the help of a model driven merging procedure, patches are classified as “planar”, “convex” or “concave”. To do so, knowledge about the boundary angles of each model object is used to perform the merging. This leads to \( N \) different representations of the scene, where \( N \) is the number of object models. The boundaries between adjacent patches are classified as “jump” or “crease” edges. In addition to this local information, some global values are computed. These are the perimeter of the set of non-background pixels, the number of connected background components and the number of connected background components within the convex hull formed on the set of non background pixels. The so obtained \( N \) scene representations are handed over to the recognition stage, which uses a knowledge-based evidence accumulation. Knowledge about the model objects under different views is classified in three types of evidence conditions. In the morphological class, there are statements about the global features. The perimeter of the non-background pixels for example has to be in a prescribed interval. In the second class statements about patch information are made and the third class deals with patch relations. To each evidence condition, a list of length \( N \) containing evidence weights is associated. Each scene representation is then tested for the evidence conditions and if they hold true, their weights are used to compute similarity values to each model object. The one with the highest value is selected as the result of the recognition.
• **Primitives**: Planar, concave and convex surface patches, global morphological information and patch relations

• **Matching**: Unstructured evidence accumulation.

• **Results**: A rule base with 32 rules for 10 objects was built. 6 objects were segmented with the described method and 4 were manually segmented. 31 scenes containing one of the 10 objects were successfully recognized. An example of a 2-object scene is also given.

• **Advantage**: There is no backtracking, which makes the recognition procedure fast. It has polynomial time complexity. It is flexible and easy to extend.

• **Disadvantage**: The knowledge-based merging procedure assumes that there is only one object in the scene. It won’t work with multiple object scenes. No mechanism to obtain the recognized object’s position.

### 2.5 Kuono and Okamoto

In [KOO91] a model based vision system for a maintenance robot of a nuclear power plant is presented. The main objective is to recognize objects like valves in gray-level images. In the model base, these objects are described in terms of the hierarchical PART-OF relation. Objects typically consist of several parts. The parts are modeled with primitives like cylinders etc. and the primitives are divided into a number of so called “Feature Generation Units” (FGE). FGE’s are parts of the primitives that generate 2-D features. A cylinder for example consist of 2 3-D disks which generate 2-D ellipses and a cylinder side which generates a 2-D parallel line pair. The 2-D features in the object models are view dependent and so they are computed for each model from many different views. Rank values according the number of viewpoints from which they are visible are associated with each of them. They are used to build up a strategy graph which helps in the recognition step to prune the search space when the possible model objects for a detected 2-D feature have to be found. Feature extraction is done by applying an edge detector to the input image and vectorizing the obtained edge pixels to line segments. An iterative hypothesize-and-test method then identifies certain types of features in certain regions of the image. Heuristics help to find initial guesses. With the precomputed strategy and the found features possible interpretations can
be found from the model data base. These hypotheses are then verified by projecting the 3-D model onto the image and comparing the so obtained 2-D features with the features in the image.

- **Primitives**: 3-D volumetric primitives like cylinder, sphere, cone etc. View dependent 2-D primitives like circle and parallel line are precomputed from them.

- **Matching**: With the help of the precomputed strategy graph, detected line segments are matched to hypothesized features.

- **Results**: The vision system was tested with some real images and various background illuminations but it was not yet combined with the robot.

- **Advantage**: Works with real images from a real world application.

- **Disadvantage**: Very specific object class.

### 2.6 Summary

More complete overviews of can be found in [BJ85] [Bra82] [CD86] and in the special issues of PAMI about interpretation of 3-D scenes [PAM91] and [PAM92]. We consider the work of Fan and Medioni and the one of Flynn and Jain good examples of model based recognition for range data. Even scenes with multiple object scenes can be handled nicely with the heuristics introduced by Fan and Medioni. However, both approaches are slow through the use of graph matching algorithm and because of expensive range data acquisition and segmentation. The low level part of the system of Flynn and Jain contains a knowledge based merging of adjacent patches similar to the one already used by Jain and Hoffman. Dickinsons and Pentlands approach improves performance and admits also objects having a more general geometry. Through the use of simple geometric 3-D primitives as they are often found in mechanical engineering CAD\(^1\) systems, their system may be easily interfaced to a such a system for realizing a complete CIM\(^2\) solution. The same holds true for the system of Flynn and Jain, which directly uses models constructed with a CAD system as input to the recognition system. No backtracking at all is used in

---

\(^1\)Computer Aided Design
\(^2\)Computer Integrated Manufacturing
the approach of Jain and Hoffman, which makes their recognition scheme fast and attractive for robot vision systems. Kuono’s and Okamoto’s contribution is the only one of the reviewed works which realizes a complete robot vision system, i.e. where the output of the recognition algorithm actually is used to control a robot. Although other work exists which describes complete vision systems, in general only little work is done for complete 3-dimensional robot vision systems.

The “COR” project was very ambitious under these considerations. In order to achieve its goals it was decided to use known technology wherever it was available and to adapt or modify it for the needs of the task. The demanded execution speed of the system had a major influence on the design of the complete vision system as well as on the recognition part. It was therefore definitely worth to look in more depth into the evidence based recognition scheme. With respect to data acquisition it appeared that no known techniques could directly be applied in the “COR” vision system. The following chapter therefore describes how the use of silhouettes which is well known in 2-D vision systems is adapted to arrive at a new 3-D sensing technique.
Segmentation of contours

3.1 Extracting contours

Gray-level images of a scene are taken by a CCD video camera, the optical axis of which lies in the plane defined by the upper rim of the tray. Behind the tray there is an illuminated background. With this arrangement the periphery of transparent objects is outlined perfectly well through the refraction effects. Their interiors are separated by a dark line from the light background. It is possible to trace that line and generate a perfect silhouette from it (fig. 3.1). As can be seen from the images in figure 3.1a the boundaries of the glass are clearly visible and it can be expected that there is no problem to detect edges in such an image. Indeed, experiments showed that even a simple gradient operator applied to such an image generates a strong enough response. Nevertheless we use a Canny operator [Can83], mainly because in a later processing step the $\theta(s)$ diagram of the contour is needed.

For further processing we are interested in a coordinate string of the contour as well as in the filled-in silhouette, since both can be used to generate additional information pertaining to the scene, namely space occupancy maps and maps of regions belonging to objects made of glass. Therefore a binary labeled image separating foreground and background is needed. To obtain it, the thresholded gradient magnitude image is subjected to a thinning operator which results in an image in which the contour of the silhouette is clearly visible and extends
Figure 3.1: Gray level images of a typical scene (a), its edge filter output (b) and the generated silhouette (c)

across the entire image (fig. 3.1b). Above the contour there is no response of the edge detector at all due to the homogeneous illumination of the background. Below the contour, several edges may be visible. A connected component labeling, where the component above the contour gets one color and all the rest including the contour gets a different one, easily partitions such a scene (fig. 3.1c). It has to be noted, that in this simple way it is not possible to obtain the holes in the foreground (scene objects). For example the handle of a cup would be filled completely and only the outer contour of it remains visible. This has the disadvantage that information is lost but the advantage that the contour is defined by just one string of coordinate pairs which can be processed very easily and efficiently.
3.2. Significant points and segments

The coordinate pair string of the contour is extracted from the binary labeled image by following the border between black and white regions, starting at the left end of the image. For this purpose, we use the technique described in [Ilg86]. A 2x2 pixel window is moved along the border; depending on the values of the 4 pixels, the direction of the next move is chosen (fig. 3.2). Because of the previous labeling step, we can guarantee that the foreground is 4-connected so there are no ambiguities and the move direction is always uniquely determined. As the window moves along the border, its position is recorded. All recorded positions form the contour as a string of coordinate pairs. In the following section it is shown how such a string can be segmented.

3.2 Detecting significant points and segments in digital curves

It is difficult to define unambiguously what constitutes a significant point of a contour. Depending on purpose, vocabulary and data representation, different points can be selected. The localization of these points also is not clear. Many algorithms have been published but problems still exist e.g. with smooth transitions between contour parts and small regions with high curvature. For these reasons, we have developed a hybrid split-and-merge algorithm that uses local and global criteria. We introduce the concept of a "separator domain". Instead of pinpointing individual unreliable corner and inflection points, we mark small regions on the contour as "separator domains". As a consequence, the segments between the separator domains are much cleaner and can therefore be approximated more reliably with the help of a vocabulary of graphic primitives. Output of the algorithm is a symbolic description of the contour in the form of a list of line and arc segments. Each element of the list belongs either to the class of contour segments or to the
class of separator domains and is characterized through various attributes. The algorithm was tested with a large number of examples from a real world application and it worked well.

3.2.1 Finding significant points in digital curves

Vision systems usually contain steps which work in a top-down direction as well as others working bottom-up. The same can hold true for subproblems in such a system e.g. curve partitioning by significant points. In bottom-up algorithms one tries to detect significant points directly by analyzing small local groups of curve points for certain properties like curvature or turning angle. Where these properties have extrema or discontinuities significant points are assumed. Problems with these approaches derive from the digital nature of the curves where quantization effects and other noise make it hard to compute derivatives. Falsely determined points or wrong positions of them may result. Examples of algorithms that try various approaches to overcome these difficulties can be found in [RJ73], [RW75], [FD77], [SS78], [AB84], [Ced78], [KR78], [TC89a]. All these algorithms are purely data driven, no a priori knowledge is used.

In contrast, the other main paradigm uses at least some rudimentary a priori knowledge to detect significant points. The curve is assumed to be composed of a finite set of primitives like straight line segments or arcs. Stated more generally, it is tried to approximate the curve in certain regions through various fits. Different measures of goodness of fit guide this procedure. Significant points are defined as points between adjacent segments. Choosing the vocabulary of the graphic primitives is critical and domain specific, these approaches therefore tend to be less general than pure bottom up approaches. Other problems lie in the dependency on the order in which the fits are applied and in the fits themselves which sometimes encounter hard limits when implemented digitally (see chapter 3.2.7). Representative examples of these algorithms can be found in [Pav80], [Wu84], [BA85], [Dun86], [WD84], [Ima86].
3.2. Significant points and segments

3.2.2 A hybrid approach to detect significant points in digital curves

To overcome the problems of the local algorithms as well as those of the global approaches, we propose a hybrid split-and-merge algorithm that combines the advantages of both worlds. Before we come to the details of the algorithm, let's think a bit more fundamentally about segmentation.

Segmentation partitions a contour into segments which are typically large structures. Different algorithms may find different positions of these segments, but compared to the size of the segments, the changes are small. Segment position can be computed reliably and uniquely. Corners separate segments. They are typically small structures down to the size of one point. The position of that corner point is often dependent on the segmentation algorithm and it is therefore hard to determine it reliably.

Even in physical reality, object corners are often not very precisely localized. Different material properties or manufacturing processes for example can blur object corners. Another point to be considered is that digital camera systems still have quantization noise which prevents exact localization especially in sharp corners (high frequencies). For all these reasons, we introduce the concept of the separator domain. Instead of computing imprecise individual corner or inflection points, small regions on the contour called "separator domains" are determined. To each "separator domain" as well as to the segments a certainty factor is attached. These certainty factors lie in the range [0..1] and express always how much confidence is put into the domain or segment. A value of 0 means no confidence whereas 1 means complete confidence.

Our segmentation algorithm comprises three stages:

1. In the first step, a first choice of significant points is localized with a maximum curvature criterion. To each point a separator domain (section 3.2.5) and a certainty factor is attached. Inflection points and flat transitions between segments are not detected in this step mainly because known methods are too sensitive to noise. It is more reliable to detect them on more structured entities than points. On the other hand we do get some wrong points which do not correspond to significant object marks through the presence of noise. To correct this weakness
and to get a symbolic description of the contour, we add a second step, which works globally.

2. The segments between the found separator domains are approximated through primitives (section 3.2.7) which are straight line segments and circles in the current implementation. Certain criteria like the goodness of fit control the approximation and introduce new corner points when necessary (section 3.2.6). Besides that it is also possible to detect inflection points not seen in stage 1. The second step of the algorithm usually contributes to oversegmentation.

3. A final step is necessary to clean up, i.e. to remove superfluous separators and to merge segments (section 3.2.8). We do so by computing a certainty factor for each corner domain using a heuristic evaluation function. Values like segment length and goodness of fit are used in that function. Corner points having values below a certain limit are eliminated by merging the adjacent segments. As long as there are points with small certainty factors, this step is repeated.

The following sections explain in detail the three steps of the algorithm and how the separators are found.

### 3.2.3 Seed corner point detection

Seed points are used as initial guesses of corner points. We find them as points with curvature higher than a certain threshold. Since these points are used just as a first hypothesis which may be eliminated later, the threshold does not have a large influence on the overall result. However, instead of using a fixed threshold, it is calculated with the help of a curvature histogram of the whole contour. Curvature for each point is normalized to a value in the interval [0,100]. Typical contours can have large approximately straight sections. Their low curvature values contain no useful information and therefore they are not allowed to contribute to the histogram. The threshold $\kappa_t$ for the curvature is computed based on the histogram according to one of the following two criteria:

1. $\kappa_t$ is chosen such that a prescribed percentage $\nu_1$ of points has a smaller curvature than $\kappa_t$. 
2. $\kappa_t$ is chosen such that the histogram values fall below a prescribed percentage $\nu_2$ of the maximum value.

Points $p_i$ having a curvature $\kappa(p_i)$ greater than the threshold curvature $\kappa_t$ are selected as seed points. If many adjacent points fulfill the criterion, then the one with the highest curvature in that neighborhood is chosen as the seed point. All seed points are then handed over to the separator domain finder.

### 3.2.4 Algorithms for computing $\theta(s)$ and curvature

For curvature calculation, we compute first $\theta(s)$, the direction angle versus the arc length of the contour. For continuous curves $(x(s), y(s))$, $\theta_i(s)$ is defined as the angle of the tangent to the curve in position $i$ (3.1).

$$\theta(s_i) = \frac{dy(s)/ds}{dx(s)/ds}$$  \hspace{1cm} (3.1)

In the case of digital curves, different algorithms exist to compute $\theta(s)$. A naive approach is to compute the secant instead of the tangent. Equation 3.2 shows how $\theta(s)$ at position $s_i$ can be calculated with that method.

$$\theta(s_i) = \arcsin\left(\frac{y_{i+k} - y_{i-k}}{\sqrt{(x_{i+k} - x_{i-k})^2 + (y_{i+k} - y_{i-k})^2}}\right)$$  \hspace{1cm} (3.2)

It is clear that in the case $k = 1$ the tangents of $\theta$ on the digital curve can only vary in discrete increments of $\frac{1}{2}$. To get a finer resolution of $\theta(s)$ values of $k > 1$ are usually chosen. But even with larger values for $k$, $\theta$ changes only in discrete steps. Figure 3.4 shows $\theta(s)$ of the contour from figure 3.7 computed with a secant over 11 points (i.e. $k = 5$). Small jumps in $\theta$ are clearly visible. These have a dramatic effect on the curvature calculation since for that, the derivative of $\theta(s)$ has to be computed. A low pass filter applied to $\theta(s)$ would remove the spikes, but it also would blur the edges which is not desirable. To overcome that, an edge preserving smoothing technique was presented in [API90]. Nevertheless, problems exist with 3.2 for the $\theta(s)$ computation: a) it is not clear how to choose $k$, b) the parameter $k$ means that the curve is smoothed and the angle calculation therefore looses precision and
Figure 3.3: Problem with the line fit which can lock in situations like (b), orthogonal to the contour

c) the method is still noise sensitive since only two points \( s_{i-k} \) and \( s_{i+k} \) of the contour are used for the calculation.

Mainly the last point leads to the idea of using a local support that takes into account a certain number \( k \) of points around \( s_i \). To compute \( \theta(s) \) in position \( i \), the contour is approximated in a small neighborhood of \( s_i \) through a straight line as defined in section 3.2.7. Then the direction angle of that straight line is used as \( \theta \) of the contour in point \( i \). Usually, the fitted line represents a more accurate approximation of the tangent to the contour than the secant method. Another advantage is the robustness against noise since the fit itself is a smoothing filter. Problems arise in very narrowly curved parts like corners with radii smaller than \( k \). Figure 3.3 shows such a situation in the lower left corner of the example contour from figure 3.7. It happens there that the fit locks orthogonally to the direction of the contour which leads to a completely wrong \( \theta(s) \) and curvature \( \kappa(s) \) in that point as figure 3.5 shows. To prevent that, we interpolate \( \theta \) in these situations with the help of the neighbors. Finally, curvature \( \kappa(p_i) \) is calculated from the difference \( \theta(p_{i+1}) - \theta(p_{i-1}) \).
3.2. Significant points and segments

Figure 3.4: Diagram (a) shows $\theta(s)$ of the contour from figure 3.7 computed with a secant over 11 points. (b) is a magnified section from (a) with $s$ in the range $[150,450]$. The steps in $\theta(s)$ is quantization noise resulting from computing $\theta$ with the secant method.
Figure 3.5: $\theta(s)$ (a) and $\kappa(s)$ (b) of the contour part from figure 3.3. Without correction, the fit locks in the corner to a wrong position which results in wrong $\theta$ and $\kappa$ values. The interpolation result is also shown.
3.2. Significant points and segments

3.2.5 Delimiting separator domains

Initially, each separator domain consists only of the seed corner point. Then, it is enlarged by adding all points around the seed point having the same sign of curvature. This leads to a domain which typically is too large. Figure 3.6a shows this stage of the algorithm for a typical situation. In an ideal corner for example, the resulting domain is about as large as the size of the kernel of the \( \theta(s) \) computation. Also in the presence of noise the domain may get too large. As a final step, the domain is therefore shrunk as much as possible. Two criteria control this procedure. A first criterion prevents the domain from shrinking to less than the minimal domain length of 3 points. A second criterion makes sure that points with enough curvature change stay in the domain. For that \( \kappa_{p_i} \), the curvature with respect to \( p_i \) is computed. For that, line fits are made symmetrically on both sides of \( p_i \). They are made on both sides of \( p_i \) between the end points of the domain \( p_{i-k} \) and \( p_{i+k} \). Successively \( k \) is decremented so that the end points move closer to \( p_i \). For each \( k \) new fits are computed. The difference of successive fitted line angles gives the curvatures \( \kappa_{p_i}^{(i+k)} \) and \( \kappa_{p_i}^{(i-k)} \). If it changes more than a certain amount, the shrinking procedure stops on that side. The procedure is repeated until on both sides one of the two criteria is fulfilled.

![Figure 3.6: Initial corner domain contains all points around \( p_i \)(a). Shrinking of the domain (b) and the resulting domain\(^1\)(c)](image)

---

3.2.6 Approximation of segments

Regions on the contour between separators are now approximated by a set of primitives. The selection of these primitives is dependent on the curves to be approximated.

---

\(^1\)To make it easier to draw \( \alpha \), secants instead of line fits are shown in this figure
Chapter 3. Segmentation of contours

Figure 3.7: Example contour of a plate with a fork, a glass and a soup bowl in it. The marked region is used to illustrate the algorithm

analyzed. For the analysis of the contours of the "cafeteria tray problem" a set composed of straight line and circular arc is appropriate because all objects can be well approximated with these two primitives. The two primitives are implemented with non-iterative line and circular arc fits. A detailed description of these fit procedures can be found in section 3.2.7. Thanks to the separator domains the segments to be approximated are cleaner so that higher demands on the quality of the fits can be used which leads to a more accurate description. Always, both line and arc fits are done delaying the decision which one to use till the merging step. If the fits are good enough the analyzed segment is called a segment hypothesis and the procedure is continued with the next candidate segment on the contour.

If a segment can’t be fitted accurately enough, a new segmentation point is introduced. In contrast to the seed corner point detection where a local criterion is used, this time it is possible to distinguish between corner and inflection points since we use the global criterion of the fit. The new segmentation point is found by looking at the segment and the secant connecting start and endpoint. With the help of runs that we define as a sequence of contour points lying on one side of the secant we decide whether a new segmentation point has to be set and what kind it is. If a corner point is appropriate to set we should have just one run, whereas inflection points produce at least two runs (fig. 3.8). Problems arise with noise which can lead to many wrong inflection points. But the runs generated in these situations typically are much smaller than runs around real inflection points. These situations therefore are easy to resolve with a minimum length criterion applied to the runs.

Two additional criteria for inflection points are in effect to prevent the
3.2. Significant points and segments

Figure 3.8: One run is seen in the situation of a new corner point (a) whereas around inflection points two runs are seen.

Partitioning of segments that have only little change in direction: First, the difference of the maximum direction angle \( \theta_{max} \) and the minimum direction angle \( \theta_{min} \) of the segment points has to be larger than a certain threshold. And second, the differences of the maximum direction angle and the angles on the start respectively the end point, too, have to be larger than a certain angle. As inflection point the one with the smallest angle is taken in right to left transitions and in left to right transitions it is the one with the largest angle. Positions found with that procedure are not very precise. But the larger the direction change in the segment, the better an inflection point is detectable. This is also reflected in the definition of the certainty factor \( \text{c}^{inf} \) shown in formula 3.3. High values of \( \text{c}^{inf} \) express a large change of direction in the segments generally lead to better and more reliable inflection points. However, generally inflection points are not as reliably determinable as corner points so that \( \text{c}^{inf} \) is always much smaller than 1 through the normalisation of the direction change to \( \pi \). Such a certainty factor assignment looks reasonable.

\[
\text{c}^{inf} = \frac{\theta_{max} - \theta_{min}}{\pi}
\]  

(3.3)

However, these inflection points are expanded into separator domains whereby their exact position loses importance.

When the criteria for inflection points fail, a corner point is assumed. Its position is found with a slightly modified criterion used already by [DP73]. Start and end point are connected with a straight line and the point with the
maximum distance to that line is selected. Since such a single point is very noise sensitive, it is not taken as the new corner point as suggested in [DP73]. Instead, the curvature in a small neighborhood around this point is analyzed and the point having maximal curvature is taken as the corner point. Again, as with the inflection points, these corner points are used as new seed points for separator domains.

Since all criteria applied are very tight, the procedure typically generates an oversegmentation, but that will be corrected in the final merge step (section 3.2.8).

### 3.2.7 Fit primitives for approximation

Two types of primitives are used for approximation of the contours in stage 2 as well as in stage 3: a line fit and a circle fit. This is because our contours are generated from silhouettes from objects like glass, plate, cup etc. Most of them are composed mainly of straight lines or arc segments because they correspond to inherent geometric properties or are due to the perspective mapping (see fig. 3.7).

The line fit is implemented by finding the axis of the minimal moment of inertia. The criterion to be minimized is the sum of square distances from the contour points to the approximating line. As output, one gets the direction angle $\theta$ of the line, the center of gravity $x_c, y_c$ and the standard deviation $\sigma$ of the fit. Formula 3.4–3.10 shows the definition of these values for a segment of length $L$. The following two experiments help us choose a value of $\sigma$ for acceptance of a fit:

1. A horizontal straight line having its points flipped up and down randomly by one pixel would yield a $\sigma$ of 0.35.

2. Digitized perfect straight lines generated with a Bresenham algorithm ([Bre65]) yield $\sigma = 0$ only in special cases. In all other cases, $\sigma$ is between 0.2 and 0.3 for all angles and length of the digital line. Figure 3.9 shows $\sigma$ as function of the direction angle of the straight line with different line length.

Considering these two experiments, we chose a $\sigma$ value of 0.35 for acceptance, which means a rather tight constraint.
3.2. Significant points and segments

Figure 3.9: $\sigma(\theta)$ for different length of a digitized straight line

\[
x_c = \frac{1}{L} \sum_{k=0}^{k=L-1} x_k \tag{3.4}
\]

\[
y_c = \frac{1}{L} \sum_{k=0}^{k=L-1} y_k \tag{3.5}
\]

\[
m_{xx} = \sum_{k=0}^{k=L-1} (x_k - x_c)^2 \tag{3.6}
\]

\[
m_{yy} = \sum_{k=0}^{k=L-2} (y_k - y_c)^2 \tag{3.7}
\]

\[
m_{xy} = \sum_{k=0}^{k=L-1} (x_k - x_c)(y_k - y_c) \tag{3.8}
\]
\[ \theta(s) = \frac{1}{2} \arctg \left( \frac{2m_{xy}^i}{m_{xx} - m_{yy}} \right) \]  \hspace{1cm} (3.9) \\
\[ \sigma = \sqrt{\frac{\sum_{k=0}^{L-1} \left( ((y_k - y_c) \cos \theta) - ((x_k - x_c) \sin \theta))^2 \right)} \] \hspace{1cm} (3.10) \\

For the circle fit we use the non-iterative algorithm of Thomas and Chan [TC89b] who minimize the sum of the squares of

\[ \pi \left( (x_i - x_c)^2 + (y_i - y_c)^2 \right) - \pi R^2 \] \hspace{1cm} (3.11) \\
\[ \sigma = \sqrt{\frac{\sum_{i=0}^{L-1} \left( R - \sqrt{(y_i - y_c)^2 + (x_i - x_c)^2} \right)^2}{L - 1}} \] \hspace{1cm} (3.12)

\( x_c \) and \( y_c \) are the center coordinates of the sought-for circle, \( R \) its radius. With 3.11 also a standard deviation \( \sigma \) can be calculated 3.12, but care has to be taken for setting the value for acceptance. \( \sigma \) is dependent on the opening angle of the circular arc as well as on the radius. Similar to the line fit experiments with digitized circular arcs were made to get reasonable values for acceptance values of \( \sigma \). Figure 3.10 shows \( \sigma \) values obtained by approximating with 3.11 a digitized circular arc with different radii and opening angles. As we can see, \( \sigma \) hardly gets smaller than 0.2 (except in special cases) and may be even above 0.4. Nevertheless we choose as threshold a value of \( \sigma_t = 0.35 \) for acceptance of a fit which gives us again a very tight constraint. The certainty factor assigned to the segment approximation is proportional to \( \sigma \) and normalized with the help of \( \sigma_t \) to the range \([0,1]\).

\[ c^{arc} = \frac{\sigma_t - \sigma}{\sigma_t} \quad \text{and} \quad c^{lin} = \frac{\sigma_t - \sigma}{\sigma_t} \] \hspace{1cm} (3.13)

A \( c^{lin}_i = 1 \) therefore means a very good linear approximation of segment \( i \) whereas values close to 0 means a bad or unreliable approximation.
Figure 3.10: $\sigma(R)$ for digitized circular arcs with different opening angles and radii


### 3.2.8 Elimination of oversegmentation

Unnecessary or badly set segmentation points (corners and inflection points) are removed by iteratively merging adjacent segments. Criteria used in merging are the already assigned certainty factors for corner points, inflection points and approximated segments, the quality of the newly generated segment and the length of the segments. The basic idea is to get segments as long as possible but not to loose fit quality too much. Only segments pointed to by a weak segmentation point are considered for the merging process.

![Separator domains](image)

**Figure 3.11:** Separator domains (black) with oversegmentation before (a) and after (b) the clean-up step

The algorithm works by always looking at three adjacent segments $s_i$, $s_{i+1}$ and $s_{i+2}$ and producing two new ones $f_i$ and $f_{i+1}$ out of them. Over $s_i$ and $s_{i+1}$ we get $f_i^{arc}$ and $f_i^{lin}$ with a line and an arc fit with certainty factors $c_i^{arc}$ and $c_i^{lin}$. The same is done for $s_{i+1}$ and $s_{i+2}$ which results in $f_{i+1}^{arc}$ and $f_{i+1}^{lin}$ with $c_{i+1}^{arc}$ and $c_{i+1}^{lin}$. To merge the segments, one has to fulfill two contradictory goals: On one hand one wants to make the fits as good as possible and on the other, one wants to have as long segments as possible. To find a solution to that problem, we define a heuristic evaluation function $m_{fi}$ (3.14-3.16). It
3.2. Significant points and segments

takes into account the length $l_{f_i}$ and $l_{f_{i+1}}$ of the new segments $f_i$ and $f_{i+1}$ as well as their certainty factors and maximizes the sum of their product. If the evaluation result of that function is larger than a threshold $m_t$ the merge from $s_i, s_{i+1}$ and $s_{i+2}$ to $f_i$ and $f_{i+1}$ is accepted. If this is not the case, no changes are made and the iteration moves on to the segments $s_{i+1}, s_{i+2}$ and $s_{i+3}$. This procedure is repeated as long as there are segments to merge.

\begin{figure}
\centering
\caption{Parts of the contour approximated by straight line segments and circular arcs}
\end{figure}

\begin{figure}
\centering
\caption{The complete example contour approximated by straight line segments and arcs}
\end{figure}

\begin{align}
    m_{f_i}^{arc} &= \frac{c_i^{arc}l_{f_i} + c_{i+1}^{arc}l_{f_{i+1}}}{l_{f_i} + l_{f_{i+1}}} \tag{3.14} \\
    m_{f_i}^{lin} &= \frac{c_i^{lin}l_{f_i} + c_{i+1}^{lin}l_{f_{i+1}}}{l_{f_i} + l_{f_{i+1}}} \tag{3.15} \\
    m_{f_i} &= \min(m_{f_i}^{arc}, m_{f_i}^{lin}) \tag{3.16}
\end{align}
Figure 3.14: Significant points found with two different parameter sets of the Rosenfeld-Johnson algorithm. Separator points on slowly changing transitions between two object parts (arrow) are not found.

3.3 Relations between segments

From the described algorithm we get an attributed list of contour segments and separator domains. To make such a structure usable for object recognition, it is important to map the geometric relations between the segments into relational links between the list elements.

Table 3.1 illustrates the relations between the attributes of the segmented contour from figure 3.15. The segment numbers label rows and columns. Each cell contains three numbers: the sum of the direction angles, the quotient of the lengths and the quotient of the circle radii. Comparing segment 4 for example with all others, we see that the angle sum with segment 8 is zero which can be used to hypothesize a vertical symmetry axis between these two
3.3. Relations between segments

Figure 3.15: Example scene used in conjunction with table 3.1 for demonstrating the relations between segments

segments. Additional support to that hypothesis would come from the fact that their length is almost the same (length(4)/length(8) = 1.06) and that the adjacent segments 3-9, 2-10 have (almost) the same angle sum. Building such relations can be used to construct an attributed graph. Nodes in that graph are the found segments with their attributes (fit type, certainty factor etc.). The arcs of the graph are geometric relations between the segment also described by various attributes. One of them is a certainty factor that expresses how certain a relation is. If a relation criterion is perfectly fulfilled then a certainty factor of one is assigned, otherwise the factor is decreased towards 0. The relations and attributes are used:

- **adjacency**: This relation is already expressed in the segment list. Segments being connected by a separator domain are called adjacent. As attribute we define the angle $\theta_{ab}$ between the two segments $S_a$ and $S_b$ for linear segments. No attribute is defined when $S_a$ or $S_b$ is an arc segment.

- **parallelism**: The following conditions have to be fulfilled to consider two segments $S_a$ and $S_b$ as parallel:
  1. $S_a$ must be approximated by a straight line having direction angle $\theta_a$
  2. $S_b$ must be approximated by a straight line having direction angle $\theta_b$
3. \( \theta_a \) modulo \( \pi \) must be "about equal" to \( \theta_b \) modulo \( \pi 

4. \( S_a \) and \( S_b \) should face each other, i.e. the connecting line between the center of \( S_a \) and \( S_b \) should be "about orthogonal" to \( S_a \) and \( S_b \).

The term "about equal" of the angles is necessary to overcome noise or quantization errors in the segmentation. In the current implementation, the angles \( \theta_a \) and \( \theta_b \) may not differ more than \( \theta^\Pi = \frac{\pi}{60} \). The 4th condition expresses that we are mainly interested in symmetric parallel relations because in many of our objects they are present. This condition is measured with the the angle \( \alpha \) as defined in figure 3.16b. Nevertheless, parallelism is not only restricted to them through the use of a rather large threshold value \( \alpha_t = \frac{\pi}{2} \) for the angle \( \alpha \).

The quality of the parallel relation is expressed with a certainty factor \( c^p \). A value close to 1 means a good parallelism whereas values close to 0 express bad parallelism. The parallel relation is considered good if the two segments \( S_a \) and \( S_b \) are reliable (have high certainty factors), if their length \( l_{S_a} \) and \( l_{S_b} \) are similar, if their directions \( \theta_a \) and \( \theta_b \) are similar and if \( S_a \) and \( S_b \) face each other directly. \( c^p \) is defined considering these

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \theta_1 )</th>
<th>( \theta_2 )</th>
<th>( \theta_3 )</th>
<th>( \theta_4 )</th>
<th>( \theta_5 )</th>
<th>( \theta_6 )</th>
<th>( \theta_7 )</th>
<th>( \theta_8 )</th>
<th>( \theta_9 )</th>
<th>( \theta_{10} )</th>
<th>( \theta_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.06</td>
<td>-1.75</td>
<td>-2.32</td>
<td>-1.54</td>
<td>0.05</td>
<td>-0.43</td>
<td>2.72</td>
<td>1.60</td>
<td>2.39</td>
<td>1.83</td>
<td>0.03</td>
</tr>
<tr>
<td>2</td>
<td>1.00</td>
<td>0.57</td>
<td>0.07</td>
<td>0.39</td>
<td>2.93</td>
<td>1.83</td>
<td>1.63</td>
<td>0.41</td>
<td>0.04</td>
<td>0.54</td>
<td>0.91</td>
</tr>
<tr>
<td>3</td>
<td>-1.75</td>
<td>-3.56</td>
<td>-4.14</td>
<td>-3.35</td>
<td>-1.76</td>
<td>-2.24</td>
<td>0.91</td>
<td>-0.21</td>
<td>0.57</td>
<td>0.02</td>
<td>-1.78</td>
</tr>
<tr>
<td>4</td>
<td>1.77</td>
<td>1.00</td>
<td>0.12</td>
<td>0.39</td>
<td>5.19</td>
<td>3.23</td>
<td>2.88</td>
<td>0.73</td>
<td>0.58</td>
<td>1.06</td>
<td>1.63</td>
</tr>
<tr>
<td>5</td>
<td>1.55</td>
<td>1.00</td>
<td>-20.07</td>
<td>-20.07</td>
<td>22.84</td>
<td>5.20</td>
<td>5.49</td>
<td>20.07</td>
<td>0.22</td>
<td>-20.07</td>
<td>-20.07</td>
</tr>
<tr>
<td>6</td>
<td>-2.32</td>
<td>-4.14</td>
<td>-4.71</td>
<td>-3.93</td>
<td>-3.34</td>
<td>-2.82</td>
<td>0.33</td>
<td>-0.79</td>
<td>0.60</td>
<td>-0.56</td>
<td>-2.36</td>
</tr>
<tr>
<td>7</td>
<td>15.33</td>
<td>8.67</td>
<td>1.00</td>
<td>6.00</td>
<td>45.00</td>
<td>28.00</td>
<td>25.00</td>
<td>6.33</td>
<td>0.67</td>
<td>8.33</td>
<td>14.00</td>
</tr>
<tr>
<td>8</td>
<td>-0.08</td>
<td>-0.05</td>
<td>1.00</td>
<td>1.00</td>
<td>-1.14</td>
<td>-0.26</td>
<td>-0.27</td>
<td>1.00</td>
<td>0.00</td>
<td>-0.01</td>
<td>1.00</td>
</tr>
<tr>
<td>9</td>
<td>-1.54</td>
<td>-3.35</td>
<td>-3.34</td>
<td>-1.55</td>
<td>-2.03</td>
<td>1.12</td>
<td>0.00</td>
<td>-0.79</td>
<td>0.23</td>
<td>-1.57</td>
<td>1.00</td>
</tr>
<tr>
<td>10</td>
<td>2.56</td>
<td>1.44</td>
<td>0.17</td>
<td>1.00</td>
<td>7.50</td>
<td>4.67</td>
<td>4.17</td>
<td>1.06</td>
<td>0.11</td>
<td>1.39</td>
<td>2.33</td>
</tr>
<tr>
<td>11</td>
<td>-0.08</td>
<td>-0.05</td>
<td>1.00</td>
<td>1.00</td>
<td>-1.14</td>
<td>-0.26</td>
<td>-0.27</td>
<td>1.00</td>
<td>0.00</td>
<td>-0.01</td>
<td>1.00</td>
</tr>
<tr>
<td>12</td>
<td>0.05</td>
<td>-1.76</td>
<td>-2.34</td>
<td>-1.55</td>
<td>0.04</td>
<td>-0.44</td>
<td>2.71</td>
<td>1.39</td>
<td>2.37</td>
<td>1.82</td>
<td>0.02</td>
</tr>
<tr>
<td>13</td>
<td>0.34</td>
<td>0.19</td>
<td>0.02</td>
<td>0.13</td>
<td>1.00</td>
<td>0.62</td>
<td>0.56</td>
<td>0.14</td>
<td>0.01</td>
<td>0.19</td>
<td>0.31</td>
</tr>
<tr>
<td>14</td>
<td>0.07</td>
<td>0.04</td>
<td>-0.88</td>
<td>-0.88</td>
<td>1.00</td>
<td>0.23</td>
<td>0.24</td>
<td>-0.88</td>
<td>0.01</td>
<td>-0.88</td>
<td>0.01</td>
</tr>
<tr>
<td>15</td>
<td>-0.43</td>
<td>-2.24</td>
<td>-2.82</td>
<td>-2.03</td>
<td>-0.44</td>
<td>-0.92</td>
<td>2.23</td>
<td>1.11</td>
<td>1.89</td>
<td>1.34</td>
<td>-0.46</td>
</tr>
<tr>
<td>16</td>
<td>0.55</td>
<td>0.31</td>
<td>0.04</td>
<td>0.21</td>
<td>1.61</td>
<td>1.00</td>
<td>0.89</td>
<td>0.23</td>
<td>0.02</td>
<td>0.30</td>
<td>0.50</td>
</tr>
<tr>
<td>17</td>
<td>0.30</td>
<td>0.19</td>
<td>-3.86</td>
<td>-3.86</td>
<td>4.39</td>
<td>1.00</td>
<td>1.06</td>
<td>-3.86</td>
<td>-3.86</td>
<td>0.04</td>
<td>-3.86</td>
</tr>
<tr>
<td>18</td>
<td>2.72</td>
<td>0.91</td>
<td>0.33</td>
<td>1.12</td>
<td>2.71</td>
<td>2.23</td>
<td>5.38</td>
<td>4.26</td>
<td>5.04</td>
<td>4.49</td>
<td>2.69</td>
</tr>
<tr>
<td>19</td>
<td>0.61</td>
<td>0.35</td>
<td>0.04</td>
<td>0.24</td>
<td>1.80</td>
<td>1.12</td>
<td>1.00</td>
<td>0.25</td>
<td>0.03</td>
<td>0.33</td>
<td>0.56</td>
</tr>
<tr>
<td>20</td>
<td>0.38</td>
<td>0.18</td>
<td>-3.66</td>
<td>-3.66</td>
<td>4.16</td>
<td>0.95</td>
<td>1.00</td>
<td>-3.66</td>
<td>-3.66</td>
<td>0.04</td>
<td>-3.66</td>
</tr>
<tr>
<td>21</td>
<td>1.60</td>
<td>-0.21</td>
<td>-0.79</td>
<td>0.00</td>
<td>1.59</td>
<td>1.11</td>
<td>4.26</td>
<td>3.14</td>
<td>3.93</td>
<td>3.37</td>
<td>1.57</td>
</tr>
<tr>
<td>22</td>
<td>2.42</td>
<td>1.37</td>
<td>0.16</td>
<td>0.95</td>
<td>7.11</td>
<td>4.42</td>
<td>3.95</td>
<td>1.00</td>
<td>0.11</td>
<td>1.32</td>
<td>2.21</td>
</tr>
<tr>
<td>23</td>
<td>-0.08</td>
<td>-0.05</td>
<td>1.00</td>
<td>1.00</td>
<td>-1.14</td>
<td>-0.26</td>
<td>-0.27</td>
<td>1.00</td>
<td>0.00</td>
<td>-0.01</td>
<td>1.00</td>
</tr>
<tr>
<td>24</td>
<td>2.39</td>
<td>0.57</td>
<td>0.60</td>
<td>0.79</td>
<td>2.37</td>
<td>1.89</td>
<td>5.04</td>
<td>3.93</td>
<td>4.71</td>
<td>4.15</td>
<td>2.56</td>
</tr>
<tr>
<td>25</td>
<td>23.00</td>
<td>15.00</td>
<td>1.30</td>
<td>9.00</td>
<td>67.50</td>
<td>42.00</td>
<td>37.50</td>
<td>9.50</td>
<td>1.00</td>
<td>12.50</td>
<td>21.00</td>
</tr>
<tr>
<td>26</td>
<td>-0.08</td>
<td>-0.05</td>
<td>1.00</td>
<td>1.00</td>
<td>-1.14</td>
<td>-0.26</td>
<td>-0.27</td>
<td>1.00</td>
<td>0.00</td>
<td>-0.01</td>
<td>1.00</td>
</tr>
<tr>
<td>27</td>
<td>1.83</td>
<td>0.02</td>
<td>-0.56</td>
<td>0.33</td>
<td>1.82</td>
<td>1.34</td>
<td>4.49</td>
<td>3.37</td>
<td>4.15</td>
<td>3.60</td>
<td>1.80</td>
</tr>
<tr>
<td>28</td>
<td>1.84</td>
<td>1.04</td>
<td>0.12</td>
<td>0.72</td>
<td>5.40</td>
<td>3.36</td>
<td>3.00</td>
<td>0.76</td>
<td>0.08</td>
<td>1.00</td>
<td>1.68</td>
</tr>
<tr>
<td>29</td>
<td>1.01</td>
<td>4.52</td>
<td>-4.07</td>
<td>-0.70</td>
<td>-0.70</td>
<td>15.20</td>
<td>25.49</td>
<td>24.80</td>
<td>-9.07</td>
<td>-9.07</td>
<td>-0.70</td>
</tr>
<tr>
<td>30</td>
<td>-0.03</td>
<td>-1.78</td>
<td>-2.36</td>
<td>-1.57</td>
<td>0.02</td>
<td>-0.66</td>
<td>2.59</td>
<td>1.57</td>
<td>2.36</td>
<td>1.80</td>
<td>0.00</td>
</tr>
<tr>
<td>31</td>
<td>-0.08</td>
<td>-0.05</td>
<td>1.00</td>
<td>0.00</td>
<td>-1.24</td>
<td>-0.26</td>
<td>-0.27</td>
<td>1.00</td>
<td>0.00</td>
<td>-0.01</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 3.1: Relations between the segments of scene 3.15
3.3. Relations between segments

Figure 3.16: Ideal parallelism between $S_a$ and $S_b$ (a) is attributed with a high certainty factor, whereas situation (b) is considered as a distorted parallelism with a smaller certainty factor.

conditions in the following way:

$$c_p = (c_{S_a} c_{S_b}) \left(1 - \frac{|l_{S_a} - l_{S_b}|}{\max(l_{S_a}, l_{S_b})}\right) \left(1 - \frac{|\theta_a - \theta_b|}{\theta_i^p}\right) \left(1 - \frac{\alpha}{\alpha_t}\right)$$ (3.17)

- **collinearity**: Two segments $S_a$ and $S_b$ are collinear if their direction angles $\theta_a$ and $\theta_b$ are similar and if the connecting line $C_{ab}$ between $S_a$ and $S_b$ has also that same direction. A fixed angle threshold $\theta_i^l$ is used to compare the directions. If no difference between direction angles is larger than the threshold, $S_a$ and $S_b$ are said to be collinear. Figure 3.17 illustrates this definition. The certainty factor $c_l$ is composed of the product of the certainty factors $c_{S_a}$ and $c_{S_b}$ of the two segments $\theta_a$ and $\theta_b$ as well as of a term which is proportional to the maximum angle difference. Its range is also [0,1] with higher values expressing better collinearity.

$$c_l = c_{S_a} c_{S_b} \left(1 - \max(|\theta_a - \theta_b|, |\theta_a - \theta_{ab}|, |\theta_b - \theta_{ab}|) \frac{\theta_i^l}{\theta_i^l}\right)$$ (3.18)

- **concentricity**: $S_a$ and $S_b$ have to be represented by an arc and their center points have to be "approximately" at the same place. We have to
use such fuzzy descriptions again because of the possibility of noise and errors. In the current implementation we use the center points $x_a, y_a$ and $x_b, y_b$ to calculate their distance $d_{ab}$

$$d_{ab} = \sqrt{\frac{(x_a - x_b)^2 + (y_a - y_b)^2}{r_a}} < r_t$$  \hspace{1cm} (3.19)$$

which has to be smaller than the fix threshold $r_t$. The certainty factor $c^e$ is again composed of the product of the two segments certainty factors $c_{S_a}$ and $c_{S_b}$ and of a term which is proportional to $d_{ab}$.

$$c^e = c_{S_a} c_{S_b} \left(1 - \frac{d_{ab}}{r_t}\right)$$  \hspace{1cm} (3.20)$$

- **symmetry**: Since this relation is more complex, and interesting from a more general point of view, the methods of finding symmetries are explained in the separate section 3.4

Applying the defined relation to the example contour from figure 3.15 we get the relations between segments as shown in figure 3.18.

### 3.3.1 Results

The output of the described algorithm is a description of the contour in the form of an attributed graph. Nodes represent segments and arcs represent geometric relations between these segments. Arrows as well as nodes are described through

\(^2\)symmetry relations are treated seperately and are not shown in this graph
3.3. Relations between segments

<table>
<thead>
<tr>
<th>$S_a$</th>
<th>$S_b$</th>
<th>cf</th>
<th>relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0.014</td>
<td>parallel</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>0.184</td>
<td>collinear</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>0.233</td>
<td>parallel</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>0.018</td>
<td>parallel</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>0.181</td>
<td>concentric</td>
</tr>
</tbody>
</table>

Figure 3.18: Graph representation of a simple example contour (fig. 3.15)

attributes like type of fit, goodness of fit, length, associated contour points etc. All graph elements have associated with them a certainty factor as a measure of belief. The algorithm was tested in a number of examples from a real world application [APR+92], a vision system for manipulating objects like cups, plates, forks etc. on a cafeteria tray. Figure 3.13 shows the segmented contour of an example from that application and figure 3.12 a part of it (more examples can be found in the appendix in figure A.1, A.2 and A.3). As we can see, the segmentation is meaningful in the sense that round parts of the contour lead to approximation with circles. The handle of the fork for example is very nicely represented by two almost concentric arcs and the round bottom part of the soup plate gets two almost symmetric arcs.

3.3.2 Conclusion

A split-and-merge algorithm for finding significant points and segments in digital curves was presented. In this scheme corner points and inflection points are described through small regions on the contour, the so called separator

3 the relation graph for figure 3.13 is in the appendix in figure A.4
domains. Between these domains are linearly or circularly fitted segments. Certainty factor are assigned to the segments as well as to the separator domains. The algorithm was tested on example contours of silhouette images from a real world application. Some results were discussed in this section.

3.4 Symmetries in contours

Many objects in the real world as well as in man-made worlds are symmetric or have at least partially symmetric properties. Symmetries can play a central role in image understanding systems dealing with objects of that kind. In multiple object scenes they can help to recover occluded parts of shapes. For image compression they are important since only one part of the symmetric objects has to be represented. As experiments show, they play a key role in image understanding by the human visual system. For a technical vision system this means, that for recognition it can be a central issue to identify and localize symmetries. In the “world” of cafeteria trays for example, many objects have rotational symmetries (cup, glass, soup bowl, plate, soup plate, dessert bowl) or can be approximated in 3-D by generalized cylinders or in 2-D by ribbons (cutlery). These symmetries are conspicuous in silhouettes even when the objects are partially occluded or in tilted positions (fig. 3.21). The following sections describe methods to find these symmetries.

3.4.1 Methods working on coordinate points

The most obvious way to compute symmetries is to directly define relations between contour points and points on the symmetry axis. A very sound way to do so is the “Symmetry Axis Transform, SAT” [BN78] and even more its derivate, the “Smoothed Local Symmetry, SLS” by Brady and Asada [BA84].

Points of a SAT symmetry axis are the centers of a maximal circle that touches at least two points of the bounding contour. The union of all those centers form the symmetry axis, which is piecewise smooth as was shown by Bookstein [Boo79]. Problems with the SAT arise at branching points where the maximal circle touches the contour in more than two points. The generated axis splits there and it is not guaranteed, that all global axes of a shape are found. Figure 3.19a illustrates that for a rectangle.
3.4. Symmetries in contours

Brady's SLS overcomes these problems with a different condition which points on the symmetry axis have to meet (fig. 3.20). Let $P_a$ and $P_b$ be two points on the contour and $n_a$ and $n_b$ the normals to the contour in these points. Then the point $P_{sls}$ in the middle of $P_a - P_b$ is called a local symmetry point if the angle $n_a \angle (P_a - P_{sls})$ and $n_b \angle (P_b - P_{sls})$ are equal. The relation of Brady's condition to the SAT is also obvious in figure 3.20. Whereas the SLS point lies in the middle of the secant of the maximal circle touching in $P_a$ and $P_b$, the SAT point lies in the center of the same circle. In many situations this does not make a big difference but in discontinuities on the contour where the SAT fails, the SLS definition is able to produce the missing axis points. In the rectangle example, all local and global axes can therefore be found (fig. 3.19b).
this would mean that $O(n^2)$ comparisons have to be made, which is quite expensive in terms of computation. An implementation of the SLS axes has to take that into account, otherwise it won't be of practical use. The next section, for that reason, presents an alternative approach that uses the SLS definition only for the verification of symmetry axes.

3.4.2 Methods working on $\theta(s)$

Instead of processing the plain coordinate values of a contour, $\theta(s)$ the arclength versus the direction angle representation of the contour as defined in section 3.2.4 can be used. Symmetries relative to an axis appear in that representation as symmetries relative to a point in $\theta(s)$. It's interesting to note that a rotation of the contour means a vertical shift of the $\theta(s)$-function whereas a translation has no effect on $\theta(s)$. Figure 3.21 shows these properties with synthetic examples, axis-symmetric segments of the contour correspond to point-mirrored parts in the $\theta(s)$-function.

![Figure 3.21: Contours and the corresponding $\theta(s)$ function. A rotation of the contour means a vertical shift in $\theta(s)$](image)

A symmetry detection algorithm can take advantage of these properties. Instead of detecting a symmetry relative to an axis in the contour it can look for a mirror point of two segments in the $\theta(s)$-representation (The $\theta$-value of
the mirror point is 90 degrees off the direction of the symmetry axis). An algorithm based on that idea is the following one:

1. Find the significant points of the contour \( C \). These points partition \( C \) into \( M \) "meaningful" segments \( c_m \).

2. Identify the segments \( c_m \) in the \( \theta(s) \) representation which results in segments \( \theta_m(s) \) with length \( l_m \).

3. For each \( m \) in \( M \) do
   
   (a) Point-mirror the segment \( \theta_m(s) \) at the origin and name the so obtained segment \( \tilde{\theta}_m(s) \)
   
   (b) Compute the distance \( \sigma_m \) (3.21) between \( \theta(s) \) and \( \tilde{\theta}_m(s) \)
   
   (c) The position \( p \) where \( \sigma_m \) is minimal defines the starting point of the symmetric section to \( \tilde{\theta}_m(s) \). The symmetry point lies in between and the axis lies in between the corresponding contour segments.
   
   (d) Compute the points of the symmetry axis with the SLS criterion between the points of \( \tilde{\theta}_m(s) \) and the \( l_m \) points after position \( p \) on the contour.

Given the current point-mirrored segment \( \tilde{\theta}_m(s) \) and the contour part of the \( \theta(s) \)-function to be searched, we define the distance measure \( \sigma_m(s) \) with equation 3.21.

\[
\sigma_m(s) = \sum_{i=s}^{i=s+l_m-1} \left| (\tilde{\theta}_m(s - i) - \overline{\tilde{\theta}_m}) - (\theta(i) - \overline{\theta}) \right| \tag{3.21}
\]

Figure 3.22 illustrates the evaluation of \( \sigma_m \). The segment \( \tilde{\theta}_m(s) \) is moved along \( \theta(s) \) by shifting it by its mean value \( \overline{\theta(s)} \). This is to compensate for a rotation of the contour. The absolute value of the difference is summed over the comparison interval. If \( \tilde{\theta}_m(s) \) is exactly point symmetric to \( \theta(s) \) then the sum will be zero. For that reason, no normalization of \( \sigma_m(s) \) is necessary. Only the minimum has to be found. The distance measure \( \sigma_m \) is a key issue of the symmetry detection algorithm. To detect the axes, reliable matches of the segments \( \theta_m \) have to be found. This again is only possible, if the
segments show enough variation. Usually this is the case when curvy contours are analyzed. Unfortunately the contours we are working with do not always have this property. Mostly they can be very well approximated by straight lines or circular arcs. Their $\theta(s)$ function therefore is composed only of linear segments which is not ideal for the proposed algorithm. To overcome that problem, two heuristics are invoked: A match of the current point-mirrored segment $\tilde{\theta}(s)$ against itself has to be prevented and second, it is advantageous to do the match always just in a neighborhood of the point mirrored segment and not on the whole contour. The neighborhood for example can be defined as a part on the contour lying between two concave corners. As we will see in section 4.2.5 such a neighborhood definition is necessary anyway when contours of multiple objects are treated.

In principle, it would be possible to use the $\theta$-value of the mirror point to determine the direction of the symmetry axis and subsequently find the contour points that correspond to each other and the axis point that belongs to them. But the match in the $\theta(s)$-representation might not be exact. Therefore we prefer to verify the symmetry in the original contour. In the current implementation, Brady's smoothed local symmetry criterion (SLS) [BA84] is used for that. Through that, the symmetry analysis algorithm can be seen as an enhanced two step SLS algorithm. The $\theta(s)$ correlation generates hypothetical axis points and the SLS condition is used to select the right ones. This preselection reduces the number of points that have to be checked with the SLS criterion dramatically (fig. 3.23). Beside that, the preselection prevents
3.4. Symmetries in contours

![Figure 3.23: Brute force SLS algorithm vs. our implementation with selection of segments prior to SLS. Instead of testing each pair of contour points, only a small selection is tested.](image)

the SLS of generating all local and global axes, which typically have to be ordered according their importance in a post-processing step anyway.

In [API91] it was shown how this symmetry detection algorithm with its heuristics can be used to derive 3-D gripping information for a robot from silhouettes. Figure 3.24 shows a segmented contour with the corresponding segments on $\theta(s)$. The solid part of the contour is the analyzed neighborhood selected with the said heuristics and the found symmetry axes are marked. Although good results were produced with the presented algorithm with typical contours from one specific application, the used heuristics mean a loss of generality. It also looks promising to incorporate more of the results of the segmentation algorithm from section 3.2. The attributed graph of contour segments is a powerful data structures very well suited for symmetry analysis. Properties like direction angle or segment type can be used for a structured match between segments without accessing individual contour points anymore. The following section explains, how this can be done.

3.4.3 Structural matching for symmetry detection

Similarly to the previous section where conditions for symmetry between contour points were stated, rules can be formulated for symmetries between structural elements like straight lines or circular arcs as obtained from the segmentation algorithm from section 3.2. The advantage of such a procedure is, that less data has to be checked since the conditions are stated for more
complex entities. On the other hand, it can also mean a loss of precision, because the segments are an approximation of the original data points.

To find symmetry axes in a segmented contour, it is necessary to define small axis pieces, so called "microsymmetries" ([Dav77]) between pairs of segments. They exist between straight line segments as well as between circular arc segments. Each microsymmetry represents a part of an axis hypothesis. Its importance is weighted with a certainty factor which is built on the certainty factors of the generating segments as obtained from the segmentation. These weights help in clustering the local axes to global ones. The symmetry axis hypotheses are found by clustering the microsymmetries. From the strongest clusters a verification based on conditions between adjacent segments finally determines the symmetry axes.

Between two line segments $S_i$ and $S_j$ the following microsymmetries are defined:
3.4. Symmetries in contours

1. if \( S_i \) and \( S_j \) are neighbors, i.e. \(|i - j| = 1\) then two hypotheses are generated:
   
   (a) the angle bisector (fig. 3.26a)
   
   (b) the perpendicular bisector of the line connecting the two end points of \( S_i \) and \( S_j \) (fig 3.26b)

2. if \(|i - j| > 1\) then the following additional hypotheses are generated\(^4\):
   
   (a) the perpendicular bisector of the line connecting the two start points of \( S_i \) and \( S_j \)
   
   (b) the perpendicular bisector of the line connecting start points of \( S_i \) with end point of \( S_j \)
   
   (c) the perpendicular bisector of the line connecting end points of \( S_i \) with start point of \( S_j \)

\[\text{Figure 3.25: Local axis hypothesis (microsymmetries) on a synthetic shape}\]

Figure 3.25 shows all possible local axes for a synthetic example. As we can see, besides the ones that we as a human consider as "natural" axes others that can not be aggregated into global axes exist. To sort out the false ones and to find the strong ones, a grouping procedure has to follow the microsymmetry generation step. To facilitate that step each microsymmetry is weighted with a certainty factor. The idea is, that more secure segments should generate

---

\(^4\)In this case Davis only generates 4 hypothesis, he doesn't use the bisector between \( S_i \) and \( S_j \)
Figure 3.26: For perfect symmetry (a), the angle $\phi$ is 0 which maximizes the certainty factor. For disturbed symmetry (b), the certainty factor decreases proportional to $\cos(\phi)$

more secure microsymmetries. If segment $S_i$ and $S_j$ with their associated certainty factors $c_{f_i}$ and $c_{f_j}$ form a common microsymmetry $m_{ij}$, then $c_{f_{ij}}$, the certainty factor of $m_{ij}$ is defined as

$$c_{f_{ij}} = c_{f_i}c_{f_j}\cos(\phi)$$ (3.22)

The definition of the angle $\phi$ is obvious from figure 3.26. This cf assignment gives those microsymmetries a high rating that are generated from well approximated segments ($c_{f_i}$ and $c_{f_j}$ are high) and when $\phi$ is small ($\phi = 0$ means perfect symmetry). After generating all local symmetry axis hypotheses and computing their certainty factors, they have to be aggregated. The idea is, that global axes are supported by many local axes so that clusters of local axes can
3.4. Symmetries in contours

Figure 3.27: Local axis hypotheses (microsymmetries)(a) and the selected one (b)

be used to detect the global ones. To find these clusters, the Hough transform [DH72] is well suited. For that, all axis hypotheses have to be represented in the Hesse normal form [Hes65] by the two parameter $\rho$, the distance of the origin and $\theta$, the direction angle (equation 3.23).

$$x \cos \theta + y \sin \theta - \rho = 0$$

(3.23)

Each microsymmetry $m_i$ is represented in that form by a pair $\langle \rho_i, \theta_i \rangle$ in the $\rho-\theta$ space. Speaking in Hough transform terminology, this means that there is one vote for the pair $\langle \rho_i, \theta_i \rangle$. Additionally, this vote is weighted with the certainty factor $c_{f,m}$ of $m_i$. To find maxima in the $\rho - \theta$ space, it is partitioned into discrete buckets. The ones with the most votes in are then selected and their $\langle \rho - \theta \rangle$ pairs identify the corresponding symmetry axes.

In figure 3.27 an example with a real contour of a soup bowl is shown. In 3.27a all detected microsymmetries are drawn with a constant length. It's clearly visible that there are microsymmetries in the corners for example, but the great majority falls more or less to the central part of the bowl. The clustering algorithm then selects one axis in the central part as the symmetry
axis. That it is not perfectly straight up as it should be has to do with the partitioning of the $\rho - \theta$ space for clustering. The bucket can not be chosen too small because then no clear maxima would be found. But a larger bucket size means less resolution to determine $\rho$ and $\theta$.

3.4.4 Conclusion of symmetry axis computation

This section explored how symmetry axes can be derived from contours. Traditional methods like the pure raster oriented “Symmetry Axis Transform, SAT” or its derivate the “Smoothed Local Symmetry” axes are general but lack efficiency and/or generate axes in a rather unstructured way. This means that important axes have to be found with a post-processing grouping, which often is model driven so that generality is lost.

The method proposed in this section is based on the fact that axis symmetries in the contour correspond to point symmetries in the $\theta(s)$-function. These symmetries are easier to detect because a one dimensional matching can be used to find the correspondences. The algorithm proposed has two stages. In the first one, symmetry axis hypotheses based on the $\theta(s)$ matching are computed and the second one is a verification with the help of Brady’s SLS criterion. The algorithm works well for curvy contours. However, if contours are not varying enough, for example if they are composed only of straight lines, false matches may result. Heuristics are used to overcome that difficulty. A more fundamental problem of the algorithm is that the contour has to be segmented first for doing the matching. Depending on the algorithm for that, different dominant points may be found which can result in different symmetry axes. However, an oversegmentation usually prevents that since smaller segments are easier to match. But to completely overcome that problem, it would be better to do the match in the curvature representation instead of the $\theta(s)$, similar to the method proposed by Wolfson in [Wol90].

As an alternative, a second method based on symmetry conditions between segments of the contour was presented too. Axes are computed by generating small pieces of possible axes for each pair of contour segments, so called microsymmetries. By clustering the microsymmetries to major clusters, the global axes are identified. It was shown how the method works with simple contours. Clustering is a problem, since a trade-off between precision and completeness has to be found. If too many clusters are allowed, the resulting axes are precise but many false axes are generated too. For more complex
3.4. Symmetries in contours

contours, a post-processing grouping step may be necessary because of that, by which the attractiveness of that method is reduced. This is even more so because as with the first approach, the contour has to be segmented too for the symmetry analysis.
Leer - Vide - Empty
Object recognition

In machine vision two major areas of research and development interests can be identified: In one of them one tries to recover unknown objects in unstructured environments whereas the other deals with known objects in simply structured environments. Analyzing outdoor scenes and guiding autonomous vehicles are typical applications for the first field. Results are still poor and solutions lack generality and performance. More results are achieved in the second research area where some applications in robot vision are already in use in industrial environments. Two important object recognition paradigms are in use. In one of it, scene and models are represented by attributed graphs and recognition is done with subgraph matching with various search strategies and constraints for pruning. In section 4.1 this technique is applied to segmented contours obtained as described in chapter 3. In the other technique, matching is done by looking for local features which are used in the condition part of rules. Fulfilled rules evaluate to an array of evidence weights, each of it representing the amount of evidence the condition votes for or against a particular object. All resulting evidences are accumulated without introducing bias at an early stage. Recognition is done by selecting the object with most positive evidence. Section 4.2 explains that technique and shows how to use it for the object recognition from contours formed by several objects.
4.1 Recognition based on graph matching

A very sound and common paradigm in machine vision is to view the recognition problem as a search problem: sensor data has to be matched with corresponding object model data. As instrumentarium to solve the search problem typically graph matching algorithms are used. Unfortunately they are NP-complete which makes them impractical for realistic applications. To overcome this, typically, various constraints are applied to restrict the search space. A sound and up-to-date description of these techniques can be found in W.E.L. Grimson's book “Object Recognition by Computer” [Gri90]. In the following sections it is outlined how this technique could be used for recognizing objects in contours.

4.1.1 Model representation

Model data is obtained with the segmentation of contours as described in chapter 3. To build the model data base, each object is presented to the system, its contour is segmented and the relations between the segments are calculated. This implicitly creates a graph the nodes of which are segments of the contour and the arcs of which are segment relations. Two types of nodes, linear segment and circular arc segment, have been considered. Common attributes of both are:

- segment type. Either linear or circular arc is possible.
- center of gravity. \( x_c \) and \( y_c \) in image coordinates
- length in pixels of the segment.
- certainty factor as computed during segmentation

Additional attributes of the linear segment are:

- direction angle of the linear segment

Additional attributes of the circular arc are:

- center of the circle in image coordinates
4.1. Recognition based on graph matching

- radius of the circle in image coordinates
- start angle of the circular arc
- end angle of the circular arc

Objects which are not rotationally symmetric show different contours depending on the viewpoint. The handle of a soup bowl or a cup for example is visible only under certain viewing angles. For these cases it is necessary to build model graphs for each view that shows essentially different object properties.

4.1.2 Scene representation

Scenes are segmented in the same way as model objects are. Therefore from the segmentation we obtain an attributed graph as scene description as before.

4.1.3 Matching

The matcher has to solve two problems: It has to find the largest subgraph in the scene graph that maps to a part of a model graph and it has to find the geometric transformation between the model centered coordinates and the scene coordinates. Both tasks are closely interrelated and it is hard to do one without the other. A naïve way to solve the problem is to combine all $M$ model nodes with all $S$ scene nodes and match those who have similar attributes, which leads (with $M \geq S$) to

$$\frac{M!}{(M - S)!}$$

match tries. Even for small $M$ and $S$ this is far too much and very inefficient. For a practically usable algorithm one must therefore cut down the number of tries by leaving out all combinations that are obviously incompatible. This decision is made with the precomputed attributes of the nodes and arcs. There are

- unary constraints in which attributes of model nodes have to correspond to attributes of the scene nodes
• binary constraints in which relations between matching pairs of scene
nodes and model nodes have to correspond to already matched pairs.

Applying these two constraints in the match algorithm leads to a list of
matched node pairs in every leaf of the search tree. This list corresponds to
an interpretation of the scene. But even when the constraints are tight, this
interpretation has to be considered carefully. Wrong matches may be in the
list because of limited accuracy or noise. A scene of a mirrored model object
for example would pass with no problems both constraints but it would be the
wrong scene interpretation. The interpretation therefore can only be taken as
a hypothesis that has to be verified. For that, the geometric transformation
between model and scene is computed and used as a third, most expensive but
also most restricting constraint. It is also an important consideration for the
design of the matching algorithm that the segmentation may not be unique.
Even when the same object is presented to the system twice in the same
position, the two graphs obtained from the segmentation can have different
numbers of nodes. It can therefore not be assumed that all model nodes of a
model will match to the scene or part of it. This difficulty is handled with a
prescribed amount of allowed mismatches.

Another problem for the matcher are scenes with multiple objects. A
situation like that is detected if the matcher finds a reliable match but there are
still a lot of unmatched nodes in the scene graph. In that case, it is necessary
to make a hypothesis about the number of objects in the scene and split the
scene graph of the unmatched nodes. This is typically done by using some
heuristics.

Fan et al. [FMN88] demonstrated how this can be done with range data
segmented into surface patches. Using a similar scheme for recognizing objects
in contours, we define the following constraints for the matching algorithm.

**Unary constraint**

For unary constraints a similarity measure $\nu$ between a model node $m$ and a
scene node $s$ is computed. To do so, $m$ and $s$ have to be of the same type,
linear or circular arc. If this is the case, $\nu$ is computed and if it is smaller
than a prescribed value $\nu_u$, $m$ and $s$ are said to be compatible. The inexact
comparison is necessary because there are noise and segmentation errors so that one can not expect to find exactly the same attributes in the scene as in the models. $v$ is computed with the help of a normalized difference measure $\xi$ so that its values lie in $[0, 1]$. For two scalars $p$ and $q$ this measure $\xi(p, q)$ is defined in equation 4.2. If $p$ and $q$ are identical, then the maximal similarity of $\xi(p, q) = 0$ is obtained. If the differences between $p$ and $q$ get larger, $\xi(p, q)$ gets larger too. A value of $\xi(p, q) = 1$ means maximal dissimilarity.

$$\xi(p, q) = \frac{|p - q|}{\max(p, q)} \quad (4.2)$$

For linear segments $\nu^{lin}$ is obtained by computing $\xi(l_m, l_s)$ for the lengths $l_m$ and $l_s$ of the two segments. The unary constraint for linear segments is therefore

$$\nu^{lin} = \frac{|l_m - l_s|}{\max(l_m, l_s)} < \nu^{lin} \quad (4.3)$$

where $\nu^{lin}$ is a predetermined threshold.

For the circular arc segment, there are more attributes used to compute $\nu^{arc}$. These are

- $\xi_l$ between the length $l_m$ and $l_s$
- $\xi_r$ between the radii $r_m$ and $r_s$
- $\xi_a$ between the opening angles $a_m$ and $a_s$.

The overall similarity $\nu^{arc}$ is computed by a weighted sum of the three normalized measures $\xi_l, \xi_r$ and $\xi_a$ and their weights $w_l, w_r$ and $w_a$

$$\nu^{arc} = \frac{w_l \xi(l_m, l_s) + w_r \xi(r_m, r_s) + w_a \xi(a_m, a_s)}{w_l + w_r + w_a} \quad (4.4)$$

and the unary constraint is satisfied for the circular arc if

$$\nu^{arc} < \nu^{arc} \quad (4.5)$$
with $\nu_1^{urec}$ being a predetermined threshold.

**Binary constraint**

Binary constraints are invoked every time a pair of nodes $p_i = (m_i, s_i)$ has passed the unary constraint. $p_i$ is compared to all $N$ already matched pairs $(m_j, s_j)$ and if there is no compatibility $(m_i, s_i)$ is discarded. Binary tests are:

- **Distance consistency:** The distance between segments is not changed by rotation or translation in the image plane. It is therefore independent of the unknown coordinate transformation between scene and image. Yet it is not invariant to partial occlusion. But occlusion has a drastic influence only in those situations where the two segments are almost collinear. Since these are rare in our models, the distance can be used very effectively for filtering out partial matches. If $d_m$ and $d_s$ are the distances between the center of gravity of $(m_i, m_j)$ and $(s_i, s_j)$, then they are said to be compatible if
  \[ \xi(d_m, d_s) = \frac{|d_m - d_s|}{\max(d_m, d_s)} < d_t \]  
  (4.6)
  with $d_t$ being a constant threshold.

- **Direction consistency:** The direction angle between linear segments is also invariant to rotation and translation. If $\theta_1$ is the angle between the two model nodes $(m_i, m_j)$ and $\theta_2$ the angle between the corresponding scene nodes $(s_i, s_j)$ then the pair $(m_i, s_i)$ is said to be compatible if
  \[ |\theta_1 - \theta_2| < \theta_t \]  
  (4.7)
  with $\theta_t$ being a constant threshold.

- **Relation consistency:** As Lowe ([Low87]) already mentioned, the relations "collinear", "parallel" and "symmetric" are viewpoint independent and therefore easily usable to check the compatibility between two segments. The relation consistency constraint states, that a new pair $p_i = (m_i, s_i)$ has to correspond to the relations of the already matched pairs. If $r_1$ is the relation between $(m_i, m_j)$ and $r_2$ the relation between $(s_i, s_j)$ then $p_i$ is compatible if
4.1. Recognition based on graph matching

- \( r_1 \) is equal to \( r_2 \)
- either \( r_1 \) or \( r_2 \) or both of them is NULL (there is no relation between \( r_1 \) and \( r_2 \))

• **Uniqueness:** Each model node can match only one scene node. A new pair \((m_i, s_i)\) is therefore only consistent with all \( N \) already matched pairs if \( m_i \neq m_j \) and \( s_i \neq s_j \) for \( j \in [1, N] \)

Transformation constraint

The geometric transformation between the matched model and scene nodes is used to verify the matched nodes that passed the unary and the binary constraint. To do so, the transformation has to be estimated first. With that estimation, all model nodes are transformed to the image plane and compared with the scene nodes. If the difference is small enough, the match is accepted.

The transformation consists of a translation vector \( \vec{T} = (T_x, T_y) \) and a rotation angle \( \alpha \). It is fully determined by two corresponding model/scene node pairs. The translation is the difference of the centroid of gravity of the two model nodes and the corresponding rotated scene nodes. To compute the rotation, let \( \vec{a} \) be the center of model node \( m_a \) and \( \vec{b} \) be the center of model node \( m_b \). Correspondingly, are \( \vec{u} \) and \( \vec{v} \) are the centers of the corresponding scene nodes \( s_u \) and \( s_v \). With the help of \( \vec{c} = \vec{a} - \vec{b} \) and \( \vec{w} = \vec{u} - \vec{v} \) the cosine of the rotation angle \( \alpha \) is defined as

\[
\cos(\alpha) = \frac{|\vec{w} \cdot \vec{c}|}{\sqrt{|\vec{w} \cdot \vec{c}|^2 + |\vec{w} \times \vec{c}|^2}} \tag{4.8}
\]

and the sine of \( \alpha \) as

\[
\sin(\alpha) = \frac{|\vec{w} \times \vec{c}|}{\sqrt{|\vec{w} \cdot \vec{c}|^2 + |\vec{w} \times \vec{c}|^2}} \tag{4.9}
\]

A match typically consists of more than just two pairs of nodes. To overcome measurement errors it is therefore advantageous to use all of them to compute the transformation. This can be done with a least-squares approach to all pairs. The error function to be minimized is
\[ E = \sum_{i=1}^{N} \left( x_i - (\cos(\alpha)u_i - \sin(\alpha)v_i + T_x) \right)^2 + \left( y_i - (\sin(\alpha)u_i + \cos(\alpha)v_i + T_y) \right)^2 \quad (4.10) \]

Usually, 4.10 is written with the origin of the centroid shifted by the average of each of the coordinates. With \( x'_i = x_i - \bar{x}, y'_i = y_i - \bar{y}, u'_i = u_i - \bar{u}, v'_i = v_i - \bar{v} \) and

\[
T_x' = T_x - (\bar{x} - (\cos(\alpha)\bar{u} - \sin(\alpha)\bar{v})) \\
T_y' = T_y - (\bar{y} - (\sin(\alpha)\bar{u} + \cos(\alpha)\bar{v})) \quad (4.11)
\]

the sum of the squared errors can be written as

\[ E = \sum_{i=1}^{N} \left( x'_i - (\cos(\alpha)u'_i - \sin(\alpha)v'_i + T'_x) \right)^2 + \left( y'_i - (\sin(\alpha)u'_i + \cos(\alpha)v'_i + T'_y) \right)^2 \quad (4.12) \]

This sum minimizes when \( T'_x = 0 \) and \( T'_y = 0 \) so we obtain for the translation vector \( \overrightarrow{T} = (T_x, T_y) \)

\[
T_x = \bar{x} - (\cos(\alpha)\bar{u} - \sin(\alpha)\bar{v}) \\
T_y = \bar{y} - (\sin(\alpha)\bar{u} + \cos(\alpha)\bar{v}) \quad (4.13)
\]

With that the expression for \( E \) can be simplified to

\[ E = \sum_{i=1}^{N} \left( x'_i - (\cos(\alpha)u'_i - \sin(\alpha)v'_i) \right)^2 + \left( y'_i - (\sin(\alpha)u'_i + \cos(\alpha)v'_i) \right)^2 \quad (4.14) \]

or with the substitutions

\[
A = \sum_{i}^{N} (x_i^2 + y_i^2) \text{ and } B = \sum_{i}^{N} (u_i^2 + v_i^2) \\
C = \sum_{i}^{N} (x_i'u'_i + y_i'v'_i) \text{ and } S = \sum_{i}^{N} (y_i'u'_i - x_i'v'_i) \quad (4.15)
\]
the least squares sum is

\[ E = A - 2(\cos(\alpha)C + \sin(\alpha)S) + B \]  \hspace{1cm} (4.16)

Its extrema occur when \( \cos(\alpha)C = \sin(\alpha)S \) and since \( \sin(\alpha)^2 + \cos(\alpha)^2 = 1 \) we find at the minima of the least squares sum the cosine and the sine of the rotations angle \( \alpha \) as

\[ \cos(\alpha) = \frac{C}{\sqrt{C^2 + S^2}} \quad \text{and} \quad \sin(\alpha) = \frac{S}{\sqrt{C^2 + S^2}} \]  \hspace{1cm} (4.17)

The least squares method to recover the transformation can also be used to obtain a measure of goodness of fit. At the minimum of equation 4.16 is \( \cos(\alpha)C = \sin(\alpha)S \), so that \( E \) can be written

\[ E = A - 2\sqrt{C^2 + S^2} + B \]  \hspace{1cm} (4.18)

and the root-mean-square error

\[ q = \sqrt{E/N} \]  \hspace{1cm} (4.19)

is a measure of quality of the fit. A transformation is accepted if \( q \) is smaller than a prescribed threshold \( q_t \).

**Multiple object scenes**

The described matching algorithm is designed only to find a correspondence between the model graphs and the graph of a single object scene. If there are multiple mutually occluding objects in a scene, matching may not be done anymore reliably and uniquely. Because of the occlusion the unary constraint loses power since segment attributes like length may be changed. Between segments of different objects, false relations can be detected and disturb the binary relations. To overcome these problems, the matcher tries to split the graph of a multiple object scene into individual parts. It does this, if no match is found for the whole scene. The scene graph is cut apart between neighboring segments that from a concave corner. This scheme follows the
widely used heuristics in range data analysis, where scenes are split between surface patches forming a concave angle. The so obtained subgraphs are treated as single object scenes and analyzed by the matcher. Graphs consisting just of one node usually belong to the background and therefore are discarded.

Valuation of a match

To each match that passed all constraints a measure of goodness $G$ is attached. It is defined as

$$G = \frac{\max(N_m, N_s) + (1 - \bar{v})}{2}$$

with $N_m$ being the ratio between the number of matched model nodes and the number of total model nodes and $N_s$ is the ratio between the number of matched scene nodes and the number of total scene nodes. $\bar{v}$ is the average of all similarities $v_i$ computed by the unary constraint. The value of $G$ lies between 0 and 1 with higher values meaning a better match. Recognition is done by selecting the match with the highest value of $G$.

4.1.4 Graph-matching-based recognition with silhouettes

The described matching algorithm with its three constraints was applied to the problem of recognizing dishes in silhouettes. As model objects, the following pieces are chosen:

1. a cup
2. a drinking glass
3. a soup bowl
4. a salad bowl
5. a plate
6. a soup plate
7. a dessert bowl
With these objects, a model database is built. To do so, each object is presented to the system in its normal position. Its silhouette is segmented with the method described in 3 and the obtained segment relation graph is stored in the model data base. With the exception of the soup bowl and the cup, all objects are rotation symmetric so that it is sufficient to store just one view. The cup and the soup bowl have handles that are visible in the contour only under certain view points. For these two objects, at least two views have to be taken. The complete model base consists of 9 model views. It is shown in figure 4.1.
Figure 4.1: Object models for recognition with graph matching
4.1. Recognition based on graph matching

As we can see with the symmetric objects in these figures, the segmentation is not always perfect and in some cases generates asymmetric results. This happens mainly because the objects stand on a tray which is not completely leveled. For that reason their silhouettes are slightly slanted which can cause the segmentation algorithm to generate asymmetric results. Of course it would be possible at least to build the models from objects placed perfectly on the tray. But it would be hard for a realistic application to fulfill this condition all the time. Situations like that should be allowed and it is the matcher’s responsibility to handle it correctly.

Figure 4.2: A simple scene for the matcher

A simple input scene for the matcher is shown in figure 4.2. Even when subgraphs of the scene graph would match well to certain model graphs, no overall match is obtained by the matcher. This happens because there is always a large number of unmatched nodes, which is not allowed. To solve this problem, the scene is split at concave corners, which results in two graphs and three degenerated graphs with just one node each. The latter are discarded because a valid object hypothesis graph must have at least 2 nodes. The other two graphs match with the model “Glass” (fig. 4.3) and “Cup view 1” (fig. 4.4).
Glass
$G = 0.95$

<table>
<thead>
<tr>
<th>$s_i$</th>
<th>$m_i$</th>
<th>$\nu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.251</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>.108</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>.000</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>.022</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>.116</td>
</tr>
</tbody>
</table>

Cup (v 1)
$G = 0.77$

<table>
<thead>
<tr>
<th>$s_i$</th>
<th>$m_i$</th>
<th>$\nu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>.177</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>.060</td>
</tr>
</tbody>
</table>

Figure 4.3: Found match to the "glass" hypothesis

Cup (v 1)
$G = 0.98$

<table>
<thead>
<tr>
<th>$s_i$</th>
<th>$m_i$</th>
<th>$\nu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>.019</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>.000</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>.057</td>
</tr>
</tbody>
</table>

Glass
$G = 0.65$

<table>
<thead>
<tr>
<th>$s_i$</th>
<th>$m_i$</th>
<th>$\nu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
<td>.133</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>.060</td>
</tr>
</tbody>
</table>

Figure 4.4: Found match to the "cup" hypothesis
A more realistic scene with substantial occlusion is shown in figure 4.5. Applying the matcher with its constraints to that graph results in no found correspondence to a model object. The graph therefore is split into the three subgraphs as in figure 4.6 and figure 4.7. These are the initial object hypotheses. Again, the single noded graphs which are the linear segments of the tray to both sides of the scene are discarded. With these three subgraphs, the matcher is able to find matches. The first subgraph matches to the models “Glass”, “Salad Bowl” and “Cup”, with “Glass” being the best match as figure 4.6 shows. That match consists of 3 nodes with the best matching score. The matches with the “Salad Bowl” model and the “Cup” model are weaker because only two nodes match for each of it.

For the other two subgraphs, part of the salad bowl and part of the cup (fig. 4.7), no match is found. In the case of the salad bowl, this happens because both segments seen in the scene are occluded too much. The unary constraint which compares the length of the segments therefore rejects a match. For the cup, no match is found because the number of unmatched nodes is too high for each possible model.
Table 4.6: Matches found with the first object hypothesis

Table 4.7: Split object hypothesis with no match found
4.1. Recognition based on graph matching

4.1.5 Conclusion of graph matching based recognition

Object recognition based on graph matching is well established for range data. For that domain, a main problem is still the slow execution time of these algorithms because of complex 3-D data structures. A way to overcome that drawback is to use simpler data structures like the silhouettes are. They are easily obtainable and cheap to process. In this section, it was shown how graph matching can be applied to object recognition in silhouettes. Scenes and model objects are segmented in the same way to arrive at view dependent graphs. A model base containing a total of 9 views for 7 objects was built and a strategy for splitting multiple object scenes into single object hypotheses was developed. It is based on the heuristics, that concave corners separate objects. To reduce the search space of the matching algorithm, three constraints are applied. The simplest one just compares segment attributes. Segments only can match if they are similar “enough”. In the second constraint, pairs of matched model and scene segments are checked for consistency with the already matched pairs. As a third constraint, the geometric transformation from the scene segments to the model segments is finally computed. If this can be done precisely enough, a match is accepted and the corresponding model object and its transformation to the scene is the result of the recognition. The technique was demonstrated on two scenes.

Problems encountered have their origin mainly in the ambiguity of the segmentation and in occlusion in the presence of multiple objects. Segmentation can change with slightly tilted or rotated objects. The top rim of a glass for example is seen from the regular view point as an elliptic arc. If the glass is tilted slightly towards the camera, the top rim will be seen as a straight line segment. When the matcher still should be able to match such a situation, it has to relax the constraint that only segments of the same type can match. It must be allowed, that circular arcs with a large enough radius can match a linear segment. But as consequence this again reduces the selectivity of the unary constraint so that many new false matches will occur. Another problem arises with the number of segments. In the presence of reflections or under certain critical view angles it can happen that segments seen as a whole under a certain aspect can be broken into a couple of smaller ones in an other view. To deal with that, the matcher has to allow some unmatched nodes. But this again can lead to additional false matches.

Multiple mutually occluding objects again have an impact on the selectivity of the matcher’s constraints. Segments can be completely or just partially
occluded. Relations between them can be missing and new false ones may be seen. As consequence, the constraints again have to be relaxed.

Graph matching based recognition has been applied successfully in object recognition with range data ([FMN88]) and recognition in binary images ([Eic92]). Reasons for that may be stronger constraints in 3-D data or special salient local features of the object models. In the case of recognition from silhouettes, the graph matching approach seems to be problematic for the reasons listed.

4.2 Recognition based on evidence accumulation

A major drawback of graph matching is its enormous computational cost because of the use of backtracking. Although there are techniques to cut down on that as the previous section demonstrated, the fundamental drawback of backtracking remains. Compared to that the human vision system is very fast and Agosta [Ago90] concludes that “...the eye rarely resorts to backtracking”. It is perhaps also the wrong way for computer vision systems. Under these aspects the evidence based recognition technique represents an attractive alternative. It uses all features available from the segmentation in a rather unbiased way. Since a particular feature may be part of many different objects, its presence is used to assign some evidence to all of these possible objects simultaneously. With a large quantity and variety of features clusters will form in the evidence space and the “best” one can be selected as recognition result. Note that the well known Hough transform ([DH72]) can also be seen as an evidence based recognition system. Image features (such as edge points) are used to increment many accumulator cells of particular objects (such as straight lines) at the same time. Recognition is done by looking for the accumulator cells with the highest counts. There is a difference, however: evidence weights in the evidence based recognition technique are signed real numbers whereas they are typically positive numbers in the Hough transform.

Based on this technique Jain and Hoffman [JH88] introduced a vision system for recognizing single objects in range data scenes. 10 different objects presented to the system individually in arbitrary positions were used. A laser range scanner provided range data which were then segmented into patches with attributes. Between them, geometric relations were computed. In an intermediate step patches were merged under the assumption of an ob-
ject hypothesis. Various attributes of the patches were then measured using a rule-based framework for a statistical classification of the object. The system presented here is similar in the evidence accumulation scheme but it uses a different data acquisition module and different feature representations. Multiple object scenes and occlusion are also allowed.

The recognition system consists of two parts, a domain specific rule base and a general evaluation part.

The rule base stores knowledge about the problem domain in the form of a set of production rules. Each rule contains a condition part and an action part. The condition part contains statements on the occurrence of a feature, or on an attribute value being in a certain range or a combination of such expressions. The action part contains a list of weights expressing positive or negative evidence to be distributed over the set of hypotheses for objects. Weights are in the range \([-1, 1]\) and their position in the list corresponds to a particular object in the list of admitted objects.

To compare it better with the traditional search algorithm for model based recognition, the rule base can also be viewed as being composed of a model base and of a symbolic scene description.

4.2.1 Model representation

The rule base and with it the model base is defined around the chosen features, which are dependent on the low level segmentation process and the image sensor. In the case of contours segmented as described in chapter 3 various features can be derived from the attributes of the segments. Different objects may generate the same features and different features may be generated by the same object. Also the segmentation may fail or produce wrong results. It is therefore important for the robustness of the system that the recognition process not only depends on a few features. For a reliable recognition it is necessary to access all information of the contours and to generate a wide variety of features. It is also important to have some features that are well discriminating. Global features are of less practical importance because they are useless in the presence of occlusion.

The selection of features is guided by the characteristics of typical contours, which have lateral symmetry for rotational symmetric objects as already noted in section 3.4. If only the top part is viewed, things get even simpler since that
part roughly can be approximated by a rectangle. The feature definition takes advantage of that fact by measuring all features relative to the so called pivot point \( P_{piv} \). As \( P_{piv} \) the highest point in the contour is chosen. Around \( P_{piv} \) the so called "Contour Region of Interest, ROI" is defined. Delimiter of the ROI are the two concave corners to the left and to the right of \( P_{piv} \). Within the ROI, the two convex corners closest to \( P_{piv} \) are identified. The meaning of the ROI is to select the topmost rectangle of the contour. The two convex corners within the ROI define a side of that rectangle (fig. 4.8a). In general a definition like that is not such a good idea since the position of \( P_{piv} \) has to be known to identify the features and in arbitrary orientations of the contour this is a hard problem. But for the application of identifying objects on a cafeteria tray, this is not a severe limitation because the objects (and thus their contours) are very unlikely in arbitrary rotated positions.

For a single object contour, the following features can be defined:

- **scale variant features**: The values of these features depend on the position of the object in the workspace. But with the help of the perspective mapping from scene to image and because the possible range of distances between objects and camera is known, it is possible to determine admissible ranges for each feature and model object.

  - **The maximum apparent height of the contour**, \( h_{piv} \). This value is measured in the contour and it is obtained in image coordinates.
  
  - **The length of the ROI**, \( l_{ROI} \). Measures the length in pixels from the left concave corner point of the ROI to the right one.
  
  - **The distance between the convex ROI corner points**, \( dx_{ROI} \). Measures the distance in pixels between the convex corner to the left of \( P_{piv} \) to convex corner to the right of \( P_{piv} \).
  
  - **The distance between the concave ROI corner points**, \( dc_{ROI} \). Measures the distance in pixels between the concave corner to the left of \( P_{piv} \) to concave corner to the right of \( P_{piv} \).
  
  - **The circle fit in \( P_{piv} \)**. Instead of a linear approximation, a circular arc is fitted to the contour in a small neighborhood of \( P_{piv} \). Attributes are the radius and the quality of the fit.

- **scale invariant features**: 
  
\[ \text{\footnote{see section 4.2.5 why this is also essential for multiple object scenes}} \]
4.2. Recognition based on evidence accumulation

- **The angle between adjacent line segments.** It is the difference between the direction angles of the adjacent segments. When the line fits are of high quality, the angle can be computed very precisely. Some objects have very distinct angles so that this feature may be very discriminating.

- **The ratio \( \frac{d_{ROI}}{d_{ROI}} \).** This feature is well suited to discriminate small elongated ROI's from wide ones.

- **The linear fit in \( P_{pi} \).** In a small neighborhood of \( P_{pi} \) the contour is approximated by a straight line. Attributes are the direction angle as well as the quality of the fit.

- **Material of object in the ROI.** The gray values along the symmetry axis is compared with the gray values of the background. If these values are similar, then the object must be transparent in that region. Otherwise it is not transparent or occluded by a non transparent object.

**Figure 4.8:** Rectangular region on the contour defined by the ROI (a). Definition of contour features within the ROI (b)
These features are used to build a rule base (4.2.3) with $R$ rules. One part of that base are the so called evidence vectors $\zeta_i$. For each model object $i$ such a vector of length $R$ is defined. The elements of $\zeta_i$ are positive and negative numbers $w_t$ expressing how much the rule $t$ (i.e. the presence of a feature) votes for (positive weight) or against (negative weight) object $i$. The model data base then consists of $N$ evidence vectors $\zeta_i$ when $N$ is the number of objects in the base. Note that this is a very compact data structure since in typical applications $N$ is in the range from 10 to 20 and $R$ is smaller than 60. Computing these vectors can be done off-line. The assignment of the weights is an intuitive process that is done by hand in the current implementation. Although in some cases the relation between a certain feature and an object model is clear and the setting of the weight is straightforward, there are many cases with ambiguities. The subjective opinion of the “weight setter” has therefore an influence in these cases. Only a large and varied sets of features can reduce that influence.

### 4.2.2 Scene representation

As a first symbolic scene description, the segmentation process produces a list of features from the feature catalog described in the previous section. Applying all $R$ rules to these features leads to an even more compact representation of the scene than the segmented contour string. It is the so called instance vector $\ell$ of length $R$. Elements of $\ell$ are just the number 1 and 0. An element $\ell_j = 0$ expresses that the conditions the rule $j$ states are not found in the scene and a 1 expresses that they are found, i.e. the rule numbered $j$ fires.

### 4.2.3 Definition of rule base

The formal definition of the rule base corresponds to the definition given by Jain and Hoffman in [JH88]. For completeness and ease of reference we give a short presentation of the main ideas.

The rule base consists of a set of production rules. Each of it is composed of a condition part and an action part. The action part tells which object evidence cells will be incremented. The action part of all rules together can be represented as a matrix $M$ of evidence weights. Having $N$ model objects, all their evidence vectors $\zeta_i$ of length $R$ form $M$ with dimension $N \times R$. In a row $j$ of $M$ are all evidence weights of rule $Z^j$. The condition part consists
of a list of features $F$ and associated with that a list of attributes $A$ and a list of bounds $B$. Evaluating a rule means testing all features and their attributes and checking whether they lie in the bounds.

Formally, a rule $Z^j$ consists therefore of the following 4 parts:

$$Z^j = \{ F^j, A^j, B^j, W^j \}$$

$F^j$ The list of features involved in the rule

$A^j$ The list of attributes of the features

$B^j$ The list of bounds corresponding to $F^j$ and $A^j$

$W^j$ The list of length $N$ of evidence weights for each individual object. The weight $w^j_i$ represents the degree to which satisfaction of evidence condition $j$, given by $\{F^j,B^j,A^j\}$, supports the hypothesis $H_i$ that the observed object is object $i$.

Each $w^j_i$ could be any value in the range $[-1..1]$, but to reduce the subjective influence of the rule maker, all $w^j_i$ are restricted to one of five possible values:

$$w^j_i = \begin{cases} 1 & \text{rule } j \text{ strongly supports } H_i \\ 0.5 & \text{rule } j \text{ tends to support } H_i \\ 0 & \text{rule } j \text{ contains no information about } H_i \\ -0.5 & \text{rule } j \text{ tends to refute } H_i \\ -1 & \text{rule } j \text{ strongly refutes } H_i \end{cases}$$

As an example, the list of features $F$ could contain just the feature $dx_{ROI}$, the distance between the convex ROI corner points. The list of bounds $B$ then consists also only of one element each, say $(75 \ldots 80)$. Such a rule states, that for the rule to fire there must be a feature $dx_{ROI}$ in the interval $75..80$. The THEN part of the rule expresses what this condition means for each object, so for example the list of weights

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \end{pmatrix}$$

gives no evidence for object 1..5 and 7..11 but 0.5 for the object 6, which means that $dx_{ROI}$ in the range $75..80$ is an observation that votes for object 5 only.
4.2.4 Evaluating the rule base

In contrast to the rule base itself the evaluation of the rule base is completely domain independent. All it has to know is the structure of the rules. As input it reads the rule base and the scene features and as output it produces an index into the model base. Its execution consists of two steps, which will be explained in the following.

1. Apply all rules \( \{ Z \}^R_{j=1} \) to all features extracted from the scene. The result of that stage is a vector \( \underline{e} \) of length \( R \), called instance vector. The elements \( \underline{e}_j \) of \( \underline{e} \) are:

\[
\underline{e}_j = \begin{cases} 
1 & \text{if the features in the scene fulfill rule } Z^i_j \\
0 & \text{otherwise}
\end{cases}
\]

2. Compute for every object \( i \) a similarity measure \( \tau_i \) involving the object's evidence vector \( \underline{c}_i \) and the instance vector \( \underline{e} \).

An entry of 1 in the instance vector element \( \underline{e}_j \) means, that the condition of rule \( j \) are fulfilled in the scene: The features \( F^j_i \) lie in the bounds \( B^j \). \( \underline{e} \) serves as a mask vector that selects the rules relevant to the scene. To make a decision for an object \( i \), it is meaningful to have elements of \( \underline{e}_j = 1 \) always when the entries of \( \underline{c}_i \) are positive and to have 0 entries in \( \underline{e}_j \) when the corresponding elements of \( \underline{c}_i \) are negative. A standard rank correlation measure between \( \underline{e} \) and \( \underline{c}_i \) is difficult to use because it interprets positive and negative elements of \( \underline{c}_i \) in the same way. Therefore we use similarity measure \( \tau_i \) as Jain and Hoffman did in their work [JH88] so that the corresponding formulas can be found there. \( \tau_i \) is in the range \([-1..1]\). A value of \( \tau_i \) close to 1 means that there is a good match between the observed features and the positive evidence features corresponding to object \( i \). On the other hand, a value close to -1 means a good match between the observed features and the negative evidence features of object \( i \). Object recognition is done by determing the object (model) with the largest value of \( \tau_i \).

4.2.5 Evidence-based recognition with silhouettes

In the following the recognition scheme described above will be applied to the problem of recognizing objects in contours. The object set to be recognized is
a set of dishes as it can be found in a typical cafeteria. It consists of:

1. a spoon
2. a tea spoon
3. a knife
4. a fork
5. a cup
6. a drinking glass
7. a soup bowl
8. a salad bowl
9. a plate
10. a soup plate
11. a dessert bowl

Contours of these objects are generated and segmented with the methods described in chapter 3. For such an application it is not a big loss of generality to assume that the objects usually are in a upright standing "normal" position. Contours of these objects are therefore not seen in arbitrary position and the conditions for the definition of the feature pivot point $P_{piv}$ from section 4.2.1 are fulfilled. But it is quite possible that objects are tilted a little bit from the normal position when they stand on a non-planar surface. A salad bowl that partially stands on a knife may be such an example. That case is no problem as long as $P_{piv}$ still lies between the two concave corners of the top of the object. However, if the tilting is too large and $P_{piv}$ passes over one of the concave corners, the features are not defined as supposed anymore and the recognition will fail. Figure 4.9 shows the limitation of the tilt angle.

Another problem is how to handle multiple objects in the contour and the resulting occlusion. For that, two basically different approaches are possible.

One could analyze the whole contour and identify all objects at once. The segmentation algorithm given in section 3 is already designed in view of such a solution since it segments the whole contour. The problem with that approach
is however, that often not all objects are seen in the silhouettes because of occlusion. Figure 4.10 shows an example with a cup inside a soup bowl. There really is too little information available from the silhouette to make any decision about all objects in the scene. Nevertheless some objects are clearly visible and it is easy to identify them.

Figure 4.10: Hidden objects in a Silhouette

The alternative is an iterative process. In each step it is tried to recognize just one object. Criteria like ideal removability of an object by a robot may help to select an object to focus on. After recognition, the object is deleted in the data structure and removed from the scene. In the next iteration, the remaining features or the ones computed from a newly taken picture are used.

Whereas the first technique makes sense for a vision system with no feedback to the scene, the second approach can reduce complexity and increase
efficiency in robot vision systems where a robot manipulates the objects. A clearing task like the one presented here is per definition an iterative process so the iterative approach for recognition, too, is appropriate. The key issue is to define a focus of interest on the contour and to select only the features lying in this region for recognition. Two heuristics are applied for that:

1. The object which includes the highest contour point is always the one to be removed next.

2. It is assumed, that concave corners on the contour separate objects.

The idea behind assumption 1 is that in such a way we always focus on the object which presumably has the best visibility and grippability at the same time. This simplifies path planning for the robot a lot but also it implies, that the highest object always can be removed. This is not always true. The foot of the dessert bowl for example could be covered by the rim of a plate so that the plate should be removed first even when the dessert bowl is higher. But these cases are very rare so that the assumption is reasonable. Assumption 2 implies that the contour of any individual object is convex, which also is not true in general. But the definition of the model features around $P_{piv}$ is limited to the topmost convex part of an object anyway so that assumption 2 means no additional limitation (besides the one that come from this feature definition).

Another problem with multiple objects contours is that some of the defined features lose importance. Objects may be stacked for example so that the measured height is composed of the height of the involved objects. For that reason, heights are only usable for negative evidences. If an observed height is smaller than object's $i$ model height, then negative evidence has to be assigned to object $i$. Considerations like this have to be made for every feature and have an influence of the design of the rules. But with a large variety of rules it is possible to compensate for the reduced usefulness of some features.

**Design of an example rule base**

To formulate rules, a setup with a camera, a lens and an area in front of the camera where objects may be placed has to be chosen first. This is necessary to determine quantitative ranges of scale dependent features. To do so, each model object is presented to the system in two distances to the camera, as close as possible and as far away as possible with the constraints given by the tray.
In both positions all features from the feature catalog from page 70 and their ranges are computed. Each of it is used to formulate a rule or a whole family of rules. The following list shows some examples:

- **The highest point in the ROI,** \( h_{piv} \): The height of each individual object was measured in the contour with respect to the rim of the tray. Through the perspective distortion of the camera and the fact, that the object may be posed in a certain range of distances to the camera, each object has a range of possible apparent height values. Objects being similar in height may have overlapping height ranges. With these rules, it is possible to exclude an object \( i \) from the set of possible interpretations as soon as \( h_{piv} \) is smaller than the lower limit of object \( i \)'s height range. Since objects may be stacked, it is impossible to state a positive or negative evidence if \( h_{piv} \) is higher than the upper limit of the highest objects height range. But the smaller the value of \( h_{piv} \) gets, the more negative evidence can be assigned to larger objects. In the following an example of such a rule is given: The list of features \( F_j \) of that rule contains only the element \( s\text{-}roi\text{-}height^2 \), bounds \( B_j \) is 0..45. If \( h_{piv} \) is in the range of 0..45 pixel, then it is impossible that the objects Cup (Cu), Glass (Gl), Soup Bowl (SoB) and Salad Bowl (SaB) are present in the scene because their silhouette is always larger than 45 pixels in any position. These objects therefore get the maximum negative evidence weight. Nothing can be inferred with this rule for the other objects. They get 0 therefore as evidence weights.

\[
\text{# weights:} \ (\text{Soup spoon}, \text{Spoon}, \text{Knife}, \text{Fork}, \text{Cup} \\
\text{Glass}, \text{Soup bowl}, \text{Salad bowl}, \\
\text{Plate}, \text{Soup plate}, \text{Dessert bowl}) \\
\text{#} \\
\text{Height of Salad bowl, Cup, Glass, Soup bowl} \\
\text{and Dessert bowl is > 45 pixel} \\
((s\text{-}roi\text{-}height \ bet \ (0 \ 45)) \\
(0 \ 0 \ 0 \ -1 \ -1 \ -1 \ -1 \ 0 \ 0 \ -1))
\]

- **Distance between the convex ROI corner points,** \( dx_{ROI} \): A diameter of the ROI can also be measured in the silhouette. It is the width of the projection of the object or part of it in the area around \( h_{piv} \). Due to occlusion, this diameter can be composed of different objects and

\footnote{Feature names in the rule base differ from the names used in the explanations here. A complete correspondence table can be found in appendix C}
therefore can be larger than a single object's diameter. Yet it can't be smaller. Non rotation symmetric objects like the salad bowl (SaB) may be seen from different orientations which leads to a certain range of the ROI diameters. The following example rule states, that when the measured $d_{ROI}$ in an image is in the range of 170 to 240 pixels, there is a positive evidence for the soup bowl (SoB) and the salad bowl (SaB).

```
# ROI distance of Soup bowl and Salad bowl
# is in [170,240]
((s-roi-corn-dist bet (170 240))
 (0 0 0 0 0 0 .5 .5 0 0 0))
```

- **Length of the ROI, $l_{ROI}$**: The length of the selected contour part around $h_{piv}$ is also scale dependent. If $l_{ROI}$ is larger than a prescribed value, nothing can be inferred from it since it may be composed of many different objects. But for small enough values it is very likely that the ROI is part of a cutlery piece because no other object is so small or has such small object parts. With the experimental set up, it was found that $l_{ROI}$ smaller than 50 pixels gives positive evidence for cutlery pieces.

```
# Rule for the length of the ROI, 
# small number -> cutlery
((s-roi-len distance (0 50))
 (.5 .5 .5 0 0 0 0 0 0 0))
```

- **The relation $l_{ROI}/d_{cROI}$**: This is scale independent feature can be used as evidence for cutlery pieces as well as for the dessert bowl. A typical situation is a handle pointing out of a bowl. A spoon in a cup for example. In that case, $l_{ROI}$ typically is much larger than $d_{cROI}$ because the handle is long and narrow. The same can happen with the dessert bowl with its narrow foot. It was found, that a quotient $l_{ROI}/d_{cROI}$ larger than 4 gives clear evidence either for cutlery pieces or for the dessert bowl.

```
# Rule for the quotient of the ROI, 
# high number -> cutlery
((s-roi-len quotient (4 20))
 (.5 .5 .5 0 0 0 0 0 .5))
```

- **Miscellaneous rules**: Other features extracted include various fits of part of the contours and a symmetry axis. The circle fit in $h_{piv}$ for example can be used to discriminate the cutlery pieces from the other
objects since usually their silhouettes are small and elongated leading to a good fit of a small circle in $h_{piv}$. Another special rule deals with the material of the objects. It is easy to detect whether a symmetry axis lies in the area belonging to a transparent object (see section 4.2.1). When no transparency is detected, nothing can be stated since a transparent object may be occluded by a non-transparent one. But if there is transparency detected, it is pretty sure, that there must be a Glass, Salad Bowl or a Dessert Glass.

\[
\text{# Glass, Salad Bowl & Dessert Bowl}
\]
\[
\text{# are made of glass}
\]
\[
((s-roi-sym-axis\ mat\ (1\ 1))
\quad (0\ 0\ 0\ -1\ .5\ .5\ -1\ -1\ -1\ .5))
\]

The complete rule base for this experimental set-up is listed in appendix D.

A simple example

To test the recognition scheme, various scenes were presented to the system. Figure 4.11 shows a typical example with 6 objects on the tray: a plate, a glass, a cup, a dessert bowl, a fork and a knife.

![Silhouette with features extracted from ROI](image)

**Figure 4.11: Silhouette with features extracted from ROI**

It is clearly visible that the heuristics that delimits the ROI by the concave corners on both sides of $P_{piv}$ makes sense. The upper part of the dessert bowl
is selected for feature extraction. And when the dessert bowl is removed the
drinking glass will be selected and after that the cup.

The features extracted in the ROI are listed in table 4.1. Note that there
are more features generated than used by the rules. This is because the
segmentation and the feature generation run completely independent of the
recognition. Always all possible features are extracted and the decision which
ones to use is left to the recognition step.

Table 4.1: List of extracted features

These features are interpreted by the rules partially explained in the pre¬
vious section and completely listed in appendix D. There are 20 rules in that
rule base. To explain the recognition process, all weights (the THEN parts)
and the instance vector \( \ell \) is shown in table 4.2.5. The four 1 entries in \( \ell \) say
that the condition parts of rules 6,17,19 and 20 were fulfilled in the scene. This
means, that

1. the feature \( dx_{ROI} \) is observable in the scene and its value lies in the
interval of 140 to 150 pixels (rule 6)

2. a symmetry axis belonging to an object is in the scene and it is transparent
there (rule 17)

3. the quotient \( \frac{j^*}{d_{cROI}} \) is in the interval 4 to 20 (rule 19)

4. a feature \( dc_{ROI} \) is observed in the scene and its value is in the interval
of 0 to 50 pixels (rule 20)
The similarity values of table 4.3 are obtained as explained in section 4.2.4. The object with the highest number, the dessert bowl, is considered to be recognized.

Wrong results are produced by the system mainly when too few features or wrong ones are generated. A typical problem situation occurs when the contours of two objects happen to have the same height in one image and are merged. This can happen through the perspective distortion of the camera even if all objects have different heights. Figure 4.12 illustrates such a case. There the ROI is composed of the cup and the glass which results in completely wrong features. To handle such a situation or only to detect it is almost impossible with the assumptions from page 77 and the relative feature definition with the pivot point $P_{piv}$. Since these cases are rare and the system works correctly in many other cases we think that the solution consists in an extension and not a replacement of the current system. A discussion of this extension is presented in section 4.3.
Table 4.3: Similarities for the example scene given in figure 4.11. The object in the ROI is a dessert bowl

4.2.6 Conclusions relative to the evidence-based recognition scheme

In this section we have shown how the evidence based recognition technique originally developed for recognizing single objects in range data could be adapted and extended for recognizing multiple objects in complex scene contours. The recognition system was a part of a robot vision system that worked iteratively by recognizing one object at a time. Heuristics were used to select

Figure 4.12: Failure of feature extraction in ROI
a region of the contour supposed to belong to a single object that would be the
best choice to be removed next. Scale dependent features within that region
were defined relative to a so-called pivot point. Model knowledge was stored
in a production rule base. Part of that base are evidence vectors representing
the model objects (the columns of the matrix in table 4.2.5). They were ob-
tained by measuring and/or computing all feature values in various positions.
A scene vector is derived from the extracted scene features by applying all
rules to them. There is no bias in the evaluation, all features are treated on an
equal footing. Recognition is done by computing a similarity value between
the scene vector and the model vectors. There is no possibility to backtrack
in the evidence accumulation scheme, but when only low similarity scores are
achieved, rejection is generated as result. By relying only on these simple data
structures, the system is efficient. Execution times of the implemented system
are not noticeable on a workstation in comparison to the low level processes.
Results with moderately complex scenes composed of up to 6 objects out of
11 model objects and a rule base with 20 rules were reported.

The main advantage of the system is seen in the unbiased evaluation of the
features at the object level. The evaluation process is not dependent on the
presence of a certain feature i.e. the system is fault-tolerant against missing
or perturbed features, as long as enough redundant rules and features are
available which can contribute to the decision. Even if only little positive
evidence for the correct object is observed, there may be enough negative
evidence for all remaining objects so that they all can be excluded from the
recognition result. The height of the ROI feature is such an example. If the
height is already small enough, all taller objects can be excluded and only
few survive for the correct solution. However it is advantageous to have at
least some strongly discriminating features. All this makes the evidence-based
recognition scheme robust against segmentation errors. Because there are no
structural dependencies between the features, the system is well suited for
analyzing only a part of a scene as it was demonstrated. This works well even
if the scene decomposition is not correct with respect to the objects contained
in the scene. As we will see later, this property can also be used very nicely to
extend the system for integrating features from multiple views of the scene.

Problems encountered have to do more with the feature definition or with
the weight assignment than with the evaluation mechanism itself. It is impor-
tant to have many features that are determined independently of each other.
Only this precaution can ensure that a clear maximum will form in the evidence
space. The evidence weight assignment is a more or less intuitive process and
4.3 Requirements for a 3-D vision system

In chapter 3 a method to segment and derive symbolic information from contours was presented and in section 4 it was shown how this information can be used for an object recognition system. To use these modules for a robot vision system in which a robot manipulates the objects observed by the vision system, at least three additional problems have to be solved:

- **Occlusion**: Section 4.2.5 describes a technique to handle occlusion in contours. But this technique fails when the contours of two objects merge at the same height (fig. 4.12) or when flat objects lie on the ground base of the tray.

- **Position information in world coordinate system**: For a robot to grip an object, it needs to know, besides its identity, its position and orientation in space.

- **Large variety of features**: In a real world application, factors like noise or reliability of hardware components tend to interfere with reliable recognition. It is important to have built-in redundancy by the use of many diverse features.

The only way to solve all problems is to increase the number of views that are taken of the scene. The unseen flat object for example is a principal problem when using contour data only. It can only be solved with a picture taken from a different view point. The same holds true for the problem of how to get 3-D position and orientation. Only for very limited arrangements would it be possible to get that information with one picture from a single view point. The redundancy and variety of features can easily be improved by additional pictures. For these reasons, the vision system for the application COR is built around three pictures from three different view points. It is presented in section 5.2.
Leer - Vide - Empty
The project “Cooperating Robot”

5.1 Overview

The project “Cooperating Robot”\(^1\) (COR) was launched as a large interdisciplinary project and was promoted by the Mechatronics working group of the Swiss Federal Institute of Technology. The Mechatronics group comprises the Institutes of Robotics, Electronics, Control Theory, Communication Technology and the Chair for Electrotechnical Constructions.

In order to focus the work and to be able to monitor and demonstrate progress, it was decided to launch a concrete and challenging task which consisted in building a complete system for clearing cafeteria trays. The vision system was to analyze the constellation of dishes and cutlery pieces on a tray to such an extent that it could communicate to the robot the information necessary to correctly grasp a piece and put it aside. When a sequence of these operations had cleared a tray, it was moved away, and the next tray to be cleared would move into position.

The vision system used three CCD cameras to get the necessary visual

\(^1\) Kooperierender Roboter mit visuellen und taktilen Fähigkeiten
data, and the interpretation of the scene relied fully on the object model based
approach. The decision procedure used can be briefly characterized as rule-
based evidence accumulation. The project extended over four years, and
completed with a successful demonstration of the full system at the exhibition
"Industrial Handling 92" in Zürich.

The contribution in this chapter is limited to the discussion of the vision
part of the system with special focus on the recognition part. For other aspects
of the COR project like the safety system with speech input, the three-fingered
smart hand with a multitude of sensors well as the robot control system,
see [Vis92].

5.2 The COR vision system

Up to now it is generally agreed that there is no theoretical foundation for a
universal computer vision system. Systems built to solve general problems
are very complicated and show poor performance.

In contrast the COR project tried to solve a very concrete and well defined
vision problem. This definition helps a lot in finding a solution since the vision
system can take full advantage of all known facts about the problem domain.
For that reason it is a more specialized system and is not as elaborate as a
universal system. But it is also more successful in its specialized field. Aloi-
monos and Rosenfeld [AR91] call such a task oriented approach “Purposive
Vision”. For the COR vision system, purposive vision means also the use of
several heuristics. These are

- The vision system always just looks for one object in the scene. The
  assumption is that there is always such an object and that there is enough
  visual information available to identify it. It is also assumed, that objects
  do not interfere with each other when of of them is taken away.

- It it the highest object in the scene the system is looking for. This
  exploits the heuristics that the highest object has the fewest occlusion
  and is the easiest to grip. It is also assumed, that this object may be
  gripped without collision with other objects in the scene (in addition
  there is some collision avoidance capability provided by sensors in the
  fingers of the robot hand).
5.2. The COR vision system

- The object is selected by delimiting the contour at the concave corners nearest to the highest point (see 4.2.1). The assumptions here are that objects are convex at least in their top region.

The vision system is based on the techniques already described in section 4.2.5. But a total of three cameras is used instead of one. In addition to the first camera a second one takes an additional lateral silhouette. From above the tray, a third camera takes a gray-level image. It is used to generate additional features not seen in the silhouettes, to see flat objects on the ground of the plate and to back up the object pose calculation. Through that, the system is capable of deriving 3-D object pose information necessary for the robot to grip an object.

5.2.1 Data acquisition

To obtain visual data of the tray, the method with the illuminated background as already presented in section 3.1 is used. The difference now is the utilization of two cameras to get two lateral silhouettes and a different background illumination technique for the silhouette pictures. Instead of a constant light source in the background, two flash lights are used. Each of it is synchronized with the shutter of the corresponding camera. The flash is turned on only during the time the shutter is open, which has the effect of a very high ambient light suppression. This property is important to increase the reliability of the system when it has to work in places where the ambient light can not be controlled. Because of limited space, the flashes are used indirectly by illuminating a reflecting background. Additionally to the two silhouette cameras, a third one is placed above the tray. From this position, it takes a gray-level image that is analyzed for various features. Details of that analysis are described in [YJ93]. Figure 5.1 gives an overview of the complete set-up.
Figure 5.1: Set-up of the COR vision system
5.2. The COR vision system

5.2.2 Calibration

Visual processing normally is carried out in the so called image coordinate system. Coordinates are measured in pixels in the images, which is sufficient for feature extraction and recognition. However, to guide a robot to some exactly specified location in space, coordinates in the actual space of the objects, the so called world coordinate system are needed\(^2\). To do so, it is necessary to determine the transformation between these two coordinate systems. Each camera of the COR vision system has its own image coordinate system, therefore three perspective matrices are needed. They can be computed with the help known control points in the images. For the COR system, an automatic procedure that used a sphere moved by the robot to a set of predefined positions as control points was developed. Through that use of the robot in the calibration procedure the inaccuracy of the robot could be partially compensated. A complete description of the calibration procedure can be found in [Tro93].

5.2.3 Feature extraction

The images from all three cameras are processed independently of each other in the low level processing steps. This allows a coarse parallelization on a multiprocessor machine. This fits well into the selected hardware platform, a four-processor Stardent 3000. Two processors are assigned to the two silhouettes and the remaining two process the image from above. Through that, operations like filtering the gray-level images with the Canny operator, extracting the contours with their “ROI’s” and analyzing the top picture for circles and ribbons are carried out in parallel. At the end of these processes, all silhouette features as well as all top view features are put together to form a feature list, which is handed over to the high level parts of the system. These are handled just by one processor because they are not time critical and hard to parallelize.

\(^2\)Actually things are more complicated, since the robot’s movement are defined in its own coordinate system and an additional transformation is necessary to get to the world coordinates.
5.2.4 Object recognition in the COR vision system

Object recognition is done with the evidence accumulation system described in section 4.2. Due to the higher demands on the system - use of features from multiple views and requirement to find identity and position of the object - some modifications have to be made. These are:

- Each feature gets an additional attribute view that identifies the camera the feature was extracted from. Values of this attribute are
  - left for features from the left view
  - right for features from the right view
  - top for features from the view from above
  - comb for features from a combination of views (see below).

This additional attribute is used like a selector in the rules. The precondition of a rule is only fulfilled, if the feature lies within the specified bound and if it is coming from the specified view. It is also possible, to use the word any as a selector to indicate that the view doesn’t matter. Such an example is the following rule, which states that if the height of the ROI in any view is below 54 pixels the corresponding list of evidence weights will be used in the similarity calculation.

\[
((s\text{-}roi\text{-}height \text{ any bet } (0 \ 54))

(0 \ 0 \ 0 \ 0 \ -1 \ -1 \ -1 \ 0 \ 0 \ 0 \ -1))
\]

- All features carry geometric information, mostly in image centered coordinates which can be used to determine the position and orientation in space of the recognized object. While evaluating all rules, a list of all features that supported the resulting object with positive evidence is built. The result of the recognition is the identity of the object and this list of features. A separate module described in section 5.2.4 finally computes the position of the object in world coordinates from that information.

- The features used to describe the model objects as well as the scenes are the ones from page 70 plus the following new ones:
  - Radius of the "cover", \( \rho_c \): Under the assumption, that the highest points \( h_{pio}^L \) and \( h_{pio}^R \) in both views belong to the same object and
that this object is rotation symmetric in the part around $h_{piv}$, a radius $\rho_c$ of that part can be computed in world coordinates with the help of the perspective matrices of both cameras. This can be done independently for both views. The difference of both so obtained radii $\rho^L$ and $\rho^R$ can be used to verify the assumptions. If it is too big, $h^L_{piv}$ and $h^R_{piv}$ do not belong to the same object. The difference $\rho^L - \rho^R$ therefore is used to form a quality attribute of the feature $\rho_c$.

<table>
<thead>
<tr>
<th>Object</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glas</td>
<td>34</td>
</tr>
<tr>
<td>Cup</td>
<td>37</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>43</td>
</tr>
<tr>
<td>Salad Bowl</td>
<td>55</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>60</td>
</tr>
<tr>
<td>Soup Plate</td>
<td>103</td>
</tr>
<tr>
<td>Plate</td>
<td>115</td>
</tr>
</tbody>
</table>

Figure 5.2: Ranges of the values of feature $\rho_c$ for all objects.

- Circles from the view above: Many of the objects used have the shape of generalized cylinders. If these objects are in a regular upright standing position on the tray, their top is seen as a circle reflecting the perspective distortion of the camera. Since all objects and their possible positions on the tray are known in advance, the diameters are also known and can be detected with a Hough Transform. A further attribute of the feature is a quality measure calculated from the percentage of the points contributing to the Hough circle.

- Ribbons from the view above: Borderlines of cutlery pieces lying on the tray can be approximated by polygonal sequences. Groups of them having a common axis are called "ribbons". They can be used to recognize objects not seen in the silhouettes or to distinguish between the different cutlery pieces. Again they have a quality measure and an object identity hint as attributes. This hint is computed by matching ribbon models of cutlery pieces with ribbons in the scene.

- In addition to the rules from the example rule base from section 4.2.5 the following new rules can be defined for the new features:
To write rules for the feature $\rho_c$ the top diameter of each object was measured. Since objects may vary due to fabrication tolerances, a certain range for $\rho_c$ is assigned to each object. These ranges may overlap for different objects (fig. 5.2). If this is the case, the range of the feature value is split into separate non overlapping ranges and in an overlapping part for the formulation of the rules. For example $\rho_c$ for the cup is in the range of 37mm..42mm and for the glass it is 34mm..39mm. Therefore, 3 different rules are formulated. One covering the range 34mm..37mm, one for 37mm..39mm and one for 39mm..42mm.

```plaintext
# Cup
((s-roi-world-rad any bet (39 42))
 (0 0 0 0 .5 0 0 0 0 0))
```

```plaintext
# Glass and Cup
((s-roi-world-rad any bet (37 39))
 (0 0 0 0 .5 .5 0 0 0 0 0))
```

```plaintext
# Glass
((s-roi-world-rad any bet (34 37))
 (0 0 0 0 0 .5 0 0 0 0 0))
```

Positive evidences are given in the non overlapping ranges just for the cup or the glass and in the overlapping range for both of them.

The circles seen by the camera above the scene are measured in image coordinates for each object. Since the distance between camera and tray is large compared to possible height variations of the object, the variation of the radii of stacked object is not very large. Nevertheless, there are also overlapping ranges for some objects. They are handled in the same manner as in the previous rule.

**Recognition example**

Figure 5.3 shows an example scene that is used to illustrate the recognition process. The setup with a cup inside the soup bowl and a soup plate standing above is pretty artificial and should not happen in the real world. Nevertheless, it was chosen to show that the vision system is not limited to standard scenes³.

³Examples of a complete sequence of a clearing of a tray are shown in appendix B
The segmentation of the two silhouettes is easily done and the part of the soup spoon’s handle selected as ROI. Together with the analysis of the view from above, 28 features as listed in table 5.1 are obtained and handed over to the recognition stage.

Applying the rules from appendix E to these features results in 21 fulfilled rules firing. The measured features relevant for these rules were the following:

- a distance between the convex ROI corner points $dx_{ROI}$ in the range 0..40
Table 5.1: Features extracted from the example scene given in figure 5.3
The CORvision system

### Table 5.2: Computed similarities for the example scene given in figure 5.3.
The difference between the $\tau$ of the cutlery pieces is so small because there is only one feature, the ribbon, that contains information to distinguish between the individual pieces.

<table>
<thead>
<tr>
<th>Object</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soup Spoon</td>
<td>1.000000</td>
</tr>
<tr>
<td>Tea Spoon</td>
<td>0.996024</td>
</tr>
<tr>
<td>Knife</td>
<td>0.996024</td>
</tr>
<tr>
<td>Fork</td>
<td>0.996024</td>
</tr>
<tr>
<td>Cup</td>
<td>0.408250</td>
</tr>
<tr>
<td>Glass</td>
<td>0.421077</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>0.577351</td>
</tr>
<tr>
<td>Salad Bowl</td>
<td>0.001191</td>
</tr>
<tr>
<td>Plate</td>
<td>0.001633</td>
</tr>
<tr>
<td>Soup Plate</td>
<td>0.577352</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>0.464240</td>
</tr>
</tbody>
</table>

- a distance between the concave ROI corner points $d_{\text{ROI}}$ in the range 0..50
- a quotient $\frac{l_{\text{ROI}}}{d_{\text{ROI}}}$ in the range 4..20
- symmetry axes $S_l$ and $S_r$ in both silhouettes
- circles with various radii in the view from above
- a ribbon in the view from above

From the evidences provided by these rules, the similarities from table 5.2 are computed. As result, the soup spoon gets the highest rating and is therefore selected as the recognized object that will be removed next. The fact that it gets 1.0 which is the highest possible rating means, that all rules that contain positive evidence for the soup spoon actually fired and contributed to the solution. Very close to the soup spoon are the other cutlery pieces, which all get the same rating. This is because there is only the ribbon rule which can discriminate the cutlery pieces. In the example, a ribbon which matches to the soup spoon was found.
Calculation of position and orientation in space

The object model features are described in an object centered coordinate system \( K_m \). Its origin is chosen in the area center of gravity of the object for the cutlery pieces. For all other objects, it is on the top object rim of the symmetry axis of the objects. The procedure to find the object's position and orientation in space is the same for all objects. It consists of finding a translation vector \( \mathbf{T} = (T_x, T_y, T_z) \) and a rotation vector \( \mathbf{R} = (R_x, R_y, R_z) \) between \( K_m \) and the world coordinate system \( K_w \). \( \mathbf{R} \) is described by the angles \( \phi \) and \( \Theta \) and a rotation angle \( \alpha \) of the z-axis of \( K_m \) (fig. 5.4).

Figure 5.4: Translation \( \mathbf{T} \) and rotation \( \mathbf{R} \) between world coordinate system \( K_w \) and model object coordinate system \( K_m \)

\( \mathbf{T} \) and \( \mathbf{R} \) can be determined with the help of the list of features that contributed to the recognition of the object. Each element of it carries some geometric information with itself, but all of them except \( \rho_c \) which is computed through a combination of two features already, are in image centered coordinates. None of them permits a complete reconstruction of the position in world coordinates by itself. To do that, pieces of geometric information of features have to be transformed into 3-D space information. Each trans-
The CORvision system contributes its part to the solution. Since different transformations can be used to compute the same position information part, it is also possible to cross-check the obtained position. In some cases just one feature is used to compute a partial position while in other cases two features from different views are combined.

The features used alone are:

- **The real ROI radius** $\rho_c$: This feature refers to the top part of the object represented by a flat rectangular approximation. Its center therefore corresponds to the origin of $K_m$ so that the translations vector $\vec{T}$ is obtained with that feature.

- **The ribbon from the view from above**: This feature is only defined for cutlery pieces. The origin of $K_m$ for them is in the area center of gravity of the object, the $y$ axis runs along the handle and the $z$ axis is orthogonal to it. The ribbons geometric information consists of a center point $P_{rc} = (x_{rc}, y_{rc})$ in image coordinates and a direction angle $\alpha_r$ of the ribbon axis. $P_{rc}$ transforms with the top camera’s perspective matrices into the $x$ and the $y$ component of $\vec{T}$ and $\alpha_r$ corresponds directly to $\alpha$.

Combinations of features were used in the following cases:

- **Left symmetry axis with right symmetry axis**: The symmetry axis feature’s geometric information consists of a straight line with a starting point on the contour defined in image coordinates. With the help of the perspective matrices of the two cameras each of the axes can be projected into a plane in space. Their intersection represents an estimate of the object’s symmetry axis in space. This combination allows to compute the two angles $\phi$ and $\Theta$ as well as the translation vector $\vec{T}$.

- **Left or right symmetry axis with circle from top view**: This combination is only applicable if a constraint on the circles is in effect: Reflections on the tray or inside plates can cause the Hough transform to detect circles there having the same radius as the recognized object. If these circles also would be used for the position, a completely wrong position would result. The constraint therefore says, that only those circles lying close enough to the symmetry axes’ center projected into the view from above may be used. A circle in the view from a above is
only detected if the object is standing upright. It's center point and the
d end point of the symmetry axis can be used with the corresponding
perspective matrices to compute the object's translation vector \( \vec{T} \). \( \Theta = 0 \)
and \( \phi = \frac{\pi}{2} \) is constant since the objects symmetry axis is assumed to be
standing upright.

In that manner, each component of \( \vec{T} \) and \( \vec{R} \) is computed for every possible
combination of the available features. In a typical situation, there is at least one
possibility for each component. A special case is the rotation angle \( \alpha \). Since
for the feature extraction all objects except the cutlery pieces are assumed to
be rotation symmetric, there is no geometric information available to compute
\( \alpha \). This causes a problem for the cup and the soup bowl, which have handles
and therefore are not exactly rotation symmetric. To compute \( \alpha \), separate
procedures are invoked for each of them:

1. For the cup, \( \alpha \) is computed by explicitly searching for the handle in
the view from above. To do so, the area around the cup is analyzed by
going around the cup's center on a circular path. The intensity values
on that path are used as a one-dimensional signal. In the derivation of
that signal, the handle manifests itself with a typical zero crossing. The
position of that zero crossing allows to compute the angle \( \alpha \) (fig. 5.6
and 5.7).

This is a rather simple scheme and reflections or objects which are too
close to the cup can make the algorithm fail. But during the operational
phase of the system this happened only in about 5% of the cases. A
wrong detection has anyway no drastic consequences, since the gripper
is able to detect and avoid collisions.

2. For the soup bowl, \( \alpha \) is computed with the help of reflections in the
handles. The slightly concave handles give very neat circular reflections,
which are reliably detectable with the Hough transform in the view from
above. These circles are features already used by the rule evaluation
and therefore are included in the list of positive evidence features of the
recognition. Together with the circle of the bowl rim, the rotation angle
\( \alpha \) can easily be computed.

\footnote{A slanted object would produce an ellipse in the view from above, which is not detectable
with the implemented Hough transform}
The components of $\mathbf{T}$ and $\mathbf{R}$ finally are computed by averaging the individually obtained components. Typically there are at most 3 individual components computed, which is not enough to make some statistical analysis. To detect inconsistencies, only differences between individual components are observed. If they get too large, an error message indicating that no position can be computed is generated. The same message is generated if any components of $\mathbf{T}$ or $\mathbf{R}$ could not be computed due to missing features.
Example of position calculation

The scene shown in figure 5.5 with a cup, a soup bowl, a soup dish and a knife is used as an example for the position calculation. The recognition stage successfully identifies the cup as the next object to be gripped and hands over the list of features that led to that result to the position calculation module. These are:

- a circle $C_1^t$ from the outer rim of the cup
- a circle $C_2^t$ from the inner rim of the cup
- a circle $C_3^t$ from reflections of the inside of the soup bowl
- a symmetry axis $S_l$ from the left view
- a symmetry axis $S_r$ from the right view
- the real ROI radius $\rho_c$

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{Comb.} & T_x [\text{mm}] & T_y [\text{mm}] & T_z [\text{mm}] & \Theta [^\circ] & \phi [^\circ] & \alpha [^\circ] \\
\hline
S_l \leftrightarrow C_1^t & 55.22 & 603.63 & -302.01 & 90.00 & 0.00 & \text{Nil} \\
S_r \leftrightarrow C_1^t & 54.50 & 603.29 & -311.20 & 90.00 & 0.00 & \text{Nil} \\
S_l \leftrightarrow C_2^t & 55.55 & 604.02 & -302.00 & 90.00 & 0.00 & \text{Nil} \\
S_r \leftrightarrow C_2^t & 54.65 & 603.57 & -311.19 & 90.00 & 0.00 & \text{Nil} \\
S_l \leftrightarrow S_r & 53.65 & 601.78 & -306.85 & 86.97 & 4.88 & \text{Nil} \\
\rho_c & \text{Nil} & \text{Nil} & -305.28 & \text{Nil} & \text{Nil} & \text{Nil} \\
\hline
\end{array}
\]

Table 5.3: Example of the position calculation

All of them except $C_3^t$ are used for the position calculation. $C_3^t$ is discarded because it is too far away from the center of the combination of the symmetry axes. Table 5.3 lists all combinations that lead to the final translation vector $\vec{T}$ and rotation vector $\vec{R}$:

The angle $\alpha$ was found to be $150^\circ$ with the previously described special procedure for the cup.

This example illustrates the redundancy of the position calculation. All components of $\vec{T}$ and $\vec{R}$ except $\alpha$ could be computed with different features,
which reduces the influence of each individual feature. If one is missing or calculated not precisely enough, the others still contain enough information for a complete position calculation. The values of $\Theta$ and $\phi$ also illustrate the drawback of simply averaging all components. Although the cup is slightly slanted, the combination between $S_{||r}$ and $C_t$ was made. For these combinations $\Theta_i$ and $\phi_i$ have constant values of $90^\circ$ and $0^\circ$, which is imprecise in that case. A better computation of $\Theta$ and $\phi$ is obtained with the combination of the symmetry axes $S_l$ and $S_r$ from the two side views. But in averaging all elements have equal weights so that the imprecise values override the precise one to a certain extent.

Figure 5.6: Computing the location of the cup handle. (a) Cup in the view from above. (b) Gauss filtered intensity of circular path around cup.
Conclusion of position calculation

The shown procedure for the computation of the position of the recognized object in the world coordinate system is simple and straightforward. To compute the three translation and the three rotation parameters, all features conveying geometric information are used. The redundancy in the features itself can lead to multiple possible positions. They are combined by computing the averages. In most cases handled during the operational phase of the system, the position so obtained was accurate within a couple of millimeters and degrees. But there were also cases with one bad position which by the averaging had some influence on the end result. Usually, in these cases some features had bad quality attributes. To reduce the influence of them to the position calculation, it would therefore be better to weight each partial position with a certainty factor computed from the quality of the features.
Figure 5.7: Computing the location of the cup handle. (c) Derivative of (b). (d) Zoom of (b) with found angle marked at 72°
Leer - Vide - Empty
Discussion and conclusion

This thesis demonstrates that contours of objects can provide a valuable contribution to the input data of a 3-D vision system. Although more direct methods like range data for data acquisition exist, contours as a data source were chosen because the system should be able to deal with objects made of glass and with specularly reflecting objects. It is shown, how contours can be generated from gray-leveled images and how they can be segmented into meaningful parts. It is possible to compute relations like “collinear” or “parallel” between the segments. Since many objects have symmetric parts, special emphasis was given to the computation of symmetry relations between segments. The algorithm proposed for doing that is based on a transformation to the $\theta - s$ space, the “direction angle versus the arc length” representation of the contour. Symmetries with respect to an axis are transformed to point symmetries in that representation. It was shown, that they are detectable with a one-dimensional correlation scheme. Segments and relations were then combined into a graph representation of the contour with segments being represented by nodes and relations by arcs. Such a data structure is well suited for the recognition processing steps. With two well known methods, graph matching and evidence accumulation, it was shown how object recognition can be done with contours. To build a recognition system for a real world application, the evidence accumulation based approach was selected because of its efficiency and its potential for realizing a multi-knowledge fusion system. To do so, extension to the basic system were made that enabled it to incorporate information coming from two silhouettes and one gray-leveled image. Additional algorithms for the object
position calculation in space were developed. All algorithms were integrated into the vision system of the application COR and the capabilities of the system were demonstrated at an international exhibition.

6.1 Conclusion

At first sight it might look strange in the age of range data, to include such simple entities as silhouettes into the input data for a 3-D robot vision system. Range data contain direct 3-D information where silhouettes seem to have only limited information about a 3-D scene. As it turned out, there are aspects of range data which limit their use for practical applications: Scene object should ideally have a diffusely reflecting surface. Transparent or specularly reflecting surfaces are not permitted, shadow areas where no data is available cause problems of incomplete surface information and high processing costs. Silhouettes on the other hand do not have these properties. They are easy obtainable, cheap in processing and have no special requirements with respect to the object surfaces. Yet, as it was shown in this thesis together with the gray level image from above they contain enough information for a complete 3-D recognition and grasping task. Besides reliable contour segmentation the evidence based recognition scheme was crucial for this success. The latter, through its unstructured way of combining pieces of information is very well suited for combining results from many different sources. As it turned out, a major advantage of the described scheme is its inherent redundancy. The recognition result does not depend on the presence of certain features, all have equal rights and as long as enough positive evidence is observed, a result can be computed. Especially for an application in a industrial environment this is a major advantage not to be underestimated. It happened for example during the operational phase of the COR vision system that one camera could not deliver a picture because of hardware failure or because an operator was standing in front of a camera. Usually, this did not affect at all the correct recognition of an object because enough information was computable from the pictures delivered by the remaining two cameras. The same is true for the position calculating algorithm, which also combines in an unstructured way all available information and incrementally builds up the solution. We feel, that this simple but very effective method of combining information has proved its reliability and robustness and it is definitely worth to consider it for further applications.
6.2 Future research directions

The very tight project specifications allowed the design of very specific algorithms and a couple of heuristics that simplified the solution. While it is our belief that this is the key to a successful robot vision application, it also is the limiting factor of the system. It is hard to generalize the developed algorithms for a different problem domain, say scenes containing objects which are not rotationally symmetric or unknown objects. But the overall strategy of the system, namely to have several independent sources of information which are combined in a central evaluation mechanism, is very well suited for generalization. In fact, in addition to the modules that process silhouettes and the gray-level image from the view from above new ones could be added that process range images or color images. The second main limitation of the system comes from its fixed programmed data flow. Features are extracted and then they are evaluated to obtain a result. While this was suitable for the well-defined COR environment, it is too restricting for a more general problem domain. There should be a mechanism which allows the high level evaluation part to interact with the low level data acquisition modules. The used evidence accumulation scheme is not prepared for such an extension. For future research it would therefore be interesting to investigate extensions or alternatives that allow such a feedback.
Leer - Vide - Empty
A

Additional results relative to segmentation
Figure A.1: Silhouette (a) of a real scene, its contour with the separator domains (b) and the segmentation result with circular arcs and straight line segments (c)
Figure A.2: Silhouette (a) of a real scene, its contour with the separator domains (b) and the segmentation result with circular arcs and straight line segments (c)
Figure A.3: Silhouette (a) of a real scene, its contour with the separator domains (b) and the segmentation result with circular arcs and straight line segments (c)
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.112</td>
<td>parallel</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>0.092</td>
<td>parallel</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>0.030</td>
<td>parallel</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>0.003</td>
<td>parallel</td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>0.076</td>
<td>collinear</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>0.076</td>
<td>parallel</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>0.142</td>
<td>parallel</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.093</td>
<td>collinear</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>0.108</td>
<td>parallel</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>0.137</td>
<td>concentric</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>0.007</td>
<td>parallel</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>0.013</td>
<td>collinear</td>
</tr>
<tr>
<td>8</td>
<td>21</td>
<td>0.002</td>
<td>parallel</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>0.130</td>
<td>parallel</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>0.300</td>
<td>parallel</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>0.221</td>
<td>parallel</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>0.015</td>
<td>parallel</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>0.052</td>
<td>parallel</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>0.013</td>
<td>parallel</td>
</tr>
<tr>
<td>12</td>
<td>21</td>
<td>0.056</td>
<td>parallel</td>
</tr>
<tr>
<td>13</td>
<td>17</td>
<td>0.013</td>
<td>parallel</td>
</tr>
<tr>
<td>16</td>
<td>21</td>
<td>0.006</td>
<td>parallel</td>
</tr>
</tbody>
</table>

**Figure A.4:** Relations between the segments of figure 3.13
Leer - Vide - Empty
B

Sequence of a clearing of a tray
Appendix B. Sequence of a clearing of a tray

<table>
<thead>
<tr>
<th>Object</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Soup Spoon</strong></td>
<td>1.000000</td>
</tr>
<tr>
<td>Tea Spoon</td>
<td>0.996024</td>
</tr>
<tr>
<td>Knife</td>
<td>0.996024</td>
</tr>
<tr>
<td>Fork</td>
<td>0.996024</td>
</tr>
<tr>
<td>Cup</td>
<td>0.408250</td>
</tr>
<tr>
<td>Glass</td>
<td>0.421077</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>0.408250</td>
</tr>
<tr>
<td>Salad Bowl</td>
<td>0.421077</td>
</tr>
<tr>
<td>Plate</td>
<td>0.001633</td>
</tr>
<tr>
<td>Soup Plate</td>
<td>0.577352</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>0.464240</td>
</tr>
<tr>
<td>Object</td>
<td>$\tau$</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
</tr>
<tr>
<td>Soup Spoon</td>
<td>0.445437</td>
</tr>
<tr>
<td>Tea Spoon</td>
<td>0.445437</td>
</tr>
<tr>
<td>Knife</td>
<td>0.445437</td>
</tr>
<tr>
<td>Fork</td>
<td>0.454258</td>
</tr>
<tr>
<td>Cup</td>
<td>-0.333332</td>
</tr>
<tr>
<td>Glass</td>
<td>0.803362</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>-0.617423</td>
</tr>
<tr>
<td>Salad Bowl</td>
<td>0.539241</td>
</tr>
<tr>
<td>Plate</td>
<td>-0.999996</td>
</tr>
<tr>
<td>Soup Plate</td>
<td>-0.541798</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>0.371393</td>
</tr>
</tbody>
</table>
Appendix B. Sequence of a clearing of a tray

<table>
<thead>
<tr>
<th>Object</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soup Spoon</td>
<td>0.454258</td>
</tr>
<tr>
<td>Tea Spoon</td>
<td>0.445437</td>
</tr>
<tr>
<td>Knife</td>
<td>0.445437</td>
</tr>
<tr>
<td>Fork</td>
<td>0.445437</td>
</tr>
<tr>
<td><strong>Cup</strong></td>
<td>0.707107</td>
</tr>
<tr>
<td>Glass</td>
<td>-0.093972</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>0.577351</td>
</tr>
<tr>
<td>Salad Bowl</td>
<td>0.001191</td>
</tr>
<tr>
<td>Plate</td>
<td>0.001633</td>
</tr>
<tr>
<td>Soup Plate</td>
<td>0.577352</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>-0.999998</td>
</tr>
<tr>
<td>Object</td>
<td>( \tau )</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Soup Spoon</td>
<td>0.445437</td>
</tr>
<tr>
<td>Tea Spoon</td>
<td>0.445437</td>
</tr>
<tr>
<td>Knife</td>
<td>0.454258</td>
</tr>
<tr>
<td>Fork</td>
<td>0.445437</td>
</tr>
<tr>
<td>Cup</td>
<td>-0.999996</td>
</tr>
<tr>
<td>Glass</td>
<td>-0.736959</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>-0.333332</td>
</tr>
<tr>
<td><strong>Salad Bowl</strong></td>
<td><strong>0.684168</strong></td>
</tr>
<tr>
<td>Plate</td>
<td>-0.999996</td>
</tr>
<tr>
<td>Soup Plate</td>
<td>-0.541798</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>-0.824844</td>
</tr>
<tr>
<td>Object</td>
<td>$\tau$</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td>Soup Spoon</td>
<td>0.445437</td>
</tr>
<tr>
<td>Tea Spoon</td>
<td>0.445437</td>
</tr>
<tr>
<td>Knife</td>
<td>0.454258</td>
</tr>
<tr>
<td>Fork</td>
<td>0.445437</td>
</tr>
<tr>
<td>Cup</td>
<td>-0.999998</td>
</tr>
<tr>
<td>Glass</td>
<td>-0.999998</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>-0.333332</td>
</tr>
<tr>
<td>Salad Bowl</td>
<td>0.001191</td>
</tr>
<tr>
<td>Plate</td>
<td>0.001633</td>
</tr>
<tr>
<td><strong>Soup Plate</strong></td>
<td>1.000000</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>-0.999999</td>
</tr>
<tr>
<td>Object</td>
<td>( \tau )</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Soup Spoon</td>
<td>0.890871</td>
</tr>
<tr>
<td>Tea Spoon</td>
<td>0.890871</td>
</tr>
<tr>
<td>Knife</td>
<td>0.890871</td>
</tr>
<tr>
<td><strong>Fork</strong></td>
<td>0.895314</td>
</tr>
<tr>
<td>Cup</td>
<td>-0.999999</td>
</tr>
<tr>
<td>Glass</td>
<td>-0.919735</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>-0.863293</td>
</tr>
<tr>
<td>Salad Bowl</td>
<td>-0.873566</td>
</tr>
<tr>
<td>Plate</td>
<td>-0.999997</td>
</tr>
<tr>
<td>Soup Plate</td>
<td>-0.999998</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>-0.922135</td>
</tr>
</tbody>
</table>
### Appendix B. Sequence of a clearing of a tray

<table>
<thead>
<tr>
<th>Object</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soup Spoon</td>
<td>0.445435</td>
</tr>
<tr>
<td>Tea Spoon</td>
<td>0.445435</td>
</tr>
<tr>
<td><strong>Knife</strong></td>
<td>0.454256</td>
</tr>
<tr>
<td>Fork</td>
<td>0.445435</td>
</tr>
<tr>
<td>Cup</td>
<td>0.000000</td>
</tr>
<tr>
<td>Glass</td>
<td>0.000000</td>
</tr>
<tr>
<td>Soup Bowl</td>
<td>0.408248</td>
</tr>
<tr>
<td>Salad Bowl</td>
<td>0.000000</td>
</tr>
<tr>
<td>Plate</td>
<td>0.000000</td>
</tr>
<tr>
<td>Soup Plate</td>
<td>0.000000</td>
</tr>
<tr>
<td>Dessert Bowl</td>
<td>0.000000</td>
</tr>
</tbody>
</table>
Correspondence table for names in feature base files

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Attribute name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>s-roi-len</td>
<td>s</td>
<td>$l_{ROI}$</td>
</tr>
<tr>
<td></td>
<td>dist</td>
<td>$d_{CROI}$</td>
</tr>
<tr>
<td></td>
<td>quot</td>
<td>$h_{piv}$</td>
</tr>
<tr>
<td>s-roi-height</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s-roi-corn-dist</td>
<td></td>
<td>$d_{xROI}$</td>
</tr>
<tr>
<td>s-roi-corn</td>
<td>l-x l-y</td>
<td>position of left ROI side</td>
</tr>
<tr>
<td></td>
<td>r-x r-y</td>
<td>position of right ROI side</td>
</tr>
<tr>
<td>s-roi-lin-fit</td>
<td>l-dir r-dir</td>
<td>dir. of linear fit on the lft and rgt side of the ROI</td>
</tr>
<tr>
<td></td>
<td>l-abs r- abs</td>
<td>offset of linear fit on left and right side</td>
</tr>
<tr>
<td></td>
<td>l-len r-len</td>
<td>number of points which fulfilled the fit criterion</td>
</tr>
<tr>
<td></td>
<td>l-qual r-qual</td>
<td>quality of the fit on left and right side</td>
</tr>
</tbody>
</table>

125
Appendix C. Correspondence table for names in feature base files

s-roi-cir-fit
1-x 1-y center of fit circle on the left side of the ROI
1-r r-r radii of both fits
1-len r-len nr. of pts on both sides which fulfilled the fit crit.
1-qual r-qual quality of both fits
r-x r-y center of fit circle on the right side of the ROI

s-height-lin-fit
dir direction of the fitted line in $P_{piv}$
abs offset of the fitted line in $P_{piv}$
len number of points which fulfilled the fit criterion
qual quality of the fitted line in $P_{piv}$

s-height-cir-fit
x y center of the circle fitted in $P_{piv}$
r radius of the circle fitted in $P_{piv}$
len number of points which fulfilled the fit criterion
qual quality of the circle fitted in $P_{piv}$

s-roi-sym-axis
dir direction of symmetry axis
x y point on the contour belonging to the sym. axis
len number of points which fulfilled the fit criterion
qual quality of the linear fit of the symmetry axis
glass axis belongs to an object made of glass

s-roi-world-rad
radius $\rho_c$
qual quality of the computation of $\rho_c$
x y z position of $\rho_c$ in the world coordinate system

t-circle
x y center of Hough circle
search model radius
found detected radius
perc percentage of found points
l-dist r-dist dist. of cntr. to projected left. or right. sym. axis

t-ribbon
obj-id recognition hint for the object id
qual quality of the ribbon
mid-x mid-y center point of ribbon
dir direction of ribbon
hp-rline distance of $P_{piv}$ projected to 3-D to ribbon axis
Simple example rulebase

# Weights for the Model objects are in the following order:
# SSpoon TSpoon Knife Fork Cup Glass Soup Bowl Salad Bowl
# Plate SPlate DBowl
# 20 rules for 11 object models

# Rules for the diameter in left/right views
# (image coordinate system)
# -----------------------------------------------
# Plate diameter
1:((s-roi-corn-dist bet (390 440)) (0 0 0 0 0 0 0 0 .5 0 0))

# SPlate & Plate diameter
2:((s-roi-corn-dist bet (340 390)) (0 0 0 0 0 0 0 0 .5 .5 0))

# SPlate diameter
3:((s-roi-corn-dist bet (290 340)) (0 0 0 0 0 0 0 0 .5 0))

# SaBowl & SouBowl
4:((s-roi-corn-dist bet (170 240)) (0 0 0 0 0 0 .5 .5 0 0 0))

# DBowl
5:((s-roi-corn-dist bet (150 170)) (0 0 0 0 0 0 0 0 0 .5 0 0 0 0.5))

# DBowl & Cup
6:((s-roi-corn-dist bet (140 150)) (0 0 0 0 .5 0 0 0 0 0 0.5 ))

# DBowl & Cup & Glass
7:((s-roi-corn-dist bet (120 140)) (0 0 0 0 .5 .5 0 0 0 0 0 0.5))
# Glass & Cup
8:((s-roi-corn-dist bet (110 120)) (0 0 0 0 .5 0 0 0 0 0))

# Glass
9:((s-roi-corn-dist bet (90 110)) (0 0 0 0 0 .5 0 0 0 0 0))

# Cutlery
10:((s-roi-corn-dist bet (0 40)) (.5 .5 .5 0 0 0 0 0 0 0))

# Rules for the height in left/right views
# (image coordinate system)
# -------------------------------
# DBowl
11:((s-roi-height bet (0 115)) (0 0 0 0 0 0 0 0 0 0 -1))

# Glass
12:((s-roi-height bet (0 95)) (0 0 0 0 -1 0 0 0 0 -1))

# Cup & SouBowl
13:((s-roi-height bet (0 75)) (0 0 0 0 -1 -1 -1 0 0 0 -1))

# SaBowl
14:((s-roi-height bet (0 45)) (0 0 0 0 -1 -1 -1 -1 0 0 -1))

# SPlate
15:((s-roi-height bet (0 35)) (0 0 0 0 -1 -1 -1 -1 0 -1 -1))

# Plate
16:((s-roi-height bet (0 15)) (0 0 0 0 -1 -1 -1 -1 -1 -1))

# Rules to distinguish between glass objects
# ------------------------------------------
# Glass & DBowl are made of glass
17:((s-roi-sym-axis material (1 1))
   (-1 -1 -1 -1 .5 -1 .5 -1 .5))

# Rules for the circle fit in the highest point
# ---------------------------------------------
18:((s-height-cir-fit rad (2 12) qua (0 1.5))
   (.5 .5 .5 0 0 0 0 0 0 0))

# Rule for the quotient of the ROI, high number -> cutlery
# -------------------------------------------------------
19:((s-roi-len quotient (4 20)) (.5 .5 .5 0 0 0 0 0 0 .5))

# Rule 30 for the distance of the ROI, small number -> cutlery
# -----------------------------------------------------------
20:((s-roi-len distance (0 50)) (.5 .5 .5 0 0 0 0 0 0))
Complete rulebase for the COR vision system

# Weights for the Model objects are in the following order:
# SSpoon TSpoon Knife Fork Cup Glass Soup Bowl Salad Bowl
#   Plate SPlate DBowl
# # 46 rules for 11 object models

# R. for the diameter in lft/rgt views (img. coord. sys.)
# -----------------------------------------------
# Plate diameter
1:((s-roi-corn-dist any bet (270 300)) (0 0 0 0 0 0 0 0 .5 0 0))

# SPlate diameter
2:((s-roi-corn-dist any bet (230 270)) (0 0 0 0 0 0 0 0 .5 0))

# SaBowl & SouBowl
3:((s-roi-corn-dist any bet (135 170)) (0 0 0 0 0 0 0 0 0 .5 0 0 0))

# SaBowl
4:((s-roi-corn-dist any bet (120 135)) (0 0 0 0 0 0 0 .5 0 0 0 0))

# DBowl
5:((s-roi-corn-dist any bet (105 120)) (0 0 0 0 0 0 0 0 0 .5))

# Cup
6:((s-roi-corn-dist any bet (95 105)) (0 0 0 0 .5 0 0 0 0 0 0))
# Glass & Cup
7:((s-roi-corn-dist any bet (80 95)) (0 0 0 0 .5 .5 0 0 0 0 0 0))

# Glass
8:((s-roi-corn-dist any bet (75 80)) (0 0 0 0 0 .5 0 0 0 0 0 0))

# Cutlery
9:((s-roi-corn-dist any bet (0 40)) (.5 .5 .5 .5 0 0 0 0 0 0 0 0))

# R. for the height in left/right views (img. coord. sys.)
# -----------------------------------------------
# DBowl
10:((s-roi-height any bet (0 80)) (0 0 0 0 0 0 0 0 0 0 -1))

# Glass
11:((s-roi-height any bet (0 70)) (0 0 0 0 -1 0 0 0 0 -1))

# Cup
12:((s-roi-height any bet (0 55)) (0 0 0 -1 -1 0 0 0 0 -1))

# SouBowl
13:((s-roi-height any bet (0 54)) (0 0 0 -1 -1 -1 0 0 0 -1))

# SaBowl
14:((s-roi-height any bet (0 30)) (0 0 0 -1 -1 -1 0 0 0 -1))

# SPlate
15:((s-roi-height any bet (0 20)) (0 0 0 -1 -1 -1 0 -1 -1))

# everything but cutlery is higher that 13
16:((s-roi-height any bet (0 13)) (0 0 0 -1 -1 -1 -1 -1 -1))

# R. for the radius in the top view (img. coord. sys.)
# -----------------------------------------------
# Plate
17:((t-circle top rad (168 177)) (0 0 0 0 0 0 0 0 .5 0 0))

# SPlate
18:((t-circle top rad (150 160)) (0 0 0 0 0 0 0 0 .4 0))

# SouBowl
19:((t-circle top rad (82 96)) (0 0 0 0 0 .5 0 0 0 0))

# Handles of SouBowl
20:((t-circle top rad (12 18)) (0 0 0 0 0 .5 0 0 0 0))

# Handles of SouBowl, gives also evidence for SSPoon
21:((t-circle top rad (18 20)) (0 0 0 0 0 .5 0 0 0 0))
# gives evidence for SSpoon
22:((t-circle top rad (20 25)) (0 0 0 0 0 0 0 0 0 0 0))

# DBowl
23:((T-circle top rad (63 74)) (0 0 0 0 0 0 0 0 0 0 .5))

# DBowl & Cup
24:((t-circle top rad (60 63)) (0 0 0 .5 0 0 0 0 0 .5))

# Cup & Glass
25:((t-circle top rad (50 60)) (0 0 0 .5 .5 0 0 0 0 0))

# SaBowl
26:((t-circle top rad (40 50)) (0 0 0 0 0 0 .5 0 0 0))

# R. to distinguish between glass objects
# -----------------------------------------
# Glass & DBowl are made of Glass
27:((s-roi-sym-axis any material (1 1))
   (0 0 0 -1 .6 -1 .4 -1 -1 .4))

# R. for the circle fit in the highest point
# -----------------------------------------
28:((s-height-cir-fit any rad (2 12) qua (0 1.5))
   (.5 .5 .5 .5 0 0 0 0 0 0 0))

# Rule for the quotient of the ROI, high number -> cutlery
# ---------------------------------------------------------
29:((s-roi-len any quotient (4 20)) (.5 .5 .5 .5 0 0 0 0 0 0 0))

# Rule for the distance of the ROI, small number -> cutlery
# ---------------------------------------------------------
30:((s-roi-len any distance (0 50)) (.5 .5 .5 .5 0 0 0 0 0 0 0))

# Rule for the ribbons, -> cutlery
# -------------------------------
31:((t-ribbon top qua (-10 10)) (.5 .5 .5 .5 0 0 0 0 0 0 0))
32:((t-ribbon top rib-id (9 9)) (0 0 0.1 0 0 0 0 0 0 0 0))
33:((t-ribbon top rib-id (10 10)) (0.1 0 0 0 0 0 0 0 0 0 0))
34:((t-ribbon top rib-id (11 11)) (0 0 0 0.1 0 0 0 0 0 0 0))
35:((t-ribbon top rib-id (12 12)) (0 0.1 0 0 0 0 0 0 0 0 0))
# Dummy rule to get the symmetry axes into the used feature list
# -----------------------------------------------
# to make sure that the features from the dummy rules get into
# the list of used features, for the recognition, all weights
# should have the same positive (just a very little one) value
36: ((s-roi-sym-axis any dir (-180 180))
   (.001 .001 .001 .001 .001 .001 .001 .001 .001 .001 .001))

# Dummy rule to get the world radius into the used feature list
# -----------------------------------------------
37: ((s-roi-world-rad any bet (0 900))
   (.001 .001 .001 .001 .001 .001 .001 .001 .001 .001 .001))

# World radius
# -----------
# SPlate
38: ((s-roi-world-rad comb bet (103 108) qua (-.05 .05))
   (0 0 0 0 0 0 0 0 .5 0))

# Plate
39: ((s-roi-world-rad comb bet (115 125) qua (-.05 .05))
   (0 0 0 0 0 0 0 .5 0))

# DBowl
40: ((s-roi-world-rad comb bet (43 47) qua (-.05 .05))
   (0 0 0 0 0 0 .5))

# Cup
41: ((s-roi-world-rad comb bet (39 42) qua (-.05 .05))
   (0 0 0 0 .5 0 0 0 0 0 0))

# Glass
42: ((s-roi-world-rad comb bet (34 37) qua (-.05 .05))
   (0 0 0 0 .5 0 0 0 0 0))

# Glass und Cup
43: ((s-roi-world-rad comb bet (37 39) qua (-.05 .05))
   (0 0 0 0 .5 .5 0 0 0 0 0))

# SouBowl
44: ((s-roi-world-rad comb bet (62.5 65) qua (-.05 .05))
   (0 0 0 0 0 0 .5 0 0 0 0))

# SouBowl und SaBowl
45: ((s-roi-world-rad comb bet (60 62.5) qua (-.05 .05))
   (0 0 0 0 0 0 .5 .5 0 0 0))

# SaBowl
46: ((s-roi-world-rad comb bet (55 60) qua (-.05 .05))
   (0 0 0 0 0 0 .5 0 0 0))
List of Figures

3.1 Gray level images of a typical scene (a), its edge filter output (b) and the generated silhouette (c) .......................... 14

3.2 Contour extraction by following the border between black and white regions (cracks) ................................. 15

3.3 Problem with the line fit which can lock in situations like (b), orthogonal to the contour ......................... 20

3.4 Diagram (a) shows $\theta(s)$ of the contour from figure 3.7 computed with a secant over 11 points. (b) is a magnified section from (a) with $s$ in the range [150,450]. The steps in $\theta(s)$ is quantization noise resulting from computing $\theta$ with the secant method. ........................................... 21

3.5 $\theta(s)$ (a) and $\kappa(s)$ (b) of the contour part from figure 3.3. Without correction, the fit locks in the corner to a wrong position which results in wrong $\theta$ and $\kappa$ values. The interpolation result is also shown. ........................................... 22

3.6 Initial corner domain contains all points around $p_i(a)$. Shrinking of the domain (b) and the resulting domain$^1$(c) ........ 23

3.7 Example contour of a plate with a fork, a glass and a soup bowl in it. The marked region is used to illustrate the algorithm 24

3.8 One run is seen in the situation of a new corner point (a) whereas around inflection points two runs are seen. ........ 25

3.9 $\sigma(\theta)$ for different length of a digitized straight line ........ 27
3.10 \( \sigma(R) \) for digitized circular arcs with different opening angles and radii ......................................................... 29

3.11 Separator domains (black) with oversegmentation before (a) and after (b) the clean-up step ......................................................... 30

3.12 Parts of the contour approximated by straight line segments and circular arcs ......................................................... 31

3.13 The complete example contour approximated by straight line segments and arcs ......................................................... 31

3.14 Significant points found with two different parameter sets of the Rosenfeld-Johnson algorithm. Separator points on slowly changing transitions between two object parts (arrow) are not found. ......................................................... 32

3.15 Example scene used in conjunction with table 3.1 for demonstrating the relations between segments ............................. 33

3.16 Ideal parallelism between \( S_a \) and \( S_b \) (a) is attributed with a high certainty factor, whereas situation (b) is considered as a distorted parallelism with a smaller certainty factor. ......................................................... 35

3.17 The segments \( S_a \) and \( S_b \) are collinear if they have similar directions \( \theta_a \) and \( \theta_b \) and if the direction of the connection line \( C_{ab} \) is similar to that also. ......................................................... 36

3.18 Graph representation of a simple example contour (fig. 3.15) ......................................................... 37

3.19 SAT (a) and SLS (b) of a rectangle ......................................................... 39

3.20 SLS definition compared to SAT ......................................................... 39

3.21 Contours and the corresponding \( \theta(s) \) function. A rotation of the contour means a vertical shift in \( \theta(s) \) ......................................................... 40

3.22 Distance criterion used to compare sections of \( \theta(s) \) ......................................................... 42

3.23 Brute force SLS algorithm vs. our implementation with selection of segments prior to SLS. Instead of testing each pair of contour points, only a small selection is tested ......................................................... 43
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.24</td>
<td>Segmented contour, the corresponding segments of $\theta(s)$ and the found symmetry axes</td>
</tr>
<tr>
<td>3.25</td>
<td>Local axis hypothesis (microsymmetries) on a synthetic shape</td>
</tr>
<tr>
<td>3.26</td>
<td>For perfect symmetry (a), the angle $\phi$ is 0 which maximizes the certainty factor. For disturbed symmetry (b), the certainty factor decreases proportional to $\cos(\phi)$</td>
</tr>
<tr>
<td>3.27</td>
<td>Local axis hypotheses (microsymmetries) (a) and the selected one (b)</td>
</tr>
<tr>
<td>4.1</td>
<td>Object models for recognition with graph matching</td>
</tr>
<tr>
<td>4.2</td>
<td>A simple scene for the matcher</td>
</tr>
<tr>
<td>4.3</td>
<td>Found match to the “glass” hypothesis</td>
</tr>
<tr>
<td>4.4</td>
<td>Found match to the “cup” hypothesis</td>
</tr>
<tr>
<td>4.5</td>
<td>Scene with substantial occlusion</td>
</tr>
<tr>
<td>4.6</td>
<td>Matches found with the first object hypothesis</td>
</tr>
<tr>
<td>4.7</td>
<td>Split object hypothesis with no match found</td>
</tr>
<tr>
<td>4.8</td>
<td>Rectangular region on the contour defined by the ROI (a). Definition of contour features within the ROI (b)</td>
</tr>
<tr>
<td>4.9</td>
<td>Allowed (a) amount of tilt of contour and disallowed (b) amount</td>
</tr>
<tr>
<td>4.10</td>
<td>Hidden objects in a Silhouette</td>
</tr>
<tr>
<td>4.11</td>
<td>Silhouette with features extracted from ROI</td>
</tr>
<tr>
<td>4.12</td>
<td>Failure of feature extraction in ROI</td>
</tr>
<tr>
<td>5.1</td>
<td>Set-up of the COR vision system</td>
</tr>
<tr>
<td>5.2</td>
<td>Ranges of the values of feature $\rho_e$ for all objects</td>
</tr>
<tr>
<td>5.3</td>
<td>Example scene for recognition</td>
</tr>
</tbody>
</table>
5.4 Translation $\mathbf{T}$ and rotation $\mathbf{R}$ between world coordinate system $K_w$ and model object coordinate system $K_m$ .............................. 98

5.5 Example scene for position calculation ................................. 101

5.6 Computing the location of the cup handle. (a) Cup in the view from above. (b) Gauss filtered intensity of circular path around cup. ................................................................. 103

5.7 Computing the location of the cup handle. (c) Derivative of (b). (d) Zoom of (b) with found angle marked at $72^\circ$ .................. 105

A.1 Silhouette (a) of a real scene, its contour with the separator domains (b) and the segmentation result with circular arcs and straight line segments (c) ................................. 112

A.2 Silhouette (a) of a real scene, its contour with the separator domains (b) and the segmentation result with circular arcs and straight line segments (c) ................................. 113

A.3 Silhouette (a) of a real scene, its contour with the separator domains (b) and the segmentation result with circular arcs and straight line segments (c) ................................. 114

A.4 Relations between the segments of figure 3.13 ....................... 115
# List of Tables

3.1 Relations between the segments of scene 3.15 .......................... 34

4.1 List of extracted features ....................................................... 81

4.2 Example evidence weights matrix with instance vector for a particular scene .................................................. 82

4.3 Similarities for the example scene given in figure 4.11. The object in the ROI is a dessert bowl ................................. 83

5.1 Features extracted from the example scene given in figure 5.3 ................................. 96

5.2 Computed similarities for the example scene given in figure 5.3. The difference between the $\tau$ of the cutlery pieces is so small because there is only one feature, the ribbon, that contains information to distinguish between the individual pieces. 97

5.3 Example of the position calculation ........................................... 102
Leer - Vide - Empty
Bibliography


Curriculum Vitae

I was born in Lucerne, Switzerland, on February 27, 1957. I visited the elementary school in Thalwil, Kanton Zürich and the Sekundar Schule in Zürich at the “Freies Gymnasium”, Zürich. From 1973 to 1977 I absolved an apprenticeship in Electronics with “Landis & Gyr”, Zug. Subsequently I visited the “Interkantonales Technikum Rapperswil” and graduated in Electrical Engineering in 1980. Afterwards I prepared for the entrance examination to the Swiss Federal Institute of Technology (ETH) in Zürich and after successful completion I studied Computer Science there and received an M.Sc. in Computer Science Engineering (Dipl. Informatik-Ing. ETH) in 1984 with a diploma thesis at the Geographical Institute of ETH. I continued my work at that place for almost a year and then I worked as a software engineer in a start-up enterprise in California and later in an engineering company in Zürich. Since October 1987 I am working as a research assistant at ETH Zürich in the image science group of the Communication Technology Institute with Prof. Dr. Olaf Kübler.