A Module for Automated Generation of Planar Object Descriptions

Zoran Kalafatić and Slobodan Ribarić
Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

The article describes the implementation of a module for automated generation of descriptions of planar objects. The module is based on dividing the objects in a scene into parts by using local symmetries, and describing the relations among detected parts. The relations among parts are represented by the knowledge representation scheme based on Petri net theory (KRP). The module is intended to be a part of a computer vision system using the KRP scheme for representing knowledge about objects in the scene.

Keywords: computer vision, automated description generation, local symmetries, object decomposition, scheme KRP.

1. Introduction

A module for automated generation of scene descriptions is a component of a subsystem for automatic acquisition or learning, which, on the other hand, is one of the basic components of knowledge-based vision systems.

The implemented module for automated generation of planar object descriptions is a part of a computer vision system shown in Fig. 1. The module takes input images of the composed planar objects and generates their descriptions

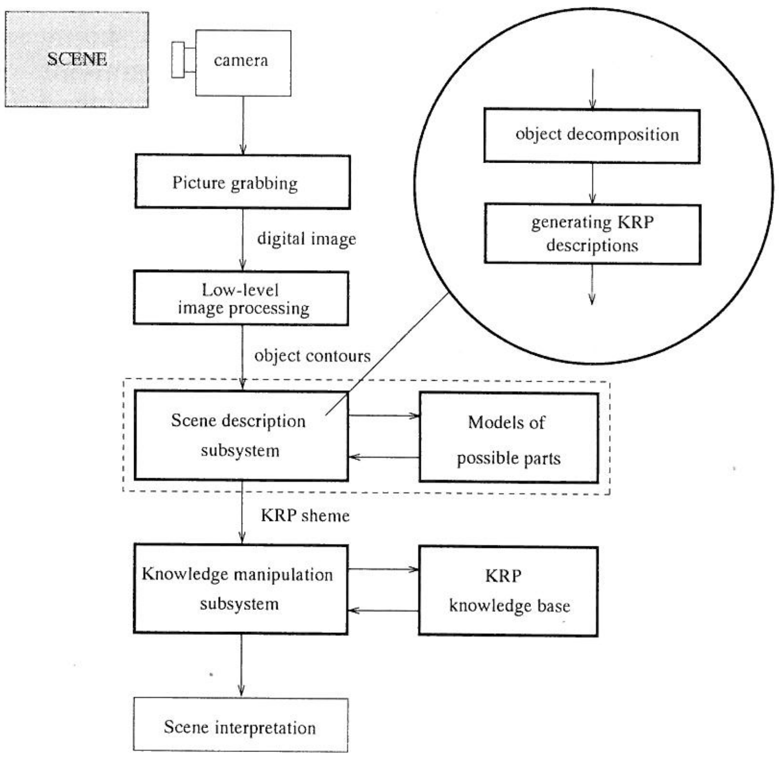


Fig. 1 A computer vision system using the KRP scheme.

in the KRP (Knowledge representation scheme based on Petri Nets) notation. This KRP notation can be used both in the learning phase of the system, and in the object recognition phase during the scene interpretation process.

The module enables automated building and updating the knowledge base (learning). In the scene interpretation process, the KRP notation representing the structure of the planar objects in the scene serves as input to a knowledge manipulation subsystem (Fig. 1), which searches the KRP knowledge base to determine a possible scene interpretation.

Our module for automatic description is based on the assumption that the planar objects in the scene can be divided into building parts, and then, by encoding the parts and the relations among them, a consistent description can be obtained. The above assumption is based on psychological experiments which support the theory that the human recognition process might be based on recognition by components [1]. Also, some authors have pointed out that shape representations based on components provide rich descriptions, make "important facts" explicit, and have good stability with respect to configuration [4].

The operations of the module for automated generation of planar object descriptions can be viewed as a two-phase process. First, planar objects in the scene are divided into parts. This phase is based on local symmetries [2]. In the second phase, the relations between the detected parts should be determined and suitably represented.

The problem of object decomposition can be treated in different ways. One approach tries to find the evidences on the contour that suggest the possible subpart joins. Usually some heuristic criteria are used, but no generally applicable method has been proposed. Criteria for finding joins are usually based on curvature discontinuities and on finding points of sharp concavity [1], [2], [3]. Several studies have shown the importance of two very different descriptors for shape: symmetry structure and curvature extrema. Leyton [7] has proved a theorem which expresses an important relationship between symmetry and curvature extrema.

Our approach is based on finding the instances of parts from a set, defined in advance. The possible parts are defined by using models, and the instances in a scene are to be found using some kind of matching. In fact, in this approach the object decomposition problem could be defined in terms of pattern recognition. In that representation, the input pattern, i.e. a scene, can be viewed as a composition of unknown parts which can be overlapped. The module has to detect such overlapped parts and classify them into classes of building parts. To deal with overlapping, some kind of local information should be applied. We use Brady's local symmetries for shape representation in our object decomposition module. This representation technique has this very important feature of being local, so that it can deal with overlapping. Furthermore, local symmetry axes are found to suggest plausible parts [4].

When a decomposition of an object into parts is found, it can be described by expressing the relations among the detected parts. To represent the relations, we use the KRP scheme. The main problem is to choose the appropriate set of relations which enables a useful description of a scene or a "natural" description that approximates a description generated by a human.

The next section reviews the definition of local symmetry and points out some of its properties which can be used for shape decomposition. The third section shortly sketches our object decomposition algorithm. In the fourth section the definition of the KRP scheme is given, and its applicability to scene description is illustrated. Finally, Section 5 deals with automated generation of KRP scene descriptions, which is illustrated on a simple example.

2. Local symmetries

We accepted the definition of local symmetry introduced by Brady and Asada [2]. Its geometry was given as a part of the definition of Smoothed Local Symmetry (SLS). A local symmetry describes a special relation between two points lying on a contour. A symmetry appears if an axis can be constructed so that the small pieces of contour around the two points can be reflected into each other through the constructed axis, as illustrated in Fig. 2a.

The local symmetries of an object can be found point-by-point, by testing every point on the

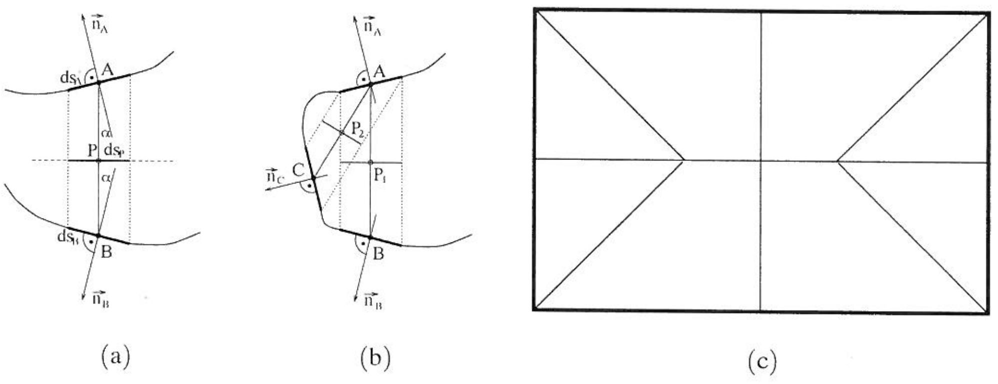


Fig. 2. (a) The geometry of a local symmetry. (b) A point on the contour can form several local symmetries. (c) Local symmetries of a rectangle.

contour against all others. The algorithm is simple, but the time complexity is $\mathcal{O}(n^2)$, where n is the number of contour points. Another shortcoming is substantial sensitivity to the noise. Brady and Asada developed a program that computes the SLSs of a shape by approximating the contour by a series of straight lines and circular arcs. This method reduces the cost of computation and also significantly reduces the noise. Our program for finding the symmetry axes also approximates the contour, but we use the polygonal approximation. It is simpler than curvilinear, yet gives satisfying results. The bounding contours of objects in the scene are represented by sequences of linear segments, so the local symmetries are computed by analytically comparing each segment to all others.

Brady proposed that local symmetries could be used for object decomposition [2]. He proposed two heuristic rules to be applied to find subpart joins. The first rule is in fact a generalization of "matched concavities" heuristic, and the second covers some cases where the first one cannot be used. But, still these heuristics could be used only for a restricted set of configurations. Connell [4] built an object decomposition and description system which relies upon Brady's smoothed local symmetries, assuming that they suggest plausible axes for the parts of the object. His system uses some heuristic criteria to reduce the set of symmetries and to connect symmetries which are (possibly) parts of the same original symmetry. The segmentation program computes parameters for each of the symmetries found, joins symmetries which are different sections of the same primitive region, and finally it chunks the image into a collection of non-overlapping pieces based on the extended symmetries found. The description module computes symbolic descriptors for the shapes of each of the pieces. It also determines which pieces are joined together and how. Finally, a semantic network describing parts and relations among the parts is generated. Rom and Medioni [10] presented their object decomposition system based on symmetry. Their system, like the Connel's [4], chunks the objects in the scene into "intuitive" parts. Our system, on the contrary, uses models of possible building parts.

It turned out that smoothed local symmetries work well only if the object is composed of elongated parts [4, 5]. On approximately round regions an axis-based representation is unstable and, moreover, it doesn't provide an intuitively acceptable analysis. Fleck developed a companion technique called Local Rotational Symmetries (LRS). This technique is based on finding the regions which are approximately circular and describing them in terms of center and radius of curvature. However, she uses the combination of the two representations (SLS + LRS) to provide complete descriptions of object shapes [5]. Our system uses a similar technique for dealing with round regions [6].

3. Object decomposition

For our experiments, we chose a set of building parts and composed them to form planar objects. The parts can be overlapped. The decomposition procedure is the most complex part of our module, using many heuristics. The space restrictions prevent us from going into detail, so only a rough sketch will be given.

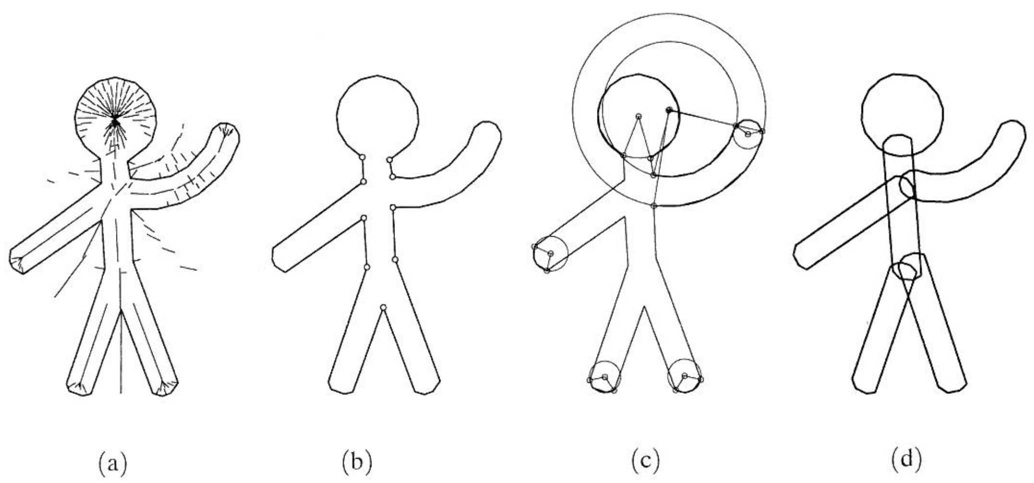


Fig. 3. Decomposition of an object into parts.

The details can be found in [6]. Our object decomposition procedure begins with finding local symmetries among contour segments. The set of symmetries computed for a compound object can contain much more local symmetry axes than it would be obtained by computing symmetries for all parts separately. Some of the original symmetry axes are broken due to the loss of a piece of the contour, which occurs when two parts are connected, and many symmetries appear between contour segments of different building parts. This is illustrated in Fig. 3a (note the difference with the symmetries of individual building parts in Fig. 4). The set of symmetries is then reduced using several heuristic criteria, e.g. short symmetries, symmetries relating distant contour segments and non-intrinsic symmetries [3] are rejected. We try to extend symmetries which can be joined together, assuming that they might be the parts of one symmetry, broken due to subpart joins. In order to be joined, the symmetries must satisfy several heuristic criteria. The regions they describe have to be joined smoothly and should not extend the bounding contour. The outer symmetries are not used for model matching, but they suggest probable subpart joins (similarly as the matched concavities heuristic [2]). These points are used as a valuable assistance in some of the subsequent algorithms. Figure 3b shows such points found by the implemented algorithm. The next step is finding the approximately round regions, i.e. the sequences of linear segments that can be approximated by circular arcs. The local symmetries are unstable in such regions, so it is much more appropriate to represent these regions in terms of center and radius. We use that information in the model matching procedure. The approximation arcs for a simple scene are shown in Fig. 3c.

The decomposition algorithm uses models of possible building parts, which are built in the learning phase. Some of the used building parts are shown in Fig. 4, together with the computed symmetries. The learning phase is one-shot process. That means that each building part is presented only once, in referent position. The system determines the outer contour of the presented part, approximates it linearly, and computes the local symmetries. Then it searches for approximately round regions and describes them in terms of center and radius. All that information is then stored in a model.

The program matches the symmetries and arcs found for a compound object against the models of possible parts. The procedure is invariant to scale, position and orientation of building elements, but we use the fact that the relative size of parts is fixed. So, when the first part is identified, the scale for the whole scene is determined and that parameter is used to ease the subsequent processing. The strategy is simple: the program tries to find the easiest match, i.e. the most emphasized features are used first. When some feature becomes explained (an interpretation is found), the program marks it as used, thus reducing the information to be resolved and easing the subsequent processing. When most of the object contour is explained with detected parts and no more interpretations can be found, the procedure ends.

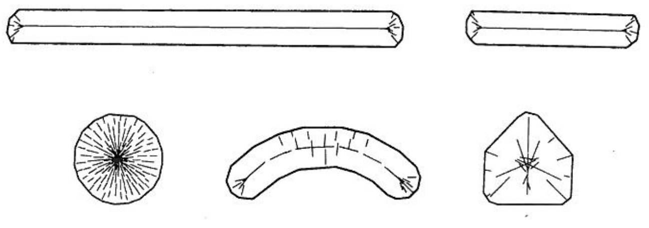


Fig. 4. Some of the used building parts - contour approximations and the computed local symmetries.

We consider the longest symmetries to be the most significant, assuming that they preserved most of their original information. So the program tries to explain the longest symmetries first. The symmetry matching is based on several heuristic parameters describing the region which the symmetry subtends. The main parameters are the angle between contour segments that form the symmetry and the aspect ratio (average width of a region divided by its length) of the subtended region. When a match is obtained, i.e. a possible part is detected, the interpretation is tested for feasibility. The detected part is then substituted with the appropriately transformed model, and the matching between the transformed model and the scene is examined. The system chooses the best interpretation. An example of object decomposition is shown in Fig. 3d. The figure shows the transformed models of the detected parts.

The decomposition subsystem has been tested on series of scenes, composed of a known set of parts. It is difficult to estimate the reliability of the system, but we tried to do so by a simple experiment. A person, not familiar with the system, composed a series of scenes, and we counted the successfully decomposed objects. The subsystem successfully decomposed 69 % of composed objects, i.e. all building parts were detected. In the other 31 % , usually only one of the parts was not detected or was misrecognized. The ratio of successfully recognized parts in the scenes was about 90 %.

4. KRP scheme

The knowledge representation scheme KRP [9] is based on Petri net theory[8]. Petri nets are a formal model originally used for modeling a large class of systems. Formally, a Petri net is defined as a 5-tuple: $PN = (P, T, I, O, \mu)$, where P is a finite set of places, T is a finite set of transitions, input function I and output function O map transitions to bags of places. The marking μ is used to define the dynamic behaviour of a Petri net. It is very important for inference mechanisms of the KRP scheme, but not for describing objects, so it will not be discussed here.

A Petri net can be represented as a bipartite directed multigraph containing two types of nodes: *places* and *transitions*. Places are graphically represented by circles, while transitions are represented by bars.

The KRP scheme is defined by adding two functions to the definition of Petri nets, so it forms a 7-tuple [9]: $PN = (P, T, I, O, \mu, \alpha, \beta)$, where $\alpha : P \rightarrow D$ is a one-to-one and onto function associating a fact or object to every place, and $\beta : T \rightarrow \Sigma$ is an onto function associating a description of a relationship among facts or objects to every transition. D is the set of concepts used to represent objects and facts from the real world. Σ is the set of concepts used to describe relationships among objects and facts.

In this way, a Petri net (KRP net) can represent objects and the relationships among them. To illustrate the terms used in the definition of the KRP scheme and to show how a KRP net can be used to describe a scene, a simple example is given in Fig. 5.

5. Generating KRP descriptions

As shown in the previous section, the KRP scheme can be used to describe relationships among objects. Having a decomposition of a compound object, the KRP scheme can be used

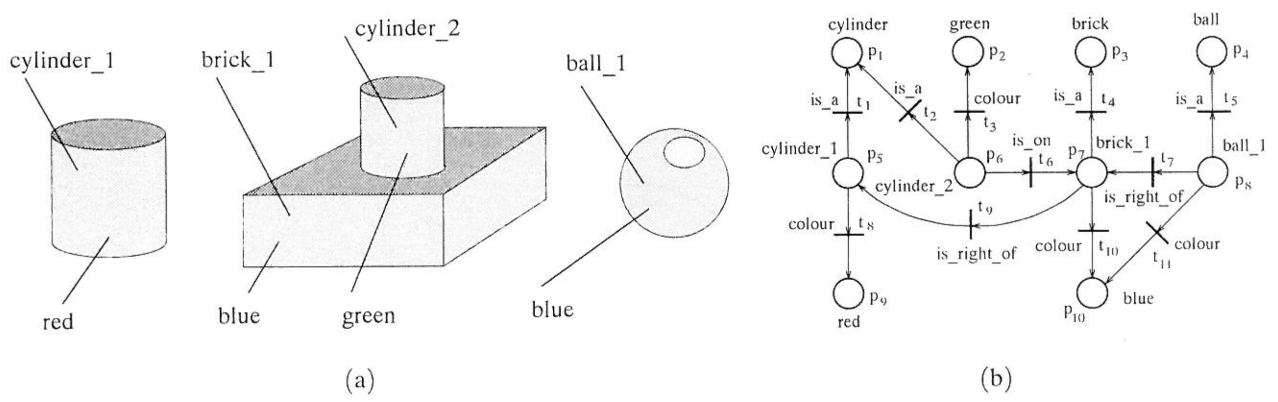


Fig. 5. A simple scene and the corresponding KRP net.

to describe the relationships between parts and to obtain the object description.

To obtain a description based on such a decomposition, the relationships among the detected parts have to be computed and represented by the KRP scheme. Each of the detected parts is represented by a place. Each class of parts is also represented by a place. The type of a particular part is expressed by a hierarchical link, connecting the place representing the part and the place representing the class, through a transition representing the *is_a* relation (e.g. *'d_002 is_a building_part_3'*). The models contain all attributes used to describe the parts. The main problem is to define the set of relations used to describe the relationships among the detected parts.

The positions and orientations of the detected parts are known, so various descriptors can be computed to represent the relationships among them. The generated descriptions should be used as inputs to the KRP knowledge base, and they should appropriately represent the structure of objects in the scene. Similar objects should produce similar descriptions, invariant to position, orientation and scale. We tend to obtain a "natural" description i.e. to approximate the description generated by a human. The choice of relations to be used is quite subjective, but we feel that they should express whether two objects *touch* each other, *overlap* or *cross*. The relative positions can be expressed by relations *left, right, up, down, about middle* etc. Such a description would not contain numerical values, but *symbolic descriptors* representing ranges of values.

In the implemented description subsystem, we express only the relationships between the parts which are connected. Because every object is composed of a connected set of parts, the relationship among two distant parts can be determined indirectly. To simplify the computation of the relationships between parts, we define a *local coordinate system* for each model of building parts. It is defined for the building part in the base position, i.e. in the position in the model. When a part is detected in the scene, the linear transformation is computed which transforms the model into the scene. The same transformation is used to transform the local coordinate system. The local coordinate system is in fact the bounding rectangle, which defines the terms *left, right, up, down.* It also determines the *characteristic axis*, used to express the angle between parts.

Finding the relationship between two objects is simplified by computing the relative positions of the respective transformed rectangles. If the parts don't intersect or touch each other, the relationship is not determined. Otherwise, the position of the join is computed and expressed in both local coordinate frames, using symbolic descriptors. Each side of a local coordinate frame is divided into five regions. For example, the left side regions are: *left_up*, *left_up_around_quarter*, *left_around_middle*, *left_down_around_quarter* and *left_down*. These descriptors embody the symbolic coordinates, which determine positions in a local coordinate frame. Besides the relative position of the parts, the relative orientation is computed. The angle is also discretized into regions with symbolic descriptors. The step is 15°, so the symbolic descriptor *angle_about_30* covers the angles from 22.5° to 37.5°. Another descriptor represents the relative size of the overlapping area. Its values are *overlap_very_small*, *overlap_small*, *overlap_average*, *overlap_large*, *overlap_very_large*. To describe the relationship between two rectangles, we use descriptors *crossing* and *join*, and specify the position in local coordinate frames. For each pair of parts, two sets of symbolic descriptors are computed, regarding each part as a reference. Such a description preserves information, i.e. the scene can be reconstructed. Of course, the quantization introduces some error. To illustrate the procedure, a simple example is given.

In Fig. 6a a simple object consisting of 3 parts is shown. The local coordinate frames are also shown. The arrows point the direction left-right. The system detected the parts $d001$ and $d002$ as being of type $m1$, and the part $d000$ of type $m0$. The system determined the relations between the detected parts and formed the corresponding KRP scheme, shown in Fig. 6c. The relations are associated with the transitions, e.g. t_1 describes the position of the part $d000$ relatively to $d001$: *crossing_up_down; left; overlapping_very_small; angle_about_315*. The other transitions have the following meanings: t_2 = *join_down; around_middle; overlapping_average; angle_about_45*, t_3 = *crossing_up_down; right; overlapping_very_small; angle_about_225*, t_4 = *join_up; left_around_quarter; overlapping_average; angle_about_135*. Fig. 7 shows another example of KRP description generation.

6. Concluding remarks

The implemented module for generating the object descriptions is intended to be a component of a subsystem for automatic knowledge acquisition or learning of a knowledge-based computer vision system. The generated KRP structures describe the objects in the scene and serve as inputs to a knowledge manipulation module. In this way the automated building and updating of KRP knowledge base is enabled. In the scene interpretation procedure, the implemented module provides the automated generation of the KRP structures describing the planar objects in the scene which are to be recognized.

The description subsystem consists of two modules. The first module divides the objects in the scene into parts and the second determines the relationships between the detected parts and builds the corresponding KRP structures. This article also shows how the KRP scheme can be used to describe the decomposed objects. The components of the KRP scheme are identified, and a set of relations used to describe the object structure is proposed. The procedure is illustrated on a simple example.

Future work should include integration of the implemented description module with the knowledge manipulation module and extensive testing of its suitability to the tasks in a real computer vision system. It is to be expected that some adjustments will have to be made to tune the modules for working together.

Other directions for future work should be advancing the object decomposition subsystem to

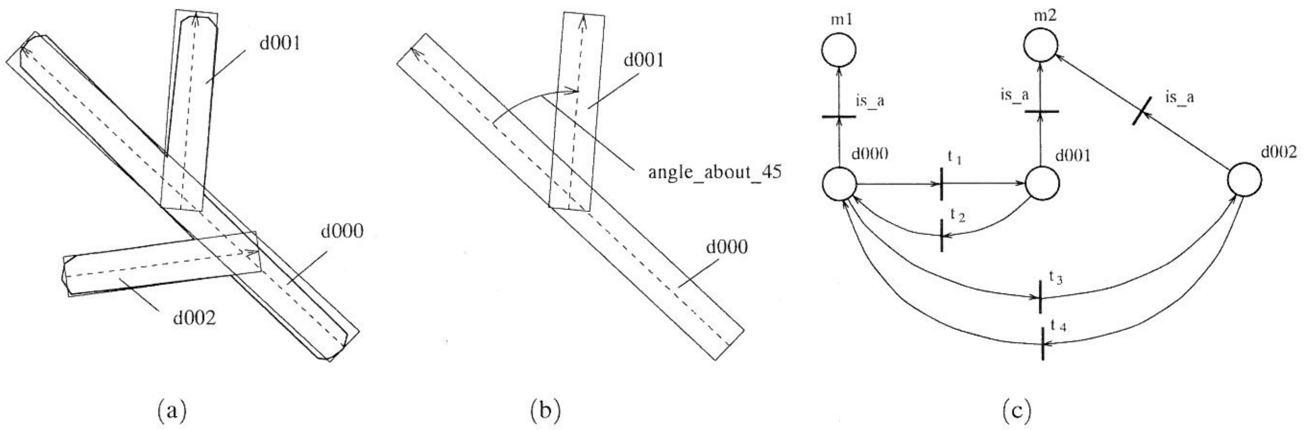
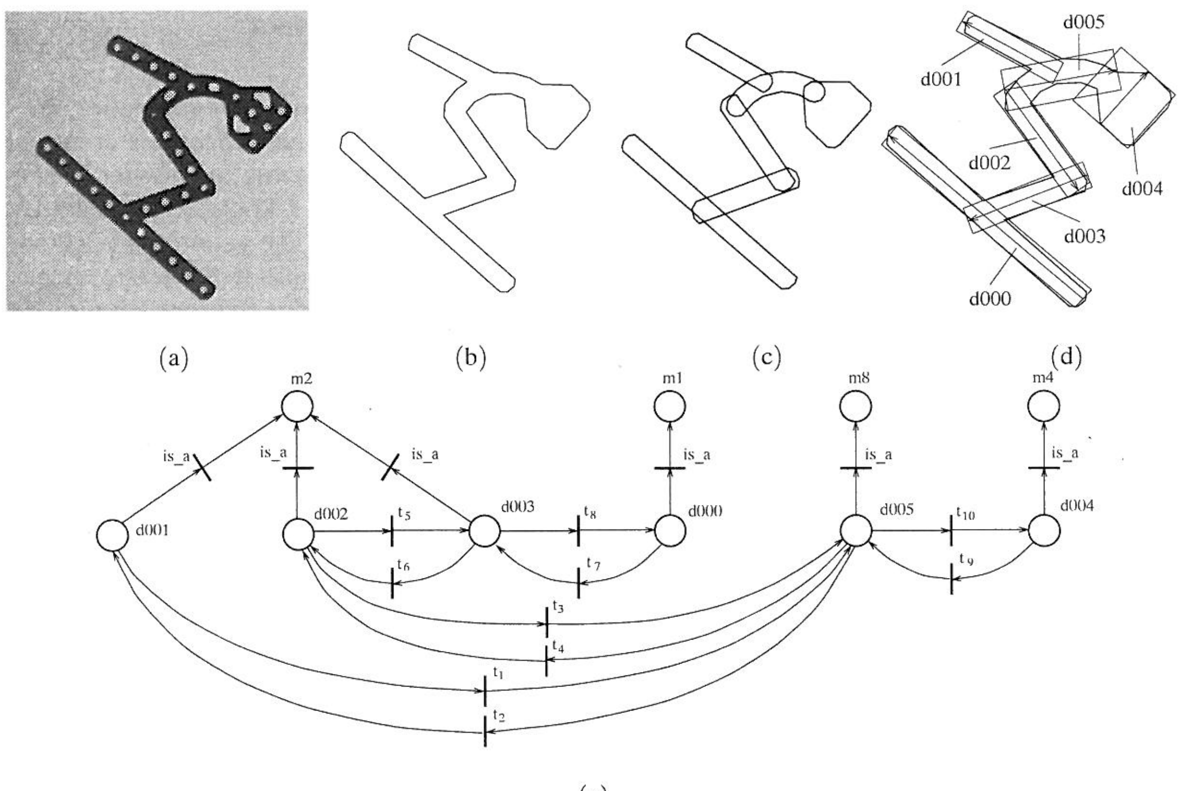


Fig. 6. An example for representing the relationship between parts and the generated KRP scheme.



t1: join_up; left_around_quarter; overlap_small; angle_about_225
t2: crossing_up_down; left; overlap_small; angle_about_135
t3: join_down; left; overlap_small; angle_about_60
t4: join_up; left; overlap_small; angle_about_300
t5: join_down; left_around_quarter; overlap_average; angle_about_255
t6: join_down; right; overlap_average; angle_about_105
t7: crossing_up_down; right_around_quarter; overlap_small; angle_about_60
t8: join_down; around_middle; overlap_average; angle_about_300
t9: join_down; right; overlap_small; angle_about_315
t10: join_up; around_middle; overlap_small; angle_about_45

(f)

Fig. 7. An example of KRP object description. (a) Grey-scale input image. (b) Object contour. (c) Decomposition into parts. (d) Analyzing the relationships among the detected parts. (e) Generated KRP net. (f) Transitions representing the determined relations.

increase its flexibility and robustness, aiming at the ability to deal with the "natural" scenes, and exploring the possibilities of using some similar techniques in three-dimensional scene analysis.

References

[1] BIEDERMAN, I. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing 32*, 1 (1985), 29–73.

[2] BRADY, M., AND ASADA, H. Smoothed local symmetries and their implementation. *Int. Journal of Robotics Research 3*, 3 (1984), 36–61.

[3] CHO, K., AND DUNN, S. M. Hierarchical local symmetries. *Pattern Recognition Letters 12*, 6 (1991), 343–347.

[4] CONNELL, J. H., AND BRADY, M. Generating and generalizing models of visual objects. *Artificial Intelligence 31*, 5 (1987), 159–183.

[5] FLECK, M. M. Local rotational symmetries. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Miami Beach, Florida, 1986), Computer Society Press, pp. 332–337.

[6] KALAFATIĆ, Z. Describing the compound two-dimensional objects using models and local symmetries. Master's thesis, Elektrotehnički fakultet, Zagreb, 1994. In Croatian.

[7] LEYTON, M. Symmetry-curvature duality. *Computer Vision, Graphics and Image Processing 38*, 3 (1987), 327–341.

[8] PETERSON, J. L. Petri nets. *Computing Surveys 9*, 3 (1977), 223–251.

[9] RIBARIĆ, S. Knowledge representation scheme based on Petri net theory. *Int. Journal of Pattern Recognition and Artificial Intelligence 2*, 4 (1988), 691–700.

[10] ROM, H., AND MEDIONI, G. Hierarchical decomposition and axial shape description. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-15*, 10 (1993), 973–981.

Contact address:
Zoran Kalafatić, Slobodan Ribarić
Faculty of Electrical Engineering and Computing
Unska 3, Zagreb, Croatia
e-mail: zoran.kalafatic@fer.hr

ZORAN KALAFATIĆ received the B.Sc. and M.Sc. degrees in Electrical Egineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 1990 and 1994 respectively. Currently he is a doctoral student, working as a researcher at the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb. His interests include Image Processing, Computer Vision and Pattern Recognition.

SLOBODAN RIBARIĆ received the Dipl.-Ing., M.Sc. and Doctorate degrees all in Electrical Engineering from the Faculty of Electrical Engineering, University of Ljubljana, Slovenia in 1974, 1976 and 1982, respectively. He is Full Professor of Computer Science and Engineering at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. His interests include Computer Architecture, Image Processing, Computer Vision and Pattern Recognition. Prof. Ribarić is author of more than seventy papers. He is also the author of four books and co-author of one. Ribarić is a member of the IEEE, and the Croatian Association of Electrical Engineers.