RECOGNIZING ARTICULATED OBJECTS IN ROBOTIC TASKS

Silvia Anton, Theodor Borangiu, Florin Daniel Anton

University Politehnica of Bucharest, Dept. of Automation and Applied Informatics
313, Spl. Independentei sector 6, RO-060032, Bucharest
E-mail: tunaru@cimr.pub.ro

Abstract: *The paper presents a graph-based method for articulated object recognition and handling using skeleton computation. Handling articulated objects is a difficult task because one cannot define a default grasping position relative to the centre of mass for a recognized object. The method presented in the following pages is a possible approach in automated assembly/sorting tasks using industrial vision-robot systems for articulated objects where the articulated objects can be recognized irrespective to the different possible angles between theirs articulated segments and can be grasped without any risk of collision.*

Keywords: *artificial intelligence, robot vision, image processing, skeletonization, articulated object recognition.*

1. INTRODUCTION

It is known that the *skeleton* is frequently used in image analysis as a space domain *shape descriptor*. The difficulty with shape in this case is that what we are trying to abstract or represent cannot be generalized: it seems rather to strongly depend on the usage for which we intend making available the resulting representation (e.g. automatic visual inspection of parts, guidance vision for robots, tracking objects on moving scenes).

The *feature space* is used in cluster analysis and pattern identification. In this respect, a particular object is a *point* in the feature space, while a particular feature f_i of an object O_j, $j = 1,..., o$ is the projection of that object onto the axis X_{fi} of the $\mathbf{X}_f$ space (see Fig. 1).

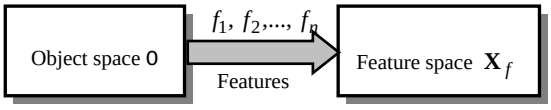

Fig. 1. Mapping from the Object Space to the Feature Space.

A basic approach to representing the structural shape of a region is to reduce it to a graph. This reduction may be carried out by determining the *skeleton of the region* via a thinning (also called skeletonizing) algorithm.

2. SKELETONIZATION AND PRUNING

A classical technique in pattern recognition is to work on a contour representation of the object in order to extract the features which will be used to classify it. This method is widely used in object recognition but is not working if the objects have articulated levers. An alternative approach consists in representing the object by a pattern obtained by *thinning* it as much as possible. The result of this process is a set of idealized lines which is called the *skeleton* or *medial axis* of the input pattern, and is the thinnest representation of the original pattern that preserves its topology, and can be therefore used for identification.

The methods allowing obtaining the skeleton are called *thinning* or *skeletonization*. The detection of end points, junction points and curve points of the medial axis is important for a structural description that captures the topological information embedded in the skeleton.

Skeletonization is a technique used for reducing foreground regions in a binary image to a skeletal remnant that largely preserves the extent and connectivity of the original region, while throwing away most of the original foreground pixels. To see how this works, imagine that the foreground regions in the input binary image are made of some uniform slow-burning material [8]. Light fires simultaneously at all points along the boundary of this region and watch the fire move into the interior. At points where the fire travelling from two different boundaries meets itself, the fire will extinguish itself and the points at which this happens form the so called `quench line'. This line is the skeleton.
Another way to think about the skeleton is to considerit as the loci of centres of bi-tangent circles that fit entirely within the foreground region being considered [7], [1]. Fig. 2 illustrates this principle for a rectangular shape.

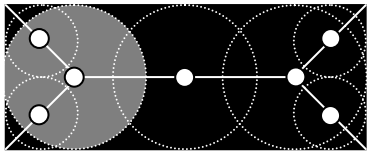

Fig. 2. Skeleton of a rectangle defined in terms of bi-tangent circles.

Most of the existing algorithms to generate digital skeletons produce a non-connected skeleton, which is useless for shape description applications since homotopy is not preserved and characteristic points such as junction points or endpoints in the continuous case are lost.
On the contrary, a thinning process guarantees the condition for obtaining one-pixel thick and connected skeletons [9].

Thinning is a morphological operation that is used to remove selected foreground pixels from binary images, somewhat like erosion or opening [4], [6]. It can be used for several applications, but is particularly useful for skeletonization. Like other morphological operators, the behaviour of the thinning operation is determined by a structuring element. The thinning operation is related to the hit-and-miss transform and can be expressed quite simply in terms of it. The thinning of an image I by a structuring element J is:

$$thin(I, J) = I - hit - and - miss(I, J) \quad (1)$$

where the subtraction is a logical subtraction defined by $X - Y = X \cap NOTY$.

In everyday terms, the thinning operation is calculated by translating the origin of the structuring element to each possible pixel position in the image, and at each such position comparing it with the underlying image pixels. If the foreground and background pixels in the structuring element exactly match foreground and background pixels in the image, then the image pixel underneath the origin of the structuring element is set to background (zero). Otherwise it is left unchanged. Note that the structuring element must always have a one or a blank at its origin if it is to have any effect. The choice of the structuring element determines under what conditions a foreground pixel will be set to background, and hence it determines the application for the thinning operation.

The effects of a single pass of a thinning operation over the image are further described. In fact, the operator is normally applied repeatedly until it causes no further changes to the image (i.e. until convergence). Alternatively, in some applications, e.g. pruning, the operations may only be applied for a limited number of iterations. This effect can be achieved using morphological thinning by iterating until convergence with the structuring elements shown in Fig. 3, and all their 90° rotations (4×2 = 8 structuring elements in total).

In fact what we are doing here is determining the octagonal skeleton of a binary shape the set of points that lie at the centres of octagons that fit entirely inside the shape, and which touch the boundary of the shape at least two points. This skeletonization method is guaranteed to produce a connected skeleton.

	0	0
1	1	0
	1	

0	0	0
	1	
1	1	1

Fig. 3. Structuring elements for skeletonization using morphological thinning.

At each iteration, the image is first thinned by the left hand structuring element, and then by the right hand one, and then with the remaining six 90° rotations of the two elements. The process is repeated in cyclic fashion until stability is reached [10] (none of the thinnings produces any further change) and the object is reduced to a set of one-pixel width connected lines. Usually, the origin of the structuring element is at its centre.

If the skeleton is considered as a connected graph, each vertex can be labelled as an end point or a junction point, while an edge is just made of curve points. In this way a skeleton X of an object O can be considered as the union of the end points, the junctions and the curve points of X, i.e.

$$X = EndPo\,\mathrm{int}\,s(X)\ \mathrm{Y}\ JunctionPo\,\mathrm{int}\,s(X)\ \mathrm{Y}\ CurvePo\,\mathrm{int}\,s(X) \tag{2}$$

As a consequence, a skeleton X can be partitioned into N branches $S_i(X), i = 1,..., N$, i.e.

$$X = \mathop{\mathrm{Y}}_{i=1}^{N} S_i(X) \tag{3}$$

Fig. 4 shows the result of this thinning operation on a simple binary image.

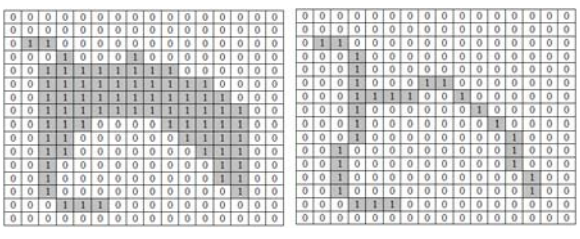

Fig. 4. Example skeletonization by morphological thinning of a simple binary shape, using the above structuring elements.

As with thinning, slight irregularities in a boundary will lead to spurious branches in the final image which may interfere with recognition processes based on the topological properties of the skeleton. Despurring or pruning can be carried out to remove spurs of less than a certain length but this is not always effective since small perturbations in the boundary of an image can lead to large spurs in the skeleton [5].

Skeletons produced using the above method often contain undesirable short spurs produced by small irregularities in the boundary of the original object. These spurs will be removed using pruning, which is in fact just another sort of thinning. The structuring elements for this operation is shown in Fig. 5.

0	0	0
0	1	0
0		

0	0	0
0	1	0
		0

Fig. 5. Structuring elements used for Pruning.

At each thinning iteration, each element must be used in each of its four 90° rotations. Pruning is normally carried out for only a limited number of iterations to remove short spurs, since pruning until convergence will actually remove all pixels except those forming closed loops.

3. RECOGNITION OF ARTICULATED OBJECTS

In order to be able to recognize an object with articulated levers, let us compute the skeleton. Depending on the lighting conditions the image can be noisy and the result must be given as a input to a pruning algorithm, if pruning loops are chosen depending on the light. The resulting skeleton can be approximated with a graph which is modifying depending on the angle of the levers (see Fig. 7), but the length of branches is the same. If the angle between two branches is very small or zero depending of the articulations positions, two or more graph branches are combined into one. In this case the structure of the graph is modified.

Following this principle the algorithm presented in Fig. 6 was developed in order to discriminate between articulated objects:

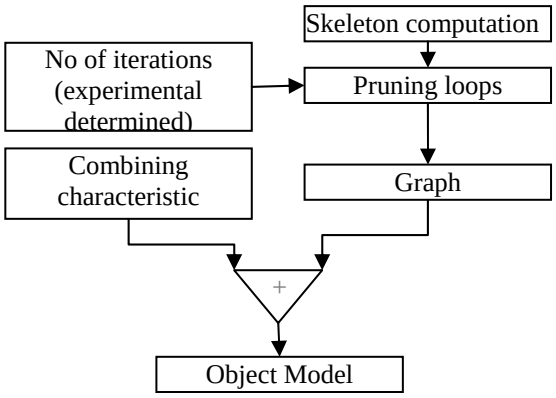

Fig. 6. Model construction.

The algorith for model construction based on skeleton computation is given below:

1. The skeleton is computed and a number of pruning loops are executed, depending on the light conditions.
2. A graph is computed, and the user selects each branch and assigns a combining characteristic; the user will also specify whether an angle between two branches will remain constant or will be variable and whether there exists another pair of branches which will form an angle with the same opening.
3. If a branch is positioned between two or more branches it will be ignored, but the length of the branches which are connected with this one will be increased with half of the middle branch.
4. If a branch is smaller than the number of the pruning loops it will not be considered.

Fig. 7. The skeleton for different angles of the articulations.

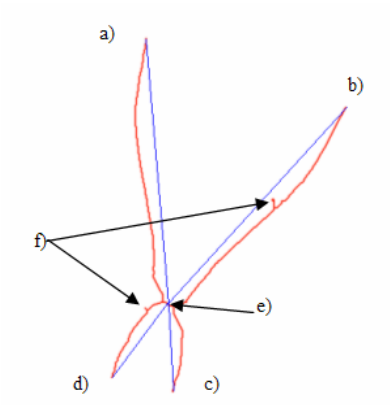

Fig. 8. The computed features.

To be more clear let's take the example from Fig. 8:

Step 1: The skeleton is computed and 3 pruning loops are executed.

Step 2: The graph is computed as follows:

a) The first branch is detected and the length is measured (branch a=273 pixels= 136.5 mm) (the ratio mm/pix=0.5)
b) The branches are measured in clockwise order, resulting: (b=274 pixels=137 mm, c=93 pixels=46.5 mm, d=90 pixels = 45 mm)
c) The user defines angles (blue lines) and dependencies between branches: (angle formed by a,d = angle formed by c,b,), and also assigns combining characteristics: (branch a=always single, b=always single, c and d can be connected).

Step 3: The middle branch (e) is detected and measured (e=10 pixels=5 mm), the size of the connected branches is raised with 2.5 mm, resulting (a=139 mm, b=139.5 mm, c=49 mm, d=47.5 mm).

Step 4: Branches f) are detected and ignored being considered as noise.

After processing the skeleton results a set of data [3] which has the following structure:

- Pruning loops: 3,
- Number of branches: 4,
- Branch sizes: 139, 139.5, 49, 47.5,
- Angles: A1, A2, A3, A2 (A1=angle between a and b, A2 appears two times because the angles between b,c and d,a are equal)
- Combining Characteristics: s, s, c1,c1 (a and b are single (s), and c and d can be combined (c)).

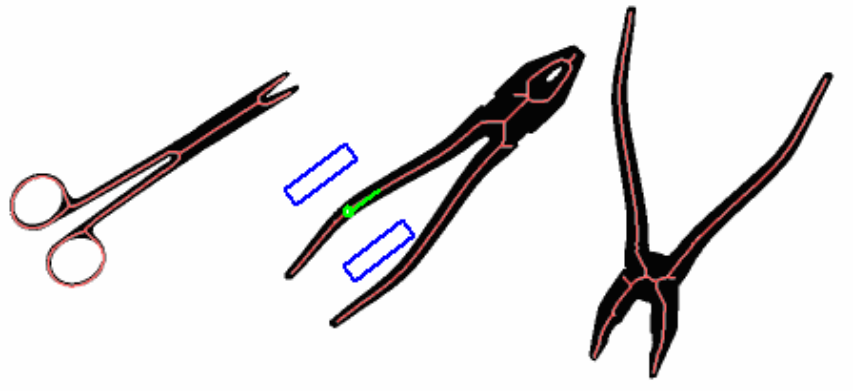

Fig. 9. Positioning the gripper.

In the recognition phase a similar structure is created the difference being the order of branches in the structure. After comparing the branch sizes the recognition system can easily find the corresponding branches from the learned structure and the structure created at the recognition step and compare them.

4. GRASPING ARTICULATED OBJECTS

In most robot applications, after an object is recognized the vision-robot system will try to grasp the object and place it in a desired location. Dealing with articulated objects is more difficult because the grasping position depends on the articulations and not on the centre of mass; in such situations two grasping methods can be defined:

The first method allows you to identify a branch of the skeleton and the gripper will grasp always on the same branch relative position (let's say on the middle of the branch) and with an orientation which depends on the orientation of the branch. This method is simple but cannot be applied always because the grasping may lead to collisions with:

- other branches of the articulated object currently identified, due to the fact that the reference branch may have a variable position relative to other branches of the same skeleton (collision with the recognized object).
- other objects placed too close relative to the recognized object.

The second method does not impose a predefined grasping position, but allows the system to compute at run time a grasping position acording to an algorithm we have developed which avoids both the collisions with the other components of an object and with the objects close to the recognized one.

The algorithm we have designated as "clear grip" consists in finding a grasping position which will allow to move the object into a desired position, and has the following structure:

1) For each branch of the skeleton
2) Find a range of connected pixels which are at least L/2 away from the branch extremity (L is the length of the gripper fingerprint)
3) For each pixel,
4) Find a direction tangent to the skeleton, dir1
5) Place the gripper opened on the image, with the centre on the current pixel and the orientation dir1
6) Define a rectangular area which is given by the movement of the gripper fingerprints from the maximum opening to the opening where the gripper enters in contact with both its fingers with the edges of the object (see Fig. 9). Here, a modified gauge predicate named calliper is applied, which it will be discussed later.
7) If the areas defined at step 6 contain black pixels, or the distance between the two fingerprints is less than the minimum opening the grasping position is not valid, go to step 3.
8) If the area does not contains black pixels then repeat the algorithm from the step 5 varying dir1 with $\pm 25^{o}$, and choose the grasp having the minimum distance between fingers at the position of contact with the object.

The function of the gauge predicate *caliper* is further described referring to how a robot's gripper picks up an isolated blob-like object [2]. The centre of the gripper is placed at $[X_1, Y_1]$,

and has two parallel jaws of length L($L \geq 0$)(see Fig. 10).

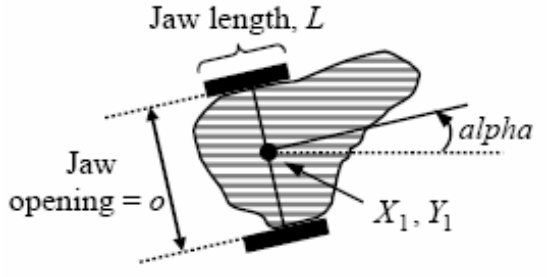

Fig. 10. The caliper predicate.

The gripper opens by moving the jaws along a line inclined at angle *alpha* to the horizontal (the direction of x_{vis} the abscisse of the frame attached to the image plane). The opening of the jaws as they close to grasp the object is given by *o* – the output data of the caliper predicate.
There are two possible modes of operation of calliper, depending upon whether or not the object can slip over the table top as it is being picked up (the robot arm is assumed to be absolutely rigid). If the object slips as it is being picked up, then it will eventually touch both jaws. The opening *o* is then merely a function of the object's geometry and orientation. (in our case the object keeps the stablle position and the gripper slips).

5. CONCLUSIONS

The presented skeleton based recognition and grasping method for articulated objects was tested in the Robotics Laboratory of the University Politehnica of Bucharest using an Adept Cobra 600 TT robot, with AVI board and a Panasonic GP-MF602 camera.
The performance of the algorithm depends on the following factors:

1. The number of black pixels from the image – this factor affects mostly the skeletonization algorithm
2. The lighting conditions – the algorithm is highly dependent on the lighting conditions. If the light is not properly calibrated the pruning algorithm must be executed many times in order to obtain a good image. Also if the pruning is executed too many times it is possible to delete good pixels from the skeleton. The pruning loops are given by the user after the system was calibrated and also after few tests are executed.
3. The complexity of the object – if the object is very complicated (has many nodes) the execution of the matching algorithm will take more time.

During the tests, the articulated objects where recognized and grasped correctly in a proportion of 90%. When the objects were touching, the recognition was not working, but this is not affecting the grasping. Both grasping methods previously discussed were used for articulated objects: the one in which the grasping position (static) is attached to a branch (position and orientation), and another computed at run time (dynamic) by inspecting each branch in order to find a valid grasping position.

Future work aims to develop methods to enhance the performance of the recognition and grasping algorithms (execution speed), a method to determine automatically the pruning loops and also a method to discriminate between objects which superpose or touch.

REFERENCES

[1] Ballard, D., Brown, C., Computer Vision, Ed. Prentice-Hall, Chap. 8, 1982.
[2] Borangiu Th., Intelligent Image Processing in Robotics and Manufacturing, Publishing House of Romanian Academy, Bucharest 2004.
[3] Borangiu, Th., Anton, F.,D., Tunaru, S., Dogar, A., Object recognition method using a network oriented skeleton computation, Proceedings volume of the 14th Int. Conf. Robotics in Alpe-Adria-Danube Region RAAD'05, Bucharest, May 26-28, 2005.
[4] Davies, E., Machine Vision: Theory, Algorithms and Practicalities, Ed. Academic Press, pp. 149-161, 1990.
[5] Gonzalez, R., Woods, R., Digital Image Processing, Ed. Addison-Wesley Publishing Company, pp 518 – 548, 1992.
[6] Haralick, R., Shapiro, L., Computer and Robot Vision, Vol. 1, Ed. Addison-Wesley Publishing Company, Chap. 5. pp 168 – 173, 1992.
[7] Image Processing Learning Resources, http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_ top.htm
[8] Jain, A., Fundamentals of Digital Image Processing, Prentice-Hall, Chap. 9, 1989.
[9] Serra, J., Image analysis and mathematical morphology, Ed. Academic Press, 1992.
[10] Soille, P., Morphological image analysis: principle and applications, Springer, 1999.