MVA '92 IAPR Workshop on Machine Vision Applications Dec. 7-9,1992, Tokyo

PRACTICAL CONSIDERATIONS ABOUT A 2D ALGORITHM FOR OBJECT ORIENTATION IN INDUSTRIAL APPLICATIONS

Curt L Orbert, Ewert W Bengtsson and Bo G Nordin
Centre for Image Analysis, Uppsala University

Mailing address: Lägerhyddvägen 17, S–752 37 Uppsala, Sweden
Phone: +46 18 18 34 71 *Fax:* +46 18 55 34 47 *Email:* Curre@cb.uu.se

ABSTRACT

We here present a solution to a common problem in IMV, i.e. to identify and estimate the orientation of touching mechanical parts on a plane surface.

After an initial thresholding step, we use a distance transformation to create a distance map. We separate the objects by using watershed segmentation on the distance map. Some objects may be segmented into several parts. For every segment and every hole we calculate the centre of gravity for its surrounding edge pixels. By this we have an estimation of the orientation of the objects. For the objects that consist of only one segment without holes, we construct a circle around the centre of gravity. We plot the values of the distance map on this circle line as a function of the angle and identify its maxima and minima. For the overall control algorithm we use fuzzy logics.

To verify the identification and to get a better estimation of the orientation of the objects, we finally do an edge matching using the distance map which gives us quantitative measurements of how well the edges match. This will give us more accurate estimations than can be achieved by statistical methods.

Laboratory tests have shown that our algorithm will perform quite well on the shop–floor under certain restrictions about the distance between the camera and the objects.

MATCHING

Edge matching : In order to determine the orientation of objects we want to match the edges of an object in an image with the edges of the same object in a reference image. We do that by translating and rotating one of them until its edge pixels match the edge pixels of the other image as well as possible. As measurements of the distances between the two sets of edge pixels we use the method reported by Barrow et al [1]. One of the edge images is first transformed into a distance map with some distance transformation, i.e. the chamfer algorithm, see Borgefors [2].

To compute the distance transformation for an image with edges, we calculate the distance from every pixel to the nearest edge pixel. Let us call the list of edge pixel coordinates we get from transforming and rotating them the **Pattern Set**.

We define an **Edge Matching Function** (EMF), which we use to measure the similarity between two edges. Since we are dealing with distances a weighted Euclidean norm seems to be a natural choice. The value of the EMF gives an estimation of the mean distances between the edge and the pattern.

Iterative search method : Since the iterative search for minimum is the most time consuming part of the algorithm, it is important to find an efficient search algorithm. Finding a good match is the same as finding a minimum in the Edge Matching Function, i.e. we have a nonlinear optimization problem. We shall write the problem as an overdetermined nonlinear system and then use the Gauss–Newton method to solve it.

The Gauss–Newton method assumes that we have several functions, all approximately equal to zero and that each of them is a function of several variables. The Gauss–Newton method will try to minimize every single distance between the Pattern Set and the edges of the object, in contrast to the EMF which will be based on the square root of the mean value of the squares of the distances. This will cause a small bias in the optimal position.

Ideally we will continue an iteration until ΔEMF is equal to zero, but since the Pattern Set always is moved to an exact pixel position a value of zero is normally not possible to achieve. If we had a continuous function to minimize instead, the value of ΔEMF would decrease, when approaching a minimum. In our case it will decrease until it is close to the minimum, when suddenly it will increase. This is due to the change in shape of the Pattern Set on translation and/or rotation. We can use that as a sign to stop the iteration, because from that point the behaviour of the iteration is hard to predict.

Our algorithm shortens the computation time considerably compared with the fastest algorithm found in the literature. The small bias can be corrected by using some other method as a last correcting step. Even with this change the method is more than twice as fast as other methods. For more details about this iterative edge matching method see Orbert [3].

Ideas for improvement : Even with a very good search algorithm we will not be able to reach the global best match from every start orientation, but only from

those who are in the neighbourhood of the global best match. How close we have to be is determined by the shape of the object. The very first and generally safe method is to check every orientation all over the image in order not to miss the global best match. This would however lead to extremely long cpu time.

The things that are invariant between the two edge images will depend on the situation we are working with. Our algorithm is intended for a robot which is to pick up some mechanical parts from a horizontal and plane surface for an assembly task. We can also manage the camera the way we want and hold it in a known position perpendicular to the surface. We also assume no overlapping of the mechanical parts in the images, but we will allow them to touch each other.

With all these arrangements we will have an edge matching of objects that are just translated and rotated in the image plane. We will try to calculate the orientation of an object by using characteristics from its outer edge and holes. It will be sufficient to do this accurately enough to be good as a start orientation for the iterative search for the global best match.

WATERSHED SEGMENTATION

A common problem with binary images generated by a segmentation algorithm is to split the domains into smaller ones in a controlled way. While it is easy to do this interactively by drawing lines in the image it is a much more difficult task to formulate rules for this operation in a computer language and thus automate the procedure. The domains that need to be split may be caused by touching parts or by objects with complex shapes.

We have used a watershed segmentation algorithm reported by Vincent and Soille[4] as the base for our algorithm, in fact we have just done some small, but important, modifications of their algorithm for our use of this method.

The algorithm of Vincent and Soille is fast because it sorts the pixels according to their values. After the sorting phase all pixels of the objects belong to a cluster depending on their distance to the nearest edge pixel. Vincent and Soille have in their algorithm tried to label as many pixels as possible as belonging to the different segments and to avoid labelling pixels as belonging to a watershed line. In many applications it is more important to know the watershed lines exactly than the segments.

We can easily change the algorithm to mark the watershed lines better, by adjusting the rules for labelling pixels. We just let the pixel belong to a watershed line, if at least two neighbours belong to different segments.

We have now made the watershed lines as connected as possible, but we have too many different segments. By examining the value of each pixel that has been labelled as watershed and comparing it with the minimum value of the most recently labelled segment of those that are neighbours to the watershed pixel, we will get a difference between them. This difference should exceed a certain threshold value, otherwise we will relabel all pixels belonging to that segment including the watershed pixels.

In some applications (eg our) it is important that the watershed lines are identical regardless of the orientation of the object. In general they will not be identical due to the digital approximation of the spatial domain of the image and thus cannot be corrected in the procedure of the watershed segmentation. We will instead correct it by selecting the pixels of the contour which are closest to the pixels at each end of the watershed line. From these two pixels on each side of the "waistline" of the object we perform an iterative search for the two pixels that have the shortest distance between themselves, still being on each side of the object.

Our modifications to the algorithm of Vincent and Soille will give us an algorithm that preserves the shape and the number of segments better. In figure 1 below we can see how our modificated algorithm will work on an image with three touching objects.

PRELIMINARY ORIENTATION

After we have extracted the objects in the image and applied the watershed segmentation algorithm described above, we have to identify the objects in the image. For each segment we calculate the maximum value of their distance maps and their peripheries. If the objects contain more than one segment, we calculate the distances between their centra of gravity. For objects with holes we calculate the length of the peripheries of the holes and the distance between the centra of gravity of the holes and the centre of gravity of the segment.

We have used fuzzy sets, first presented by Zadeh[5] in 1965, to implement an identification procedure. In this we calculate a normally distributed fuzzy set around the value of each parameter mentioned above and use the minimum rule to get the fuzzy number of the probability that we have a certain object in the image. For more details about the theory of fuzzy sets and its applications we refer to Terano et al[6].

When we have identified an object, we must estimate a good orientation of it in order to reach the global minimum of the Edge Matching Function. For objects consisting of two or more segments the centre of gravity for each of them will be enough to get a good estimate of its orientation. Also for objects with a hole we can easily find an estimate from the centre of gravity of the hole and that of the segment.

To be able to estimate the orientation of objects consisting of just one segment without holes we have developed a new tool, which we call the **Circle Profile**. We create this by constructing a circle that has a radii equal to the maximum value of the distance map inside the segment and is centered at the centre of gravity of the outer contour of the segment. Then we plot the value of each pixel in the distance map that belong to the periphery of the circle as a function of the angle between the radii drawn out to the pixel and the horizontal axis of the image.

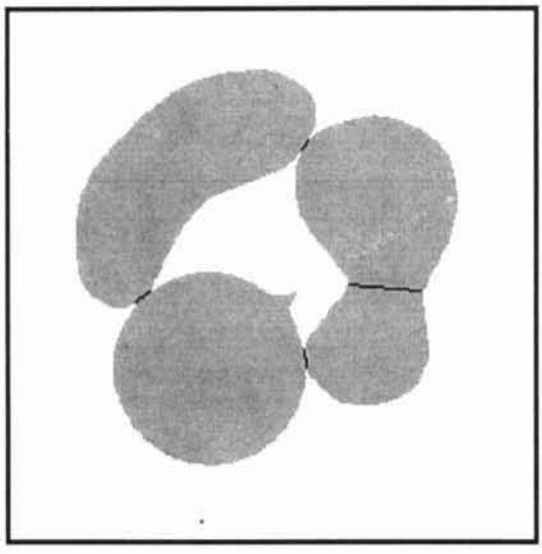

Figure 1. *An image of three objects, where the pixels of the watershed lines are shown in black.*

The Circle Profile will have a unique shape for every object. We use the position and value of the maxima and minima of the Circle Profile in addition to the calculated value of the derivatives around those points as parameters for estimating the orientation.

A perfectly square object, for example, will of course not give one unique rotation, instead we get four symmetrical rotations. In cases like this we have to accept that there will be more than one possible start orientation for the matching algorithm.

Another case is when we have a perfectly circular disc, which will give us no unique direction in rotation at all. But in this case we know that it is a circular disc we are working with, because no other object will act like that.

RESULTS

We have done many laboratory tests on objects with different shapes and in varying degrees of contact. The results have been similar to the example in figure 1 and the figures in table 1, which we therefore consider to be representative for the algorithm.

Table 1. ***Figures of the example shown in figure 1***

Object	Θ_0	EMF_0	X	Y	Θ	EMF
Disc	87.3	0.76	0.3	-0.2	-87.2	0.70
Banana	-167.1	0.93	0.5	-0.7	-169.4	0.60
2 Segm	-103.4	0.83	-0.1	0.2	-104.9	0.65

In the left half of table 1 we have the figures of the preliminary locations and in the right those of the best match. From those figures we can see that the preliminary orientations of the objects are very close to the best matches. The translations shown are calculated from the centre of gravity for each object. For more details about our algorithm see Orbert [7].

LABORATORY EXPERIMENTS

So far we have assumed that the objects are laying on a flat surface, and that the camera is mounted in a position perpendicular to the surface. The intial experiments were done using paper silhouettes scanned by a flatbed scanner (Canon CLC 500), which gives a perfectly orthogonal projection. But what happens under more realistic conditions with 3D objects viewed by a camera? If the distance between the camera and the surface is large the situation will be equivalent to an orthogonal projection, but for use on the shop–floor we need to know how the camera distance is related to the accuracy of the estimate of the orientation for the objects our algorithm gives.

In figure 2 below we can see that the position of the objects in the image determines how much of the objects' sides we see. This is best seen from the cylinder in the upper right corner and the cylinder to the right in the centre, which both have a height of 40 mm. The cylinder in the lower right corner is just 5 mm high, so we can not see the sides of it. The cylinder in the upper left corner is 10 mm high and the one in the centre to the left is 20 mm. All cylinders have the same diameter, 18 mm.

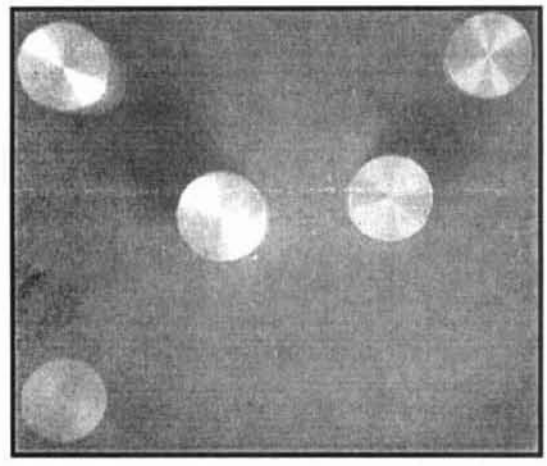

Figure 2. *An image of five cylinders with different heights and distances to the image centre. The distance to the camera from the plane is 46 cm.*

From this image, and also from basic geometry, we can understand that the size of the projection of the sides of the objects in the images is inversely proportional to the distance to the camera, but the size is also proportional to the distance between the object and the centre of the image, and also to the height of the object. We can than formulate an equation consisting of the parameter mentioned above as

$$a = p \times D/(h \times R), \tag{1}$$

where p is the proportionality factor, a is the distance between the optical axis and the centre of the object, D is the distance between the camera and the plane surface the object are laying on, h is the height of the object, and R is the resolution in the image.

We have tested this by applying our algorithm on the cylinders in figure 2 with different distances between the camera and the objects. We must also have in mind the practical use of the value of the EMF in our algorithm, when we choose it. Let us assume that 64 % of the pixels we use in the Pattern Set have the value 0 and the rest have the value 1. That will give us an EMF value of 0.6, a very good match. If we instead have equal amount of zeroes, ones and twos we will end up with a value of appoximately 1.3. Thus we consider the estimate of the orientation performed by the algorithm inaccurate when the value of the EMF is larger than 1.3 for the final match.

With that as a criteria to decide when the distance to the centre of the camera is at its maximum for the cylinders, we could calculate the proportionality factor p in equation 1. The two cylinders in the centre of the image in figure 2 and the one in the upper left corner are placed at distances where the value of the EMF is approximately equal to 1.3.

From our experiments we can conclude that we can estimate the proportionality factor p to 5.5.

PRACTICAL CONSIDERATIONS

Let us consider what it would mean in a real life situation. Let us assume that we have a plane surface sized 0.5 m$\times$ 0.5 m, and an over–looking camera mounted 2 m above the surface. For simplification reasons assume that the resolution of the camera is 1 pixel/mm, and that the object distance from the optical axis is no more than 35 cm. Equation 1 tells us that the height of the object need to be less than 32 mm. A higher camera position will allow even higher objects.

If we want to estimate the orientation of the object still better, we can for example do that by mounting a camera in the robot arm. and iteratively move the camera until the centre of gravity of the object in the image coincides with the centre of the image, and in that way get rid of effects due to varying camera angles. Let us assume the resolution of this camera is 5 pixels/mm. By using equation 1 again we can conclude that we need to move the optical axis of the camera to a position less than 14 mm from the centre of gravity of the object.

Equation 1 expresses the worst case in the sense that it assumes that you really see the sides of the objects. In our experiments to determine the proportionality factor p we had to work hard with light mounted beside the scene to see the sides. This difficulty is also evident in figure 2 above, where the sides of the cylinders are hard to see. With the diffuse light that should be used this effect will be less significant.

In equation 1 we have assumed that the shape of the object change only because the sides of it will be visible in the image. This is not true, because the shapes of the objects will change, even for a very thin object, when it is moved out to the edges of the image, because of the changed viewing angle. A perfectly circular disc will appear oval for example. But to be able to handle 3D objects one should use a camera mounted as far away as possible i.e. with a very narrow viewing angle of the scene as shown above. Because of that we have not been able to measure any change in the shapes of the objects due to this effect.

CONCLUSIONS

Our main purpose with this work has been to show that it is possible to use the edge matching technique with distance maps to determine the orientation of objects for industrial applications. We have shown that our algorithm can manage almost any kind of shape of the objects, if we just can segment the objects from the background. The objects may touch each others.

Our algorithm can not handle objects where the only clue to the orientation is the structure on the surface of the object, for instance a coin. For such objects we need a separate approach to find the rotation using the edges inside the object.

The laboratory test of how well our algorithm will manage on the shop–floor shows that our algorithm can identify and estimate the orientation of objects on a plane surface in most situations. As the above analysis and experiments show we can handle the 3D scenes quite well with our basically 2D algorithm, if we just mount the camera appropriately for the task.

The fact that our method is highly automated and general in the learning of the object, will make it easy to use on the shop–floor.

REFERENCES

1. **H G Barrow, J M Tenenbaum, R C Bolles and H C Wolf, 1977**, *Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching*, Proc of the 5th Int Joint Conf of AI, Cambridge, Massachusetts, 659–663.
2. **Gunilla Borgefors, 1984**, *An Improved Version of the Chamfer Matching Algorithm*, Proc of the 7th Int Conf on Pattern Recognition, Montreal, Canada, 1175–1177.
3. **Curt L Orbert, 1992**, *Iterative Methods for Edge Matching Using Distance Transformations*, Proc of Second Int Conf on Automation, Robotics and Computer Vision, Singapore, CV-3.3.1 – 5.
4. **Luc Vincent and Pierre Soille, 1991**, *Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations*, IEEE Trans on Pattern Analysis and Machine Intelligence **13**(6), 583–598.
5. **Lotfi A Zadeh, 1965**, *Fuzzy Sets*, Information and Control **8**, 338–353.
6. **Toshiro Terano, Kiyoji Asai and Michio Sugeno, 1992**, *Fuzzy Systems Theory and Its Applications*, Academic Press, London.
7. **Curt L Orbert, Ewert W Bengtsson and Bo G Nordin, 1992**, *Automated Recognition and Precise Estimation of Orientation of Objects for Industrial Applications*, SPIE Proc of Intelligent Robots and Computer Vision XI: *Algorithms, Techniques and Active Vision* **1825**, Boston.