VERIFICATION OF GRAPHICAL PRIMITIVES IN GRADIENT DIRECTION IMAGES

Ulf Hönisch, Universität Bremen, Informatik

Postfach 330440, D-2800 Bremen 33

Abstract: Object recognition seems to be a more model driven task than realized in today´s computer systems. So model driven verification of well known objects in special supposed positions becomes an important task. Those known objects may be represented as CAD-models which have been suggested and investigated for object recognition purposes in the past [Bhanu, 1987] [Ikeuchi, 1987] [Gmür, 1988] [Coy, 1989] [Henderson, 1990].We describe a new approach for the recognition of objects represented by CAD-models by verifying their graphical primitives in the direction image. The approach is based on some well known computer graphics algorithms. We use the primitive type as well as position and orientation parameters all calculated from given transformation parameters. The operations are organized to work in a strong model driven interpretation process. The gradient direction image we are working at is much closer to the geometric CAD-models than a captured greyscale image and is close enough to the captured data to prevent the influence of numerous thresholds. The described operations work on 2D primitives in 2D images, nevertheless they may be used in 3D context after a suitable projection.

Key Words: Image Analysis, Model Based Analysis, Graphical Primitives, Verification, CAD Models, Proving Operations

1. INTRODUCTION

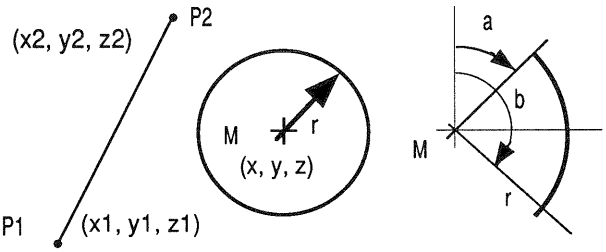
We suppose that human object recognition is a process which use expectation and a priori knowledge in a more rigid way than implemented in existing vision systems. Expectations based on a special context allow us to handle most situations rapidly when they fit reality. But even when an expectation is wrong the context gives enough information to correct it and to continue the process. On the opposite side, recognition of a simple shape in an abstract, isolated test situation may be very crucial. Therefore we may think of recognition as a simulation of this context. Hypotheses of the expected objects arise from their simulations and have to be verified after a projection into the image frame. Understanding recognition as a search problem in the space given by all possible objects and their different positions and orientations the simulation should reduce the space dramatically.

Translating this idea into a technical system leads to a model based approach consisting of two steps. First model and location hypotheses are generated, the second step looks for a successful verification of their projection in the image. While this paper concentrates on the verification task, we assume that there is a given set of hypotheses. The extraction of corresponding interest points and the calculation of transformation hypotheses is described in [Hönisch, 1986] [Huttenlocher, 1987] [Lamdan, 1988].

2. GRAPHICAL PRIMITIVES

For computer aided generation, pictures are handled as complex graphical structures. A structure which can not be subdivided further is called a primitive. Graphics interfaces like GKS, PHIGS and QuickDraw provide operations defined on those graphical substructures. They allow the definition of drawings based on a given primitive set. The representation consists of primitives and their topology. In general, subdivision is not unique because the representation scheme is not unique [Requicha, 1982]. Therefore an object may be represented by different subdivisions leading to different primitive sets.
Machine vision usually starts with the captured picture. Syntactic pattern recognition is based on the structures described above [Fu, 1974]. But the transformation necessary to map a pixel set into a set or structure of primitives is not unique too. So, matching the generated data bottom up against an existing representation can never be precise. If there is a limited number of model and location hypotheses there is an alternative approach by using modelbased proving operations [Coy, 1989]. Proving may be done at different levels somewhere between the captured image and the model [Shirai, 1983]. At the lowest level it is possible to look for the sociability of model instance and picture contents. The latter is not affected by filter or segmentation operations. We are searching for those proving operations suitable for graphical primitives.

Proving operations based on those primitives defined in below should allow a specific verification using primitive features. They should work on data not affected by preprocessing algorithms. Beside the result of the verification, the operation should provide the real parameters of the indicated primitive. The operation allow the verification of complex structures like those given by CAD-models and the approximation of a synthetical scene description to a given greyscale image.



Picture 1: Line, circle and arc primitives and their parameters used in AutoCADs DXF-format. LINE(x1, y1, z1, x2, y2, z2), start point P1 = (x1, y1, z1), end point P2 = (x2, y2, z2), CIRCLE(x, y, z, r), centre M = (x, y, z), radius r and ARC(x, y, z, r, a, b), centre M = (x, y, z), radius r, start angle a, end angle b.

3. PRIMITIVES AND SCENE ANALYSIS

The verification task requires an existing hypothesis giving us name, position and orientation of a model in the picture. Therefore the location of each hypothetical primitive in an image frame is well known and should be verified iconically within the image frame. Template matching as a well known technique to compare two-dimensional models and image structures on pixel level becomes only efficient if the number of possible templates is very small. But the primitives we are looking for appear in a wide variety of instances produced by translation, rotation and scaling.

Edge following techniques well known for binary and greyscale images produce any contour description based on a given image. In general, those algorithms do not use information about the contour shape. Starting at a special point the algorithm climbs along the contour based on the black to white change or the gradient direction. This is an uncontrolled approach because any shape may be followed in any orientation independent of a priori knowledge about contour details. Further more, the advantage of contour based approaches not to visit all existing image points may be damaged by an exhaustive search for appropriate start points. A model based start point generation may prevent this.

The use of other techniques allows a direct access to shape information. The Hough-Transformation has been developed to detect curves given analytically [Hough, 1962]. Its application to straight line and arc detection is described in [Ballard, 1982]. Those curves we are looking for determine the parameters defining the transformation space which is searched for a special curve instance based on type information. We are interested in algorithms using both type and location information.

To derive a complete segmentation of polygons Vieweg and Carlsohn [Vieweg, 1990] suggest a method for modelbased contour following. A model independent preprocessing reduces the intensity image to a binary one consisting only of straight line elements. Starting at a given point a hypothesis is generated to estimate the location of the next object corner. It must lay straight along the given line. At this hypothesized line consisting of n pixel n hypotheses based on two intersecting lines are extracted from the model. Hypotheses are verified by comparing the results of the convolution operation with the Sobel Operator and the data in the gradient image. The approach performs a controlled search, reducing the number of visited points. Unfortunately, the approach demands edge images with lines limited to one pixel width.

The projection between a symbolic object description and a raster oriented representation is also discussed in computer graphics. A widely used algorithm for straight line generation in a rasteroriented frame is given by Bresenham [Bresenham, 1965]. We consider straight line verification as a reverse Bresenham generation. Algorithm 1 shows the verification of contour points at iteratively calculated image locations. Starting at point P1 the line is verified if the end point P2 is reached. This simple approach can only prove artificial straight lines in binary images. Start and end points have to be defined exactly. The line has to be generated by the same algorithm or must exactly match its result. Both conditions can not be accepted for image analysis where a searched region defining a straight line may vary in position, location and width from those ideals. So similar lines have to be accepted by verification.

```
WHILE (value = FOREGROUND)
   BEGIN
      x := x + 1;
      y := a· x + b;
      y := round(y);
      value := getpixel(x, y);
   END
```

Algorithm 1: The schema of straight line verification follows the idea of Bresenham´s line drawing algorithm.

The slope a = dy/dx given by the model determines the next pixel examined by Algorithm 1. We can reduce this iterative calculation to a neighborhood search by transforming the intensity image so that the slope of each pixel is explicitly given. Since the slope of a straight line is orthogonal to the direction φ of the gradient we have to remember its features.

5. GRADIENT IMAGE

The proving operations in this paper are based on the gradient of the intensity image. This vector is given for all pixel locations (x, y) by its amount g and its direction φ:

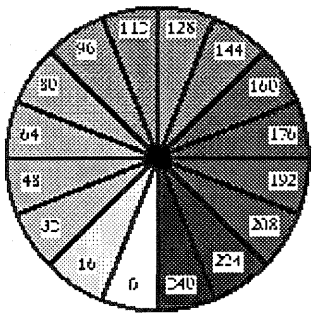
$$g=\sqrt{dx^2+dy^2} \qquad \text{(Eq. 1)}$$

$$\varphi=\arctan\frac{dy}{dx} \qquad \text{(Eq. 2)}$$

where

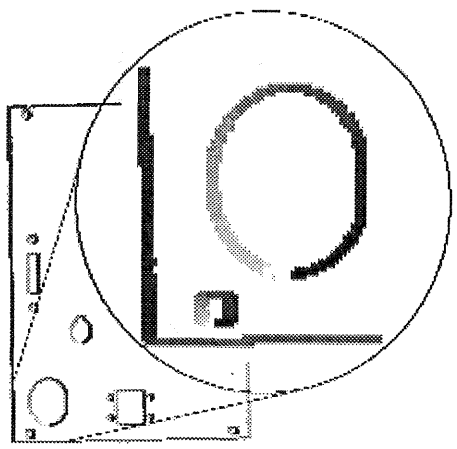
$$dx_{ij} = x_{i+1} - x_{ij} \qquad dy_{ij} = y_{ij+1} - y_{ij} \qquad \text{(Eq. 3)}$$

φ points to the direction with the strongest variation of the amount; g gives the quantity of this variation while stepping from (x_i, y_i) in this direction. Both values $g = f(x, y)$ and $\varphi = f(x,y)$ may be represented iconically in the gradient and the direction image which state local intensity variations. We consider only those points belonging to a contour which is indicated by a great amount of gradient .



Picture 2: Greyscale coded direction segment representing the tangents which approximate the circle.

For further discussion, we subdivide the set of all gradient directions in 16 segments, each covering 22.5 degrees. Each sector is depicted by a greyscale value from the set G = {0, 16, 32, ..., 240}. An image region with constant gradient direction looks homogeneous whereas those containing differences show significant variations. Therefore the front panel in picture 3 shows homogeneous straight lines and structured arcs. Lines of the same orientation may vary in the coded direction. This is caused by different material sides which turn the sign of the slope.Details shown in picture 3 state that a straight line is represented by a set of neighboring pixels covering only two neighboring directions. A circle consists of pixels representing all possible directions, neighboring pixels belong to neighboring direction segments.



Picture 3: A front panel contour, depicted by the greyscale coded direction.

The size of the gradient mask (1 x 2 and 2 x 1 pixel) guarantees a minimal contour width of two pixels within a binary image with a maximal intensity slope. Using intensity images the resulting width depends on the underlaying slope and on the threshold. Typically it is 2 to 6 pixel while using a threshold of 30 for the test images.

6. STRAIGHT LINE VERIFICATION IN DIRECTION IMAGE

Because the slope is explicit represented in a direction image it is possible to decide locally whether or not a special pixel belongs to a given line with a well known slope. The compatibility of the attachment with the lines extension is calculated by a recursive line point verification. A suggested line called ModelLine given by the start and end point is verified by the following algorithm:

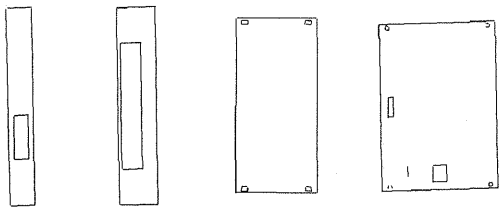
```
FUNCTION verify_line (ModelLine, PictureLine)
BEGIN
    search for start point
    calculate the orientation of ModelLine
    verify_linepoint (Point,Orientation)
    approximate PictureLine
    compare ModelLine with PictureLine
END

PROCEDURE verify_linepoint (Point, Orientation)
BEGIN
    IF pixel orientation = Orientation THEN
        BEGIN
            Point.x := Point.x + 1
            verify_linepoint (Point, Orientation)
            Point.x := Point x - 2
            verify_linepoint (Point, Orientation)
            Point.x := Point.x + 1
            Point.y := Point.y + 1
            verify_linepoint (Point, Orientation)
            Point.y := Point.y - 2
            verify_linepoint (Point, Orientation)
        END
END
```

Algorithm 2: Straight line verification and recursive verification of line points in the 4 connected neighborhood.

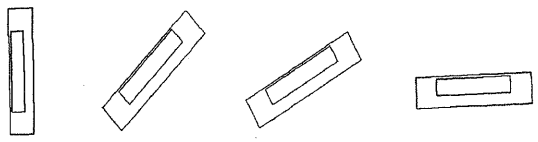
For low contrast images the four-connectedness is susceptible. We got better results using the eight-connectedness. Similar approaches are well known for segmentation purposes. Our routine is called from a high processing level. supplied with model details. Its recursive search goes on until this detail conditions are violated. No thresholds are used except for those defined in the verification routine. Here the necessary inexactness is given by the search area for a start point, the tolerance area for line end points and the allowed orientation angle.

The straight line verification described above has been tested on pictures of several aluminum plates. All tests are carried out with a search area of +/- 4 pixel and a tolerance value of 10 pixel. The transformation parameters may be seen as ideal conditions because they have been generated interactively.



Picture 4: Verified straight lines of some front panels.

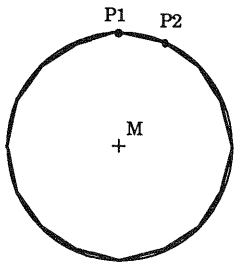
All straight lines above a minimal length of 10 pixels have been verified successfully. Actually, the algorithm does not distinguish between edges with the same direction and different orientation. So the opposite boundaries of a very small hole are indistinguishable without further model information. This happens at the little screw wholes with an extension up to 6 pixel. The verification of larger primitives is invariant under rotation and translation.



Picture 5: Verified straight lines of a rotated object.

7. VERIFICATION OF CIRCLES AND LINES

Straight line verification is organized as contour following of pixels with constant direction. Circles and arcs have no such simple direction features. Their generation algorithms may be classified as raster oriented approaches for creating the "true" line and those interpolating the shape by polygons, Bezier-curves, B-Splines etc. Disadvantages of both approaches are well known in computer graphics. The raster oriented approach needs new primitives which usually demand different data structures and algorithms e.g. extended clipping, transformation and projection algorithms. These operations may map a primitive type to a new one and the circle and arc generation algorithms can not prevent the necessity for higher ordered curve interpolation. The interpolation approach just looks for a curve approximation. Obviously, this approach is more efficient than the raster-oriented on. It is a useful alternative if its precision is acceptable and is applied here.



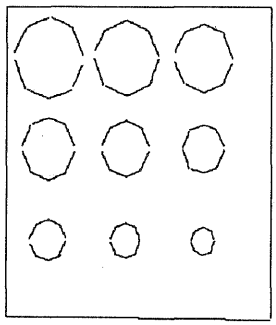
Picture 6: Circle approximation by direction segments.

The segments created in the direction image are already an approximation of the curve (picture 6). Although the curve is given by a set of discrete pixels located on the curve to be approximated the pixel values classify them as straight line points. In this way circle verification is reduced to the verification of a set containing straight lines. The verification is always done clockwise. When reaching a new line, the direction code is incremented.

```
FUNCTION verify_circle (ModelCircle, PictureCircle)
BEGIN
    initialize i
    calculate start point Pi
    increment i
    DO
        calculate the next point Pi
        verify_line (line(Pi-1, Pi),PictureLine)
        increment i
    WHILE(i < 360/MINANGLE)
    approximate PictureCircle
    compare ModelCircle with PictureCircle
END
```

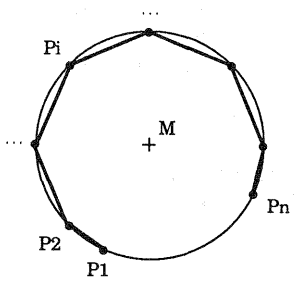
Algorithm 3: Circle verification using an interpolation approach.

Picture 7 states the results of processing circles of different size. All circles with a diameter larger than 20 pixels can be verified successfully. Smaller ones fall short of the minimal region size for straight line approximation.

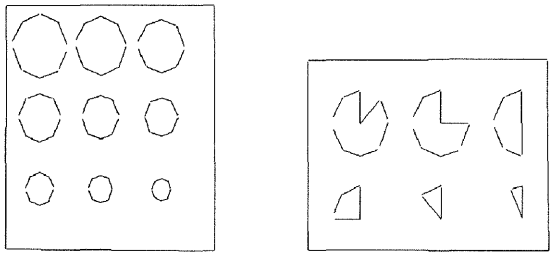


Picture 7: Results of circle verification using an interpolation approach based on 8 straight lines. The circle diameters lie within 30 to 85 pixels.

Arc verification follows a closely related way. Instead of a fixed start point the first and last point have to be calculated by the circle equation. Additionally all involved sectors are determined.

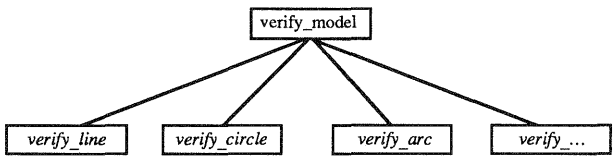


Picture 8: Arc verification where M = (8,8), r = 5, alpha = 202,5 and beta = 112,5 based on 8 interpolating straight lines.



Picture 9: Verification results of arcs varying in size (85 to 30 pixel with 360 degree)and angle (22,5, 45, ... 315 degree).
Further 2D primitives within the DXF-Format may be projected to one of the presented types. Composed structures like squares, rectangle, etc. may be defined for verification purposes. Complex structures are verified by a sequence of primitive verifications (picture 11).

The application of the described proving operations is not limited to two dimensional objects. Once the 3D model primitives have been projected to the image frame using a more powerful transformation they may be verified in the same manner. Some important CAD-models like Boundary Representations (BReps) are based on the primitives we are using here and we hope to verify these pretentious models after some further work.



Picture 10: Hierarchy of the discussed verification routines

8. THE ADVANTAGE OF PROVING OPERATIONS

The proving operations described above consist of a

- type and parameter specific pixel segmentation
- type specific approximation
- similarity function

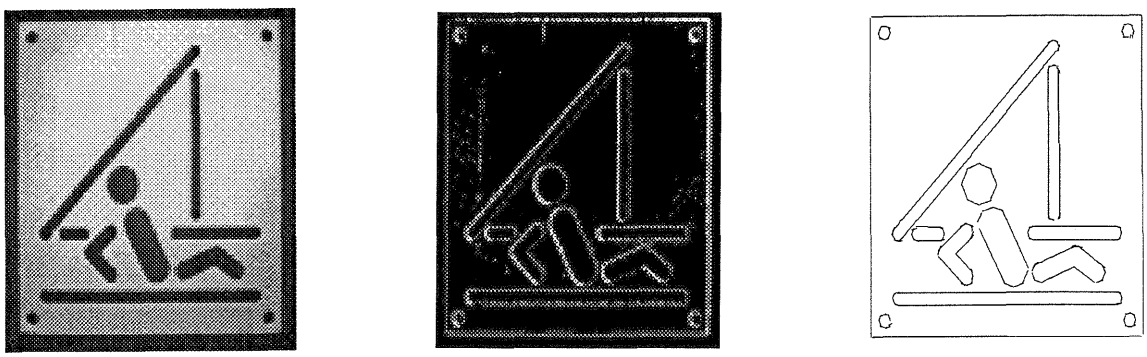
The combination of these tasks which are usually spread about different levels like segmentation, feature extraction and classification within a single operation is the foundation of the obtained modularity.

The discussed proving operations offer the capabilities to decide wether there is a special image portion. Because of the modular concept in using primitives, these special portions cover single primitives, substructures as well as complex structures, including such useful basic forms like squares, rectangles and others. If there is a manageable set of hypothesis this approach prevents a costly bottom up check with unknown borders, types, orientation and location. Within an interactive image analysis system such verification routines may become a powerful tool for recognition and conversion tasks. But even when there is no user to secure the number of hypothesis usually there is enough a priori knowledge useful for limitations. Therefore we are engaged in developing a control structure suitable in the depicted context.

The application of the described proving operations is not limited to two dimensional objects. Once the 3D-model primitives have been projected to the image frame using a more powerful transformation they may be verified in the same manner. Some important CAD-models like Boundary Representations (BReps) are based on the primitives we are using here and we hope to verify these pretentious models after some further work.

9. CONCLUSIONS

We have discussed algorithms for the verification of graphical primitives. The operations are organized to work in a strong model driven interpretation process. The gradient direction image we are working at is much closer to the geometric CAD-models than a captured greyscale image and is close enough to the captured data to prevent the influence of numerous thresholds. The described operations work on 2D primitives in 2D images, nevertheless they may be used in 3D context after a suitable projection.



Picture 11: Intensity and direction image of a tin product and those elements, verified by the described algorithms.

10. REFERENCES

Ballard, 1982. Ballard, D.H.; Brown, C.M., Computer Vision, Prentice Hall, Englewood Cliffs, pp. 123 ff.

Bhanu, 1987. Bhanu, B. Ho, C., CAD-based 3d object recognition for robot vision, IEEE Computer, pp. 19-36, August.

Bresenham, 1965. Bresenham, J.E., Algorithm for Computer Control of a Digital Plotter, IBM System Journal, Vol. 4, No. 1, pp 106 ff.

Coy, 1989. Coy, Wolfgang; Hönisch, Ulf, Cyclic Recognition of simple industrial scenes composed of CAD-objects, in: Proc. IIIrd International Computer Analysis of Images and Patterns Conf. CAIP ´89(Leipzig, Sept. 89).

Henderson, 1990. Henderson, Th. C. et. al.CBCV: A CAD-based vision system, Bild und Ton, Vol. 43, No.12, pp 364-368.

Hönisch, 1986. Hönisch, Ulf, Modellierung binokular erzeugter Bilder, Diplomarbeit, Universität Bremen.

Hough, 1962. Hough P.V.C., Method and means for recognizing complex patterns, U.S. Patent 3,069,654.

Fu, 1974. Fu, K.S., Syntactic Methods in Pattern Recognition, Academic Press, New York.

Gmür, 1988. Gmür, E.; Bunke, H. PHI-1: Ein CAD-basiertes Roboter-Sichtsystem Mustererkennung, Springer-Verlag, Berlin, Heidelberg 1988, pp. 240.

Huttenlocher, 1987. Huttenlocher, Daniel P.; Ullman, Shimon; Object Recognition using Alignment, Proc. of the 1´st Int. Conf. on Computer Vision, pp. 102-111, London.

Ikeuchi, 1987. Ikeuchi, K., Generating an interpretation tree from a cad model for 3d-object recognition in bin-picking tasks. International Journal of Computer Vision, Vol. 1, No. 2, pp.145-167.

Lamdan, 1988. Lamdan, Yehezkel; Schwartz, Jacob T.; Wolfson, Haim J.; Object Recognition by Affine Invariant Matching Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 335-344, Ann Arbor, Michigan.

Requicha, 1982. Requicha, A.A.G., Voelcker, H.B., Solid Modeling: A Historical Summary and Contemporary Assessment, IEEE Computer, Graphics & Applications, Vol 2:2, March 1982, pp. 9-24.

Shirai, 1983. Shirai, Y., Koshikawa, K.,Oshima, M., Application of 3-D Models to Computer Vision, Computer & Graphics, Vol. 7, No. 3-4, 1983, pp. 269-275.

Vieweg, 1990. Vieweg, Andreas; Carlsohn, Matthias, Modellgesteuerte Konturverfolung zur vollständigen Segmentierung von Bildern. In: Mustererkennung ´90, Springer Verlag, Berlin1990, S. 509-510.