IDENTIFING OBJECTS FOR FACTORY TRANSPORTATION ROBOT

Andras Barta*, István Vajk*

**Department of Automation and Applied Informatics, Budapest University of Technology and Economics, H-1521, Budapest, Hungary, e-mail: abarta8063@aol.com*

Abstract: One fast growing area of autonomous mobile robots is factory floor transportation robots. They perform simple object manipulation or transportation functions. The calculation of the movement of the robot requires an internal model of the surrounding objects. This paper investigates how a 2-D model can be created using a single gray-scale picture. The first part of the paper explores the possibilities of converting the picture to a form that can be used for model identification. The second part examines how this preprocessed data can be used to identify predefined objects. A simulated factory floor is used to evaluate the algorithm. *Copyright © 2000 IFAC*

Keywords: Optimization, Modeling, Mobile robots

1. INTRODUCTION

There are more and more real world applications for mobile robots. One fast growing area is a factory floor robot, where they perform simple object manipulation or transportation functions. In many cases a special sensory system is used to perform the navigation task. Putting wires or magnets in a floor is a common procedure, but more sophisticated methods, like differential GPS, can also be applied. The disadvantage of this approach is that the robot unable to manipulate objects. A better, more flexible solution is if the robot creates an internal model of the surrounding objects. To program these object manually is time consuming, requires continuous human intervention and is not applicable, where moving object are also present. A better approach is to use input devices to get information about the surroundings. Such devices are widely available (video camera, ultra-sound system, magnetic sensor) and it is not difficult applying them and getting the information to the robot's processing units. Every robotic system is designed for a different task. Their internal model should match the external requirements. It is enough to build a model with minimal complexity suitable for the application. For factory floor robots it is a reasonable requirement to represent every object directly. A transportation robot has to perform most of its task in two-dimension, therefor for many application it is sufficient to build a two dimensional model. The presented method is designed to build a two-dimensional model, but with a little modification it can be applied to three-dimensional problems too. The model creation has to go through the following steps:

1. Collecting data from the input devices

2. Transforming them to a form, more usable for model building. In this case 2D-pixel map.

3. Model creation and optimization

4. Path planning

This paper only deals with step 3 in detail. The algorithm to create the 2-D representation is not obvious at all. The critical issue in this step is collect data, which contains enough information to create an unambiguous model. This might require multiple source of information. It can be done for example, by using stereo camera or solar sensors. From the stereo image by vertical edge detection and depth calculation the 2-D projection can be determined. Ultrasound systems are inherently good in measuring distances, but using it for modeling is quite difficult. It requires frequency modulation, several ultrasound sensors and very complex data processing.

This paper investigates how a 2-D model can be created using a single gray-scale picture. The first part of the paper explores the possibilities of converting the picture to a form that can be used for model identification. The second part examines how this preprocessed data can be used to identify predefined objects.

2. IMAGE PREPROCESSING

A computer-generated pixelmap is used to simulate the factory floor. Objects from an object library (Fig. 2.) placed on a 256 x 256 pixelmap. The image consists of single and composite objects, but overlapping are not allowed. It is printed and photographed by a digital camera. Fig. 2. shows the grayscale intensity image.

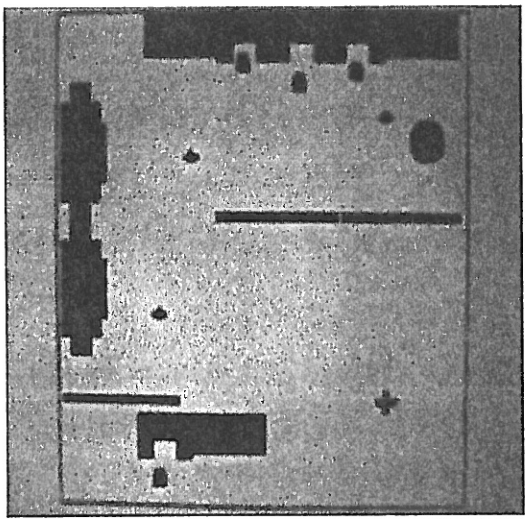

Fig. 1. Grayscale intensity image

The algorithm uses pixel matching to identify the objects, therefore the picture has to be scaled to real object size. The rectangular frame lines can be used for size identification. In a real applications any markings can be used for this purpose.

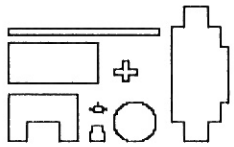

Fig. 2. Predefined Objects

The lines of the rectangle can be identified easily using radon transform. The Radon transform calculates the projection of an image $f(x,y)$ with variable projection angles.

$$R(x',\theta)=\int_{-\infty}^{\infty} f\left(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta\right)dy'$$

where

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \cos\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

The maximums of the radon transform is reached when the projection beam is parallel to a line. With the known frame lines the picture can be cut and rescaled to the actual size. The model identification and validation require the calculation of a cost function. Since this paper uses the pixel data a perimeter data for identification, the pixelmap and the perimeter data of the objects are needed. The objects are easily separated from the background by simply comparing the pixel values to a gray level. If the intensity difference between the objects and the background is sufficiently large than it is a simple and fast method separating the objects. Otherwise more complicated processes are needed.

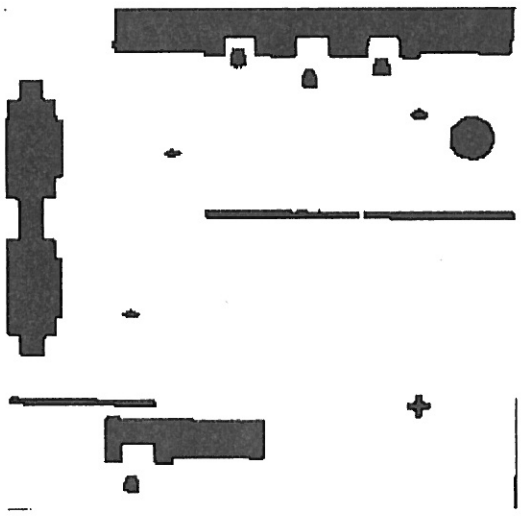

Fig. 3. Binary pixelmap

The perimeter pixels of the object can be determined from the binary objects or directly from the gray scale image. The advantage of working with the gradient is that it contains more information. It is especially useful, when the resolution of the image is low compared to the object size. The gradient of the image can be calculated with a convolution mask. Several convolution masks are used for gradient calculation: simple different operators, Roberts operator, Sobel operator, etc.

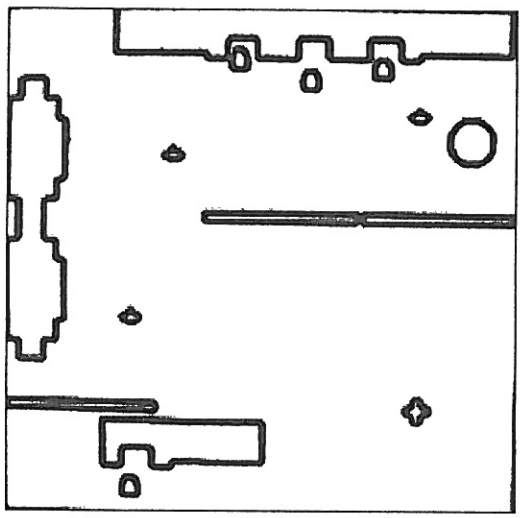

Fig. 4. Gradient image

A simple one-dimensional difference operators can be used:

$$G_x = \frac{1}{2}[1 \quad 0 \quad -1], G_y = \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}.$$

Since the gradient is a vector quantity it's magnitude can be calculated by any vector norm, for example the eucledesian norm:

$$|\mathbf{G}| = \sqrt{G_x^2 + G_y^2}.$$

3. OBJECT IDENTIFICATION

The image has to be further processed to gain more information that can be used for model identification. The section describes the most frequently used parameters and features and how they can be used for object identification. In the cases where the task is to identify only a single, separate object is quite simply. It is much more difficult to solve those problems, where multiple objects have to be identified simultaneously. The situation occurs where objects touch or cover each other. In this case the 2-D picture is represented by the combination of objects. Finding this combination requires a global optimization algorithm. In a typical application the incoming data are noisy and incomplete, thus it is not always possible to get an accurate model. It is a reasonable requirement for a mobile robot not to fail in any circumstances. A global optimization algorithm is good framework for object identification, since always provides a solution.

3.1 Shape parameters

From the two-dimensional pixelmap several parameters can be calculated to identify the shape of the object (Jähne, 1997). If enough shape parameters can be obtained from the image then the objects can be directly identified by classification. The area (A), perimeter (P) and the bounding box describes the object, but they depend on the size. Size independent parameters can be defined too.

Circularity: Circularity is a dimensionless number, reaching its minimum value for circles ($\approx 4\pi$).

$$c = \frac{P^2}{A}$$

Moments: Moments can be defined for both binary and gray value images. The central moments of an object are

$$m_{p,q} = \sum_A (x_1 - \langle x_1 \rangle)^p (x_2 - \langle x_2 \rangle)^p$$

The quantity $\langle \mathbf{x} \rangle = (\langle x_1 \rangle, \langle x_2 \rangle)$ is the center of mass. Size independent moments can be derived by normalizing with $m_{0,0}$,

$$\overline{m}_{p,q} = \frac{m_{p,q}}{m_{0,0}^{(p+q+2)/2}}$$

From the moment several other shape parameter calculated, for example *eccentricity*

$$\varepsilon = \frac{(m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2}{(m_{2,0} + m_{0,2})^2},$$

which is similar to circularity, except it is a little better defined, zero for circle and one for lines.

Fourier descriptors: Fourier descriptor uses the contour of the object to describe it. It can be defined in Cartesian a polar coordinate systems. The polar coordinates, since the $r(\Theta)$ radial boundary must be single valued, can not be used for every type of objects. In Cartesian coordinates the boundary of the

object is described by the path length parameter p from a starting point.

$$z(p) = \sum_{u=-\infty}^{\infty} \hat{z}_u \exp\left(\frac{2\pi i u p}{P}\right) \quad ; u \in \mathbb{Z}$$

$$\hat{z}_u = \frac{1}{P}\int_0^P z(p)\exp\left(\frac{-2\pi i u p}{P}\right)dp$$

To obtain scale invariant Fourier descriptors the coefficients can be normalized by $|\hat{z}_1|$. The Fourier descriptors can be used also to identify a local part of the boundary. In order to get a closed curve it has to be traced backward to the starting point.

Linear boundary: The above parameters are quite general, they can describe any type of objects. For special objects, however simpler descriptors can also be used. For example objects, with piecewise linear boundaries can simple be parameterized by line length and angles.

Splines: Splines can be used to model general shape objects. The boundary curve is represented by only a few control points, thus requiring little storage space. The curve does not go through the control points. A point on the curve can be calculated from a parameter and from the control points.

$$\mathbf{P}(\mu) = f(\mu, \mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N)$$

Usually the endpoint are represented by $\mu = 0$ and $\mu = 1$ parameter values. Several different types of splines can be defined. The most frequently used ones are the Bezier-curve, B-spline, β-splines (Burger, Gillies, 1989).

3.2. Low level features

The most critical part of object recognition is finding common parameterized features in different objects, which then can be used as bases to describe or identify the objects. The most frequently used low-level features are lines, curve segments, junctions, corners, regions. These parameterized models are fitted to the image. First the intensity discontinuities are identified with edge detection methods, then a line, or any parameterized curve hypothesis created to represent the edge. The hypotheses is then validated or rejected by comparing the feature hypotheses to the data. The junctions and endpoint of the curves are also used for object identification (Henricson, Heitger, 1994). There are several methods for feature extraction. The combination of several features represent the objects better, therefore features are grouped together. This higher-level feature groups are used to identify the objects.

Low level features are important, since they divide the image into meaningful components. Another way to segment the image into regions, by separating the areas which pixel intensity or texture is similar.

3.3 Optimization Algorithm

The identification of composite objects is a global optimization problem. There are several optimizers that can be used for this problem: evolutionary algorithms, genetic algorithms, simulated annealing, Monte Carlo methods. Most of them fall into the stochastic algorithm category, since they manipulate the solutions randomly. Table 1. describes a general stochastic optimization algorithm for this model identification. By modifying the conditions and operators many stochastic optimization algorithm can be recreated.

Table 1 Optimization algorithm

0: Initialization

1: ***If*** *(Create-Condition)* ***CreateObject***

Perturb the solution

if *(Acceptance-Condition)* ***Accept*** *the new solution*

else *Restore the original solution*

if *(Delete-Condition)* ***DeleteObject***

if *(Stopping-Criteria)* ***End***

else goto step 1

To evaluate a solution a cost function has to be calculated. In order to get a reasonable solution at least the perimeter and pixel data is needed. The cost of every object model is calculated separately.

$$C(x, y, \alpha) = \frac{w_1 C_{area} + w_2 C_{ovrl} + w_3 C_{perim}}{w_1 + w_2 + w_3} \qquad (1)$$

where C_{area} is the object's area not covered the binary image, C_{ovrl} is the object's area overlapped by other objects and C_{perim} is the object's perimeter not matched by other object's or the pixelmap's perimeter. C_{perim} can be calculated using the binary or the gradient image directly. w_1, w_2, w_3 are weighting constants. Each component and the cost function are scaled to [0,1] interval. 1 means that the object is in the worst possible position and 0 means that the object is identified. Shape parameters and features can be used in the algorithm in the conditions and operators.

The *Stopping-Criteria* of an algorithm is very application dependent. Due to noise and inaccurate

models some objects can never be identified perfectly. In case of mobile robots that means that in the path and task planning there are always some uncertainty.

4. SIMULATION

The purpose of the simulation is to see how the above image processing and optimization tools can be combined to get a method for object identification. Both single and composite objects are identified. The identification of single objects is simpler, since shape parameters can also be used. It is really a classification problem. Since the used object library is quite small, only two shape parameters are enough to identify them: object area and circularity.

3.1 Optimization

Composite objects require a global optimization method. First a simple optimization algorithm is used with the perimeter and pixel data. Objects are created randomly on the pixelmap until the sum of their area is smaller than the pixelmap's area. A new solution is created by moving an object to a certain direction. In each step the cost function (1) is evaluated. If the new position satisfies the *Accept-condition* it is accepted, otherwise its direction is changed. Several *Accept-condition* is used. A hill climbing (accept the new solution if it's cost function is lower) and a simulated annealing (Metropolis criteria, Aarts and Korst, 1989) was tested in detail. Removing the not improving objects (stacked in the local minimas) is a critical part of the algorithm, since they block the movement of other objects.

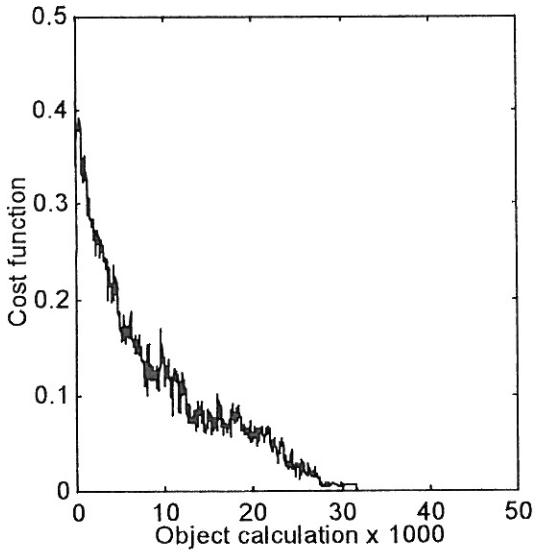

Fig. 3. Cost function of the optimization

Several methods are examined for deleting objects. Objects can be removed with a probability that is calculated from their cost. A roulette-wheel selection method and highest cost removal are also tested. The simulation is finished when the whole image is covered and the average cost of the object is lower than an acceptance limit. The best performing combination was a simulated annealing algorithm with the highest cost object removing. The temperature of every object was controlled separately and the frozen high cost objects were removed. The other algorithms are performed reasonable well too. The simulation was done in C++ on a PC (Pentium-160). To find the optimum solution it took about 20-40000 object calculation and 30-60 seconds. Fig. 3. shows a typical cost function variation during the optimization.

3.2 Features

Only lines and their junctions (corners) are used, since they are easy extract and carry a lot of information about the objects, especially when many rectangular shapes are present. There are several algorithms to extract line features. In this paper a simple local algorithm is chosen. Local in the sense, that only a localized part of the image is processed to extract a line. First a few perimeter pixel are used to generate a line hypotheses. Then the relative distance and angle from this line calculated in both directions When the points fall outside of a distance limit, the line ends. Corner features are extracted by calculating the crossing of neighboring lines. Since the contour of the object is a closed curve this method always generate a junction (corner). Then the angle and orientation of corner was calculated. For object identification a group of these basic features are used: a corner, its orientation and length of the adjoining lines. Fig.4. shows the feature group that is used in the algorithm.

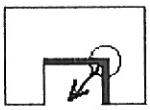

Fig. 4. Feature group

In the algorithm the features are built into the *CreateObject* operator. From the not covered part of the image a feature group is selected randomly. Then this is searched in the object library. If it is found it is placed on the image in a position to match the features. Since the object library is quite small a linear search is sufficient for this simulation. The rest of the algorithm is the same as in the previous simulations. The improvement is significant. The

algorithm with the feature group finds the optimum solution approximately hundred times faster.

5. DISCUSSION AND FUTURE RESEARCH

The purpose of this research was to find out that this global optimization is a feasible tool for modeling. The simulation shows, that a global optimization algorithm provide a good method for object identification, since it is quite easy to include different type of data. The fast convergence shows that the method can be scaled to the three-dimensional problems also. In thee-dimension object identification is very similar.

ACKNOWLEDGEMENT

The research work was supported by the fund of the Hungarian Academy of Sciences for Control Research and by the OTKA T029815 fund. The supports are kindly acknowledged.

REFERENCES

Aarts E. and J. Korst (1989). *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons

Birk A. (1996). Learning Geometric Concepts with an Evolutionary Algorithm, *Proc. of the Fifth Annual Conference on Evolutionary Programming*, pp. 83-91

Burger P.and D. Gillies (1989). *Interactive Computer Graphics*, Addison-Wesley Publishing Company

Cherkassky V. and F. Mulier (1998). Learning from data, John Wiley & Sons

English T.M. (1998). Evaluation of Evolutionary and Genetic Optimizers: No Free Lunch, *Proc. of the Fifth Annual Conference on Evolutionary Programming*, pp. 163-169

Henricson O. and F. Heitger (1994). The Role of Key-Points in Finding Countours, *Third European Conference on Computer Vision, Volume II*, pp. 371-382

Jähne B. (1997). *Digital Image Processing*, Springer Verlag, Berlin, pp. 507-519

Kortenkamp D. and R. P. Bonasso, R. Murphyl (1998). *Artificial Intelligence and Mobile Robot, Case Studies of Successful Robot Systems*, AAAI Press/MIT

Raidl G.R. (1996). Skilful Genotype Decoding in EAs for Solving the Cutting Problem, *Proc. of the Fifth Annual Conference on Evolutionary Programming*

Rissanen J. (1989). *Stochastic Complexity in Statistical Inquiry*, World Scientific Publishing Co. Pte. Ltd.

Wolpert D.H. and W.G. Macready (1995). No Free Lunch Theorems for Search, The Santa Fe Institute