PERIODICA POLYTECHNICA SER. TRANSP. ENG. VOL. 32, NO. 1–2, PP. 65–72 (2004)

SOFTWARE STRUCTURE FOR A CAMERA BASED MOBILE ROBOT

Béla KULCSÁR, Balázs GÓDOR and Gábor BOHÁCS

Department of Building Machines, Material Handling Machines and Manufacturing Logistics
Budapest University of Technology an Economics
H–1111 Budapest, Bertalan L.u. 7-9, Hungary
e-mail: kulcsar-bela@eagt.bme.hu, godor-balazs@eagt.bme.hu, bohacs-gabor@eagt.bme.hu

Received: Oct. 5, 2003

Abstract

Mobil robots are commonly used in various application fields. Military projects, space research, deep-sea research, and various robots at work and at home belong to the most significant applications. Material handling and warehousing are also possible environment. In these applications mobile robots are used as autonomous transporting vehicles. However, cooperative systems are rare. This paper presents a proposed software structure for a cooperatively working, camera based mobile robot for order-picking activities in warehousing. Mobile robot applications require complex software background. The implementation of the free navigation, obstacle avoidance and motion control modules is also presented.

Keywords: mobile robot navigation, image processing.

1. Introduction

The main goal of this current research is to develop a mobile robot platform for order picking which operates cooperatively together with its operator. The robot performs two main tasks: first a human following behaviour takes place, which saves the operator from driving the vehicle. The research of this task will be carried out later, therefore this paper does not deal with this problem.

After finishing the order-picking, the robot has to return to the unload position autonomously (*Fig. 1*). As a result of the previous research a mobile robot has been created which makes possible to carry out real experiments in the laboratory of the Department of Building Machines, Material Handling Machines and Manufacturing Logistics, BUTE. Currently only one sensor is mounted – a CCD camera, directed forward and down at an angle of $-45°$. A stationary PC via radio link implements image processing and generation of the motion command in order to keep the weight of the robot at the minimum.

This paper is dedicated to present the software structure needed for the operation of the robot, and the implementation of the obstacle avoidance and free navigation modules.

Obstacle avoidance deals with the recognition of obstacles in the camera view. There are two groups of obstacles: first there are obstacles with known

positions (obstacles displayed on the map), and there are unexpected obstacles as well. Recognising objects from the first group may result positive effect, because it can be an important reinforcement signal on the actual position. On the other hand known obstacles have also a negative effect, because depending on the robot orientation and position they may hide important navigation points. Unexpected obstacles not only hide navigation points, but also change the path of the robot.

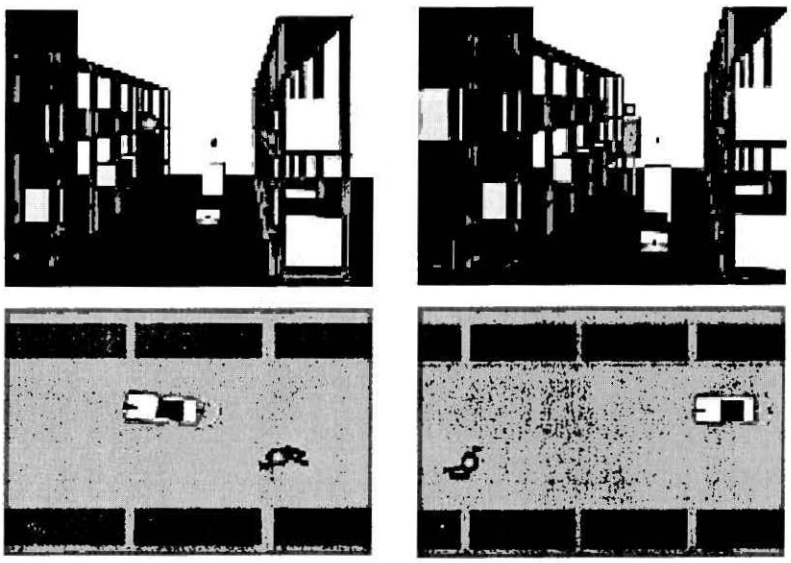

Fig. 1. Robot concept

The first question of the free navigation problem is to find appropriate navigation objects. These objects must be unambiguously identifiable, and must be sufficient in number as well. There are two fundamental types of the landmark based navigation [1]: in some cases navigation points are continuously sensed, however, in other cases the next navigation point is out of the sensing area. In most cases the movement to the next navigation point is not 'blind', because there are other supplementary navigation objects, such as lines, walls, corridors which help the robot to reach the next point. A corridor following example is presented in [2]. So we can conclude from the above that continuously identifiable navigation objects are necessary. These objects can be unique (OCR codes, barcodes, numbers) or of standard types such as doors or crossings. These objects can be natural or artificial. Artificial landmarks are normally easier to identify, but these must be created which is an extra task.

2. Software Structure

For fulfilling it's task, the robot needs to dispose of appropriate software background. A modular software structure has been chosen, as the developed and the future modules have to connect smoothly. Parts of the robot software are as follows:

- human following
- obstacle avoidance
- path planning
- free navigation
- precise positioning
- motion control

Human following is applied during the cooperation with the human worker in the order-picking phase, however, after finishing this, the robot needs to be changed to the free navigation, in order to reach the unload station autonomously. Free navigation requires also path planning, which is executed only once at the beginning of an autonomous movement phase. In the final phase the robot uses the precise positioning movement. Obstacle avoidance works continuously in all of the three phases as a safety module. Motion control module serves as master of the structure. It not only controls the robot's motion, but also sums the outputs of the other modules. Methods for the above modules, except for the human following and precise positioning ones are presented in this paper. The structure of the software is presented in *Fig. 2*.

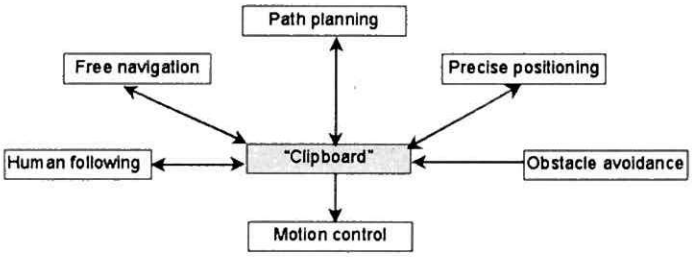

Fig. 2. Software structure

The central element of the structure is the 'Clipboard' which is an accessible file for all of the modules. All other parts use this file for read and write on the following information.

Path planning module inputs end the human following and precise positioning phases, as in these cases a new route has to be planned for free navigation. Path planning outputs series of navigation points that have to be found after each other. Finding new series of barcodes starts the free navigation, which signals back on reaching the end of the route. This module also gives continuously motion commands in order to move forward along the path. During human following, the

module carries out only position identification. The end of the route signal starts the precise positioning module, which also generates motion commands and signals the end position. Human following starts and ends by a command from the operator. It also generates motion commands. Obstacle avoidance module is continuously active, and generates motion commands. As this is the safety module, its output has higher priority then the others. Motion control module inputs and sums motion commands. This software performs the effective control of the robot hardware.

3. Free Navigation Module

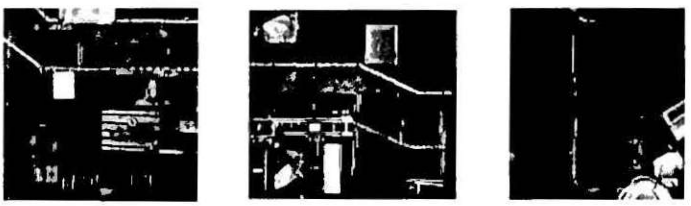

Fig. 3. Details of the testing environment

Free navigation is activated after the human following of the order-picking phase. So it is a movement between two known points. First the navigation objects have to be determined. The experimental laboratory of the department (*Fig. 3*) consists of a variety of very different objects. That makes impossible to find any natural landmarks. It should also be decided if unique landmarks alone are sufficient or not. As the environment is very complex many navigation points should be applied, in order to designate all of the path connections. Because of the environment these points should be placed relatively close to each other. Therefore no other navigation objects are needed if it is ensured, that from any map point at least one navigation object can be seen. For all that a barcode system has been developed, for barcodes are easy to be identified, and many unambiguous landmarks can easily be generated. By developing an appropriate barcode identification system the three most important questions are the size, the orientation and the information content of the label. The barcodes must be at least so big that the identification at the most distant position should also be possible (Shannon's sampling theory says, condition of an unambiguous identification is: size of the thin module should be at least 2 pixels wide on the image [4]).

There are three possibilities for barcode orientation. The possibility of laying barcodes on the floor was examined by [3] (*Fig. 4*). It required a complex and long searching process, because relative orientation of the barcode and the robot can be discretional. In case b) barcode search became simple (one directional) and fast, however barcode distortion at the edges of the camera view is significant. This deformation is almost corrected in case c) which was chosen as the best orientation. We should remark that in this case for a single navigation point multiple barcodes

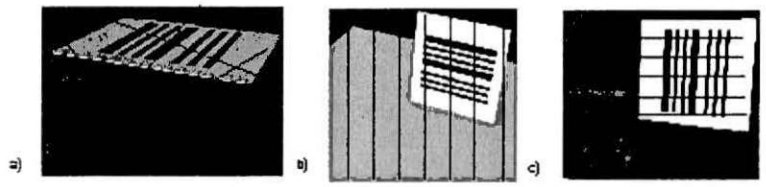

Fig. 4. Possible barcode orientations

should be fixed depending on the possible detecting directions. Each label contains a single modified Code 39 barcode character (a wide black line added at the front of the Code 39 character and a narrow one at the end of it, which enables an unambiguous identification). It gives 42 different navigation points, which are enough for this environment. The size of the barcode labels is A4.

The map of the laboratory is known by designating free spaces and obstacles on the photo of laboratory [5]. Using global navigation software, developed by [5], the determination of the optimal path between two arbitrary points of the environment is given (*Fig. 5*). For the successful navigation order of barcodes corresponding to the optimal path is to be determined.

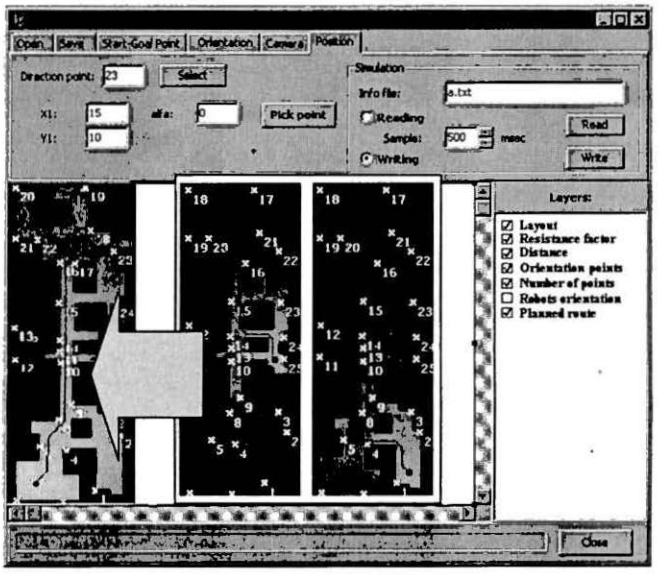

Fig. 5. Optimal path planning software

The queue of the required barcodes is given by determining the nearest barcodes to the path. The barcodes by which the robot should pass are collected into a vector. The robot should close the barcodes from left or from right or from the opposite, depending on the position of the label and the optimal path (*Fig. 6a*).

Closing a barcode is carried out using a simple visual servo algorithm (*Fig. 6b*).

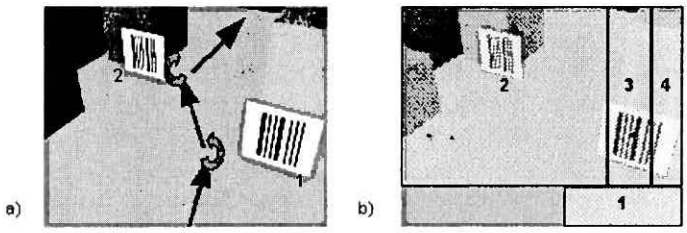

Fig. 6. Visual servo algorithm

For example closing barcode 1 depends on the position of the label. If the label is in domain 2 the robot turns left, if in 3 the robot moves forward, and if it is in domain 4 it turns right. This step is repeated until the barcode reaches domain 1. Similar methods can be applied for barcode closing from left and from the opposite. After reaching the (i)th barcode, robot searches for the $(i + 1)$th barcode of the collected vector. Finding the next label may require rotation until the next label is found. The direction of the rotation is determined by the angle of the $P_2(x_{i+1}, y_{i+1}, x_i, y_i)$ and $P_1(x_i, y_i, x_{i-1}, y_{i-1})$. After finding the next code a new closing process takes place until the last navigation point is reached. Test results of the described method are contained in chapter 6.

4. Obstacle Avoidance Module

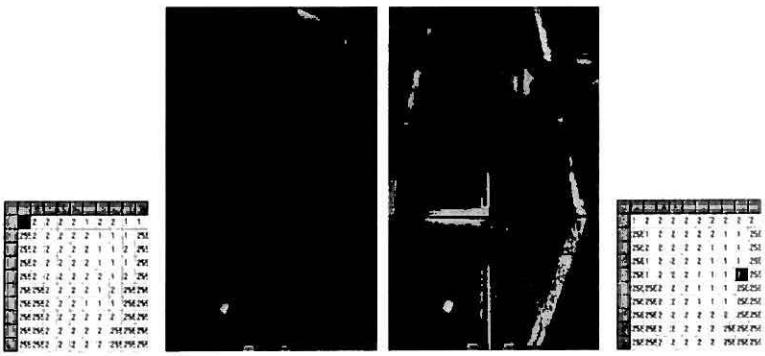

Fig. 7. Obstacle recognition software

The function of this module is the detection of obstacles and the emergency stop of the robot as well. The developed method divides the camera view into 240 × 340 mm, rectangular cells. From each cell 15 marked pixels are chosen. It means for the whole area (70 visible cells) $70 \times 15 = 1050$ pixels. The number

of pixels can't be increased further without slowing down the whole software. The occupancy of a cell is decided using a self organising clustering method. The teaching of the cells is simple and fast, however it requires a human operator. The method works quite reliably (*Fig. 7*), however, misdetection in single cells sometimes occurs. Therefore, the method was reconstructed so that occupancies of neighbouring cells influence occupancy of the actual cell. Teaching must take place under various light conditions in order to completely avoid misdetections. In the current software version, on finding an obstacle the robot immediately stops. The robot starts again, if the obstacle has been removed from the safety zone.

5. Motion Control Module

As the mobile robot currently does not have any additional sensors, such as shaft encoders, we implemented a teach-in method for the motion control as well. In this module so called elementary movements (e.g: turn slightly left, move forward...) can be taught by driving the robot in manual mode (*Fig. 8*). Elementary movements are saved on the hard disk, and called during the software run. Separate files contain each elementary movement, which enables a perspicuous programming.

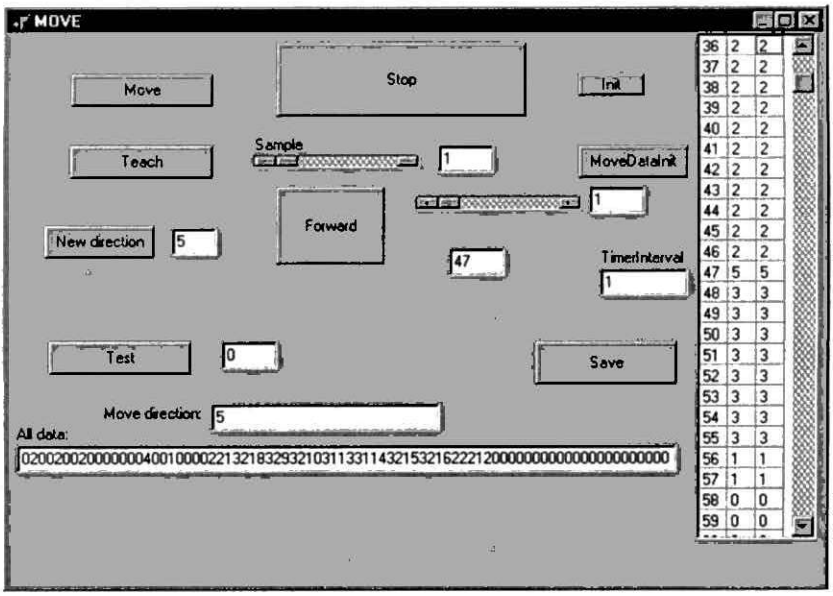

Fig. 8. Motion control software

6. Conclusion

The development of the navigational method brought important results for the research. Main advantage is that the tolerance of the label positions does not have to be smaller than some centimetres. Exactness of the robot path is about 10 cm, which fulfils our expectation for the free navigation module. The method is expected to be usable also in real warehouse environment, as barcodes or racks can be optimal navigation points. The next step of the research is the connection of single modules, and upgrade of the obstacle detection function to an obstacle avoidance module.

References

[1] BORENSTEIN, J., *Navigating Mobile Robots: Systems and Techniques*, A K Peters Ltd. Wellesley, 1996, ISBN: 156881058X, 209 pages.

[2] TOMONO, M. – YUTA, S., Mobile Robot Navigation in Indoor Environments using Object and Character Recognition, *Proceedings of the 2000 IEEE Intenational Conference on Robotics & Automation*, San Francisco, 2000 April, pp. 313–320.

[3] SCHMIDT, G., Visual Information Based Local Map Building, for a Mobile Robot Equipped with a Camera (Lokális térkép készítése vizuális információ alapján kamerával felszerelt mobil robotra), Diplomaterv, BME 2003, 76 pages. (in Hungarian).

[4] ÁLLÓ, G. et al., *Bevezetés a számítógépes képfeldolgozásba*, BME MTI Budapest, 1993, ISBN: 9634317774, 146 pages, (in Hungarian).

[5] VÁRPALOTAI, T., Position Estimation and Path Planning for a Mobile Robot, using Global Map, (Mobil robot helymeghatározása és útvonal optimalizálása globális térkép segítségével), Diplomaterv, BME 2003, 72 pages, (in Hungarian).

[6] KULCSÁR, B., *Robottechnika*, LSI Oktatóközpont, Budapest, 2000, ISBN: 9635772432, 394 pages, (in Hungarian).