Navigation of Transport Mobile Robot in Bionic Assembly System

Aleksandar Lazinica

Intelligent Manufacturing Systems
IFT
Karlsplatz 13/311, A-1040 Vienna
Tel : +43-1-58801-311141
Fax :+43-1-58801-31199
e-mail : lazinica@mail.ift.tuwien.ac.at

Branko Katalinic

Intelligent Manufacturing Systems
IFT
Karlsplatz 13/311, A-1040 Vienna
Tel : +43-1-58801-311141
Fax :+43-1-58801-31199
e-mail : katalinic@mail.ift.tuwien.ac.at

Abstract

In this paper we present the navigation of transport mobile robot in Bionic Assembly System. Bionic Assembly System is a concept of assembling system of tomorrow which basic elements are autonomous mobile robots. The basic characteristic of the system is capability of quick adaptation for different kind of products. This paper is focused on simulation of transport mobile robot's navigation in Webots software. Transport mobile robot should be designed to navigate with collision avoidance capability in the shop floor environment, flexibly coping with the changing environment.

I. Introduction

A concept of Bionic Assembly System (BAS), as an answer for novel requirements of manufacturing industry, was proposed by Katalinic [1]. The concept of the system was developed on a real industrial demand to significantly reduce the production costs of electrical motors in mass production. The main characteristic of such a system is its capability of quick adaptation for assembling different kind of products. Main elements of a system are autonomous mobile robots. They have to function autonomously, have to adapt themselves and act in strong co-relation between each other and their environment (shop-floor). Design of behaviour of these robots is a main task to be solved for making a system functioning. The ground problem of their behaviour is navigation through the shop floor. Their environment is complex and dynamically changes.

II. Autonomous mobile robots in bionic assembly system

As it is mentioned before, autonomous mobile robots are most important elements of Bionic Assembly System. There are two basic classes of them:

transport mobile robot – carries a palette on which parts are assembling together in a finished product, it begins with an empty palette and finishes with the complete assembled product,

assembly station – this is a mobile robot equipped with a robot arm; mobile robot serve as a carrier of a palette with parts and robot arm assembles them on a transport mobile robot.

Design of these two classes of mobile robots is a key problem of developing Bionic Assembly System. Such robots should be able to function autonomously and smoothly in order to cope up with unstructured and highly complex working environment of BAS.

Complete design of autonomous mobile robots involves four phases [2]:

definition of operating environment,
definition of robot tasks,
hardware design of robot and
design of robot's behaviour.

These phases have to be accomplished one following another and in close dependence between each other.

III. Navigation of mobile robots

First and fundamental problem of robots behaviour is design of mobile robot's navigation. Robot navigation is concerned with moving a robot to a specific posture to accomplish a given job, subject to internal and external constraints, in a dynamically changing environment. Using a set of sensors, the robot should recognize the environment in its neighbourhood and generate a suitable navigation plan to accomplish the job. The job of transport mobile robots in Bionic Assembly System is going from one assembly station to another with a palette caring on. In the simplified version of Bionic Assembly System assembly robots are static, so transport mobile robots should be able to avoid static and moving obstacles in their way.

IV. Simulation of robot's navigation in Webots

A. Webots Software

For simulation of Bionic Assembly System we have decided to use Webots professional software for simulation of mobile robots behaviour. This is the most realistic and reliable software of this kind at a moment. The simulation

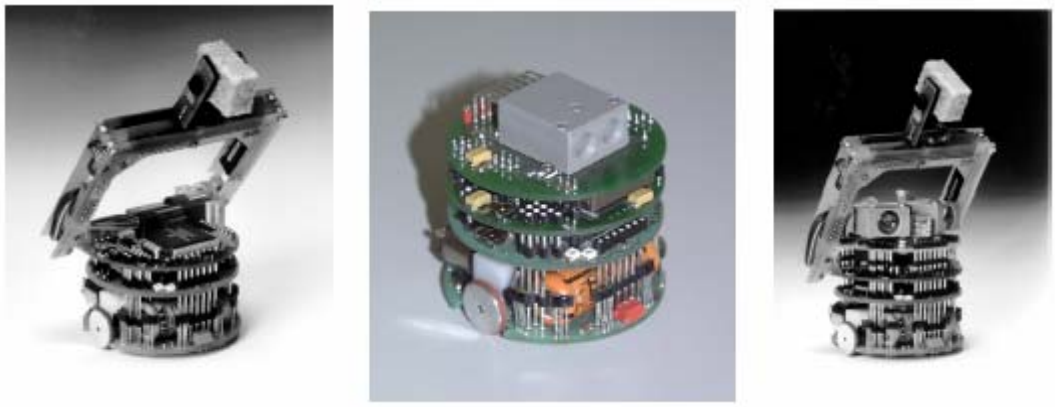


Fig. 1 From left to right: Khepera with a gripper extension, with a "linear vision" extension, and with both modules simultaneously

system used in Webots uses virtual time, making it possible to run simulations much faster than it would take on a real robot. Depending on the complexity of the setup and the power of your computer, simulations can run up to 300 times faster than the real robot when using the fast simulation mode. The graphical user interface of Webots allows you to easily interact with the simulation while it is running.

The robot´s behaviour is written using the C++ or Java programming language. Moreover, any Webots controller can be connected to a third party software program, such as MatLab, LabView, etc. through a TCP/IP interface.

Once tested in simulation your robot controllers can be transferred to real robots. This is the real power of Webots, because there is no other software that has capability of transferring the controller programs on real robots. More information about the software could be found in [3].

B. Khepera robot

We have decided to use the Khepera robots (Fig.1) in the simulation. The biggest deciding factor was that the Khepera robot has a gripper module as additional add-on. And the gripper is essential for manipulation of assembly parts. Khepera is a miniature mobile robot with functionality similar to that of larger robots used in research and education. Khepera was originally designed as a research and teaching tool for a Swiss Research Priority Program at EPFL in Lausanne. It allows real world testing of algorithms developed in simulation for trajectory planning, obstacle avoidance, pre-processing of sensory information, and hypotheses on behaviour processing, among others. Very modular at both the software and hardware level, Khepera has a very efficient library of on-board applications for controlling the robot, monitoring experiments, and downloading new software. A large number of extension modules make it adaptable to a wide range of experimentation.

C. Potential field method

There are numerous methods which are used for navigation of mobile robots [4]. The simplest and most effective one is potential field method. When you think of potential fields, picture in your mind either a charged particle navigating through a magnetic field or a marble rolling down a hill. The basic idea is that behaviour exhibited by the particle/marble will depend on the combination of the shape of the field/hill. Unlike fields/hills where the topology is externally specified by environmental conditions, the topology of the potential fields that a robot experiences are determined by the designer. More specifically, the designer (a) creates multiple behaviours, each assigned a particular task or function, (b) represents each of these behaviours as a potential field, and (c) combines all of the behaviours to produce the robot's motion by combining the potential fields [5]. The fundamental block of potential fields is the action vector, which corresponds to the speed and orientation of the robot. Each behaviour outputs a desired output vector. We have two forms of behaviour (two potential fields) – the attractive and repulsive field. The attractive potential causes the robot to be attracted to the goal, and repulsive potential causes the robot to be repulsed from the obstacle.

Attractive potential field (Fig. 2(a)) is described by following equations:

$$\Delta x_A = \alpha * (d - r) * \cos(\theta), \quad (1)$$

$$\Delta y_A = \alpha * (d - r) * \sin(\theta), \quad (2)$$

where:

θ – angle between the robot and the goal,
d – distance between the robot and the goal,
r – raidus of a goal,
α – constant with which the field could be scaled.

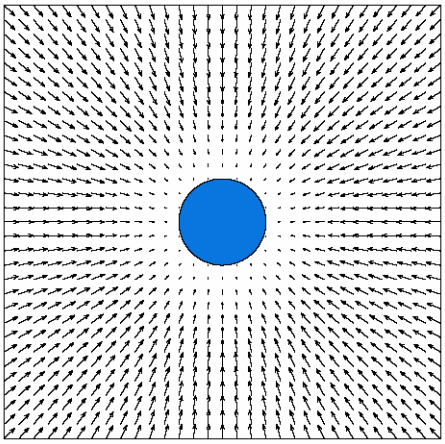
Repulsive potential field (Fig. 2(b)) is described by following equations:

$$\Delta x_R = -\beta * (s + r - d) * \cos(\theta), \quad (3)$$

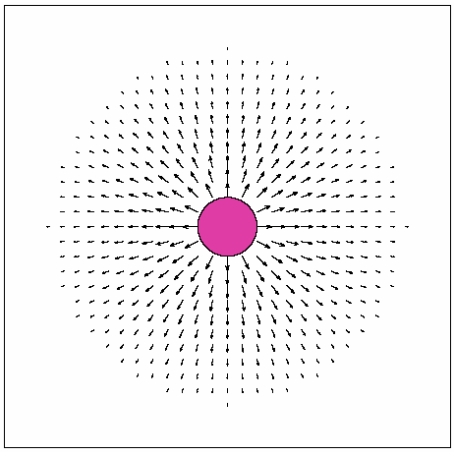
$$\Delta y_R = -\beta * (s + r - d) * \sin(\theta), \quad (4)$$

where:

θ – angle between the robot and the obstacle,
d – distance between the robot and the obstacle,
r – radius of a obstacle,
β – constant with which the field could be scaled.



(a)



(b)

Fig. 2 Attractive (a) and repulsive (b) potential field

The resulting field (Fig. 3) is the sum of attractive and repulsive field:

$$\Delta x = \Delta x_A + \Delta x_R , \quad (5)$$

$$\Delta y = \Delta y_A + \Delta y_R , \quad (6)$$

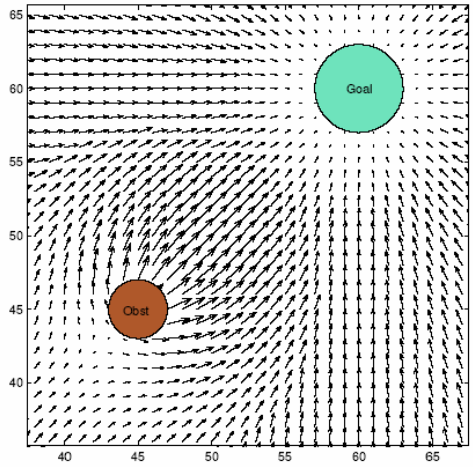


Fig. 3. The resulting potential field generated by attractive and repulsive field

This collection of vectors is called a potential field because it represents synthetic energy potentials that the robot will follow.

How does the robot choose its behaviour now? It determines the Δx using equation (5) of the potential field generated by its two behaviours, determines Δy using equation (6), and computes:

the velocity $v = \sqrt{\Delta x^2 + \Delta y^2}$ (7)

and the angle $\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$, (8)

and then sets the speed to v and the direction to θ. As it moves through the world, it makes observations of the environment, identifies new action vectors, and chooses new directions/speeds. Since the speed is function of potential fields and they are functions of distance, as the robot is closer to the goal and more far from the obstacle its speed is higher and higher. When the robot is near to the obstacle, its speed is lower.

The biggest problem in this method is computing of robot's actual position. Since the Khepera robot has the GPS (Global Positioning System) extension, this problem was easily solved in Webots. If we are working with real robots, we should use encoders since GPS is not yet so precise particularly in close spaces. The gps function used in Webots is:

gps = robot_get_device("gps"); //it gives the robot possibility to use the gps device//
gps_enable(gps,10); //it enables gps readings, every 10 ms//
matrix=gps_get_matrix(gps); //the matrix in which are x,y, z coordinates and robot angles are stored//
x_now=gps_position_x(matrix); //put the x coordinate read from the gps device in x_now//
y_now=gps_position_y(matrix); ////put the y coordinate read from the gps device in y_now//
z_now=gps_position_z(matrix); ////put the z coordinate read from the gps device in z_now//

D. Simulation

The simplified version of Bionic Assembly System that we have developed in Webots is shown in Fig. 4. It is consisted of following parts:

3 Transport mobile robots
3 Assembly stations
robot pool
recharging station
storage of final product
storage of empty palettes

We are using only three transport and three assembly

robots because of the system simplification. The complete simulation should function as follows[6]:

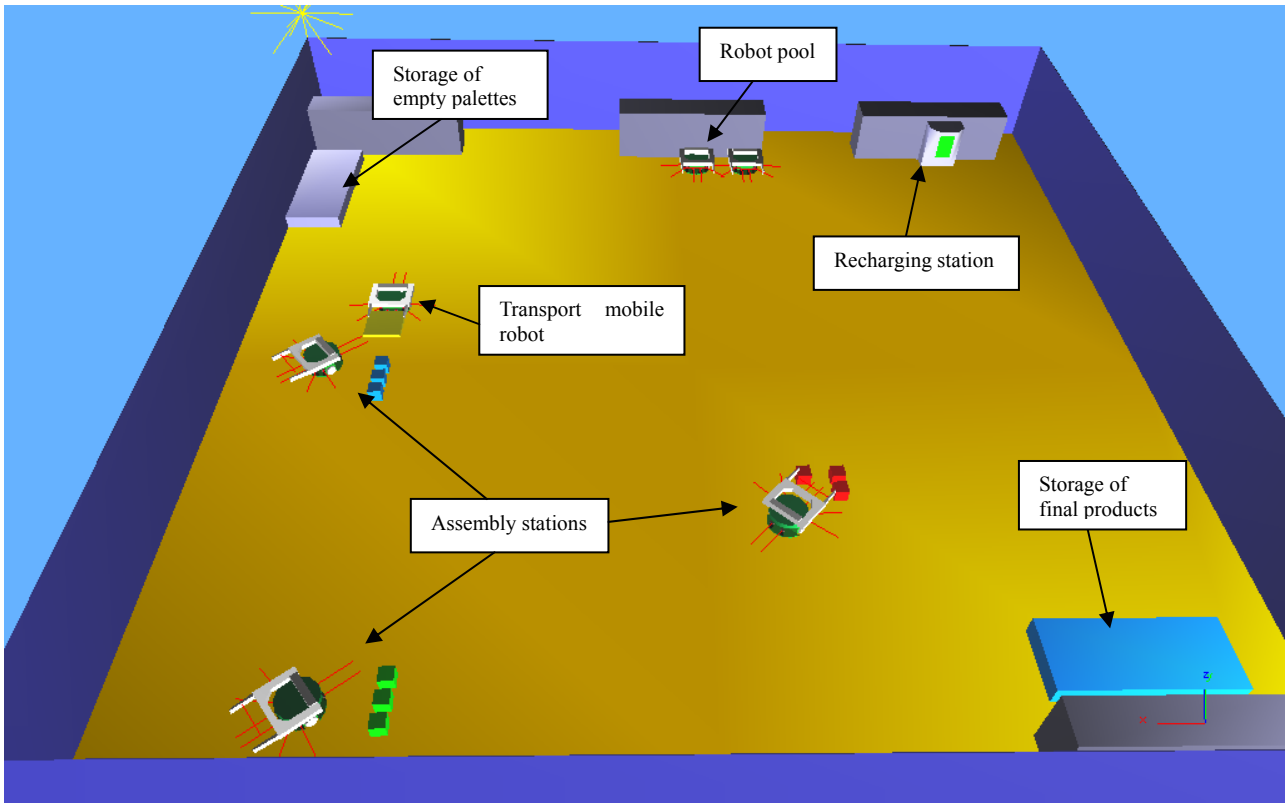


Fig. 4 Simulation of BAS in Webots environment

order which kind of product should be assembled. The products are represented by three cubes which are in three different colours: red, green and blue. Different combinations of these colours represent different kind of products. When the transport mobile robot has the type of the product it has to go to the storage of empty palettes to take one palette on which the product should be assembled. Then it has to go from one assembly station to another in order to assembly the right combination of the cubes. After the product is assembled it has to go to the storage of final product and leave the palette on a predefined place. At this moment, transport mobile robot has fulfilled his assembly mission and it is going back to the initial position (robot pool) and waits for new order to come. Since the robot is equipped with battery sensor which measures the state of the energy level, robot should go to the recharging station first if its battery level is less then 15% of full energy. By this stage one work cycle of a transport mobile robot is finished. All three transport mobile robots are assembling the products at the same time.

At this moment we have developed the controller which navigates the transport mobile robot from the beginning to the end of the cycle. The controller is written in C++ programming language and potential field method described in chapter 4 has been used. The main function looks as follows:

At the beginning every transport mobile robot should get an

```
int main() {
   robot_live(reset);
   get_the_order (); //robot gets the type of product to be assembled (colour combination)//
   potential_field (x_goal,z_goal,x_obstacle,z_obstacle,
   radius_goal,radius_obstacle,spread_goal,spread_obstacle); //robot goes to take the empty palette//

   take_the_carrier (); //robot is taking the palette//

   potential_field (x_goal,z_goal,x_obstacle,z_obstacle,
   radius_goal,radius_obstacle,spread_goal,spread_obstacle);//robot goes to 1st assembly station//
   potential_field (x_goal,z_goal,x_obstacle,z_obstacle,
   radius_goal,radius_obstacle,spread_goal,spread_obstacle);//robot goes to 2nd assembly station//
   potential_field (x_goal,z_goal,x_obstacle,z_obstacle,
   radius_goal,radius_obstacle,spread_goal,spread_obstacle); //robot goes to 3rd assembly station//
   potential_field (x_goal,z_goal,x_obstacle,z_obstacle,
   radius_goal,radius_obstacle,spread_goal,spread_obstacle); //goes to the storage of final product//
   leave_the_carrier (); //robot leaves the palette with assembled product//
   potential_field (x_goal,z_goal,x_obstacle,z_obstacle,
   radius_goal,radius_obstacle,spread_goal,spread_obstacle); //robot goes to initial position//
```

```
    return 0;
}
```

On the Fig. 5 you can see the transport mobile robot approaching the first assembly station with an empty palette with.

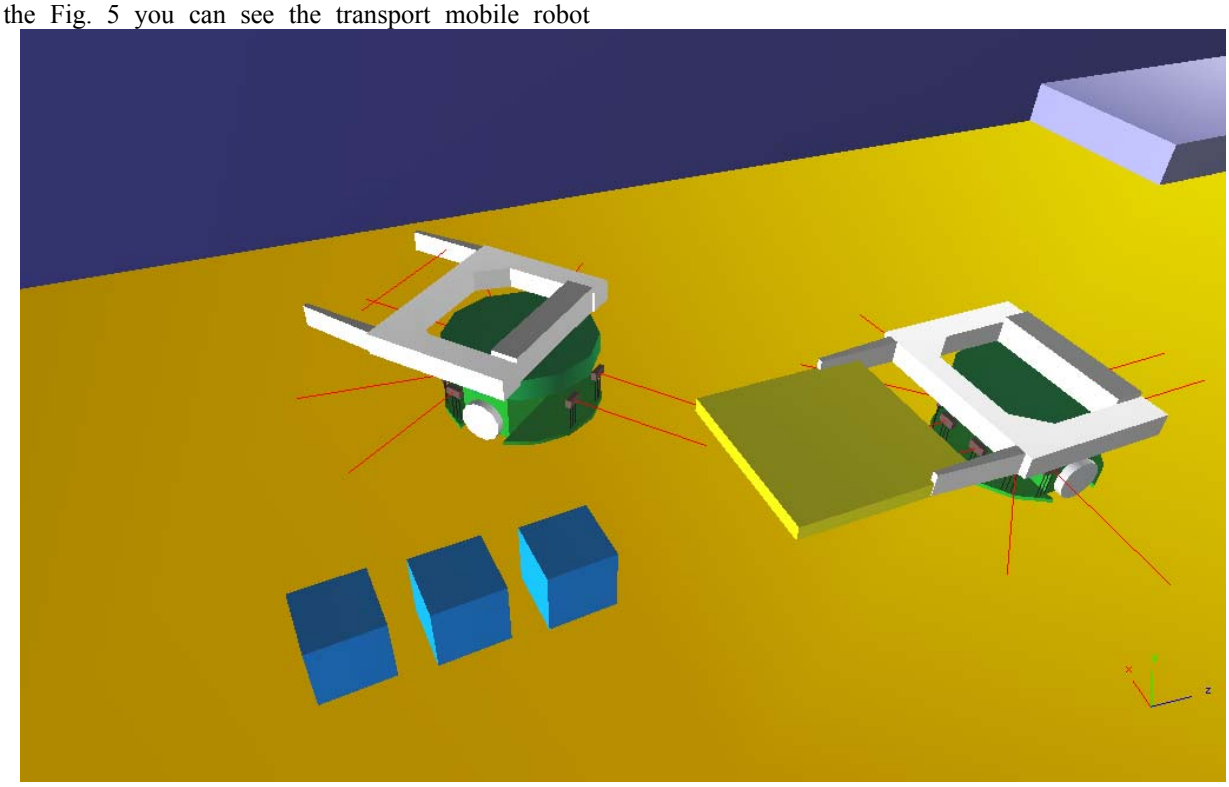


Fig. 5 Transport mobile robot approaching the assembly station robot

V. Summary and Conclusions

In this paper basic characteristics and forms of autonomous mobile robot's behaviour in Bionic Assembly System are presented. We have presented the results of simulation in Webots software. With applying a potential field method into the controller of transport mobile robot in Bionic Assembly System the robot is navigating smoothly around its environment. It is capable to go from one position to another without bumping into the obstacles which are in its way of moving. By this time we have developed only controllers for navigation of transport mobile robots around static obstacle avoidance. That means robots can not yet function in the same time. The next step is to solve the problem when two transport mobile robots have same path, i.e. to solve the problem of moving obstacles. That will be the last step in developing a controller of transport mobile robot and then we have to focus on assembly stations. Assembly stations should be able to know when the transport mobile robot has come and to put the cube on its palette. After we develop a complete simulation in Webots environment, we will try to transfer the controllers on real, physical robots and to test them in the real world.

References

[1] Katalinic, B., "Design of Scheduling Strategies for complex flexible Assembly System for the Mass Production of Electrical Motors", *Proceedings of International Workshop on Emergent Synthesis – IWES 99*, (Editor:K.Ueda), Kobe, Japan, December 6-7, 1999

[2] Kordic, V., "Design and Scheduling of Next Generation of Self-organising Complex Flexible Assembly System in CIM environment", *Dissertation (on German)*, Vienna University of Technology, Vienna, Austria, 2004

[3] Olivier,M., "Cyberbotics Ltd - WebotsTM: Professional Mobile Robot Simulation", *International Journal of Advanced Robotic Systems*, Vol. 1 (2004), pp. 40-43, ISSN 1729-8806, 2004

[4] Latombe, J.C., "Robot Motion Planning", *Kluwer Academic Publishers*, Boston, MA, USA, 1991

[5] Goodrich, M., "Potential Fields Tutorial," Class Notes, downloaded at:
http://www.ee.byu.edu/ugrad/srprojects/robotsoccer/papers/goodrich_potential_fields.pdf

[6] Katalinic,B. & Lazinica, A., "Design of autonomous mobile robots in bionic assembly system: Project concept", *Annals of DAAAM for 2003 & Proceedings of the 14th International DAAAM Symposium*, ISSN 1726-9679, ISBN 3-901509-34-8, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2003