Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

ELSEVIER

Robotics and Autonomous Systems 42 (2003) 1–15

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

# Concepts for dynamic obstacle avoidance and their extended application in underground navigation

Fuyi Xu, Hendrik Van Brussel, Marnix Nuttin*, Ronny Moreas

*Faculty of Engineering, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Celestijnenlaan 300B, B-3001 Heverlee, Belgium*

Received 8 December 2000; received in revised form 15 July 2002
Communicated by F.C.A Groen

**Abstract**

An effective approach to dynamic obstacle avoidance for mobile robots is described in this paper. The main concepts in this approach include local Polar Object Chart (POC) for defining possible ways out, obstacle definition, detour mode, anchor for preventing getting lost, steering delimiting and velocity delimiting. The presented concepts and methods were tested and verified both by simulations and experiments on a developed prototype industrial Automated Guided Vehicle (AGV). This paper also presents the extended application of some of the concepts in underground navigation.
© 2002 Elsevier Science B.V. All rights reserved.

*Keywords:* Mobile robot; Automated guided vehicle; Obstacle avoidance; Underground navigation

## 1. Introduction

The research on dynamic obstacle avoidance for mobile robots introduced in this paper was motivated by an industrial project with an automation company. The goal of the project was to develop a prototype free ranging Automated Guided Vehicle (AGV) for industrial environments [1]. This AGV, named E'GV, is typically used for transporting Euro-pallets. It has a popular three-wheel configuration, with one steer-and-drive wheel. As shown in Fig. 1, it is equipped with two SICK/PLS laser range scanners. The front scanner (1) is used for map building and object detection, while the rear scanner (2) is for pallet localisation and docking.

Dynamic obstacle avoidance is one of the key issues for practical applications of E'GV or other automated guided vehicles or mobile robots. Generally, a path of a mobile robot is statically and globally determined according to a CAD model or pre-detected information about the objects in the environment [2]. In normal situations, the mobile robot follows this path to the desired destination. When unexpected or un-modelled objects, static or moving, appear, the robot should have the ability of dynamically avoiding them, without any collision, and returning to the normal path after these objects have been passed or removed. Dynamic obstacle avoidance is also called local or reactive obstacle avoidance. In contrast to static and global path planning, the dynamic obstacle avoidance approach only uses the local information of the surrounding environment. Consequently, it cannot always generate an optimal path, but it can make the robot react fast to unexpected obstacles.

* Corresponding author. Fax: +32-16-32 29 87.
*E-mail addresses:* fuyi@ai.polymtl.ca (F. Xu), hendrik.vanbrussel@mech.kuleuven.ac.be (H. Van Brussel), marnix.nuttin@mech.kuleuven.ac.be (M. Nuttin).

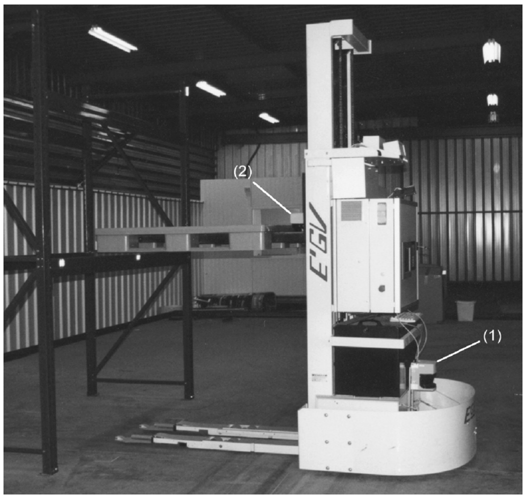

Fig. 1. E'GV: pallet-transporting truck (maximum load: 1000 kg; size: 2 × 1.2 m (length × width); typical speed: 1 m/s forward, 0.5 m/s backward).

The motivation of this research is to develop a safe and practical dynamic obstacle avoidance method for E'GVs and ordinary mobile robots. Moreover, this research also tries to develop a method for robots to explore unknown environments for future use. For this purpose, large obstacles with complicated shapes (e.g. U-shaped obstacles) need to be taken into account.

For dynamic obstacle avoidance, some methods have been proposed [3–5], and various virtual force field (VFF) approaches [6] are widely known. The force field approaches have several severe limitations, such as not allowing the robot to pass through narrow passages, causing instability of motion when travelling within a narrow corridor [6], and the unsuitability to avoid large obstacles with U-shaped configurations. Moreover, the parameters involved are empirically selected, and difficult to be tuned. The vector field histogram (VFH) approach, combined with VFF, is proposed to enhance the ability of selecting the motion direction of the robot [7,8]. However, using the Polar Obstacle Density (POD) in the VFH hides some important information about the spatial distributions of the surrounding objects, and limits the robot to find possible-ways-out in cluttered environments. The dynamics of the robot is considered in [9,10] to get a fast and smooth obstacle avoidance. Robots in these researches are circular-like, and can be regarded as extended points.

This paper introduces a new scheme for dynamic obstacle avoidance, in which human behaviours are mimicked. In this scheme, to find possible ways out of a trapped situation, a local Polar Object Chart (POC) is built using dynamic range data from a range scanner (physical or virtual). The concept of anchors is introduced to remember positions of objects to prevent getting lost. The principal steering direction of the robot is determined according to the relations between the surrounding objects and the desired path. Safe obstacle avoidance is guaranteed by a local protection mechanism that takes into account the orientation and dimensions of the robot to avoid collisions as the robot cuts corners or as the robot's rear end swings out during a turn.

The presented concepts and methods were tested and verified both by simulations and experiments on the developed E'GV. To demonstrate the effectiveness of the presented concepts, this paper also simply introduces their extended application in underground navigation for mining.

This paper is organised as follows. Section 2 briefly introduces the terminology concerning points and ranges used in this paper. Section 3 explains the concept of Polar Object Chart (POC), and gives the definition and the detection method of obstacles. Section 4 describes the determination of the principal steering direction, together with the selection of the detour mode and the concept of anchoring. Section 5 details the steering delimitings to avoid front and side collisions. Section 6 introduces the model for velocity delimiting. Section 7 presents simulations for dynamic obstacle avoidance and experimental results on E'GV, while Section 8 simply introduces the extended application in underground navigation, and Section 9 concludes and gives improvement suggestions.

## 2. Terminology of points and critical ranges

To make it easier to understand the content of this paper, it is necessary to briefly introduce some points on the path and ranges used for the presented dynamic obstacle avoidance first. The points and ranges described below are shown in Fig. 2.

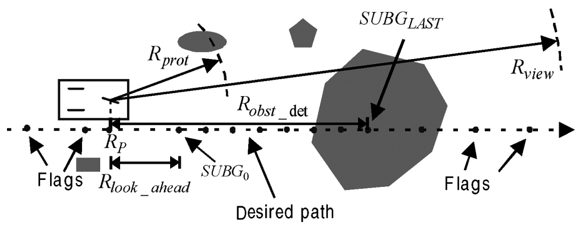

Fig. 2. Points and ranges used.

## 2.1. Flags

Flags are local points on the path, which distribute almost evenly on the current path segment tracked. They reflect the shape of the local path. They are used to calculate the reference position of the robot and sub-goals (as described below), and they also mark the path segment, where the robot should return after an obstacle is passed.

Distances between flags depend on the curvatures of the local path segments and the extent of complexity of the path network. The number of flags $N_{FLAG}$ depends on the size of the largest obstacle in the environment, ensuring that the local marked path should be "visible" (not hidden by objects) from both sides of the largest obstacle.

## 2.2. Reference position

The reference position ($R_P$) of the robot on the desired path is used to locate the robot on the path to let it know which segment of the path it is tracking, and to which segment it should return after an obstacle is passed.

The original definition of the reference position is an orthogonal projection of the current robot position onto the path [11]. This definition cannot be directly used in practice, as in some situations, it may generate several reference positions, or when the robot moves forward, its reference position on the path moves backward on the contrary. To determine the reference position, the movement history of the robot, a step-by-step evolution strategy and flags on the path need to be used.

## 2.3. Sub-goals

Sub-goals are also local points on the path, which are used for path tracking or obstacle detection. In our system, the controller for path tracking is called the sub-goal follower (called goal directed attraction method in our earlier papers [1,12]). It utilises a point ($SUBG_0$ in the figures of this paper, called "carrot" in [11,13]) on the desired path, which is a certain distance (called look-ahead distance $R_{\text{look\_ahead}}$) in front of the reference position of the robot. For path tracking, the steering wheel of the robot points to ("stares at and goes to" in human behaviours) this sub-goal, it guides the robot to the desired path.

Starting from $SUBG_0$ with shorter intervals, known as the obstacle resolution distance $R_{\text{ORD}}$, then the flags, additional sub-goals on the path, reflect the shape of the local path in more details. These sub-goals are used for obstacle detection, as described in Section 3.2.

## 2.4. View range

View range $R_{\text{view}}$ is the valid range of the scanner set for building the Polar Object Chart (POC). Only objects within this range are considered and visible in the POC.

## 2.5. Protection range

Protection range $R_{\text{prot}}$ is typically much smaller than $R_{\text{view}}$, and is used for local protection. Its value depends on the size of the robot. An object within the protection range will be checked if it will cause a possible collision with the robot.

## 2.6. Obstacle detection range

Obstacle detection range $R_{\text{obst\_det}}$ is used for checking if there is an object blocking the path. It equals or less than $R_{\text{view}}$ and greater than $R_{\text{prot}}$, and gives the robot an advanced preparation for avoiding obstacles.

The necessity of using three ranges (protection range, obstacle detection range and view range, ordered according to increasing distances), rather than a stiff distance threshold, can be seen from the effort of Ulrich and Borenstein [14] to make their robot look far ahead, and also can be seen from human's daily driving behaviours. A driver needs to move his head gently up and down to detect variations in depth while looking ahead [15].

## 3. Polar Object Chart and obstacle detection

### *3.1. Polar Object Chart (POC)*

The Polar Object Chart (POC) is a simplified local representation of the surrounding environment which offers an effective means to find possible-ways-out when obstacles are encountered, and to navigate the robot safely around obstacles.

The POC is extracted from a Global Perceptual Space containing a dynamic geometrical map of the environment. The latter is updated on the fly using a laser range scanner while the robot is moving. Using a Global Perceptual Space to perceive the environment adds the ability to keep track of objects that go out of the field of view of the sensor system. It is also possible to introduce virtual objects to the Global Perceptual Space to mark out areas that should be avoided.

The POC stores range values for 360 directions around the momentary position of the robot. The range values give the distance to the closest object in the Global Perceptual Space, i.e. a line segment or cluster [2], in the corresponding direction. Only objects that are within the view range $R_{\text{view}}$ of the POC are considered in this process.

Fig. 3 shows a geometrical map and the related range values stored in the POC. The directions in the POC are labelled with a number greater than or equal to 0 indicating the membership to surrounding objects. Directions that intersect with an object within the view range get a label greater than 0, while those that do not, get label 0. Directions pointing to the same object are given an identical label to indicate that they belong to the same object.

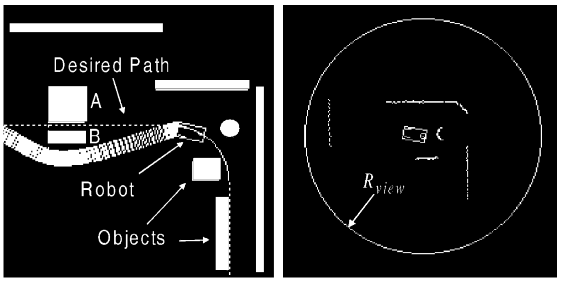

Fig. 3. Environment and related range values in the geometrical map.

The labelling results in an initial clustering of directions that belong to the same identifiable object. Next, the left and right borders of these clusters are investigated to see if they can be combined with their left or right neighbouring cluster. A cluster is combined with its neighbour, if the passage in-between is smaller than the needed minimum corridor width $D_{\text{cor\_w}}$ (equals the width of the robot $Rob_{\text{W}}$ augmented with safety distances at both sides). Any non-object directions, i.e. directions with label 0, between the two clusters that are combined, are transformed into object directions to join the latter. The result is not only that all line segments and clusters that belong to the same physical object are combined into one object, but also the objects that are too close to each other are considered as a single monolithic object. This combination is very important when the environment is cluttered. In experiments in an underground mine, it was found that sometimes, at some directions, the SICK scanner cannot get correct range data (very big values) due to laser reflection, which causes "holes" in the perception. This combination effectively eliminates these holes.

The above found clusters are used to classify the surrounding environment into possible-way-out and no-way-out sectors. The borders of the clusters directly define the classification. The clusters correspond to the no-way-out sectors, and the sectors in-between correspond to the possible-way-out sectors.

Fig. 4 shows the result of this classification for an example environment. For the sake of easy

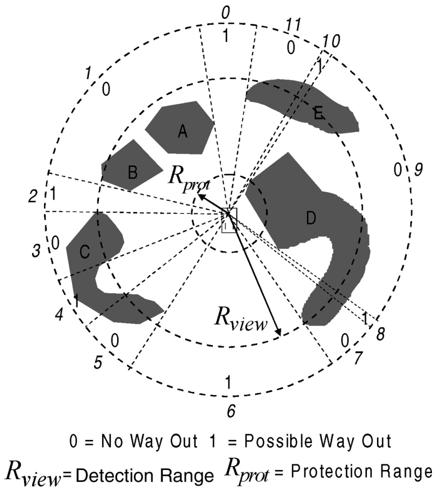
0
1
11
0
10
1
1
0
A
E
B
$R_{prot}$
0
9
2
1
D
3
0
C
4
$R_{view}$
1
8
0
5
0
7
1
6
0 = No Way Out 1 = Possible Way Out
$R_{view}$= Detection Range $R_{prot}$ = Protection Range

Fig. 4. Polar Object Chart around the robot.

understanding and simplicity, only the physical objects rather than their geometrical representation are shown (this is also done for the following figures). Possible-way-out sectors are labelled with 1, and no-way-out sectors with 0. The visible distance between object $A$ and $B$ is smaller than $D_{\mathrm{cor\_w}}$, these two objects are combined, and the relevant range data for the gap in-between are interpolated (can be seen from Fig. 3, the gap between $A$ and $B$ is "closed" in the right figure). Due to the limited range of the view field ($R_{\mathrm{view}}$) of the robot, object C is split into three parts. For object D, by the limitation of perception, one possible way out is generated, this object is also split into three parts. Consequently, sectors 0, 2, 4, 6, 8 and 10 are possible-way-out sectors, among which sectors 4 and 8 are virtual, not physical. Sectors 8 and 10 only denote a possible-way-out direction, its range depends on the resolution of the POC, in our research, 1° is used.

### 3.2. *Obstacle definition and judgement*

Maybe it has been noticed that two terminologies, object and obstacle, are used distinctively in this paper. The distinction of object and obstacle is also made in human's daily life. They have different meanings; a surrounding object is not an obstacle if this object does not block the path even if it is close. In our research, in order to make a close tracking to the path and to prevent unnecessary object detouring in a tight environment, an obstacle is distinguished from ordinary objects. An obstacle is defined as a near object that is on the local path or in the desired moving direction of the robot, as shown in Fig. 5, with the robot being regarded as a point.

To judge if there is an obstacle around the robot, the POC with range values and the local path are used. The local segment is represented by a number of succeeding points: sub-goals.

If, as in Fig. 5a, one of these sub-goals $SUBG_N$ within the obstacle detection range is shadowed and hidden by an object, and its previous sub-goal $SUBG_{N-1}$ is shadowed but not hidden by the same object, this object is on the path. We say that a sub-goal is *shadowed* by an object if the sub-goal and the object are located in a same sector in the POC (for example, both the sub-goal $SUBG_N$ and the object are

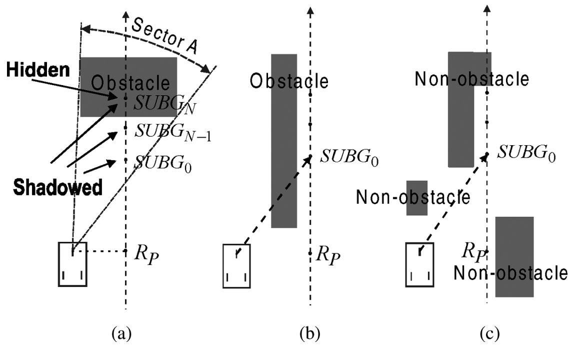

Fig. 5. Obstacle and non-obstacle.

located in sector A, see Fig. 5a). A sub-goal is *hidden* by an object if the range value in the direction of the sub-goal is less than the distance between the robot and the sub-goal, or, in other words, if the sub-goal is invisible due to the existence of the object. In Fig. 5b, the current sub-goal $SUBG_0$ is hidden (of course, is also shadowed) by an object. In both situations, the object is regarded as an obstacle. In Fig. 5c, according to the definition, all of the objects are not obstacles, at least for the actual robot position.

The number of sub-goals $N_{\mathrm{SUBG}}$ depends on the obstacle detection range $R_{\mathrm{obst\_det}}$ and the obstacle resolution distance $R_{\mathrm{ORD}}$, which is the distance between two neighbouring sub-goals. The obstacle resolution distance $R_{\mathrm{ORD}}$ is determined by the minimum size of obstacles desired to be reliably detected once by the algorithm described here. This distance is not very important, because even if a smaller obstacle cannot be detected at the actual robot position, as the sub-goals are moving, it will be finally detected at a later position. A small obstacle resolution distance makes small obstacles easier to be detected by the above algorithm.

## 4. The principal steering direction

In the situation that there is no obstacle detected, the principal steering direction of the robot is determined by the local path and the path tracking controller, even though some modifications are necessarily made to avoid cutting corners or collisions with rear objects during a turn. When an obstacle is encountered, an obstacle avoidance process mimicking human behaviours is triggered.

After finding an obstacle on the road, human follows three steps to pass it:

(1) Decides the detour direction.
(2) Walks around the obstacle.
(3) Finds the old road (visible again on the other side of the obstacle) and returns to it.

### 4.1. Detour mode

There are two detour modes: the left detour and the right detour. Using only local information, it is impossible to make general criteria for selecting an optimal detour mode. An optimal detour mode depends on the shape of the local path, the shape of the obstacle, the relative position of the obstacle to the path, the pose of the robot, and so on. For the sake of simplicity, in our system, the detour mode of the robot is chosen according to the distances of the visible borders (borders of the relevant sector in the POC) to the path, in practical implementation, to the first hidden sub-goal $SUBG_N$ of the sub-goals within the obstacle detection range. As shown in Fig. 6, $Dist_R$ is shorter than $Dist_L$, hence the right detour mode is chosen. An exception is made when some flags are visible behind the obstacle. In that case the detour is chosen on the side where the flags are seen to avoid unnecessary detouring.

The selected detour mode will be kept until the current obstacle is passed or removed. Using detour mode makes the robot have strong abilities to avoid large U-shaped obstacles.

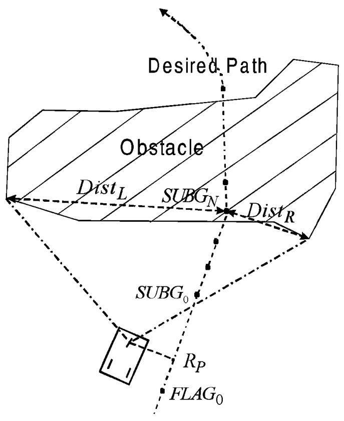

Fig. 6. Detour mode selection.

### 4.2. Obstacle anchoring

To mimic the human behaviour of walking around an obstacle, the robot needs to have the human-like ability of keeping in mind which obstacle he is detouring to prevent getting lost or being trapped in an endless loop. For this purpose, the concept of "obstacle anchoring" is used. An anchor is a representative point for an object. When the robot moves to a new position, anchors are used to determine the relationships between the current surrounding objects and the previous objects, to let the robot know which objects are old and which objects are newly detected. Typically, an anchor can be selected as the nearest point on an object, as shown in Fig. 7. To increase reliability, more points on an object, such as points on the left and right borders, may be used as anchors for a single object.

At position $P_1$, anchor (point) $A_{P_1}$ is selected to represent the current obstacle $A$. When the robot moves to position $P_2$, first of all, the direction of anchor $A_{P_1}$ is calculated in the new local frame, the sector in which this anchor is located is searched in the new POC around $P_2$. In general, because of the discrete characteristics of the geometrical map, the inaccuracy of the range values and newly generated errors of localisation, it is almost impossible to find the corresponding point of $A_{P_1}$ in the new local frame. In practice, a small sector around the direction of $A_{P_1}$ is searched. In the new local frame, if there is an object $\tilde{A}$ that has a distance shorter than the required minimum corridor width $D_{\mathrm{cor\_w}}$ to $A_{P_1}$ (this contributes to strong capacities to endure localisation errors of objects), this object is regarded as the same object with the object $A$ represented by $A_{P_1}$. After the sector, in which the object $\tilde{A}$ is located, is found in the new POC, the new anchor

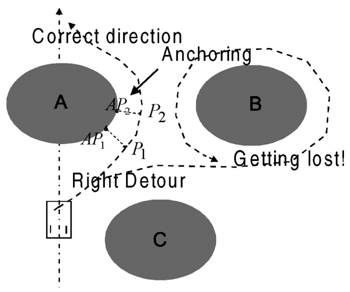

Fig. 7. Obstacle anchoring.

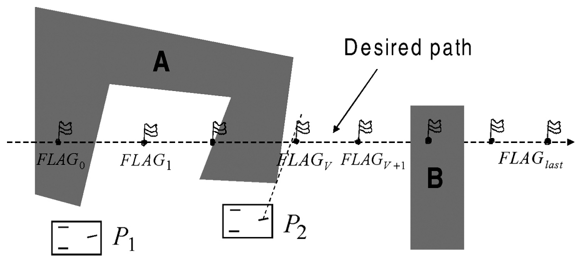

Fig. 8. Releasing and re-anchoring obstacles.

$A_{P_2}$ for obstacle $A$ can be determined. Otherwise, the old obstacle will be regarded as being removed.

### 4.3. Obstacle releasing

The system observes the positions of the path flags with respect to the current obstacle to check if the current obstacle can be released. The obstacle is released if it is either passed or removed. In our implementation, as shown in Fig. 8, the current position of the robot is $P_2$, from the last flag $FLAG_{\text{last}}$, the robot searches backward for the first visible flag $FLAG_V$, from which all of the forward flags (from $FLAG_{V+1}$ to $FLAG_{\text{last}}$) are not hidden by the current obstacle. If the current obstacle is behind the steering wheel of the robot and the distance between the robot and $FLAG_V$ is not too large (say 400 cm), the obstacle is considered to be passed. If however $FLAG_V$ is very close (say less than 200 cm), the obstacle is always considered to be passed in order to avoid unnecessary detouring.

In Fig. 8, after the robot moves to position $P_2$, the current obstacle $A$ is released and a new obstacle $B$ will be anchored.

### 4.4. Determination of the principal steering direction

During the obstacle detouring, the robot runs around the obstacle according to the detour mode until the obstacle is passed or removed. Its principal steering direction is calculated based on the method shown in Fig. 9. During the obstacle avoidance process, the initial principal steering direction $\alpha_i$ of the robot will aim at point $P$ that is $\frac{1}{2}Rob_{\text{W}} + D_{\text{r\_ob}}$ away from the detoured border (in the direction $\alpha_0$), where $Rob_{\text{W}}$ is the width of the robot, and $D_{\text{r\_ob}}$ is the required side safety

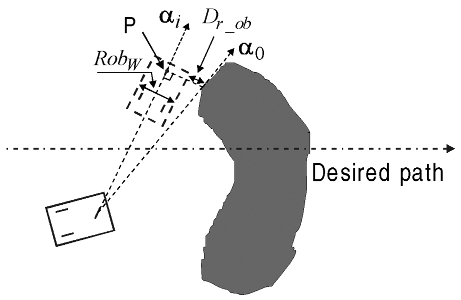

Fig. 9. Principal steering direction of the robot in a left detour.

distance. $\frac{1}{2}Rob_{\text{W}} + D_{\text{r\_ob}}$ is not greater than one half of the required minimum corridor width $D_{\text{cor\_w}}$. The border of the obstacle (possible way out) is searched in the POC. Because of the view range $R_{\text{view}}$ is relatively large, this border determination method of the principal steering direction enables the robot to directly "dash" to the border of the obstacle instead of to move along its contour.

## 5. Steering delimiting

### 5.1. Avoiding collisions with front objects

The principal steering direction of the robot is just a direction of a relatively good possible way out in the view range $R_{\text{view}}$. It cannot make the robot avoid collisions with local front objects. To protect the front parts of the robot, two front protection poles, the left pole and the right pole, are defined in the protection range $R_{\text{prot}}$ to delimit the final steering command. These poles are located at two sides of the principal steering direction, as shown in Fig. 10a, indicating the

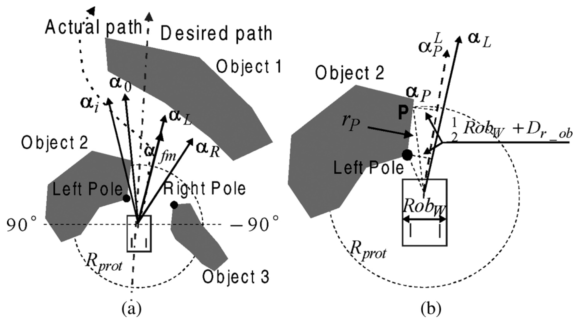

Fig. 10. Left and right poles for front protection.

points on an object that require the most attention. $\alpha_0$ is the direction of the obstacle border (Object 1) selected for detouring, $\alpha_i$ is the principal steering direction. The left and right poles are defined according to the side safety distance $D_{\text{r\_ob}}$ and the width of the robot. The poles are used to define the two steering limits $\alpha_L$ and $\alpha_R$. Fig. 10b illustrates the calculation of the left pole. Within the protection range $R_{\text{prot}}$, direction vector $\alpha_0$ divides the semicircle around the robot into two parts, the left part, from 90° to $\alpha_0$, and the right part, from −90° to $\alpha_0$. For the left part, considering one point, for example, point $P$ in direction $\alpha_P$, it has a range value $r_P$ in the geometrical map. In order to avoid collision with point $P$, the steering wheel should at least aim at the sub-direction $\alpha_P^L$, $\alpha_P^L = \alpha_P - \arcsin((0.5 \times Rob_W + D_{\text{r\_ob}})/r_P)$. The point that is most restrictive, i.e. the one with the rightmost direction vector among these sub-directions like $\alpha_P^L$ is called the left pole, it gives the steering limit $\alpha_L$ for avoiding collisions with the left objects.

Fig. 11 shows the typical relations among the principal steering direction $\alpha_i$ and two steering limits, $\alpha_L$ and $\alpha_R$. In the case of Fig. 11a, $\alpha_i$ is to the right of $\alpha_L$ and the left of $\alpha_R$, the poles will have no effect on the steering angle. In the case of Fig. 11b (as the case shown in Fig. 10a), the steering angle will be delimited by $\alpha_L$ or $\alpha_R$, depending on which side the principal steering direction is located. Finally, in the case of Fig. 11c, the steering angle will be delimited by the steering limit belonging to the closest pole. The steering angle after applying the active steer limit to the principal steering angle is called $\alpha_{\text{fm}}$.

### 5.2. Avoiding collisions with rear objects

The method of avoiding collisions with side and rear objects comes from the apparent fact that, in ordinary situations, if there is no dangerous object in front of the robot and if pure translation (without rotation) is

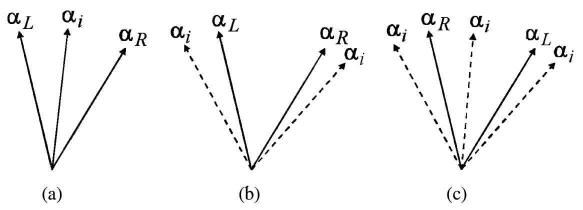

Fig. 11. Typical relations of $\alpha_i$, $\alpha_L$ and $\alpha_R$.

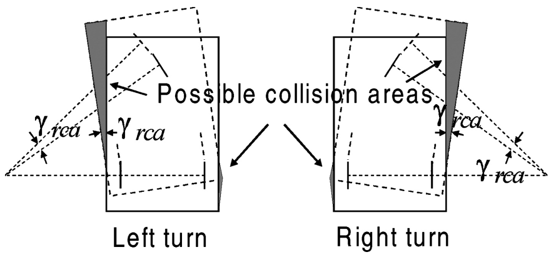

Fig. 12. Possible collision areas.

guaranteed, there will be no collision between the rear of the robot and objects. Rear collisions happen only in the situations that the robot turns left or right. When the robot turns left, the possible collision areas are the front-left and the back-right parts, which will occupy new spaces. When the robot turns right, the possible collision areas are the front-right and the back-left parts, as shown in Fig. 12.

Under the assumption of no slippage, the instantaneous rotational centre of the robot can be calculated according to the steering direction, and the new occupied areas are easy to be calculated. To simplify the calculation and make the robot safer, an extended rectangular robot frame is adopted, with each side $d_{\text{td}}$ away from the rectangular robot frame, and two sector areas with the same angle $\gamma_{\text{rca}}$ are generated, as shown in Fig. 13, and defined by:

$$\gamma_{\text{rca}} = \omega \times T_S$$

where $T_S$ is the sampling interval, and $\omega$ the current angular rate of the robot. Obviously, when $\omega$ equals zero, the above two possible side collision areas disappear.

The above two sectors have different radii, depending on the current velocity of the robot and the steering angle. For the sake of simplicity, the centres and radii

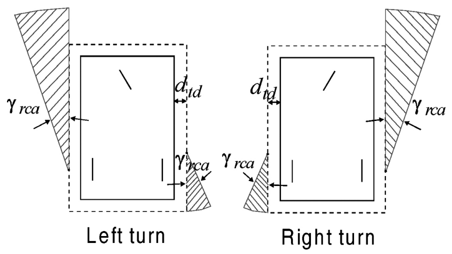

Fig. 13. Simplified possible collision areas.

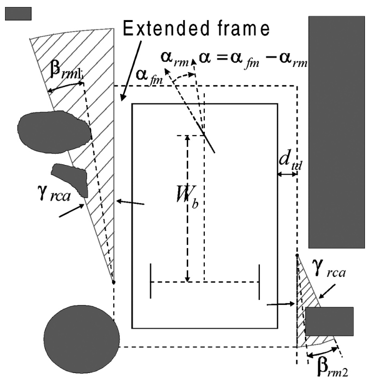

Fig. 14. Modification of the steering direction.

of the two sectors are fixed in the software, depending on the maximum velocity.

Whether or not the steering direction obtained in the above procedures should be modified depends on if there is an object in one of the possible collision sectors. If there is, the required modification value can be calculated easily. Fig. 14 shows the second modification on the steering direction.

The necessary least modification angle $\beta_{\rm rm}$ is the maximum of $\beta_{\rm rm1}$ and $\beta_{\rm rm2}$. $\beta_{\rm rm}$ should be converted into the modification angle $\alpha_{\rm rm}$ to the steering angle

$$\alpha_{\rm rm} = \arcsin\frac{\beta_{\rm rm} \times W_{\rm b}}{V \times T_{\rm S}}$$

where $W_{\rm b}$ is the wheelbase (distance between the steering wheel and the rear wheels) of the robot, $V$ the velocity of the steering wheel.

The final steering angle $\alpha$ of the robot is as follows:

$$\alpha = \alpha_{\rm fm} - \alpha_{\rm rm}.$$

## 6. Velocity delimiting

Due to objects in the environment and robot's limited rotation velocity, the drive velocity should be adjusted. The model in Fig. 15 is used to control the velocity when considering the existence of objects. In this model, a semi-ellipse, with a long axis $E_{\rm L\_Axis}$ and a short axis $E_{\rm S\_Axis}$, and a semicircle, with a radius $r_{\rm C}$, are used. The semicircle acts as a virtual bumper. When all of the objects are beyond the semi-ellipse, the robot can move at the desired maximum velocity $V_{\max}$, if there is no rotation. When there are objects

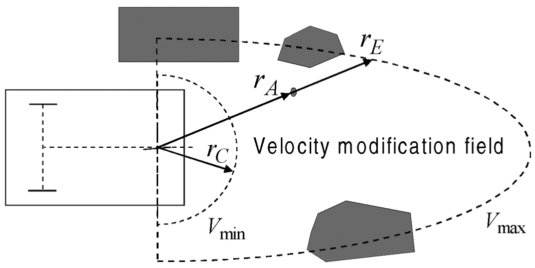

Fig. 15. Model for velocity controller.

in the modification field between the semi-ellipse and the semicircle, a velocity between the maximum velocity $V_{\max}$ and the minimum velocity $V_{\min}$, which is greater than zero and is used to keep the robot moving, will be calculated according to the distances to the objects. For example, $A$ is an object in the modification field, with a distance $r_{\rm A}$ to the steering wheel, $r_{\rm C}$ and $r_{\rm E}$ are the respective radii of the circle and the ellipse in this direction. The linear velocity $V_{\rm A}$ of the robot determined according to $A$ is

$$V_{\rm A} = (V_{\max} - V_{\min}) \times \frac{r_{\rm A} - r_{\rm C}}{r_{\rm E} - r_{\rm C}} + V_{\min}.$$

The velocity $V_{\rm L}$ of the robot equals the minimum value of $V_{\rm A}$ calculated according to all the objects in the modification field. This $V_{\rm L}$ is calculated without considering the rotational motion of the robot. The final velocity $V$ of the steering wheel is

$$V = V_{\rm L}\left(1 - 2\frac{(1-\delta)|\alpha|}{\pi}\right),$$

where $\delta$ is a pre-set factor, it defines the ratio of the velocities when the absolute value of the steering angle $\alpha$ equals $\pi/2$ (rotates on the spot) and 0 (moves without rotation), respectively.

If there is an object within the range of the alarm distance $R_{\rm alarm}$ (equals $r_{\rm C}$, the radius of the semicircle in Fig. 15), the robot stops and alarms.

## 7. Simulations and experiments on dynamic obstacle avoidance

### 7.1. Simulations

In order to simulate the dynamic obstacle avoidance method presented in this paper, together with the path

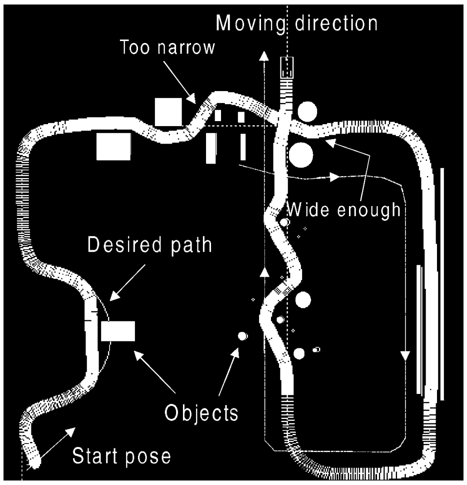

Fig. 16. Simulation 1 on dynamic obstacle avoidance.

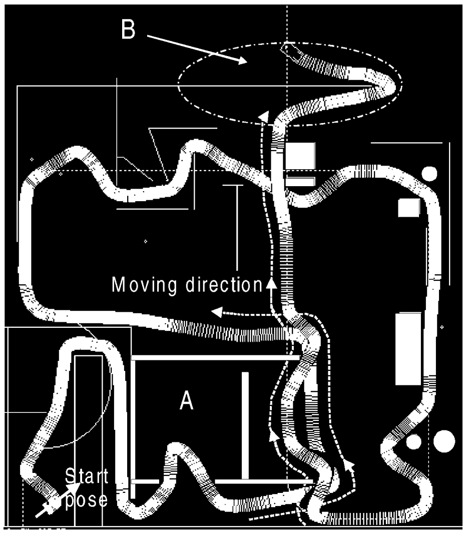

Fig. 17. Simulation 2 on dynamic obstacle avoidance.

tracking strategy developed for E'GV, the Global Perception Space is loaded with some a priori objects, while others can be introduced dynamically at random places by the user. To perceive the virtual surrounding environment, a virtual laser range scanner that can provide the 360° perception information with uniform-distribution random noise from −8 to 8 cm is used. Fig. 16 shows the results of one simulation. Some of the parameters used are shown in Table 1, $Rob_L$ is the length of the robot.

In Fig. 16, the dotted line without arrowheads is the desired path, and the thick white trail of the robot stands for its actual trace. Some circular-like objects with random sizes and positions in certain ranges are added temporarily during the tracking for the middle path segment in this figure. The density of the actual trace reflects the velocity of the robot, the brighter the trace is, the slower the robot moves.

Table 1
Some parameters used in the simulations

| | | | |
|---|---|---|---|
| $Rob_L$ (cm) | 150.0 | $Rob_W$ (cm) | 80.0 |
| $D_{cor\_w}$ (cm) | 120.0 | $D_{r\_ob}$ (cm) | 15.0 |
| $R_{view}$ (cm) | 800.0 | $R_{obst\_det}$ (cm) | 360.0 |
| $R_{prot}$ (cm) | 200.0 | $R_{alarm}$ (cm) | 50.0 |
| $R_{look\_ahead}$ (cm) | 50.0 | $R_{ORD}$ (cm) | 40.0 |
| $N_{SUBG}$ | 10 | $N_{FLAG}$ | 25 |
| $V_{max}$ (cm/s) | 120.0 | $V_{min}$ (cm/s) | 5.0 |
| $E_{L\_Axis}$ (cm) | 400.0 | $E_{S\_Axis}$ (cm) | 180.0 |

Fig. 17 shows the simulation result in a more complicated environment, the obstacle sizes in this simulation are much larger than those in Fig. 16. The robot needs to explore a way out and sometimes runs into a dead-end (such as a room, many obstacle avoidance approaches are easily trapped in this kind of U-shaped obstacles). The advantage of using a view range $R_{view}$ in building the POC is that the robot can look far ahead and decide its next behaviour earlier. With the help of the presented methods, while exploring room *A* (with a size of 800 cm × 900 cm) in Fig. 17, the robot can detect that there is no other way out except the entrance, it directly gets out. The width of the opening at the upper-right corner of room *A* exactly equals $D_{cor\_w}$, due to noise (for safety reasons, objects are also extended a little bit in simulations), it is regarded as not being wide enough for the robot to be able to pass.

It must be pointed out that sometimes, in an unknown environment, or even in a previously well-modelled environment, because of the nature of the dynamic perception and the limited range of the view field ($R_{view}$) of the robot, a wrong detour direction is possibly chosen. For example, in area *B* in Fig. 17, if the wall is long enough at the right side of the path, the robot possibly selects the left detour mode and tries to find a way out in the left area, it will cause a deadlock. Mimicking human behaviour also can solve this problem. In an unknown environment, one currently optimal detour direction can be tried first. If there is no way out found in this direction after a long distance has been explored, the robot should turn back to the fork place and try another direction. Some anchors are necessarily used in this process so that the robot does not get lost. Up to now, this detour trial function has not been implemented in our system.

### 7.2. Experiments: application to the E'GV

The concepts and methods presented in this paper for dynamic obstacle avoidance have been implemented in E'GV [1]. Some parameters used in E'GV are listed in Table 2.

Table 2
Some parameters used in E'GV

| | | | |
|---|---|---|---|
| $Rob_L$ (cm) | 200.0 | $Rob_W$ (cm) | 120.0 |
| $D_{cor\_w}$ (cm) | 180.0 | $D_{r\_ob}$ (cm) | 10.0 |
| $R_{view}$ (cm) | 700.0 | $R_{obst\_det}$ (cm) | 700.0 |
| $R_{prot}$ (cm) | 200.0 | $R_{alarm}$ (cm) | 70.0 |
| $R_{look\_ahead}$ (cm) | 100.0 | $R_{ORD}$ (cm) | 5.0 |
| $N_{SUBG}$ | 120 | $N_{FLAG}$ | 25 |
| $V_{max}$ (cm/s) | 100.0 | $V_{min}$ (cm/s) | 10.0 |

In E'GV, the POC was built in 360°, only the front scanner was used for map building, the rear objects in the POC were loaded from memory. The detection range was 7 m, objects were registered in the global map (which is used by the POC) very soon. The latency of the scanners was less than 200 ms, when the

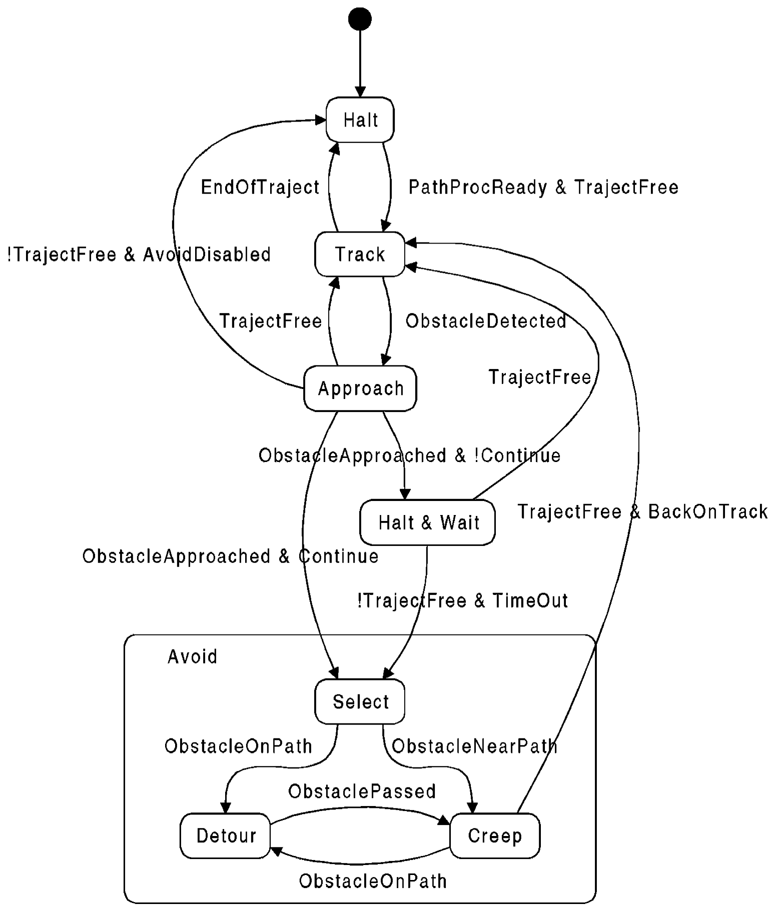

Fig. 18. State diagram of obstacle avoidance module in E'GV.

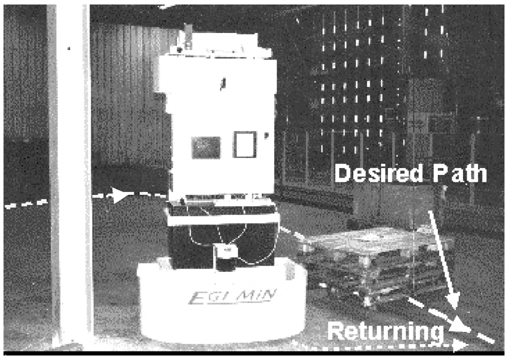

Fig. 19. Actual obstacle avoidance of E'GV.

vehicle moved at 1 m/s, the distance covered between two samplings was less than 20 cm, there was no trouble to detect obstacles and avoid them in experiments. This is due to the strong capacities of the anchors to endure localisation errors of objects.

In order to limit its operating range, some virtual objects were used. Fig. 18 shows the state transition diagram of the obstacle avoidance module in E'GV. To increase safety of operations, some halts and waits were inserted. In the on-board industrial PC i486 (66 MHz), an integral calculation period (positioning, control, geometrical map building etc.) required about 100 ms processing time. With respect to the complete cycle, the calculation time of the POC can be ignored:

- In Track Mode, E'GV tracks the set points (sub-goals) of the Path Processor.
- If an obstacle is encountered on the trajectory, E'GV will approach it to a certain distance.
- At the desired approach distance, E'GV will go into Avoidance Mode, or optionally will first Halt and Wait for a certain time before going into avoidance.
- E'GV will detour around the obstacle if it blocks the path. The Path Processor is switched off.
- If the obstacle does not block the path, E'GV switches to Creep Mode. The Path Processor stays active in combination with the Steer Delimiter.

Fig. 19 shows the situation in which the E'GV was avoiding an obstacle and smoothly passing between two objects. This dynamic obstacle avoidance function satisfies the requirements of the practical application of E'GV in industrial environments.

## 8. Extended application in underground navigation

It is currently recognised by mining society that autonomously guided underground mining vehicles may greatly increase the efficiency of mining production and improve human safety. Autonomous underground navigation has been extensively studied in recent years [16–19].

One of the most difficult issues in underground navigation is to switch to a desired drift at intersections. For this purpose, the system should be able to find the expected intersection and distinguish the desired drift from other drifts at the intersection. As described above, the POC is built to separate the local environment into different parts, it provides a possible way to find the expected intersection with certain degree of freedom (number of connected drifts) and locate the desired drift. To explore a new method for underground navigation, simulation software based on the above concepts presented for dynamic obstacle avoidance was developed. A simulation result for underground navigation is shown in Fig. 20. In the simulation, the following methods were used:

- Similar with the above simulations for obstacle avoidance, a virtual range scanner with noise was used in the simulation for underground navigation,

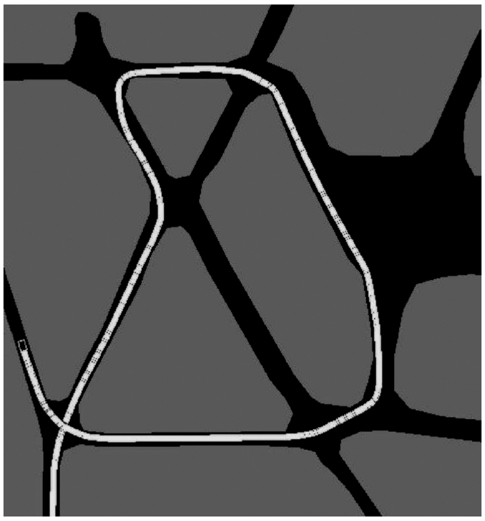

Fig. 20. Simulation of the extended application in underground navigation.

it provided a real-time 180° scanning of the local environment, information about the rear environment of the robot was provided by previous data.

- The desired path was described as a sequence of drifts connected through intersections with certain degrees of freedom.
- To increase the reliability of detecting the expected intersection and the desired drift, conformation based on distance travelled was used.
- Walls (separated objects) were numbered counterclockwisely from the right wall of the current drift. Thus, the desired drift can be described as either being at the left side of the $n$th wall or being at the right side of the $(n-1)$th wall.
- Different to the dynamic obstacle avoidance approach described above, the detour mode was determined according to the topological distributions of the drifts along the desired path, and it was used to follow a wall in a drift. Following the left wall in a drift is equivalent to using right detour of the left wall, and following the right wall is equivalent to using left detour of the right wall.
- Anchors were used to remember positions of walls and guide the robot to move from the current drift to the desired drift, during which no wall can be followed.
- Similar with dynamic obstacle avoidance, the principle steering direction, steering delimiting and velocity delimiting were used in the simulation to prevent collisions.

## 9. Conclusions and suggestions for improvement

From simulations and the actual experiments of E’GV in industrial environments, the dynamic obstacle avoidance scheme presented in this paper is verified to have the following features:

- All the parameters involved have physical meanings, easy to be determined.
- The system has a strong ability to select optimal detour direction and to smoothly pass through narrow corridors.
- The system has a strong ability to run in relatively complicated or unknown environments.The concepts presented for dynamic obstacle avoidance can also be used in other applications. They provide a possible solution to underground navigation for mining industry, according to our preliminary simulation.

Besides the basic function of avoiding obstacles, using the new obstacle avoidance method has an additional effect on the path planning. In general, at the stage of static path planning, to avoid collisions, the path is deliberately refined. If the obstacle avoidance method described here is used, the pre-designed path can be relatively coarse, even allowing collisions with the environment at corners. Thanks to the use of the POC, the robot is able to navigate safely around corners and avoid collisions. This makes the path planner simpler. Therefore, the expression of the path can be simple, even just a combination of line segments.

Besides the detour trial function presented in Section 7.1, there are some aspects allowing for improvement:

- It is better to combine the dynamic perception information with the global known information about the environment in selecting the detour mode. In a partly known environment, it is possible and it can get a detour direction as optimal as possible.
- If sonar sensors are used to perceive the environment, a certainty POC needs to be built.

## Acknowledgements

The financial support of EGEMIN company is gratefully acknowledged. This research has partly been carried out with the financial help of IWT (Flemish Institute for Industrial Scientific and Technological Research).

Basic research results used in this project have been obtained in the framework of the Inter-University Attraction Poles, Office of the Prime Minister, projects IUAP-50 and IUAP4/24/ImechS.

## References

[1] H. Van Brussel, R. Moreas, F. Xu, J. Vercammen, E’GV: a truly free-ranging AGV for industrial environments, in: Proceedings of the 31st International Symposium on Robotics, Palais des Congrès, Montreal, Quebec, May 14–17, 2000.

[2] J. Vandorpe, Navigation techniques for the mobile robot LiAS, Ph.D. Thesis 97D3, PMA/Katholieke Universiteit Leuven. ISBN 90-5682-060-5.

[3] J.R. Andrews, N. Hogan, Impedance control as a framework for implementing obstacle avoidance in a manipulator, in: D.E. Hardt, W. Book (Eds.), Control of Manufacturing Processes and Robotics Systems, ASME, Boston, MA, 1983, pp. 243–251.
[4] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, St. Louis, MO, March 25–28, 1985, pp. 500–505.
[5] D. Fox, W. Burgard, S. Thrun, Controlling synchro-drive robots with the dynamic window approach to collision avoidance, in: Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems, Osaka, Japan, 1996.
[6] Y. Koren, J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, in: Proceedings of the IEEE Conference on Robotics and Automation, Sacramento, CA, April 7–12, 1991, pp. 1398–1404.
[7] J. Borenstein, Y. Koren, The vector field histogram—fast obstacle avoidance for mobile robots, IEEE Journal of Robotics and Automation 7 (3) (1991) 278–288.
[8] I. Ulrich, J. Borenstein, VFH+: reliable obstacle avoidance for fast mobile robots, in: Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 16–21, 1998, pp. 1572–1577.
[9] D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance, IEEE Robotics Automation Magazine 4 (1) (1997).
[10] R. Simmons, The curvature-velocity method for local obstacle avoidance, in: Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN, 1996.
[11] A.L. Rankin, C.D. Crane III, D.G. Armstrong II, Evaluating a PID, pure pursuit, and weighted steering controller for an autonomous land vehicle, Proceedings of SPIE 3210 (1997) 1–12.
[12] F. Xu, R. Moreas, H. Van Brussel, A new dynamic obstacle avoidance method for mobile robots, in: Proceedings of the Third European Advanced Robotics Systems Masterclass and Conference—Robotics 2000, University of Salford, Manchester, UK, April 12–14, 2000.
[13] J.S. Wit, Vector pursuit path tracking for autonomous ground vehicle, Ph.D. Dissertation, Department of Mechanical Engineering, University of Florida, 2000.
[14] I. Ulrich, J. Borenstein, VFH*: local obstacle avoidance with look-ahead verification, in: Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, April 24–28, 2000, pp. 2505–2511.
[15] Fr. Paesschierssens, F. Konincks, G. Vervust, Driving in Belgium—mastering the Belgian highway code, New Traffic Books NV, Aalst-Hofstade, Belgium, March 1996, p. 31. ISBN 90-6847-052-3.
[16] Y. Laperriere et al., Pilot production with autoguided truck at Brunswick mining, in: Proceedings of the CIM Conference, Montreal, Que., May 1998.
[17] L. Bloomquist, E. Hinton, Towards an infrastructure guidance system, in: Proceedings of the First International Workshop on Advances in Robotics for Mining and Underground Applications, Brisbane, Australia, October 2–4, 2000.
[18] J. Bakambu, F. Xu, V. Polotski, P. Cohen, Guideless Autonomous System for Underground Navigation, CIM Conference Vancouver, BC, April 28–May 1, 2002.
[19] J.M. Roberts et al., Autonomous control of underground mining vehicles using reactive navigation, in: Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, April 2000.

**Fuyi Xu** (1967) received B.Eng., M.Eng. and Ph.D. in mechanics from Shanghai Jiaotong University in 1988, in computational mechanics from Huazhong University of Science and Technology in 1991, and in mechanical engineering from Tsinghua University, PR China, in 1995, respectively. From 1995 to 1998, he was a lecturer in the Laboratory of Control and Navigation, Department of Precision Instruments, Tsinghua University. From May 1998 to February 2001, he worked as a researcher on mobile robots in the division of Production Engineering, Machine Design and Automation (PMA), Department of Mechanical Engineering, Katholieke Universiteit Leuven (K.U. Leuven), Belgium. From February 2001, he worked as a researcher in Perception and Robotics Laboratory of Ecole Polytechnique de Montreal, Canada. His research interests include experimental and computational mechanics, navigation, control and mobile robot.

**Hendrik Van Brussel** (1944) is the full professor in mechatronics and automation at the Faculty of Engineering, K.U. Leuven, Belgium, and Chairman of its Department of Mechanical Engineering. He received his B.Sc., ME (Technisch Ingenieur) degree from Hoger Technisch Instituut, Oostende, Belgium, in 1965, and his M.Sc., EE (Burgerlijk Ingenieur) and Ph.D. degrees from K.U. Leuven, Belgium, in 1968 and 1971, respectively. From 1971 to 1973 he was active in Bandung, Indonesia, establishing a Metal Industries Development Centre and as an associate professor at Institut Teknologi Bandung. He was a pioneer in robotics research in Europe and an active promoter of the mechatronics idea as a new paradigm in machine design. He has extensively published on different aspects of robotics, mechatronics and flexible automation. His present research interest is also shifting towards holonic manufacturing systems, behaviour based robots, and micro and precision engineering, including microrobotics. He is a Fellow of SME and IEEE and in 1994 he received a Honorary Doctor degree from the 'Politehnica' University in Bucarest, Romania and from Rheinisch-Westfälische Technische Hochschue (RWTH), Aachen, Germany. He is a member of the Royal Flemish Academy of Belgium for Sciences and Fine Arts and President of CIRP (International Institution for Production Engineering Research) for the period 2000–2001.

**Marnix Nuttin** studied electrical engineering at K.U. Leuven, with as specialisation computer systems and automation and he obtained his Master's degree in 1992. He obtained his Ph.D. degree in 1998 at the Division of Production Engineering, Machine Design and Automation, K.U.Leuven. His research interests are learning robots for assembly and mobile manipulation, programming by demonstration, shared autonomy for wheel chair control and behaviour based robot co-operation. He worked on several projects, e.g. the European project B-Learn II developing learning techniques for robotic applications, a project with Procter Gamble etc.

**Ronny Moreas** was born in 1970 at Tongeren, Belgium, obtained the degree of Industrial Engineer in Electronic Engineering from the Industriele Hogeschool v/h Rijk in Hasselt, Belgium, in 1993, and an engineering degree in computer science at M.Sc. level from the Katholieke Universtiteit Leuven, Belgium, in 1996. Since 1996, he is with the Department of Mechanical Engineering of K.U. Leuven, working on mobile robot applications for industrial environments and recently on path planning for payload support systems for planetary exploration. At present, he is an ICT manager at the same department.