Proceedings of the
2007 IEEE Intelligent Vehicles Symposium
Istanbul, Turkey, June 13-15, 2007

Obstacle Avoidance for Autonomous Ground Vehicles in Outdoor Environments

Cellini M., Mati R., Pollini L., and Innocenti M., *Senior Member IEEE*

***Abstract* — The paper presents an obstacle avoidance algorithm to be used for autonomous ground vehicles applications. The proposed method improves some of the limitations of the recently developed Null Space Based Behavioral Control. The technique divides the problem into tasks, which are associated to increasing priority. Activities with lower priority do not interfere with those having higher priority. The scenario is supposed known only partially, and the complete environment is reconstructed during the mission, with the aid of stereoscopic vision sensors. The validity of the method is currently verified via computer simulations.**

I. Introduction

Obstacle avoidance is one of the more complex problems to be addressed within the context of autonomous vehicles guidance design. Difficulties increase if the initial knowledge of the scenario is limited, and the outside environment must be reconstructed online, during the motion of the vehicle, and without apriori information and/or clues.

The literature offers a large number of methods for the solution of the obstacle avoidance problem, and several of them use modifications of the potentials algorithms adapted to represent vehicle trajectories and paths. Potential-based techniques have the advantage of being straightforward and of easy implementation. One of the limitations encountered by these methods is the presence of local minima, which can be addressed in several ways, for instance by using harmonic functions [1]. In addition, the complexity of the scenarios is limited under the application of this methodology.

Fuzzy Logic and Neural Networks have been used in the past and are used currently in the development of obstacle avoidance algorithms [2][3][4]. They can be considered as "intelligent systems" in that they use strategies of optimality based on various types of training, and data processing structure. Performance can be very good, however they require fine tuning, and training.

The present work deals with the design of an obstacle avoidance scheme for a specific vehicle: the autonomous ground system Ulisse [5]. Ulisse is a three-wheeled vehicle originated from a motorized golf kart, and modified in the Department of Electrical Systems and Automation of the University of Pisa, to serve as a test bed for autonomous control, guidance, and navigation research. The vehicle is currently using stereoscopic vision sensors, and clustering techniques in order to detect and isolate obstacles within the field of view of the cameras.

For the purpose of this work, we assume partial knowledge of the outside scenario: some obstacles are well known apriori (such as buildings, lakes, prohibited areas, and other obstacles, like those available in up-to-date navigation maps), others must be identified during the motion. A path planning algorithm is used, which incorporates known obstacles, and the proposed obstacle avoidance system is switched on when need arises.

The problem of obstacle avoidance is addressed by decomposing it into smaller and simpler subtasks, so that the procedure is modular, and more versatile. These concepts were used in a recently developed method called Null Space Based Behavioral Control (or NSBBC) [6], which however has several limitations in its application to large trajectories in an outdoor environment. The method in [6] will be reviewed and improved, so that it can be applied to the problem at hand.

II. Null-Space-Based Behavioral Control

The NSBBC method addresses obstacle avoidance in a multiple obstacles context by subdividing the scenario in smaller tasks of lesser complexity. The basic idea originates in Robotics, with the control of redundant manipulators. The added degrees of freedom are used to minimize a specific functional so that the manipulator can

The Authors are with the Department of Electrical Systems and Automation (DSEA), University of Pisa, Via Diotisalvi 2, 56126 Pisa, ITALY. (e-mail: minnoce@dsea.unipi.it). This work was performed with the support of the Italian Ministry for University Research under Grant 2005097207..

1-4244-1068-1/07/$25.00 ©2007 IEEE.

execute accessory tasks, without modifying the path followed by the end effector.

Let us consider a generic manipulator; the end effector velocity can be written as a function of the joints variables set as:

$$\dot{x}_e = J_e(q)\cdot\dot{q} \tag{1}$$

where $\dot{q}$ is the nth-dimensional joint velocities vector, and $\dot{x}_e$ is the end effectors' velocity vector. By inverting eq. (1), it is possible to compute the joints velocity needed, in order for the end effector to move at a given speed as shown in eq. (2).

$$\dot{q} = J_e^{\dagger}(q)\cdot\dot{x}_e \tag{2}$$

In general, additional degrees of freedom are used to obtain a solution of (2) that minimizes some joint velocity norm. It is possible, however, to get a non minimum norm solution, and to use the redundancy for other objectives. A possible non minimum norm solution that uses the Jacobian pseudo inverse $J_e^{\dagger}(q)$ is given by:

$$\dot{q} = J_e^{\dagger}(q)\cdot\dot{x}_e + \left(I - J_e^{\dagger}(q)\cdot J_e(q)\right)\cdot\dot{q}_a \tag{3}$$

The term $\dot{q}_a$ in eq. (3) is projected onto the $Ker(J_e)$; thus $\dot{q}_a$ has no effect on the end-effector trajectory x_e and can be used to manage the redundancy.

This technique ca be used in a task oriented paradigm for control of a robot manipulator where x_e is the priory task and $\dot{q}_a$ can be used to perform additional tasks that do not effect the end effector motion [6].

The property outlined above can be generalized from the well known case of redundant manipulators, to problems requiring splitting an objective into tasks having different priorities, such as the motion control of an autonomous vehicle within an environment with obstacles [7].

In an obstacle avoidance scenario, the vector $\dot{q}$ represents the planar velocity, and the vector x_e becomes a variable associated to the task. With this position, eq. (2) becomes:

$$\dot{p}_d = v_d = J^{\dagger}(p_d)\cdot\dot{\sigma} \tag{4}$$

Where $p_d = p_d(t)$ is the time dependent desired position at some time t, v_d the vehicle's desired velocity, J the matrix of velocities allowed for the task, and σ the process variable that must be controlled. Eq. (4) is modified with the addition of a control term for closed loop position error elimination; thus:

$$\begin{aligned} v_d &= J^{\dagger}(p_d)\cdot\left(\dot{\sigma} + \Lambda\cdot\tilde{\sigma}\right) \\ \tilde{\sigma} &= \bar{\sigma} - \sigma \end{aligned} \tag{5}$$

where $\bar{\sigma}$ is the desired value of the process variable, and Λ is a positive definite matrix. We can use σ e J to define tasks appropriately, and then compute the speed necessary to accomplish them.

If there are multiple simultaneous tasks, a supervisor must be designed for assigning priority and subsequent ordering. For this, it is possible to use the relationship:

$$\begin{aligned} v_{i+1} = J_{i+1}^{\dagger}(p)\cdot\left(\dot{\sigma}_{i+1} + \Lambda_{i-1}\cdot\tilde{\sigma}_{i-1}\right) + \\ + \left(I - J_{i+1}^{\dagger}(q)\cdot J_{i+1}(q)\right)\cdot v_i \end{aligned} \tag{6}$$

The supervisor starts with the lowest priority task (index 0), up to the task with the highest priority.

As an example, let us consider a vehicle moving on a planar surface, which must reach a target avoiding point mass obstacles. Each obstacle is enclosed by a safety circle, which can not be crossed by the vehicle. Two tasks can be identified:

1. to reach the target,
2. to remain outside the safety circles.

The critical task is the obstacle avoidance, and it receives higher priority. In order to avoid a collision, the task acts on the vehicle's velocity in the direction of the obstacle. The magnitude is a function of the distance to the obstacle and it become zero in proximity of the safety circle. (for instance if equal to the circle's radius *d*). If the vehicle goes beyond the circle's limit, the speed sign is reversed. In the example, the parameters J_o, σ_o, $\bar{\sigma}_o$ in (6) have the following meaning:

J_o represents the velocity direction on which the task can operate,

σ_o is the distance between vehicle and obstacle

$\bar{\sigma}_o$ is the safety circle radius.

Therefore we have:

$$J_o = \left(\frac{p - p_o}{\|p - p_o\|}\right)^T ; \sigma_o = \|p - p_o\|; \bar{\sigma}_o = d \tag{7}$$

Thus:

$$v_o = J_o^{\dagger}\cdot\Lambda_o\cdot\left(d - \|p - p_o\|\right) \tag{8}$$

Where

p is the vehicle position, p_o is the obstacle position

Since the process yielding the vehicle to the goal must be able to impose any velocity direction, J_g is a 2x2 identity matrix, which identifies all possible velocities on the plane. The planar position is the task variable (σ_g), which must be reach the desired value ($\bar{\sigma}_g$). Therefore:

$$J_g = I; \sigma_g = p; \bar{\sigma}_g = p_g \tag{9}$$

and:

$$v_g = \Lambda_g \cdot \left(p_g - p \right) \qquad \textbf{(10)}$$

Matrix J_g is full rank, thus its null space is empty. The processes with priority lower than the main one are thus projected onto an empty subspace and to not influence the motion of the vehicle.

III. NSBBC: THE SUPERVISOR

The priority of the various processes is managed by a supervising algorithm, which determines the state of the vehicle with respect to the scenario, and defines the rules to be used. In reference [2] no mention was made on how the rules were obtained, therefore a numerical simulation was performed in order to define their possible structure. In the case of a single obstacle, the main rule is that if the distance between target and vehicle is greater than the distance between target and obstacle, then the primary task is obstacle avoidance and secondary task is reaching the target (and the other way around of course).

If the scenario has many obstacles, to each one a process is associated, which indicates speed and direction necessary for the avoidance. The supervisor orders the obstacles by their distance to the vehicle, with their relative priority. The obstacles that were already passed are given a priority lower than the one relative to the target.

IV. LIMITATIONS OF THE NSBBC ALGORITHM

Extensive simulation tests were performed in order to evaluate the performance of the algorithm. The values of Λ_o and Λ_g in equations (8) and (10) were selected according to what suggested in [7]. The numerical values are:

$$\Lambda_o = 0.5; \Lambda_g = 1$$

1) One obstacle

If the line connecting the vehicle and the target crosses the obstacle, the two processes (vehicle to target, and obstacle avoidance) create the same vector. This means that the projection of v_g on the null space of J_o gives a zero vector and the vehicle will proceed on a straight line following the vector v_o until it stops at the perimeter of the safety circle.

The problem does not arise of course when the obstacle is not in the Line of Sight as shown in Figure 1, and the vehicle continues to the target.

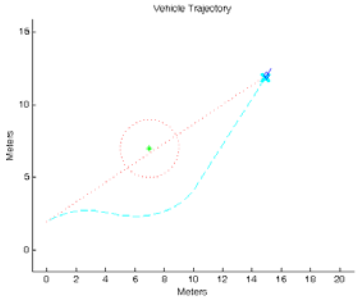

Fig.1: Vehicle Trajectory with one Obstacle outside the LOS.

In conclusion, with one obstacle the vehicle always reaches the target, except for the singularity case.

2) More obstacles

When safety circles intersect, the algorithm naturally creates a velocity vector as a combination of components directed towards the obstacles. This yields an "attractive" effect to the obstacles groupings. This feature increases as the ratio Λ_o / Λ_g increases. In this situation, it can happen that the vehicle goes through an area where two safety circles intercept, and since there is a different priority as a function of distance, it may enter the safety circle of the more distant obstacle.

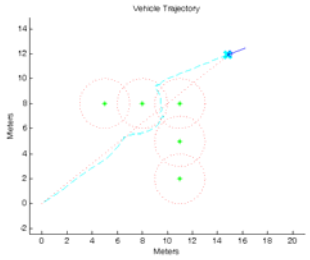

Fig.2: Vehicle Trajectory with more Obstacles near to each other.

Obviously, the more separated the obstacles are, the easier is for the vehicle to avoid all the safety circles in order to reach the target.

V. OBSTACLE GROUPING

If the safety circles intersect each other, a possible solution is to collect the obstacles into a larger set. Let us suppose that the safety circle of the obstacle at highest priority intersects those relative to other obstacles as shown in Figure 2. A possible strategy to guide the vehicle outside the entire set of obstacles is to keep the process relative to the highest priority obstacle unchanged, and to vary the others. This is achieved very simply, by inverting the vector computed by the obstacle avoidance algorithm relative to all the obstacles, except

for the critical one (the one nearest to the vehicle). Let us examine in detail the modification to the NSBBC algorithm for this particular case. Define the obstacles area

$$o_i\left(p_{o_i}\right)=\left\{w\in\mathbb{R}^2 \mid \left\|w-p_{o_i}\right\|\le d\right\} \quad \textbf{(11)}$$

where d is the safety radius. Define the obstacles-set

$$\Omega=\left\{o_i, i=1..N\right\} \quad \textbf{(12)}$$

where N is the number of obstacles. Define the primary obstacle as

$$o_o=\left\{o_i\in\Omega \mid \left\|p-p_{o_i}\right\|=\min_{k=1..N}\left(\left\|p-p_{o_k}\right\|\right)\right\} \quad \textbf{(13)}$$

The obstacles are divided into groups. Each group contains only obstacles whose relative distance is smaller than the safety circle's radius. If an obstacle has a non intersecting safety circle, then it belongs to a group consisting of itself only. The Cluster-set C is:

$$C=o_o\cup\left\{o_i\in\Omega \mid \exists S\left(o_i\right)\right\} \quad \textbf{(14)}$$

where the sequence $S\left(o_i\right)$ is define as follow:

$$S\left(o_i\right)=\left\{o^j\in\Omega, j=1..k \mid o^k\equiv o_i, o^1\equiv o_o, \left(o^{n+1}\cap o^n\right)_{n<k}\neq\varnothing\right\} \quad \textbf{(15)}$$

During the construction of the velocity vector, if an obstacle is identified as belonging to the critical group, then there is a sign change variation in the weight Λ_o. Although this does not guarantee mathematically the entrance in a safety circle, it greatly reduces the probability. Define the function

$$\Phi\left(o_i\right)=\begin{cases}1, o_i\in C\\ 0, otherwise\end{cases} \quad \textbf{(16)}$$

The weight of the obstacle o_i can be calculate using the following expression

$$\Lambda_{o_i}=\begin{cases}-\Lambda_o, \Phi\left(o_i\right)=1, o_i\neq o_o\\ \Lambda_o, otherwise\end{cases} \quad \textbf{(17)}$$

where Λ_o is the weight associate to the generic obstacle avoidance task. The modification introduced here is applied to the scenario of Figure 2, and the results are shown on Figure 3. The grouping concept improves the obstacle avoidance, but it can fail when, for instance, the starting point of the vehicle is well within the concave area described by the group. In this case, safety circles may be violated by the computed trajectory. An example is shown in Figure 4.

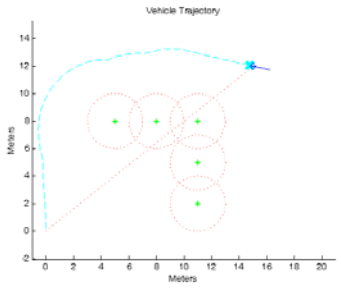

Fig.3: Obstacle Avoidance Trajectory with modified NSBBC Algorithm for Scenario of Figure 3.

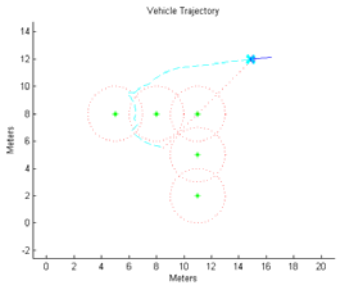

Fig.4: Example of possible Drawback of the Grouping Algorithm

A second potential issue in the grouping algorithm is the effect of variations in the weight Λ_o. When computing a trajectory that goes outside a group of obstacles, it is appropriate to generate a path that is located as close as possible to the obstacles. This is done by increasing Λ_o. In this case, however, there is the possibility that the vehicle reach a point where the resulting velocity vector is zero, and the vehicle can not reach the target. This singularity arises for the following conditions:

1. The vehicle is on the safety circle.
2. The resultant of the target task vector and the vectors relative to all obstacles is parallel to the vector needed to avoid the obstacle having highest priority.

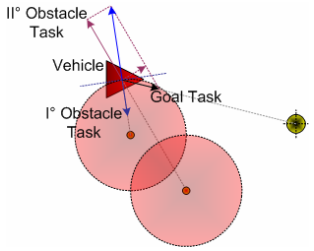

Fig.5: Potential Singularity in the Grouping Algorithm.

A graphical example is shown in Figure 5. If the vehicle were to move from the current position, a velocity vector component would be created, tangent to the circle,

making the vehicle return to the initial point. If the weight associated to secondary obstacles is sufficiently high, the situation may arise before the obstacle is declassified (being farther than the target), and the vehicle stops. By increasing Λ_o the vehicle stops earlier, whereas by increasing Λ_g the vehicle stops closer to the target.

VI. Shadow Avoiding Device

A task called Shadow Avoiding Device (SAD) was proposed and developed to improve the problems encountered by the grouping algorithm described in the previous section. SAD is based on human experience when dealing with his/her motion towards a light source. If a person is located in the shadow cone of an object, and wants to reach the light source, then his/her response is to move toward the lighted zone closest to the obstacle. The target projects Omni directional light beams, and creates shadow cones behind obstacles (or groups of obstacles). If the vehicle is in the shadow of an obstacle in the group, it will try to exit moving in the direction of the light source. The shadow cone built with lines tangent to the safety circle and starting from the target (for each obstacle). At the end, a shadow cone for the entire group is defined as the envelope of the single cones. The group border is identified much in the same way with lines departing from the vehicle's current position. The two points generated by intersecting the cones are the light points towards which the vehicle will move in order to avoid the obstacles.

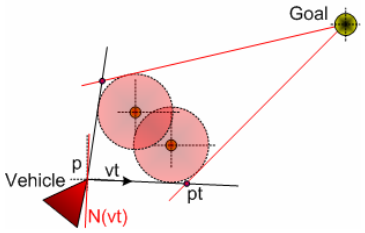

Fig.6: Implementation of S.A.D.

The choice of which point depends on different factors, for instance we can choose the one that minimizes the distance between the vehicle and the target. To reach the desired point, the task needs full control over the vehicle's motion. The Jacobian matrix is therefore the identity matrix, like that of the target, so that all the subtasks with lower priority will participate in the construction of the SAD velocity vector.

The control variable σ_{SAD} is the vehicle's position and the desired position ($\bar{\sigma}_{SAD}$) is the position of the chosen light point. We have therefore:

$$J_{SAD} = I; \sigma_{SAD} = p; \bar{\sigma}_{SAD} = p_t \qquad (18)$$

Once the obstacle avoidance scheme is set up (with the above improvements), the supervising algorithm needs to be constructed with the priority rules for implementation. Heuristic reasoning can be used for this purpose. If the vehicle is outside the shadow of a group of obstacles, then the SAD priority is the lowest, and it does not influence the motion of the vehicle. If the vehicle enters a shadow cone, then SAD has a higher priority. In this case, if the distance between vehicle and shadow cone exit point is larger than the distance between primary obstacle and the same point, there is a collision danger. The highest priority is then given to the avoidance of the primary obstacle (SAD will assume priority level 2). When the vehicle is inside the shadow zone, only the closest obstacle is considered. The light point is visible by construction (see Figure 6), and the vehicle will not cross other obstacles in order to reach it.

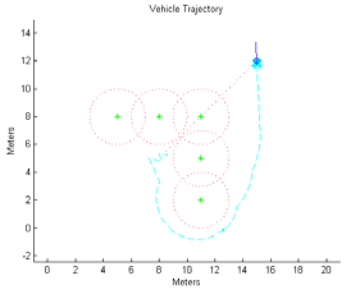

Fig.7: Collision Avoidance using N.S.B.B.C. with S.A.D.

VII. Modified NSBBC in a Simulated Scenario

The algorithm described in the previous section was adapted for implementation on the Ulisse vehicle [5]. The vehicle has two cameras, which are used to reconstruct 3D information of the surrounding environment. Pairs of frames and pairs of congruent points are periodically processed and features extracted using the SIFT algorithm and grouped to form solid obstacles. The details of the vision system are described in [9]. At this point, the obstacle avoidance algorithm is introduced. Since the obstacles are not point mass models as required in [3], we can either consider a sufficiently large safety perimeter to incorporate the entire obstacle, or create a safety perimeter made of smaller circles of constant radius. The latter choice was selected in this work. A data-base must also be included in the overall process capable or memorizing obstacles as they appear on the sensors. For computational purposes, only obstacles at a distance larger than ¼ the safety circle of an obstacle already in memory are included. This guarantees that the algorithm is capable of grouping, and the memory is not

overloaded.

The ordered list of obstacles is then passed to the obstacle avoidance algorithm, in order to build the appropriate velocity vector sequence to be commanded to the vehicle.

VIII. TESTS

Currently, the proposed algorithm has been implemented using a numerical simulation under MATLAB environment.

Consider a triangular obstacle between vehicle and target. Figure 8 shows the trajectories using standard NSBBC collision avoidance (green path), and the proposed method with grouping and SAD (blu path).

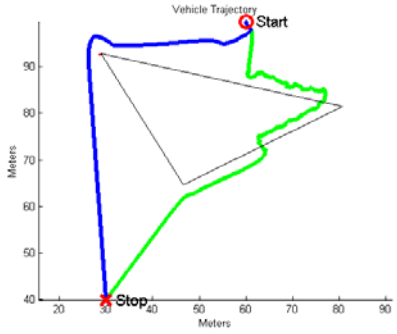

Fig.8: Simple triangular Obstacle

Figure 9 show a simulation representing a vehicle going through two obstacles that create a "canyon like" structure. Again the improvement in the results is evident.

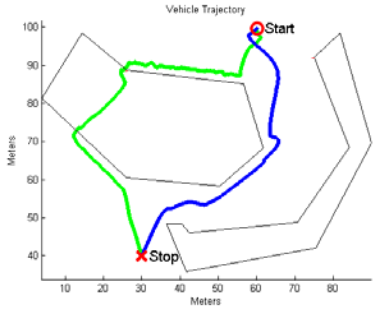

Fig.9: Trajectory through a Canyon

The last simulation considers a starting point within the concavity of an obstacle. The two different solutions are shown in Figure 10, where the standard NSBBC method fails, unlike the proposed method.

IX. CONCLUSIONS

The present work proposes a modification of an obstacle avoidance algorithm recently proposed in the literature. The use of the algorithm, with the addition of obstacle grouping, and shadow cones avoids collisions in critical cases, as demonstrated via numerical simulation. At the current time, field testing is in progress using the autonomous vehicle Ulisse, developed by the Automation group of the University of Pisa.

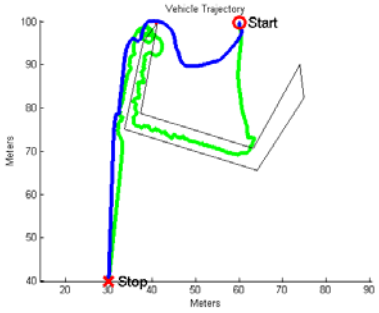

Fig.10: Trajectory Comparison in the presence of Concave Obstacles

REFERENCES

1. Jin-Oh Kim, Pradeep K. Khosla Real-Time Obstacle Avoidance Using Harmonic Potential Functions *IEEE Transaction on robotics and automation*, Vol .8 No.3, June 1992.
2. Siripun Thongchai, Kazuhiko Kawamura Application of Fuzzy Control to a Sonar-Based Obstacle Avoidance Mobile Robot *International Conference on Control Applications* Anchorage, Alaska, USA September 2000.
3. A. Chohr;, A. Farah**, and M. Belloucif Neuro-Fuzzy Expert System E-S-CO- V for the Obstacle Avoidance of Intelligent Autonomous Vehicles (IA V).
4. Priti K. Gaonkar, Anthony DelSorbo and Kuldip S. Rattan An Intelligent Robot System using Fuzzy Logic, 2005.
5. M. Cellini, R. Mati, L. Pollini, M. Innocenti Software and Hardware for Guidance and Navigation of unmanned Vehicles. *Aeronautics and Space Education Workshop*, Istanbul, June 2006.
6. S.Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*. 13(3):398-410, 1997.
7. G. Antonelli F. Arrichiello S. Chiaverini The Null-Space Based Behavioral Control for Mobile Robots *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Espoo, Finland, June 2005.
8. G. Antonelli F. Arrichiello S. Chiaverini Experiments of Formation Control with Collision Avoidance using the Null Space Based Behavioral Control *IEEE Mediterranean Conference on Control and Automation*, Ancona, Italy, June 2005.
9. L. Pollini , F. Greco , R. Mati and M. Innocenti, A. Tortelli, "Stereo Vision Obstacle Detection based on Scale Invariant Feature Transform Algorithm" *accepted for pubblication in Proceedings of AIAA Guidance, Navigation, and Control Conference*, Hilton Head, South Carolina, 20-23 Aug 2007