Reactive Obstacle Avoidance and Control Action Continuity. Application to the ND Algorithm

Conference Paper · September 2008

CITATIONS
0

READS
39

4 authors:

Daniel Hernández-Sosa
Universidad de Las Palmas de Gran Canaria
92 PUBLICATIONS **645** CITATIONS
SEE PROFILE

Jorge Cabrera
Universidad de Las Palmas de Gran Canaria
77 PUBLICATIONS **383** CITATIONS
SEE PROFILE

Antonio Carlos Domínguez Brito
Universidad de Las Palmas de Gran Canaria
63 PUBLICATIONS **338** CITATIONS
SEE PROFILE

Josep Isern
Universidad de Las Palmas de Gran Canaria
52 PUBLICATIONS **274** CITATIONS
SEE PROFILE

Reactive Obstacle Avoidance and Control Action Continuity. Application to the ND Algorithm

Daniel Hernandez, Jorge Cabrera, Antonio Dominguez and Josep Isern

***Abstract*—Endowing a physical agent with the capability of moving in a safe and agile manner is an issue of capital importance during the development of these systems. In this context, our work is related to obstacle avoidance in general, and specifically to ND algorithms. Contrary to many previous methods, the ND approach is not aimed at devising a general motion law, but operates over a reduced set of possible situations that are treated by its particular motion law. The big earning of this idea is that it eases the design of control, as now motion laws are specific to every identifiable situation. However, it also raises new issues as nothing guarantees the smoothness of motion commands when the diagnostic changes. Extensive experimentation with an implementation of this method has revealed that often negligible changes in the perceived environment induce changes in the identification of the situation that have as a consequence large modifications of commanded velocities. These unnecessary and drastic changes in the commanded velocities seriously degrade the coherence and agility of the agent movement. In this paper a modification of the ND approach is presented, in order to improve this aspect of the method, along with experimental results.**

***Index Terms*—Autonomous Robots, Obstacle Avoidance.**

I. Introduction

ENDOWING a physical agent with the capability of moving in a safe and agile manner is one of the first problems that must be addressed during the development of these systems. The study and design of obstacle avoidance algorithms has been in the research agenda of researchers in the mobile robots field for nearly thirty years and very different approaches have been presented.

Of special interest are those methods that are suitable for operation in an unknown and uncertain world, a scenario where classical geometrical path planning methods, which rely on a priori availability of a map of the environment, are not applicable. Many of the prevalent and better known of these methods can be classified as reactive sensor-based path planning methods and have been extensively tested in indoor scenarios. All these methods divide their execution into two stages. First, a local map is used to integrate sensor range data along time. In a second stage, a local path is planned through free space towards the goal. The differences among methods are centred in how this local path is obtained from the local map.

Some obstacle avoidance methods employ the local map to create an artificial potential field [15][7] that should drive the robot towards the goal. Algorithms based on potential fields show up important instabilities and are not suitable for navigating narrow passages. Borenstein and Koren [2] developed a different approach, reducing the local path planning to the search of open passages in a polar histogram that is built from the local map. This method has been improved to take into consideration the robot kinematic constraints [16] and a global path planner [17].

Other approaches, like the curvature velocity method [14], the beam curvature method [4] or the dynamic window based methods [5] [3] [1] have been proposed to take into consideration the robots kinematics and dynamic constraints in order to guarantee that a planned local trajectory is physically feasible, a question commonly neglected in previous methods.

An aspect that all these methods have in common is that the different situations that an agent must face while navigating in its environment are finally treated using a single motion law, where factors as the width of the selected open passage, its angular position relative to the actual direction of motion or the relative position of the goal, are weighted in a control law to derive the commanded linear and angular velocities.

A different approach, termed Nearness Diagram (ND) Navigation, is presented by Minguez et al in [12] [6]. This method uses a "divide and conquer" strategy to classify the local navigation problem into *situations*. As we will show in section II, different situations can be recognized, using a simple decision tree, depending on factors such as the width of the open space in front of the robot or valley, if the goal point is within that valley, etc. Contrary to many previous methods, in the ND method it is not intended to devise a general motion law, but to treat every situation using a particular motion law. The big earning of this approach is that it eases the design of motion laws, as these are now specific to every possible situation. However, this approach also raises new issues as nothing guarantees the smoothness of motion commands when the situation changes.

The ND method and its descendants have been designed to address many of the common pitfalls exhibited by many previous reactive obstacle avoidance algorithms, like U-shaped local traps, oscillatory motions, excessive dependence on relative position of the goal, or feasibility of planned trajectories due to the kinematic or dynamic constraints of the robot or its shape. To be capable of dealing with these problems, the most recent proposals for obstacle avoidance have evolved from simple modules towards complex architectures that include components for local map building and maintenance, path planning and reactive components that take into consideration the shape of the robot or its kinematic and dynamic constraints [8] [13].

Instituto de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI) - ULPGC.
E-mail: dhernandez@iusiani.ulpgc.es

Extensive experimentation with the ND method has demonstrated that it is capable of addressing many of the aforementioned problems, but has revealed also some deficiencies related to the identification of situations and its temporal coherence. Concretely, nothing in the ND approach precludes rapid changes in the identification of the current situation. Situations are recognized using crisp sensor-dependant decisions and, besides, this process is also not orientated, in the sense that the system is not conscious of the situation that it was negotiating in the previous cycle. Although a high degree of reactivity is desirable, as unanticipated situations may show up unexpectedly, some mechanism should be implemented in order to deal with command discontinuities. Experiments have shown that often negligible changes in the perceived environment induce changes in the identification of the situation that have as a consequence large modifications of commanded velocities. These unnecessary and drastic changes in the commanded actions seriously degrade the coherence and agility of the agent movement. In this paper a modification of the ND approach will be proposed and evaluated to improve this aspect of the method.

II. The Nearness Diagram (ND) Algorithm

In the following section we will briefly introduce the ND (Nearness Diagram) algorithm for avoiding obstacles [11], specifically its ND+ version [10]. The ND is a reactive algorithm that uses, as input information, sensory data provided by a laser range finder which produces periodically a data scan. Based on this sensory information the algorithm decides which action should be carried out by the robot in order to reach a goal position avoiding collision with obstacles.

The algorithm operation is divided in three sequential phases:

1) **Calculation of ND Diagrams**. In the first phase two polar diagrams are generated, PND and RND, from sensory data. The first of these diagrams, PND, represents the nearness of the obstacles from the central position of the robot, the second one, RND, represents the nearness of the obstacles from the robots bounds. Details about how to calculate both diagrams can be found in [11].
2) **Selection of a Navigable Valley**. Using the PND diagram obtained in the previous phase, the algorithm looks for discontinuities in the diagram. A discontinuity is defined as the difference between two polar values such that it is wide enough to allow the robot to pass through the obstacle points given by the polar values. Thus, for instance, for a circular robot the difference should be greater than the robot diameter. A valley is formed between two "adjacent discontinuities" found in the diagram. The valley which is closer to the goal in angular terms will be selected first. Next a *navigability test* is applied on it, if the test is positive, the valley is selected, if not the algorithm proceeds equally with the next valley available, until it founds one navigable valley. Details about how to calculate the diagram discontinuities, the valleys, and the navigability test can be found also in [11].
3) **Situated Actions**. Based on the navigable valley selected in the previous phase and the RND diagram calculated in the first phase the algorithm diagnoses six different situations in ND+ [10] (five situations in ND [11]) in order to generate an action aimed at driving the robot to the goal while avoiding obstacles. In turn, those situated actions are classified into two groups.
   - **High Safety (HS) Situations**. When there are no obstacles near the robot closer than a given distance, called *security distance*, which defines a *security zone* surrounding the robot, it is said that the robot is in *High Safety*. The algorithm considers three types of high safety situations.
     - *HSWR*. The robot is in high safety and the navigable valley is *wide*.
     - *HSNR*. The robot is in high safety and the navigable valley is *narrow*. To distinguish between *wide* and *narrow* valleys an angular width threshold is used. If valley width is greater than the threshold the valley is considered *wide*, otherwise it is considered *narrow*. A typical value for this angular width threshold is $\frac{\pi}{2}$ [11].
     - *HSGR*. The robot is in high safety and the goal is inside the navigable valley.
   - **Low Safety (LS) Situations**. When there are obstacles inside the security zone, it is said that the robot is in *low safety* (the RND diagram is used to know if any obstacle is in the security zone). Three situations are considered in low safety situations:
     - *LS1*. There are obstacles only on one side of the robot respect to the discontinuity closer to the goal of the navigable valley.
     - *LS2*. There are obstacles on both sides of the robot respect to the discontinuity closer to the goal of the navigable valley.
     - *LSGR*. There are obstacles in the security zone and the goal is inside the navigable valley.

   Based on the previous taxonomy of situations the algorithm determines which actions should be taken in order to avoid obstacles, if any, at the same time that the robot is driven to the goal. The corresponding decision tree is presented in Fig. 1, extracted from the original paper, for clarification purposes. Thus, for high safety situations in HSWR the robot is driven to the side of the valley closer to the goal, in HSNR the robot is commanded to the central part of the valley, in HSGR the robot goes directly to the goal position. As to low safety situations, in LS1 situations the algorithm try to drive the robot a bit away from the side with obstacles without deviating too much from the goal direction, in LS2 the robot is driven centred between the closer obstacles at both sides of the robot which are inside the security zone. Finally, in LSGR the robot is driven to the goal with a deviation depending on how close the obstacles are inside the security zone. For details consult [10].

Additionally the ND algorithm can be used with a grid map where laser scans are registered in order to combine the

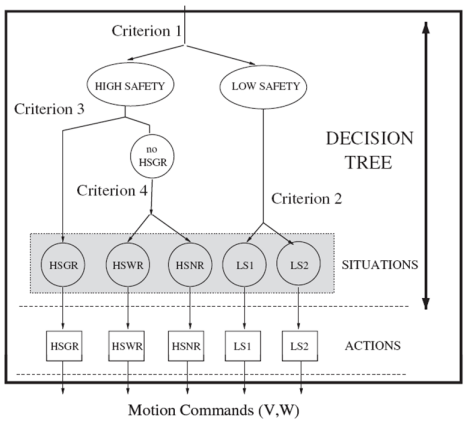

Fig. 1. The ND decision tree for situation identification

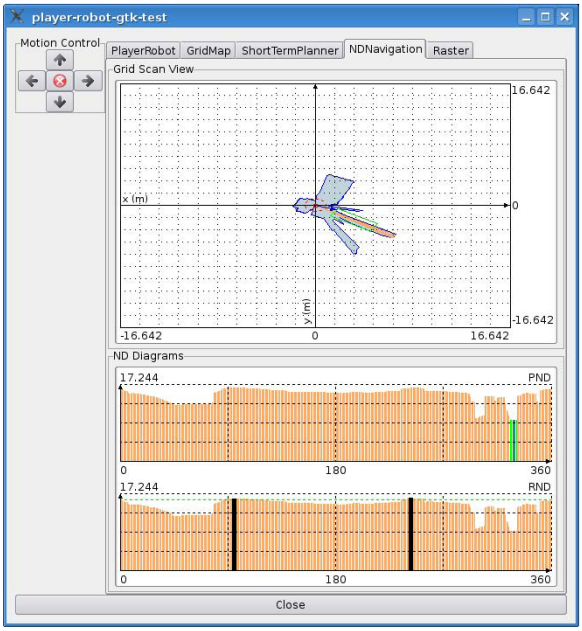

Fig. 2. The ND+ version of the ND algorithm in execution

algorithm with a short term planner to avoid the typical trap situations of obstacle avoidance reactive algorithms [9].

In Fig. 2 we can see a snapshot of our implementation of the ND+ version of the ND algorithm. In order to debug and interpret the behaviour of the system it is possible to observe the ND diagrams, the navigable valley, the closer obstacles, etc. during execution. In the figure the robot is in a low safety situation, concretely LSGR, since the goal is inside the navigable valley (highlighted in the PND diagram, around 300 deg. of Fig. 2), and there are obstacles inside the security region at both sides of the navigable valley (the closer ones are indicated by the thick lines in the RND diagram shown in the figure).

Moreover, in the same snapshot, a grid view shows the robot and its surroundings given by the sensory data. On it, we can see the direction the algorithm has decided to chose for each situation in order to avoid obstacles (denoted by the big arrow centered in the origin of the grid). Thus, during execution and using the ND view of the visual front-end of the system we can observe which decisions the algorithm is taken in order to avoid obstacles while navigating to the final goal.

Fig. 3 shows the trajectory followed by the robot driven by the algorithm in an execution where it was commanded to go to a specific location. The figure shows the grid map build during the execution, where black areas denote free space and grey areas indicate unknown space (whether occupied or not), and the white areas represent obstacles. The trajectory followed by the robot has been superimposed on the figure, where the 'x' denotes the initial position, and the circle is the robot itself at the goal destination.

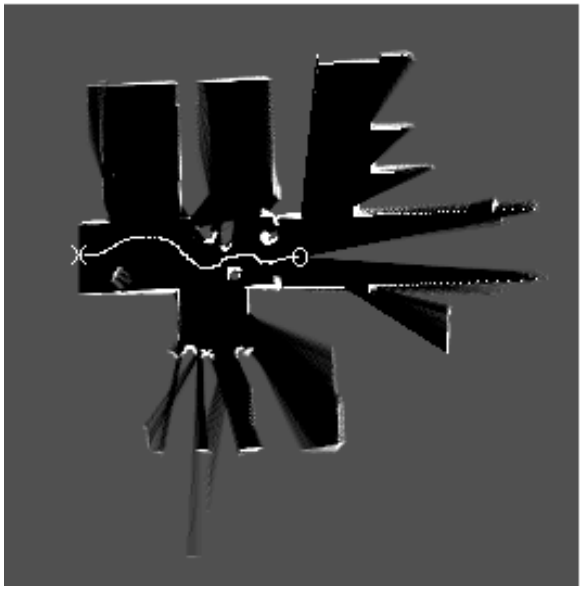

Fig. 3. ND+: Trajectory followed to get to an specific location

III. Proposed Alternative

As commented in the introduction, the ND+ algorithm results that we have obtained both from Player/Stage simulations and real world experiments show frequently sharp oscillations in the commanded and measured angular velocities, especially when medium to high maximum limits are allowed (using low limits discontinuities can be damped/masked by robot dynamics). These oscillations are also observed in the commanded translational speed, due to the fact that both command signals are coupled. In most cases, these abrupt changes were not motivated by the selection of a different valley from the ND diagram, they are consequence of some transitions between different situations.

This result is not completely unexpected, as there is not any explicit mechanism in the control strategy that can guarantee continuity or smoothness between successive control actions generated under the aforementioned circumstances. In fact, ND+ algorithm is an extension of a previous version that tries to address control action continuity as one of their objectives. However, this analysis was only developed for a subset of possible situation transitions, the most frequent in the kind of "narrow navigation" problems considered, or, as termed by the authors "troublesome scenarios". In other related work [10],

authors make a reference to this issue indicating that they have used a hysteresis mechanism to reduce the discontinuity effect, although no additional details are given.

The origin of the problem can be located in the decision tree used to label the current robot situation. Independently of the implementation, a border line is created so slight sensor data fluctuations could lead to different situation selection. Under this point of view, the control actions are also highly sensitive to parameterization, as on the same sensor reading, a small change in one parameter can induce significant changes on the robot behaviour. In our opinion, this problem is not easy, if not impossible, to solve globally. So, direct filtering mechanisms cannot be applied without compromising robot reactivity or even safety. On the other hand, imposing limitations on valley selection could be another alternative, but special care should be taken on trying to preserve robot opportunity to select new promising valleys toward the goal.

We consider, however, that the problem can be alleviated if a kind of membership function would be taken into account when a new command is to be produced from the estimated robot situation. The approximation proposed in this paper consists in defining a measure of situation certainty in order to determine how trustful the action command generated by the correspondent control rule is. Once computed, this certainty degree is used to calculate the final commanded angular velocity for the robot weighting the value proposed from the situated action and the current angular velocity.

A. Certainty Estimation

According to the decision tree used for determining the current situation the following factors are considered:

- HS_F: High/low security distance factor.
- GC_F: Goal contained in valley factor.
- NV_F: Narrow/wide valley factor.

In a first step, the modified algorithm maps the factors involved in the selection of the current situation to a 0-1 range. For the security distance factor HS_F an additional distance (Δ_{HS}) is used to reduce/extend the established security threshold (Thr_{HS}) and determine when the situation is clearly low safety, with obstacles (OD) detected inside the reduced security region; or clearly high safety, with no obstacles detected inside the extended security region. An obstacle detected on the border of the security region is translated into a 0.5 mapped factor value.

The goal contained factor GC_F evaluates the angular distance between the goal and the valley centre (GD). The valley aperture (VA) is modified in a quantity (Δ_{GC}) to map the goal distance to a 0-1 range. A goal detected on the border of the region results in a 0.5 mapped factor value.

The narrow valley factor NV_F is also mapped using a comparison between the current valley length and the reference value (Thr_{NV}) extended/reduced by a band (Δ_{NV}) defined as a percentage. A valley equal in length to the reference value gives a 0.5 mapped factor value.

The formulas used for this 0-1 mapping are the following ones (limit conditions not showed for simplicity):

$$\begin{aligned} HS\_F &= 1 - (Thr_{HS} + \Delta_{HS} - OD)/(2 * \Delta_{NV}) \\ GC\_F &= 1 - (VA/2 + \Delta_{GC} - GD)/(2 * \Delta_{GC}) \\ NV\_F &= (Thr_{NV} + \Delta_{NV} - VA)/(2 * \Delta_{NV}) \end{aligned} \quad (1)$$

Once the factors have been mapped, the situation certainty can be obtained using some evidence fusion method. In our tests, simple minimum or multiplicative rules, following the algorithm decision tree, have produced adequate results. Specifically, the multiplicative rule has been applied in the experiments presented in this paper.

B. Performance Evaluation

Although visual estimation is normally sufficient for preliminary analysis of the results, a simple benchmark measure has been defined and used in this work for a more objective quantitative comparison. This measure contains the following factors: time, curvature, translational acceleration and rotational acceleration. The ideal trajectory is considered to be a straight line joining the initial and goal points, following a trapezoidal velocity profile, using maximum acceleration and velocities. This ideal case will return a value of 1.0 for each factor.

The time factor ($TimeF$) is computed dividing the minimum expected execution time, considering initial and final points and maximum acceleration and velocities, by the robot elapsed time to reach the goal.

The curvature factor ($CurvF$) is calculated as the quotient between mean curvature ratio of the robot trajectory and a value of maximum curvature ratio. The curvature ratio estimation is saturated to that max value to avoid the division by zero condition.

The translational acceleration factor ($TAccelF$) is computed the quotient between the mean acceleration of the ideal robot trajectory from origin to destination and the mean translational acceleration for the robot real trajectory.

Finally, the rotational acceleration factor ($RAccelF$) is computed from the mean of the measured rotational acceleration ($MeanRotAccel$) using the formula $1/(1+MeanRotAccel)$.

IV. Experiments

Some experiments have been conducted in order to evaluate the effect of the proposed solution on the robot behaviour. Two versions of ND+, with and without situation certainty evaluation, have been implemented on the Player/Stage environment to compare their performance.

The first set of experiments use a simplified obstacle configuration to facilitate the illustration of the results. In the final part a more complex execution is presented.

A. Scenario 1

Figure 4 shows the obstacle configuration used for the first experiment, including the robot initial position and its trajectory to the final (labeled with time stamps) position using the standard version of the ND+ algorithm. The plot in figure 5 depicts the commanded angular velocities and the situation transitions along the robot path from the origin to

the goal. Figure 6 illustrates the situation transitions in more detail. It is possible to verify how certain situation transition have practically no effect on command continuity, for example around the 25-30 seconds interval; while others induce a more noticeable distortion, for example between 30 and 35 seconds.

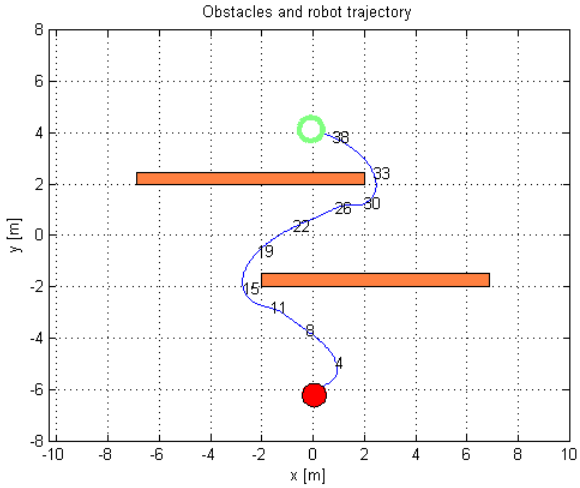

Fig. 4. Scenario 1: setup and trajectory using the standard algorithm

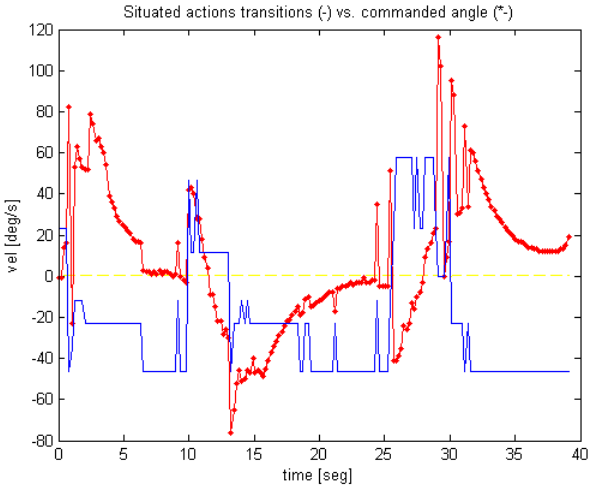

Fig. 5. Situation transitions and commands from standard algorithm (scenario 1)

Finally, figures 7 and 8 include the robot velocities and accelerations corresponding to the execution of the standard ND+ algorithm. Here, the discontinuities observed in the commanded angular velocity graph finally show up as high robot acceleration values.

The same figures are now repeated for the modified version of the algorithm: trajectory in figure 9, commands and situation transitions in figure 10, velocities in figure 11 and accelerations in figure 12. The results show how the smoothed commands generated by the certainty-based version are responsible for the lower acceleration values measured, without affecting significantly trajectory safety or elapsed time from robot start until the goal is reached. The visual impression of

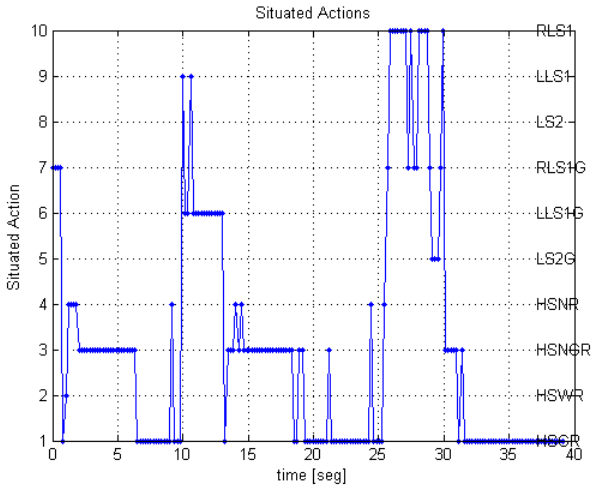

Fig. 6. Detailed situation transitions (standard algorithm)

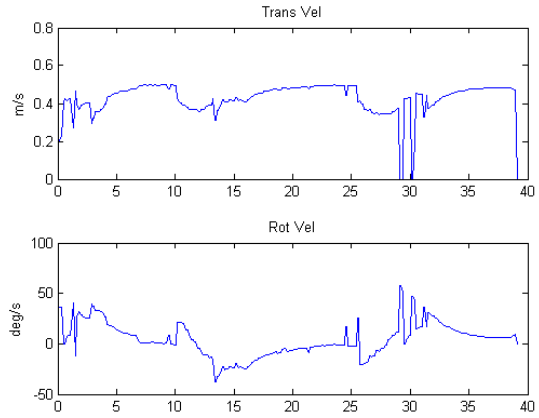

Fig. 7. Robot velocities from standard algorithm (scenario 1)

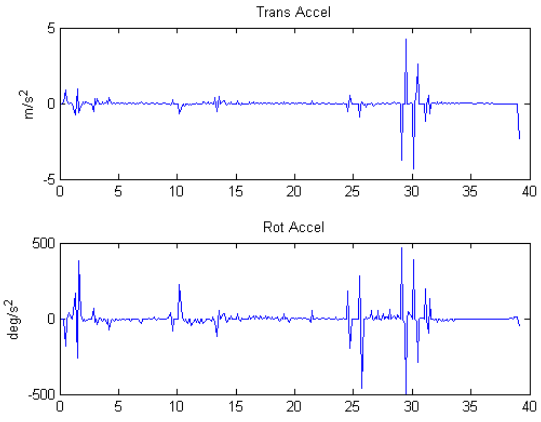

Fig. 8. Robot accelerations from standard algorithm (scenario 1)

improvement is corroborated by the benchmark measure, that gives better values for the modified version of the algorithm. The table I resumes the benchmarking factor values computed for this comparison.

As can be observed from the results, some discontinuities, although attenuated, are still present. These occurrences are

	$TimeF$	$CurvF$	$TAccelF$	$RAccelF$
Standard	0.52	0.39	0.18	0.72
Modified	0.52	0.41	0.47	0.85

TABLE I
BENCHMARKING FOR SCENARIO 1

associated with new transitions where the identified situation has a high certainty level, and must be preserved in order to avoid robot collisions or missed targets.

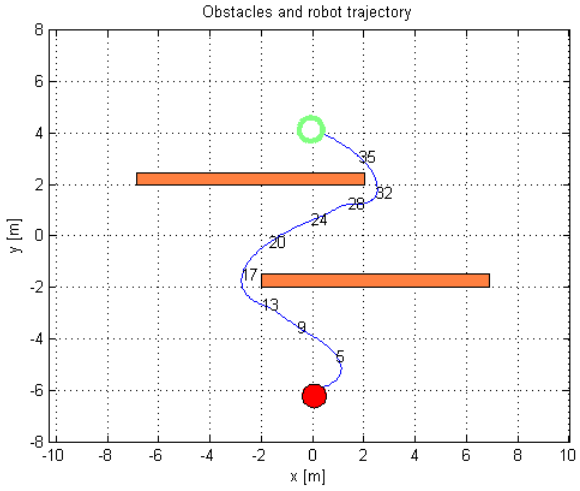

Fig. 9. Scenario 1: setup and trajectory using the modified algorithm

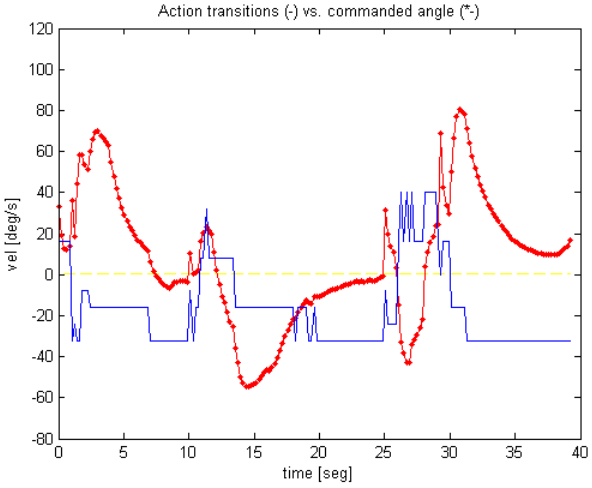

Fig. 10. Situation transitions and commands from modified algorithm (scenario 1)

B. Scenario 2

Using a more complicated scenario, similar results have been obtained. The figures 13 to 15 resume the collected data for this second case. The table II resumes the benchmarking factor values computed for this test.

C. Scenario 3

The final experiment presented in this paper has been obtained from one of the maps distributed with Player/Stage,

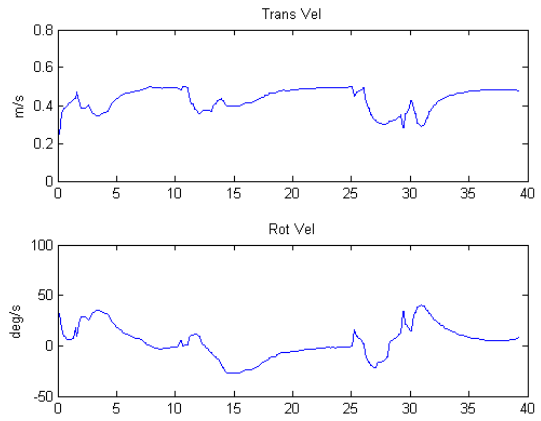

Fig. 11. Robot velocities from modified algorithm (scenario 1)

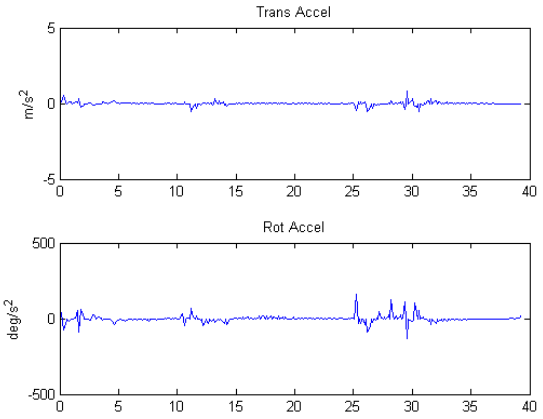

Fig. 12. Robot accelerations from modified algorithm (scenario 1)

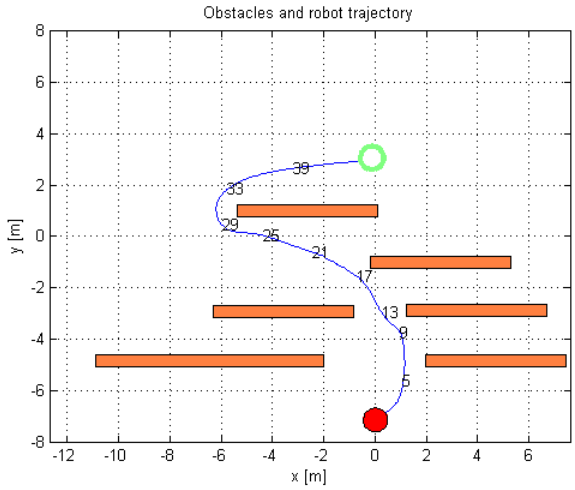

Fig. 13. Scenario 2: setup and trajectory using the standard algorithm

namely the hospital. In this case, a set of destination points have been introduced sequentially for both algorithms. This is needed as we are using these algorithms without a path planner. Figure 16 shows the map used for this experiment, including the robot initial position and its trajectory, labeled with timestamps, from the initial point (triangle) to the dif-

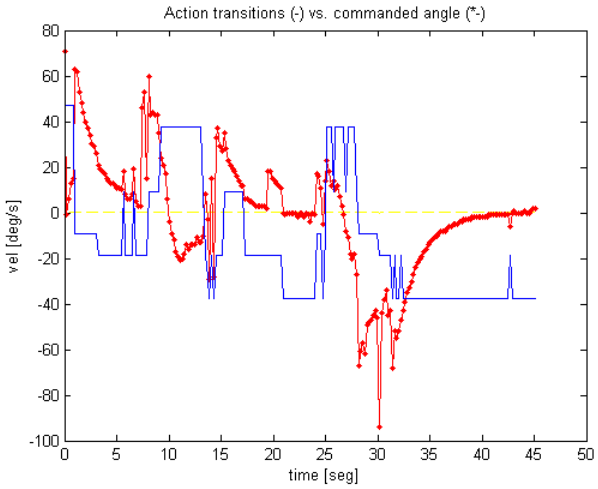

Fig. 14. Situation transitions and commands from standard algorithm (scenario 2)

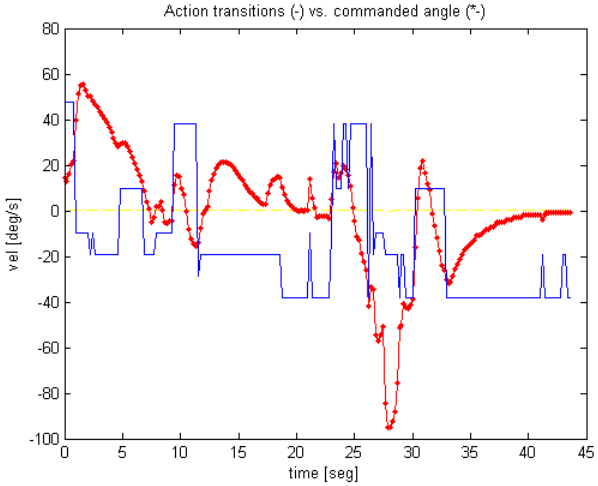

Fig. 15. Situation transitions and commands from modified algorithm (scenario 2)

ferent goals (circles) using the modified version of the ND+ algorithm. The resultant commands and situation transitions are presented in figures 17 and 18. Here again, the modified algorithm is assigned a better value by the benchmarking than for the standard version (0.21 and 0.16 mean values respectively), although in this case the measure is affected by the interaction with the simulation for providing the successive goals.

V. Conclusion

In this paper a certainty-based method has been applied to reactive obstacle avoidance in order to improve control action continuity. The ND algorithm has been analyzed under this perspective, concluding that control action smoothness cannot be guaranteed under high velocities and rapid situation changes. The experiments have demonstrated that the modified algorithm presented in this paper shows a better stability without compromising reactivity. A benchmarking function has been defined to provide and objective measure for comparison.

	$TimeF$	$CurvF$	$TAccelF$	$RAccelF$
Standard	0.45	0.50	0.24	0.80
Modified	0.46	0.54	0.32	0.88

TABLE II
Benchmarking for scenario 2

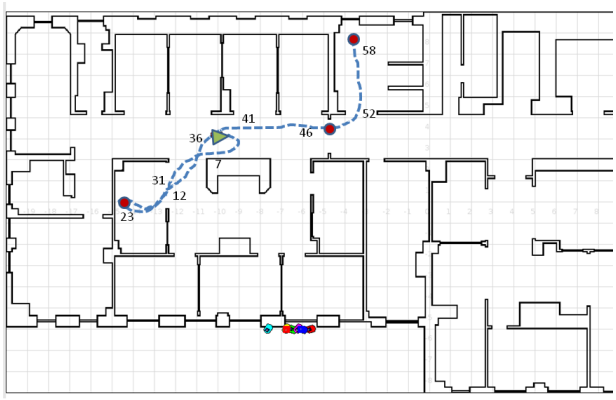

Fig. 16. Scenario 3: setup and trajectory using the standard algorithm

References

[1] Kai Oliver Arras, Jan Persson, Nicola Tomatis, and Roland Siegwart. Real-time obstacle avoidance for polygonal robots with a reduced dynamic window. In *Proc. Int. Conf. on Robotics and Automation*, pages 3050–3055, 2002.

[2] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, April 1991.

[3] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proc. Int. Conf. on Robotics and Automation*, pages 341–346, 1999.

[4] J.L. Fernandez, R. Sanz, J. A. Benayas, and pp . Dieguez, A. R. . Improving collision avoidance for mobile robots in partially known environments: the beam curvature method. . Elsevier Science B.V. 46-4. April 2004. Nearness diagram (ND) navigation: collision avoidance in

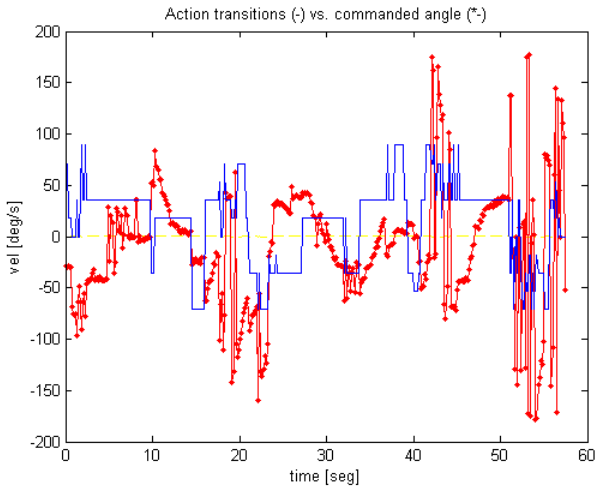

Fig. 17. Situation transitions and commands from standard algorithm (scenario 3)

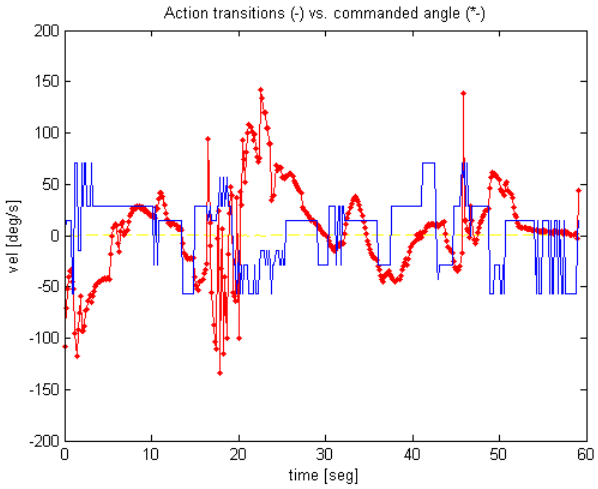

Fig. 18. Situation transitions and commands from modified algorithm (scenario 3)

troublesome scenarios. *Robotics and Autonomous Systems*, 46(4):205–219, Ap. 2004.

[5] Dieter Fox, W. Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation*, 4(1):23–33, March 1997.

[6] L. Montano J. Minguez, J. Osuna. A divide and conquer strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[7] M. Khatib and R. Chatila. An extended potential field approach for mobile robot sensor-based motions. In *Proc. of International Conference on Intelligent Autonomous Systems (IAS'95)*, page 490496, 1995.

[8] J. Minguez L. Montesano and L. Montano. Extending reactive collision avoidance methods to consider any vehicle shape and the kinematics and dynamic constraints. *International Journal of Advanced Robotic Systems*, 3(1):85–91, 2006.

[9] J. Minguez, L. Montano, N. Simeon, and R. Alami. Global nearness diagram navigation (GND). *IEEE Int. Conf. on Robotics and Automation, Seoul, Korea*, pages 33–39, 2001.

[10] J. Minguez, J. Osuna, and L. Montano. A "divide and conquer" strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. *Proceedings. ICRA '04. 2004 IEEE International Conference on Robotics and Automation, 2004.*, 4:3855–3862 Vol.4, April 26-May 1, 2004.

[11] Javier Minguez and L. Montano. Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, Feb. 2004.

[12] Javier Minguez and Luis Montano. Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, Febrery 2004.

[13] Javier Minguez and Luis Montano. Extending reactive collision avoidance methods to consider any vehicle shape and the kinematics and dynamic constraints. *IEEE Transactions on Robotics*, 2008.

[14] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *International Conference on Robotics and Automation*, pages 3375–3382, April 1996.

[15] R.B. Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Proc. Int. Conf. on Robotics and Automation*, volume 2, pages 566–571, 1990.

[16] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1572–1577, 1998.

[17] I. Ulrich and J. Borenstein. VFH*: Local obstacle avoidance with look-ahead verification. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2505–2511, April 2000.