The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Safe Obstacle Avoidance for Industrial Robot Working Without Fences

N. Pedrocchi, M. Malosio, L. Molinari Tosatti

Abstract— **Until now, the presence of fences is a technological barrier for the adoption of robots in Small Medium Enterprises (SME). The work deals with the definition of an intrinsically safe algorithm to avoid collisions between an industrial manipulator and obstacles in its workspace (Standard ISO 10218-1). The suggested strategy aims to offer an industrial solution to the problem: an off-line analysis of the workspace is performed to have an exhaustive and intrinsically description of the static obstacles and a safe spatial grid of "pass-through points" is calculated; an on-line algorithm, based on an enhanced Artificial Potential Field evaluates the most suitable points to avoid collisions against obstacles and perform a realtime replanning the path of the robot. A Matlab toolbox that elaborates STL CAD files has been developed to obtain a full description of the workcell, and the avoidance algorithm has been designed and implemented in a standard industrial controller. Various experimental results are reported by using a COMAU NS16 arm manipulator.**

I. Introduction

The collision avoidance problem deals with the planning and the control of the motion [1]. In the literature, planning approaches are well suited to achieve a target position in known static environments [2], [3], [4], while real-time obstacle avoidance methods allow reactive motion behaviour in dynamic and unstructured environments, whose knowledge is strictly local, provided by suitable sensors during the motion [5], [6], [7], [8], [11]. Very interesting combinations of these approaches have also been investigated [9], [10] for mobile robots navigation.

SMEs scenarios present different aspects if compared with mobile robots navigation [13]: industrial robots (IRs) follow a trajectory always well-defined that is planned to perform the task avoiding all the static obstacles inside the workcell, and four different situations can occur, as shown in Figure 1: (1) neither the trajectory nor the velocity can be changed since this would produce a process failure; (2) the trajectory can be modified while the cycle time is imposed; (3) the trajectory cannot be changed whereas the slowing down or the interruption of the task execution is allowed; (4) both the trajectory and the velocity can be modified in runtime. Furthermore, applications can be classified as follows: *desktop applications* where the robot size is lower than the obstacles ones; *shop floor applications* where the robot size is comparable or greater than the obstacles ones. In the former group, the standard scenario consists in the small size assembly where the robot is like a third hand of the human operator. A critic feature consists on the fact that the human movements are fast and not easily predictable. Moreover, the workspace of the robot is quite limited (*i.e.*, simple and short trajectories), the obstacles configuration of the workspace changes quickly and they cannot be easily overcome. All these characteristics do not allow an easy re-plan algorithm and, in the authors' opinion, the problem can be mainly shifted towards the identification of the presence of obstacles within the workspace. Concerning the latter scenario, the robot performs extensive trajectories and the suspension of the task execution is a restrictive strategy since the robot usually has free space to avoid obstacles, respecting application requirements (Figure 1). However, in the authors' knowledge there are very few experimental results concerning collision avoidance algorithms applied to industrial scenario. The paper tries to overcome this lack and an intrinsically safe strategy is suggested for *shop floor applications*. The base idea of the algorithm is that the trajectory can be subdivided into a collection of nodes and an off-line process identifies a grid of alternative "pass-through points" for each node. During the motion execution, an on-line control strategy, based on an Artificial Potential Field algorithm [5], selects, at each instant, the most suitable point within the grid of pre-calculated "pass-through points" for the next node of the trajectory, according to the constraints imposed by static and dynamic obstacles. In order to allow an easy implementation of the algorithm in industrial controllers, the point selected by the algorithm is sent to the controller as a target for its inner trajectory planner. The paper reports as proof of concept the first experimental results using a COMAU NS16 robot that demonstrates the effectiveness of the algorithms in an industrial-like scenario.

SCENARIO (1) VELOCITY modification NOT allowed and PATH REPLAN NOT allowed	
Mechanical/Laser/Plasma/waterjet cutting; Arc/Laser welding Polishing/Grinding/Deburring/ Painting (finishing)	**FENCES MANDATORY**
SCENARIO (2) VELOCITY modification NOT allowed, PATH REPLAN allowed	
Special applications where sync with different stations is mandatory or where there is a limited execution time	**FENCES MANDATORY/UNNECESSARY (depends on task)**
SCENARIO (3) VELOCITY modification allowed, PATH REPLAN NOT allowed	
Painting, Deburring (rough)	**FENCES MANDATORY**
Gluing, Cleaning	**FENCES UNNECESSARY**
SCENARIO (4) VELOCITY modification allowed, PATH REPLAN allowed	
Spot Welding, Drilling	**FENCES MANDATORY**
Pick'n'place, Palletising, Material/food handling, Assembly/Micro ass./ Disassembly Measurement, Riveting	**FENCES UNNECESSARY**

Fig. 1. Possible scenarios in Small Medium Enterprises and applications.

N. Pedrocchi, M. Malosio, L. Molinari Tosatti are with ITIA-CNR, Milan, Italy nicola.pedrocchi@itia.cnr.it

II. DESCRIPTION OF THE ALGORITHM

Usually, collision avoidance strategies based on Artificial Potential Field, as [5], [6], [13], solve the path planning problem by imposing repulsive and attractive forces, where the sources are respectively the obstacles (static and dynamic) and the target. Two main drawbacks of this approach can be identified: it cannot be easily integrated in industrial robots controls because it requires to substitute the real-time path planning algorithm already implemented in controls; the path followed by the robot cannot be verified and validated before execution since it is completely generated in run-time.

The suggested algorithm approach is quite different: a cloud of safe-points, *i.e.,* that are safely achievable by the robot with respect to static obstacles, is computed around the *nominal trajectory* (defined by the user) and, for each of these points, the repulsive forces generated by static and by dynamic obstacles are calculated. The computed resultant force is not imposed to the TCP (or to the joints) but it defines only the evasive direction of the movement for the robot, *i.e*, at each instant the new target position sent to the robot position controller is chosen within the pre-calculated grid, so that it keeps the robot farther than a minimum distance from the obstacles and that it lies in the direction "nearer" to the one calculated as resultant.

The required operations are performed by two different modules: an off-line pre-processing of the workspace and an on-line control strategy. The former operation allows to take into consideration all the information concerning the workcell, needed in the task planning phase; the latter is the control algorithm that modifies the robot behaviour in order to avoid the obstacles. Furthermore, let us denote as v^{obst} and as V^{obst}_{MAX} respectively the obstacle velocity and the maximum allowed one, we impose that the replanning strategies are applied only if ($v^{obst} < V^{obst}_{MAX}$), otherwise no replanning is possible with an acceptable robot behaviour, *i.e.*, the movements requested could not be achieved by the manipulator, and/or the human reaction that is caused by a too fast modification of the trajectory performed by the robot could be completely unforeseeable and, consequently, dangerous.

A. Off-line processing: Nominal Path and grid of "Pass-Through" points.

The first step of the algorithm concerns with the analysis of the task that the robot has to execute taking into account the kinematic of the robot and the presence of obstacles in the work cell. In order to have a robust robot behaviour feasible in industrial scenarios and exploiting standard industrial controllers, we developed a simple recursive method that calculates a grid of "pass-through points", *i.e.*, a set of points safely achievable by the robot at different instants.

As first operation, the algorithm transforms the *nominal trajectory*, *i.e.*, an user-defined continuous sequence of linear, circular or spline segments, in the *nominal path*, *i.e.*, a sequence of discrete n_n points $\{\mathbf{P}^i\}_{i=1...n_n}$ that lie on the *nominal trajectory*. The second step consists on the calculation for each point $\mathbf{P}^i$ of a set of evasive points, which lie on the plane containing $\mathbf{P}^i$ and orthogonal to the trajectory, and which belong to different circumferences concentric to $\mathbf{P}^i$ (see Figure 2). Denoting as n_c the number of circles and as n_r the number of nodes for each circle, the set $T^i \equiv \{\mathbf{T}^i_{j,k}\}_{j=1...n_c, j=1...n_r}$, that collects all the "pass-through" points corresponding to $\mathbf{P}^i$, is introduced. Practically, the subindex j corresponds to a different "warning level", since the distance from the point $\mathbf{P}^i$ and the point $\mathbf{T}^i_{j,k}$ increases with j, and the subindex k corresponds to a different "evasive direction". Finally, the set of all the evasive points so calculated is denoted as $T \equiv \{T^i\}_{i=1...n_n}$ and it represents the grid of "pass-through" points. The radii of the circles and the number of the rays can be defined by the user depending on the different typologies of application. Once defined T, both the inverse kinematic and the interference of the robot with objects in the environment are tested and not acceptable points are deleted from T.

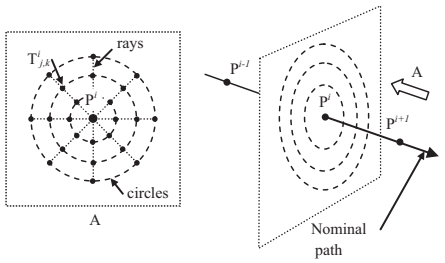


Fig. 2. "pass through points" for each node of the nominal path.

B. Off-line processing: Static obstacle description and repulsive force calculation.

The second problem to be solved is an exhaustive description of the workcell. To reach this goal, an algorithm for the analysis of the STL CAD file format has been conceived and implemented and published in [14]. Through the STL standard, the environment can be described as a group of tessellated surfaces with triangular elements. The preprocess module performs the refinement of the STL model increasing the mesh density close to edges and vertexes, and modifies the 3D model increasing it up in order to take into consideration a safe-tolerance user definable distance. The result is a new STL model of the environment where the original surfaces are translated and the corners are smoothed in order to have a safe-intrinsic description of the workcell. In the sake of simplicity, in this article the environment is modeled as a set of l connected triangular elements denoted hereafter as e_k, with $k = 1...l$. Denoting as $\mathbf{x}$ a generic point with respect to the absolute frame of reference $\{\mathscr{A}\}$, the set of points describing the surface of the triangular element will be denoted as:

$$\sigma_k \equiv \left\{ \mathbf{x} \in \mathbb{R}^3 \,:\, \mathbf{x} \text{ belongs to } e_k \right\}. \qquad (1)$$

The algorithm takes into account two sources for the artificial potential field: (1) each element of the STL model;

(2) the end strokes of the joints.
Concerning the former, the idea is that each element of the STL model is similar to a mass point where the mass value is expressed by the part of its area seen from a generic point **P** of the workspace (see Figure 3).
The first problem to face is that the generic e_k element can be completely/partially hidden by other elements, and the calculation of the part of area that actively affects the point **P** is necessary. From a mathematical point of view, the segment between the point **P** and the point **x** is denoted as

$$L(\mathbf{P},\mathbf{x}) \equiv \left\{ \mathbf{P}+t(\mathbf{x}-\mathbf{P}) : \mathbf{P},\mathbf{x} \in \mathbb{R}^3, \mathbf{P} \neq \mathbf{x}, t \in [0,\ 1] \right\}, \tag{2}$$

and, consequently, the corrected area of each element k is

$$\sigma_k^*(\mathbf{P}) \equiv \left\{ \mathbf{x} \in \sigma_k : \forall j \neq k \quad L(\mathbf{P},\mathbf{x}) \wedge \sigma_j = \emptyset \quad \right\}. \tag{3}$$

If the element e_k is completely hidden by another element, the set is empty, *i.e.*, $\sigma_k^* = \emptyset$. The $\sigma_k^*(\mathbf{P})$ is the set description of the flat surface corresponding to the element e_k seen by the point **P**, and *it could be not connected*. However, as shown in Figure 3, the value of the area of σ_k^* is not a correct estimation for the "mass" of the element. Indeed, as in physics for the flux calculation methods, the surface has to be projected orthogonally towards the observation point **P**. In order to calculate it, θ denotes as the angle between the unit normal vector $\mathbf{n}_k$ corresponding to the element e_k and the unit vector $\mathbf{d} = (\mathbf{P}-\mathbf{x})/\|\mathbf{P}-\mathbf{x}\|$. The effective weighted value of the *active area* $A_k \equiv A_k(\mathbf{P})$ is:

$$A_k(\mathbf{P}) = \int_{\sigma_k^*(\mathbf{P})} (\cos\theta)^2 \, d\sigma_k^* \tag{4}$$

The suggested expression for the artificial force $\mathbf{F}^{env}$ generated by the environment in the point **P** is:

$$\mathbf{F}^{env} \equiv \mathbf{F}^{env}(\mathbf{P}) = \lambda_T \sum_{k=1}^{l} \left(\frac{A_k(\mathbf{P})}{\|\mathbf{P}-\mathbf{G}_k\|\eta_T} \right) \mathbf{n}_k \tag{5}$$

where the terms η_T and λ_T are respectively a distance weight and a suitable constant to tune the behaviour of the system; $\mathbf{G}_k$ denotes the nearest point of e_k to **P**.

Regarding the latter source of repulsive force, $\mathbf{q} \equiv \mathbf{q}(\mathbf{P})$ denotes the joint position vector corresponding to the position **P** obtained applying the inverse kinematics and as $\mathbf{J} \equiv \mathbf{J}(\mathbf{P})$ the jacobian matrix. Furthermore, $\mathbf{q}^-$ and as $\mathbf{q}^+$ denote the vectors containing the negative and positive end-strokes joints positions respectively. For the k-th joint the repulsive torque generated by the presence of the end-strokes is imposed to be equal to:

$$C_k^{stroke} \equiv C_k(\mathbf{P}) = \\ = \lambda_{q_k} \left(q_k^+ - q_k^-\right)^{\eta_{q_k}} \left[-\frac{1}{\left(q_k^+ - q_k\right)^{\eta_{q_k}}} + \frac{1}{\left(q_k - q_k^-\right)^{\eta_{q_k}}} \right] \tag{6}$$

where the terms η_{q_k} and λ_{q_k}, both positive, are respectively a distance weight to determine the strength of the behaviour of the algorithm near end strokes and a suitable constant to tune the behaviour of the system. Note that they can be imposed differently for each joint. The corresponding force

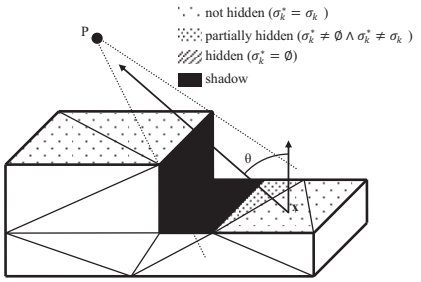


Fig. 3. Possible cases for the calculation of the area of the k-th element.

$\mathbf{F}^{stroke} \equiv \mathbf{F}^{stroke}(\mathbf{P})$ applied in the point **P** due to all the $C_k^{endstroke}$ is:

$$\mathbf{F}^{stroke} = \left(\left[\begin{array}{ccc} C_1^{stroke} & \dots & C_{dof}^{stroke} \end{array} \right] \mathbf{J}^{-1} \right)^T \tag{7}$$

Finally, the repulsive force imposed by static obstacles $\mathbf{F}^{static} \equiv \mathbf{F}^{static}(\mathbf{P})$ is calculable by the analytical expression:

$$\mathbf{F}^{static} = \mathbf{F}^{stroke} + \mathbf{F}^{env} \tag{8}$$

C. On-line processing: avoidance algorithm

The model for the dynamic obstacle is needed. A suitable description consists on consider that at each instant the measurement system describes the obstacles as a limited collection of points that lie to their bounding surfaces, like in the tracking of markers in the artificial vision.
Denote as n_o the number of different obstacles point, as $\mathbf{O}^k$ the position of the k-th one. Moreover, if the actual distance between the obstacle and the trajectory is greater than D (*warning distance*) no avoidance algorithm is applied and if it is lower than d (*forbidden distance*) task execution is suspended (see Figure 4).
Each dynamic obstacle is a source of repulsive force as in the APF strategies, and for the generic k-th one, the repulsive force $\mathbf{F}^{dyn,k} \equiv \mathbf{F}^{dyn,k}(\mathbf{P})$ in the point **P** of the workspace is:

$$\mathbf{F}^{dyn,k} = \\ \begin{cases} \lambda_o & \text{if } \|\mathbf{O}^k - \mathbf{P}\| < d \\ \dfrac{\lambda_o d^2}{D^2 - d^2} \left(\dfrac{D^2 - \|\mathbf{O}^k - \mathbf{P}\|^2}{\|\mathbf{O}^k - \mathbf{P}\|^2} \right) \dfrac{\mathbf{O}^k - \mathbf{P}}{\|\mathbf{O}^k - \mathbf{P}\|} & d < \|\mathbf{O}^k - \mathbf{P}\| < D \\ 0 & \text{if } \|\mathbf{O}^k - \mathbf{P}\| > D \end{cases} \tag{9}$$

where λ_o is a user suitable constant that allows to tune the system. The resultant repulsive force $\mathbf{F}^{rep} \equiv \mathbf{F}^{rep}(\mathbf{P})$ is given by adding both dynamic than static forces:

$$\mathbf{F}^{rep}(\mathbf{P}) = \sum_{k=1}^{n_o} \mathbf{F}^{dyn,k} + \mathbf{F}^{static} \quad \text{and} \quad \mathbf{f}^{rep}(\mathbf{P}) = \mathbf{F}^{rep}/\|\mathbf{F}^{rep}\|. \tag{10}$$

In APF strategies this force is balanced by an attractive force, and the resultant is exerted on the robot. In the suggested algorithm the resultant force is calculated only to chose the best evasive "pass-through" point among the available ones. Denote as $\left\{\mathbf{P}^i\right\}_{i=1\dots n}$ the *actual path* the

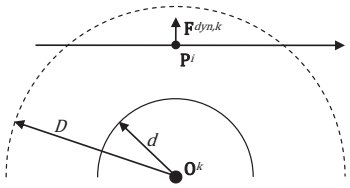
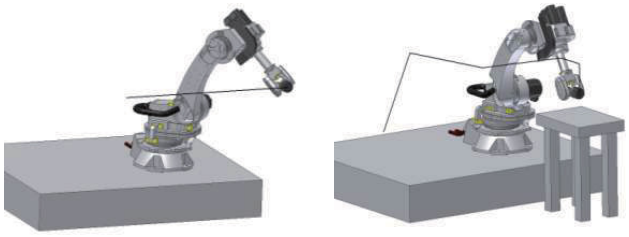


Fig. 4. Dynamic obstacles description and calculation of the new target on the basis of the repulsive force. D is the Warning Distance, if the obstacles are farther than it no modifications to the trajectory are applied; d is the Forbidden Distance, if the distance is less than it the task is suspended.



(a) Linear Trajectory and no static obstacles within the workspace (b) Complex trajectory and a simple static obstacle within the workspace

Fig. 5. The two tested environment for the algorithm.

robot has to follow, where each element is selected on line during the task execution within the set of available nodes T. When the robot moves from the nodes $\mathbf{P}^{i-1}$ to the node $\mathbf{P}^i$ the algorithm imposes the new target point $\mathbf{P}^{i+1} = \mathbf{T}_{j,l}^{i+1}$ where the level of warning j and the direction l are chosen in order to maintain the TCP far away from the obstacle and greater than the minimum distance d and in order to be quite parallel to the projection of the resultant force $\mathbf{f}^{rep}(\overline{\mathbf{P}}^{i+1})$ to the plane orthogonal to the trajectory in the point $\overline{\mathbf{P}}^{i+1}$. In comparison with respect a traditional APF strategy, this technique is more suitable for industrial scenario because only a new target position is sent to the robot controller and the interpolation and the safe execution of the new trajectory is completely demanded to it. In order to achieve a good robot behaviour, the control algorithm has to perform fast response to avoid the obstacle but it has also to guarantee a predictable response according to the human operator perception, *i.e.*, the trajectory change has to be smooth. The reduction of the execution task velocity, modifying the speed override, ovr^{obst}, is a suitable solution.

The implemented algorithm calculates the override on the basis of the distance between the obstacles and the TCP of the robot and on the obstacles velocity. Denoting as λ_{dist}^k the term that take into account the distance between the k-th obstacle and the TCP, and denoting as λ_{vel}^k the term that take into account the velocity of the k-th, and denoting as V_{min}^{obst} the minimum obstacle velocity so that if the actual speed is smaller, no velocity reduction is applied. Therefore, the speed override $ovr \equiv ovr(\mathbf{P}, v^{obst})$ is:

$$\begin{aligned}
\lambda_{dist}^k &= \begin{cases} 1 & \text{if} \quad \|\mathbf{O}^k - \mathbf{P}\| > D \\ \dfrac{\|\mathbf{O}^k - \mathbf{P}\| - d}{D - d} & \text{if} \quad d < \|\mathbf{O}^k - \mathbf{P}\| < D \\ 0 & \text{if} \quad \|\mathbf{O}^k - \mathbf{P}\| < d \end{cases} \\
\lambda_{vel}^k &= \begin{cases} 1 & \text{if} \quad v^{obst,k} < V_{min}^{obst} \\ \dfrac{V_{max}^{obst} - v^{obst,k}}{V_{max}^{obst} - V_{min}^{obst}} & \text{if} \quad V_{min}^{obst} < v^{obst,k} < V_{max}^{obst} \\ 0 & \text{if} \quad v^{obst,k} > V_{max}^{obst} \end{cases} \\
ovr\,[\%] &= ovr_{min} + (100 - ovr_{min})\, min_{k=1..n_o}(\lambda_{dist}^k \lambda_{vel}^k)
\end{aligned} \tag{11}$$

where ovr_{min} is the minimum override velocity the robot has to guarantee. When the obstacle velocity is greater than the maximum allowed or the distance is lower less than the minimum one, the task is suspended.

III. Experimental results

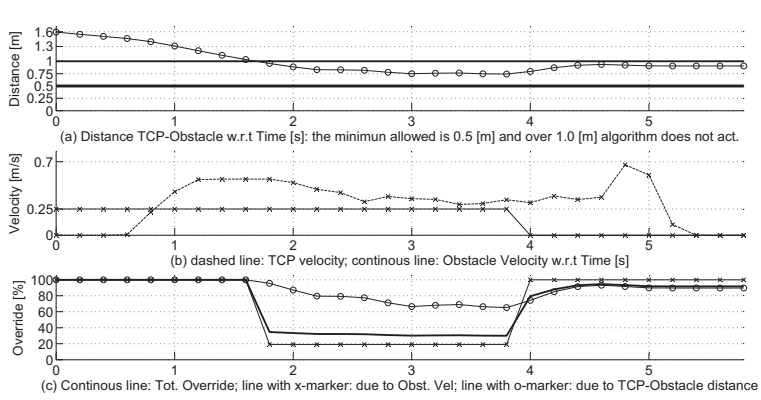
The algorithm has been tested on a COMAU NS16 available at ITIA laboratory that is a serial anthropomorphic robot arm with a maximum extension of 1.650 [m]. A toolbox for the analysis of the STL file and of the nominal path has been developed in Matlab®. Before the execution of the test, the industrial controller sends, through an ethernet socket, the information about the nominal trajectory to a PC, and it receives the grids of all the "pass-thorough" points safely achievable. The on-line replanning algorithm has been implemented on COMAU C4G controller and PDL2 language. The test cases have been developed using a virtual obstacle and its position is imposed by a Matlab user interface and it is sent via TCP-IP socket to the C4G controller, while the robot joints position are sent to the Matlab application to update a virtual 3D environment and to elaborate the experimental results. Two series of tests have been carried out. In the first test the robot has to follow a path with no near static obstacle (see Figure 5-(a)); in the second a more complex environment and trajectory are used (see Figure 5-(b)).

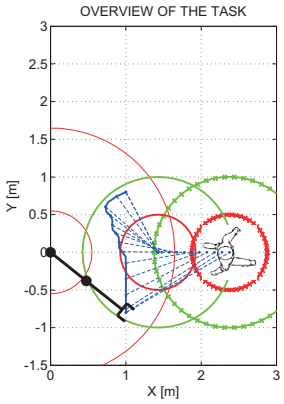
A. No Forces exerted by the environment

The experiments have been performed imposing an horizontal linear trajectory of 1,600 [mm] length far away from the static obstacles, *i.e.*, the environment forces are neglectable. The robot nominal linear velocity has been fixed at 1,000 [mm/s] (achievable only in absence of obstacles).

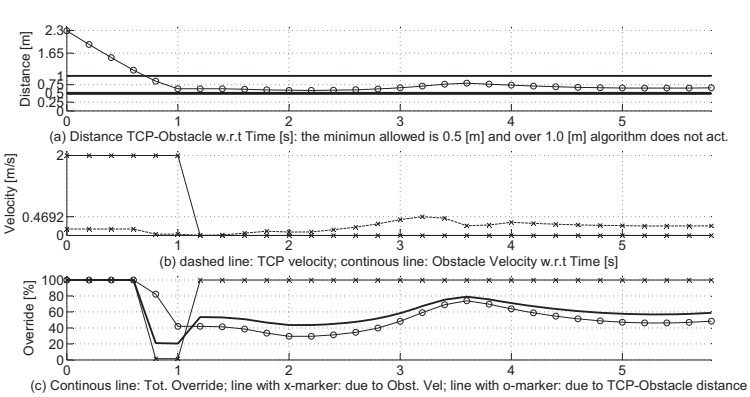
(Test 1) The operator is slowly approaching perpendicularly to the robot trajectory, with a linear velocity of 250 [mm/s]. Figure 6-(Test 1) shows that the robot keeps a safety distance from the human operator. Note that the algorithm tries to maintain the robot outside the *Obstacle Warning Zone*, however for the last point of the trajectory this behaviour would cause the not completion of the task. To face this problem if the last node is outside the *Obstacle Forbidden Zone*, the robot is allowed to reduce its distance from the obstacle and achieve the goal of the task, and the override modification is always kept active.

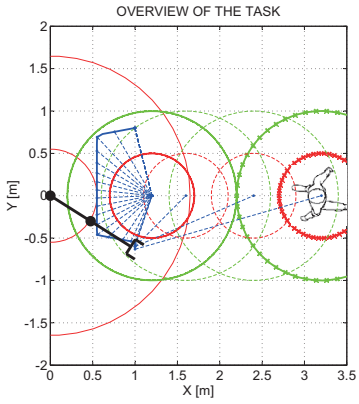
(Test 2) The human is quickly approaching perpendicularly to the robot trajectory, with a linear velocity of 2000 [mm/s]. By increasing the obstacle velocity any appreciable change is noticed in the robot trajectory. When the human stops



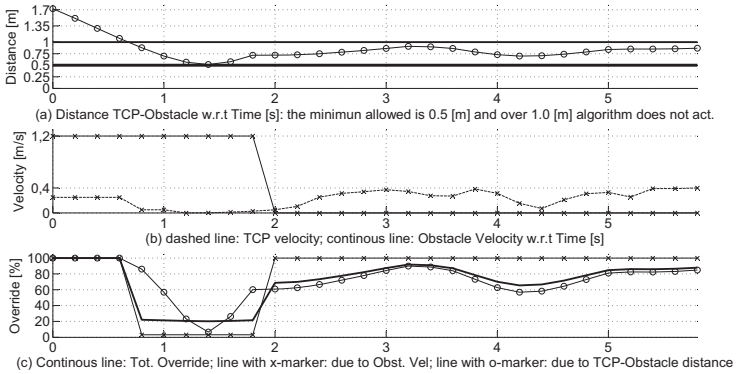


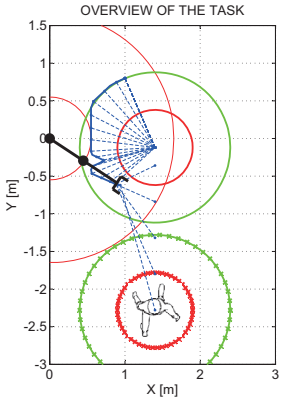
(a) Test 1, perpendicular approach; 250 [mm/s] as human velocity



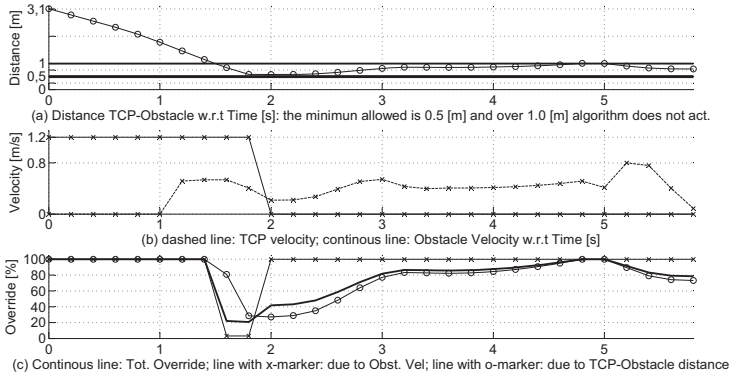


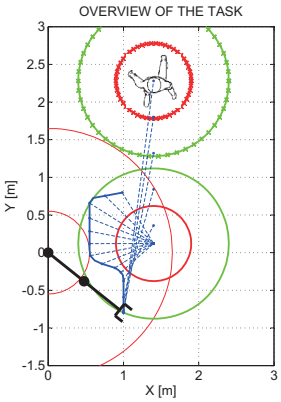
(b) Test 2, perpendicular approach; 2000 [mm/s] as human velocity





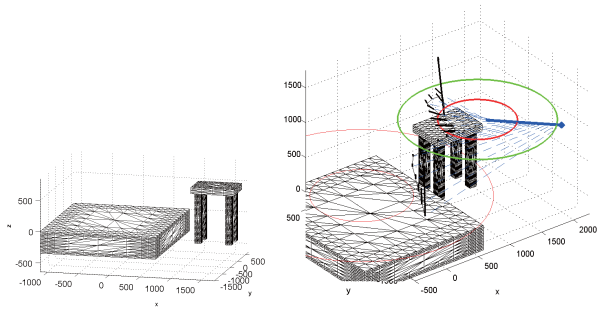
(c) Test 3, collinear approach; 1.2 [mm/s] as human velocity





(d) Test 4, opposite approach; 1.2 [mm/s] as human velocity

Fig. 6. Tests where the environment does not exerts forces.



(a) STL-Model of the environment after processing (b) Calculated forces due to the environment and to the dynamic obstacles.

Fig. 7. Test 2 (see Figure 5)

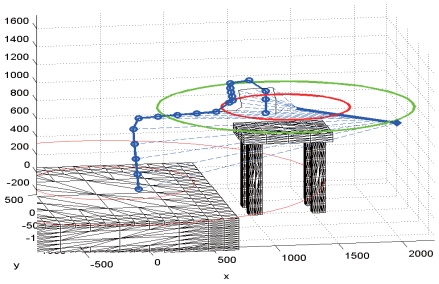


Fig. 8. Actual (circles-line) and nominal (thin-line) trajectory in test 2 (see figure 5-(b)).

its motion, the robot controller looks for the trajectory that allows the greatest distance between the TCP and the human, within the available grid of "pass-through" points.

(Test 3) The human is moving parallelly to the robot trajectory in the same direction, with an offset of 400 [mm] and with a velocity of 1,200 [mm/s]. Note that at time t=1.4 [s] the TCP goes in the *Obstacle Forbidden Zone* and the task is suspended.

(Test 4) The robot TCP and the human operator are moving in opposite directions. The distance between the robot and the obstacle decreases fastly (see Figure 6-(Test 4)). When the distance decreases below a certain limit the correction due to the algorithm starts to correct and deforme the original trajectory. Note that the robot does not stop the motion and is able to avoid the obstacle.

B. Complex environment

The second test (Figure 5) is more complex and a structured environment is imposed. Due to the limited workspace of the available robot the experimental results are not exhaustive. The Figure 7-(a) shows the STL model of the work cell, and the Figure 7-(b) displays the resulting force calculated taking into account both the environment and the human operator that approaches to the robot. As shown in Figure 8, the robot does not intersect the *Obstacle Forbidden Zone* near the obstacle and a trajectory that conclude correctly the task is found by the controller. The reduction of the velocity, modifying the override value, is mandatory to have a good and satisfiable robot behaviour. Various tests have been performed and they prove the robustness of the algorithm and its feasibility for many different industrial applications.

IV. Conclusion

This paper has addressed the problem of obstacles collision avoidance by combining different strategies. The algorithm provides a pre-processing toolbox developed in Matlab to exhaustively model the robot cell by a STL CAD file. A grid of evasive "pass through" points is calculated in order to have a deterministic behaviour of the robot in the collision avoidance. The dynamic obstacles have been modeled as a set of dimensionless points surrounded by a *warning* and a *forbidden* zone. An APF technique has been developed in order to choice, at each instant, the optimum position for the manipulator. Experimental results proved the effectiveness of the algorithm and its feasibility for industrial applications.

V. Acknowledgement

This work has been [partially] funded by the European Commissions Sixth Framework Programme under grant no. 011838 as part of the Integrated Project SMErobot^{TM}.

References

[1] La Valle. *Planning Algorithms*. University of Illinois, 2005.
[2] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
[3] F. Aurhennamer, Voronoi Diagrams: A Survey of a Fundamental Geometric Data Structure ACM Computing Surveys, Vol 23, No 3, September 1991.
[4] K. Jiang, L. D. Seneviratne and S. W. E. Earles, A Shortest Path Based Path Planning Algorithm for Nonholonomic Mobile Robots, Journal of Intelligent and Robotic Systems 24: 347:366, 1999.
[5] O. Khatib, Real-time Obstacle avoidance for robot manipulator and mobile robot. The International Journal of Robotics and Automation, 5(1): 90 - 98, 1986
[6] M. Khatib, R. Chatila. An extended potential field approach for mobile robot sensor-based motions, In Proc. of Int. Conf. on Intelligent Autonomous Systems, pages 490-496, 1995.
[7] B. R. Perry, Tesar, D., *The Development of Distance Functions and Their Higher-Order Properties for Artificial Potential Field-Based Obstacle Avoidance*. University of Texas at Austin, February, 1995.
[8] J. Reif, M. Sharir, Motion Planning in the Presence of Moving Obstacles Journal of the ACM (JACM), v.41 n.4, p.764-790, July 1994
[9] O. Brock, O. Khatib, High-speed Navigation Using the Global Dynamic Window Approach. Proc. IEEE Int. Conf. on Rob. and Aut., Detroit. Michigan USA, May 1999.
[10] S.S. Ge, Y.J. Cui, Dynamic Motion Planning for Mobile Robots Using Potential Field Method, Autonomous Robots Volume: 13, Issue: 3, November, pp. 207-222, 2002,
[11] J. Agirrebeitia, R. Aviles, I. F. De Bustos, G. Ajuria A new APF strategy for path planning in environments with obstacles Mechanism and machine theory, 2005, vol. 40, no6, pp. 645-658
[12] O. Brock, O. Khatib, Executing Motion Plams for Robots with Many Degrees of Freedom in Dynamic Environments, In Proc. Int. Conf. on Robotics and Automation, volume 1, pages 1-6, 1998.
[13] N. Pedrocchi, M. Malosio, L. Molinari Tosatti, G. Ziliani, Obstacle Avoidance Algorithm for Safe Human-Robot Cooperation in Small Medium Enterprise Scenario 40th International Symposium on Robotics - ISR 09, May 10-13, 2008.
[14] M. Malosio, N. Pedrocchi, L. Molinari Tosatti, Algorithm to Offset STL Geometries CAD09, International CAD Conference and Exhibition, Reno, Nevad, June 8-12, 2009