A Convergent Reactive Collision Avoidance Approach using Fast Marching Methods

Afonso Ferreira, *MeAer Student, IST,* Rodrigo Ventura, *Institute for Systems and Robotics, IST,*

***Abstract*—This work presents a novel way of using Fast Marching Methods for reactive collision avoidance. A new controller is proposed, proving convergence to the goal position as well as convergence of the path to the gradient lines of the navigation function. In addition, proof of obstacle avoidance is also shown, taking into account the robot's dynamic and kinematic constraints. It is assumed the robot is using a distance measuring sensor and an implementation is made such that the robot is aware of its visibility limits. To the best of our knowledge, this is the first convergent solution to guarantee obstacle avoidance not only in a pre-established map, but also in dynamic environments, where new static obstacles may appear. When compared to a convergent Dynamic Window Approach implementation, this controller shows improved stability, with no oscillations near obstacles and narrow passages.**

***Index Terms*—Fast Marching Methods, Reactive Collision Avoidance, Mobile Robots, navigation function, Lyapunov Theory**

I. Introduction

MOST robotic applications require the ability of navigating autonomosly to a given goal position, while avoiding existing obstacles. Therefore, robot navigation and obstacle avoidance techniques have been an important research topic since the 80s. Between several methods such as Dynamic Window Avoidance [1] and Nearness Diagram Navigation [2], one of the popular solutions was the use of Artificial Potential Functions.

Andrews and Hogan [3] and Khatib [4] were the first to suggest the idea of imaginary forces acting on a robot as an approach to this problem. Since then, a lot of different solutions emerged but they all had something in common: obstacles would apply repelling forces to the robot and the goal would exert an attractive force. Despite the beautiful simplicity of these methods, some supposedly inherent limitations where found in [5]: trap situations due to local minima and oscillations in narrow passages.

Nevertheless, Artificial Potentials have since then been a part of both path planning and real-time controllers.

One of the most recent breakthroughs in Artificial Potentials has been the introduction of Fast Marching Methods (FMM), introduced by Sethian in [6] , which has been increasingly used in robot motion planning since 2006. The use of Fast Marching Methods in [7] and [8] showed how this potential field can be used successfully to generate paths both for single robot navigation and robot formations. However, generating a path using gradient descent, that it's then followed by controller, is only one of the ways of using FMM. Only using this generated path means there is a lot of the information contained within this potential field that it is being thrown away.

Another ways of computing Artificial Potentials free of local minima were developed in [9], [10] and [11], but they also only mention the use of these Artificial Potentials for path planning.

This lead to the conclusion that most Artificial Potentials are either being used for path planning or assume to be able to control the robot directly in the velocity space, and do not take into account the robot's dynamic and kinematic constraints.

A very popular method used for reactive collision avoidance has been the Dynamic Window Approach (DWA), introduced in [1]. DWA considers the robot's dynamic and kinematic constraints, and in [12] an approach is shown that proves collision avoidance together with convergence to the goal. However, the way the robot reacts to unexpected obstacles was not defined. It is also going to be shown that this algorithm has downsides: oscillations near obstacles and performance depending largely on the robot's characteristics.

This research leads to the conclusion that there seems to be missing a real-time controller based on minima-free potentials that proves convergence to the goal and with proven obstacle avoidance also in the presence of obstacles not contained in the original map, taking into account the robot's acceleration constraints.

Therefore, a new way of using FMM to create a real-time controller will be presented to be used for reactive collision avoidance. FMM guarantees the goal is the only local minimum and we will show proof of convergence to the goal and convergence of the robot's path to the gradient lines of the navigation function. We will also show this solution to be an improvement on DWA, showing its performance is not affected by changes in the robot's constraints, that the robot reaches its goal in less time and that it does not suffer from oscillations near obstacles or narrow passages

It will also be shown under which conditions new obstacles can appear, introducing the notion of the robot being aware of its visibility limits. After the robot's visibility area is defined, obstacle avoidance in dynamic environments will be proven.

September 22, 2015

II. Previous Work used in this paper

A. Fast Marching Methods

FMM is a computational technique that allows to track the propagation of waves through complex interfaces. It can be applied to any number of space dimensions and work under very complex speed laws, such as abrupt changes in speed and in the presence of discontinuities. FMM, introduced by Sethian in [6], approximates the solution of a boundary value partial differential equation:

$$|\nabla\phi|F = 1,\ F > 0$$
$$Front = \Gamma(t) = \{(x,y)|\phi(x,y) = t\} \quad (1)$$

This concept that the wave front only travels forward means the information travels only "one way", and allows us to construct the solution in a "downwind" fashion, that shows to be very computationally faster [13].

FMM was first applied to optimal path planning in [14]. These results were summarized in [15] and it is shown that given a cost function $H(x_1, x_2, ..., x_n)$ we are able to construct the path $\gamma(\tau) \rightarrow \Re^n$ between two points A and B, minimizing the integral:

$$\int_{A=\gamma(0)}^{B=\gamma(L)} H(\gamma(\tau))d\tau \quad (2)$$

where L is the total length of γ. If we now consider ϕ as the minimal cost to travel from A to any point (x, y), we have:

$$\phi = \min_{\tau} \int_{A}^{(x,y)} H(\gamma(\tau))d\tau \quad (3)$$

It is than shown we can calculate ϕ using FMM by solving the equation $|\nabla\phi| = H$ and obtain the path through back-propagation from B to A by solving the ordinary differential equation

$$\frac{dX(s)}{ds} = -\nabla\phi \quad (4)$$

This result lead to many future work on how we can design this cost function in order to get the desired trajectories, and will be explained in the next subsection.

B. Eikonal equation and Path Planning

The Eikonal equation (5) is a boundary value problem and we can therefore obtain an approximate solution using FMM. In this subsection we will present some physical concepts about this equation and show how they were applied to path planning with obstacle avoidance.

$$|\nabla\phi(r)|P(r) = 1 \quad (5)$$

The Eikonal equation (5) is a non-linear partial differential equation that describes a wave propagation in a non-homogeneous medium, with $P(x)$ being the medium velocity and $r \in \Re^2$ in 2D. $\phi(r)$ is a potential field with value equal to the arrival time of the wave front to point r. As explained in the previous subsection, this has been extensively used in robot path planning because a gradient descent through $\phi(r)$ obtains the path travelled by the wave, which corresponds to the minimum-time path.

These potential functions are minima-free regardless of the medium velocity field we use. This property of the field can be derived from the properties of an electromagnetic wave traveling through a medium.

The concept that light always travels through the minimum-time path leads to the notion that in the presence of a velocity differential, light is refracted to higher velocity regions (fig. 1). This means that if the propagation velocity is lower near obstacles, light (and the obtained path) will be "deviated" away from them.

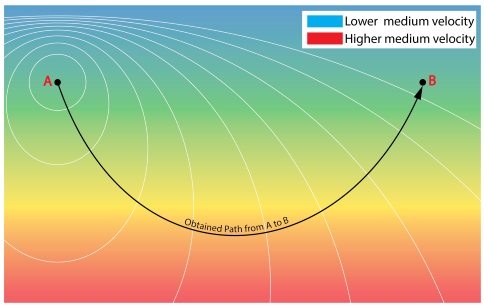


Fig. 1: Light traveling through a continuous change in medium velocity

In [7] and [8] it was shown how $P(r)$ can be modeled to simulate this repulsive force from the obstacles by using a repulsive potential $R_P(r)$ that represents the distance to the nearest obstacle. $R_P(r)$ can be obtained using either the Extended Voronoi Transform (EVT) or FMM by propagating a wave front starting at every obstacle. $P(r)$ can then be calculated through $R_P(r)$ with the method that suits best the problem we are trying to solve. In subsection III-B it will be presented the method we have chosen to design the navigation function.

III. A Proven Convergent Controller

A. Robot Model and Environment

We will use the notation $r = (x, y)$ and adopt a unicycle robot model described by:

$$\dot{x} = v\cos(\theta) \quad (6)$$
$$\dot{y} = v\sin(\theta) \quad (7)$$
$$\dot{\theta} = \omega \quad (8)$$
$$\dot{w} = \frac{\tau}{J} \quad (9)$$

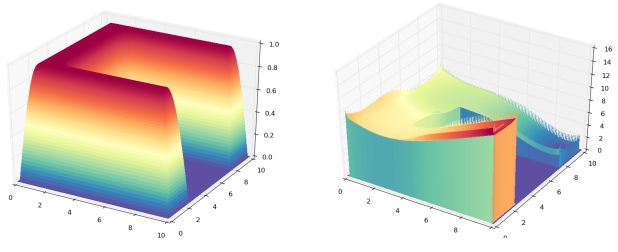
This model can be feedback linearized as in [16], ending up with the model used in [17], which consists of a double integrator $\ddot{r} = u, r \in \mathbb{R}^2$.

As for the environment we assumed the robot's sensors can give us an occupancy grid map, i.e., a grid map with each cell with a value corresponding to its state, free or occupied. We also assume this map takes into account the robot's dimensions. This means that if the robot's position is located at a free cell, the robot does not intersect an obstacle at any point.

B. Navigation Function

As mentioned in subsection II-B, we are going to calculate the medium velocity $P(r)$ through $R_P(r)$, which will be obtained using FMM. Since FMM is ran twice this method is often called FMM^2. We used the following formula:

$$P(x) = \sin\left(\frac{\pi}{2d} clip(R_P(r), 0, d)\right) \quad (10)$$



(a) $P(r)$ (b) $\phi(r)$

Fig. 2: Medium velocity and Navigation Function obtained

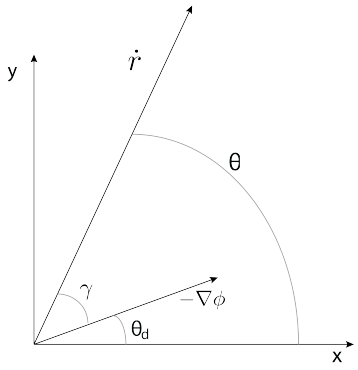


Fig. 3: Considered angles

The function $clip(R_P(r), 0, d)$ returns $R_P(r)$ clipped between 0 and d. An example both $P(r)$ and the navigation function $\phi(r)$ can be seen in fig.2. Note that $P(r)$ goes to 0 near the obstacles, which will be a key point in the proof of obstacle avoidance.

IV. Normal Acceleration controller

The Normal Acceleration controller was designed for the robot to follow the gradient lines of the navigation function $\phi(r)$, while converging to the goal position. After deriving the normal acceleration a_n, we show that this can be done independently from the robot's tangential acceleration a_t, by proving the robot's trajectory will always be same, regardless of a_t and $|\dot{r}|$.

For the calculation of the normal acceleration a_n the following rotational kinematics were taken into account, together with the angles defined in Fig. 3:

$$a_n = \omega v \tag{11}$$
$$\gamma = \theta - \theta_d \tag{12}$$
$$\dot{\gamma} = \omega - \dot{\theta_d} = \frac{a_n}{v} - \dot{\theta_d} \tag{13}$$

To prove the robot will follow the gradient lines in order to converge to the goal as fast as possible (with respect to $\phi(r)$), we need γ to converge to zero. This leads to the following theorem:

Theorem 1 (γ-convergence)**.** *The function*

$$L_1 = \frac{1}{2}\gamma^2 \tag{14}$$

is a Lyapunov function and any a_n:

$$a_n = v\left(-k_n'\gamma + \dot{\theta}_d\right), \textit{with } k_n' \geq 0 \tag{15}$$

satisfies the following inequality:

$$\dot{L}_1 < 0,\ \forall(\gamma, v) \neq (0,0) \tag{16}$$

Proof. The function $L_1 = \frac{1}{2}\gamma^2$ is clearly positive definite with a global minimum at $\gamma = 0$. Differentiating with respect to time gives:

$$\dot{L_1} = \gamma\left(\frac{a_n}{v} - \dot{\theta}_d\right) \tag{17}$$
$$= -k_n'\gamma^2 \tag{18}$$

Therefore, considering $v \neq 0$, $\gamma = 0$ is globally asymptotically stable □

A. Convergence to goal

So far we have proven that γ goes to zero as long $v \neq 0$, which means the robot will follow the gradient lines. Intuitively, this means the robot would eventually converge to the goal position. However, we will define k_n' using a second Lyapunov Function that proves the robot reaches its goal. The main reason for this step to be needed has to do with obstacle avoidance and it will be made clear in subsection VI. This leads to the following theorem:

Theorem 2 (Convergence to goal)**.** *The function*

$$L_2 = k\phi + L_1 \tag{19}$$

is a Lyapunov function and any a_n:

$$a_n = \begin{cases} v\left(-k_{n_c} v\gamma + \dot{\theta}_d\right) & , cos(\gamma) \geq 0 \\ v\left(-k_{n_c} v\gamma + \frac{kv|\nabla\phi|\cos(\gamma)}{\gamma} + \dot{\theta}_d\right) & , cos(\gamma) < 0 \end{cases} \tag{20}$$

with $k_{n_c} > 0$, satisfies the following inequality:

$$\dot{L}_2 < 0,\ \forall(\gamma, v) \neq (0,0) \tag{21}$$

Proof. The function $L_2 = k\phi + L_1$ is clearly positive definite with a global minimum at $(r, \gamma) = (r_{goal}, 0)$. Differentiating with respect to time gives:

$$\dot{L}_2 = k\dot{r}^T\nabla\phi - k_n'\gamma^2 \tag{22}$$
$$= -kv|\nabla\phi|cos(\gamma) - k_n'\gamma^2 \tag{23}$$

As shown in (15), the only constraint is that k_n' needs to be non-negative. Therefore, we can than define it as:

$$k_n' = k_n v, \text{with } v > 0 \text{ and } k_n > 0 \tag{24}$$

If we now define k_n, using the positive constant k_{n_c}, as being:

$$k_n = \begin{cases} k_{n_c} & , cos(\gamma) \geq 0 \\ k_{n_c} - \frac{k|\nabla\phi|cos(\gamma)}{\gamma^2} & , cos(\gamma) < 0 \end{cases} \tag{25}$$

Replacing $k'_n = k_n$ equation in (22) gives:

$$\dot{L}_2 = v\left(-k|\nabla\phi|cos(\gamma) - k_n\gamma^2\right) \tag{26}$$

$$= \begin{cases} v\left(-k|\nabla\phi|cos(\gamma) - k_{n_c}\gamma^2\right) & , cos(\gamma) \geq 0 \\ -k_{n_c}v\gamma^2 & , cos(\gamma) < 0 \end{cases} \tag{27}$$

$$\Rightarrow \dot{L_2} < 0 \text{ , for } \forall(\gamma, v, r) \neq (0, 0, r_{goal}) \tag{28}$$

This way, we have proof that L_2, being a continuous and differentiable function with a non-positive derivative, is a suitable Lyapunov function and that for $v \neq 0$, $(\gamma, r) = (0, r_{goal})$ is globally asymptotically stable □

As we will show in section V, $v > 0$ for $t > 0$, ensuring our permise is correct and that the system is globally asymptotically stable.

B. Calculating $\dot{\theta}_d$

This is one of the crucial parts of this controller and the one depending on the use of FMM. Usually, the calculation of $\dot{\theta}_d$ could be impractical because it would lead to a significant error in real applications. However, because we are using FMM, we will show how we can use the eikonal equation in our advantage. Using the notation $[\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial^2\phi}{\partial x^2}, \frac{\partial^2\phi}{\partial y^2}, \frac{\partial^2\phi}{\partial xy}] = [\phi_x, \phi_y, \phi_{x^2}, \phi_{y^2}, \phi_{xy}]$ we will derive $\dot{\theta}_d$.

$$\theta_d = tan^{-1}\left(\frac{\phi_y(r)}{\phi_x(r)}\right) \tag{29}$$

$$\Rightarrow \dot{\theta_d} = \frac{1}{|\nabla\phi|^2}\left(\begin{bmatrix}\phi_{xy}\phi_x - \phi_{x^2}\phi_y & , & \phi_{y^2}\phi_x - \phi_{yx}\phi_y\end{bmatrix}\right)\dot{r} \tag{30}$$

Which can also be writen as:

$$\dot{\theta_d} = \frac{1}{|\nabla\phi|^2}\dot{r}^T(\nabla^2\phi M\nabla\phi), \text{with } M = \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix} \tag{31}$$

In (31) it is seen that $\nabla^2\phi(r)$ has to be calculated. In other scenarious we would need to use a discrete approximation to calculate all the second derivatives, as in [18]:

$$f_{x^2}(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}, \text{with } h = x_1 - x_0 \tag{32}$$

Because, in practice, $\phi(r)$ is a discrete potential, and it is already an approximation, this would lead to a significant error. However, because we are using FMM, we can use the information given by Eikonal equation (5) to minimize this error, and therefore have much better results.

$$\begin{aligned}
&|\nabla\phi|^2P^2 = 1 \Leftrightarrow \\
\Leftrightarrow &\nabla\phi\cdot\nabla\phi = \frac{1}{P^2} \\
\Rightarrow &\frac{d}{dr}\left(\nabla\phi\cdot\nabla\phi\right) = \frac{d}{dr}\left(\frac{1}{P^2}\right) \\
\Leftrightarrow &\nabla^2\phi\cdot\nabla\phi = -\frac{1}{P^3}\nabla P \\
\Leftrightarrow &\begin{bmatrix}\phi_x\phi_{x^2} + \phi_y\phi_{xy}\\ \phi_y\phi_{y^2} + \phi_x\phi_{xy}\end{bmatrix} = -\frac{1}{P^3}\begin{bmatrix}P_x\\ P_y\end{bmatrix} \\
\Leftrightarrow &\begin{bmatrix}\phi_y\phi_{xy}\\ \phi_x\phi_{xy}\end{bmatrix} = \begin{bmatrix}-\frac{P_x}{P^3} - \phi_x\phi_{x^2}\\ -\frac{P_y}{P^3} - \phi_y\phi_{y^2}\end{bmatrix}
\end{aligned} \tag{33}$$

After these calculations, we are able to write (30), using (33), as:

$$\begin{aligned}
\dot{\theta_d} &= \frac{1}{|\nabla\phi|^2}[-\frac{P_y}{P^3} - \phi_y\phi_{y^2} - \phi_{x^2}\phi_y \ , \ \frac{P_x}{P^3} + \phi_{y^2}\phi_x + \phi_x\phi_{x^2}]\dot{r} \\
&= [-\frac{P_y}{P} - P^2\phi_y\phi_{y^2} - P^2\phi_{x^2}\phi_y \ , \\
&\qquad \frac{P_x}{P} + P^2\phi_{y^2}\phi_x + P^2\phi_x\phi_{x^2}]\dot{r}
\end{aligned} \tag{34}$$

This calculation means we only have two second order derivatives (ϕ_{x^2} and ϕ_{y^2}) instead of three, and has very good results in practice. This calculation shows how we can further use the information given by the Eikonal equation for robot navigation.

C. a_n- a centripetal acceleration

This section serves to show that we can separate the normal and tangential accelerations, by showing the path will be independent from the latter.

Considering $v_{field} = (\nabla^2\phi(r)M\nabla\phi(r))$ and β : $\dot{r}^T\cdot v_{field} = v|v_{field}|\cos(\beta)$, we can rewrite a_n as follows:

$$a_n = \begin{cases} v^2\left(-k_{n_c}\gamma + \frac{1}{|\nabla\phi|^2}|v_{field}|cos(\beta)\right) & , cos(\gamma) \geq 0 \\ v^2\big(-k_{n_c}\gamma + \frac{k|\nabla\phi|\cos(\gamma)}{\gamma} + \\ \qquad \frac{1}{|\nabla\phi|^2}|v_{field}|\cos(\beta)\big) & , cos(\gamma) < 0 \end{cases} \tag{35}$$

This leads to the following theorem:

Theorem 3 (Centripetal Acceleration)**.** *Using a_n as defined in eq.(35), the path the robot will take will always be the same, regardless of a_t and v.*

Proof. Using the previous equation to calculate the radius of curvature $R = \frac{v^2}{a_n}$ will give us:

$$\frac{1}{R} = \begin{cases} -k_{n_c}\gamma + \frac{1}{|\nabla\phi|^2}|v_{field}|cos(\beta) & , cos(\gamma) \geq 0 \\ -k_{n_c}\gamma + \frac{k|\nabla\phi|\cos(\gamma)}{\gamma} + \\ \qquad \frac{1}{|\nabla\phi|^2}|v_{field}|\cos(\beta) & , cos(\gamma) < 0 \end{cases} \tag{36}$$

□

Therefore, R is a function of (r, γ, β) and does not depend of a_t or v. This leads to the conclusion that regardless of the robot's velocity, the path will stay the same because it will have, at every point, the same radius of curvature. Higher velocities v will lead to higher normal acceleration. However, because a_n is proportional to v^2, the path stays the same.

We can interpret this normal acceleration as the result of a centripetal force. It is as if the robot is connected to a moving pin by a string. If the robot has higher speeds, the centripetal force applied by the string will increase and the path of the robot will stay the same. This conclusion allows us to define a_t by predicting the robot's path in order to satisfy the robot's input constraints, i.e., to make sure this imaginary string doesn't break, and will be described in the next section.

V. A Feasible Tangential Acceleration Controller

As stated in the previous subsection we can now assume to be able to predict the robot's path by integrating forward in time. This separation of dynamics was also proposed in [19], [20] and [21], being the latter the most related to our work.

A general strategy of defining a_t is going to be presented, in order to respect the acceleration constraints and in the next section we will show how to adapt this solution to guarantee collision avoidance with new obstacles. We also assume the robot cannot revert its direction, as in [21].

Let the constraints for a_n and a_t, along this path's arc length s, be defined as:

$$\left(\frac{a_t}{a_{t_{max}}}\right)^2 + \left(\frac{a_n}{a_{n_{max}}}\right)^2 - 1 \leq 0 \text{ ,for } a_t \geq 0 \tag{37}$$

$$\left(\frac{a_t}{a_{t_{min}}}\right)^2 + \left(\frac{a_n}{a_{n_{max}}}\right)^2 - 1 \leq 0 \text{ ,for } a_t < 0 \tag{38}$$

These equations lead to the following inequality:

$$a_{t_{min}}\sqrt{1-\left(\frac{a_n}{a_{n_{max}}}\right)^2} \leq a_t \leq a_{t_{max}}\sqrt{1-\left(\frac{a_n}{a_{n_{max}}}\right)^2} \tag{39}$$

We now define $v_{crit}(s)$ as the maximum velocity the robot can have at a point so that $-a_{n_{max}} \leq a_n(s) \leq a_{n_{max}}$:

$$v_{crit}(s) = \sqrt{a_{n_{max}} R(s)} \tag{40}$$

Let now $V_i(s)$ be the characteristic constructed by forward integration of the following system, starting from the state (s_i, v_i):

$$\dot{s} = v \ , \ \dot{v} = A_t^- = a_{t_{min}}\sqrt{1-\left(\frac{a_n}{a_{n_{max}}}\right)^2} \tag{41}$$

This characteristic $V_i(s)$ gives us the velocity v along the arc length s if the robot is breaking at maximum capacity from the current position s_i forward, respecting the input constraints. We can write $V_i(s)$ as:

$$\frac{dV_i(s)}{dt} = A_t^-(s) \tag{42}$$

And with $ds = V_i(s)dt$:

$$V_i(s)dV_i(s) = A_t^-(s)ds \tag{43}$$

Integrating both sides gives us:

$$\frac{1}{2}V_i(s)^2 = C + \int_{s_i}^{s} A_t^-(s)ds \Leftrightarrow \tag{44}$$

With the initial condition we are able to find C, leaving the following formula:

$$V_i(s)^2 = v_i^2 + \int_{s_i}^{s} 2A_t^-(s)ds \tag{45}$$

Now, considering a discrete scheme with arc interval Δs:

$$V_i(s)^2 = v_i^2 + \sum_{s_i}^{s} 2A_t^-(s)\Delta s \tag{46}$$

Definition V.1 (Feasible State)**.** *State (s_i, v_i) for which $V_i(s) \leq v_{crit}(s)$, $\forall s \geq s_i$, meaning the robot can follow the trajectory respecting the acceleration constraints as long it breaks from the present state (s_i, v_i) forward, i.e., $a_t(s) = A_t^-(s)$,$\forall s \geq s_i$.*

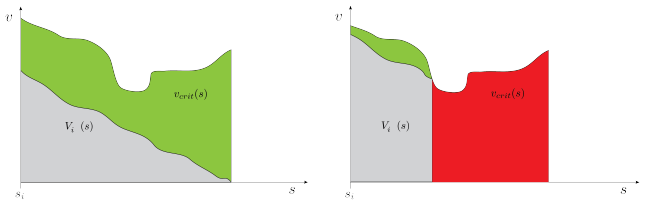
This definition can be observed in fig. 4, where the difference between a feasible state and a infeasible state is showed. In the feasible state scenario we see that if the robot starts breaking at the present state s_i the robot's velocity will always be lower than v_{crit} for the entire path and therefore the predicted trajectory, as explained in the previous section, will be maintained. Otherwise, in a infeasible scenario, a point of no return is reached, and there is no input combination that will allow the robot to follow the predicted path.

Assuming we start from a feasible state (s_0, v_0), if we can always choose $a_t(s_i)$ such that the next state (s_{i+1}, v_{i+1}) is also feasible, we guarantee the robot will always be able to follow the predicted path maintaining the acceleration constraints. This rationale leads to the following algorithm, from now on referenced as feasible tangential controller:

1) For each cycle i
2) Calculate $R(s)$
3) Calculate $v_{crit}(s)$
4) Calculate $V_{i+1}(s)$ with $a_t(s_i) = a_{t_{max}}\sqrt{1-\left(\frac{a_n}{a_{n_{max}}}\right)^2}$
5) Choose $a_t(s_i)$ to be used following:
   a) If there is s_j for which $v_{crit}(s_j) < V_{i+1}(s_j) : a_t(s_i) = A_t^-(s_i)$
   b) Else: $a_t(s_i) = a_{t_{max}}\sqrt{1-\left(\frac{a_n}{a_{n_{max}}}\right)^2}$
6) Return to 1.

As we can see, the robot only accelerates if it leads to a feasible state (s_{i+1}, v_{i+1}). Otherwise, the robot breaks. Therefore, this shows that this algorithm, starting from a feasible state i, always leads to a feasible state $i+1$.

This method can be related to the CDWA scheme, defining a a_n as in the previous section and with only two discretization points. We could test more accelerations other than the maximum acceleration at each point but there would be little improvement in the robots speed and it would have a some



(a) Feasible state (s_i, v_i) (b) Infeasible state (s_i, v_i)

Fig. 4: Showing Feasible vs Infeasible state

computational cost. Therefore, we chose to keep the algorithm simple and fast, still obtaining excellent results, showed in section VII. It could also be shown that this approach, as $\Delta s \to 0$ the results obtained would converge to the optimal-time solution, found in [21].

This method also shows that $v > 0$ for any $t > 0$ because to have $v(s) = 0$ would mean $v_{crit}(s)$ would need to approach 0, which would mean the trajectory would need to have a sharp edge. $v > 0$, $\forall t > 0$, as we mentioned in IV-A, proves that $\dot{L_2} < 0$ and that the robot converges to the goal in finite time.

VI. Obstacle Avoidance

The navigation function is obtained solving the Eikonal equation, with the medium velocity going to zero near the obstacles.

$$|\nabla\phi(x)|P(x) = 1,\ P(x) \to 0 \implies |\nabla\phi(x)| \to +\infty \quad (47)$$

This means that near the obstacles the navigation function will have infinite values and leads to the next theorem:

Theorem 4 (Free-collision path)**.** *Regardless of the initial condition, the a_n controller will define a path that does not intersect any obstacle.*

Proof. As proved in Theorem 2, the Lyapunov function will have a non-positive derivative:

$$\dot{L}_2 \leq 0 \text{ ,for } t > 0 \quad (48)$$

This way, the value of the Lyapunov function will always be bounded by its initial value $L_2(r_0, \gamma_0)$, proving the robot will never reach a position adjacent to an obstacle.

$$L_2(r_{obstacle}, \gamma) = +\infty + L_1(\gamma) > L_2(r_0, \gamma_0) \quad (49)$$

□

Now that we have proven that the a_n controller always defines a free-collision path to the goal, the only question remaining is if the robot can follow this path meeting the acceleration constraints.

As shown in the previous section, as long as the initial state is a feasible state, the robot will be able to follow the path. At this point we have proven the robot never collides with an obstacle contained in the original map. Now the scenario where new obstacles appear is left to study.

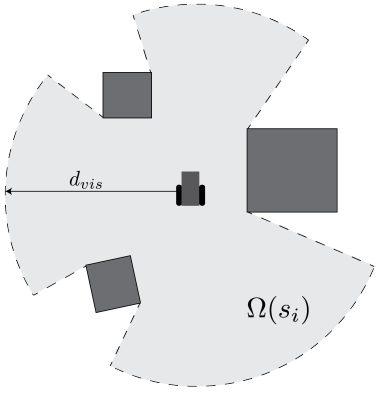
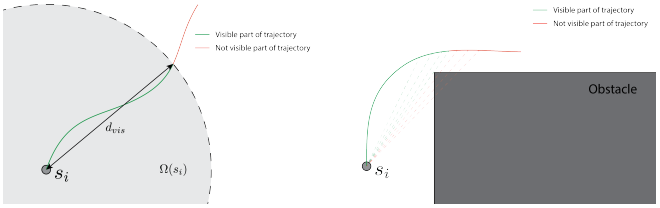


Fig. 5: Visible area - $\Omega(s_i)$



(a) d_{vis} restraint (b) Presence of obstacles

Fig. 6: $\Omega(s_i)$

A. Dynamic Environment- A Safe Tangential Acceleration Controller

In order to prove the robot won't collide with new obstacles it needs to be aware of another of its limitations: visibility. Here, we assume the robot to be using distance measuring sensor(as an Hokuyo Laser Range Finder) that it is able to see accurately at a distance d_{vis} and obviously, the visibility is blocked by obstacles, as we can see in fig. 5. To simplify the scenario, we assume the sensor's angular range is never an issue, although, as will be clear later, it could also be considered.

Definition VI.1 (Visible Area)**.** *The visible area $\Omega(s_i)$ is the area in which no new obstacles appear. Therefore, new obstacles only appear, in a worst case scenario, at the boundary of this visible area.*

For this method, the robot has to be aware of which part of the predicted trajectory is in the visible area. Following the restraints of the sensor, considering the current robot position $r(s_i) = (x(s_i), y(s_i))$, a point s_k of the predicted trajectory is visible,i.e., $s_k \in \Omega(s_i)$, if the straight line between s_i and s_k does not intersect any obstacle and if:

$$\sqrt{(x(s_i) - x(s_k))^2 + (y(s_i) - y(s_k))^2} \leq d_{vis} \quad (50)$$

In fig. 6 we can see two scenarios that clarify the previous statement. We now write $v_{safe}(s_i, s)$ as follows:

$$v_{safe}(s_i, s) = \begin{cases} v_{crit}(s), \text{ if } s \in \Omega(s_i) \\ 0 \text{ , } if s \notin \Omega(s_i) \end{cases} \quad (51)$$

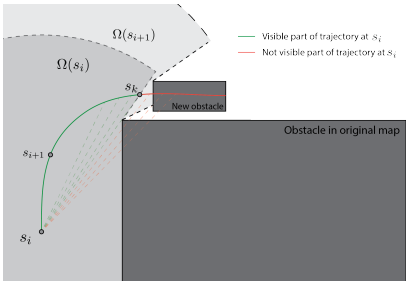


Fig. 7: $\Omega(s_i)$ and $\Omega(s_{i+1})$

This allows the following definition:

Definition VI.2 (Safe State)**.** *State (s_i, v_i) for which $V_i(s) \leq v_{safe}(s)$, $\forall s \geq s_i$, meaning the robot can follow the trajectory and stop in its current visible area $\Omega(s_i)$ respecting the acceleration constraints as long starts breaking at the present state (s_i, v_i), i.e., $a_t(s) = A_t^-(s)$,$\forall s \geq s_i$. Therefore, it is safe in a sense that can stop where we are certain no new object will appear.*

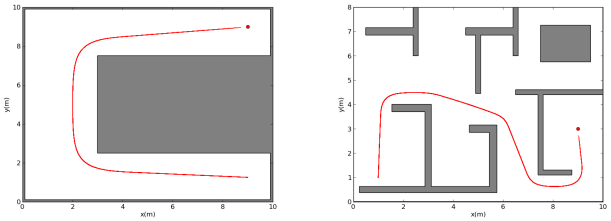
After this definition, it is now possible to see that if the input at s_i is always chosen such that (s_{i+1}, v_{i+1}) is also safe, we prove the robot never collides with new obstacles.

It is now clear that using the algorithm described earlier in this section, if we now chose the $a_t(s_i)$ to be used as $a_t(s_i) = A_t^-(s_i)$ if $\exists s_j : v_{safe}(s_i, s_j) < V_{i+1}(s_j)$, we guarantee the robot is always capable of coming to a complete stop within its current visible area, still following the pre-defined path and meeting its acceleration constraints. This can be seen in fig. 7. In this figure, assuming s_i is safe, the robot can come to a complete stop before s_k and will only accelerate now if it can guarantee that s_{i+1} is also safe, i.e., if it can also stop before reaching s_k. As we can see, we show the robot will then never collide with the obstacle that it becomes visible only at s_{i+1}.

Now that we've established the robot can always stop before hitting new obstacles, the question remaining is what happens to the navigation function $\phi(r)$ when new obstacles appear. What we propose is that it stays the same when new obstacles appear that don't interfere with the pre-established path and that $\phi(r)$ is only updated when it is safe to follow the new navigation function $\phi_{new}(r)$. This solution is presented in the following algorithm, from now on referenced as Safe Tangential Controller:

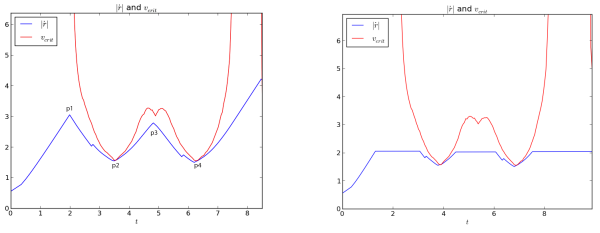
1) For each cycle i
2) If there is new obstacle that intersects trajectory:
   a) If (s_i, v_i) is safe according to $\phi_{new}(r)$:
      i) $\phi(r) = \phi_{new}(r)$
      ii) Go to 3
   b) Else:
      i) $a_t(s_i) = A_t^-(s_i)$
      ii) Go to 1
3) Calculate $R(s)$, $v_{crit}(s)$ and $v_{safe}(s_i, s)$
4) Calculate $V_{i+1}(s)$ with $a_t(s_i) = a_{t_{max}}\sqrt{1 - \left(\frac{a_n}{a_{n_{max}}}\right)^2}$
5) Choose $a_t(s_i)$ to be used following:
   a) If there is s_j for which $v_{safe}(s_i, s_j) < V_{i+1}(s_j)$: $a_t(s_i) = A_t^-(s_i)$
   b) Else: $a_t(s_i) = a_{t_{max}}\sqrt{1 - \left(\frac{a_n}{a_{n_{max}}}\right)^2}$
6) Return to 1.

To best of our knowledge, this is the first algorithm that proves the robot never collides with new obstacles, introducing the concept that the robot is aware of its visibility limits. How the robot reacts in several different scenarios will be shown in section VII.



(a) Map1- $t_f = 9.33s$ (b) Map2- $t_f = 11.19s$

Fig. 8: Obtained path



(a) Map1- $v_{max} = 5m/s$ (b) Map1- $v_{max} = 2m/s$

Fig. 9: Plot of $|\dot{r}|$ and v_{crit} over time with different maximum velocities

VII. Results

In this section we will present the results obtained through simulation of the developed controller. For this we are going to use two different maps, a simple one (fig. 8(a)), from now on referred as Map1, and a more complex one, composed of a higher number of obstacles, Map2 (fig.8(b)). We will show how the robot performs in several scenarios, including the appearance of obstacles not contained in the initial map, in order to validate the developed theory. We will then perform a comparative analysis between the achieved algorithm and DWA. In fig. 8 we also can find the total travelled time (t_f) for both maps with $v_{max} = 6m/s$ and $a_{t_{max}} = a_{n_{max}} = -a_{t_{min}} = 1.5m/s^2$

A. Results for the Feasible Tangential Acceleration controller

In this section we are going to analyze the tangential acceleration controller. For this two scenarios are going to be used: a first one, where the robot's maximum velocity (v_{max}) is $6m/s$ and a second one with $v_{max} = 2m/s$. In both we are using the feasible tangential controller described in section V.

In both this situations the robot performs as expected and we are going to analyze fig. 9(a) to show this. As we can see, from 0 to $p1$ the robot accelerates at maximum capacity taking into account the normal acceleration needed to perform that part of the path. At $p1$, the robot starts to break in order to reach $p2$ at a velocity lower than v_{crit}. Between $p2$ and $p3$ it accelerates again and so fourth. The same rational can also be applied to fig. 9(b) and we can see that this method works also when the robot's velocity limits come into play. In more complicated scenarios, such as in Map2, although v_{crit} follow a much more complicated pattern, we can see that this controller also shows the expected results (fig.10).

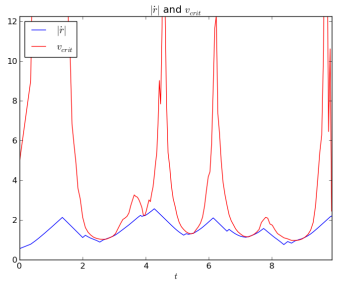


Fig. 10: Map1- $|\dot{r}|$ and v_{crit} over time with $v_{max} = 5m/s$

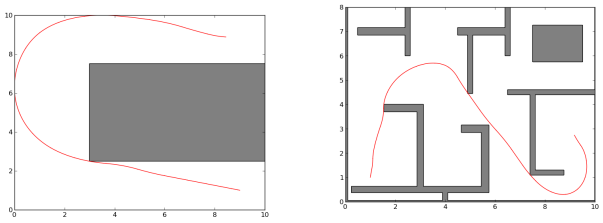
Therefore, the results agree with the theory developed in V, and we can conclude that this controller makes the robot follow the path described by the normal acceleration control, defined in IV, meeting the acceleration contraints.

B. Comparison with DWA

In this section we will present the results obtained with the implementation of CDWA. As stated in section I, we found this to be the most suitable scheme to compare our algorithm with because it presents most of the main characteristics our does: convergence to goal and guarantees obstacle avoidance while respecting input constraints. However, we are going to show that CDWA, in order to prove obstacle avoidance while maintaining the simplicity needed to online implementation, still represents a greedy search and leads to significant overshoot over certain conditions. In fig.11, we can see the results for Map1 and Map2, for a robot with $a_{max} = 1.5m/s^2$ and $v_{max} = 5m/s$. As we can see, both trajectories present high overshoot and exhibit a significant higher total travelled time than the algorithm presented in this paper, with $t_f = 11.5s$ for Map1 and $t_f = 12.5s$ for Map2. However, if we take the same conditions as the ones used in CDWA ($a_{max} = 1.5m/s^2$ and $v_{max} = 1.2m/s$), we get the trajectories showed in fig. 12, which are closer to the ones presented in CDWA. This leads to the conclusion that the efficiency of CDWA depends on the ratio between a_{max} and v_{max}, against our controller which behaves similarly regardless of the conditions used. We believe that this kind of stability and predicability of our controller is one of the main advantages we obtained over previous work.

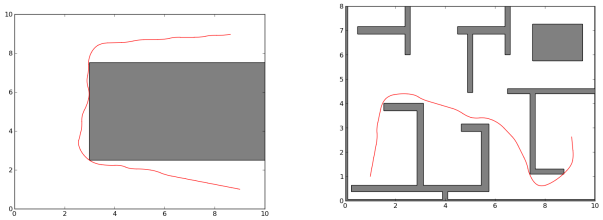
C. Stability

This section serves to show a simple example of how the robot behaves traveling through a narrow passage. As we can see in fig. 13, our algorithm does not exhibit any kind of oscillation, while the CDWA does, refuting the work presented in [5] that stated that oscillations in narrow passages was an inherent limitation potential field methods.



(a) Map1- $t_f = 11.5s$ (b) Map2- $t_f = 12.5s$

Fig. 11: Obtained path with CDWA with $a_{max} = 1.5m/s^2$ and $v_{max} = 5m/s$

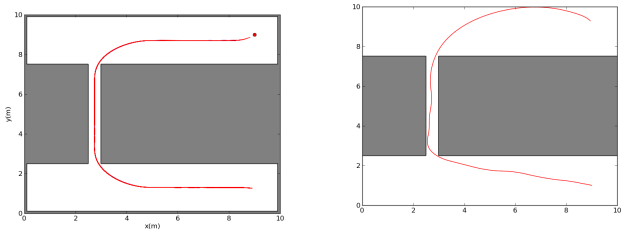


(a) Map1- $t_f = 18s$ (b) Map2- $t_f = 13.5s$

Fig. 12: Obtained path with CDWA with $a_{max} = 1.5m/s^2$ and $v_{max} = 1.2m/s$

D. Obstacle Avoidance

It was already possible to see that no collisions occur, as predicted in theory, with obstacles present in the original map. Therefore, this section serves to show the safe tangential controller at work (section VI-A). First we will show how this implementation affects the robots time performance, even when new obstacles don't appear, because the robot always travels in a safe state, i.e., it is always capable of coming to a complete stop in its current visible area. These results can be seen in table I. As expected, as d_{vis} decreases, the robot needs to have lower velocities in order to ensure a safe state, and therefore the total travelled time t_f is higher.



(a) Our solution- $t_f = 11.29s$ (b) CDWA- $t_f = 12.0s$

Fig. 13: Trajectories obtained in a narrow passage

$d_{vis}\,(m)$	3	2	1
$t_f\,(s)$	8.58	9.13	11.62

TABLE I: How travelled time is affected by changes in d_{vis}

E. New obstacles- non-intersecting trajectory

As we suggested in subsection VI-A, the navigation function $\phi(r)$ is not updated when new obstacles appear that do not intersect the trajectory. As we can see in fig.14 this solutions is reasonable and the robot's trajectory is not affected by these obstacles.

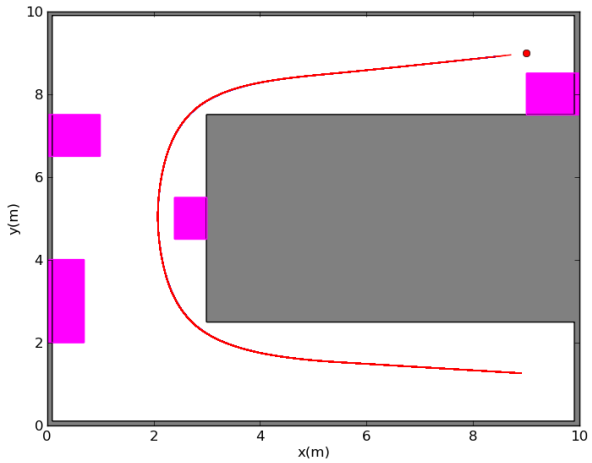


Fig. 14: Robot trajectory when new obstacles appears

1) New obstacles- intersecting trajectory: In this subsection we will show how the robot behaves when its trajectory is intersected with new obstacles, in three different scenarios. In the first scenario, fig. 15 , the new obstacle becomes visible to the robot at $s1$ and, because the obstacle is far, it can safely update the navigation function and the robot avoids the new obstacle with no difficulty.

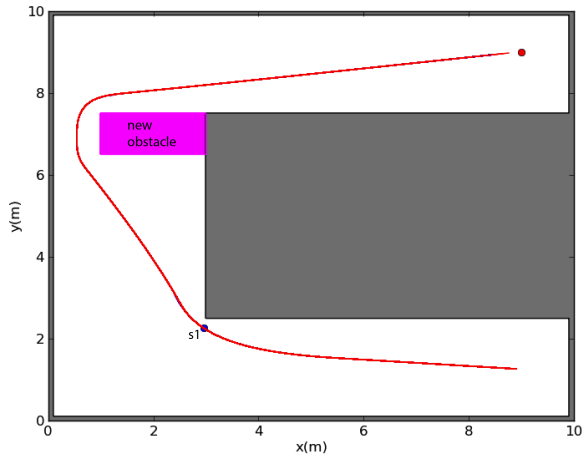
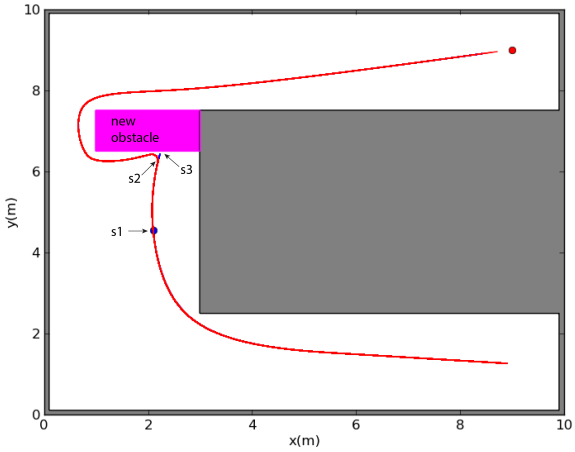
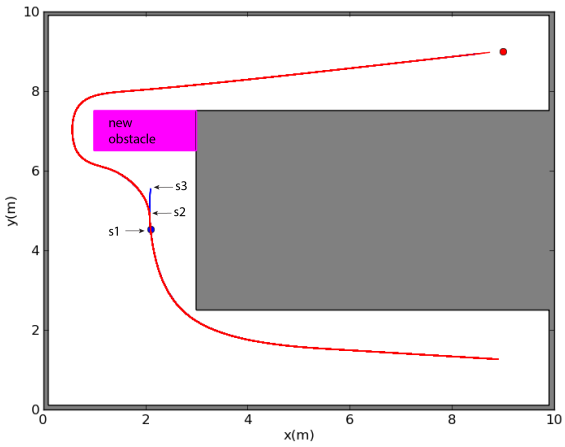


Fig. 15: Robot trajectory when new obstacle appears



(a) $d_{vis} = 2\,m$



(b) $d_{vis} = 1\,m$

Fig. 16: Robot trajectory when new obstacle appears

Let us now observe two more extreme scenarios. In the first one, fig. 16(a), the robot is using a $d_{vis} = 2\,m$, i.e., it trusts its sensor will always detect an obstacle before it is less then two meters away from the robot. The robot then detects the obstacle at $s1$, when it is 2 meters away from the new obstacle. As described in subsection VI-A, the robot starts breaking, always ensuring it can stop until $s3$, before hitting the obstacle. The blue line is the trajectory the robot is calculating at this point. The robot keeps doing so until at $s2$, where the robot has a low enough velocity for it to be able to update the navigation function $\phi(r)$ to $\phi_{new}(r)$ still being in a safe state. The robot then follows the new navigation function, avoiding the new obstacle. As we can see, this result shows the robot does not collide with a new obstacle as predicted in theory, but the trajectory of the robot is not very smooth.

Let us now observe the scenario in fig. 16(b). Here, the robot still detects the obstacle when it is two meters away, but we were now using $d_{vis} = 1\,m$. It is possible to see the same rational as presented above applies, but the robot now switches navigation functions earlier and the result is a much

smoother trajectory. This leads to the conclusion that, when using this algorithm in a real implementation, we advise to use a d_{vis} inferior to the sensors true capacity in order to achieve a smoother trajectory, not close to the robot's limits.

VIII. Conclusion

In this paper, a new way of using Fast Marching Methods for reactive collision avoidance is presented.

The proposed approach shows it is feasible to separate the normal and tangential dynamics.

Then, it was possible to derive a normal acceleration controller that proves the robot's convergence to the goal and the convergence of its trajectory to the gradient lines of the navigation function obtained using FMM.

Afterwards, a tangential controller is designed that allows to follow a defined trajectory meeting the robots acceleration constraints. It is then proved the robot never collides with any obstacles contained in the map at initialization. Afterwards, a safe implementation of a tangential controller is shown, in order to also prove collision avoidance with new obstacles, not contained in the original map. This was done by introducing the notion that the robot is aware of its own visibility limits and is always able to stop where it is certain no new obstacles appear.

The proposed solution is then compared with a Dynamic Window Approach implementation (CDWA), and it is shown that CDWA leads to trajectories with more overshoot and with higher total times to complete the trajectory to the goal position.

It is also shown our solution does not present oscillations in narrow passages, which is a common problem of Artificial Potential Methods.

Therefore, we conclude this to be a promissing new alternative to DWA and it would be interesting to study how this solution can be extended to robot formations.

References

[1] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[2] J. Minguez and L. Montano, "Nearness Diagram (ND) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

[3] J. R. Andrews and N. Hogan, "Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator," *Control of Manufacturing Processes and Robotic Systems , Eds. Hardt, D. E. and Book, W., ASME, Boston*, 1983.

[4] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." *IEEE International Conference on Robotics and Automation, St. Louis, Missouri*, 1985.

[5] Y. Koren, Y. Koren, S. Member, S. Member, J. Borenstein, J. Borenstein, A. Arbor, and A. Arbor, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *Proceedings of the IEEE*, pp. 1398–1404, 1991.

[6] J. Sethian, "A Marching Level Set Method for Monotonically Advancing Fronts," *Proc. Nat. Acad. Sci.*, 1996.

[7] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, no. July 2015, 2006.

[8] S. Garrido, L. Moreno, and P. U. Lima, "Robot formation motion planning using Fast Marching," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 675–683, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2011.05.011

[9] J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," 1992.

[10] Y. Wang and G. S. Chirikjian, "A new potential field method for robot path planning," pp. 977–982, 2000.

[11] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.

[12] L. N. Ogren, P., "A convergent dynamic window approach to obstacle avoidance," *IEEE Trans. Robot.*, vol. 21, p. 188195, 2005.

[13] J. a. Sethian, *Fast Marching Methods and Level Set Methods*, 1998.

[14] R. Kimmel and J. Sethian, "Fast Marching Methods for Robotic Navigation with Constraints," *Center for Pure and Applied Mathematics Report, Univ. of California, Berkeley*, 1998.

[15] J. a. Sethian, "Fast Marching Methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[16] J. R. T. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 6, pp. 933–941, 2003.

[17] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, 1999.

[18] H. Pina, *Métodos Numéricos*. Escolar Editora, 2010, pp. 155–157.

[19] V. D. C. D. Gadola, M. and L. Manzo, "A Tool for Lap Time Simulation," no. September 2015.

[20] M. Lepetič, G. Klančar, I. Škrjanc, D. Matko, and B. Potočnik, "Time optimal path planning considering acceleration limits," *Robotics and Autonomous Systems*, vol. 45, no. 3-4, pp. 199–210, 2003.

[21] E. Velenis and P. Tsiotras, "Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation," *Journal of Optimization Theory and Applications*, vol. 138, no. 2, pp. 275–296, 2008.