Decentralized & prioritized Navigation and Collision Avoidance for Multiple Mobile Robots

Giannis Roussos and Kostas J. Kyriakopoulos

Abstract We present an algorithm for decentralised navigation of multiple mobile robots. Completely decentralised Navigation functions build a potential field for each robot that is employed in a feedback control law. The potential field incorporates limited sensing and explicit prioritisation. A non-circular sensing area creates asymmetrical sensing by reducing the influence of robots and obstacles behind each robot, introducing implicit priorities resembling "rules of the road". Static and moving obstacles are taken into account, as well as malfunctioning robots. A decentralised feedback control law based on the gradient of the potential field ensures convergence and collision avoidance for all robots, while respecting a lower speed bound. Simulation results demonstrate the efficacy of the proposed algorithm.

1 Introduction

Decentralised navigation has become popular in a wide variety of robotic applications involving multiple mobile robots, while it is also being investigated from the point of view of multi-agent systems. In most multi-robot applications an increased level of decentralisation is desired to allow for greater performance, flexibility and computational efficiency. Moreover, a properly decentralised approach can offer some level of robustness with respect to single robot failures, limiting their effect on the rest of the robots.

A wide variety of methods for navigation has emerged, employing various techniques. One class of methods handles the problem in a two step approach [8]: the workspace is initially divided into cells, which are then used to formulate the navigation problem as a graph search problem. Artificial potential or vector fields guide the robots between cells, following the sequence provided by the graph search. An

Giannis Roussos, Kostas Kyriakopoulos
Control Systems Lab, School of Mechanical Engineering, National Technical University of Athens, 9 Heroon Polytechniou Street, Zografou 15780, Greece. e-mail: {jrous, kkyria}@mail.ntua.gr

extension to multi-robot navigation is presented in [2]. Although this approach is intuitive, it requires considerable pre-calculations and thus a-priori knowledge. Moreover, performing the cell decomposition in the combined state space of all robots and solving the graph search problem can become challenging for many robots.

Another class of methods uses artificial potential fields [6] to directly derive feedback controllers steering the robots over the entire workspace. A common weakness of these methods is the appearance of local minima away from the goal that can prevent convergence. A special class of potential fields, Navigation Functions (NFs) [7], can ensure the existence of a single, global minimum. The NF methodology has been developed for a wide class problems, offering formal performance guarantees and computational efficiency, while its real-time feedback nature can compensate for measuring and modeling errors. NFs have been applied to multi-agent problems ranging from robotic navigation [3] to Air Traffic Control (ATC) applications [11].

In this paper we further refine the concept of limited sensing in the proven NF methodology, which combined with a feedback control law designed on the principles of [11] yields a completely decentralised solution for multi-robot navigation and collision avoidance in a workspace with obstacles. Our approach requires no a-priori computation or knowledge and does not rely at all on centralised controllers. The only information that each robot needs is its position within the workspace and knowledge about other robots and obstacles within a sensing area around it. Thus, our algorithm is completely distributed and its computational cost does not depend on the total number of robots.

The construction of the potential fields incorporates priorities, both in explicit and implicit form. The former is achieved by assigning priority classes to the robots and allowing high priority ones right of way. Moving and static obstacles are assigned the highest priority. Moving obstacles have been also considered in [1], but their motion is assumed to be known a-priori, as the algorithm pre-calculates the complete trajectories of the robots. Malfunctioning robots can also be treated as moving obstacles, thus offering some fault tolerance. Implicit priorities resembling "rules of the road" are introduced by using a non-circular sensing area, so that the potential of each robot is mostly influenced by robots and obstacles in front of it.

The rest of this paper is organised as follows: Section 2 defines the problem considered, followed by Section 3 where the proposed potential field is described. In Section 4 the feedback control scheme is presented and simulation results are given in Section 5. The conclusions of the paper are summarized in Section 6.

2 Problem Statement

We assume a scenario involving N mobile robots modeled as kinematic unicycles:

$$\dot{\boldsymbol{q}}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \mathbf{J}_i \cdot u_i,$$
$$\dot{\phi}_i = \omega_i,$$

where $\boldsymbol{q}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^\top$ is the position vector of robot i with respect to a global frame $\mathscr{E}$, ϕ_i its heading angle, i.e. the angle between the robot's longitudinal axis and the global x axis, and $\boldsymbol{J}_i = \begin{bmatrix} \cos(\phi_i) & \sin(\phi_i) \end{bmatrix}^\top$. Each circular robot i of radius r_i is driven via the linear velocity u_i and the angular velocity ω_i. For the linear motion a desired speed u_{di} is assumed, acting as a lower bound for the absolute linear velocity. All robots are operating within a common workspace around the origin of $\mathscr{E}$ with radius R_w[1], while the information available to each of them is restricted to other robots and/or obstacles within its sensing area $\mathscr{A}_i$.

The objective here is to drive each robot i to its destination $\boldsymbol{q}_{di}$ while avoiding all collisions. We want to enforce some form of prioritization, so that robots with high priority can maintain right of way versus lower priority ones. Our aim is a completely decentralised solution that will also consider static and moving obstacles.

3 Completely Decentralised Navigation Functions

Decentralisation in the Navigation Functions (NFs) methodology has been introduced by allowing each robot to ignore the targets of other agents and navigate independently using its own NF-generated potential field. Limited sensing is a key factor for decentralisation: it takes into account the finite range of real sensors and greatly limits the information that each robot needs to acquire and process, significantly improving the applicability and scalability of the algorithm in large scenarios. In [3] limited sensing range has been introduced in NFs in a C^0 fashion, assuming a priori knowledge of the total number of agents. This requirement has been eliminated in [4], where a switching sensing graph is used, resulting in a hybrid system. However, this approach does not ensure global stability, as blocking situations may be reached. Thus, convergence occurs only if the switching of the sensing graph eventually stops. A completely locally computable NF has been presented in [9], but only for single-agent problems and with the assumption that at each time instant there is at maximum one visible obstacle. This effectively means that the algorithm solves one collision at a time, which is not optimal in a multi-agent scenario.

A completely decentralised scheme for a NF has been presented by the authors in [10], incorporating limited sensing range and explicit priorities in an absolutely locally computable potential field that can take into account multiple robots according to their priorities, as well as static and moving obstacles. The work presented here further develops this concept, by using non-circular sensing areas for each robot (see Figure 1a). Each robot can use its full sensor range in the forward direction in order to acquire as much information as possible to plan its trajectory, while the effective sensing range is reduced in the rear direction. Such a sensing scheme introduces implicit prioritisation, as it can create situation with "asymmetrical" sensing between robots (see Figure 1b). In graph theory terms, this means that the communication

[1] In the case of a non-circular workspace the algorithm presented here can still be applied by employing an appropriate transformation to a spherical workspace, as shown in [12]

graph is no longer undirected. Moreover, the shape of the sensing area improves the computational efficiency of the algorithm, as it allows a finer selection of the neighboring obstacles and robots that contribute to the potential field.

The potential field presented here can be combined with a control scheme similar to the one in [11] to provide decentralised, non-cooperative navigation for multiple robots. In fact, any controller that can ensure a decreasing rate for the potential's value over time is applicable. Thus, the use of the potential field presented here is not limited to unicycle-like robots but can also be applied to other types of kinematics (holonomic or non-holonomic), if combined with an appropriate control scheme.

The decentralised Navigation Function (NF) we use is of the form [5]:

$$\Phi_i = \frac{\gamma_i + f_i}{\left((\gamma_i + f_i)^k + G_i \cdot \beta_i\right)^{1/k}}, \tag{1}$$

where γ_i is the target function, f_i the cooperation function, G_i the obstacle function and finally β_i is the workspace boundary function. The potential Φ_i attains its maximum value of 1 on the boundary of collisions and has a single minimum of 0 at the destination. Our contribution focuses in the construction of the obstacle function G_i, where we incorporate the non-circular sensing scheme presented in section 3.2, which combined with the priority classes described in section 3.1 allows the use of both explicit and implicit prioritisation between the robots.

3.1 Priority classes

Explicit prioritisation is introduced in the Navigation Functions (NFs) algorithm by assigning each robot i, $i \in \{1,\ldots,N\}$ a priority class $c_i \in \mathbb{N}$. Lower values of c_i represent higher priority robots, with $c_i = 0$ denoting either uncontrolled or malfunctioning robots, or obstacles (stationary or moving). The assignment of priorities can be performed independently of the navigation algorithm presented here, based on task classification, robot capabilities, etc. We define the *threat set* T_i of robot i as the set of all obstacles and robots of the same or higher priority class, i.e. with the same or lower c_i: $T_i \triangleq \left\{ j \in \{1,.,N\} \setminus \{i\} \,\middle|\, c_j \leq c_i \right\}$. Priorities define the sensing relations between neighboring robots: each robot i ignores any other robots that belong to lower priority classes, i.e. any robot j with $c_j > c_i$, and only considers robots and obstacles belonging to its threat set T_i. Thus, robots performing high priority tasks can maintain right of way, while lower priority robots steer around them.

Moving and static obstacles are handled like uncontrolled robots; they are assigned the maximum priority class, $c_i = 0$ and are avoided by all normally operating robots. Moreover, if a robot i is known to experience a degradation of its navigation and collision avoidance capabilities it is assigned the highest priority class $c_i = 0$ and is treated as an obstacle by other robots[2]. This classification scheme means that

[2] Online priority reassignment is outside the scope of this work, as it is assumed that it will be handled by an independent fault detection system

two robots i and j have mutual sensing between them, i.e. they both take each other into account to navigate, $i \in T_j$ and $j \in T_i$, if and only if $c_i = c_j \neq 0$, i.e. they belong to the same priority class, other than the highest one. Otherwise, if one of the robots, say i, belongs to a higher priority class (even the highest one), $0 \leq c_i < c_j$, then $i \in T_j$ but $j \notin T_i$. Thus, at all combinations of c_i, c_j where at least one of them is nonzero, i.e. $max(c_i, c_j) > 0$, there is at least one-way sensing between robots i and j. This ensures that all collisions are avoided, at least by one of the two involved robots. Of course, in the unfortunate case that 2 robots i and j malfunction simultaneously, i.e. $c_i = c_j = 0$, any collisions between them can not be avoided, though all other normally operating robots will still manoeuvre around both of them.

3.2 Limited sensing

The effective sensing area used by each robot consists of a semicircle of radius R_{sr} in the rear semi-plane and a semi-ellipse with semimajor and semiminor axes R_{sf}, R_{sr} (with $R_{sf} > R_{sr}$) respectively in the forward semiplane, as shown in Figure 1a. Range R_{sf} should be less or equal to the maximum range allowed by the robot's sensors. The boundary of the sensing area around the robot is then given by:

$$R_s(\theta) = \begin{cases} \frac{R_{sr} R_{sf}}{\sqrt{(R_{sr}\cos(\theta))^2 + \left(R_{sf}\sin(\theta)\right)^2}}, & \theta \in (-\frac{\pi}{2}, \frac{\pi}{2}) \\ R_{sr}, & otherwise \end{cases} \tag{2}$$

Angle $\theta \in (-\pi, \pi]$ is measured from the forward direction of the robot, see Fig. 1a.

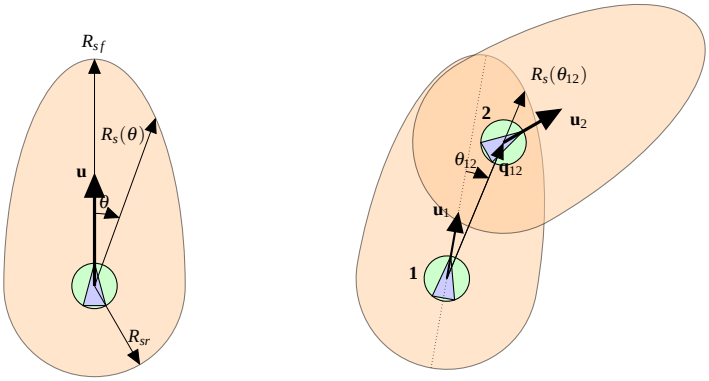


Fig. 1: Left: The non-circular sensing area used for each robot, consisting of a semicircle in the rear and a semi-ellipse in the front of the robot.
Right: Implicit prioritisation of robot 2 with respect to robot 1: Robot 1 senses robot 2 but is outside robot 2's sensing area, thus only robot 1 will manoeuvre.

Similarly, for each neighbor j of robot i we define the bearing angle θ_{ij} between the relative position vector $\boldsymbol{q}_{ij} = \boldsymbol{q}_j - \boldsymbol{q}_i$ and the forward direction of robot i, see Fig. 1b. The effective sensing range of robot i in the direction of $\boldsymbol{q}_{ij}$ is $R_s(\theta_{ij})$, using (2). In the special case that $R_{sf} = R_{sr}$ the sensing zone becomes a circle, as in [10]. The elliptical shape offers a simple way for an adjustable forward sensing range in a C^1 fashion, though other C^1 curves may be used if required by specific applications.

The contribution of robot j to the potential field of robot i is based on the basic obstacle function $\hat{g}_{ji}$, which is defined as in previous NF approaches:

$$\hat{g}_{ij} = \left|\left|\boldsymbol{q}_{ij}\right|\right|^2 - r_{ij}^2 \tag{3}$$

where $r_{ij} \triangleq r_i + r_j$. By the above definition, $\hat{g}_{ij}$ is zero when robot j touches robot i, i.e. when $\left|\left|\boldsymbol{q}_{ij}\right|\right| = r_{ij}$, and increases as the robots move away from each other.

Each robot can sense other robots or obstacles that are inside the above sensing area, i.e. whenever $\left|\left|\boldsymbol{q}_{ij}\right|\right| \leq R_s(\theta_{ij})$. The effective sensing range $R_s(\theta_{ij})$ is used, as presented in [10], to derive the normalised obstacle function $\bar{g}_{ij}$:

$$\bar{g}_{ij} = \frac{\hat{g}_{ij}}{R_s(\theta_{ij})^2 - r_{ij}^2} \tag{4}$$

Finally, the contribution g_{ij} of robot (or obstacle) j to robot i's potential is derived:

$$g_{ij} = \begin{cases} L(\bar{g}_{ij}), & \left|\left|\boldsymbol{q}_{ij}\right|\right| \leq R_s(\theta_{ij}) \\ 1, & \left|\left|\boldsymbol{q}_{ij}\right|\right| > R_s(\theta_{ij}) \end{cases} \tag{5}$$

where the shaping function $L(x)$ is $L(x) = x^3 - 3x^2 + 3x$, chosen to satisfy:

$$L(0) = 0 \qquad L(1) = 1$$
$$L'(x) > 0 \ \forall x \in [0,1) \qquad L'(1) = L''(1) = 0$$

By the above definition, g_{ij} is zero when robots i and j collide, i.e. $\left|\left|\boldsymbol{q}_{ij}\right|\right| = r_{ij}$ and increases up to 1 at the boundary of the sensing area, i.e. when $\left|\left|\boldsymbol{q}_{ij}\right|\right| = R_s(\theta_{ij})$. Outside the sensing area of robot i, g_{ij} is constantly 1. Using the above properties of $L(x)$ it can be verified that g_{ij} is by construction C^2 in the interior of the free space, i.e. away from collisions, where $\hat{g}_{ij} \in [0, +\infty)$. This allows the potential Φ_i to be C^2, as it is required for it to be a Navigation Function [7]. Function $g_{ij} = g_{ij}\left(\left|\left|\boldsymbol{q}_{ij}\right|\right|\right)$ is plotted in Figure 2. Since g_{ij} is constantly 1 when $\left|\left|\boldsymbol{q}_{ij}\right|\right| \geq R_s(\theta_{ij})$, each robot i is only affected by other robots $j \in T_i$ inside its sensing area. It should be noted here that although $\hat{g}_{ij} = \hat{g}_{ji}$, the nondimensional functions g_{ij} and g_{ji} are not equal in general, since θ_{ij} and θ_{ji} are different. This difference between g_{ij} and g_{ji} introduces the asymmetrical sensing in the construction of the potential fields of robots i and j.

The complete obstacle function G_i is then constructed: $G_i = \prod_{j \in T_i} g_{ij}$. The priority classes defined in 3.1 are used here to allow robot i to ignore j when $c_i < c_j$, forcing robot j to manoeuver around i. Thus, only knowledge of those robots in T_i that are

within the sensing area of i is required: $G_i = \prod_{j \in \tilde{T}_i} g_{ij}$, where the "close threat" set $\tilde{T}_i = \left\{ j \in T_i \,\middle|\, ||\boldsymbol{q}_{ij}|| < R_s(\theta_{ij}) \right\} \subset T_i$ comprises threats in the sensing area of i.

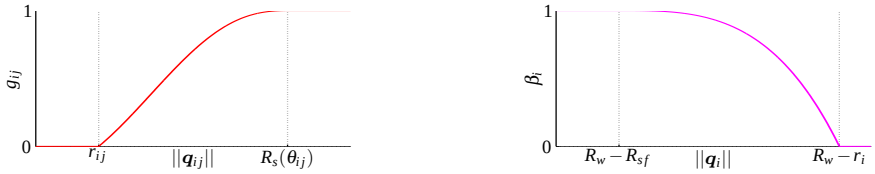


Fig. 2: Left: Obstacle function g_{ij} with respect to distance $||\boldsymbol{q}_{ij}||$ between robots i and j
Right: Workspace boundary function β_i with respect to $||\boldsymbol{q}_i||$

Similarly to g_{ij}, β_i is designed to contain the effect of the workspace boundary in a zone of width R_{sf}. The normalised workspace boundary function $\bar{\beta}_i$ is based on its dimensional counterpart $\hat{\beta}_i$:

$$\bar{\beta}_i = \frac{\hat{\beta}_i}{(R_w - r_i)^2 - (R_w - R_{sf})^2}, \qquad \hat{\beta}_i = (R_w - r_i)^2 - ||\boldsymbol{q}_i||^2$$

The effective boundary function β_i used in Φ_i is then defined similarly to g_{ij}:

$$\beta_i = \begin{cases} L(\bar{\beta}_i) & ||\boldsymbol{q}_i|| \geq R_w - R_{sf} \\ 1, & ||\boldsymbol{q}_i|| < R_w - R_{sf} \end{cases} \tag{6}$$

β_i becomes zero when robot i touches the workspace boundary, i.e. $||\boldsymbol{q}_i|| = R_w - r_i$, and varies in a C^2 fashion to exactly 1 when robot i is at a distance equal to or higher than R_{sf} away from the boundary, i.e. when $||\boldsymbol{q}_i|| \leq R_w - R_{sf}$, see Figure 2.

3.3 Potential Construction

For the target function γ_i we use the following nondimensional form:

$$\gamma_i = \frac{||\boldsymbol{q}_i - \boldsymbol{q}_{di}||^2}{R_w^2} \tag{7}$$

Since the largest distance between any $\boldsymbol{q}_i$, $\boldsymbol{q}_{di}$ inside the spherical workspace of radius R_w is $2R_w$, γ_i is equal to or lower than 4 for any combination of $\boldsymbol{q}_i$, $\boldsymbol{q}_{di}$.

The cooperation function f_i is used here as in [3]:

$$f_i(G_i) = \begin{cases} a_0 + \Sigma_{l=1}^{3} a_l G_i^l, & G_i \leq X \\ 0, & G_i > X \end{cases} \tag{8}$$

where $a_0 = Y$, $a_1 = 0$, $a_2 = \frac{-3Y}{X^2}$, $a_3 = \frac{2Y}{X^3}$ and X, Y are positive parameters. X sets a threshold for G_i, so that the cooperation function f_i is activated when $G_i < X$. Parameter Y defines the maximum value of f_i, attained when $G_i = 0$.

The final result of using the above defined G_i, β_i and γ_i in (1) for a setup with 3 obstacles is shown in Figure 3. The target $\boldsymbol{q}_{di}$ is set in the center of the workspace and 3 obstacles are included. The potential field is shown over the entire workspace, along with the values of G_i, β_i, γ_i and Φ_i along the positive x axis, that crosses through the center of one of the obstacles that is placed between the target and the workspace boundary. In this example we have assumed that the cooperation function f_i is not activated, i.e. $f_i = 0$ everywhere. Since we cannot plot the potential for all possible robot orientations, we have used $R_s(\theta_{ij}) = R_{sf}$ everywhere for simplicity in the figure. As the figure demonstrates, G_i and β_i become less than 1 only within the sensing range R_{sf} of the obstacle and workspace boundary, respectively. The dotted blue line shows the value of Φ_i for $G_i = \beta_i = 1$ everywhere, i.e. without the effect of any obstacles or the workspace boundary. As expected, this coincides with the actual Φ_i outside the sensing area of obstacles and the workspace boundary.

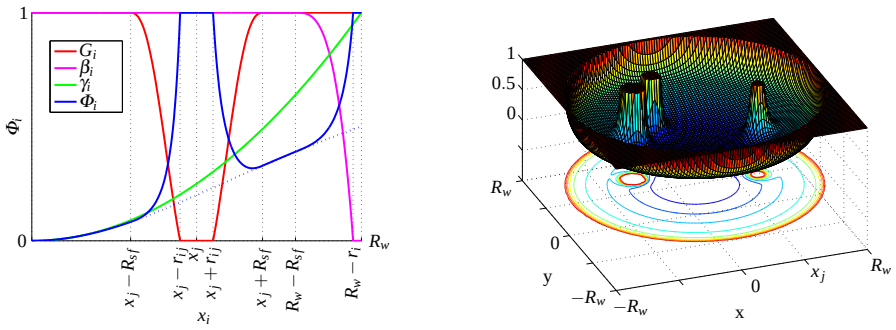


Fig. 3: Left: Obstacle function G_i, workspace boundary function β_i, target function γ_i and the resulting Navigation Function Φ_i on the line $y_i = 0$, $x_i \in [0, Rw]$.
Right: Navigation Function potential field in a workspace with 3 obstacles and local sensing.

It has been shown in [10] that the potential field constructed above is a Navigation Function, providing almost global navigation and collision avoidance for all values of k higher than a finite lower bound k_0. Moreover, since in the construction presented here only robots and obstacles inside the non-circular sensing area $\mathscr{A}_i$ affect Φ_i, the number of g_{ij} that contribute to Φ_i at any time is significantly reduced with respect to [10] and previous approaches with global sensing. This significantly boosts the computational efficiency of the algorithm, especially in scenarios involving many robots. Simulation experience with NFs indicates that the minimum value of the exponent k required to eliminate local minima and render (1) a NF increases with the number of contributing robots and obstacles. Thus, the exponent k needed for the potential presented here is in most cases lower than the one required in [4].

4 Completely Decentralised Navigation

The proven navigation properties of the potential field Φ_i described above can be used to drive each robot to its destination while avoiding collisions. In fact, any controller that can maintain a decreasing rate for each potential field Φ_i, i.e. $\dot{\Phi}_i < 0$ can be employed in combination with the potential field presented previously to stabilise each robot to its target, while also avoiding collisions. Such a controller has been presented in [11] for unicycle-like vehicles moving in 3D space . We can derive a similar controller for planar unicycles by neglecting the vertical velocity input. The resulting control scheme employs the projection of the gradient $\nabla_i \Phi_i = \begin{bmatrix} \Phi_{ix} & \Phi_{iy} \end{bmatrix}^\top$ on robot's i longitudinal (heading) direction: $P_i = \boldsymbol{J}_i^\top \cdot \nabla_i \Phi_i$ where $\boldsymbol{J}_i = \begin{bmatrix} \cos(\phi_i) & \sin(\phi_i) \end{bmatrix}^\top$. Moreover, we use the partial derivative $\frac{\partial \Phi_i}{\partial t}$, which sums the effect of all but the i^{th} robots' motion on Φ_i:

$$\frac{\partial \Phi_i}{\partial t} = \sum_{j \neq i} \nabla_j \Phi_i^\top \cdot \boldsymbol{J}_j u_j$$

where $\nabla_j \Phi_i = \frac{\partial \Phi_i}{\partial \boldsymbol{q}_j}$ is the gradient of Φ_i with respect to $\boldsymbol{q}_j$.

The proposed control law for the linear velocity u_i is:

$$u_i = \begin{cases} -\operatorname{sgn}(P_i) U_i, & \frac{\partial \Phi_i}{\partial t} \leq U_i \left(|P_i| - \varepsilon \right) \\ -\operatorname{sgn}(P_i) \frac{U_i \varepsilon + \frac{\partial \Phi_i}{\partial t}}{|P_i|}, & \frac{\partial \Phi_i}{\partial t} > U_i \left(|P_i| - \varepsilon \right) \end{cases} \tag{9a}$$

where U_i is the nominal velocity:

$$U_i = \begin{cases} u_{di}, & ||\boldsymbol{q}_i - \boldsymbol{q}_{di}|| > d_i \\ \frac{||\boldsymbol{q}_i - \boldsymbol{q}_{di}||}{d_i} \cdot u_{di}, & ||\boldsymbol{q}_i - \boldsymbol{q}_{di}|| \leq d_i \end{cases} \quad \text{and} \quad \operatorname{sgn}(x) \triangleq \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0. \end{cases}$$

A small positive constant ε is used to ensure a decreasing rate of Φ_i and $\operatorname{sgn}(x)$ is:

$$\operatorname{sgn}(x) \triangleq \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0. \end{cases}$$

U_i matches identically the reference signal u_{di} away from the target $\boldsymbol{q}_{di}$ and reduces continuously to 0 inside a ball of radius d_i around $\boldsymbol{q}_{di}$. The angular velocity used is:

$$\omega_i = \begin{cases} 0, & M_i \geq \varepsilon_\phi \\ \Omega_i \cdot \left(1 - \frac{M_i}{\varepsilon_\phi} \right), & 0 < M_i < \varepsilon_\phi \\ \Omega_i, & M_i \leq 0, \end{cases} \tag{10}$$

where: $M_i \triangleq \dot{\phi}_{\boldsymbol{nh}_i} \left(\phi_i - \phi_{\boldsymbol{nh}_i} \right)$, $\Omega_i \triangleq -k_\phi \left(\phi_i - \phi_{\boldsymbol{nh}_i} \right) + \dot{\phi}_{\boldsymbol{nh}_i}$. The *nonholonomic* heading angle $\phi_{\mathbf{nh}i}$ represents the heading of $\operatorname{sgn}(p_i) \nabla_i \Phi_i$:

$$\phi_{\mathbf{nh}i} \triangleq atan2\left(\mathrm{sgn}\left(p_i\right)\Phi_{iy}, \mathrm{sgn}\left(p_i\right)\Phi_{ix}\right), \tag{11}$$

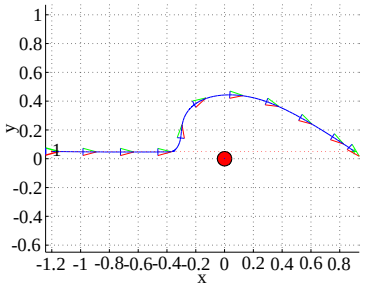
where: $\mathrm{atan2}(y,x) \triangleq \arg(x,y), \quad (x,y) \in \mathbb{C}$ and $p_i = \mathbf{J}_{di}^{\top} \cdot (\boldsymbol{n}_{i1} - \boldsymbol{n}_{i1d})$ is the position vector with respect to the destination, projected on the longitudinal axis of the desired orientation. Thus, $\mathrm{sgn}(p_i)$ is equal to 1 in front of the target configuration and -1 behind it. Finally, ε_ϕ is a small positive constant and k_ϕ a positive gain.

The principles of this control scheme can be found in detail in [11]. The stability analysis presented there does not rely on the specific Navigation Function used and can be also applied to the algorithm here to formally guarantee convergence and collision avoidance.

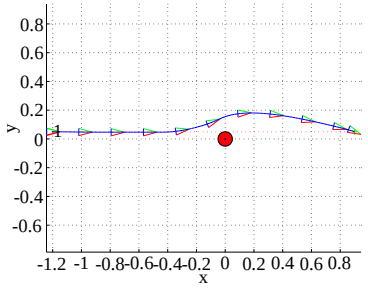
5 Simulation Results

In order to demonstrate the effect of the noncircular sensing area to the performance of the algorithm we present simulation results below. The first simulation scenario is a simple example with a robot navigating around one static obstacle. Although not a challenging scenario, this can give a clear view of the performance and efficiency improvements that our algorithm achieves compared to the circular sensing scheme in [10]. A maximum sensing range of 0.5 length unit is assumed, so for the new sensing we have used $R_{sr} = 0.15$, $R_{sf} = 0.5$, i.e. the full sensor range is exploited in the forward direction, but only 30% of it in the rear. Results from the same example using circular sensing are included, using two different sensing ranges, $R_s = 0.5$ and $R_s = 0.15$. The resulting paths are shown in Figures 4a-4c, along with statistical information in Table 4d. Compared to the full sensor range in 4a, the new sensing scheme path in 4b is less conservative. In both 4a and 4b cases the robot starts turning at around the same position, near $x = -0.3$, as the forward sensing range is the same. However, the significantly shorter sensing range to the sides and rear of the robot with the new sensing scheme results in a much smaller deviation from the straight line path. Using a reduced circular sensing of radius 0.15 in 4c results in a more aggressive turn as the robot starts maneuvering later, and eventually covers longer distance to reach the target. The improvements of the algorithm presented here are reflected in the total length of the paths shown in table 4d, as well as the computation time, because of the reduced interaction between the robot and the obstacle. Finally the total absolute turning angle $A = \int |\omega|\, dt$ is reduced, as less maneuvering is used.

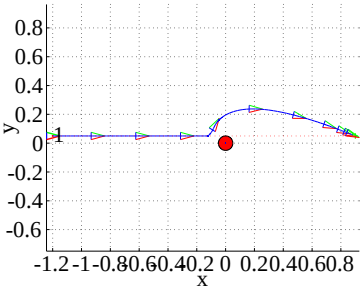
The second simulation example is a multirobot scenario similar to the one used in [10]: 4 low priority robots are moving in parallel, while a high priority one is crossing their paths. Results are presented in Figure 6a for a circular sensing $R_s = 0.5$ and in 6b for the noncircular sensing scheme with $R_{sr} = 0.15$ and $R_{sf} = 0.5$. As noted on the figures, the new algorithm results in much smaller deviations, allowing the robots to reach their targets quicker.



(a) Circular sensing with $R_{sr} = R_{sf} = 0.5$



(b) Noncircular sensing with $R_{sr} = 0.15$, $R_{sf} = 0.5$



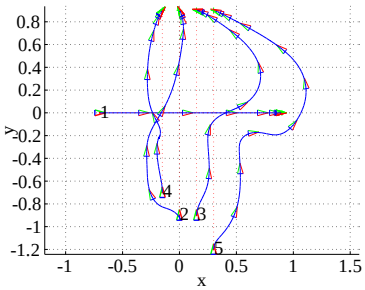
(c) Circular sensing with $R_{sr} = R_{sf} = 0.15$

Sensing Scheme	(a)	(b)	(c)
Computation Time	25*sec*	20*sec*	26*sec*
Path Length	7.56	5.13	6.59
Total steering angle	6.11	3.66	5.00

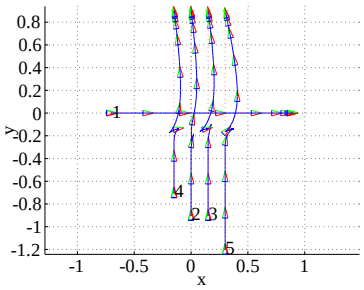
(d)

Fig. 4: Simulation results: Obstacle avoidance using various sensing schemes

Fig. 5: Simulation Results: A high priority robot crosses the paths of 4 lower priority ones moving in parallel



(a) Circular sensing, $R_s = 0.5$. Path length: 23.89



(b) Noncircular sensing, $R_{sf} = 0.5$, $R_{sr} = 0.15$. Path length: 20.9

6 Conclusions

We have presented an algorithm for multi-robot navigation and collision avoidance using the Navigation Functions framework. A novel sensing scheme is implemented

in the method, allowing implicit prioritisation in a rules-of-the-road fashion. Explicit prioritisation is also taken into account, as well static or moving obstacles and uncontrollable robots. Simulation results show significant performance and efficiency improvements with respect to previous work using the NF framework.

7 Acknowledgements

The authors of this paper want to acknowledge the contribution of the European Commission through project iFLY.

References

1. Ahmadzadeh, A., Motee, N., Jadbabaie, A., Pappas, G.: Multi-vehicle path planning in dynamically changing environments. In: 2009 IEEE international conference on Robotics and Automation, pp. 2148–2153. IEEE Press (2009)
2. Ayanian, N., Kumar, V.: Decentralized feedback controllers for multi-agent teams in environments with obstacles. In: IEEE Int. Conf. on Robotics and Automation, pp. 1936–1941 (2008). DOI 10.1109/ROBOT.2008.4543490
3. Dimarogonas, D.V., Kyriakopoulos, K.J.: Decentralized navigation functions for multiple robotic agents with limited sensing capabilities. Journal of Intelligent and Robotic Systems **48**(3), 411–433 (2007)
4. Dimarogonas, D.V., Kyriakopoulos, K.J., Theodorakatos, D.: Totally distributed motion control of sphere world multi-agent systems using decentralized navigation functions. 2006 IEEE International Conference on Robotics and Automation pp. 2430–2435 (2006)
5. Dimarogonas, D.V., Loizou, S.G., Kyriakopoulos, K.J., Zavlanos, M.M.: A feedback stabilization and collision avoidance scheme for multiple independent non-point agents. Automatica **42**(2), 229–243 (2006)
6. Khatib, O.: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research **5**(1), 90–98 (1986). DOI 10.1177/027836498600500106
7. Koditschek, D., Rimon, E.: Robot navigation functions on manifolds with boundary. Advances in Applied Mathematics **11**(4), 412–442 (1990)
8. Lindemann, S., Valle, S.L.: Smoothly blending vector fields for global robot navigation. In: 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005. Omnipress (2005)
9. Lionis, G., Papageorgiou, X., Kyriakopoulos, K.J.: Locally computable navigation functions for sphere worlds. Proceedings of the 2007 IEEE International Conference on Robotics and Automation pp. 1998–2003 (2007)
10. Roussos, G., Kyriakopoulos, K.J.: Completely decentralised navigation of multiple unicycle agents with prioritization and fault tolerance. In: 49^{th} IEEE Conference on Decision and Control (2010). To appear
11. Roussos, G., Kyriakopoulos, K.J.: Decentralised navigation and collision avoidance for aircraft in 3D space. 2010 American Control Conference, Baltimore, USA (2010)
12. Tanner, H.G., Loizou, S., Kyriakopoulos, K.J.: Nonholonomic navigation and control of cooperating mobile manipulators. IEEE Transactions on Robotics and Automation **19**(1), 53–64 (2003)