Implementation of Robot Formation Control and Navigation Using Real-Time Panel Method

Abdel-Razzak MERHEB, Yunus ATAS, Veysel GAZI, and Nilay SEZER-UZOL

***Abstract*— In this paper, potential flow calculations for incompressible inviscid flows are used to develop a collision free navigation algorithm for a robot swarm. Robots are equiped with laser scanners which are used to detect the obstacles in the environment. Then, a real-time panel method is used to calculate the streamlines of the potential flow around the complex shaped rigid objects, providing the robots with safe trajectories to the target. The swarm of robots is also forced to keep a desired formation during navigation using potential functions. Potential flow algorithms provide navigation with smooth paths to the target. The algorithm can be used in dynamic environments in real-time as the changes in the medium are detected.**

I. Introduction

Potential flows have inherent properties such as smooth streamlines which do not cross each others or the objects in the environment. Moreover, they do not suffer from the local minima problem to the same extend as the other potential functions methods. Therefore, they are suitable for collision free path planning and navigation in robots, ground and marine vehicles, or unmanned air vehicles (UAV's). Realizing this potential, there have been recent investigations using potential flows. A method that uses stream functions to produce paths convenient for an aircraft-like vehicle is developed in [1]. They provide analytical solution for the case in which there is a single stationary or moving circular obstacle in the environment. In the case of multiple circular obstacles since analytical solution is not tractable, they treat each obstacle separately. The algorithm is tested on RoboFlag/Robocup experimental platform showing promising results. In [2] Sullivan et al. use similar procedure to develop a navigation algorithm for a group of flying UAV's. In addition to the stream functions they use virtual spring forces to maintain a predefined formation for the robot group. In [3] Ye et al. discuss a similar procedure for a swarm of robots. The coordinated motion control for the robot swarm is guaranteed by using artificial inter-agent forces, while safe navigation is provided using stream functions augmented with repulsive potential functions based obstacle avoidance. Moreover, they discuss the stagnation point problem and a hydrodynamics based analytical solution is provided. In a subsequent work in [4], they extend the work of [3] and consider a more realistic connectivity algorithm with different singular association rules and test it using simulation. In addition, a testbed for future experimental implementations is discussed.

To the best of our knowledge the first work using panel method for robot navigation is the paper of Kim and Khosla [5]. There, the authors first build an artificial potential field using the flowlines found using panel method with harmonic functions. The authors use real shaped obstacles discretized with panels and sink or source singularities are placed at each panel to express the rigid boundary of the obstacles. The stagnation point problem is also stated and defined, and two solutions are suggested. After that, a control strategy for collision-free navigation of a nonpoint robot or a manipulator in this artificial field is developed. Also, two solutions are stated for the structural local minimum problem that occurs in mobile robots and manipulators unable to be considered as point robots. Another pioneering study on using potential flows as well as the panel method for robot or UAV navigation is the work by Zhang and Valavanis [6], [7]. In [6], panel method is used to generate a collision free path for mobile robots navigating in an uncertain obstacle filled 2-D environment. To minimize the complexity they fit simple larger convex polygons to the obstacles (which are not necessarily in the same shape as the obstacle). The panel method is applied to calculate the flow lines to form safe trajectories for the robots to reach the target. They consider also a dynamic environment in which at each instance the panel method is used to solve for the flow lines based on the data sensed from the environment and generate a sequence of way-points. The algorithm is tested using numerical simulations to generate the discrete way-points and no implementation or consideration of low-level agent dynamics is done. In [7] the algorithm is extended to 3-D space. There, they transform arbitrary shaped obstacles using 24 simple convex polyhedrons to simpler shaped ones to simplify computational complexity.

In a recent study, Uzol et al. [8] use panel method to develop a target tracking methodology for a swarm of UAV's using unsteady streamline patterns in real environments. They use the map of a real city (from Google maps) but design the panels manually off-line. Obstacle avoidance is guaranteed by following the streamlines generated, while collision avoidance between the agents is provided by setting a repulsive singularity element at each agent. The algorithm is tested using numerical simulations.

In the above studies only [1] considers implementa-

tion/testing on real robotic systems. Howeveri as mentioned above, they consider only circular obstacles. Similarly, the works in [5], [6], [7], and [8], which consider complex shaped obstacles and panel method do not consider implementation on real robotic system.

This paper combines the ideas from the above studies to implement the potential flow concept on real robotic system composed of differentially driven non-holonomic robots. The robots navigate in an unknown environment and use their laser scanners for sensing and obstacle detection. Based on the sensed data the panel method is used to solve in real-time for the flow lines to generate safe path to the target. When the robots encounter new obstacles (which they were unable to detect before because they were too far or the obstacles were occluded by other objects/obstacles), they re-solve for the streamlines in real-time to re-plan their paths. Moreover, we use potential functions in order to keep cohesiveness or to achieve and keep a predefined formation in the group during the motion. The algorithm is implemented in a laboratory environment with real robots and real complex shaped obstacles. This work is build on an earlier work in [9] where implementation was performed on a different robotic setup under the assumption that the map of the environment is known a priori.

II. Panel Methods

To solve the potential flow problem around rigid bodies, numerical methods are usually used since the flow around complicated geometries are often impossible to solve analytically. Panel methods are one of these techniques which can be applied around objects with complex shapes. In this section, we briefly review the panel method based on the treatements in [10] and [11].

The potential field induced by an object placed in a flow can be found by solving the system of equations araised from the boundary condition constraints [10]. Rigid objects plunged into a flow with freestream velocity $\overrightarrow{\mathbf{Q}}$ prevent the flow from passing through their boundaries. The potential flow is always tangent to the boundary [11]

$$\overrightarrow{\mathbf{Q}} \cdot \overrightarrow{\mathbf{n}} = 0 \tag{1}$$

where $\overrightarrow{\mathbf{n}}$ is the normal to the surface [10]. To use the potential flow for navigation, we consider an unknown environment with a freestream velocity $\overrightarrow{\mathbf{Q}}_{inf}$ and a sink placed at the target resulting in an attracting velocity $\overrightarrow{\mathbf{q}}_{sink}$. In such environment, the velocity of the flow is formed by the freestream velocity, the induced velocity of present obstacles $\overrightarrow{\mathbf{q}}_{induced}$, and the velocity due to the effect of the sink. The tangency boundary condition is then given by

$$(\overrightarrow{\mathbf{Q}}_{inf} + \overrightarrow{\mathbf{q}}_{sink} + \overrightarrow{\mathbf{q}}_{induced}) \cdot \overrightarrow{\mathbf{n}} = 0 \tag{2}$$

or

$$-\overrightarrow{\mathbf{q}}_{induced} \cdot \overrightarrow{\mathbf{n}} = (\overrightarrow{\mathbf{Q}}_{inf} + \overrightarrow{\mathbf{q}}_{sink}) \cdot \overrightarrow{\mathbf{n}} \tag{3}$$

Note that if the environment has other components such as other source and/or sink points, their effects would be added to the right hand side of (3). To apply the panel method, we start by discretizing the obstacles using panels. A vortex is then placed at the middle of each panel, and the boundary condition is applied at this point. The problem is then reduced to finding the strengths of the singularity elements placed at each panel which satisfy the boundary condition in (3). It is then easy to find the induced velocities known as the effect of the presence of the obstacles at any point in the space. The induced velocities along with the freestream velocity and the velocity due to the sink form the flow velocity at any coordinate, thus the velocity the robot has to apply for safe navigation.

For 2-D applications as performed in this study, the boundaries of the complex shaped geometries are discretized into straight line segments as panels. The point vortex elements are used as singularity elements and placed at the mid point of each segments which is also the control point of the panel. The velocity at a point (x, y) induced by a point vortex with strength Γ_k located at (x_k, y_k) is

$$\overrightarrow{\mathbf{q}}_{induced} = u_{induced}\,\overrightarrow{\boldsymbol{i}} + v_{induced}\,\overrightarrow{\boldsymbol{j}} \tag{4}$$

where u and v are the x and y components of the induced velocity given by

$$u_{induced} = \frac{\Gamma_k}{2\pi} \frac{y - y_k}{(x - x_k)^2 + (y - y_k)^2} \tag{5}$$

$$v_{induced} = -\frac{\Gamma_k}{2\pi} \frac{x - x_k}{(x - x_k)^2 + (y - y_k)^2} \tag{6}$$

The velocity components at a point (x, y) induced by the sink point placed at the target of coordinates (x_f, y_f) with strength Γ_{sink} are given by

$$u_{sink} = -\frac{\Gamma_{sink}}{2\pi} \frac{x - x_f}{(x - x_f)^2 + (y - y_f)^2} \tag{7}$$

$$v_{sink} = -\frac{\Gamma_{sink}}{2\pi} \frac{y - y_f}{(x - x_f)^2 + (y - y_f)^2} \tag{8}$$

Then, equation (3) can be written as

$$-\begin{bmatrix} u_{induced} \\ v_{induced} \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \end{bmatrix} = \left(\begin{bmatrix} U_{inf} \\ V_{inf} \end{bmatrix} + \begin{bmatrix} u_{sink} \\ v_{sink} \end{bmatrix} \right) \cdot \begin{bmatrix} n_x \\ n_y \end{bmatrix} \tag{9}$$

where, U_{inf} and V_{inf} are the x and y components of the freestream velocity, and n_x and n_y are the x and y components of the normal vector $\overrightarrow{\mathbf{n}}$, respectively.

Assuming that there are N panels in the environment, equation (9) is applied to calculate the induced velocities at the control points of the panels (these velocities must be tangencial to the surface). Considering the effect of all the other panels results in a system of linear equations in the form

$$-\begin{bmatrix} K_{11} & \dots & K_{1N} \\ \vdots & & \vdots \\ K_{N1} & \dots & K_{NN} \end{bmatrix} \cdot \begin{bmatrix} \Gamma_1 \\ \vdots \\ \Gamma_N \end{bmatrix} = \begin{bmatrix} RHS_1 \\ \vdots \\ RHS_N \end{bmatrix} \tag{10}$$

where K_{ij} is the effect of the vortex placed at panel j on the control point of panel i, Γ_i is the strength of the singularity element placed at panel i, and RHS_i is the effect of both

the sink point placed in the target and the free stream flow on panel *i*.

After solving the systems of equations, the vortex strengths are used by the robot to calculate the velocity that take it safely through the obstacles to reach the target as

$$u_{robot} = U_{inf} + u_{sink} + \sum_{i=1}^{N} u_{induced} \tag{11}$$

$$v_{robot} = V_{inf} + v_{sink} + \sum_{i=1}^{N} v_{induced} \tag{12}$$

where *N* is the total number of panels and $\sum u_{induced}$ and $\sum v_{induced}$ are the *x* and *y* coordinates of the total induced velocities by all panels at the location point of the robot, respectively.

The advantage of the panel methods arises from the fact that they use the boundary condition (no flow across the surface of any solid object) applied on the panels after the discretization of the complex objects. This means that the calculations will be restricted only on the boundaries of the obstacles, rather than on the whole flow field. This means simplicity and speed, which are very important factors in robot navigation algorithms. Given these advantages, one should also note that the panel method is a discretization/approximation of a continuous flow. To obtain better approximation one needs to use higher number of panels. However, As the number of panels increases the computational cost of the method increases as well.

III. Group Cohesiveness and Formation Control

In this article in addition to the flow lines for navigation potential functions for inter-agent interactions will also be used. Traditionally, potential functions have also been used for robot path planning and navigation. In the past decade, researchers started applying them also for inter-agent interactions in swarms and multi-robot systems. To achieve cohesiveness, an inter-agent potential function must be designed such that it is attractive for far-range inter-agent distances. However, in the mean time in order to avoid inter-agent collisions it must be repulsive for short-range distances (see for example [12]). Let us denote the position of agent i at time t with $p_i(t)$, and let $p^\top = \left[p_1^\top, \ldots, p_M^\top\right]$, where *M* is the number of agents in the swarm. A potential function which satisfies the above properties is

$$J(p) = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} a_{ij} \left[ln\left(\|p_{ij}(t)\|\right) + \frac{d_{ij}}{\|p_{ij}(t)\|^2} \right] \tag{13}$$

where $p_{ij}(t) = p_i(t) - p_j(t)$ is the vector connecting the positions of robots *i* and *j*, a_{ij} is a positive gain used to weight the potential function between the two robots, and d_{ij} is the distance at which the attraction and repulsion forces between agents *i* and *j* balance each other.

If only aggregation and cohesiveness is desired one can choose $d_{ij} = d$ for all pairs (i, j) and for some $d > 0$. In the case of formation control d_{ij} represent the desired inter-agent distances in the geometric formation. By appropriately choosing the values of d_{ij} it is possible to achieve any desired geometric shape. However, note that the potential functions methods suffer from the local minima problem. In other words, with potential functions as the one above, in general, it is not possible to guarantee that the desired formation will be achieved from any initial agent position and only local results can be obtained.

IV. Robot Control

Consider an application where a robot swarm has to reach a target point (the coordinates of which are known) while avoiding collisions in an unknown environment. Besides reaching the target safely, the robots are also expected to keep cohesive or to form and keep a predefined formation during navigation. Panel method and potential functions are used repectively to find the safe streamlines to the target and hold formation. The robots are asked to reach the target point while the environment is totally unknown for them and contains obstacles with complex shapes. With the relative position of the target given, the robots have to discover the surrounding, find the obstacles, apply the panel method, and use the calculated streamlines along with the formation potential function to move. The algorithm is applied in real-time and is repeated continuously while the robots move in the environment. As new obstacles are detected, the panel method is applied again and the navigation paths are updated.

The swarm used is consisting of non-holonomic robots moving in 2-D. Each agent/robot has the dynamics

$$\dot{x}_i(t) = v_i(t)\cos(\theta_i(t)) \tag{14}$$

$$\dot{y}_i(t) = v_i(t)\sin(\theta_i(t)) \tag{15}$$

$$\dot{\theta}_i(t) = w_i(t) \tag{16}$$

where $x_i(t)$ and $y_i(t)$ are the Cartesian coordinates components of the position $p_i(t) = [x_i(t), y_i(t)]^\top$, and $\theta_i(t)$ is the steering angle of the i^{th} agent at time *t*. The control inputs of this agent are the linear speed $v_i(t)$ and the angular speed $\omega_i(t)$.

The algorithm we currently consider is a leader-based algorithm. However, other strategies are also possible. In the current setting, the leader uses the streamlines to navigate while two follower robots pursue it trying to hold the predefined formation. After calculating the velocity at its position using the streamlines, the leader uses the x and y components of the calculated velocity to determine the control input as

$$v_l(t) = \|[v, u]^\top\| \tag{17}$$

$$\omega_l = -K_l mod((\theta_l - \theta_{ld} + \pi, 2\pi) - \pi) \tag{18}$$

where, u and v are the x and y components of the velocity of the flow at the position of the leader robot, K_l is a positive proportional gain, θ_l is the actual orientation of the leader robot, and θ_{ld} is its desired orientation found by

$$\theta_{ld} = \arctan\left(\frac{v}{u}\right)(mod\ 2\pi) \tag{19}$$

In order for the follower robots to achieve the desired shape while navigating to the target, they are required to

follow the sum of the negative gradient of the potential function and the flowline velocity

$$\dot{p}_i(t) = \begin{bmatrix} \dot{x}_i(t) \\ \dot{y}_i(t) \end{bmatrix} = G_i(p) = \begin{bmatrix} u_{if} - \nabla J_{x_i}(p) \\ v_{if} - \nabla J_{y_i}(p) \end{bmatrix} \quad (20)$$

where, $\nabla J_{x_i}(p)$ and $\nabla J_{x_i}(p)$ are the x and y components of the gradient of the potential function, and u_{if} and v_{if} are the x and y components of the velocity of the flow at the position of robot i. The control inputs of the follower robots are calculated as

$$v_i(t) = \|G_i(p)\| \quad (21)$$

$$\omega_i = -K_i mod((\theta_i - \theta_{id} + \pi, 2\pi) - \pi) \quad (22)$$

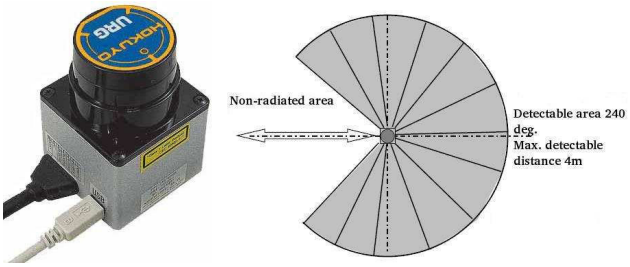
where, $K_i > 0$ is a proportional gain, and the desired orientation of the robots is found by

$$\theta_{id} = \arctan\left(\frac{G_{yi}(p)}{G_{xi}(p)}\right)(mod\ 2\pi) \quad (23)$$

V. Experimental setup

The experiments are performed using a set-up consisting of a 240×340 cm arena with walls of height 15 cm, three KheperaIII robots, and a desktop computer. KheperaIII robots are mini mobile robots equipped with 9 infrared sensors used in our application for obstacle avoidance, 5 ultrasonic sensors, and two DC motors with mounted incremental encoders. The control unit of the robots is the KoreBot LE platform based on a 400 MHz processor with 32 MBytes flash memory and 64 MBytes RAM and a Linux kernel 2.6 as operating system. The robots communicate with each others and with a computer using a wireless LAN IEEE 802.1 compact flash card.

In order to discover the environment and detect obstacles, Hokuyo URG-04LX laser scanner shown in Fig. 1 is integrated to the KheperaIII robots of the swarm. The compact size ($50mm \times 50mm \times 70mm$), light weight, high performance (range up to 4m and 240^o scan with 0.36^o resolution) and low power consumption (2.5W) of the Hokuyo URG-04LX laser sensor make it useful for mini-robot applications. More information on the integration of the laser scanner on the robots can be found in [13]. With 240^o scanned with 0.36^o resolution, the laser scanner provides us with data of 682 samples. However, in order to decrease the computational complexity we use only the scanned data with angle of 45^o towards the target point, i.e. the robot reads and evaluates only the data with a 45^o perspective in the direction of the target. This is intuitive since the robot aims to move always towards the target and has to find any obstacle that interfere its path. Note that this is not a general requirement. Instead it is a practical choice we made in the given small environment and robots with low computational power. We would like to emphasize here that the computational complexity of panel method is directly related to the number of panels used since the dimensions of the system of equations in (10), which needs to be solved in realtime, depends on the number of panels. For example, we have tested experimentally that it takes the KheperaIII robots with 400 MHz processor and 64 MB of RAM 0.69 sec. to solve (10) given 56 panels,



(a) URG-04LX laser sensor (b) Detection range

Fig. 1. Hokuyo URG-04LX laser scanner.

whereas it takes them 3.385 sec. given 120 panels. Given robots with more extensive computational capabilities or a dedicated processor for implementing the panel method all the data from the lasers can be used (and is recommended to be used). In fact, the panel method provides a numerical solution which is an approximation of the analytical solution. Taking very small number of panels can result in inconsistent solution and there is a trade-off between the accuracy and the speed of the solution. Therefore, the above choice, i.e., using only data within 45^o degrees towards the target point, is an intermediate practical solution which decreases the computational complexity and allows us to implement the algorithm with the available hardware. However, in large environments densely populated with obstacles, it might be needed to use all the data from the lasers. In order to implement the potential function based formation control strategy the robots need to know their distances to the other robots. Currently they are not equipped with capabilities which can distinguish the other robots from the obstacles[1]. Therefore, in the implementations here the robots use their odometry information to calculate their global position data and exchange this information using TCP/IP via a wireless ethernet with other robots to calculate the inter-agent distances. This results in implementation imperfections and inaccuracies due to odometry errors and communication delays. Such errors will be avoided in systems with more accurate sensing capabilities. Still the results obtained are good enough to illustrate the procedure.

The algorithm starts with the leader scanning the environment and detecting the obstacles within the 45^o region in its front. It then solves the system of equations in (10) to find the vortex strength vector, and sends the followers the obstacles data along with the strength vector. Once all the elements of the swarm have the obstacles information and the strength vector they use them to generate the flowlines using the panel method. Then, all the robots use these flowlines during navigation to avoid obstacles and reach the goal point safely. While navigating towards the target, the robots exchange their positions via wireless ethernet and apply the potential function to maintain a predefined distance between

[1]

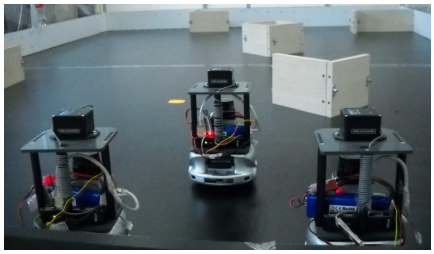


Fig. 2. KheperaIII Robots at the experimental arena.

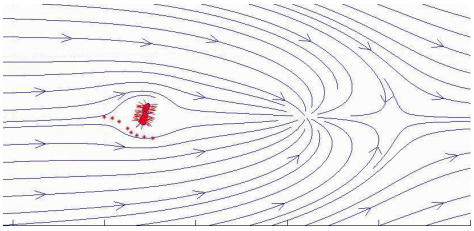
each other. Thus, the formation of the swarm is ensured while it is approaching the target.

VI. Experimental Results

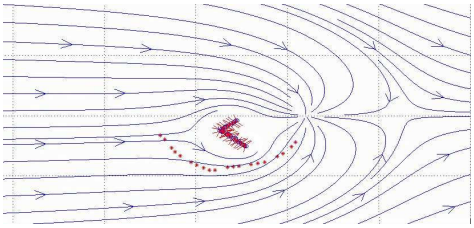
Three KheperaIII robots are used to implement the algorithm: one leader and two followers. The leader robot scans the surrounding using its laser scanner, detects the obstacles and applies the panel method. As was mentioned above, in order to decrease the computational complexity the data only within 45^o between the robot and the target is used for panel calculations[2]. Moreover, in order to further simplify the computations scans are performed once in several steps.

The robots are expected to form an isosceles triangle with equal sides of 35 cm between the leader and each follower, and a side of 30 cm between the two follower robots. While the gain of the artificial potential function used to regulate the distance between the follower robots and the leader is assigned as $a_{ij} = 4$, the gain of the function between the follower robots is $a_{ij} = 1$. Moreover, the gain of the angular speed input of the leader robot is set as $K_l = 1$, while the gain of the angular speed input of the follower robots is $K_i = 0.225$. These gains were selected by trial and error and it is not claimed that they are the best values. Fig. 2 shows the agents of the swarm placed into the arena and ready to start navigation. First we perform an experiment with one robot and two obstacles. Fig. 3(a) shows the flow lines around the first obstacle initially seen by the robot. By following the closest line, the robot ensures its safe navigation until it passes the obstacle. Note that at this position the robot is unable to detect the second obstacle located behind the first obstacle and out of the sensor range[3]. After the robot passes the first obstacle it detects the second obstacle and recalculates the flow lines which become as shown in Fig. 3(b). Similarly, after passing the second obstacle the flow lines become as shown in Fig. 3(c). The dots in these figures show the path of the robot in its motion from its initial position to the target position.

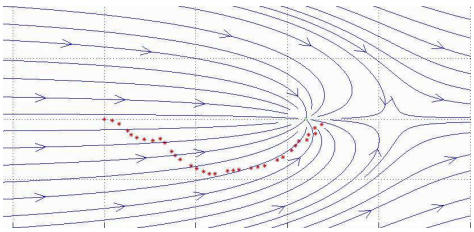
Fig. 4 and Fig. 5 show the results for two different experiments. The plots in Fig. 4 are obtained by using the odometry data and the sensor readings collected by



(a) Path of the robot while avoiding the first obstacle



(b) The second obstacle is detected and being avoided

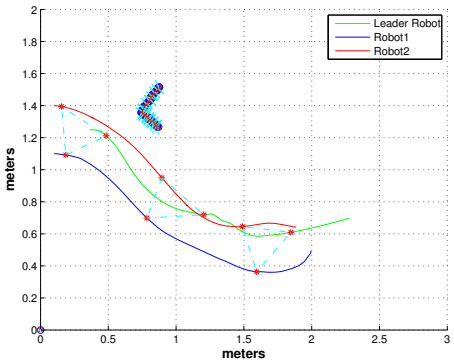


(c) After avoiding the second obstacle (no obstacle detected)

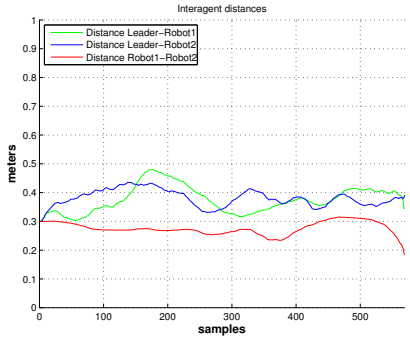
Fig. 3. Path of a robot with two obstacles.

the leader, whereas those in Fig. 5 are obtained using an overhead camera installed for monitoring purposes. Fig. 4(a) shows the paths of the agents as well as the shape of the formation at particular instances of time, whereas Fig. 4(b) depicts the inter-agent distances with respect to the sample number. As can be seen from the figures the desired triangular shape is preserved during navigation with small error. The main reason for the seen error is the fact that the inter-agent distances are not measured directly and are computed from the data communicated which is effected negatively by the odometry errors and the delays in communication. Note that sample reception time changes non-homogenously from 0.007 seconds to 0.5 seconds. Incorporating relative inter-agent sensing will improve the performance. Note also that Fig. 4(a) represents the real motion of the robots in the arena and the dimensions of the axes are in meters. The total time of one experiment is about 150 sec. and the average speed of the robots is 0.016 m/sec. (130 sec. for a total of 2.4 m traveled and 162.5 sec. for 2.67 m).

In Fig. 5(a) one can see the paths of the robots drawn on an partial image of the arena obtained from an overhead camera. These paths are extracted by postprocessing a video of a recorded experiment. Fig. 5(b) shows the corresponding inter-agent distances. One can see once more that the interagent distances are preserved.

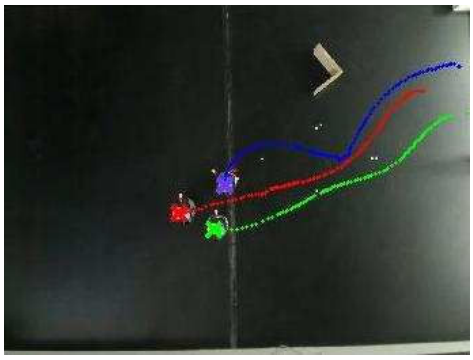


(a) Paths of the robots

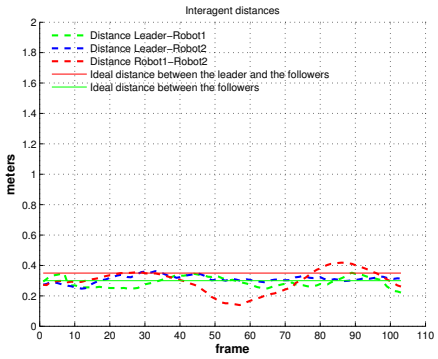


(b) Inter-agent distances

Fig. 4. Paths and inter-agent distances.



(a) Paths of the robots



(b) Inter-agent distances

Fig. 5. Paths and inter-agent distances.

VII. Conclusions And Future Works

In this paper, the idea of using potential flow field calculated by the panel method in real-time for robot navigation is presented in real experimental setup. We use a leader-based strategy for navigating the swarm of robots. Potential functions are also used to set the inter-agent forces acting between the robots resulting in the formation a predefined geometrical shape. While the leader uses streamlines to find its way to the target, the follower robots update their positions so the general form of the swarm is not broken. The algorithm is tested successfully in laboratory environment giving optimistic results. To overcome the increasing computational complexity in densely populated environments one can use separate dedicated processor for panel calculations. The algorithm can be used in dynamic environments as well.

Acknowledgments

The authors would like to thank Andaç T. Şamiloğlu and Engin Karataş for their help in processing the video images.