Multi-Agent Gathering With Collision Avoidance and a Minimax Distance Criterion - Efficient Algorithms and Hardware Realization

Bhaskar Vundurthy, Student Member, IEEE and K. Sridharan, Senior Member, IEEE

***Abstract*—Multiple autonomous agents working cooperatively have contributed to the development of robust large-scale systems. While substantial work has been done in manufacturing and domestic environments, a key consideration for small hardware agents engaged in collaborative factory automation and welfare support systems is limited area and power on-board. When the agents attempt to meet for performing a task, it is natural for them to encounter obstacles and it is desirable for each agent to optimize its resources during its navigation. In this paper, we develop efficient geometric algorithms to find a point, termed as the *gathering point* (and denoted by P_G), for the agents that *minimizes the maximum of path lengths*. In particular, we present an $O(n \log_2 n)$ time algorithm for calculation of P_G for an environment with two agents and n static polygonal obstacles. We then use the notion of a *weighted minimax point* to derive an efficient algorithm (with complexity of $O(k^2 + kn \log_2 n)$) for computing P_G for an environment with k agents and n obstacles. An enhancement to a dynamic environment is then presented. We also present details of an efficient hardware realization of the algorithms. Each agent, equipped with only an ATmega328P microcontroller and no external memory, executes the algorithms. Experiments with multiple agents navigating amidst static as well as dynamic obstacles are reported.**

Keywords: Cyber-physical systems, Agent Gathering, Minimax Criterion, Obstacles, Computational Geometry, Efficient Algorithms, Hardware Realization

I. Introduction

Multi-agent systems constitute an important component of cyber-physical systems and play a key role in industrial automation [1]. They assist humans to perform a variety of tasks including pick and place, and exchange of parts for facilitating manufacturing. The agents operate in the presence of different types of machines for cutting, grinding and other purposes [2].

Welfare support systems [3], [4] represent an important domain for multi-agent collaboration. Typical workplaces include individual homes and public nursing institutions. Multiple agents may be engaged to meet and exchange food items, cutlery etc. This is particularly receiving increased emphasis in view of the need for appropriate implementation of aging-in-place [5]. Here too, the agents negotiate various objects such as chairs and tables as they endeavour to meet.

Industrial as well as domestic environments also involve moving objects (besides static ones) such as humans and utility carts that the agents need to handle suitably. Multi-agent technologies comprise of software-based agents (such as those based on the Java Agent Development Framework) [2], hardware agents [6] or a combination of both. Hardware agents may consist of robot arms [1], mobile robots [3], [6] or other autonomous systems. The work described in this paper is on hardware agents and in particular, autonomous mobile robots. Mobile robots operate with power constraints and it is desirable for each robotic agent to operate for several hours without recharging (of the batteries on-board). This goal can be related to the 'travel' distance for each robot before they meet (to exhange supplies). The objectives of this work are therefore as follows.

1) Define a measure of distance that takes into account the constraints on resources (power, area) for *each of the agents* in a multi-agent scenario.
2) Develop an *efficient* algorithm for gathering of two agents (meeting the distance criterion) amidst n static obstacles.
3) Extend the approach to gathering of k agents amidst obstacles.
4) Enhance the strategy to handle dynamic obstacles.
5) Provide experimental results on gathering of agents with limited hardware on each.

Since our objective is to reduce the frequency of recharging of *each agent*, an intuitive approach to define a distance measure is based on ensuring that the difference between the distances travelled by any two agents (before meeting) is small. In other words, it is beneficial to *minimize the maximum Euclidean distance* travelled by the agents before they meet. We refer to the location that achieves this as the *minimax point* (and denote by P_M). P_M corresponds to an environment without static obstacles. The *minimax point* is also motivated by prior work on optimal location of emergency services such as hospitals [7].

For an environment with obstacles that autonomous agents need to negotiate, we refer to the location that minimizes the maximum Euclidean distance as the *gathering point* (and denote by P_G). The development of an efficient algorithm for computing P_G requires an appropriate model of the obstacles for collision avoidance. We assume that the area used by machines, furniture etc. can be represented by polygons of arbitrary shape (they could be non-convex). We assume further that our agents are small and each can be represented by a point mass (similar to assumptions in prior work [8]).

The contributions of this paper are as follows. We develop a number of theoretical results that facilitate computation of P_G. We then present efficient algorithms (with low asymptotic complexity) for calculation of P_G when the environment consists of (i) two agents and n obstacles and (ii) k agents and n obstacles. Efficient algorithms facilitate quick computation of P_G given information on obstacle locations. They also allow update of P_G whenever the position of the agents change or when the obstacle locations change slightly. We then consider

Bhaskar Vundurthy and K. Sridharan are with the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai - 600036, India; Email for correspondence: sridhara@iitm.ac.in

moving objects in the same environment and develop a path following algorithm that each agent can incorporate to move to P_G safely even when dynamic objects are present.

Another contribution of the work presented is efficient hardware realization of the proposed algorithms. We depict an implementation of the gathering point computation on small robots, each equipped with only a microcontroller. The memory on the microcontroller is shown to be adequate for implementation of the proposed algorithms. No external storage is required. Further, position information and distance (to obstacle) information are obtained using simple and low-cost units. The approach does not require a central controller for effecting the gathering of multiple agents.

Prior work on hardware multi-agent systems has examined coordination and other tasks [9] but *there does not appear to be efforts on development of efficient geometric algorithms for the gathering task amidst obstacles when, in particular, a distance criterion is employed*. It is worth noting that the problem studied here is different from the classical path finding (or shortest path calculation) problem amidst obstacles since the destination is known in the latter. While there are some efforts in the domain of operations research with respect to facility location [10], [7], [11], obstacles have not been considered in these works. A detailed discussion of related literature is presented in section II.

The organization of the rest of this paper is as follows. Section III introduces the terminology used in the paper. Section IV presents the efficient algorithm for computing P_G for a pair of agents amidst n obstacles. Section V develops the low-complexity algorithm for k agents. The extension to dynamic obstacles is presented in section VI. Experimental results are given in section VII. Section VIII concludes the paper.

II. Related Work

The focus of this paper is on developing fast geometric algorithms for the gathering problem in the presence of obstacles and the minimax distance criterion. To this end, we first review prior work in the domains of computational geometry and operations research on related problems. We then summarize recent work in multi-agent consensus.

A. Prior Work in Geometric Algorithms

Work on minimax criterion has been reported as early as 1960s in the context of location theory [10]. The usefulness of the minimax criterion in defence applications has been addressed in [12] where the author mentions the advantage of minimizing the maximum distance (as opposed to the total or average distance) to trouble spots to save time in the context of deployment of airborne soldiers.

The authors in [7] present a finite solution procedure for the minimax problem (in the absence of obstacles) for Euclidean and rectilinear distance measures based on geometric arguments. No asymptotic complexity analysis is available in [7]. The authors in [13] generalize the work in [7] by considering the l_p metric and establish the efficiency of their procedure via actual computation times. An $O(n(\log n)^3(\log\log n)^2)$ algorithm has been reported in [11] to find a point in two dimensions that minimizes the maximum weighted distance to a point in a set of n given points. However, obstacles have generally not been considered in these formulations. The authors in [14] consider meeting of a pair of agents with line of sight communication between them and present an $O(n^3 \log n)$ algorithm under the minimax metric where n is the total number of vertices. A provably correct algorithm for rendezvous of agents (equipped with omni-directional range-limited visibility sensors) in a simply connected, non-convex environment has been reported in [15]. In general, efficient geometric algorithms (with low asymptotic complexity) for gathering of a large number of agents amidst multiple static obstacles *incorporating a distance criterion do not appear to be available*. Further, no moving objects have been considered.

B. Prior Work on Consensus in Multi-Agent Systems

Early work in multi-agent consensus includes [16], [17] and [18]. An article by Cao et al. [9] summarizes the contributions in the last decade to distributed multi-agent coordination. Han et al. [19] provide necessary and sufficient conditions for robust first and second-order consensus for a class of multi-agent dynamical systems. Consensus in multi-agent systems with sampled position and velocity data has been studied in [20]. The flocking problem for multi-agent systems is studied via model predictive control in [21]. The authors in [2] present the state of the art in applications of industrial agents. They point to the potential of multi-agent coordination in military, defense and humanitarian relief applications. Sampled-data based event-triggered control for networked systems has been recently explored in [22]. The authors in [23] examine the key contributing factors to acceptance of agents in industrial environments.

In summary, considerable work has been done on control and related aspects of consensus in multi-agent systems. However, it is valuable to explore geometric aspects (such as distance optimization) pertaining to multi-agent coordination since they have a direct impact on the energy consumption. Further, it is of interest to examine the complexity (in terms of area, weight etc.) of an efficient hardware realization.

III. Assumptions and Definitions

A. Agents and Their Characteristics

We assume, throughout this paper, identical agents. No communication with a central controller is required to meet the twin objectives of collision avoidance and minimax distance criterion. However, inter-agent communication is permitted. In particular, one of the agents accepts the initial location information from all other agents and computes the gathering point P_G. This point is then communicated back to the (other) agents. All the agents are assumed to have knowledge of the locations of the static obstacles.

The agents are assumed to be equipped with position encoders for localization, on-board distance sensors for handling dynamic obstacles and communication modules for interactions between agents. Details regarding the definition of the distances are presented in section III-B. Additional information

on how the agents handle different types of obstacles are provided in sections VI and VII.

B. Definitions

The minimax point (denoted by P_M) for k agents located at $\{P_1(x_1, y_1), P_2(x_2, y_2), \cdots, P_k(x_k, y_k)\}$ is defined as the point that minimizes the maximum Euclidean distance from each of the k agents as given in Eq. (1).

$$P_M(x, y) = min\{\max_{1 \le i \le k} \sqrt{(x - x_i)^2 + (y - y_i)^2}\} \quad (1)$$

It is worth noting that, for a set of three non-collinear agents P_1, P_2 and P_3, the minimax point P_M corresponds to the circumcentre of the triangle formed by P_1, P_2 and P_3 if the triangle (formed) is acute-angled. When P_1, P_2 and P_3 form a right triangle or an obtuse-angled triangle, P_M corresponds to the midpoint of the longest side.

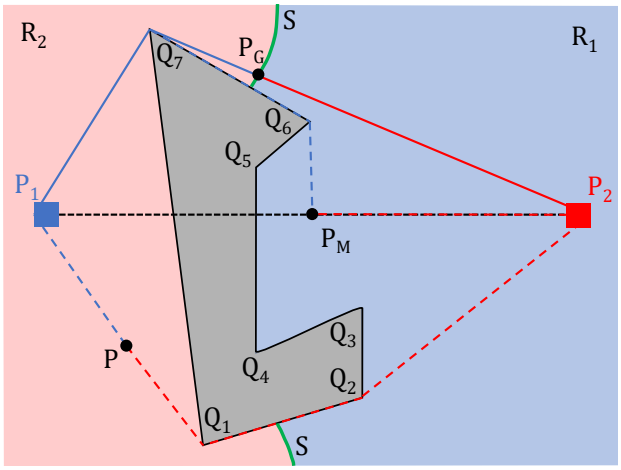


Fig. 1. Gathering point P_G for a pair of agents

When obstacles are introduced, the calculation of the point that minimizes the maximum distance is more complex. The notion of gathering point for this scenario is expressed by Definition 1.

Definition 1: Gathering point P_G for k agents amidst n static obstacles is defined as the point external to all obstacles attaining the minimum of maximum of distances computed from k agent locations (to various points in the plane). ■

The gathering point, P_G, for two agents in the presence of a non-convex polygonal obstacle is illustrated in Fig. 1. We note that there are two paths between the agents through the vertices of the obstacle and each of these is a candidate for the shortest path. P_G corresponds to the mid-point of the shortest of the two paths from agent P_1 to agent P_2 (a proof of this is presented in section IV). The midpoint of the line segment P_1P_2 would be the minimax point, P_M, if there had been no obstacles. The significance of P_G can be observed from the difference in distances to P_M and P_G from their initial location.

In the presence of static polygonal obstacles, agents move from one obstacle vertex to another before arriving at the gathering point P_G. The distance travelled by an agent from its initial location to an intermediate location (for instance, a vertex of an obstacle) may be thought of as the *weight* on the agent at that intermediate location. For instance, in Fig. 1, the weight on agent P_1 at vertex Q_6 is the distance travelled by P_1 to reach Q_6 (which is $(P_1Q_7 + Q_7Q_6)$). Using the notion of weights, we can define a *weighted minimax point* (denoted by P_M^W) as per Eq. (2).

$$P_M^W(x, y) = min\{\max_{1 \le i \le k} (\sqrt{(x - x_i)^2 + (y - y_i)^2} + d_i)\} \quad (2)$$

The definition in (2) assumes k agents with weights given by d_i, $i \in \{1, 2, \cdots, k\}$. P_M^W is used to compute P_G in the algorithm in section V.

The development of efficient algorithms also requires the notions of *visibility* and *last turn* since agents need to avoid collisions with the interior of the obstacles. Two arbitrary points, T_1 and T_2, are visible to each other, if the line segment joining T_1 and T_2 does not intersect the interior of any obstacle. For instance in Fig. 1, P_1 is visible to Q_1 but not to P_2. The *last turn* location is defined as the point (vertex) that an agent reaches before becoming visible to the destination point. The *last turn* location for agent P_2 (to reach P) in Fig. 1 is Q_1 while the last turn location for agent P_1 to reach P_G is Q_7.

IV. Computing P_G for a pair of agents amidst n polygonal obstacles

In this section, we consider one generalization of the case discussed in section III. In particular, we consider two agents (located at P_1 and P_2) and n obstacles (each with a total of c vertices where c is a constant) and present an $O(n \log_2 n)$ time algorithm to compute P_G.

The key ideas are as follows. The gathering point P_G for a pair of agents is shown in Fig. 1. Let the curve S be the locus of all points that are equidistant to P_1 and P_2. Thus, on one side of S lies region R_1 (shown in blue), that is the collection of all points in the plane farthest from P_1. Therefore, the maximum of distances from P_1 and P_2 to any point in R_1 would be the distance from P_1. Minimizing this over R_1 brings one back to the curve S. The same applies to region R_2 (shown in red). The location of gathering point P_G can thus be narrowed down to curve S. Lemma 1 extends this idea to n obstacles.

Lemma 1: The gathering point P_G for two agents moving amidst n polygonal obstacles is the point on the shortest path (from one agent to another) that is equidistant from both agents.

Proof: This can be established as an extension of the no obstacles case where P_M corresponds to the mid-point. When obstacles are present, the path joining the two agents need not be just as one piece: it is, in general, a collection of segments. The total length of the collection is of interest and the location corresponding to half this length determines a potential gathering point. Since several such collections can exist, we choose the one that has the smallest total length which corresponds to the shortest path. The midpoint of the shortest path corresponds to P_G. **Q.E.D.**

We now present **Algorithm *Gathering_point_pair*** that uses Lemma 1 to compute the gathering point for two agents amidst n polygonal obstacles. We assume that P_1 first communicates its location to P_2. P_2 then uses the following algorithm to compute the gathering point, P_G, and transmits the same back to P_1.

Algorithm *Gathering_point_pair*

INPUT: Two distinct, initial locations of the agents, denoted by P_1, P_2. n non-intersecting polygonal obstacles with c vertices each, labeled $(O_1, O_2, \cdots, O_n)$.

OUTPUT: Gathering point P_G

Step 1: Compute the shortest path (S_P) from P_1 to P_2 amidst n polygonal obstacles.

Step 2: Compute the mid point of S_P while moving from one vertex to the other starting from P_1. Output this midpoint as the gathering point P_G.

Theorem 1: The time complexity of ***Algorithm Gathering_point_pair*** *is* $O(n \log_2 n)$.

Proof: Step 1 of the algorithm computes the shortest path from P_1 to P_2 amidst n obstacles with c vertices each (where c is a constant) in $O(n \log_2 n)$ time based on the approach in [24]. Given the shortest path, Step 2 takes $O(n)$ time to compute the point equidistant from P_1 and P_2 (since the $O(n)$ vertices [24] along the shortest path are available from Step 1 and we can obtain the length of each segment in the path in constant time). Thus, the overall time complexity of **Algorithm *Gathering_point_pair*** is $O(n \log_2 n)$. **Q.E.D.**

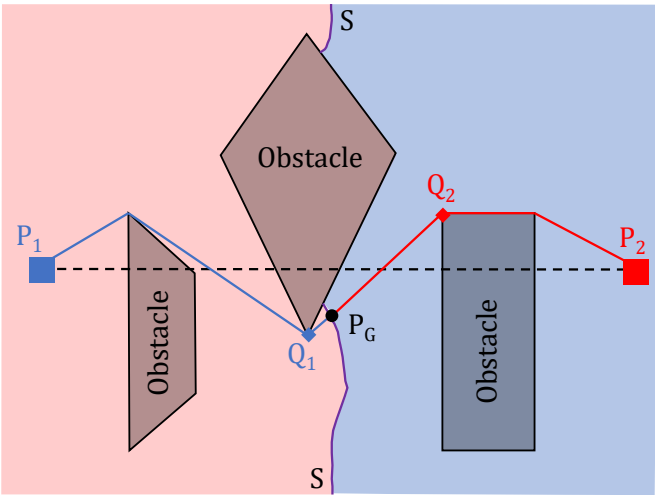


Fig. 2. Gathering point (P_G) for two agents in the presence of three polygonal obstacles

Fig. 2 illustrates the gathering point P_G for two agents located at P_1 and P_2 amidst three obstacles with four vertices each. Curve S is the collection of points equidistant from the two agents P_1 and P_2. Gathering point P_G is the point of intersection of this curve S and the shortest path from P_1 to P_2. It is worth noting that **Algorithm *Gathering_point_pair*** computes this gathering point P_G without the need to compute the curve S, thus saving on computational time.

Remark 1: In step 1 of **Algorithm *Gathering_point_pair***, the shortest path from P_1 to P_2 need not be unique. So, in principle, there can be multiple choices for the gathering points. However, to ensure that the agents gather at one point, one of the shortest paths (when there are many) is chosen by the agent at P_2. The midpoint of this path is then communicated to the agent at P_1. In the implementation, this is accomplished by an array which stores (and updates) the shortest path. It is worth noting that this approach obviates the need for a central controller.

We have so far assumed only a pair of agents. We next consider generalization to k agents moving amidst n polygonal obstacles.

V. Extension to k Agents Moving Amidst n Obstacles

In this section, we present an $O(k^2 + kn \log_2 n)$ time algorithm for calculating P_G of k agents amidst n obstacles. Each obstacle is assumed to have c vertices where c is a constant.

We begin by presenting the main ideas for computation of P_G in section V-A. These results capture a generalized scenario where the points are any arbitrary locations in the plane. Similarly, the weights at these locations are merely non-negative real numbers.

A. Main Ideas

For two agents moving amidst obstacles, the calculation of P_G can be pursued by dividing the plane into two regions, each being farthest from one of the two agents. As the number of agents increases to k, the division of the plane into k regions becomes complex. We therefore develop an alternate procedure based on the notion of weighted minimax point (P_M^W) defined in section III. We first present three results (Lemmas 2, 3 and 4) to compute P_M^W for two, three and k agents with arbitrary weights associated to their locations. These form the basis for developing an algorithm to compute P_M^W. The gathering point, P_G, is obtained using P_M^W.

Lemma 2: For two agent locations P_1 and P_2 with weights d_1 and d_2 respectively, the weighted minimax point P_M^W is given by

$$P_M^W = \begin{cases} P_i, & \text{if } d_i \geq (l_{12} + d_j) \\ P_{12} & \text{otherwise} \end{cases} \tag{3}$$

for $i, j \in \{1, 2\}$ with $i \neq j$. l_{12} is the Euclidean length of the line segment P_1P_2 and $P_{12} = \frac{l_{12}(P_1+P_2)+(d_1-d_2)(P_1-P_2)}{2\times l_{12}}$.

Proof: Let P be an arbitrary point on line segment P_1P_2 that divides it in the ratio $r : (l_{12} - r)$, $0 \leq r \leq l_{12}$. Thus, the distances from P_1 and P_2 to P are given by

$$\begin{aligned} d_{PP_1} &= d_1 + r \\ d_{PP_2} &= d_2 + l_{12} - r \end{aligned} \tag{4}$$

Without loss of generality, let the first condition in Eq. (3) be expressed as $d_1 \geq (l_{12} + d_2)$. For any non-negative constant z, we thus have

$$\begin{aligned} d_1 - z &= l_{12} + d_2 \\ \Rightarrow d_{PP_1} &= d_1 + r \\ \Rightarrow d_{PP_2} &= d_1 - z - r \end{aligned} \tag{5}$$

Thus, max($\{d_{PP_1}$,$d_{PP_2}\}$) is d_{PP_1}. The minimum for d_{PP_1} occurs when $r = 0$ which indicates that the minimax point is P_1. Similarly, for the condition $d_2 \geq (l_{12} + d_1)$, P_2 is the minimax point. This proves the first condition.

The minimax point in the absence of the constraint described in Eq. (5), is the point on line segment P_1P_2 that is equidistant from P_1 and P_2. We have from Eq. (4),

$$\begin{aligned} d_{PP_1} &= d_{PP_2} \\ \Rightarrow d_1 + r &= d_2 + l_{12} - r \\ \Rightarrow r &= \frac{l_{12} + (d_2 - d_1)}{2} \end{aligned} \tag{6}$$

The expression for P_{12} then follows from the section formula which divides line segment P_1P_2 in the ratio $(r : l_{12} - r)$ where r is given by Eq. (6). **Q.E.D.**

We now present Lemma 3 to compute P_M^W for three agents with arbitrary weights at their locations. We denote by l_{ij} the length of line segment P_iP_j. Further, P_{ij} is the weighted minimax point of agent locations P_i and P_j (with weights d_i, d_j) computed using Lemma 2. d_{ij} is the weight computed at P_{ij}. The length of line segment from P_{ij} to P_k is denoted by ${}_{ij}l_k$. It is worth noting that the locus of points equidistant from two agent locations with associated weights is a branch of a hyperbola (explained further in proof of Lemma 3).

Lemma 3: For three agent locations P_1, P_2 and P_3 with weights d_1, d_2 and d_3 respectively, the weighted minimax point P_M^W is given by

$$P_M^W = \begin{cases} P_i, & \text{if } d_i \geq (l_{ij} + d_j) \ \& \ d_i \geq (l_{ki} + d_k) \\ P_{ij}, & \text{if } d_{ij} \geq ({}_{ij}l_k + d_k) \\ P_{123} & \text{otherwise} \end{cases} \tag{7}$$

for $i, j, k \in \{1, 2, 3\}$ with $i \neq j \neq k$. P_{123} is the point of intersection of the three branches of hyperbolas constructed on the three sides with their respective weights.

Proof: The first two conditions follow from the proof of Lemma 2. When these two conditions do not apply, the minimax point is the point equidistant to the three agents with their corresponding weights. The locus of points (P) equidistant to two agents P_1, P_2 (with weights d_1, d_2) is a hyperbola as shown in Eq. (8).

$$\begin{aligned} PP_1 + d_1 &= PP_2 + d_2 \\ \Rightarrow PP_1 - PP_2 &= d_2 - d_1 \end{aligned} \tag{8}$$

The point equidistant to the three agents (which corresponds to P_M^W) is identical to the point of intersection of the three hyperbolas constructed on the three sides formed by initial locations of agents. **Q.E.D.**

Figures 3 (a) and 3 (b) illustrate the two cases in Lemma 2 for computing the weighted minimax point for two agents with non-zero weights. Fig. 3 (c) illustrates the last case in Lemma 3 and various terms used in computing the same. Numerical illustrations of Lemmas 2 and 3 are presented in Appendix A. It is worth noting that the conditions in Lemma 3 correspond to cases where either one or two of the points are sufficient

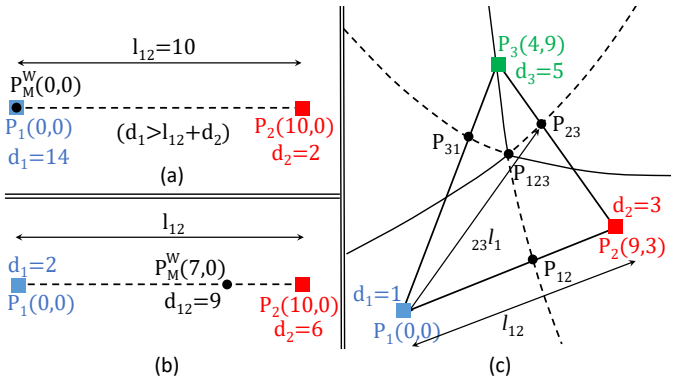


Fig. 3. Minimax point for two and three agents with non-zero weights

to compute the weighted minimax point. We now extend the ideas to k agents.

Lemma 4: The weighted minimax point of k agents is identical to the weighted minimax point of three agents (among k agents) calculated using Lemma 3.

Proof: When the weights on all the agents are equal, it follows from [25] that three points suffice to determine the weighted minimax point. If the weights are unequal, the last case in Lemma 3 holds. When there are more than three agents, the hyperbolas do not intersect (in general) at a single point. From the properties of a polygon, it follows that a maximum of three hyperbolas can intersect at a single point. The agents corresponding to these three hyperbolas determine the triplet that constitutes the weighted minimax point. **Q.E.D.**

The remaining $k-3$ points satisfy the following condition:

$$d_{P_M} \geq d_i + l_{iP_M} \tag{9}$$

where d_{P_M} is the weight at the minimax location, d_i is the weight at the i^{th} location and l_{iP_M} is the length of the line segment joining P_i and P_M. Thus, these points do not play a role in computing the minimax point. We now present **Algorithm *Minimax_point_weighted*** to compute the weighted minimax point for k agents, using Lemmas 3 and 4.

Algorithm *Minimax_point_weighted*

INPUT: k distinct initial locations of the agents denoted by $P_1, P_2, \cdots, P_k$. Weights at these k locations are denoted by the set $D = \{d_1, d_2, \cdots, d_k\}$.

OUTPUT: Weighted minimax point, P_M^W

Step 1: Choose the largest, second largest and third largest values from D and denote by (d_a, d_b, d_c) respectively where $a, b, c \in \{1, 2, \cdots, k\}$ and $a \neq b \neq c$.

Step 2: Compute Q_M as the weighted minimax point of three agent locations P_a, P_b, P_c with weights (d_a, d_b, d_c) using Lemma 3. Let d_m be the corresponding weight at Q_M.

Step 3: Return Q_M as the weighted minimax point P_M^W if Eq. (10) holds (where l_{im} represents the Euclidean distance between P_i and $Q_M$and **Stop**.

$$d_m \geq l_{im} + d_i \qquad \forall \quad i \in \{1, 2, \cdots, k\} \tag{10}$$

Otherwise, choose an agent location and its weight P_d, d_d (where $d \in \{1, 2, \cdots, k\}$) that does not satisfy Eq. (10) and proceed to **Step 4**.

Step 4: Compute weighted minimax points for the three combinations (P_a, P_b, P_d), (P_a, P_c, P_d) and (P_b, P_c, P_d) using Lemma 3. Denote these points as Q_x, Q_y, Q_z and their weights as d_x, d_y, d_z respectively. Replace d_m with the maximum of d_x, d_y, d_z and Q_M with its corresponding minimax point. Replace (P_a, P_b, P_c) with the corresponding combination of d_m and proceed to **Step 3**. ■

Fig. 4 illustrates the algorithm for five ($k = 5$) agent locations. Step 1 picks three agent locations P_1, P_2, P_4 with highest weights (Fig. 4 (a)) and denotes them as (P_a, P_b, P_c). Step 2 computes the weighted minimax point (Q_M) and its weight (d_m) for (P_a, P_b, P_c) using Lemma 3. Q_M is the point of intersection of three branches of hyperbolas constructed on each side of $\triangle P_a P_b P_c$ (Fig. 4 (a)).

Step 3 then uses Eq. (10) to check if Q_M minimizes the maximum of distances to all agent locations. Since P_3 does not satisfy Eq. (10), P_3 is renamed as P_d (Fig. 4 (b)). Step 4 computes the weighted minimax point for the three combinations (P_a, P_b, P_d), (P_a, P_c, P_d) and (P_b, P_c, P_d) and identifies that the combination with highest weight is (P_a, P_b, P_d). Fig. 4(b) illustrates Q_x as the point of intersection of the three branches of hyperbolas for $\triangle P_a P_b P_d$. Eq. (10) of Step 3 now reveals that Q_x is indeed the weighted minimax point for all agent locations and is thus the output (P_M^W).

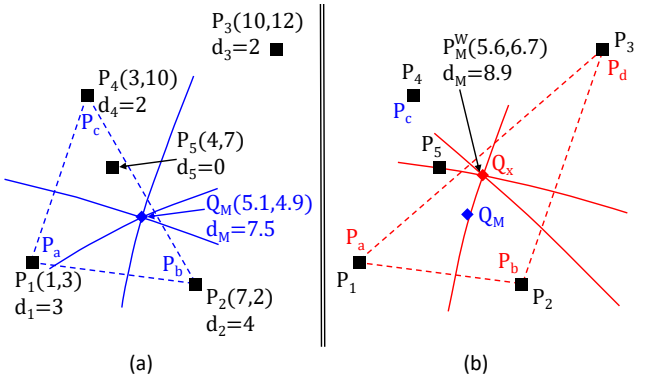


Fig. 4. Illustration of **Algorithm *Minimax_point_weighted*** (a) Steps 1 and 2 (b) Steps 3 and 4

The convergence of **Algorithm *Minimax_point_weighted*** follows from the fact that the distance value d_m increases with each iteration. Since there are a finite number of points (k), the algorithm terminates when d_m reaches its maximum value. The complexity of **Algorithm *Minimax_point_weighted*** is expressed by Theorem 2.

Theorem 2: ***Algorithm Minimax_point_weighted*** *takes $O(k^2)$ time where k is the number of agents.*

Proof: Step 1 computes the first, second and third maximum in $O(k)$ time. Step 2 uses Lemma 3 to compute P_M^W in constant time. Step 3 verifies the condition in Eq. (10) for all k agents and thus takes $O(k)$ time. Step 4 computes three minimax points and then repeats Step 3 at most k times, thus taking $O(k^2)$ time. Hence, the overall complexity is $O(k^2)$. **Q.E.D.**

B. *Efficient Algorithm for k-Agent Gathering*

Algorithm *Minimax_point_weighted* presented in section V-A is used in developing an efficient algorithm to compute the gathering point P_G. Polygons formed from the vertices of the obstacles and the agent locations contribute to the location of P_G. In particular, the *gathering point turns out to be the weighted minimax point of one of the polygons constructed from the obstacle vertices and agent locations* (an outline of the proof of this is part of Lemma 5).

However, a direct algorithm based on this idea has high computational complexity. This is in view of the fact that for k agents moving amidst n polygonal obstacles with c vertices each, we have up to ${}_{cn+k}P_k$ polygons to consider (and shortest paths from the agents to the vertices of these polygons).

Fig. 5 illustrates the gathering point for six agents moving amidst three polygonal obstacles (with four vertices each). One of the ${}_{4*3+6}P_6$ combinations is shown in Fig. 5 as the polygon $Q_1Q_2Q_3Q_4Q_5Q_6$. *Weight d_i (where $i \in \{1, 2, \cdots, 6\}$) at a location Q, corresponds to the distance travelled by agent P_i from its initial location to Q.* This can be observed in Fig. 5 where d_3 (at Q_3) is the distance travelled by agent P_3 from its start location to Q_3, which is the length of the line segment P_3Q_3. The weighted minimax point P_M^W of this polygon $Q_1Q_2Q_3Q_4Q_5Q_6$ has the least maximum distance value and corresponds to the gathering point P_G.

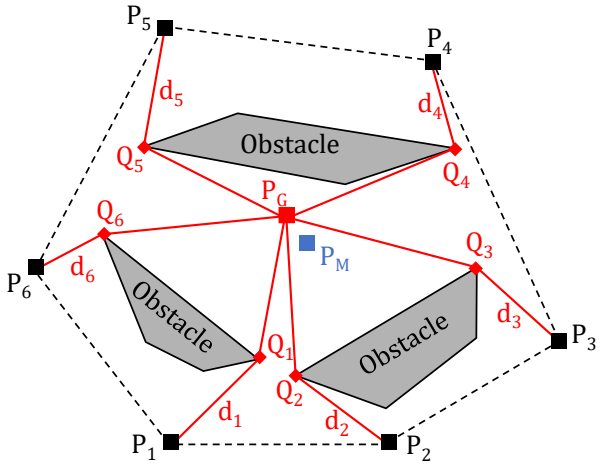


Fig. 5. P_G and P_M for six agents amidst three polygonal obstacles

The proposed efficient algorithm is based on the following observation. The point P_M, which is the minimax point of k initial locations of agents with zero weights, is 'on the same side' as P_G as illustrated in Fig. 5. It is worth noting that P_M and P_G are both visible to the last turn locations to P_M (which are $Q_1, Q_2, Q_3, Q_4, Q_5, Q_6$). Thus, by identifying the last turn locations to P_M, one can immediately identify the polygon $Q_1Q_2Q_3Q_4Q_5Q_6$ whose weighted minimax point is the gathering point P_G. Therefore, the search for the gathering point P_G can be reduced from multiple polygons to one. This observation is established formally in Lemma 5.

Lemma 5: Let P_M be the minimax point for k agents with zero weights and P_G be the gathering point amidst n polygonal obstacles. Further, let $Q_1, Q_2, \cdots, Q_k$ be the last turn locations for these k agents attempting to meet (amidst

obstacles) at the minimax point P_M.

If P_M lies external to all obstacles, the weighted minimax point of these last turn locations is the gathering point P_G.

Proof: Let the maximum of distance values to P_M and an arbitrary point (denoted by P) be r_M and r_P respectively. It follows from Eq. (1) that the maximum of distances is lowest for P_M compared to any arbitrary point in the plane (P). We thus have,

$$r_M \leq r_P \tag{11}$$

Since the distance between two points generally increases with introduction of obstacles, we have Eq. (12).

$$\begin{aligned} r_M &\leq s_M \\ r_P &\leq s_P \end{aligned} \tag{12}$$

where the maximum of distances (lengths of the shortest path from agents' locations) to P_M is s_M and to P is s_P. Further, consider $Q_1, Q_2, \cdots, Q_k$ to be the last turn locations of agents attempting to gather at P_M. Let the weighted minimax point of $Q_1, Q_2, \cdots, Q_k$ be P_M^W and the maximum of distances value to P_M^W be s_W. In order to prove that P_M^W is the gathering point P_G, it is adequate to show

$$s_W \leq s_P \tag{13}$$

It follows from Eq. (2) that the weighted minimax point of the polygon constructed using last turn locations has the lower maximum of distances value compared to any other point with same last turn locations. We thus have

$$s_W \leq s_M \tag{14}$$

It follows from Eq. (11) that the maximum of distances value to P_M (which is r_M) will increase with addition of obstacles (Eq. (12)). The only point with lower maximum of distances value compared to P_M would be the weighted minimax point of its last turn locations as given by Eq. (2) and Eq. (14).

Therefore, the point with lowest maximum of distances value in the presence of obstacles is the weighted minimax point P_M^W which is in turn the gathering point P_G as given by Definition 1. We thus have Eq. (13). **Q.E.D.**

Lemma 5 can be used in developing **Algorithm *Gathering_point_efficient*** that computes the gathering point P_G for k agents moving amidst n obstacles with c vertices each. We assume agents at $P_1, P_2, \cdots P_{k-1}$ transmit their locations to the agent at P_k which then executes the following algorithm and computes the gathering point P_G. The location of P_G is then communicated back by P_k to all the other agents, namely $P_1, P_2, \cdots, P_{k-1}$.

Algorithm *Gathering_point_efficient*

INPUT: k distinct, initial locations of the agents denoted by $P_1, P_2, \cdots, P_k$. n non-intersecting polygonal obstacles with c vertices each, denoted using the set $O = \{O_1, O_2, \cdots, O_n\}$.

OUTPUT: Gathering point P_G.

Step 1: Calculate the minimax point P_M of polygon $P_1P_2\cdots P_k$ (with zero weights). Verify if P_M is contained in any of the n obstacles. Identify the obstacle (if any) that contains P_M, as O_j (where $j \in \{1, 2, \cdots, n\}$) and exclude it from the set O.

Step 2: Calculate the shortest path for the k agents to reach P_M amidst the obstacles. Denote the last turn locations in the shortest paths before reaching P_M as $Q_1, Q_2, \cdots, Q_k$ and the corresponding path lengths as $d_1, d_2, \cdots, d_k$.

Step 3: If O_j is empty, goto **Step 4** else goto **Step 5**.

Step 4: Replace P_M with the weighted minimax point (P_M^W) of polygon $Q_1Q_2\cdots Q_k$ computed with weights $d_1, d_2, \cdots, d_k$ using **Algorithm *Minimax_point_weighted***. Compute the shortest paths for k agents from initial locations to P_M. Identify the last turn locations to reach P_M as $R_1, R_2, \cdots, R_k$ and the distances along shortest paths to reach them as $f_1, f_2, \cdots, f_k$. Go to **Step 6**.

Step 5: Replace P_M with the gathering point of polygon $Q_1Q_2\cdots Q_k$ computed with weights $d_1, d_2, \cdots, d_k$ and one polygonal obstacle O_j. Include O_j back into the set O and compute the shortest path from initial locations to P_M. Denote the last turn locations as $R_1, R_2, \cdots, R_k$ and the corresponding lengths of shortest paths as $f_1, f_2, \cdots, f_k$.

Step 6: If $R_i = Q_i \,\forall\, i \in \{1, 2, \cdots, k\}$, output P_M as the gathering point P_G and **Stop**. Else, replace Q_i with R_i and d_i with $f_i \,\forall\, i \in \{1, 2, \cdots, k\}$. Exclude O_j (if any) from the set O and return to **Step 3**.

The correctness of this algorithm follows from Lemma 5. **Algorithm *Gathering_point_efficient*** is illustrated in Fig. 5. Steps 1 and 2 compute the minimax point P_M and paths from agents' initial locations. Since P_M is not contained in any obstacle (Step 3), Step 4 computes the weighted minimax point (P_G in Fig. 5) of last turn locations $Q_1, Q_2, \cdots, Q_6$. Recomputing shortest paths to P_G does not affect the last turn locations (Step 6). Thus weighted minimax point computed in Step 4 is output as the gathering point P_G. The convergence of this algorithm follows from the fact that there are only a finite number of polygonal obstacles to be considered in computing the gathering point.

*Theorem 3: The asymptotic time complexity of **Algorithm Gathering_point_efficient** is $O(k^2 + kn\log_2 n)$ where n is the number of obstacles with c vertices each and k is the number of agents.*

Proof: Step 1 takes $O(k^2)$ time for computing the minimax point as given by Theorem 2. It takes additional $O(n)$ time for checking if P_M lies in any of the n obstacles. Computation of shortest paths to P_M from k agents in Step 2 takes $O(kn\log_2 n)$ as given by [24]. Computation of minimax point in Steps 4 and 5 once again take $O(k^2)$ time and the corresponding shortest paths take $O(kn\log_2 n)$ time. Step 6 returns the gathering point P_G or reassigns variables for further computation. Thus, the overall time complexity is $O(k^2 + kn\log_2 n)$. Q.E.D.

Remark 2: **Algorithm *Gathering_point_efficient*** stops with calculation of the gathering point P_G. The task of computing the shortest path to P_G is left to the individual agents which can perform this in parallel. This has the advantage of keeping the time complexity of the algorithm low while at the same time, reducing the information to be transmitted (by P_k) to each of the $k-1$ agents.

A special case for the algorithm is when the minimax point, P_M, is contained within an obstacle. This is handled without any increase in the overall time complexity, viz. $O(k^2 + kn \log_2 n)$ and corresponds to computations in step 5 of the algorithm.

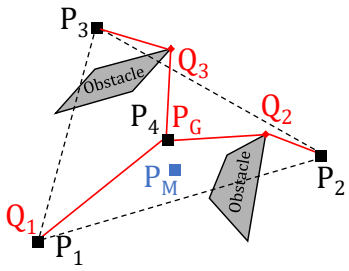


Fig. 6. Gathering point P_G is identical to agent location P_4

Another special case is illustrated in Fig. 6. Here, the gathering point P_G (computed by the agents) coincides with initial location of one of the agents (P_4). While agent P_4 remains at P_G, the remaining agents follow the shortest path to P_G.

The information transmitted between various agents leads to a communication complexity of $O(k)$ [26], [27]. Appendix B provides further details on the length of information communicated.

VI. Enhancement to Environments with Dynamic Obstacles

In an industrial environment involving multiple hardware agents, one (moving) agent could itself be an obstacle to another. Further, there may be automated guided vehicles and humans moving between stations. In the domestic setting too, dynamic obstacles are common. The algorithms presented so far handle k agents moving amidst static obstacles with distance optimization as the goal. In this section, we answer the following question: *Can the paths generated for each agent avoiding collision with the interiors of the static obstacles be used as such when handling dynamic obstacles ?*

We present an algorithm for *path following in the presence of dynamic obstacles* that enables retaining the location (of P_G) computed with merely the static obstacles by **Algorithm *Gathering_point_efficient***. It is assumed that the agents are equipped with distance sensors and communication modules. As indicated in Remark 2, the agents calculate the shortest path (in parallel) to P_G amidst static obstacles. Dynamic obstacles in the path of an agent to P_G are detected by the distance sensor.

Every agent that encounters an obstacle broadcasts its current location and receives the locations of similarly obstructed fellow agents. The agents use this information to identify if they are obstructing each other. In such a scenario, the agent with the highest index number proceeds further while the other agents wait for their turn.

Algorithm *Path_Following*

INPUT: Initial locations of all agents and static obstacles. Gathering point P_G and paths of various agents to P_G. Information from distance sensor on each agent.

OUTPUT: All agents gather at P_G

Step 1: Translate each agent along its corresponding path to P_G until the distance sensor on an agent (say P_i) detects an obstacle. If all agents arrive at P_G, **Stop**.

Step 2: Compare the location of detected obstacle with the already stored information on static obstacles. In case of a match, conclude that the detected obstacle is indeed a static obstacle and return to **Step 1**. Else proceed to **Step 3**.

Step 3: Broadcast the current location of the agent (P_i) along with its index number (i) to fellow agents and wait for reception of a similar message from a fellow agent.

Step 4: If P_i does not receive a reply, conclude that the detected obstacle is an unknown dynamic obstacle (not a fellow agent) and proceed to **Step 7**.

Step 5: If P_i receives a reply from one or more fellow agents, compare their locations to verify if the agents are obstructing each other. If True, proceed to **Step 6**, else proceed to **Step 7**.

Step 6: If the index i is greatest among all the obstructing agents (received in **Step 5**), conclude that the agent P_i should proceed further and return to **Step 1**. Else proceed to **Step 7**.

Step 7: Stop the current agent (P_i) and monitor distance sensor readings. If the sensor continues to detect an obstacle, return to **Step 3**. Else return to **Step 1**.

Remark 3: The choice to stop momentarily is adopted so that the distance criterion and therefore the energy considerations are met. Other approaches to handle dynamic obstacles, in general, lead to greater travel distance for one or more agents.

VII. Experimental Results

In this section, we present the details of an efficient hardware realization of the algorithms on small robots that act as autonomous agents. The hardware on the systems is small so as to keep cost, weight and power consumption low. Each robot is equipped with an 8-bit, 20 MHz ATmega328P microcontroller that executes the algorithms (some numerical computation aspects are presented in section VII-A). Further, position information is gathered via two MOC7811 sensors on each robot. These are inexpensive and adequate for small agents. The robots are also equipped with an ultrasonic sensor for dynamic obstacle detection (ultrasonic sensors are simple, low-cost and lightweight). The entire arrangement is powered by a 12V, 1.3AH sealed maintenance-free battery. The communication among agents is realized with the help of Xbee-PRO RF modules. These units are low cost and low power wireless sensor networks operating at 2.4 GHz with a transmission range of up to 90m.

A. Numerical Aspects in Implementation of the Algorithms on Resource-Constrained Platforms

The proposed algorithms involve symbolic computation to arrive at the gathering point P_G. This can be observed in Lemma 3 where the minimax point is found by solving two or more second degree equations in two variables. The ATmega328P microcontroller supports 32KB of Flash Memory, 2KB SRAM and 1KB EEPROM. Assuming 4 bytes of storage for each of the x and y coordinates, a maximum of 256

vertices can be handled by the microcontroller simultaneously. Since microcontrollers are not equipped to perform symbolic computation, we develop a numerical procedure to solve the second degree equations in two variables.

The last case of Lemma 3 gives the minimax point as the point of intersection of three hyperbolas as shown earlier in Fig. 3 (c). The point of intersection of any two curves (denoted by L_1 and L_2) is the point at which both curves have a function value equal to 0. Further, if one curve (for example, L_1) is sampled at a fixed value as shown in Fig. 7, it gives a set of points (denoted by $S_i \ \forall \ i \in \{1, 2, \cdots, m\}$, $m \in \mathbb{Z}^+$) on the curve L_1. The values of these points (S_i) when substituted in the curve L_2 determine their proximity to the point of intersection. The point in S_i with the substituted value closest to zero is the best approximation of the point of intersection.

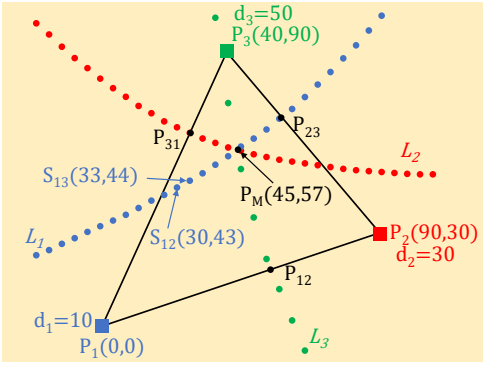


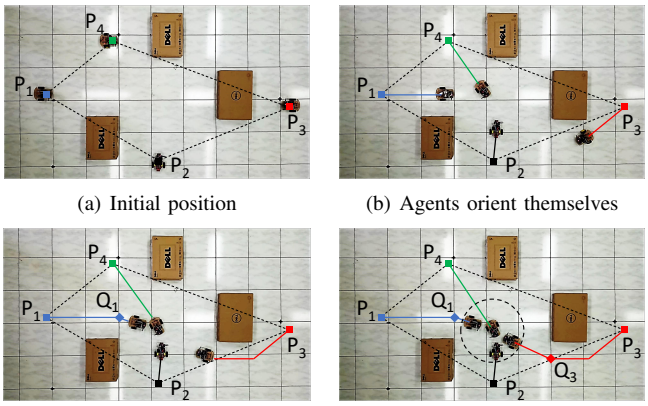
Fig. 7. Numerical aspects in implementation of the algorithms

The selection of sampling distance depends on (i) storage space of the microcontroller and (ii) size of the robots. The 32KB flash memory of ATmega328P allows for a maximum of 4096 points. If the points under consideration range a maximum of $10m$ on X and Y axes, the allowable sampling distance is $(\frac{10\times3}{4096})$ which is $0.7cm$. However, the (robotic) agents used in the experiments have an actual resolution of only $3cm$. We thus have a sampling distance of $3cm$ for all the experiments conducted in the following section.

Fig. 7 illustrates this procedure for the locations and weights given. The three hyperbolas L_1, L_2 and L_3 are converted into collection of points with a uniform separation of $3cm$ on the X-axis. Two sampled points S_{12} and S_{13} are shown in Fig. 7. The point at which its value is closest to zero is the minimax point $(45, 57)$. This compares well with the minimax point $(44.82, 57.47)$ when computed symbolically using MATLAB.

B. Summary of Experiments

We begin with an experiment involving four agents moving amidst three static obstacles. Initial positions of the agents at P_1, P_2, and P_3 are transmitted to the agent at P_4 which then uses **Algorithm *Gathering_point_efficient*** to compute P_G. Appendix B describes briefly the communication aspects. Fig. 8 captures this experiment. The four initial locations are shown in Fig. 8(a). Once the agents have the location of gathering point P_G, they orient themselves towards it as shown in Fig. 8(b). Figures 8(c) and 8(d) show the last turn locations Q_1, Q_3 of agents starting at P_1 and P_3. Gathering of agents is shown in Fig. 8(d).

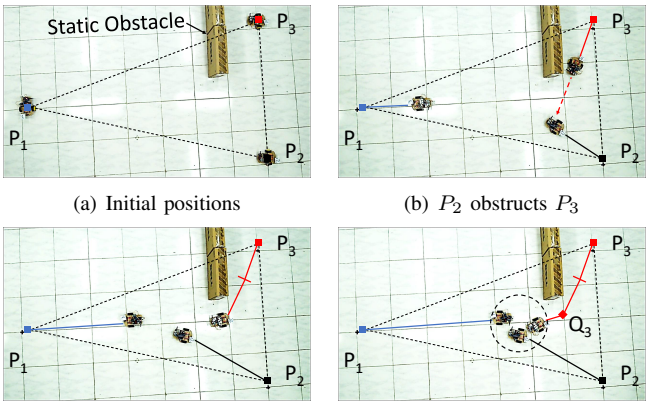


(a) Initial position (b) Agents orient themselves

(c) Reorientation to reach P_G (d) Gathering achieved

Fig. 8. Gathering of four agents in the presence of three static obstacles

We now present another experiment where an agent encounters a fellow agent on its path to P_G. Fig. 9 summarizes this experiment. Initial locations of three agents and one static obstacle are shown in Fig. 9(a). As P_1, P_2 and P_3 start to move towards P_G, the ultrasonic sensor mounted on P_3 detects P_2 as an obstacle (shown in Fig. 9(b)). P_3 halts (and waits) as given by **Algorithm *Path_Following*** until P_2 is no longer an obstacle (Fig. 9(c)) and then continues to move towards P_G. Agents achieve gathering as shown in Fig. 9(d).



(a) Initial positions (b) P_2 obstructs P_3

(c) P_3 resumes moving towards P_G (d) Gathering achieved

Fig. 9. Gathering of three agents while one agent obstructs another's path

Remark 4: As observed from this experiment, an agent detecting another agent as an obstacle is also handled by **Algorithm *Path_Following*** presented in section VI. Distance sensors on the agents facilitate detection of dynamic obstacles and the agent that arrives first at a location common to the path of two agents proceeds first while the other agent waits.

Additionally, we have performed an experiment on gathering of two agents amidst obstacles that makes use of **Algorithm *Gathering_point_pair***. The study has also included experiments with humans moving, with larger number of static obstacles and with multiple agents approaching a location simultaneously.

VIII. Conclusions

A multi-agent gathering problem that is of interest in cyber-physical systems and welfare support systems is studied in this paper. Efficient geometric algorithms are presented for finding the gathering point of two or more agents amidst obstacles minimizing the maximum Euclidean distance of travel of the agents. An efficient hardware realization of the algorithms on multiple small robots is also presented.

We have assumed identical agents in this paper. A natural extension of the problem considered in this paper would be to a scenario where the agents are non-identical and have different velocity profiles. It is worth noting that the gathering point remains the same since it is independent of the velocities of the agents. An interesting problem would then be to compute the point in space (called, for example, as *time_gathering_point*) where the agents take minimum time to arrive while taking their respective velocity profiles into consideration. When dynamic obstacles are involved, the agents could recompute the *time_gathering_point* and move towards it in order to minimize the total time taken for all agents to meet. It would also be of interest to explore the problem of computing a point to gather that would simultaneously optimize the time as well as the distance travelled.

IX. Appendix A

We now present numerical illustrations to compute the weighted minimax point P_M^W as given by Lemmas 2 and 3. In Fig. 3 (a), $P_1(0,0)$ and $P_2(10,0)$ describe locations of agents with weights d_1 and d_2 given by 14 and 2 respectively. The length of line segment P_1P_2, namely l_{12}, is 10 units. Since $d_1 > l_{12} + d_2$, it follows from Lemma 2 that the weighted minimax point P_M^W is the agent location P_1.

In Fig. 3 (b), agents are located at $P_1(0,0)$ and $P_2(10,0)$ with weights d_1 and d_2 given by 2 and 6 respectively. With these values, we have $d_1 < l_{12} + d_2$ and $d_2 < l_{12} + d_1$. It thus follows from Lemma 2 that the weighted minimax point P_M^W is P_{12}. For computation of P_{12}, we treat agent locations P_1 and P_2 as vectors and all other entities (d_1, d_2, l_{12}) as scalars, shown in Eq. (15).

$$\begin{aligned} P_{12} &= \frac{l_{12}(P_1+P_2)+(d_1-d_2)(P_1-P_2)}{2\times l_{12}} \\ \Rightarrow P_{12} &= \frac{10\times(\begin{bmatrix}0\\0\end{bmatrix}+\begin{bmatrix}10\\0\end{bmatrix})+(2-6)\times(\begin{bmatrix}0\\0\end{bmatrix}-\begin{bmatrix}10\\0\end{bmatrix})}{2\times 10} \\ \Rightarrow P_M^W &= P_{12} = \begin{bmatrix}7\\0\end{bmatrix} \end{aligned} \tag{15}$$

In Fig. 3(c), we illustrate the computation of weighted minimax point for three agent locations $P_1(0,0), P_2(9,3), P_3(4,9)$ with weights d_1, d_2 and d_3 given by 1, 3 and 5 respectively. Since $d_i < (l_{ij} + d_j)$ & $d_i < (l_{ki} + d_k)$ $\forall\ i,j,k \in \{1,2,3\}$ with $i \neq j \neq k$, the first condition in Eq. (7) is invalid. Similar computations show that the second condition in Eq. (7) is also invalid. Thus, the weighted minimax point P_M^W is P_{123}.

Locus of points equidistant from two agent locations with their associated weights is a branch of the hyperbola as given by Eq. (8). P_{123} is the point of intersection of three such hyperbolas constructed on the three pairs of agent locations (P_1, P_2), (P_2, P_3) and (P_3, P_1) as shown in Fig. 3(c).

For instance, consider the agent pair $P_1(0,0)$ and $P_2(9,3)$ with their corresponding weights d_1 and d_2 given by 1 and 3 respectively. Let $P(x,y)$ be an arbitrary point equidistant to P_1 and P_2 with weights d_1 and d_2 respectively. The locus of P (branch of a hyperbola) passes through the point $P_{12}(5.45, 1.81)$ and is given by Eq. (16) as shown in Fig. 3(c).

$$\begin{aligned} &\sqrt{(x-0)^2+(y-0)^2}+1 = \sqrt{(x-9)^2+(y-3)^2}+3 \\ &\Rightarrow y = \frac{2\sqrt{43(2x^2-18x+43)}-27x+129}{5} \end{aligned} \tag{16}$$

Similar computations on the remaining two agent pairs lead to two additional hyperbolas that pass through points $P_{23}(5.86, 6.77)$ and $P_{31}(2.81, 6.33)$ as given by Eq. (17).

$$\begin{aligned} y &= \frac{\sqrt{57(4x^2-52x+201)}+30x-3}{32} \\ y &= \frac{36\sqrt{4x^2-16x+81}-72x+729}{130} \end{aligned} \tag{17}$$

The point of intersection of these three branches of hyperbolas is $P_{123}(4.48, 5.75)$ which is the weighted minimax point P_M^W for given agent locations and weights (Fig. 3(c)). An efficient method to implement these computations on a microcontroller is presented in section VII-A.

Appendix B

The agents at $(P_1, P_2, \cdots, P_{k-1})$ transmit their initial locations to the agent at P_k. The latter computes the gathering point P_G and transmits it back to the agents at $(P_1, P_2, \cdots, P_{k-1})$. Each agent then executes **Algorithm *Path_Following*** to gather at P_G.

The communication is implemented as follows. The agents encode their location and index number in the form of a 10-digit constant. The first four digits indicate the X-coordinate of the location while the next four indicate the Y-coordinate (in centimetres). The implementation currently allows 99 agents and hence the last two digits indicate the agent's index number ranging from 01 to 99. Index number 00 is reserved for the gathering point transmitted by the agent at P_k. For example, the encoded information transmitted by agent P_1 located at $(3142, 30)$ to P_k is of the form $\langle 3142003001 \rangle$. Similarly, the encoded form of a gathering point corresponding to $(2718, 43)$ sent by P_k to other agents is $\langle 2718004300 \rangle$. Since this approach involves transmission of k values (locations of $k-1$ agents and one gathering point) each 10 digit in size, the communication complexity [26], [27] is $O(k)$.

References

[1] R. Luo and C. Kuo, "Intelligent seven-DOF robot with dynamic obstacle avoidance and 3-D object recognition for industrial cyber-physical systems in manufacturing automation," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1102–1113, May 2016.

[2] P. Leitão, V. Mařík, and P. Vrba, "Past, present, and future of industrial agent applications," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2360–2372, Nov. 2013.

[3] T. Yamaguchi, S. Mizuno, T. Yoshida, and T. Hashimoto, "Cooperative works for agent robot and human using robot vision based on the model of knowledge, emotion and intention," in *Proc. of 1999 IEEE Int. Conf. on Systems, Man and Cybernetics*, 1999, pp. 987–992.

[4] J. Ichikawa, T. Kon, and N. Shinomiya, "Support system for caregivers with optical fibre sensor and cleaning robot," in *Proc. of 2014 IEEE 3rd Global Conference on Consumer Electronics*, 2014, pp. 636–639.

[5] D. Wang, B. Subagdja, Y. Kang, A. Tan, and D. Zhang, "Towards intelligent caring agents for aging-in-place: issues and challenges," in *Proc. of 2014 IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI)*, 2014, pp. 1–8.

[6] A. Aziz, F. Ahmad, N.Yusof, F. Ahmad, and S. Yusof, "Designing a robot-assisted therapy for individuals with anxiety traits and states," in *Proc. of 2015 IEEE Int. Symposium on Agents, Multi-agent Systems and Robotics (ISAMSR)*, 2015, pp. 98–103.

[7] J. Elzinga and D. Hearn, "Geometric solutions for some minimax location problems," *Transportation Science*, vol. 6, pp. 379–394, 1972.

[8] T. Vicsek, A. Czirok, E. Jacob, I. Choen, and O. Schochet, "Novel type of phase transitions in a system of self-driven particles," *Physical Review Letters*, vol. 75, pp. 1226–1229, 1995.

[9] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[10] R. L. Francis, "Some aspects of a minimax location problem," *Operations Research*, vol. 15, no. 6, 1967.

[11] N. Megiddo, "The weighted Euclidean 1-center problem," *Mathematics of Operations Research*, vol. 8, no. 4, pp. 498–504, 1983.

[12] R. Isaacs, "The best deployment of a naval force in the vicinity of potential trouble spots," *Internal Memorandum (CNA), Center for Naval Analysis, Arlington, VA*, pp. 31–64, 1964.

[13] Z. Drezner and G. Wesolowsky, "Single facility l_p-distance minimax location," *SIAM Journal on Algebraic and Discrete Methods*, vol. 1, no. 3, pp. 315–321, 1980.

[14] E. Wynters and J. Mitchell, "Shortest paths for a two robot rendezvous," in *Proc. of Fifth Canadian Conference on Computational Geometry*, 1993, pp. 216–221.

[15] A. Ganguli, J. Cortés, and F. Bullo, "Multirobot rendezvous with visibility sensors in nonconvex environments," *IEEE Transactions on Robotics*, vol. 25, pp. 340–352, April 2009.

[16] A. Jadbabaie, J. Lin, and A. Morse, "Coordinations of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[17] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[18] W. Ren and R. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[19] D. Han, G. Chesi, and Y. Hung, "Robust consensus for a class of uncertain multi-agent dynamical systems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 306–312, Feb. 2013.

[20] W. Yu, L. Zhou, X. Yu, J. Lu, and R. Lu, "Consensus in multi-agent systems with second-order dynamics and sampled data," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 2137–2146, Nov. 2013.

[21] J. Zhan and X. Li, "Flocking of multi-agent systems via MPC based on position-only measurements," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 377–385, Feb. 2013.

[22] X. Zhang, Q. Han, and B. Zhang, "An overview and deep investigation on sampled data-based event-triggered control and filtering of networked systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 4–16, Feb. 2017.

[23] S. Karnouskos and P. Leitão, "Key contributing factors to the acceptance of agents in industrial environments," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 696–703, Feb. 2017.

[24] J. Hershberger and S. Suri, "An optimal algorithm for Euclidean shortest paths in the plane," *SIAM Journal on Computing*, vol. 28, pp. 2215–22 566, 1999.

[25] H. Rademacher and O. Toeplitz, *The Enjoyment of Mathematics*. Princeton University Press, 1957.

[26] A. C. Yao, "Some complexity questions related to distributed computing," in *Proc. of 11th ACM Symposium on Theory of Computing (STOC)*, 1979, pp. 209–213.

[27] H. Abelson, "Lower bounds on information transfer in distributed computations," *Proc. of the IEEE 19th Annual Symposium on Foundations of Computer Science*, pp. 151–158, 1978.

Bhaskar Vundurthy received his B.E.(Hons.) in Electronics and Instrumentation Engineering from Birla Institute of Technology and Science (BITS), Pilani in 2011. He then received his M.Tech in Electrical Engineering from Indian Institute of Technology (IIT) Madras in 2013. He is currently pursuing his Ph.D in Electrical Engineering at IIT Madras. His research interests include multi-agent systems, robotics, algorithms and computational geometry.

K.Sridharan (S'84-M'96-SM'01) received the Ph.D degree from Rensselaer Polytechnic Institute, Troy, NY in 1995.

He was an Assistant Professor at Indian Institute of Technology (IIT) Guwahati from 1996 to 2001. Since June 2001, he is with IIT Madras where he is presently a Professor. He was a visiting staff member at Nanyang Technological University, Singapore in 2000-2001 and 2006-2008.

Prof. Sridharan has authored several papers, two Springer monographs and holds two patents. He is the recipient of the 2009 Vikram Sarabhai Research Award and the 2011 Tan Chin Tuan Fellowship. He is an Associate Editor of IEEE Transactions on Industrial Electronics.