Robotics: Science and Systems 2017
Cambridge, MA, USA, July 12-16, 2017

Collisions as Information Sources in Densely Packed Multi-Robot Systems Under Mean-Field Approximations[§]

Siddharth Mayya*, Pietro Pierpaoli†, Girish Nair‡ and Magnus Egerstedt*

*School of Electrical and Computer Engineering
Georgia Institute of Technology,
Atlanta, USA
Email: siddharth.mayya, magnus@gatech.edu

†Mechanical and Aerospace Engineering
University of Miami,
Coral Gables, USA
Email: p.pierpaoli@umiami.edu

‡ Department of Electrical and Electronic Engineering
The University of Melbourne,
VIC 3010, Australia
Email: gnair@unimelb.edu.au

***Abstract*—As the spatial scale of robots decrease in multi-robot systems, collisions cease to be catastrophic events that need to be avoided at all costs. This implies that less conservative, coordinated control strategies can be employed, where collisions are not only tolerated, but can potentially be harnessed as an information source. In this paper, we follow this line of inquiry by employing collisions as a sensing modality that provides information about the robots' surroundings. We envision a collection of robots moving around with no sensors other than binary, tactile sensors that can determine if a collision occurred, and let the robots use this information to determine their locations. We apply a probabilistic localization technique based on mean-field approximations that allows each robot to maintain and update a probability distribution over all possible locations. Simulations and real multi-robot experiments illustrate the feasibility of the proposed approach, and demonstrate how collisions in multi-robot systems can indeed be employed as useful information sources.**

I. Introduction

Collision avoidance constitutes one of the key mechanisms for making multi-robot systems operate in a safe and orderly manner – to keep the robots safe, collisions should be avoided at all costs. This certainly makes a lot of sense in a number of applications, where collisions have the potential to be catastrophic, e.g., for fleets of unmanned aerial vehicles or platoons of self-driving trucks. As a consequence, a rich body of work exists on the topic of collision avoidance, and the design methodologies for achieving collision-free multi-robot behaviors typically fall into two camps. The first approach is to make collision avoidance an explicit part of the design objectives, e.g., by minimizing a performance cost subject to hard robot-to-robot separation constraints. This is typically done for smaller teams, where such constraints can be enforced in a computationally feasible manner, e.g., [9, 29]. As the team size increases, however, collision avoidance tends to be ignored when designing primary, coordinated controllers for tasks such as maintaining and achieving formations, covering areas, or patrolling boundary curves, e.g., [7, 19]. But, as collision avoidance is still of paramount importance, additional safety controllers are subsequently wrapped around the coordinated controllers that take over the control authority when collisions are imminent, e.g., [2, 4].

Regardless of the approach, by necessity, collision avoidance becomes the dominant behavior of the system as the robot density increases. This is because more and more robots occupy the same space. Consequently, the robots spend more of their time avoiding each other as opposed to progressing the overall team mission. In this paper, we sidestep this issue completely and embrace collisions as a feature of the multi-robot system. In fact, as momentum is velocity times mass, collisions become less of an issue as the robots get smaller and slower. As a consequence, for truly large teams of small robots, collisions can no longer be considered catastrophic, as observed in [20], which is relevant for a number of miniature swarm robotics testbeds, such as the Kilobot testbed [25] or the Robotarium [23]. This fact is moreover observed in naturally occurring swarms, where "mild" collisions are known to happen, with fish bumping in to each other during rapid schooling maneuvers [21], or among people navigating through crowds [27].

Once one embraces the somewhat unorthodox position that robots are allowed to collide, one can observe that there is actual information to be extracted from the collisions. For example, if a robot collides frequently with other robots, it should be able to infer something about the robot density. This has been observed frequently in biological systems, where swarming animals use collisions with their neighbors to obtain information about their surroundings. Ants frequently engage in head-to-head contact to regulate traffic flow and ensure efficient transport of resources in tunnels and narrow trails [10]. Bees have been shown to use aggregation behaviors and the formation of tight clusters to find the optimal temperature spot over a varying temperature field [13]. By reacting to

[§]This research was sponsored by Grants No. 1531195 and 1544332 from the U.S. National Science Foundation. G. N Nair is supported by Future Fellowship grant FT140100527 from the Australian Research Council.

collisions among the bees in a specific way, the swarm is able to form a single large cluster around the right temperature zone, a feat which cannot be achieved by isolated bees on their own. This bee phenomenon provided inspiration behind the development of a distributed multi-robot algorithm where the robots collectively aggregate near a light source using inter-robot collisions to trigger measurements from on-board light sensors [26]. The experimental setup, outlined in [13], used inter-robot collisions to enable specific spatial formations and allow for collective decision making.

This paper takes these types of informal and anecdotal observations about collisions and investigates whether or not the collisions can be used as sufficiently rich sources of information to enable the robots to localize themselves (in particular environments). To this end, we envision a collection of robots moving around somewhat randomly in a domain with known characteristics, equipped with no other sensors than binary, tactile collision sensors. The domain is assumed to be partitioned into a set of connected cells and the task is for the individual robots to establish what cells they are currently occupying by counting collisions. We develop a probabilistic localization technique that allows each robot to compute a probability distribution over the different cells that tells the robot how likely it is that it is currently occupying a given cell.

To establish such a framework, we need mobility models as well as collision models. And, it should be noted already at this point that some of the mathematical models pursued to this end are somewhat simplistic. However, these simplifications should be understood in light of the underlying ambition to harness the power of collisions, and they can ultimately be justified by showing that they can indeed be used as generators of successful, adaptive localization schemes.

In this paper, we assume that the motion of each robot from one cell to another is derived from a static Markov chain, where the set of cells represents the states of the Markov chain. The Markovian motion model encodes the uncertainty associated with the motion of the each robot, as well as the interactions between the robots which may affect their motion in unpredictable ways. Since each robot does not know its current cell, the states of the Markov chain are hidden. However, each robot is getting some underlying information about the states via the collision measurements. Hence, we have an underlying stochastic process (the Markov chain), which, though directly immeasurable, can be measured indirectly through a second stochastic process (collision measurements). As such, representing the localization problem in a Hidden Markov Model (HMM) [24] framework allows each robot to update the probability distribution over all the states by incorporating sensory collision information at each time step. The emission probabilities of the HMM correspond to the probability of a robot experiencing a collision in a given cell. A *pointwise a posteriori probability* (PMAP) estimator [3] is used to compute the best guess of the robot's current cell based on the updated probability distributions.

The outline of this paper is as follows: Section II uses mean-field approximations and a spatial point process to determine the probability of a robot experiencing collisions. Section III subsequently introduces the Markov chain-based motion model of the robots, and develops the localization algorithm. Simulation results illustrate the efficacy of the proposed algorithm itself for robots satisfying the Markovian model. In Section IV, we implement the algorithm on a team of real mobile robots. Finally, Section V concludes the paper.

II. Collisions Among Robots

In order to use collision information to localize the robots, we first must establish a model for how often collisions occur among the robots. This section introduces both the underlying localization problem and provides estimates for the probability of a robot experiencing a collision at a given time as a function of the distribution of robots over the domain of interest.

Let $\mathcal{N} = \{1, \ldots, N\}$ be an index set of N robots deployed over a compact and connected, planar domain D, which is divided into M connected and non-overlapping cells $D_1, D_2, \ldots, D_M$, such that,

$$D = \bigcup_{j=1}^{M} D_j,$$

with the corresponding environmental index set being $\mathcal{M} = \{1, \ldots, M\}$. Now, assume that the robots move in-between cells, so that at time k, the fraction of robots in cell D_j is denoted by $\mu_j(k)$. We use $\mu(k)$ to denote the vector over all cells, $[\mu_1(k), \mu_2(k), \mu_3(k), \ldots, \mu_M(k)]^T$, and let the initial robot distribution be given by $\mu(1) = \bar{\mu}$.

In each cell, the robots may collide with other robots in that cell. Since the fraction of robots in each cell differs, and so does the size of the cell, the rate of collisions experienced by the robots will be different, as illustrated in Figure 1.

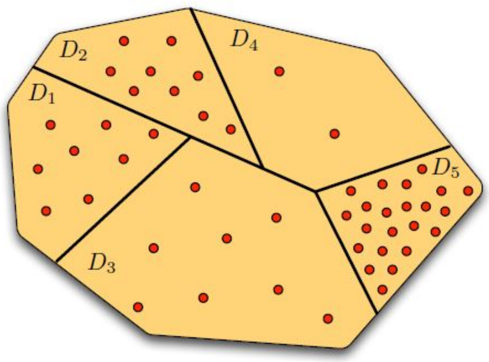


Fig. 1: An illustration of the deployment of N robots over a domain D which is divided into M connected cells D_j, $j = 1, \ldots, M$. Due to the varying number of robots in each cell and different cell sizes, the collision rates in each cell will vary - the higher the density of robots, the more likely it is for collisions to occur.

The path planning literature contains several ways in which the probability of collisions can be calculated under motion and sensory uncertainty e.g., [6, 8, 22]. In many cases, Monte Carlo based techniques have been used to obtain estimates of the collision probabilities, e.g., [16], while the majority of the analytical techniques developed compute the probability of collision along a specific path taken through the workspace in the presence of stationary or moving obstacles, e.g., [15, 30].

Neither of these approaches quite serve the purpose for this paper, where the collisions experienced by a robot in a given cell will depend on the motion of all the other robots in the cell. Luckily, mean-field theory [12, 17] provides an elegant way to simplify such complex interactions by allowing us to replace the effect of other robots on a single robot by an averaged effect. To perform the mean-field approximation, we simply assume that the effect of the other robots on a given robot can be approximated with a single collision probability. Thus, at each time step k, we will compute the probability of a robot experiencing a collision in a particular cell D_j, $j \in \mathcal{M}$, which we will denote by $\phi_j(k)$. (Note that the index here only refers to the cell since this probability is the same for all robots in a given cell under the mean-field approximation.)

To compute $\phi_j(k)$, we let $x_i \in D$ denote the position of Robot i, and let the footprint of each robot be a ball $B(x_i, r)$ centered at x_i with radius r. We assume that the robots are randomly distributed within the cell, and, in order to develop a stochastic collision model that is rich enough to be relevant yet simple enough to be analytically tractable, we make the following assumptions:

1) The size of the domain is significantly larger than the footprint of a single robot, i.e., $|D| \gg r$. This assumption prevents boundary effects from playing any significant role in the collision probability.
2) The point process [5] describing the distribution of the robots in a given cell can be approximated by a spatial Poisson point process [14], with intensity $\lambda_j(k) = \mu_j(k)N/|D_j|$. What this entails is that the expected number of robots inside any subset of D_j is proportional to the fraction of D_j that this subset covers. In other words, the intensity $\lambda_j(k)$ gives the expected number of robots per unit area in cell D_j at time k.
3) Collisions occur when the footprints of the robots overlap, i.e., when $B(x_i, r) \cap B(x_j, r) \neq \emptyset,\ i \neq j$. This is obviously physically not valid, but it both allows for mathematical simplicity yet will, as we will see, yield empirically valid results under Monte Carlo simulation scenarios.

If we say that Robot i is in a given region $\mathcal{R} \subset D_j$ whenever $x_i \in \mathcal{R}$, the spatial Poisson process property immediately tells us that the probability of having n robots in $\mathcal{R}$ (with area $|\mathcal{R}| = A$), at time k, is given by [5],

$$_jP_n(k, A) = \frac{(\lambda_j(k)A)^n e^{-\lambda_j(k)A}}{n!}.$$

Here, the prefix letter j in $_jP_n(k, A)$ indicates the cell D_j that we are interested in.

In other words, if Robot i is dropped down at a random location within cell D_j at time k, the probability of it overlapping spatially with another robot - which we take as a proxy for collision - is equal to the probability of having at least one robot in a circular region $\mathcal{R}$ of radius $2r$ centered at x_i (see Figure 2). Thus, the collision probability in cell D_j at time k is given by,

$$\phi_j(k) = 1 - {}_jP_0(k, A), \tag{1}$$

with

$$\begin{aligned} {}_jP_0(k, A) &= e^{-\lambda_j(k)A} \\ A &= \pi(2r)^2 \\ \lambda_j(k) &= \mu_j(k)N/|D_j|. \end{aligned}$$

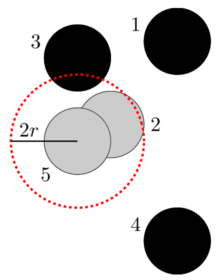


Fig. 2: Robots $1-4$ are placed randomly in a non-overlapping manner, and robot 5 is dropped at a random location. The probability of robot 5 experiencing a collision is equal to the probability of having at least one other robot in a circle of radius $2r$ centered at the robot. The robots depicted in gray have experienced collisions while the black robots have not.

We validate Eqn. (1) by comparing the derived expression for the probability of collision in a given region against a Monte Carlo simulation which computes the collision rate by repeatedly dropping a robot, in a uniformly random manner, into the region and then checking for collisions. The radius of each robot was chosen as $0.01m$ while the domain was a square of length $1m$. The number of robots N was varied between 10 to 120 and for each N, the Monte Carlo simulation was executed for $10,000$ iterations. Figure 3 illustrates that the collision probabilities predicted by the spatial Poisson process indeed reflect the collision rates experienced by the robots despite the physically somewhat dubious assumption about overlapping robots.

As the robots experience collisions while moving from one cell to another, we fix our attention on a particular robot – it does not matter which since they are all identical. Consequently, in the rest of this paper, we will suppress the subscript denoting the identity of the robot. To this end, we associate a binary variable with the collision events experienced by this robot, i.e.,

$$\gamma(k) = \begin{cases} 1 & \text{if the robot experiences a collision at time } k, \\ 0 & \text{otherwise.} \end{cases}$$

The question then becomes, given a string of such observations $\Gamma(k) = [\gamma(k), \ldots, \gamma(1)]$, can the robot figure out which cell it currently is in? In order to answer this question, we need a motion model, which is developed in the next section.

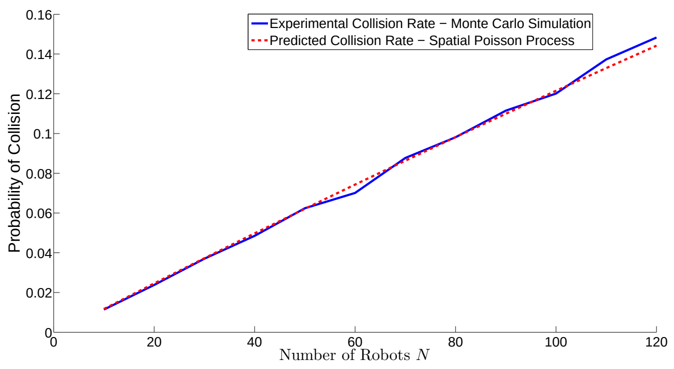


Fig. 3: Validation of the collision probabilities predicted by the spatial Poisson process model by comparing it against the collision rates obtained using a Monte Carlo simulation. The Poisson process probabilities closely match the simulation data for a varying number of robots and serve as good estimates of the true collision probabilities.

III. Collisions Through Hidden Markov Models

A. The Forward Algorithm

In the previous section, we developed estimates for the probability of a robot experiencing collisions in a given cell. In order to utilize this information to localize the robot, we need a mathematical model for the motion of the robots between the cells. Due to the lack of locational information and detailed sensing, as well as the constant interaction between robots, the motion of the robots between cells is stochastic. This lends itself naturally to a Markov chain-based motion model, where the robots move between the cells based on transition probabilities. This approach is commonly used when dealing with uncertainty in robot motion since it efficiently models the stochasticity inherent in the system, e.g., [1, 28, 31]. In this section, we exploit this Markovian motion model and develop a corresponding localization algorithm.

Let the current cell of our chosen robot at time k be denoted by $q(k)$, i.e., if the robot is in cell D_j at time k, then $q(k) = j$. (Again, since all the robots are the same, we suppress the index of the robots in the notation.) The robots move between cells based on a Markov Chain with transition matrix P,

$$P_{ij} = \text{Prob}\Big(q(k+1) = i \mid q(k) = j\Big),\ i,j \in \mathcal{M}. \quad (2)$$

These transition probabilities have to be chosen to reflect the physical accessibility of one cell from another as well as other mobility considerations. In Section III-B, a particular choice of transition probabilities is made but for now, we simply assume that these probabilities are somehow made available to us.

The mean-field approximation means that the fractions of robots inside a given cell satisfy the same Markovian properties, i.e., Eqn. (2) implies that

$$\mu(k+1) = P\mu(k).$$

Note that the states of the Markov chain are hidden, since the robots do not know which cell they are currently in. However, they are making observations in the form of collision measurements over the set $\{\Gamma_1, \Gamma_2\} = \{0, 1\}$. The observation probabilities – the probability of observing a 0 or a 1 in cell D_j – thus becomes

$$G_{lj}(k) = \text{Prob}\Big(\gamma(k) = \Gamma_l \mid q(k) = j\Big), \quad l = 1, 2,\ \ j \in \mathcal{M}.$$

Since the robots are homogeneous, we already know that the probability of observing a collision in cell D_j at time k is $\phi_j(k)$, i.e.,

$$\begin{aligned} G_{1j}(k) &= 1 - \phi_j(k) \\ G_{2j}(k) &= \phi_j(k), \quad j \in \mathcal{M}. \end{aligned}$$

Note that it is rather unusual that the observation probabilities depend on time. This is due to the fact that even though we are focusing our attention on a single robot, all the other robots are moving around at the same time and it is the statistics of this motion that ultimately determines the collision probabilities.

We have now finally obtained the hidden Markov model $\mathcal{H} = (P, G, \bar{\mu})$. This construction allows each robot to compute and update the probability of being in a particular cell at time k given observations up to and including time k.

Let $\delta_k(i)$ denote the probability of the chosen robot being in cell D_i at time k, given the sequence of observations $\Gamma(k) = [\gamma(k), \ldots, \gamma(1)]$. This is given by the posterior probability in the context of the hidden Markov model $\mathcal{H}$,

$$\delta_k(i) = \text{Prob}\Big(q(k) = i \mid \Gamma(k), \mathcal{H}\Big).$$

One way to estimate the current cell of the robot is thus to simply pick the state corresponding to the highest posterior probability,

$$q^*(k) = \underset{i \in \mathcal{M}}{\text{argmax}}\, \delta_k(i).$$

Such an estimator is called the *pointwise maximum a-posteriori probability* (PMAP) estimator [3, 24], since it picks the state with the highest posterior probability. This estimator is *optimally accurate* in the sense that it maximizes the expected number of correct estimates, e.g., [11, 18].

In order to compute $\delta_k(i)$ at each time instant k, we define a variable $\alpha_k(i)$ which denotes the joint probability of obtaining a sequence of observations $\Gamma(k)$ and being in state D_i at time k,

$$\alpha_k(i) = \text{Prob}\Big(\Gamma(k), q(k) = i \mid \mathcal{H}\Big).$$

The probability of obtaining a sequence of observations is now simply the sum of $\alpha_k(i)$ over all the states,

$$\text{Prob}\Big(\Gamma(k) \mid \mathcal{H}\Big) = \sum_{j=1}^{M} \alpha_k(j).$$

Thus, $\delta_k(i)$ can be expressed as,

$$\delta_k(i) = \frac{\text{Prob}(\Gamma(k), q(k) = i \mid \mathcal{H})}{\text{Prob}(\Gamma(k) \mid \mathcal{H})},$$

or, equivalently,

$$\delta_k(i) = \frac{\alpha_k(i)}{\sum_{j=1}^{M} \alpha_k(j)}. \tag{3}$$

We can now use the recursive forward algorithm [24] to compute values of $\delta_k(i)$. Algorithm 1 outlines the steps required to generate the PMAP estimates.

Algorithm 1 PMAP Estimation - The Forward Algorithm

```
1: $k = 1$, $\alpha_1(i) = G_{\gamma(1)i}(1)\bar{\mu}_i$, $i \in \mathcal{M}$
2: while true do
3:     $\delta_k(i) = \frac{\alpha_k(i)}{\sum_{j=1}^{M} \alpha_k(j)}$, $i \in \mathcal{M}$
4:     $q^*(k) = \text{argmax}_{i \in \mathcal{M}} \delta_k(i)$
5:     $\gamma(k+1) \leftarrow \{0, 1\}$
6:     $\alpha_{k+1}(i) = (\sum_{j=1}^{M} \alpha_k(j) P_{ij}) G_{\gamma(k+1)i}$, $i \in \mathcal{M}$
7:     $k = k + 1$
8: end while
```

Step 1 initializes the forward probabilities as the joint probability of being in cell D_i and making an observation $\gamma(1)$. The posterior probabilities are computed in step 3 using Eqn. (3). The PMAP estimator simply picks the state with the highest posterior probability in step 4. Step 6 outlines the induction step of the forward algorithm. The product $\alpha_k(j)P_{ij}$ gives the joint probability of reaching cell D_i at time $k+1$ via cell D_j at time k and making the partial observation $\Gamma(k)$. By summing over all the possible states, $\sum_{j=1}^{M} \alpha_k(j)P_{ij}$ gives us the joint probability of reaching cell D_i at time $k+1$ and making observations $\Gamma(k)$. Finally, $\alpha_{k+1}(i)$ is obtained by multiplying $\sum_{j=1}^{M} \alpha_k(j)P_{ij}$ with $G_{\gamma(k+1)i}$, which takes into account the latest collision measurement $\gamma(k+1)$.

Algorithm 1 assumes that all robots know the initial distribution of robots $\bar{\mu}$, the size of each cell $|D_j|, j \in \mathcal{M}$ and the Markov transition matrix P. There is no communication needed between the robots before or during the execution of the algorithm.

As outlined in Algorithm 1, the posterior probabilities $\delta_k(i), i \in \mathcal{M}$, which represent the probability of a robot being in cell D_i at time k given observations up to and including time k, are updated at each iteration using collision measurements. In order to analyze whether collisions among the robots indeed contain the information content required to perform localization, we study how the entropy of the posterior probability distribution $\delta_k(i), i \in \mathcal{M}$ changes as collision information is used to update it. The entropy of a probability distribution, denoted by E_k, measures of uncertainty associated with it [31], and is given by,

$$E_k = -\sum_{j=1}^{M} \gamma_k(j) \log \gamma_k(j).$$

Thus, lower the entropy of the probability distribution, the more certain the robot is about its estimate $q^*(k)$. In order to analyze how the entropy of the probability distribution changes as collision measurements are made, we consider a simplistic yet illustrative scenario where N robots are distributed randomly over the cells of the domain, however, they do not move between the cells, and simply remain in the same cell for all time $k = 1, 2, \ldots$. This motion can be derived from a Markov chain with a transition probability matrix $P = I_{M \times M}$ where $I_{M \times M}$ is the identity matrix. The task for each robot now is to use collisions in order to estimate the cell it is occupying. This scenario, while simplistic, helps us observe how the entropy of the probability distribution changes as collision measurements are made. Figure 4a shows the results of the simulation which was conducted over 1500 iterations. The cell estimate tracks the true cell of the robot within a few iterations indicating that the robot has localized itself. Indeed, as depicted in Figure 4b, the entropy E_k of the probability distribution $\delta_k(i), i \in \mathcal{M}$ reduces as the algorithm progresses. The reduction in entropy demonstrates how each robot is able to use collisions with other robots to reduce the uncertainty about its current location and thus, localize itself in the domain.

The rest of this section presents simulations where the robots are moving between cells and verifies the feasibility of Algorithm 1.

B. Simulations

In order to illustrate the operations of Algorithm 1, we consider a scenario where N robots are moving around a circular track comprised of M segments each of width w_i, $i \in \mathcal{M}$ (see Figure 5). The Markov matrix P is chosen in such a way that at each time step k, the robot can either remain in the same segment or move to the next segment. Assuming that the transition between cells occurs in only one direction, the width of each segment w_i not only affects the collision probabilities but also the congestion in each cell. We model this congestion by letting the probability of a robot staying in the same cell be inversely proportional to the width of the cell: the narrower the segment, the higher is the probability of getting stuck,

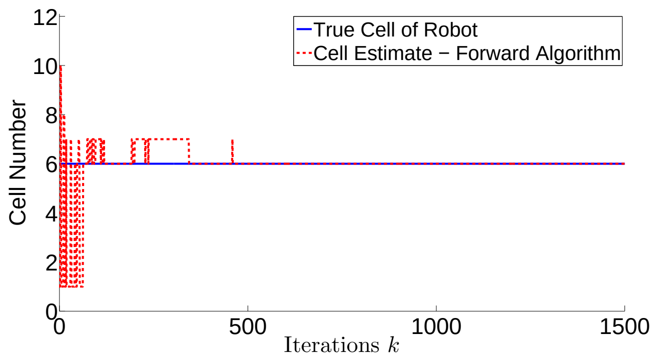
$$P(i,i) = \epsilon \frac{1}{w_i}, P(i+i,i) = 1 - P(i,i), i = 1, .., M-1$$
$$P(M,M) = \epsilon \frac{1}{w_M}, P(1,M) = 1 - P(M,M),$$

where ϵ is chosen so as to restrict the probabilities between 0 and 1.

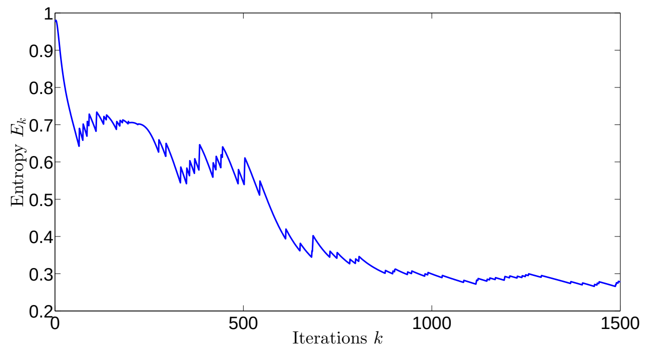
The Poisson parameter $\lambda_j(k)$ for each cell D_j is computed, as before, as the average number of robots per unit area in the cell,

$$\lambda_j(k) = \frac{\mu_j(k)N}{|D_j|}.$$

In the simulation scenario, we develop a circular track with 10 individual cells of widths $[0.2, 0.17, 0.16, 0.11, 0.09, 0.13, 0.15, 0.24, 0.26, 0.28]$ and $\epsilon = 0.07$ (Figure 5). The simulation is conducted over 300 iterations. The robots are initially placed into different cells according to the distribution $\bar{\mu}$, and are then moved between the cells based on the Markov chain probabilities. The



(a) Localization Results



(b) Entropy Reduction

Fig. 4: Illustration of the probabilistic localization technique for a single robot in a team of N robots distributed with initial distribution $\bar{\mu}$ over a domain. In this specialized scenario, the robots do not move between the cells and simply remain in the same cell. Each robot uses collisions with other robots to update the probability distribution over the states. As seen in Figure 4a, the robot is able to correctly localize itself in the domain. This corresponds to a reduction in the entropy of the posterior probability distribution $\gamma_k(i), i \in \mathcal{M}$ of the robot, as shown in Figure 4b. The reduction in entropy illustrates how collisions can indeed be used as sources of information to reduce the uncertainty regarding a robot's position in the environment and thus can be used for localization.

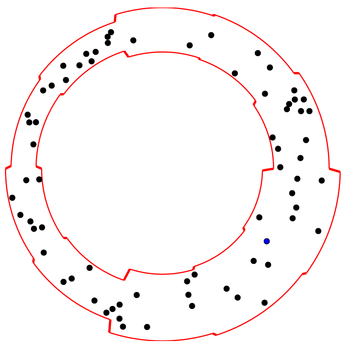


Fig. 5: A circular track with 10 cells of varying widths. The robots have been distributed randomly based on an initial distribution $\bar{\mu}$.

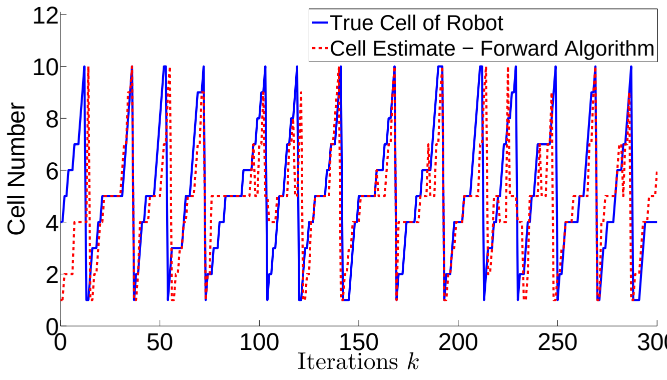


Fig. 6: Simulation results of the probabilistic localization technique for a single robot in a team of 45 robots moving in a 10-cell domain. As seen, the estimates closely track the true cell of the robot.

location of a robot within a cell, is determined by dropping it at a uniformly random location within the cell. As described in Section II, if the robot spatially overlaps with any of the other robots present in the cell, a collision is recorded.

Figure 6 shows the true cell and the cell estimates for a randomly chosen robot in the team of 45 robots traversing the circular track. As seen, using only collision measurements, the PMAP estimator generates estimates which closely track the true cell of the robot as the robot moves around the domain.

Figure 7 quantitatively analyzes the performance of the localization algorithm by illustrating the distribution of the localization error over the simulation time. The localization error, computed as the difference between the true cell index of the robot $q(k)$, and its PMAP estimate $q^*(k)$, is an integer, with 0 implying perfect localization, $\{+1, -1\}$ implying that the cell adjacent to the true cell was picked, etc. As seen, for majority of the time, the estimator either performs perfect localization, or localizes the robot to a cell adjacent to the true one.

IV. Experimental Results

Until now, we considered a discrete-time motion model for the robots, where each robot transitioned between cells at discrete time steps. In order to test the collision algorithms on teams of actual robots, this has to be modified to a continuous-time setting. To this end, let $x \in D$ denote the position of a chosen robot – again, we suppress the index of the robot, since all the robots are identical. The planar robots are assumed to move according to the following dynamics,

$$\dot{x} = u,$$

where u is the applied control velocity.

Since the robots do not know where they are based on traditional sensing modalities, we simply let each robot travel in a randomly chosen direction for a fixed duration of time T, at which point another random direction is selected. Essentially, each robot is performing a random walk where the direction of motion is changed at regular intervals of time $t_k = kT$, $k = 1, 2, \ldots$. The random walk of the robots implies that

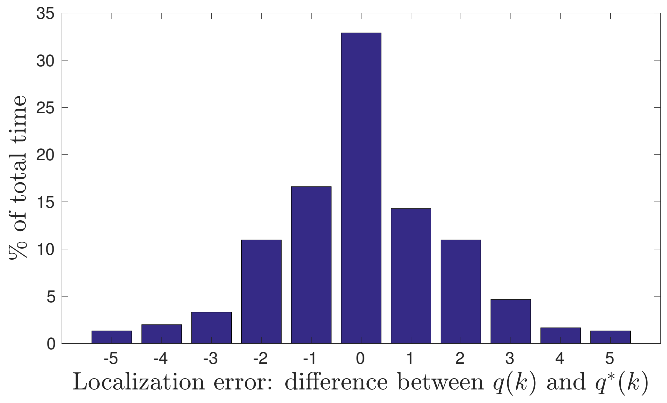


Fig. 7: Distribution of the quantized localization error for the simulation described in Section III-B. The localization error is the difference in the cell index between the true cell of the robot $q(k)$ and its PMAP estimate $q^*(k)$. As seen, the estimator perfectly localizes the robot 33% of the time. Note that a localization error of 5 and -5 correspond to the same cell (see Figure 5).

transitions between cells is stochastic in nature. Consequently, we model these transitions as a particular realization of a Markov chain with transition probability matrix P, where the states of the Markov chain are the different cells of the domain. Similar to approaches used in [1, 28, 31], we generate estimates of the transition probability matrices by conducting repeated simulations of the robots executing this random walk motion. At regular intervals of time t_k, the cell occupied by each robot is noted, and this information is used to generate the transition probability matrix P. Thus, a discrete Markov chain, evaluated at discrete times t_k, is used as a simplified and abstract representation of the motion of the robots between the cells. While this may seem like an over-simplification of the motion of the robots, it is sufficiently rich for the purposes of estimating the current cell that each robot is occupying, as will be seen later in this section.

Let $q(t_k)$ denote the index of the cell occupied by the robot at time t_k. The distribution of robots across the cells $\mu(t_k) = [\mu_1(t_k), \mu_2(t_k), \ldots, \mu_M(t_k)]^T$ evolves as,

$$\mu(t_{k+1}) = P\mu(t_k).$$

Similar to the previous section, the states of the Markov chain are hidden since the robots do not know which cells they occupy. Consequently, let $c(t)$ denote a binary variable associated with the collision of the chosen robot at continuous time t,

$$c(t) = \begin{cases} 1 \text{ if the robot experiences collision at time } t, \\ 0 \text{ otherwise} \end{cases}$$

The observations from the Markov chain thus becomes

$$\gamma(t_k) = \begin{cases} 1, \text{ if} \displaystyle\int_{t_k}^{t_{k+1}} c(t)dt > 0, \\ 0, \text{ otherwise.} \end{cases}$$

As a result, $\gamma(t_k)$ encodes whether the robot experienced any collisions over the time interval $[t_k, t_{k+1})$, and can be used as the observation variable in the hidden Markov model.

These constructions give us both the transition probability matrix P and the emission probability matrix G. As a result, we have constructed the hidden Markov model as $\mathcal{H} = (P, G, \bar{\mu})$. Proceeding as described in Section III, we can thus utilize the forward algorithm to obtain estimates $q^*(t_k)$ of the cell that the robot is occupying at time t_k.

The probabilistic localization algorithm was implemented on actual, physical robots on the Robotarium [23], which is a remotely accessible multi-robot testbed where code is uploaded via a web-interface and the experimental data is returned after the experiments. The experiments involved 12 differential-drive mobile robots traversing a simulated track with 4 cells of varying widths (see Figure 8). Each robot executed a random walk as described earlier in this section, and used the collision information for localizing itself on the track. As can be seen in Figure 8, the collisions involve physical contact between real robots and have not been simulated in any way.

Figure 9 shows the result of the localization algorithm for an experimental run of 600 seconds. As can be seen, after a few seconds, the estimate tracks the true cell that the robot is occupying. Similar to the analysis in Section III-B, Figure 10 illustrates the distribution of the localization error over the experiment time window.

V. Conclusions

In this paper, we argue that for certain classes of multi-robot systems, collisions can not only be considered as a feature of the multi-robot system, but can also be viewed as a sensing modality. We envision a team of robots moving around equipped with only binary tactile collision sensors, and use the information derived from these sensors to localize the robots in the domain. To do this, we first use mean-field approximations to compute the probability of a robot experiencing collisions in the different regions of the domain. Using this information, each robot maintains and updates a probability distribution over all the possible regions in the domain which tells the robot how likely it is that it is currently occupying a given region. Collision measurements are incorporated via a hidden Markov model framework to update this probability distribution, and a *pointwise maximum a posteriori* (PMAP) estimator is used to obtain the best guess of the robot's current region.

To the effect of providing an information-theoretic validation of this idea, we show that the entropy of the probability distribution reduces as collision measurements are used to update the probability distribution, implying that inter-robot collisions indeed contain information which can be used by a robot to reduce the uncertainty regarding its current location. After verifying the functionality and performance of the proposed probabilistic localization technique in simulation, we conduct multi-robot experiments involving actual mobile robots. In these experiments, collisions involve physical contact between the robots, in-line with our belief that for large

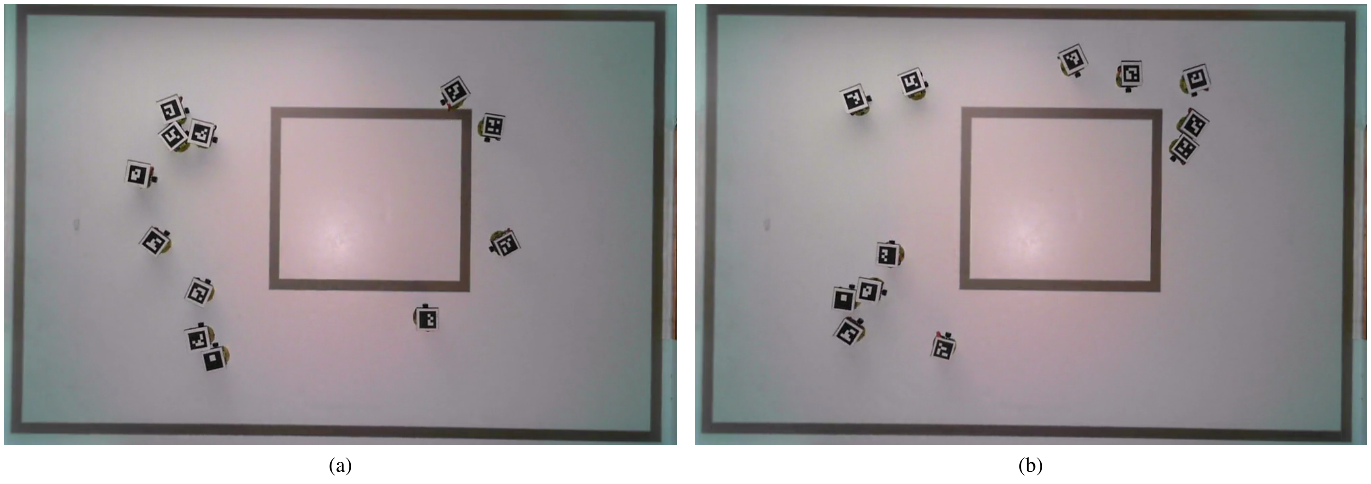


Fig. 8: Experimental Setup: 12 mobile robots are traversing a rectangular track on the Robotarium - a remotely accessible multi-robot testbed [23]. Collisions among the robots involve actual physical contact and have not been simulated in any way. Each robot executes a random-walk algorithm, and using only collision information, generates estimates of the current cell that it is occupying.

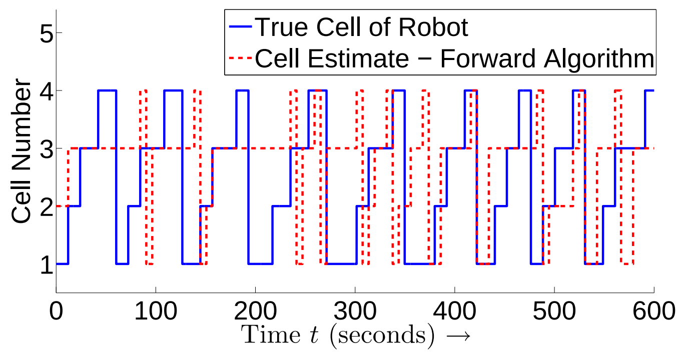


Fig. 9: Experimental results of the localization algorithm for 12 mobile robots operating on a multi-robot testbed.

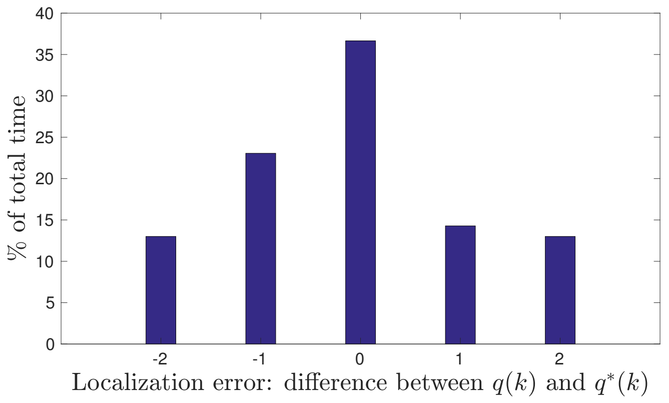


Fig. 10: Distribution of the quantized localization errors for the experiment described in Section IV. The localization error is the difference in the cell index between the true cell of the robot $q(k)$ and its PMAP estimate $q^*(k)$. As seen, the estimator perfectly localizes the robot 36% of the time. Note that a localization error of 2 and -2 correspond to the same cell (see Figure 8).

teams of smaller robots, collisions are no longer catastrophic, and in fact, can allow robots to gather information about their surroundings.

References

[1] Sean B Andersson and Dimitrios Hristu. Symbolic feedback control for navigation. *IEEE Transactions on Automatic Control*, 51(6):926–937, 2006.

[2] Ronald C Arkin. *Behavior-based robotics*. MIT press, 1998.

[3] Lalit Bahl, John Cocke, Frederick Jelinek, and Josef Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Transactions on information theory*, 20(2):284–287, 1974.

[4] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27):68–73, 2015.

[5] Sung Nok Chiu, Dietrich Stoyan, Wilfrid S Kendall, and Joseph Mecke. *Stochastic geometry and its applications*. John Wiley & Sons, 2013.

[6] S.A.M Coenen, J.J.M Lunenburg, M.J.G van de Molengraft, and Maarten Steinbuch. A representation method based on the probability of collision for safe robot navigation in domestic environments. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4177–4183. IEEE, 2014.

[7] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[8] Noel E Du Toit and Joel W Burdick. Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics*, 27(4):809–815, 2011.

[9] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25 (1):116–129, 2002.

[10] Nick Gravish, Gregory Gold, Andrew Zangwill, Michael AD Goodisman, and Daniel I Goldman. Glass-like dynamics in confined and congested ant traffic. *Soft matter*, 11(33):6552–6561, 2015.

[11] Ian Holmes and Richard Durbin. Dynamic programming alignment accuracy. *Journal of computational biology*, 5 (3):493–504, 1998.

[12] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[13] Serge Kernbach, Ronald Thenius, Olga Kernbach, and Thomas Schmickl. Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system. *Adaptive Behavior*, 17(3):237–259, 2009.

[14] John Frank Charles Kingman. *Poisson processes*. Wiley Online Library, 1993.

[15] Eckhard Kruse, Ralf Gutsche, and Friedrich M Wahl. Estimation of collision probabilities in dynamic environments for path planning with minimum collision probability. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems' 96*, volume 3, pages 1288–1295. IEEE, 1996.

[16] Alain Lambert, Dominique Gruyer, and Guillaume Saint Pierre. A fast Monte Carlo algorithm for collision probability estimation. In *10th International Conference on Control, Automation, Robotics and Vision, 2008*, pages 406–411. IEEE, 2008.

[17] Jean-Yves Le Boudec, David McDonald, and Jochen Mundinger. A generic mean field convergence result for systems of interacting objects. In *Fourth International Conference on the Quantitative Evaluation of Systems, 2007*, pages 3–18. IEEE, 2007.

[18] Jüri Lember and Alexey A Koloydenko. Bridging Viterbi and posterior decoding: a generalized risk approach to hidden path inference based on hidden Markov models. *Journal of Machine Learning Research*, 15(1):1–58, 2014.

[19] Mehran Mesbahi and Magnus Egerstedt. *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[20] Yash Mulgaonkar, Gareth Cross, and Vijay Kumar. Design of small, safe and robust quadrotor swarms. In *IEEE International Conference on Robotics and Automation (ICRA), 2015*, pages 2208–2215. IEEE, 2015.

[21] Werner Nachtigall and Alfred Wisser. *Bionics by examples*. Springer, 2014.

[22] Sachin Patil, Jur Van Den Berg, and Ron Alterovitz. Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA), 2012*, pages 3238–3244. IEEE, 2012.

[23] Daniel Pickem, Li Wang, Paul Glotfelter, Yancy Diaz-Mercado, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. Safe, remote-access swarm robotics research on the robotarium. *arXiv preprint arXiv:1604.00640*, 2016.

[24] Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[25] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[26] Thomas Schmickl, Ronald Thenius, Christoph Moeslinger, Gerald Radspieler, Serge Kernbach, Marc Szymanski, and Karl Crailsheim. Get in touch: cooperative decision making based on robot-to-robot collisions. *Autonomous Agents and Multi-Agent Systems*, 18(1):133–155, 2009.

[27] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 19–28. ACM, 2005.

[28] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *IJCAI*, volume 95, pages 1080–1087, 1995.

[29] Claire Tomlin, George J Pappas, and Shankar Sastry. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on automatic control*, 43(4):509–521, 1998.

[30] Jur Van Den Berg, Pieter Abbeel, and Ken Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

[31] Francesco Zanichelli. Topological maps and robust localization for autonomous navigation. In *IJCAI Workshop on Adaptive spatial representations of dynamic environments*. Citeseer, 1999.