Optimization and Evaluation of a Multi Robot Surface Inspection Task Through Particle Swarm Optimization

Darren Chiu[1], Radhika Nagpal[2], and Bahar Haghighat[3]

***Abstract*— Robot swarms can be tasked with a variety of automated sensing and inspection applications in aerial, aquatic, and surface environments. In this paper, we study a simplified two-outcome surface inspection task. We task a group of robots to inspect and collectively classify a 2D surface section based on a binary pattern projected on the surface. We use a decentralized Bayesian decision-making algorithm and deploy a swarm of 3-cm sized wheeled robots to inspect a randomized black and white tiled surface section of size $1m \times 1m$ in simulation. We first describe the model parameters that characterize our simulated environment, the robot swarm, and the inspection algorithm. We then employ a noise-resistant heuristic optimization scheme based on the Particle Swarm Optimization (PSO) using a fitness evaluation that combines the swarm's classification decision accuracy and decision time. We use our fitness measure definition to asses the optimized parameters through 100 randomized simulations that vary surface pattern and initial robot poses. The optimized algorithm parameters show up to 55% improvement in median of fitness evaluations against an empirically chosen parameter set.**

I. INTRODUCTION

Automated robotic inspection can serve many applications such as maintenance of bridges, wind turbines, oil and gas pipelines, and aerospace infrastructure [1], [2], [3], [4]. In many of these instances, the inspection task takes the form of a binary classification problem. The classification goal is to determine the state of an inspected surface area as "desirable" or "undesirable" based on spatially-distributed surface features. Multiple benefits can be expected from deploying swarms of mobile robots to these types of tasks. Compared to single-robot systems, swarms are resilient to failure of individuals. Compared to fixed sensor networks, robot swarms provide dynamic and flexible coverage performances [5]. In large-scale swarms, low-cost operations necessitates minimizing the cost and complexity of individual robots. This promotes the cause of swarms of miniaturized robots that employ computationally inexpensive algorithms.

The two-outcome swarm robotic surface inspection task that we consider in this work requires two main enabling mechanisms for the decision making: (i) a mechanism for the swarm to share and integrate observations made by individual robots, i.e., a sensor fusion mechanism, and (ii) a consensus mechanism to form a final classification decision out of the individual decisions, i.e., a decision fusion mechanism. A variety of distributed decision-making algorithms including Bayesian [6], [7], [8], [9], and bio-inspired [10], [11], [12], [13], [14] algorithms have been employed in studies involving robot swarms. Compared to bio-inspired algorithms, Bayesian algorithms provide a statistically grounded approach for integrating the observations and decisions of the individual robots. In a previous contribution, a Bayesian approach was demonstrated in a swarm of abstract agents tasked with classifying a monochrome environment [9]. Here, we build upon that work in three ways. First, we extend the algorithm presented in [9] by introducing a hysteresis parameter that defines a minimum observation criteria around the probability threshold before a robot updates its decision. Second, we adapt the simulation implementation from point mass dynamics to physically modeled wheeled robots by simulating proximity sensing and collision avoidance and present experiments within the physics-based Webots robotic simulator. Finally, we use a Particle Swarm Optimization (PSO) scheme and develop an automated optimization framework that leverages our Webots simulation to heuristically find a set of optimized algorithm parameters with improved decision-making time and accuracy.

This work strives to set a step towards developing algorithmic and hardware tools that support sensing tasks by robot swarms. We anticipate that (i) swarms of small-scale robots have the potential to deliver a variety of real world sensing and inspection applications, and that (ii) the statistically grounded Bayesian decision-making framework has the flexibility to be extended and applied to a broad class of spatially-distributed feature classification tasks by swarms.

II. PROBLEM DEFINITION

We define the problem that we set out to undertake as the following. A group of N robots must complete a binary classification task based on perceiving a spatially distributed feature spread over a bounded 2D surface section. A black and white binary pattern representing the spatially distributed feature is projected on the surface. The pattern's fill ratio determines the proportion of the white-colored area in the overall pattern surface area. The robots each individually inspect the surface, share their information with the rest of the swarm, and collectively determine whether the surface is covered with a majority white or a majority black pattern.

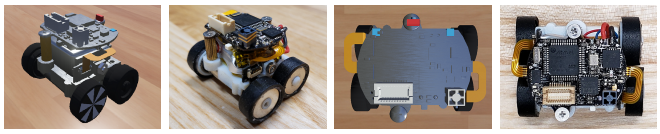
The class of real-world inspection problems that underlies our abstract problem definition here is characterized by three main features: (i) the need to inspect a bounded surface

This work was supported by a Swiss National Science Foundation (SNSF) postdoctoral fellowship award P400P2_191116 and an Office of Naval Research (ONR) grant N00014-22-1-2222 .

[1]Darren Chiu is with University of Southern California, Los Angeles, CA 90089, USA chiudarr@usc.edu

[2]Radhika Nagpal is with Princeton University, Princeton, NJ 08544, USA rn1627@princeton.edu

[3]Bahar Haghighat is with University of Groningen, Groningen, 9747 AG, Netherlands bahar.haghighat@rug.nl



(a) Simulation, side (b) Real robot, side (c) Simulation, top (d) Real robot, top

Fig. 1: We use a realistic model of the 4-wheeled Rovable robot in our simulation experiments [16]. The real Rovable robot (b,d) and its simulation model created in Webots (a,c) have similar physical properties. The robots adhere to ferromagnetic surfaces using their magnetic pincher-wheels. For scale, each wheel on the robot is 12mm in diameter.

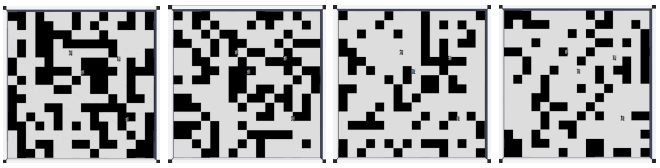
environment, (ii) the need to deliver a binary classification decision for the inspected environment, and that (iii) the feature that informs the classification decision is spatially distributed in the environment where the robots operate.

III. Simulation and Optimization Framework

Our simulation framework serves as the virtual environment in which we study the operation of our inspecting robot swarm. We use the Webots robotic simulator [15]. Within Webots, we have three main components: (i) a realistic robot model of a 3-cm sized 4-wheeled robot that inspects a target surface section, (ii) the target surfaces that the robots traverse to inspect with a black and white pattern projected onto them, and (iii) a (supervisor controller) script that collects robot positions, sensor measurements, observations, beliefs, and decisions. The robot model, shown in Figure 1, is based on the real Rovable robot [16]. Originally designed as a mobile wearable robot, Rovables are capable of wireless communication and low-power localization using their wheel encoders and on-board IMUs for inertial-based navigation [16]. Within Webots, the Rovable proto file captures the physical properties of the real robot, such as mass, center of mass, and surface contact properties. Figure 2 shows our overall Webots simulation world with four Rovable robots on patterned surfaces of size $1m \times 1m$ with different fill ratios.

Our optimization framework consists of two main components: (i) our Webots simulation world described above and (ii) a PSO-based optimization scheme where each particle in the PSO swarm is an instance of our simulated world with a specific set of algorithm parameters. The optimization goal is to find a set of algorithm parameters that enable the robots to classify the environment with speed, accuracy, and consistency. Pugh et al. showed that PSO could outperform Genetic Algorithms on benchmark functions and in certain scenarios of limited-time learning with presence of noise [17], [18]. This motivates the choice of PSO here since stochasticity is inherent to the approach employed by the robots. Multiple computationally efficient noise-resistant versions of PSO have been proposed [19], [20]. We evaluate each particle multiple times and consider a combination of average and standard deviation of observed performances. Each evaluation generates new floor patterns and robot poses.

In the PSO swarm, the velocity of particle i in dimension j is determined by three components: (i) the particle's velocity



(a) $f = 0.55$ (b) $f = 0.6$ (c) $f = 0.7$ (d) $f = 0.8$

Fig. 2: Within the Webots robotic simulator, we simulate a bounded 2D surface section of size $1m \times 1m$ with a monochromatic randomized pattern. The proportion of white in the pattern, denoted as f, is the environment fill ratio. The robots have to classify the environment as mostly white ($f > 0.5$) or mostly black ($f < 0.5$). Environments with fill ratios close to $f = 0.5$ are hardest to classify. There are a total of 256 squares in the arena, 16 squares along each side.

at the previous iteration weighted by an inertia coefficient w, (ii) a randomized attraction to the particle's own personally best visited location over previous iterations $x^*_{i,j}$ weighted by w_p, and (iii) a randomized attraction to the particle's neighborhood's best visited location over the previous iterations $x^*_{i',j}$ weighted by w_n (Equation 1a). r_1 and r_2 are random numbers drawn from a uniform distribution between 0 and 1. Attractions are determined through a fitness evaluation (Algorithm 2), discussed in Section V, that rewards fast accurate consistent decisions. Particle positions are then set at each time step using the updated velocity (Equation 1b).

$$v_{i,j} := w \cdot v_{i,j} + w_p \cdot r_1 \cdot (x^*_{i,j} - x_{i,j}) + w_n \cdot r_2 \cdot (x^*_{i',j} - x_{i,j}) \quad \text{(1a)}$$

$$x_{i,j} := x_{i,j} + v_{i,j} \quad \text{(1b)}$$

The total execution time for the PSO optimization process depends on four factors: (i) population size (N_p), (ii) individual candidate evaluation time (t_e), (iii) number of iterations of the algorithm (N_i), and (iv) number of re-evaluations of each candidate particle position within the same iteration (N_{re}). We evaluate all particles and their re-evaluations in parallel at each iteration, as a result the total time we get for the optimization procedure is as below:

$$t_{total} = t_e \cdot N_i \quad (2)$$

The individual candidate evaluation time t_e depends directly on the complexity of the simulation model utilized and its corresponding computational load.

IV. Inspection Algorithm

The structure of our proposed algorithm is shown in Algorithm 1. The algorithm enables simulated Rovable robots to inspect black and white 2D environments and classify the fill ratio as being above or below a predefined threshold level. Each robot in the swarm maintains a Bayesian model of the fill ratio that it updates using new self made observations and incoming observations from other robots. Each robot individually forms its decision about how to classify the fill ratio using a predefined credibility threshold and the posterior distribution. The robots make binary observations based on

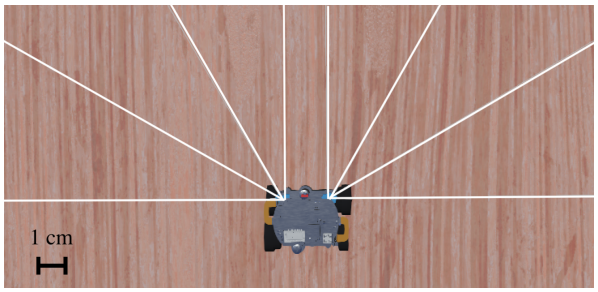


Fig. 3: Simulated Rovable with time-of-flight distance sensors, placed at 30 degree increments. A distance sensor is triggered by the parameter d given in millimeters.

their location in the environment, through an externally simulated supervisor. We model the binary color observations $C \in \{0, 1\}$ as drawings from a Bernoulli distribution. The fill ratio $f \in [0, 1]$ is unknown to the robots and is modeled as a Beta distribution that is updated based on the incoming color observations. The Beta distribution is initialized with parameters α_0 and β_0 as $Beta(\alpha = \alpha_0, \beta = \beta_0)$, where α_0 determines how regularizing the prior distribution is.

$$C \sim Bernoulli(f) \tag{3a}$$

$$f \sim Beta(\alpha, \beta) \tag{3b}$$

$$f \mid C \sim Beta(\alpha + C,\ \beta + (1 - C)) \tag{3c}$$

The robots start inspecting the environment at randomized initial locations and orientations and explore the environment using a random walk while performing collision avoidance. Our random walk is implemented by drawing from a Gaussian distribution that is upper bounded by a parameter s, where s defines the number of time steps the robot moves forward before taking a random turn. Each robot then turns by sampling a random angle drawn from a Gaussian distribution upper bounded by $\pi/2$. The collision avoidance is implemented using eight simulated time of flight sensors from the Webots library placed around the front of the robot (Figure 3). We introduce a parameter d $[mm]$ that defines the minimum sensed distance before a robot enters the collision avoidance state; d applies to all eight sensors equally.

The robots make observations at regular intervals every τ simulation time steps from their location in the environment, update their posterior distribution as in Equation 3c, and broadcast their observation. The robots pause to sample the environment for 5 simulation steps. Upon receiving a radio message containing a new observation from an emitting robot, a receiving robot updates its posterior as it would using its own observations. This forms the decision fusion mechanism mentioned in Section I in two configurations: with and without positive feedback. When positive feedback is false (u^-), robots broadcast their most recent observation. When positive feedback is true (u^+), current decision is broadcasted instead. If no decision was made then the most recent observation is sent. After every posterior update, a robot also updates its classification decision using the

Algorithm 1 Bayesian Inspection Algorithm

Input: $T, \alpha_o, \beta_o, \tau, f, r, h, p_c$
Output: $d_f \in \{0, 1\}$

$\alpha \leftarrow \alpha_o$ ▷ Initialize alpha
$\beta \leftarrow \beta_o$ ▷ Initialize beta
$d \leftarrow -1$ ▷ Initialize incomplete decision flag
while $t \leq T$ **do**
  Perform Random Walk for s Time Steps
  if τ *divides* t **then**
    $Pause()$ ▷ Stop all movement for 40ms
    $C \leftarrow$ Observed Color
    $\alpha \leftarrow \alpha + C$
    $\beta \leftarrow \beta + (1 - C)$
  end if
  $C' \leftarrow$ Message Color
  $\alpha \leftarrow \alpha + C'$
  $\beta \leftarrow \beta + (1 - C')$
  $p \leftarrow Beta(\alpha, \beta, 0.5)$
  if $d_f \neq d_{f,t-1}$ **then**
    if $p \geq p_c$ and $hysteresis$ **then**
      $d_f \leftarrow 0$ ▷ Decision majority black
    else if $(1 - p) > p_c$ and $hysteresis$ **then**
      $d_f \leftarrow 1$ ▷ Decision majority white
    end if
  end if
  if $d_f \neq -1$ and u^+ **then** ▷ Broadcast decision
    Broadcast d_f
  else ▷ Broadcast observation
    Broadcast C
  end if
  $t \leftarrow t + 1$
end while

predefined threshold θ and a credibility threshold p_c. The credibility threshold p_c is defined as the probability mass of the posterior distribution that must lie on one side of the predefined threshold $\theta = 0.5$ for the classification decision. In particular, if the posterior cumulative distribution p at $f = 0.5$ is greater than or equal to the credibility threshold p_c then the classification decision may be set to 0 (majority black) or otherwise set to 1 (majority white).

$$d_f = \begin{cases} 0 & p(f = 0.5) \geq p_c \\ 1 & p(f = 0.5) < p_c \end{cases} \tag{4}$$

A hysteresis criterion must be met before setting the decision. This is determined using the hysteresis parameter denoted as h. The hysteresis parameter defines the minimum number of observations that must have been made after the posterior cumulative distribution, p, satisfies the credibility threshold, p_c. A non-zero hysteresis parameter enforces that only after h observations have been made and p continues to satisfy the condition described in Equation 4, the classification decision may be updated correspondingly. This is shown in Equation 5 where o_i defines the number of observations by robot i after the condition in Equation 4 is satisfied and

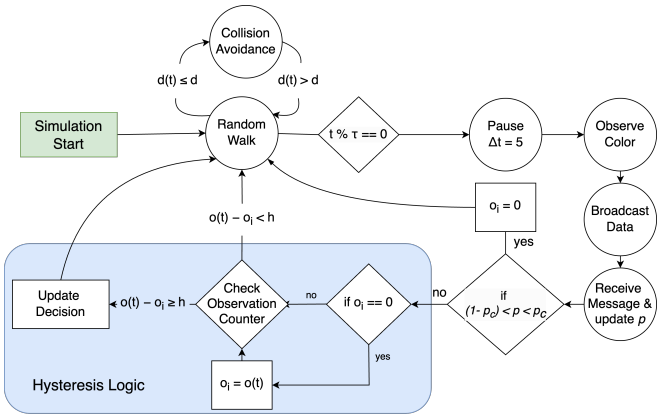


Fig. 4: The finite state machine governing a robot's behavior. $d(t)$ indicates the measured distance from any sensor at time t in mm. $o(t)$ tracks the total observations made up until simulation time step t. o_i stores the initial number of observations once the credibility threshold is passed, for the purpose of calculating the hysteresis criterion. h indicates the hysteresis parameter. p_c is the credibility threshold.

$o(t)$ tracks the total number of observations made at time step t. Note that as new observations become available to a robot, its posterior cumulative distribution gets updated and the criteria defined in Equation 4 may no longer be satisfied. In such a case, o_i is reset to 0 and the hysteresis is broken.

$$hysteresis = \begin{cases} 1 & o(t) - o_i \geq h \\ 0 & o(t) - o_i < h \end{cases} \tag{5}$$

We implement Algorithm 1 through the six-state finite state machine shown in Figure 4, where the states are denoted as circles, variable assignments as rectangles, and conditionals as diamonds. Each simulation step is $8ms$ long.

V. Experiments

We used the Amazon Web Services (AWS) cloud platform for running the Webots simulations and the optimization process. Each simulation instance was launched on a 4-core CPU with 8GB of RAM and ran for a maximum of 60 minutes in simulation time, $T = 3600s$.

The PSO parameters search space needs to be bounded before the PSO search is launched. In the following, we motivate the boundaries of our search space by considering physical features of the simulated arena. We bound the observation interval parameter, τ, by deriving the number of simulation time steps needed for the robot to cross one colored square along one side, denoted as t_s. The random forward parameter, s, is bounded by deriving the time steps required to cross the arena along one side, denoted as t_a. The collision minimum distance parameter, d, was bounded by the range of the simulated sensors. The hysteresis parameter, h, was bounded by the number of squares in the arena, corresponding to the minimum number of attainable samples from the entire arena if a robot travelled one tile per sample. The lower and upper bounds of τ and s were then expanded by an arbitrarily chosen factor of 5 with the presumption that the optimal parameter would lie within these bounds. We assumed a perfect physics model of the robot that traverses the arena at a constant speed (2.77 cm/s).

Algorithm 2 Fitness Evaluation

$f_i \leftarrow 0$ ▷ Initialize individual fitness to 0
$n_i \leftarrow 0$ ▷ Initialize counter to 0
while $t \leq T_{max}$ **do** ▷ Simulation loop
  for Robot i **do**
    if New Decision for Robot i **then**
      $n_i \leftarrow n_i + 1$
      if Decision is Correct **then**
        $f_i \leftarrow f_i + t$ ▷ Add decision time
      else
        $f_i \leftarrow f_i + T_{max}$ ▷ Add max penalty
      end if
    else if $t = T_{max}$ & No Decision **then**
      $n_i \leftarrow n_i + 1$
      $f_i \leftarrow T_{max}$
    end if
  end for
end while
return $fit = \frac{1}{N} \sum_{i=0}^{N} f_i/n_i$ ▷ N=number of robots

The PSO swarm is initialized randomly within the bounded search space, with the exception of one particle which is set to an empirically chosen location. We perform the optimizations with 15 particles, each evaluated 10 times for noise resistance, and proceed for 75 iterations. For the empirical particle, the observation interval τ is set to the number of time steps needed to cross one square, $t_s = 282$. The random forward variable s is set to the number of time steps needed to cross two squares, $s = 564$. We use an estimate of one robot body length to find the empirical collision avoidance distance $d = 50$. Lastly, the hysteresis parameter h is initialized at zero. This allows the optimization process to heuristically assess the utility of a non-zero hysteresis parameter. The final fitness assigned to a particle p_i is then a weighted sum of the average (μ) and the standard deviation (σ) across the noise evaluations as described in Equation 6. $fit_n(p_i)$ denotes a fitness evaluation obtained through a single instance of particle p_i following Algorithm 2. The parameter γ is set to 1.1 empirically, giving a slightly higher significance to consistency compared with average performance. The optimization process favors lower fitness values as iterations progress, rewarding particles that return a low fitness value across multiple noise evaluations.

$$fitness(p_i) = \mu(fit_n(p_i)) + \gamma \cdot \sigma(fit_n(p_i)) \tag{6}$$

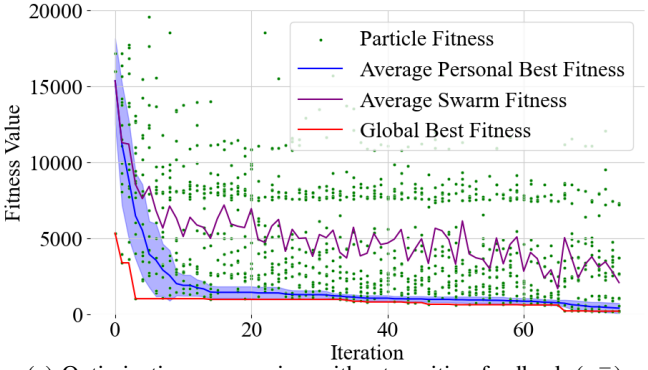
VI. Results

The optimization results are shown in Figure 5. Through iterations, the average personal best fitness of the particles can be seen to converge downward towards the global

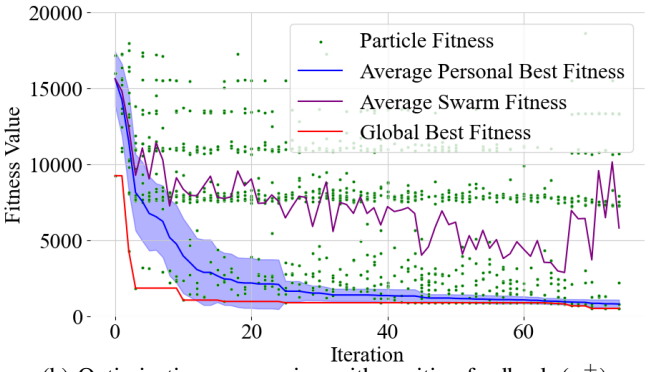
best fitness along with the standard deviation. Since the simulation evaluation is noisy and particle exploration is stochastic, the individual particle fitness values, represented as the green dots, have large variations as iterations progress. This is also seen in the average swarm fitness. Table I shows the empirical particle along with the parameter bounds, and Table II shows the optimal parameters. We discuss different optimal parameters between the two settings of positive feedback. In both optimal particles, the observation interval τ is brought towards the lower bound, where our sampling pause prevents an absolute minimum. However the hysteresis parameter, h, was brought up despite the empirical particle having no hysteresis. This combination allows the robots to make faster observations (small τ), yet remain robustly elastic in their initial and subsequent decisions (non-zero h). The optimal value of the random forward variable s differs largely depending on the use of positive feedback. When positive feedback is off, s is set lower; each robot works to greedily gather data and shape individual decisions. On the contrary, robots spend more time exploring the arena with positive feedback, indicated by a larger value for s. In this case, individual robot decisions become heavily influential. A similar rationale applies to the smaller collision avoidance parameter d obtained when no positive feedback is used; collision avoidance appears to become less critical and robots greedily prioritize sampling.

We evaluate the performance of the optimized particle against the empirical particle through 100 randomized experiments; varying the generated ($f = 0.52$) pattern projected on the arena and initial robot poses. Figure 6 shows the distribution of fitness values obtained across different fill ratios. Each dot represents a single particle evaluation calculated using Algorithm 2. The optimized parameters achieve a significantly lower median evaluation and tighter performance compared to the empirical parameters.

The obtained fitness distributions for a fill ratio of $f = 0.52$ across the 100 randomized experiments are shown in Figure 7. The distribution of fitness values shift towards the favorable side with and without positive feedback. Furthermore, the optimized cases show faster stabilization towards the correct beliefs (Figure 8). However, the experiments with positive feedback stabilize towards incorrect classifications more often compared to those without positive feedback. This occurs when robots make hasty decisions that quickly spread across the group due to the positive feedback mechanism. Interestingly, the optimized parameters require the swarm to sample new tiles at a lower rate. Despite this, it can be seen in Figure 7 that the robots are making faster correct classifications. We see that the non-zero hysteresis parameter ensures stability in the classification decision despite the apparent under-sampling, eventually benefiting the decision fusion mechanism through positive feedback. In contrast, over-sampling without hysteresis ($h = 0$) quickly leads the robots to misleading classification decisions. This is particularly detrimental with positive feedback, where broadcast incorrect decisions become adversarial for the group.



(a) Optimization progression without positive feedback (u^-)



(b) Optimization progression with positive feedback (u^+)

Fig. 5: Optimization progression using a fitness evaluation that combines decision accuracy and speed (Equation 6), 15 particles, 10 noise evaluations, and for 75 iterations. We arrive at a best found fitness of 508.532 without positive feedback and 192.026 with positive feedback. Fitness values of the 15 particles (green) and the average swarm fitness (purple) are shown. The average of personal bests (blue) along with one standard deviation (shaded blue) can be seen to converge to the global best fitness value (red). Individual particle fitness and average swarm values exhibit stochastic behavior due to randomness inherent in the robot behavior.

TABLE I: α_0: white observation prior parameter. β_0: black observation prior parameter. τ: observation interval. s: random forward parameter. d: collision avoidance distance. h: hysteresis parameter. n: boundary multiplier, in our approach $n = 5$. We derive $t_s = 282$ and $t_a = 4515$ using the robot speed (2.77 cm/s) and tile size (6.25cm) (see Section V).

$\alpha_0 = \beta_0$	τ	s	d	h
$[0, 0]$	$[t_s/n,\ t_s \cdot n]$	$[t_s/n,\ t_a \cdot n]$	$[5, 145]$	$[0,\ 128]$
0	282	564	50	0

TABLE II: Empirical parameters are shown as p. Optimal parameters are shown as $p^*_{u^-}$ and $p^*_{u^+}$, for experiments without and with positive feedback, respectively.

Parameter Set	α	τ	s	d	h
p	0	282	564	50	0
$p^*_{u^-}$	0	56	178	29	17
$p^*_{u^+}$	0	57	912	51	10

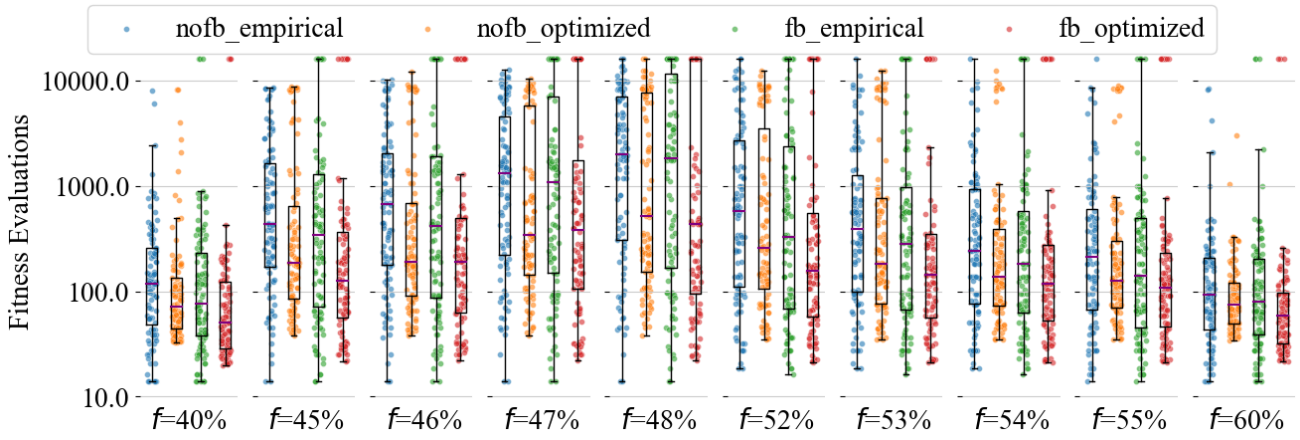
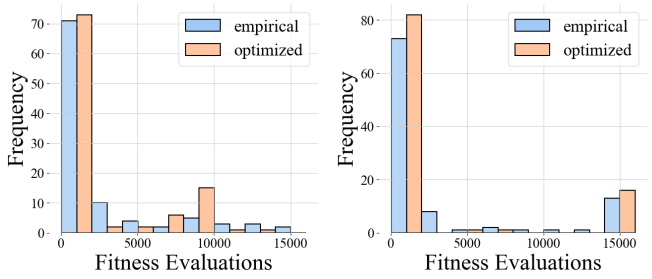
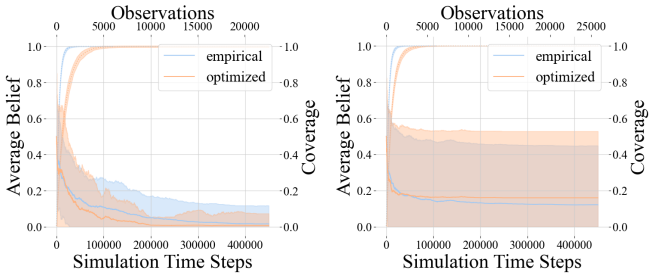


Fig. 6: Distribution of fitness values obtained across 100 randomized simulations for each given fill ratio, using optimized parameters from Table II. The optimized parameter set outperforms the empirical one in all tested fill ratios, although being specifically optimized for the case of $f = 0.52$. We observe a 55.3% reduction in the median evaluations through the optimization without positive feedback and a 51% reduction in the median evaluations with positive feedback for $f = 0.52$.



(a) No positive feedback (u^-) (b) With positive feedback (u^+)

Fig. 7: Distribution of fitness evaluations over 100 randomized simulations using a fill ratio of $f = 0.52$. We observe that through optimization the distribution of fitness values shifts favorably towards lower values, in both cases with and without positive feedback.



(a) No positive feedback (u^-) (b) With positive feedback (u^+)

Fig. 8: Average beliefs and coverage of the robots over 100 randomized simulation experiments. The simulations were randomized by rearranging the projected pattern (with the same fill ratio) and initial robot poses. The shaded regions denote one standard deviation. The pattern fill ratio was fixed to $f = 0.52$ in all cases. The optimized parameters lead the robots to converge faster towards the correct belief.

VII. Conclusion and Future Work

In this paper, we presented a simulation and optimization framework for studying a two-outcome surface inspection task using a swarm of miniaturized wheeled robots that employ a decentralized Bayesian algorithm. We built upon a previously studied inspection algorithm by introducing a new hysteresis parameter that creates elasticity around robot decisions. We used the Webots robotic simulator to perform accurate physics-based simulations of a multi robot system. Using a noise resistant variant of the Particle Swarm Optimization (PSO) method, we obtained a set of optimal parameters. Results from 100 randomized experiments revealed that the robot swarm employing optimized algorithm parameters were able to achieve a 55% improvement in median fitness evaluations without positive feedback and a 51% improvement with positive feedback compared to the empirically chosen parameters. Furthermore, the results revealed that a non-zero hysteresis parameter leads to improvements in the final decision accuracy. In future work, we plan to study two-outcome surface inspection tasks of higher complexity. In particular, we will (i) conduct simulations for inspection of geometrically complex 3D surface sections and consider dynamic environments, (ii) study commonly employed sensing modalities and signal processing methods such as camera feed or vibration sensing, and (iii) conduct real life experiments to validate the performance of the optimized parameter set obtained from simulation in reality.

Acknowledgment

We wish to thank Dr. Olivier Michel and Yannick Goumaz from Cyberbotics Ltd. for supporting the AWS simulations and Dr. Ariel Ekblaw from MIT Space Exploration Initiative for making available the robot proto file for our simulations.

References

[1] M. Bualat, L. Edwards, T. Fong, M. Broxton, L. Flueckiger, S. Y. Lee, E. Park, V. To, H. Utz, V. Verma *et al.*, "Autonomous robotic inspection

for lunar surface operations," in *Field and Service Robotics*. Springer, 2008, pp. 169–178.

[2] C. Carbone, O. Garibaldi, and Z. Kurt, "Swarm Robotics as a Solution to Crops Inspection for Precision Agriculture," *KnE Engineering*, vol. 3, no. 1, p. 552, Feb. 2018.

[3] D. Carrillo-Zapata, E. Milner, J. Hird, G. Tzoumas, P. J. Vardanega, M. Sooriyabandara, M. Giuliani, A. F. T. Winfield, and S. Hauert, "Mutual Shaping in Swarm Robotics: User Studies in Fire and Rescue, Storage Organization, and Bridge Inspection," *Frontiers in Robotics and AI*, vol. 7, p. 53, Apr. 2020.

[4] Y. Liu, M. Hajj, and Y. Bao, "Review of robot-based damage assessment for offshore wind turbines," *Renewable and Sustainable Energy Reviews*, vol. 158, p. 112187, 2022.

[5] B. Bayat, N. Crasta, A. Crespi, A. M. Pascoal, and A. Ijspeert, "Environmental monitoring using autonomous vehicles: a survey of recent searching techniques," *Current Opinion in Biotechnology*, vol. 45, pp. 76–84, Jun. 2017.

[6] M. Alanyali, S. Venkatesh, O. Savas, and S. Aeron, "Distributed bayesian hypothesis testing in sensor networks," in *Proceedings of the 2004 American control conference*, vol. 6. IEEE, 2004, pp. 5369–5374.

[7] S. Bandyopadhyay and S.-J. Chung, "Distributed estimation using bayesian consensus filtering," in *2014 American control conference*. IEEE, 2014, pp. 634–641.

[8] T. Zhao and A. Nehorai, "Distributed sequential bayesian estimation of a diffusive source in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1511–1524, 2007.

[9] J. T. Ebert, M. Gauci, F. Mallmann-Trenn, and R. Nagpal, "Bayes bots: collective bayesian decision-making in decentralized robot swarms," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7186–7192.

[10] J. T. Ebert, M. Gauci, and R. Nagpal, "Multi-feature collective decision making in robot swarms," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 1711–1719.

[11] G. Valentini, D. Brambilla, H. Hamann, and M. Dorigo, "Collective perception of environmental features in a robot swarm," in *International Conference on Swarm Intelligence*. Springer, 2016, pp. 65–76.

[12] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 kilobots: Speed versus accuracy in binary discrimination problems," *Autonomous agents and multi-agent systems*, vol. 30, no. 3, pp. 553–580, 2016.

[13] G. Valentini, H. Hamann, and M. Dorigo, "Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 2015, pp. 1305–1314.

[14] B. Haghighat, J. Boghaert, Z. Minsky-Primus, J. Ebert, F. Liu, M. Nisser, A. Ekblaw, and R. Nagpal, "An approach based on particle swarm optimization for inspection of spacecraft hulls by a swarm of miniaturized robots," in *International Conference on Swarm Intelligence*. Springer, 2022, pp. 14–27.

[15] O. Michel, "WebotsTM: Professional Mobile Robot Simulation," *arXiv:cs/0412052*, Dec. 2004.

[16] A. Dementyev, H.-L. C. Kao, I. Choi, D. Ajilo, M. Xu, J. A. Paradiso, C. Schmandt, and S. Follmer, "Rovables: Miniature On-Body Robots as Mobile Wearables," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. Tokyo Japan: ACM, Oct. 2016, pp. 111–120. [Online]. Available: https://dl.acm.org/doi/10.1145/2984511.2984531

[17] J. Pugh and A. Martinoli, "Distributed scalable multi-robot learning using particle swarm optimization," *Swarm Intelligence*, vol. 3, no. 3, pp. 203–222, May 2009.

[18] J. Pugh, Y. Zhang, and A. Martinoli, "Particle swarm optimization for unsupervised robotic learning," in *IEEE Swarm Intelligence Symposium*, 2005, pp. 92–99.

[19] E. Di Mario, I. Navarro, and A. Martinoli, "Analysis of fitness noise in particle swarm optimization: From robotic learning to benchmark functions," in *IEEE Congress on Evolutionary Computation*, 2014, pp. 2785–2792.

[20] ——, "Distributed particle swarm optimization using optimal computing budget allocation for multi-robot learning," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 566–572.