Collective Decision-Making with Bayesian Robots in Dynamic Environments

*Kai Pfister and Heiko Hamann**

Abstract

Collective decision-making enables self-organizing robot swarms to act autonomously on a swarm level and is essential to coordinate their actions as a whole. When robots only share and communicate information locally a distributed and decentralized approach is required. In a previous paper [4], an efficient method based on a distributed Bayesian algorithm was created to distinguish a binary environment. We extended it to have the capability of dealing with dynamic environments, which requires to avoid global lock-in states. In many realistic applications the robot swarm needs to adapt to (collectively) measurable changes at runtime by revising previous collective decisions. The trade-off between decision-making speed and readiness to revise previous decisions is a seemingly unavoidable challenge. We present our extension of the former approach and study how this trade-off can efficiently be balanced.

1 Introduction

In swarm robotics [1] we constrain robots to only communicate with their neighbors and limit the robot controllers to behavior-based, reactive control. In exchange we receive a scalable and often robust distributed multi-robot system. Swarm-wide coordination is, however, difficult because we lack a global communication channel. Information and especially local decisions need to be propagated through the swarm by neighbor-to-neighbor communication. Hence, collective decision-making is an essential swarm capability to coordinate the swarm and to achieve swarm-level autonomy. Collective decision-making is frequently studied in swarm robotics. For example, in best-of-n problems in which the qualitatively best option needs to be collectively picked from a set of n options [2]. A frequently studied benchmark scenario is that of collective perception [3, 4, 5]. In these studies robots move on black and white tiles and need to collectively determine whether the black or white tiles are the majority. One of the most important aspects in collective decision-making is the speed-vs-accuracy trade-off [6, 3]. A natural or artificial swarm can either be fast and

*[1]both authors are at the Institute of Computer Engineering, University of Luebeck, Germany hamann@iti.uni-luebeck.de

rather inaccurate or rather slow and accurate in its decision making. Therefore, balancing these two properties for a given use case is important in swarm engineering.

1.1 Dynamic environments

Once we consider a dynamic environment (e.g., majorities of tiles change over time in the collective perception benchmark) a specific variant of the speed-vs-accuracy trade-off seems to emerge and is a focus of this paper. The more accurate a swarm is in its decisions the slower it is in adapting to a dynamic environment. A robot swarm requires two capabilities to adapt to dynamic environments in collective decision-making: detecting the change and subsequently revising its previous decision. Change or anomaly detection is known from statistics but requires a distributed approach, as proposed in an earlier paper [7]. It is challenging to minimize false positives (detecting a change when there is none) while maximizing speed (time between environment change and collectively agreeing to have detected a change). Whether a (robot) swarm is able to reconsider and revise its previously made decision is an important but not often considered question [8, 9]. IT is challenging to carefully reduce the influence of positive feedback so that lock-in states are avoided and the swarm can remain being adaptive [10, 11]. A lock-in state is an absorbing swarm system state that cannot be left later independent of incoming evidence (measurements). Avoiding lock-ins usually requires a careful balance between the importance of new evidence, the memory of previously collected evidence and the positive feedback, for example, via messages from neighboring robots.

1.2 Related work and Bayes Bots

Collective decision-making in swarm robotics has been studied for many years. A recent review of the field is given by Valentini et al. [2]. Relevant studies about dynamic environments are the above mentioned [10, 7]. Both propose methods based on gossiping and on sliding time windows that invalidate too old sensor data. The work by Divband Soorati et al. [10] is a rare exception in literature. It proposes an algorithm for collective decision-making that avoids lock-in states and is able to deal with dynamic environments. The algorithm is simple as it counts evidence without a sophisticated model of beliefs. If a threshold is reached then the count is halved to avoid lock-ins. Another mechanism against lock-ins is a timeout. If a robot has not received evidence supporting its current opinion for 60 seconds, it resets to an undecided state with zero count of evidence.

The paper of Ebert et al. [4] on 'Bayes Bots' serves as a starting point for our study. They implement a so-called distributed Bayesian approach where each robot individually models a belief about which tile color is currently in the majority. The model is a beta distribution, in particular a probability density function $Beta$ on the interval $[0, 1]$ determined by two parameters (α, β). Each robot collects evidence (measurements of tile colors) and updates its

beta distribution accordingly. Communication with neighboring robots is only used in specific situations to implement a swarm-wide self-organization effect using positive feedback.

The Bayes-Bot approach is in contrast to more bio-inspired approaches, such as by Valentini et al. [3]. They were inspired by nest-site selection in honeybees [12] and formulate a simpler approach without an explicit internal model in each robot. Robots only count the collected evidence (e.g., opinions of neighbors) and choose either the majority or a random piece of evidence but are supported by an in-built positive feedback effect. Many other examples and studies of such 'model-free' collective decision-making have been published [13, 14, 10]. Another example for a more sophisticated approach is that of Bartashevich and Mostaghim [5]. Both, the simplistic methods and the more sophisticated methods, have reasonable use cases. For example, based on a robot's limited capabilities the Bayes-Bot approach may not be feasible, hence a simplistic (bio-inspired) approach could be required.

1.3 Approach

We propose an extension of the Bayes-Bots approach by Ebert et al. [4] to deal with dynamic environments. This requires to avoid lock-in states, to continuously monitor the environment and to carefully balance knowledge about old and new evidence as well as positive feedback by communicating with neighbors. In comparison to Divband Soorati et al. [10], we propose a more sophisticated approach that statistically models the robot's belief and implements a more robust belief reset strategy. In our approach robots collect evidence and keep an updated model of their beliefs using the beta distribution as by Ebert et al. [4]. A robot adopts one of the three states: uncommitted, in favor of a black tile majority or in favor of a white tile majority. Robots reset their state (i.e., delete history and switch to uncommitted) if they meet a robot of different opinion or if they have recently found a lot of evidence of the opposite opinion. Also, robots stop counting evidence in favor of their current opinion if they have collected a sufficient amount of samples (for details see below).

2 Scenario and Simulation

As a benchmark scenario we choose the collective perception scenario based on swarm robot simulations.

2.1 Scenario

Our scenario of choice is the so-called collective perception scenario [3, 15]. A swarm of robots has to decide which of two distributed environmental features is dominant. Previous work on collective decision-making primarily used static environments for benchmarking [2, 5, 15] including the study on Bayesian decision-making [4]. A rare exception is, for example, the study by Divband

Soorati et al. [10]. In a static environment task difficulty in the collective perception scenario is parameterized by the fill ratio of black and white tiles. The closer the proportion is to $f = 0.5$, the harder it is to make a fast and accurate decision. In our experiments instead we simulate a dynamic environment by inverting the fill ratio at halftime during the experiment. We test a variety of fill ratios f in that dynamic setup.

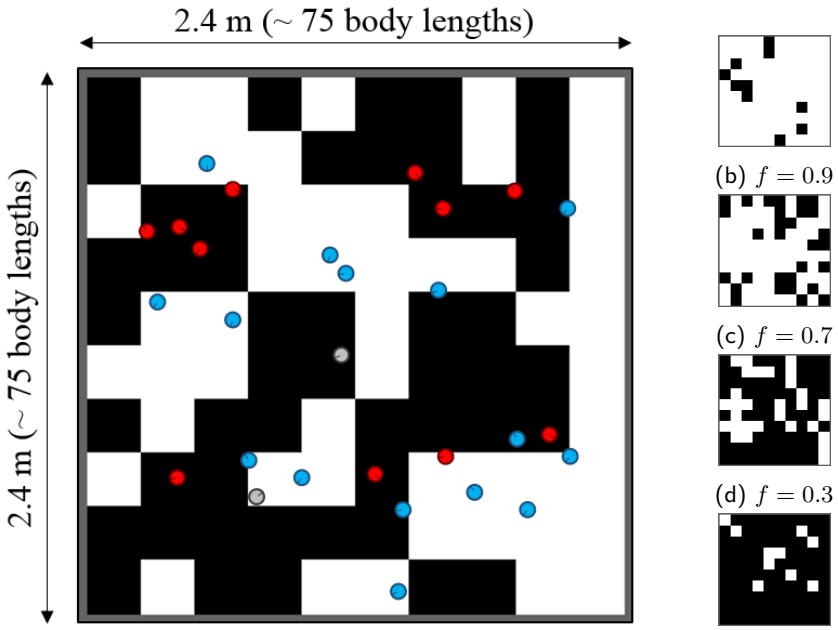
2.2 Simulations

Our study is fully based on simulations. Following Ebert et al. [4], we still focus on a concrete robot platform: the Kilobot. A Kilobot is a low-cost swarm robot to test and verify collective algorithms. Kilobots have a light-intensity sensor and communicate with each other via an infrared transmitter and a receiver for up to 7 cm. The robot uses two vibrating electric motors for differential drive locomotion [16].

As a simulation we used Kilosim [17], a fast and lightweight C++ simulator specifically designed for Kilobot robots. Up to 512 Kilobots can be simulated in a 2D environment. Kilosim is a pseudo-physical simulation in which collisions between robots are detected and treated but without full support of a physics engine. We use light patterns to simulate the black and white tiles in the environment (in a physical implementation these light patterns would be projected on the arena to be detected by the Kilobot's light sensor). The light pattern consists of 100 black and white tiles randomly arranged in a grid and limited by a gray border (see Fig. 1). Height and width of the environment are around 75 body lengths of a Kilobot ($\sim$2.4 m). Initially we place $N = 100$ robots uniformly with random orientation in the arena. The robots move with a pseudo-random walk [4]. A robot moves straight forward at a speed of 24 mm/s. We refer to simulated time in seconds. The duration of each forward movement phase is drawn from an exponential distribution ($\lambda = 4.17 \times 10^{-3}$, average durations of $1/\lambda = 240$ s). Afterwards it does a random turn at a speed of 0.5 rad/s which is uniformly drawn from $[0, 2\pi]$. When a robot moves onto the border area marked in gray it turns to the right until it detects having left the border area. We automatically created randomized light patterns with a desired fill ratio f of black and white tiles. For our simulation runs we created eight environments for $f \in \{0.1, 0.3, 0.4, 0.45, 0.55, 0.6, 0.7, 0.9\}$ which we reused in all trials (a few are shown in Fig. 1).

3 Collective Decision-Making in Dynamic Environments and Bayesian Approach

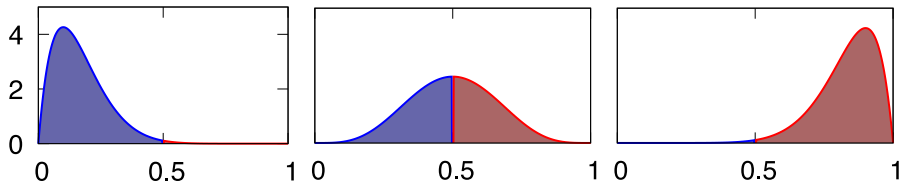
Many different techniques have been presented for the mechanism of how individual robots switch opinions in swarm robot collective decision-making, such as the majority rule [3], the voter model [18] and more sophisticated approaches [5]. Here we focus exclusively on the Bayes-Bot approach by Ebert et al. [4]. Key idea is that each robot has a simple statistical model (beta distribution) of its



(a) Simulated robot arena for environment $f = 0.5$

Fig. 1: Simulated robot arena and tiling patterns (similar presentation to [4]). (a) Top view of robot arena with 10-by-10 black and white tiling pattern. Robots are given as circles with color-code: in favor of white (blue), in favor of black (red), and uncommitted (gray). (b) Randomly generated tilings for task difficulties $f \in \{0.9, 0.7, 0.3, 0.1\}$ (ratio of white vs black tiles).

perception of black and white tiles. Each robot updates the parametrization of this model online according to collected evidence (perception of a tile) and can share the parameters with other robots. We extended this approach to intelligently react to dynamic environments. A major drawback of most published methods, including the Bayes-Bot approach, is that the swarm is absorbed in a lock-in state once it has converged. A dynamic environment, that may even invert the original situation, can possibly not be detected or at least the swarm cannot react to it by reconsidering its previous consensus. To avoid lock-in states we confine the beta distribution by a threshold (p_{lim}) that, once reached, triggers robots to stop collecting more evidence and to make a decision. This is necessary to prevent a robot from collecting too much counter evidence that may later prevent it from quickly detecting a change in the environment. In addition, robots reset their internal model once they meet robots with different opinions and more experience. A drawback is that false positives of assumed environment changes increase. Hence, we face a trade-off between overly con-



(a) $\alpha = 2, \beta = 10$ (b) $\alpha = 5, \beta = 5$ (c) $\alpha = 10, \beta = 2$

Fig. 2: Beta distributions for example parameter settings: (a) majority of black tiles, (b) fifty/fifty, and (c) majority of white tiles.

servative, slow swarms and overly progressive, fast swarms.

3.1 Distributed Bayesian Approach

Each robot collects evidence for the dominant tile color by binary color measurements $C \in \{0, 1\}$ with zero for black and one for white. It also models the unknown fill ratio f with a beta distribution $\text{Beta}(\alpha, \beta)$ and updates its posterior parameters α and β with every new observation C

$$\text{Beta}(\alpha + C, \beta + (1 - C)) \,, \tag{1}$$

as described by Ebert et al. [4]. The beta distribution models one random variable and is well suited for a binary task such as robots observing a ratio of two types of tiles.

In Fig. 2 we give three examples of how parameters α and β influence the distribution. Here, β models the number of black tiles while α models white tiles. Whether the environment is mostly black or white is decided once a robot has observed enough samples of one color and reaches a credible threshold p_c.

3.2 Being Prepared for Dynamic Environments

In the following section we introduce our extension and explain our decision-making algorithm (see Alg. 1).[1] To enable the robots to revise their previous decisions if required, we extended the Bayesian Decision-Making algorithm by Ebert et al. [4] with a reset function $\text{resetD}_f(q, d_f)$ (lines 40-43, Alg. 1). In the case a robot decided by reaching the credible threshold p_c (l. 31-35) it continues collecting samples. Once its estimated fill ratio q falls below the reset threshold p_{res} the robot increases its reset counter r by one. The robot resets itself to the init. state and starts observing the environment anew. The reset counter r ensures that robots only influence neighbors if they have a greater or equal reset counter. This prevents robots from disturbing the swarm with outdated decisions.

[1] our implementation is available online: https://gitlab.iti.uni-luebeck.de/pfister/dynamic-bayes-bots/

Algorithm 1 Dynamic Bayesian Decision-Making Alg.

Input: Observational interval τ, credible threshold p_c, reset threshold p_{res}, sample limit p_{lim}, priorparameter α_0
Output: Mean decision d_f of all robots at time t

Init counter of white observations $\alpha \leftarrow \alpha_0$
Init counter of black observations $\beta \leftarrow \alpha_0$
Init incomplete decision $d_f \leftarrow -1$
Init reset counter $r \leftarrow 0$
Init reset flag $d_{\text{res}} \leftarrow 0$
for $t \in [1, T]$ **do**
    if $d_{\text{res}} \neq 0$ **then**
        Reset: $d_{\text{res}}, d_f, \alpha, \beta$ and s
    Perform pseudo-random walk
    if τ divides t **then**
        $C \leftarrow$ observed color (0,1)
        if $q < p_{\text{lim}}$ **then**
            $\alpha \leftarrow \alpha + C$
        if $q > (1 - p_{\text{lim}})$ **then**
            $\beta \leftarrow \beta + (1 - C)$
    Let $m = (id', i', d'_f, r', \alpha', \beta')$ ▷ Msg. of neighbor
    if $d_f \neq d'_f$ and $d'_f \neq -1$ and $r \leq r'$ and $q > p_{\text{res}}$ **then**
        $d_{\text{res}} \leftarrow 1$
    if $d_f = -1$ and $d'_f \neq -1$ and $r \leq r'$ and $q < p_{\text{res}}$ **then**
        $d_f \leftarrow d'_f$
        $\alpha \leftarrow \alpha'$
        $\beta \leftarrow \beta'$
    else if $d_f == d'_f$ **then**
        if $q < p_{\text{lim}}$ **then**
            $\alpha \leftarrow \alpha + C'$
        if $q > (1 - p_{\text{lim}})$ **then**
            $\beta \leftarrow \beta + (1 - C')$
    $q \leftarrow \int_0^{0.5} \text{Beta}(\alpha, \beta)$
    if $q > p_{\text{lim}}$ **then**
        $q \leftarrow p_{\text{lim}}$
    else if $q < (1 - p_{\text{lim}})$ **then**
        $q \leftarrow (1 - p_{\text{lim}})$
    if $d_f = -1$ **then**
        if $q > p_c$ **then**
            $d_f \leftarrow 0$
        else if $(1 - q) > p_c$ **then**
            $d_f \leftarrow 1$
        Broadcast message $(id, i, d_f, r, \alpha, \beta)$
    else
        Broadcast message $(id, i, d_f, r, \alpha, \beta)$
        RESET$D_f(q,\ d_f)$

42: **function** RESET$D_f(q,\ d_f)$
43: **if** $(d_f = 1$ and $(1-q) < p_{\text{res}})$
or $(d_f = 0$ and $q < p_{\text{res}})$ **then**
44: $d_{\text{res}} \leftarrow 1$
45: $r \leftarrow r + 1$

Tab. 1: Used parameter set

reset threshold p_{res}	0.55
credible threshold p_c	0.95
sample limit p_{lim}	0.96, 0.98, 0.99, none
trial duration	6000 s
environment change	3000 s
environments	0.9→0.1, 0.7→0.3, 0.6→0.4, 0.55→0.45
observational interval τ	20 s

Running tests only on this extension lead to observations of rather conservative swarms. A lot of time passed before the swarm collectively detected the dynamic environment (temporary false negative). We especially observed this in low difficulty environments (fill ratios $f < 0.3$ or $f > 0.7$). To implement robots that are more decisive we introduced the limitation threshold p_{lim} for observations that confirm a robot's current decision ($1 - p_{\text{lim}} \leq \int_0^{0.5} \text{Beta}(\alpha, \beta) \leq p_{\text{lim}}$). A new color observation is made every τ seconds and is used to update the beta distribution with respect to the limitation threshold p_{lim} (l. 10-15). After a robot has reached p_{lim} it does not collect more confirmative samples until a counter sample has been collected (l. 12-14 and l. 26-30). This extension provokes faster decision switches because the more samples are collected the more counter samples are needed after an environmental change to trigger a reset.

Each robot broadcasts its state and evidence (lines 36-38, Alg. 1) and respectively receives its neighbors' states and evidence (l. 16). It also processes all received messages (first come first serve) and possibly overrides its own model parameters with those of neighbors or adds their evidence (l. 19-25). See table 1 for used parameters.

We extended the Bayes-Bot approach [4] to a dynamic variant with three different kinds of feedback. This aims to increase the swarm's capability to detect dynamic environments (i.e., decreasing false negatives). It helps to quickly revise the previous decision while still trying to minimize false positives (i.e., maximizing specificity).

Reset feedback (lines 17-18, Alg. 1) A robot resets itself if its neighbor has made a different decision d'_f than itself, has an equal or higher reset counter r and the robot's estimated fill ratio q is higher than p_{res}. The current belief q must be higher than p_{res} to prevent robots from resetting themselves repeatedly.

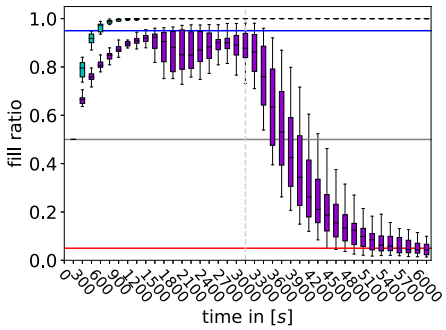


Fig. 3: Comparison static (green) vs dynamic (purple) algorithm in dynamic environment $f = 0.7 \rightarrow f = 0.3$; each boxplot gives the fill ratios of the swarm averaged over all robots (swarm size $N = 100$) for 20 runs, outliers are not shown; blue line shows upper $p_c = 0.95$ threshold, red line shows lower $p'_c = 0.05$ credible threshold; initially the tile pattern has a ration of $f = 0.7$ (majority of white tiles) that switches suddenly at time $t = 3000$ s to $f = 0.3$ (majority of black tiles), sample limit $p_{\text{lim}} = 0.99$.

Positive feedback by experienced neighbors (lines 19-22) If a robot has not made a decision d_f yet and its estimated fill ratio q is lower than the reset threshold p_{res} (robots close to a decision should not be influenced) it then overwrites its model parameters α, β and q which are communicated by an experienced neighbor that must have an equal or higher reset counter r.

Feedback by neighbor evidence (lines 23-25) If a robot and its neighbor share the same decision d_f it uses the neighbor's color observation to update its posterior parameters α and β respectively. Therefore robots reinforce their current decision by collectively gathering evidence.

4 Results

First, we compare the 'static' approach of Ebert et al. [4], that was not designed to deal with dynamic environments, to our featured 'dynamic' extension. In Fig. 3 we show the fill ratio as perceived and modeled via beta distributions individually in all robots for the static (green) and dynamic (purple) variant and its evolution over time. At time $t = 3000$ s the tiling pattern is switched instantly from the previous majority of white tiles ($f = 0.7$) to a majority of black tiles ($f = 0.3$). As expected, the static algorithm shows no reaction because the robot swarm is fully absorbed in a lock-in state (all robots finished updating their belief state). Whereas our dynamic extension of the algorithm reacts within the first 500 s. It then converges to the correct new collective response within the provided 3000 s of time. The increased variance for our dynamic approach after $t = 1600$ s as seen in Fig. 3 is due to the reset mechanism.

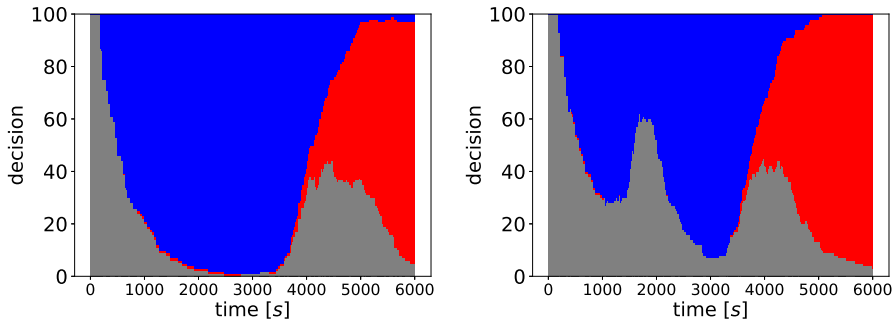


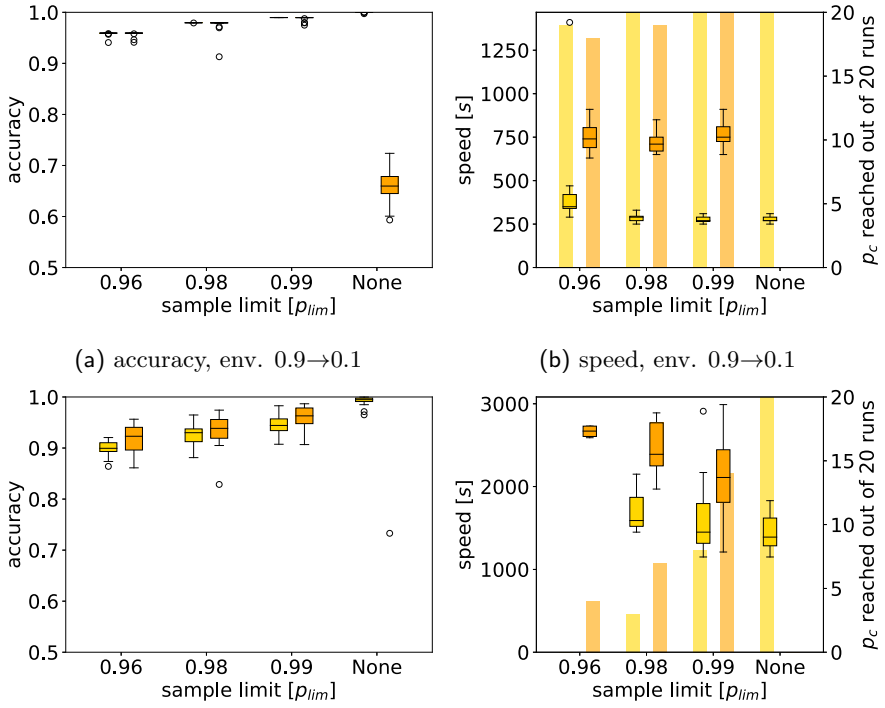
Fig. 4: Number of robots committed to blue (opinion for a majority of white tiles) or red (opinion for majority black), and gray for uncommitted; two representative runs in a $0.7 \rightarrow 0.3$ dynamic environment (switch after 3000 s), $N = 100$.

Most robots are committed (indicated by mean close to credible threshold $p_c = 0.95$) and reset (i.e., defaulting to fill ration 0.5 and uncommitted) once their observed fill ratio falls below the reset threshold $p_{\text{res}} = 0.55$ due to variances in measurements. These resets temporarily reduce the swarm decision accuracy but this drawback is necessary to implement enough adaptability to react to changes. This comparison also shows the trade-off between fast converging on the right answer while staying adaptive to dynamic environments. Our approach converges slower in the first half of the experiment than the static algorithm. This is most likely an unavoidable side-effect of reducing false negatives (i.e., not reacting to changes in the environment).

In Fig. 4 we provide the number of robots committed to blue (in favor of a white tile majority) or red (in favor of a black tile majority), and uncommitted robots over time for two representative simulation runs. In the first half of the run shown in Fig. 4a, the robots quickly form a large blue majority. Nevertheless, after the environment switches at $t = 3000$ s a sub-population of about 40 uncommitted robots remains until about $t = 5200$ s. In the first half of the run shown in Fig. 4b, the swarm experiences a wave of resetting robots due to variance (cf. discussion above) forming a temporary majority of uncommitted robots at about $t = 1800$ s again. However, after the environment switch the swarm relatively quickly forms a red majority.

Next, we examine the costs of avoiding lock-in states. One of the most crucial parameters controlling these costs in our algorithm is the sample limit p_{lim}. It limits the reachable maximum of the beta distribution even if more confirming evidence could be collected. The sample limit p_{lim} directly and partially controls the achievable accuracy, the speed of convergence, and the system's ability to avoid lock-in states.

In Fig. 5, we investigate the sample limit p_{lim} of our dynamic algorithm



(a) accuracy, env. 0.9→0.1

(b) speed, env. 0.9→0.1

(c) accuracy, env. 0.7→0.3

(d) speed, env. 0.7→0.3

Fig. 5: Accuracy and speed (boxplots) over different sample limits p_{lim} for 20 runs and two environments: $f = 0.9 \rightarrow f = 0.1$ (very simple) and $f = 0.7 \rightarrow f = 0.3$ (simple); bar charts in (b) and (d) give number of runs that collectively reached the credible threshold p_c before and after the switch.

for the very simple (0.9→0.1) and the simple environment (0.7→0.3). As accuracy we define the maximum of the robots' belief states over all time steps before/after the environment switch. The states are then averaged over the swarm (mean of all robots' fill ratios) for all 20 runs. In Fig. 5a we see the achieved accuracy for a given sample limit p_{lim} before the environment switch ($f = 0.9$, left boxplots) and after the switch ($f = 0.1$, right boxplots). With increased sample limit p_{lim}, we reach better accuracies for both, before and after the switch. All median accuracies are close to the respective sample limit p_{lim} (i.e., above $p_{\text{lim}} - 10^{-3}$) as expected. At the far right ('None'), we give data for turning the sample limit off (Alg. 1, line 25). We observe a high accuracy before the environment switch (median 0.999) but a low accuracy after (median 0.66).

In Fig. 5b we observe the achieved convergence times before the environment switch (left boxplot, yellow) and after the switch (right boxplot, orange). The bar charts give the count of how many runs reached the required threshold $p_c = 0.95$. Before the switch swarms converge quickly and there is little difference between the parameter settings. The convergence after the switch takes up to three times more time. Again, there is not much difference between the parameter settings. However, the system does not converge after the switch when setting sample limit off. Hence, the system is close to a lock-in state before the switch.

In Fig. 5c we see the achieved accuracy for the harder but still simple environment (0.7→0.3). All accuracies are lower compared to Fig. 5a due to the increased task difficulty (0.90 for $p_{\text{lim}} = 0.96$, 0.93 for $p_{\text{lim}} = 0.98$, 0.94 for $p_{\text{lim}} = 0.99$, 0.99 for none). We notice that the accuracy after the switch tends to be better compared to the accuracy before the switch (0.92 for $p_{\text{lim}} = 0.96$, 0.94 for $p_{\text{lim}} = 0.98$, 0.96 for $p_{\text{lim}} = 0.99$) except for the deactivated sample limit that shows no switching behavior (median accuracy of 0.10).

In Fig. 5d we examine the achieved convergence times (notice different scale for convergence times). As the bar charts indicate, the system converges in less than half the runs for sample limit $p_{\text{lim}} < 0.99$. We find that setting $p_{\text{lim}} = 0.99$ is a good compromise between still observing convergence before and after the switch (8 runs before and 14 runs after out of 20) while achieving good accuracy before and after the switch. When turning the sample limit off, all runs converge before the environmental switch but none converges after the switch as expected. This system behavior is the motivation why we introduced the sample limit p_{lim}.

In Fig. 6 we compare the two accuracy results from above (0.9→0.1 and 0.7→0.3) for sample limit $p_{\text{lim}} = 0.99$ to a hard environment ($f = 0.6 \rightarrow f = 0.4$) and a very hard environment ($f = 0.55 \rightarrow f = 0.45$). For more and more difficult environments the accuracy drops and the difference between before and after the switch increases. This proofs that there is no parameter set that fits all situations and most likely that cannot exist. Better accuracies could be reached by allowing the robots to collect more evidence. For example, by decreasing the observational interval τ but at the cost of loosing spatial independence of samples (i.e., risking to sample several times from the same tile). Alternatively, more evidence could be collected when robots would have more time between environment switches.

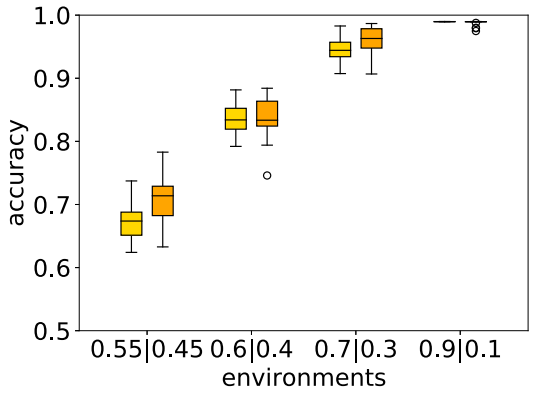


Fig. 6: Accuracy for environments of different difficulty from very hard ($f = 0.55 \rightarrow f = 0.45$) to very simple ($f = 0.9 \rightarrow f = 0.1$) over 20 runs, $N = 100$; $p_{\text{lim}} = 0.99$.

5 Conclusion

We have presented our extension for dynamic environments to the distributed Bayesian approach of collective decision-making by Ebert et al. [4]. Robots continue to collect evidence after having made their decision to monitor possible changes in the environment. A special reset procedure that includes influence by each robot's neighbors, enables each individual and therefore the swarm to avoid lock-in states. We have shown that robot swarms successfully detect environmental changes and switch their opinions collectively in the studied scenario. This adaptability to dynamic environments comes at the cost of slower convergence in periods without change or in fully static environments. Hence, the swarm designer has to decide between quick adaptations and an increased number of false positives or a slowly adapting swarm with possibly more false negatives. More research is required to develop and study robot control algorithms for adaptive collective decision-making, given that most realistic environments are dynamic.

For future work we consider to study different types of dynamic environments, such as slowly changing tile ratios [10]. We plan to explicitly address the trade-off between maximizing specificity and speed of change detection. We also want to study trust systems (i.e., identifying individual robots and monitor their demeanor), because trust is a seemingly underexplored concept in collective decision-making [19].

References

[1] H. Hamann, *Swarm Robotics: A Formal Approach*. Springer, 2018.

[2] G. Valentini, E. Ferrante, and M. Dorigo, "The best-of-n problem in robot

swarms: Formalization, state of the art, and novel perspectives," *Frontiers in Robotics and AI*, vol. 4, p. 9, 2017.

[3] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 Kilobots: Speed vs accuracy in binary discrimination problems," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 3, pp. 553–580, 2016.

[4] 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)*, 2020, pp. 7186–7192.

[5] P. Bartashevich and S. Mostaghim, "Multi-featured collective perception with evidence theory: tackling spatial correlations," *Swarm Intelligence*, vol. 15, pp. 83–110, 2021.

[6] N. R. Franks, A. Dornhaus, J. P. Fitzsimmons, and M. Stevens, "Speed versus accuracy in collective decision making," *Proc. R. Soc. Lond. B*, vol. 270, pp. 2457–2463, 2003.

[7] M. Wahby, J. Petzold, C. Eschke, T. Schmickl, and H. Hamann, "Collective change detection: Adaptivity to dynamic swarm densities and light conditions in robot swarms," in *ALIFE 2019: Conference on Artificial Life*, 2019, pp. 642–649.

[8] Y. Khaluf, P. Simoens, and H. Hamann, "The neglected pieces of designing collective decision-making processes," *Frontiers in Robotics and AI*, vol. 6, p. 16, 2019.

[9] A. Dussutour, M. Beekman, S. C. Nicolis, and B. Meyer, "Noise improves collective decision-making by ants in dynamic environments," *Proc. R. Soc. Lond. B*, vol. 276, pp. 4353–4361, 2009.

[10] M. Divband Soorati, M. Krome, M. Mora-Mendoza, J. Ghofrani, and H. Hamann, "Plasticity in collective decision-making for robots: Creating global reference frames, detecting dynamic environments, and preventing lock-ins," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4100–4105.

[11] G. Valentini and H. Hamann, "Time-variant feedback processes in collective decision-making systems: Influence and effect of dynamic neighborhood sizes," *Swarm Intel.*, vol. 9, no. 2, pp. 153–176, 2015.

[12] T. D. Seeley and P. K. Visscher, "Choosing a home: how the scouts in a honey bee swarm perceive the completion of their group decision making," *Behav. Ecol. Sociobiol.*, vol. 54, pp. 511–520, 2003.

[13] M. Montes de Oca, E. Ferrante, A. Scheidler, C. Pinciroli, M. Birattari, and M. Dorigo, "Majority-rule opinion dynamics with differential latency: a mechanism for self-organized collective decision-making," *Swarm Intelligence*, vol. 5, pp. 305–327, 2011.

[14] A. Reina, G. Valentini, C. Fernández-Oto, M. Dorigo, and V. Trianni, "A design pattern for decentralised decision making," *PLOS ONE*, vol. 10, no. 10, pp. 1–18, 10 2015.

[15] G. Morlino, V. Trianni, E. Tuci, *et al.*, "Collective perception in a swarm of autonomous robots." in *Proc. of the Int. Conf. on Evolutionary Computation (ICEC)*, 2010, pp. 51–59.

[16] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 3293–3298.

[17] J. Ebert and R. Barnes, "Kilosim," Mar. 2020. https://doi.org/10.5281/zenodo.3406865

[18] G. Valentini, H. Hamann, and M. Dorigo, "Self-organized collective decision making: The weighted voter model," in *Proceedings of the 13th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, A. Lomuscio, P. Scerri, A. Bazzan, and M. Huhns, Eds. IFAAMAS, 2014, pp. 45–52.

[19] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Blockchain technology secures robot swarms: A comparison of consensus protocols and their resilience to byzantine robots," *Frontiers in Rob. and AI*, vol. 7, 2020.