

Localization of inexpensive robots with low-bandwidth sensors

Shiling Wang, Francis Colas, Ming Liu, Francesco Mondada, Stéphane Magnenat

▶ **To cite this version:**

Shiling Wang, Francis Colas, Ming Liu, Francesco Mondada, Stéphane Magnenat. Localization of inexpensive robots with low-bandwidth sensors. 13th International Symposium on Distributed Autonomous Robotic Systems, Nov 2016, London, United Kingdom. hal-01379218

HAL Id: hal-01379218
https://inria.hal.science/hal-01379218

Submitted on 11 Oct 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Localization of inexpensive robots with low-bandwidth sensors

Shiling Wang and Francis Colas and Ming Liu and Francesco Mondada and Stéphane Magnenat

1 Introduction

Driven by consumer products, the technologies of electronics, motor and battery have made tremendous progress in the last decades. They are now widely available at prices which make affordable mobile robots a reality. This opens many new opportunities, in particular in the contexts of collective robotics.

Collective and swarm robotics focus on the scalability of systems when the number of robots increases. In that context, global localization is a challenge, whose solution depends on the environment the robots evolve in. A common approach is to measure the distance and orientation between robots [1], but this approach relies on beacons to provide absolute measurements. Yet, several state of the art algorithms require global positioning [2]. In experimental work, it is often provided by an external aid such as a visual tracker, bringing a non-scalable single point of failure into the system, and breaking the distributed aspect. Therefore, there is the need for a distributed and affordable global localization system for collective robotics experiments.

This paper answers this need by providing a distributed system using only approximate dead-reckoning and inexpensive infrared sensors measuring the grayscale intensity of the ground, without knowing the initial pose of the robots. As sensors are mounted on the robot, the localization is a local operation, which makes the approach

Shiling Wang
ETH Zürich, e-mail: shilingwang0621@gmail.com

Francis Colas
INRIA Nancy Grand Est, e-mail: francis.colas@inria.fr

Ming Liu
City University of Hong Kong, e-mail: mingliu@cityu.edu.hk

Francesco Mondada
Mobots, LSRO, EPFL, e-mail: francesco.mondada@epfl.ch

Stéphane Magnenat
Mobots, LSRO, EPFL, e-mail: stephane@magnenat.net

scalable. Our solution is based on the classical Markov [3] and Monte Carlo [4] Localization frameworks that can be seen as respectively a dense and a sampling-based implementation of a recursive Bayesian filter. While these approaches are commonly used in robots with extensive sensing capabilities such as laser scanners, their implementation on extremely low bandwidth sensors is novel, and raises specific questions, such as which distance the robot must travel for the localization to converge.

In this paper, we deploy these algorithms on the Thymio II differential-wheeled mobile robot. The robot reads the intensity of the ground using two infrared sensors (Fig. 1, middle, circled red) and estimates the speed of its wheels by measuring the back electromotive force, which is less precise than encoder-based methods. For evaluation purposes, a Vicon tracking system (`http://www.vicon.com/`) provides the ground-truth pose (Fig. 1, right). Our first contribution is a predictive model of the necessary distance to travel to localize the robot. Our second contribution is a detailed analysis of the performances of the two implementations in comparison with ground-truth data. Our last contribution is an experimental validation of online, real-time localization (Fig. 1, left). The source code and all experimental data are available online[1] and the system can be seen running in a video[2].

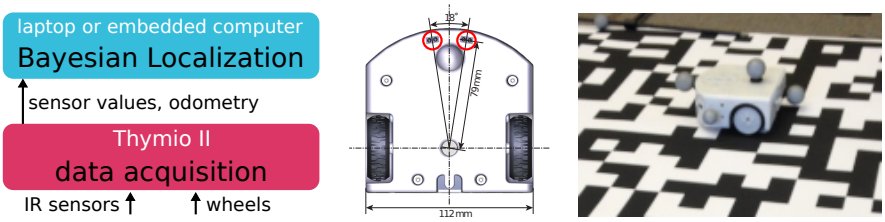


Fig. 1 The block scheme of the online system (left), the Thymio II robot with the placement of its ground sensors (middle), and markers for tracking its ground-truth pose by a Vicon system (right).

2 Related work

The main challenges of solving the localization problem on affordable mobile robots are the constraints on the environment and the limited information content that inexpensive sensors can typically provide.

The work of Kurazume and Nagata [5] first raised the idea of performing inexpensive localization through the cooperative positioning of multiple robots. Prorok et al. [1] is a modern work representative of this approach. These authors have used an infrared-based range and bearing hardware along with a distributed Monte Carlo Localization approach, allowing a group of robots to localize down to a precision

[1] `https://github.com/epfl-mobots/thymio-ground-localisation`
[2] `https://www.youtube.com/watch?v=70euPzixzus`

of 20 cm. However, this methods would require fixed robots acting as beacons to provide absolute positioning. Moreover, both radio and infrared-based range and bearing systems require complex electronics. Finally, when the cost of all robots is added, the system is far from cheap.

A cheaper approach is to use walls around an experimental arena to localize. For example, Zug et al. [6] have developed an algorithm using an array of triangulation-based infrared distance sensors. A Kalman filter algorithm is applied to localize the robot within a 2 by 1 m box. No experimental result is provided, but a simulation shows the estimated error to be within 2 cm. Dias and Ventura [7] used two horizontal lines from a VGA camera to read barcodes on the walls of an arena and localize an e-puck robot. Their system employs an Extended Kalman Filter (EKF) algorithm and reaches a precision of 1 cm and 5°. However, these systems require no obstacles between the robot and the walls, and thus are not scalable to a large number of robots.

This problem can be alleviated by detecting a known pattern visible on the ceiling and fusing this information with odometry. This approach was proven effective for localization using high-quality cameras [4]. Focusing on a low cost, Gutmann et al. [8] have developed a system using only 3 or 4 light detectors, able to localize a mobile robot in a room with an accuracy of 10 cm. However, this method requires a controlled ceiling arrangement and empty space over the robots.

Another approach is to exploit the ground for localization. Park and Hashimoto [9] proposed to localize a mobile robot over a ground equipped with randomly distributed passive RFID tags. The average localization error of this method is lower than 10 cm. However, this approach requires the ground to be equipped with tags which can become expensive and tedious to deploy when the area grows.

The system proposed in this paper builds on the idea of using information from the ground for absolute localization, with performance comparable with related work. To the best of our knowledge, existing solutions for localization of multi-robot systems either rely on an expensive adaptation of the environment using markers or beacons, or necessitate the embedding of expensive sensing capabilities on every robot. In contrast, our system is affordable since it only requires a low number of inexpensive infrared sensors onboard the robot (which are typically used as cliff detectors), and a simple grayscale pattern on the ground that can be achieved using a printed poster.

3 Model

The generic Bayesian filter used to estimate the pose of a robot in its environment uses the following variables:

- $X_{1:t}$ 2-D pose at times $1..t$, consisting of x, y coordinates and an angle θ.
- $Z_{1:t}$ observations at times $1..t$, consisting of the output of the sensors measuring the grayscale intensity of the ground.
- $U_{1:t}$ odometry at times $1..t$, consisting of the left and right wheel speeds.

It is classically formulated as a recursive Bayesian filter with the following joint probability distribution:

$$p(X_{1:t}, Z_{1:t}, U_{1:t}) = p(Z_t|X_t)p(X_t|X_{t-1}, U_t)p(U_t)p(X_{1:t-1}, Z_{1:t-1}, U_{1:t-1}). \quad (1)$$

This filter is based on a few assumptions. First it assumes that the current observation is independent on the past observations, the past states and odometry commands conditionally to the current state. It also features the Markov assumption: the next state is independent on former states, commands and observations conditionally to the previous state and the current command. Finally, it assumes that the actions are independent from the past. All these assumptions are standard and can be found in Kalman filtering and other classical models.

This joint probability distribution allows to formulate the problem as the estimation of the pose X_t at time t given the observations $Z_{1:t}$ and the commands $U_{1:t}$:

$$p(X_t|Z_{1:t}, U_{1:t}) \propto p(Z_t|X_t) \sum_{X_{t-1}} p(X_t|X_{t-1}, U_t)p(X_{t-1}|Z_{1:t-1}, U_{1:t-1}). \quad (2)$$

This inference involves two distributions to be specified: the *observation model* $p(Z_t|X_t)$ and the *motion model* $p(X_t|X_{t-1}, U_t)$, whose parametrizations depend on the robot. In addition, we define a *self-confidence metrics* and outline the implementation.

Observation model. Our observation model $p(Z_t|X_t) = \prod_{i=0,1}^{N_{\text{sensors}}} p(Z_t^i|X_t)$ assumes all sensor noises to be independent, and the ground color to be in the range of $[0,1]$ (0 being black and 1 being white). We take $p(Z_t^i|X_t) \sim \mathcal{N}(v, \sigma_{\text{obs}})$, for robot pose X_t, sensor i, and a corresponding ground intensity v according to the map. The parameter σ_{obs} is selected based on the knowledge of the sensor. Thymio II has two sensors (Fig. 1, center). For binary (black and white) patterns, we chose $\sigma_{\text{obs}} = 0.5$. For grayscale images, based on measurement on a Thymio II, we set σ_{obs} to 0.15.

Motion model. Based on the model of Eliazar et al. [10], we assume that the motion has a Gaussian error model, hence $p(X_t \mid X_{t-1}, U_t) \sim \mathcal{N}(\mu_t, \Sigma_t)$. The mean μ_t is built by accumulating the estimated displacements by dead-reckoning between times $t-1$ and t. Therefore, if Δx_t, Δy_t, $\Delta\theta_t$ are the displacement between $t-1$ and t, expressed in the robot local frame at $t-1$, μ_t is:

$$\mu_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} \text{ with } \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + R(\theta_{t-1}) \begin{bmatrix} \Delta x_t \\ \Delta y_t \\ \Delta\theta_t \end{bmatrix} \quad (3)$$

where $R(\theta)$ is the 2-D rotation matrix of angle θ. The 3×3 diagonal covariance matrix Σ_t is a function of the distance traveled, the amount of rotation, and two parameters α_{xy}, α_θ:

$$\Sigma_t = \begin{bmatrix} \sigma_{\text{xy},t}^2 & 0 & 0 \\ 0 & \sigma_{\text{xy},t}^2 & 0 \\ 0 & 0 & (\alpha_\theta|\Delta\theta_t|)^2 \end{bmatrix} \quad (4)$$

with $\sigma_{\text{xy},t} = \alpha_{\text{xy}}\sqrt{\Delta x_t^2 + \Delta y_t^2}$.

To cope with the possibility of the robot being kidnapped and therefore its pose becoming unknown, a uniform distribution with a weight p_{uniform} is added to X_t. The parameters α_{xy}, α_θ and p_{uniform} are estimated using maximum likelihood (Sect. 5).

Implementations. We compare two variants of this filter. In Markov Localization, the distributions are discretized using regular grids [3]. For our experiments, the x, y cell resolution is 1 cm and the angular resolution varies from 20° (18 discretization steps for 360°) to 5° (72 discretization steps for 360°). The estimated pose is the coordinates of the cell of maximum probability. In Monte Carlo Localization [4], the distributions are represented using samples in a particle filter. In order to extract a single pose estimate out of the many particles, we find a maximum density area in which we average the particles. It is similar to a 1-point RANSAC scheme [11]. We implemented both algorithms in Python with some Cython (`http://www.cython.org`) procedures used for time-critical inner loops. The algorithms run on an embedded computer or laptop (Fig. 1, left). Thymio II is programmed through the ASEBA framework [12], which connects to Python using D-Bus.

Self confidence. We define a self-confidence term that corresponds to the ratio of the probability mass of $p(X_t)$ that is within a distance d_{xy} and an angle difference d_θ to the estimated pose. In our experiments, we use $d_{\text{xy}} = 3\,\text{cm}$ and $d_\theta = 10°$.

4 Theoretical analysis of convergence

One can estimate the time required for the robot to localize itself in a given space, by comparing the information needed and the information gained while traveling.

Information need. For the Markov Localization approach, there is a given number of discrete cells. The amount of information needed to unambiguously specify one among them all is $\text{H}_{\text{loc}} = \log_2(N_{\text{cells}})$, with N_{cells} the number of cells.

Information gain. We can estimate the information gain at each time step. Let us consider the case of a binary map and a binary sensor. This sensor ideally yields 1 bit of information per measurement. In practice, there is a loss in information due to the sensor noise, characterized by the p_{correct} probability of the sensor to be correct:

$$\text{H}_{\text{noise}} = \text{H}_{\text{b}}(1 - p_{\text{correct}}), \tag{5}$$

where H_{b} is the binary entropy function: $\text{H}_{\text{b}}(p) = -p\log_2(p) - (1-p)\log_2(1-p)$.

We also need to take into account that the sensor measurements are not completely independent. For example, when the robot is not moving, it always observes the same place and thus cannot really gain additional information besides being sure of the intensity of the current pixel. In a discretized world, we thus need to estimate the probability of having changed cell in order to observe something new, which depends on the distance traveled and the size of the cells. This problem is equivalent to the Buffon-Laplace needle problem of finding the probability for a needle thrown

randomly on a grid to actually intersect the grid[3] [13]. In our case, the probability of changing cell is given by:

$$p_{\text{diff}} = \frac{4dh - d^2}{\pi h^2}, \tag{6}$$

with d the distance traveled and h the size of the cells.

We can then compute the conditional entropy for two successive ideal binary measurements O_{t-1} and O_t separated by d based on the conditional probability. There are two cases: either the robot has not moved enough to change cell (with probability $1 - p_{\text{diff}}$) and the new observation is the same as the old, or the robot has changed cell (with probability p_{diff}) and the new observation has the same probability to be the same or the opposite of the old. This can be summarized by the following conditional probability distribution (b/w = black/white):

$$p(O_t = \{\text{b/w}\} \mid O_{t-1} = \{\text{b/w}\}) = \begin{pmatrix} (1 - p_{\text{diff}}) + p_{\text{diff}}/2 & p_{\text{diff}}/2 \\ p_{\text{diff}}/2 & (1 - p_{\text{diff}}) + p_{\text{diff}}/2 \end{pmatrix} \tag{7}$$

After rearranging the terms of the conditional entropy, the loss of information due to redundancy in the traveled distance is:

$$\mathrm{H}_{\text{loss,d}} = 1 - \mathrm{H}_{\text{b}}(p_{\text{diff}}/2). \tag{8}$$

There is also redundancy between several sensors placed on the same robot. The probability that they see the same cell based on the distance between them is exactly the same as the probability of a sensor to see the same cell after a displacement of the same distance. The information loss due to the redundancy from the sensor placement is noted $\mathrm{H}_{\text{sensors}}$ and follows the same formula as $\mathrm{H}_{\text{loss,d}}$, but with d being the distance between the two sensors in Eq. 6.

Finally, we can approximate the information that our robot gathers at each time step by assuming that the trajectory of the robot does not loop (no redundancy between distant time steps) and that the trajectories of the sensors are independent. The information gathered in a time step is then:

$$\begin{aligned} \mathrm{H}(O_t^1, O_t^2 \mid O_{1:t-1}^1, O_{1:t-1}^2) &= \mathrm{H}(O_t^1, O_t^2 \mid O_{t-1}^1, O_{t-1}^2) \\ &= \mathrm{H}(O_t^1 \mid O_{t-1}^1) + \mathrm{H}(O_t^2 \mid O_{t-1}^2) - \mathrm{H}(O_t^2 \mid O_t^1) \\ &= 2 \cdot (1 - \mathrm{H}_{\text{noise}} - \mathrm{H}_{\text{loss,d}}) - \mathrm{H}_{\text{sensors}} \end{aligned} \tag{9}$$

This formula also ignores the uncertainty in the robot motion. With these assumptions, it is an upper bound on the average information gain.

Outlook. Faster localization can be achieved by moving at a greater speed to reduce redundancy in the successive measurements, with proportional increase in sampling frequency. If designing a new robot, better sensors would reduce cross-over noise. Setting the sensors apart would also reduce the redundancy between their information but, for our specific grid size, they are sufficiently separated in Thymio II.

[3] The needle is the segment joining the start and end points of the robot movement and the grid is the borders of the cells.

5 Empirical analysis of performance

To evaluate the performance of our localization algorithms, we remotely controlled the robot and recorded datasets covering all possible robot motions:

- `trajectory 1` and `2`: The robot alternates forward and backward movements with rotations on spot, at a speed of 3–5 cm/s.
- `linear trajectory`: The robot simply goes straight ahead along the x-axis of the map, at a speed of 3–5 cm/s.
- `trajectory with kidnapping`: The robot alternates phases of forward movement, at a speed of 15 cm/s, and turning on spot. To test the algorithm's ability of recovering from catastrophic localization failures, we perform "robot kidnapping" by relocating the robot to another part of the map every minute.

The robot moves on a 150×150 cm ground pattern containing 50×50 cells of 3×3 cm, each randomly black or white (Fig. 1, right). The robot is connected to ROS to synchronize its sensor values and odometry information with ground-truth data from Vicon, sampled at a period of 0.3 s. We chose this period so that with basic trajectories, at maximum speed the robot travels approximately half the length of one cell between every sample.

Parameter estimation. We estimated the noise parameters of the motion model using maximum likelihood, considering the error between the ground truth and the odometry data in the local frame between two time steps. Using `trajectory 1` and `trajectory 2`, we found the values for α_{xy} and α_θ to be in the order of 0.1. Similarly, using `trajectory with kidnapping`, we also found the value for p_{uniform} to be in the order of 0.1.

Basic localization. Fig. 2 shows the error in position and orientation for the first two trajectories. In these plots, *distance traveled* represents the cumulative distance traveled by the center point between the two ground sensors of the robot.

For the Markov Localization approach, all discretization resolutions allow the robot to localize down to a precision of 3 cm and 5°. However, in `trajectory 1`, the resolution of 18 discretization steps is not enough to keep tracking the orientation at a distance traveled of 50 cm and 80 cm. These both correspond to the robot rotating on spot. Finer discretizations do not exhibit this problem, they are more robust and have similar precision. Therefore, we see that an angular discretization of 36 (10° resolution) is sufficient to provide accurate tracking. In `trajectory 2`, we see that an angular discretization of 54 allows for a better angular precision than 36, but 72 does not improve over 54. All discretizations provide equal position precision.

For the Monte Carlo Localization approach, we see that on `trajectory 1`, the robot localizes already with 50k particles, but in twice the distance it takes with 100k particles. Increasing the number of particles beyond this value only marginally decreases localization time. While 50k particles are sufficient to localize on this run on average, in some trials, the robot loses orientation when it turns on spot. On `trajectory 2`, using 50k particles is not enough to localize the robot. Increasing this number to 100k leads to a good localization, except after the robot has traveled

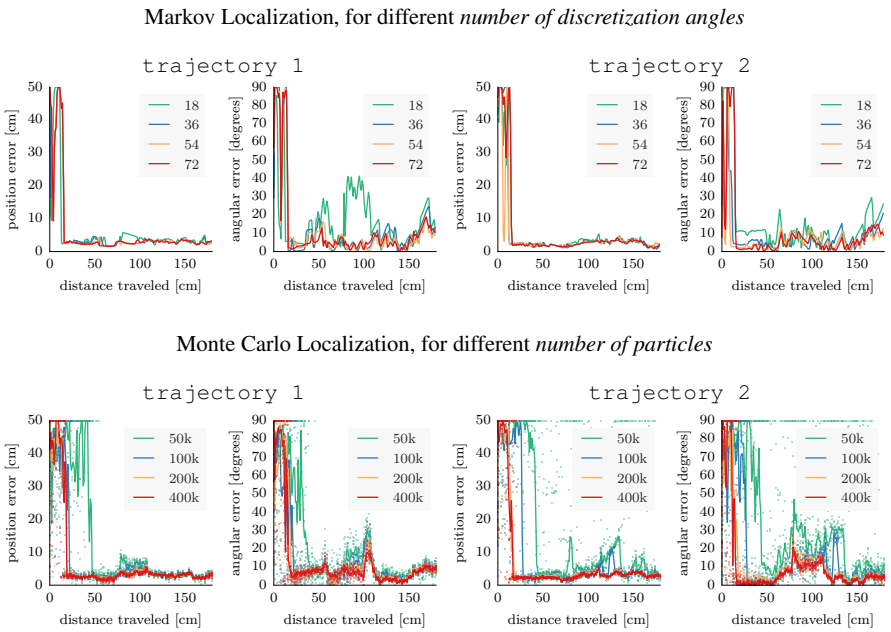


Fig. 2 The error between the estimation by the localization algorithm and the ground truth on `trajectory 1` and `trajectory 2`. For Monte Carlo Localization, the solid lines show the average over 10 trials, while the light dots show the individual trials.

130 cm. This corresponds to a long moment during which the robot rotates on spot, leading to less information acquisition, and therefore degraded performances.

Overall, both approaches have similar accuracy. When angular precision is critical, the Monte Carlo Localization approach might achieve better performance, as the Markov Localization approach is limited in precision by its angular discretization.

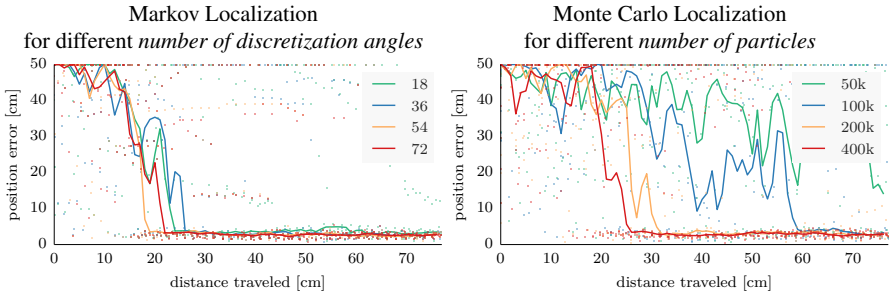


Fig. 3 The error between the estimation by the localization algorithm and the ground truth on 10 segments from the first two trajectories. The solid lines show the median while the light dots show the individual trials.

Distance to converge. Fig. 3 shows the error in position for 5 different starting points in each of the first two trajectories, in which the robot moves at a speed of 3–5 cm/s. We see that, with the Markov Localization approach, the correct pose is found after about 20 cm. There are also outliers: this happens when the robot is turning on spot, in which case there is not enough information to localize the robot. With 400k particles, the Monte Carlo Localization approach also converges after about 20 cm. Decreasing the number of particles quickly increases the distance needed for convergence, reaching 60 cm for 100k particles. Using only 50k particles, some trajectory segments fail to converge within 80 cm length.

It is interesting to compare these distances with theoretical estimates (see Sect. 4). One difficulty is that the theoretical model specifies the probability p_{correct} of the sensor to be correct. In our observation model, instead, we specify σ_{obs} the noise of the sensor. Therefore, we propose to compute p_{correct} from σ_{obs} by considering a sensor positioned over a black ground, and by assuming that all values below a threshold of 0.5 (0 being black and 1 being white) are correctly read:

$$p_{\text{correct}} = p(Z < 0.5 | X = 0) = \int_{-\infty}^{0.5} \mathcal{N}(0, \sigma_{\text{obs}}) \tag{10}$$

In these runs, we have assumed $\sigma_{\text{obs}} = 0.5$, leading to $p_{\text{correct}} = 0.84$ (Eq. 10) and $\mathrm{H}_{\text{noise}} = 0.63$ (Eq. 5). Our robot moves at 3 cm/s with a time step of 0.3 s on a grid of 3 cm×3 cm cells in which the color is known to be similar; this yields $\mathrm{H}_{\text{loss,d}} = 0.33$ (Eq. 8). Moreover, the sensors are 2.2 cm apart, which yields $\mathrm{H}_{\text{sensors}} = 0.041$. As such the robot gathers on average at most 0.036 bit per time step, or 0.040 bit/cm (Eq. 9). With a 150 cm×150 cm environment discretized with cells of 1 cm and 5° angle, the amount of information needed for the localization is $\mathrm{H}_{\text{loc}} = 20.6$. This means that, on average, the robot should not be localized before traveling around 520 cm[4]. This distance is far larger than the observed one of about 20 cm. The reason is that using $\sigma_{\text{obs}} = 0.5$ for the binary case was an arbitrary choice and therefore the value of p_{correct} does not correspond to the reality.

Computing the value of p_{correct} to match the observed convergence distance by inverting the computation, we find 0.97, corresponding to $\sigma_{\text{obs}} = 0.26$. If we use $\sigma_{\text{obs}} = 0.15$ as measured on grayscale images, we have $p_{\text{correct}} = 0.99957$. This leads to a minimum theoretical localization distance of about 14.3 cm. This value is slightly lower than our experimental results, which makes sense as it is a lower bound. Moreover, this low bound would be attained with a perfect filter observing a perfect pattern. Our filter is not perfect, because we have run it with an overestimated σ_{obs} of 0.5. Nevertheless, the filter works well, showing that it degrades gracefully with respect to imprecision in its parameters. This is important in practice, as users might not be able to provide extremely precise parameters.

Effect of map size. Fig. 4 shows the effect on the distance of convergence when the map size is reduced. The robot runs linearly on one quarter of the map, while the Markov Localization is performed on the whole map, half of it, and a quarter of

[4] see `https://github.com/epfl-mobots/thymio-ground-localisation/blob/master/theory/d_conv_from_pcorrect.py`

it. We see that reducing the map size does reduce the distance traveled necessary to converge, in accordance to the theory, because less information have to be acquired to reduce the uncertainty of the pose.

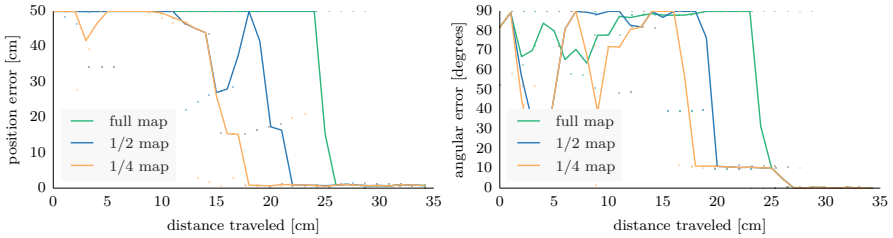


Fig. 4 The error between the estimation by the localization algorithm and the ground truth, using Markov Localization, for different map sizes on `linear trajectory`. The solid lines show the median over 3 different trajectory parts, while the light dots show the individual parts.

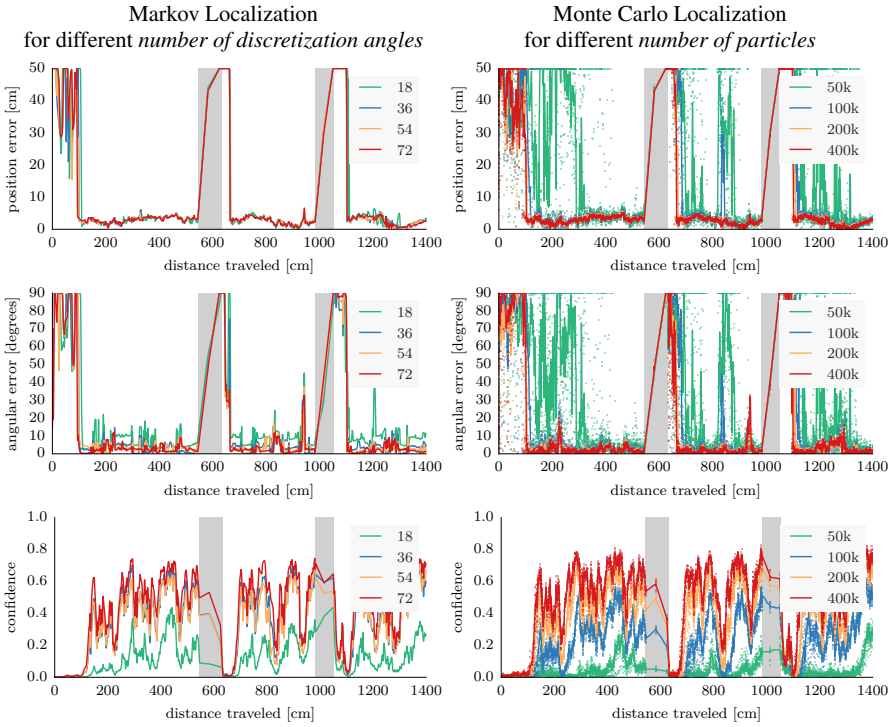


Fig. 5 The error between the estimation by the localization algorithm and the ground truth, and the self confidence of the algorithm, on the run with kidnapping. For Monte Carlo Localization, the solid lines show the median over 10 trials, while the light dots show the individual trials. The gray areas show the time during the robot is being held in the air at the occasion of kidnapping.

Robot kidnapping. Fig. 5 shows the error in position, orientation and the self confidence for the run with kidnapping. In this run, the robot is kidnapped twice, after having traveled 550 cm and 1000 cm. It takes the robot approximately 100 cm to re-localize, and does so successfully with both Markov and Monte Carlo Localization approaches. This difference of distance with previous runs is mostly due to the speed of the robot, which is about 5 times faster. With the Markov Localization approach, all discretization resolutions are approximately equivalent in position performance, but 18 leads to a lower orientation precision, as well as to a lower self confidence, due to the large discretization step. Other resolutions have a confidence of about 0.5 when the robot is localized, and this value drops below 0.1 after kidnapping, clearly showing that the algorithm is able to assess its status of being lost.

With the Monte Carlo Localization approach, the robot localizes most of the time with 100k particles, and always with 200k particles or more. With 50k particles, the robot eventually localizes, but this might take more than 2 meters of traveled distance. We see that the self confidence increases with more particles, and, similarly to the Markov Localization approach, drops after kidnapping. This confirms the effectiveness of the self confidence measure.

duration [s]	**0.64**	**1.43**	**2.15**	**2.94**	**1.97**	**2.97**	**6.08**	**12.22**
algorithm	Markov Localization				Monte Carlo Localization			
parameter	18	36	54	72	50k	100k	200k	400k
	discretization angles				particles			

Table 1 The execution duration of one step for the two algorithms.

Computational cost. Table 1 shows the execution duration of one step for the two algorithms with different parameters. These data were measured on a Lenovo laptop T450s with an Intel Core i7 5600U processor running at 2.6 GHz, and are averages over the two first trajectories. We see that with the Markov Localization approach, the duration scales linearly with the number of discretization angles. With the Monte Carlo Localization approach, the scaling is linear but amortized (50k particles is not twice faster as 100k). This is due to the selection of pose estimate, which uses a RANSAC approach and is therefore independent of the number of particles. For similar localization accuracy, the Monte Carlo Localization approach is slower than the Markov Localization approach, and therefore we suggest to use the former with an angular discretization of 36 in practical applications. However, the Monte Carlo Localization approach might be preferred if a high angular precision is required, but at least 100k particles are necessary for proper localization.

6 Real-time localization

Fig. 6 shows the performance of the online, real-time localization of a Wireless Thymio II (Fig. 1, left). We tested different grayscale images of 59×42 cm, printed

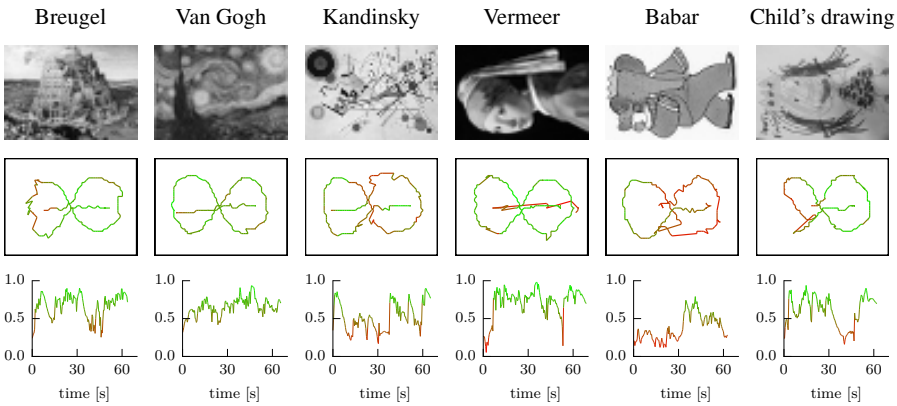


Fig. 6 The image, estimated trajectory and self confidence for different 59×42 cm grayscale patterns printed on A2 papers with a resolution of 1 pixel/cm. The robot is remotely controlled to draw a 8-shape figure, with similar motor commands for each image. Markov Localization with 36 discretization angles used. The color of the lines vary from red (confidence 0) to green (confidence 1). Trajectories are plotted starting from a confidence level of 0.2.

on A2 paper sheets with a resolution of 1 pixel/cm. We used Markov Localization with an angular discretization of 36 and a spatial resolution of 1 cm; the localization is performed every 0.4 s. The localization algorithm runs on a Lenovo laptop X1 3rd generation with an Intel Core i7-5500U processor running at 2.4 GHz, at a CPU load of approximately 80 %. The algorithm localizes the robot globally on all images. The images with more contrast lead to a more robust and faster localization, while the ones with lower contrast lead to more imprecise trajectories. In Fig. 6, we see that the parts of trajectories in red are less precise than the ones in green. This shows that the self confidence measure is effective in assessing the quality of the localization.

Embedded use. The runtime cost is dominated by the motion model, as it involves convoluting a small window for every cell of the probability space. With the current parameters, this leads to approximately 250 floating-point multiplications per cell. For comparison, VideoCore® IV, the built-in GPU in the Raspberry Pi 3 (costing €33), has a peak processing power of 28 GFLOPS. Assuming that, due to limitations in bandwidth and dependencies between data, it can only be used at 10 % of its peak floating point throughput, it could run a GPU version of our algorithm at 10 Hz on a map of 1.8 by 1.8 meters with a resolution of 1 cm. Therefore, our approach can truely be used for distributed collective experiments with affordable robots.

Scalability. The computational speed of the system and the necessary distance to converge scale linearly with the size of the localized area. Also, a prolonged use of the robot might wear off the ground, slightly shifting the grayscale intensities and jeopardizing the localization. Nevertheless, modern printing options allow to use strong supports that only degrade slowly with time. Therefore, our system can be helpful for conducting collective experiments in a laboratory environment.

Optimality. An interesting question is what is an optimal pattern. However, this question has two sides. On the one hand, a pattern could be optimal in term of

quantity of information, allowing the robot to localize as fast as possible. In that case, a white-noise pattern would be ideal. On the other hand, if the robot has to interact with humans, or only be monitored by humans, such a pattern would be highly disturbing. Our experimental results with paintings and drawings show that, even with a pattern having a lot of regularities and symmetries, our system is still able to localize. Therefore, we believe it is usable with natural patterns such as photographies, drawings and grayscale schematics.

7 Conclusion

In this paper, we have implemented and empirically evaluated Markov- and Monte Carlo-based approaches for localizing mobile robots on a known ground visual pattern. Each robot requires only inexpensive infrared sensors and approximate odometry information. We have shown that both approaches allow successful localization without knowing the initial pose of the robot, and that their performances and computational requirements are of a similar order of magnitude. Real-time localization was successful with a large variety of A2 grayscale images, using the Markov Localization approach with 36 discretization steps for angle. Should larger patterns be desired, the code could be further optimized by implementing it in the GPU, allowing it to run on inexpensive boards such as the Raspberry Pi.

In addition, we have outlined, and empirically validated, a method to estimate the localization performance in function of the sensor configuration. This method provides a guide for taking decisions about the placement of sensors in a future robot design: localization performance can be improved by placing the sensors far apart on a line perpendicular to the direction of movement of the robot; moreover, more sensors allow for collecting more information, if they are separated by the size of the smallest visual structure in the map.

These contributions to the state of the art enable absolute positioning of inexpensive mobile robots costing in the € 100 range. In the context of distributed collective robotics, they provide a solid base on which to build navigation or behavioral algorithms.

Acknowledgements The authors thank Emmanuel Eckard and Mordechai Ben-Ari for insightful comments on the manuscript and Ramiz Morina for his drawings. This research was supported by the Swiss National Center of Competence in Research “Robotics”.

References

1. A. Prorok, A. Bahr, and A. Martinoli, “Low-cost collaborative localization for large-scale multi-robot systems,” in *International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2012, pp. 4236–4241.

2. A. Breitenmoser and A. Martinoli, "On combining multi-robot coverage and reciprocal collision avoidance," in *Distributed Autonomous Robotic Systems*. Springer, 2016, pp. 49–64.
3. D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.
4. F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *International Conference on Robotics and Automation (ICRA)*. IEEE Press, 1999, pp. 1322–1328.
5. R. Y. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *International Conference on Robotics and Automation (ICRA)*. IEEE Press, 1994, pp. 1250–1257.
6. S. Zug, C. Steup, A. Dietrich, and K. Brezhnyev, "Design and implementation of a small size robot localization system," in *International Symposium on Robotic and Sensors Environments (ROSE)*. IEEE Press, 2011, pp. 25–30.
7. D. Dias and R. Ventura, "Absolute localization for low capability robots in structured environments using barcode landmarks," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 7, no. 1, pp. 28–34, 2013.
8. J.-S. Gutmann, P. Fong, L. Chiu, and M. Munich, "Challenges of designing a low-cost indoor localization system using active beacons," in *International Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE Press, 2013, pp. 1–6.
9. S. Park and S. Hashimoto, "An approach for mobile robot navigation under randomly distributed passive rfid environment," in *International Conference on Mechatronics (ICM)*. IEEE Press, 2009, pp. 1–6.
10. A. I. Eliazar and R. Parr, "Learning probabilistic motion models for mobile robots," in *Twenty-first International Conference on Machine Learning*. ACM, 2004, pp. 32–39.
11. M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
12. S. Magnenat, P. Rétornaz, M. Bonani, V. Longchamp, and F. Mondada, "ASEBA: A Modular Architecture for Event-Based Control of Complex Robots," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 2, pp. 321–329, 2010.
13. P.-S. Laplace, *Théorie analytique des probabilités, 3rd rev.* Paris: Veuve Courcier, 1820.