Cataglyphis ant navigation strategies solve the global localization problem in robots with binary sensors

Nils Rottmann[1], Ralf Bruder[1], Achim Schweikard[1] and Elmar Rueckert[1]
[1]*Institute for Robotics and Cognitive Systems, University of Luebeck, Ratzeburger Allee 160, 23562 Luebeck, Germany*
{*rottmann,bruder,schweikard,rueckert*}*@rob.uni-luebeck.de*

Keywords: Global Localization, Binary Measurements, Land Navigation, Path Integration, Cataglyphis

Abstract: Low cost robots, such as vacuum cleaners or lawn mowers, employ simplistic and often random navigation policies. Although a large number of sophisticated localization and planning approaches exist, they require additional sensors like LIDAR sensors, cameras or time of flight sensors. In this work, we propose a global localization method biologically inspired by simple insects, such as the ant Cataglyphis that is able to return from distant locations to its nest in the desert without any or with limited perceptual cues. Like in Cataglyphis, the underlying idea of our localization approach is to first compute a pose estimate from pro-prioceptual sensors only, using land navigation, and thereafter refine the estimate through a systematic search in a particle filter that integrates the rare visual feedback.
In simulation experiments in multiple environments, we demonstrated that this bioinspired principle can be used to compute accurate pose estimates from binary visual cues only. Such intelligent localization strategies can improve the performance of any robot with limited sensing capabilities such as household robots or toys.

1 INTRODUCTION

Precise navigation and localization abilities are crucial for animal and robots. These elementary skills are far developed in complex animals like mammals (O'keefe and Nadel, 1978), (Redish et al., 1999). Experimental studies have shown that transient neural firing patterns in place cells in the hippocampus can predict the animals location or even complete future routes (Pfeiffer and Foster, 2013), (Foster and Wilson, 2006), (Johnson and Redish, 2007), (Carr et al., 2011). These neural findings have inspired many computational models based on attractor networks (Hopfield, 1982), (Samsonovich and McNaughton, 1997), (McNaughton et al., 2006), (Erdem and Hasselmo, 2012) and have been successfully applied to humanoid robot motion planning tasks (Rueckert et al., 2016), (Tanneberg et al., 2019).

In this work we took inspiration from simpler insects which are able to precisely navigate by using sparse perceptual and pro-prioceptual sensory information. For example, the desert ant Cataglyphis (Figure 1a) employs basic path integration, visual piloting and systematic search strategies to navigate back to its nest from distant locations several hundred meters away (Wehner, 1987). While computational models

(a) "Cataglyphis nodus" by (Bobzin, 2013).

(b) Create 2

Figure 1: The Cataglyphis ant and the robot Create 2 from the company iRobot.

of such navigation skills lead to better understanding of the neural implementation in the insect (Lambrinos et al., 2000), they also have a strong impact on a large number of practical robotic applications. Especially, non-industrial autonomous systems like household robots (Figure 1b) or robotic toys require precise navigation features utilizing low-cost sensory hardware (Lee and Jung, 2012), (Miller and Slack, 1995), (Nebot and Durrant-Whyte, 1999).

Despite this obvious need for precise localization in low cost systems, most related work in mobile robot navigation can be categorized into two *extreme* cases. Either computational and monetary expensive sensors like cameras or laser range finders are installed (Su

et al., 2017), (Feng et al., 2017), (Ito et al., 2014) or simplistic navigation strategies such as a random walks are used like in the current purchasable household robots (Tribelhorn and Dodds, 2007). When using limited sensing, only few navigation strategies have been proposed. O'Kane et al. investigated how complex a sensor system of a robot really has to be in order to localize itself. Therefore, they used a minimalist approach with contact sensors, a compass, angular and linear odometers in three different sensor configurations (O'Kane and LaValle, 2007). Erickson et al. addressed the localization problem for a blind robot with only a clock and a contact sensor. They used probabilistic techniques, where they discretized the boundary of the environment into small cells. A probability $P_{k,i}$ of the robot being in the cell i at the time step k is allocated to each one and updated in every iteration. For active localization, they proposed an entropy based approach in order to determine uncertainty-reducing motions (Erickson et al., 2008). Stavrou and Panayiotou on the other hand proposed a localization method based on Monte Carlo Localization (Dellaert et al., 1999) using only a single short-range sensor in a fixed position (Stavrou and Panayiotou, 2012). The open challenge however that remains is how to efficiently localize a robot equipped only with a single binary sensor and odometer. Such tasks arise, for example, especially with autonomous lawn mowing robots. They usually use a wire signal to detect whether they are on the area assigned to them or not. There are also sensors that detect the moisture on the surface and can thus detect grass (Bernini, 2009). All these sensors usually return a binary signal indicating whether the sensor is in the field or outside. The aforementioned approaches can either not be applied to such localization tasks because they require low-range sensors (Stavrou and Panayiotou, 2012) or they are trying to solve the problem with even less sensor information (Erickson et al., 2008). A direct comparison between these approaches and our method is not possible since different sensors types are used. However, to be able to compare the estimation accuracy of our methods with the aforementioned algorithms, we created our test environments similar to the test envrionment used in (Stavrou and Panayiotou, 2012).

We propose an efficient and simple method for global localization based only on odometry data and binary measurements which can be used in real time on a low-cost robot with only limited computational resources. This work demonstrates for the first time how basic navigation principles of the Cataglyphis ant can be implemented in an autonomous outdoor robot.

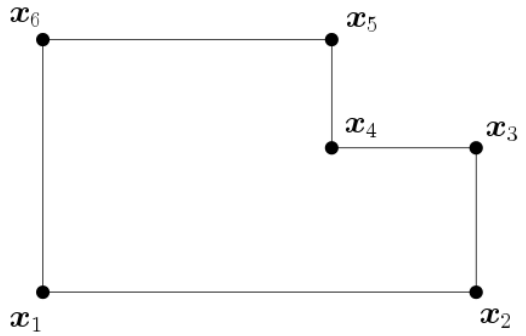

Figure 2: Boundary Map: An example map of an environment which defines the boundary by connecting vertices $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_6]$ to a polygon shape.

Our approach has the potential to replace the currently installed naive random walk behavior in most low cost robots to an intelligent localization and planning strategy inspired by insects. The methods we used are discussed in Section II. We first present a wall following scheme which is required to steer the robot along the boundary line of the map based on the binary measurement data. Inspired by path integration we determine dominant points (DPs) to represent the path estimated by the odometry. We then generate a first estimate of the pose of the robot using land navigation. Therefore, we compare the shape of the estimated path with the given boundary line. The generated pose estimate allows to systematically search for the true pose in the given area using a particle filter. This leads to a more accurate localization of the robot. Here a particle filter is required since other localization techniques, such as Kalman Filter, are not able to handle binary measurement data. In Section III, we evaluate our approach in two simulated environments and show that it leads to accurate pose estimates of the robot. We conclude in Section IV.

2 METHODS

For our proposed localization method, we assume that a map of the environment is given as a boundary map $\mathcal{M}$ defined as a polygon with vertices $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]$ with $\boldsymbol{x}_i \in \mathbb{R}^2, \forall i \in [1, n]$. In Figure 2 such a boundary map is shown. As sensor signals we only receive the binary information $s \in \{0, 1\}$, where 0 means that the sensor is outside of the map and 1 means the sensor is within the boundary defined by the polygon. We are using a simple differential drive robot controlled by the desired linear and angular velocities v and ω. Furthermore, odometry information is assumed to be given. We assume the transform between the sensor position and the odom-

etry frame, which is also the main frame of the robot and has the position $[x_\mathrm{r}, y_\mathrm{r}]^\top$ in world coordinates, is known as

$$\begin{bmatrix} x_\mathrm{s} \\ y_\mathrm{s} \end{bmatrix} = \begin{bmatrix} x_\mathrm{r} \\ y_\mathrm{r} \end{bmatrix} + \boldsymbol{R}(\varphi_\mathrm{r}) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}. \tag{1}$$

Here, $\boldsymbol{R}(\varphi_\mathrm{r})$ is the two dimensional rotation matrix with the orientation angle φ_r of the robot and Δx, Δy the distances between odometry frame and sensor position in robot coordinates. In Figure 3 the setup for the differential drive robot is depicted. For our proposed localization method, it is required that there is a lever arm between the robots frame and the sensor, such that if only the orientation of the robot changes the position of the sensor changes too.

2.1 Wall Follower

Since only at the boundary we are able to generate sensor data useful for localization, a wall follower is required. The proposed control algorithm allows to navigate the robot along the boundary line even if the binary measurements are corrupted by noise. Thereby, to increase the scanning area, the robot moves in wiggly lines along the boundary such that the mean of the sensor detection is $\mu_\mathrm{d} = 0.5$. For the update step of the sensor mean we use exponential smoothing

$$\mu_\mathrm{d} \leftarrow a_\mu \mu_\mathrm{d} + (1 - a_\mu) s, \tag{2}$$

starting with an initial mean of $\mu_\mathrm{d} = 1.0$ (assumption: start within the field). Here a_μ defines the update rate of the mean μ_d. We then calculate the difference between the desired mean and the actual mean

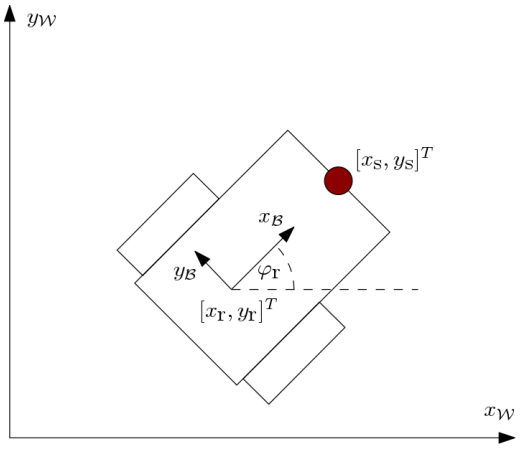

Figure 3: A differential drive robot within the 2-dimensional world frame given by $\mathcal{W}$. The robots main frame $\mathcal{B}$ is determined by the frame for the odometry. The robots position is given by x_r, y_r and the sensor position by x_s, y_s.

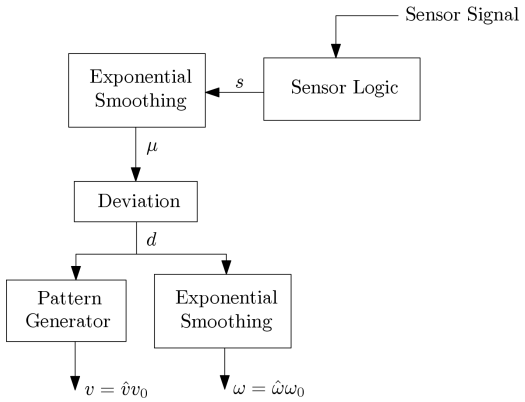

Figure 4: Wall Follower: Control method blocks representing the wall following algorithm.

$$d = 2\left(\frac{1}{2} - \mu_\mathrm{d}\right). \tag{3}$$

The difference is scaled onto the range $[-1, 1]$. We now use relative linear and angular velocities $\hat{v}$ and $\hat{\omega}$ for the control algorithm. They have to be multiplied with the maximum allowed velocities for the given robotic system v_0 and ω_0 in order to get the desired velocities v and ω. Using the difference d from Equation (3), we update the relative forward velocity for the differential drive robot $\hat{v}$ using again exponential smoothing

$$\hat{v} \leftarrow a_v \hat{v} + (1 - a_v)(1 - |d|). \tag{4}$$

Here, a_v defines an update rate and $|d|$ the absolute value of d. The relative velocity increases the longer and better the desired mean of $\mu_\mathrm{d} = 0.5$ is reached. Thereby, the relative velocity is always in the range of $\hat{v} \in [0, 1]$. The relative angular velocity of the robot is determined using the deviation d to the desired mean and a stabilizing term $\cos\left(2\pi \frac{c}{M}\right)$ to ensure robustness against noise corrupted measurements

$$\hat{\omega} = \frac{1}{2}\left(d + \cos\left(2\pi \frac{k}{K}\right)\right). \tag{5}$$

Here, k is a counter variable and K the counter divider. Thus, K should be chosen accordingly to the frequency of the operating system. A diagram of the controller as well as pseudo code can be found in Figure 4 and Algorithm (1) respectively.

2.2 Path Integration / Land Navigation

In order to get a first pose estimate for the robot we use land navigation between the given map $\mathcal{M}$ and the driven path $\mathcal{P}$ along the boundary line.

Algorithm 1 Wall Follower

- **Parameters**
  K, a_μ, a_v, v_0, ω_0
- **Inputs**
  s
- **Outputs**
  v, ω

$k = 0$
$\mu_d = 1.0$ ▷ start within the field
while true **do**
  if Mode == 0 **then** ▷ search for boundary
    $\mu_d \leftarrow a_\mu \mu_d + (1-a_\mu)s$
    if $\mu_d > 0.5$ **then**
      $v = v_0$
      $\omega = 0$
    else
      Mode $= 1$
    end if
  else if Mode == 1 **then** ▷ wall following
    $\mu_d \leftarrow a_\mu \mu_d + (1-a_\mu)s$
    $d = 2 \cdot (0.5 - \mu_d)$
    $\hat{v} \leftarrow a_v \hat{v} + (1-a_v)(1-||d||)$
    $v = \hat{v} v_0$
    $\omega = \left(d + \cos\left(2\pi \frac{k}{K}\right)\right) \cdot 0.5 \cdot \omega_0$
  end if
  $k \leftarrow k+1$
  return v, ω
end while

First, we start by generating a piecewise linear function of the orientation of the boundary line in regard to the length $\theta_b = \theta_b(x)$. This function represents the shape of the boundary and is defined as

$$\theta_b(x) = \phi_i \quad \text{for} \quad l_i \leq x < l_{i+1}, i = 1,2,\ldots,2n. \quad (6)$$

with

$$\begin{aligned} \phi_{i+1} &= \phi_i + \Delta\theta_i \\ l_{i+1} &= l_i + ||\boldsymbol{v}_i||. \end{aligned} \quad (7)$$

Here, $||\boldsymbol{v}_i||$ defines the Euclidean norm for the vector $\boldsymbol{v}_i$. We start with $\phi_1 = 0$ and $l_1 = 0$. Thereby, $\boldsymbol{v}_i = \hat{X}_{i+1} - \hat{X}_i$ with $\hat{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n, \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n, \boldsymbol{x}_1]$ consisting of the vertices X of the map and $\Delta\theta_i$ as the difference in orientation between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i-1}$.

Second, in order to represent the path driven by the robot as vertices connected by straight lines (similar to the map representation), we generate dominant points (DPs). Therefore, we are simply using the estimated position from the odometry of our robot starting with $DP = [\boldsymbol{x}_0]$ at the first contact with the boundary line. We also initialize a temporary set $S = [\boldsymbol{x}_0]$

Algorithm 2 DP Generation

- **Parameters**
  $L_{\min}$, $e_{\max}$
- **Inputs**
  $\boldsymbol{x}$
- **Outputs**
  DP

$DP = [\boldsymbol{x}_0]$
$S = [\boldsymbol{x}_0]$
if new $\boldsymbol{x}$ available **then**
  $d = ||DP_{\text{end}} - \boldsymbol{x}||$
  if $d < L_{\min}$ **then**
    $S \leftarrow [S, \boldsymbol{x}]$
  else
    $S_{\text{tmp}} = [S, \boldsymbol{x}]$
    $e = \text{errorLineFit}(S_{\text{tmp}})$
    if $e < e_{\max}$ **then**
      $S \leftarrow S_{\text{tmp}}$
    else
      $DP \leftarrow [DP, S_{\text{end}}]$
      $S \leftarrow [S_{\text{end}}, \boldsymbol{x}]$
    end if
  end if
end if

which holds all the actual points which can be combined to a straight line. We start by adding in every iteration step the new position estimate from the odometry data $\boldsymbol{x}$ to our temporary set $S \leftarrow [S, \boldsymbol{x}]$ and then check if

$$L_{\min} > ||DP_{\text{end}} - S_{\text{end}}|| \quad (8)$$

and

$$e_{\max} > \frac{1}{N-2} \sum_{i=2}^{N-1} e_i \quad (9)$$

are both true. If this is the case we add the point $S_{\text{end}-1}$ to the set of DPs and set the temporary set back to $S = [S_{\text{end}}, \boldsymbol{x}]$. Here, e_i defines the shortest distance between the point S_i and the vector given by $S_{\text{end}} - S_1$. Pseudo code for the algorithm can be found in Algorithm (2). The parameters $L_{\min}$ and $e_{\max}$ are problem specific and have to be tuned accordingly.

Third, every time a new DP is accumulated we generate a piecewise orientation function θ_r as presented above using the actual set of DPs, as long as the accumulated length between the DPs is larger then $U_{\min} \times$ Map Circumference. This ensures that the algorithm does not start too early with the comparison. The parameter $U_{\min} \in [0,1]$ represents the minimal required portion of the boundary line to identify unique positions and has to be chosen or trained ac-

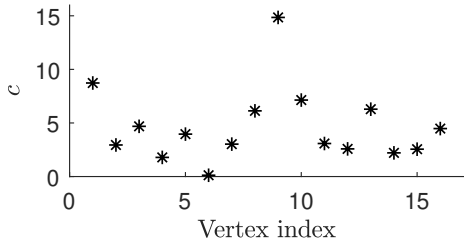

Figure 5: Correlation errors between the actual DP and the vertices of the map are shown. Here the vertex with the index 6 fulfills the required conditions and is chosen for the first position estimate.

cording to the given map. We then compare θ_r with θ_b for all vertices $\boldsymbol{x}_i, i = 1,2,\ldots,n$ of the boundary line in order to find a suitable vertex at which the robot could be. Therefore, we adjust θ_b such that $\theta_{b,i}(x) = \theta_b(x+l_i) - \phi_i$, $i = 2n, 2n-1, \ldots, n+1$. We now evaluate these functions at N linearly distributed points from $-L$ to 0 which results into vectors $\boldsymbol{\theta}_{b,i}$. Here, L is the path length driven by the robot along the boundary line. We then calculate the correlation error

$$\boldsymbol{c}_i = \frac{1}{N}\sum_{k=1}^{N} |\boldsymbol{\theta}_{b,i,k} - \boldsymbol{\theta}_r|, \tag{10}$$

where $\boldsymbol{\theta}_r$ is the evaluation of the function $\hat{\theta}_r(x) = \theta_r(x+L) - \theta_r(L)$ in the range from $-L$ to 0 at N linearly distributed points. We now have correlation errors between the actual driven path by the robot along the boundary line and every vertex of the polygon map. In Figure 5 correlation errors are graphically displayed.

Fourth, we search for the vertex with the minimal correlation error and check if the error is below a certain threshold $c_{\min}$. The parameter $c_{\min}$ has to be trained accordingly to the given map and the given robotic system. If we do not find a convenient vertex, we simply drive further along the wall until finding a vertex which follows the condition above. Thereby, we limit the stored path from the robot to the length of the circumference of the map. If a convenient vertex is found, we call this vertex from now on matching vertex, the first guess of the robots position can be determined as $\boldsymbol{x}_{est}$, where $\boldsymbol{x}_{est}$ is the matching vertex. Moreover, we can also define a orientation estimate as

$$\varphi_{est} = \text{atan2}(\boldsymbol{v}_{est,y}, \boldsymbol{v}_{est,x}), \tag{11}$$

where $\boldsymbol{v}_{est} = \boldsymbol{x}_{est} - \boldsymbol{x}_{est-1}$.

2.3 Systematic Search

Using the land navigation from above we now have a first estimate on where the robot is located. Based on this estimate we are able to start a systematic search in this region to determine a more accurate and certain pose estimate. Therefore, we use a particle filter since other localization techniques, such as Kalman Filter, can not handle binary measurement data. The general idea of the particle filter (Thrun et al., 2005) is to represent the probability distribution of the posterior by a set of samples, called particles, instead of using a parametric form as the Kalman Filter does. Here, each particle

$$\mathcal{X}_t = \boldsymbol{x}_t^{[1]}, \boldsymbol{x}_t^{[2]}, \ldots, \boldsymbol{x}_t^{[N]} \tag{12}$$

represents a concrete instantiation of the state at time t, where N denotes the number of particles used. The belief $bel(\boldsymbol{x}_t)$ is then approximated by the set of particles $\mathcal{X}_t$. The Bayes filter posterior is used to include the likelihood of a state hypothesis $\boldsymbol{x}_t$

$$\boldsymbol{x}_t^{[i]} \sim p(\boldsymbol{x}_t | \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}). \tag{13}$$

Here $\boldsymbol{z}_{1:t}$ and $\boldsymbol{u}_{1:t}$ represent the measurement history and the input signal history respectively.
The idea of our method is to reduce the particles required for global localization with a particle filter extensively by just sampling particles in an area around the position estimate retrieved by land navigation as shown above. This is nothing else than the transformation from a global localization problem to a local one. Therefore, we simply use a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with the mean $\boldsymbol{\mu} = [\boldsymbol{x}_{est}^T, \varphi_{est}]^T$ and parametrized covariance matrix $\boldsymbol{\Sigma} = \text{diag}[\sigma_x, \sigma_y, \sigma_\varphi]$. The parameters $\sigma_x, \sigma_y, \sigma_\varphi$ can be chosen accordingly to the mean and uncertainty of the error of the initial pose estimate as we will see in Section 3. In Figure 6 an example distribution of the particles around the estimated pose is shown. After sampling the particles we now are able to further improve the localization executing a systematic search along the border line using the particle filter. Thereby, the weighting of the particles happens as follows: If a particle would see the same as the sensor has measured, then we give this particle the weight $w_i = \hat{w}$, where w_i is the weight of the i-th particle. If the particle would not see the same, then $w_i = (1 - \hat{w})$. Here the parameter $\hat{w}$ has to be larger then 0.5.

3 RESULTS

In this section, we evaluate the proposed algorithms in regard to stability and performance. Stability is defined as the fraction of localization experiments were

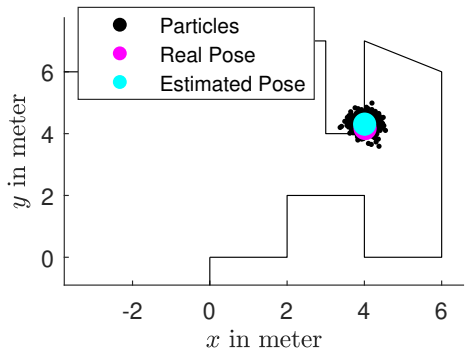

Figure 6: Sampled Particles: Particles have been sampled around the first pose estimate generated using land navigation.

the robot could localize itself from an initially lost pose in the presence of sensor noise. For that the Euclidian distance between the true and the estimated pose had to be below a threshold of 0.3 m. Performance is defined as the mean accuracy of the pose estimation and the time required to generate the pose estimate. Therefore, we created a simulation environment using Matlab with a velocity and an odometry motion model for a differential drive robot as presented in (Thrun et al., 2005). For realistic conditions, we calibrated the noise parameters in experiments to match the true motion model of a real Viking MI 422P, a purchasable autonomous lawn mower. For this calibration, we tracked the lawn mower movements using a visual tracking system (OptiTrack) and computed the transition model parameters through maximum likelihood estimation. The parameters can be found in Table 1. We used a sampling frequency of $f = 20\,Hz$ for the simulated system and the maximum linear velocity $v_0 = 0.3\,\mathrm{m\,s^{-1}}$ and angular velocity $\omega = 0.6\,\mathrm{s^{-1}}$. These values correspond approximately to the standard for purchasable low-cost robots. We set the number of $K = 100$ which leads to a period time of the wiggly lines $T = 5\,\mathrm{s}$. This is a trade-off between the enlargement of the scanning surface and the movement that can be performed on a real robot. The relative position of the binary sensor in regard to the robot frame, Equation (1), is given by $[\Delta x, \Delta y]^T = [0.3\,\mathrm{m}, 0.0\,\mathrm{m}]^T$.

3.1 Wall Follower

As mentioned before, only at the boundary we are able to generate data useful for localization since only there we can detect differences in the signal generated by the given binary sensor. Hence, we first evaluate the performance of the presented wall following algorithm. Therefore, we use the mean squared error

Table 1: Measured parameters for the velocity and odometry motion model from (Thrun et al., 2005).

	Vel. Motion Model	Odom. Motion Model
α_1	0.0346	0.0849
α_2	0.0316	0.0412
α_3	0.0755	0.0316
α_4	0.0566	0.0173
α_5	0.0592	-
α_6	0.0678	-

(MSE) to measure the deviation of the path of the sensor to the given boundary line of the map. An example of such a path along the map is given in Figure 7. Let $\boldsymbol{x}_i$ be the i-th position of the sensor and $\boldsymbol{s}_i$ the closest point at the boundary line to $\boldsymbol{x}_i$, then

$$MSE = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{s}_i - \boldsymbol{x}_i||^2, \tag{14}$$

where N represents the number of time steps required to complete one round along the boundary line. We also evaluate the mean velocity of the robot driving along the boundary

$$v_\mu = \frac{U}{T}, \tag{15}$$

where U is the circumference of the map and T the time required by the robot to complete one round. In Figure 8, Figure 9 and Figure 10, the average simulation results over 10 simulations for different noise factors in Equation (2) and Equation (4) are presented. The used map is shown in Figure 7. The noise factor hereby determines the randomness of the binary signals, thus a factor of 0 leads to always accurate measurements whereas a factor of 1 implies total random measurement results. Hence, a noise factor of 40 % signals that 40 % of the output signals

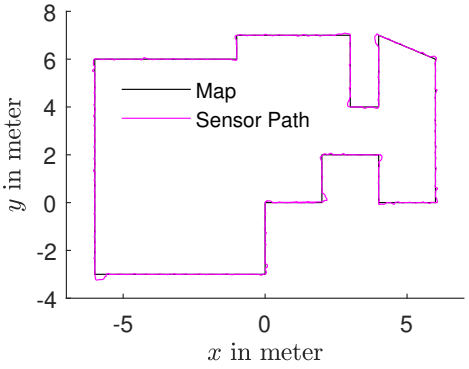

Figure 7: Example of a path along the boundary line generated using the presented wall following algorithm with $a_\mu = 0.7$, $a_v = 0.7$, $M = 100$ and a sensor noise of 10 %.

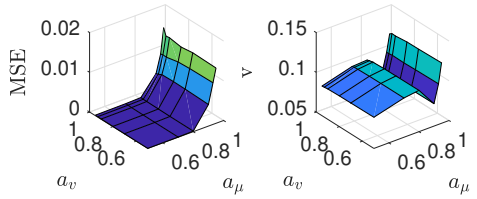

Figure 8: MSE and mean velocity for a noise factor of 0 % and varying parameters a_μ and a_v. The results shown are the average values over 10 iterations.

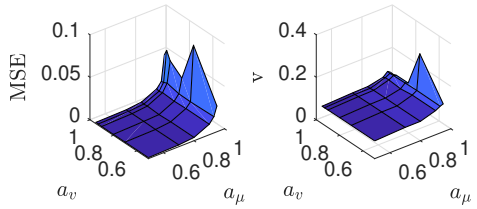

Figure 9: MSE and mean velocity for a noise factor of 20 % and varying parameters a_μ and a_v. The results shown are the average values over 10 iterations.

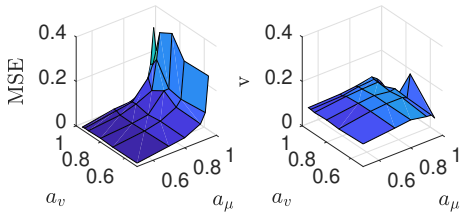

Figure 10: MSE and mean velocity for a noise factor of 40 % and varying parameters a_μ and a_v. The results shown are the average values over 10 iterations.

of the binary sensor are random.

The presented wall follower leads to a stable and accurate wall following behavior as exemplarily shown in Figure 7. With increasing noise factor, the MSE becomes larger. However, the algorithm is stable enough to steer the robot along the boundary line even by noise factors around 40 %. The algorithm seems to work best, thus leading to small MSE, for small a_μ. However, a small value for a_μ leads to large changes in the linear and angular velocities which may not be feasible on a real robot or may cause large odometry erros. We discuss this problem more intensively in Section 4. The parameter a_v seems to have no strong influence neither on the MSE nor on the mean velocity.

3.2 Land Navigation

In order to evaluate the performance and stability of the proposed shape comparison method we simulated

Table 2: Trained parameters for the presented maps from Figure 11.

	L_{min}	e_{max}	c_{min}	U_{min}
Map 1	0.5	0.01	0.2	0.5
Map 2	0.5	0.01	0.3	0.4

the differential drive robot 100 times in two different maps (Figure 11) with randomly chosen starting positions. We trained the parameters specifically for the given maps under usage of the proposed wall following algorithm with $a_\mu = 0.7$, $a_v = 0.7$, $M = 100$ and a simulated sensor noise of 10 %. The resulting parameters are presented in Table 2. We then calculated the difference of the estimated position and the true position $\Delta x = ||\boldsymbol{x}_{\text{true}} - \boldsymbol{x}_{\text{est}}||$ as well as for the estimated orientation and true orientation $\Delta\varphi = ||\varphi_{\text{true}} - \varphi_{\text{est}}||$. Histograms of these errors are depicted in Figure 12 and Figure 13. The mean time for finding this first pose estimate is around $T_1 = 336\,\text{s}$ for map 1 and $T_2 = 382\,\text{s}$ for map 2. The results show that the proposed algorithm is able to compute accurate estimates of the robots poses. Those estimates can then be further used to sample particles for a particle filter to systematically search in the region of interest for the correct pose.

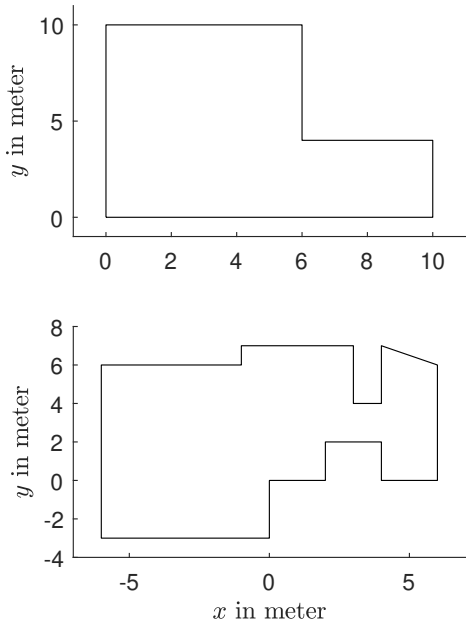

Figure 11: Maps used for the simulation. Map 1 is at the top and map 2 at the bottom. The circumferences for the maps are $U_1 = 40\,\text{m}$ and $U_2 = 53.23\,\text{m}$.

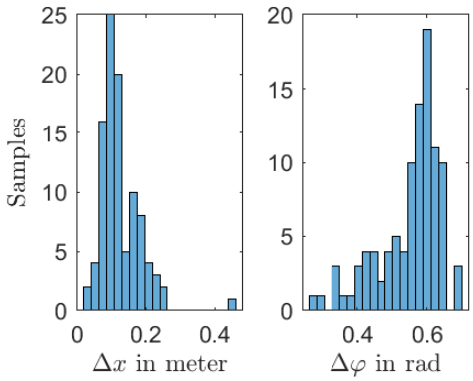

Figure 12: Evaluation results for map 1: Histogram of the difference between position and orientation estimate and true position and orientation after matching through shape comparison with $\mu_{\Delta x} = 0.13$, $\mu_{\Delta\varphi} = 0.55$, $\sigma_{\Delta x} = 0.06$, $\sigma_{\Delta\varphi} = 0.09$.

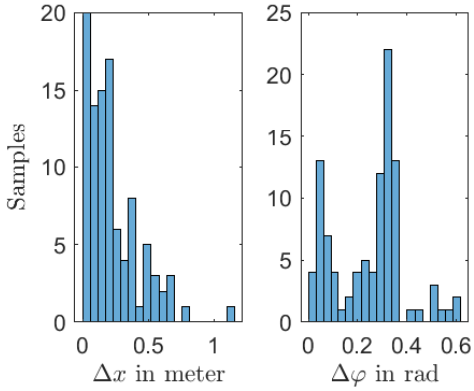

Figure 13: Evaluation results for map 2: Histogram of the difference between position and orientation estimate and true position and orientation after matching through shape comparison with $\mu_{\Delta x} = 0.23$, $\mu_{\Delta\varphi} = 0.25$, $\sigma_{\Delta x} = 0.20$, $\sigma_{\Delta\varphi} = 0.15$.

3.3 Systematic Search

We now demonstrate how the position estimate from above can be further used to systematically search for a better estimate and for executing planning tasks. As proposed, we first sample particles around the position estimate using Gaussian distributions

$$\begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} \sim \begin{bmatrix} \mathcal{N}(x_{\text{est}}, \mu_{\Delta x} + 3\sigma_{\Delta x}) \\ \mathcal{N}(y_{\text{est}}, \mu_{\Delta x} + 3\sigma_{\Delta x}) \\ \mathcal{N}(\varphi_{\text{est}}, \mu_{\Delta\varphi} + 3\sigma_{\Delta\varphi}) \end{bmatrix}. \tag{16}$$

The standard deviation $\mu + 3\sigma$, depending on the measurement results from Section 3.2, ensures that the particles are sampled in an area large enough around the initial pose estimate. In Figure 15 and Figure 16, the particles and the pose estimate are shown as well as the true position of the robot. In the following step we systematically search for the true pose of the robot by following the boundary line. The particle filter algorithm weights the particles and does a resampling if required. We stop the wall following if we are certain enough about the pose of the robot, hence the variance of the particles is less then a certain treshold. In Figure 17 the robot after the systematic search for a better pose estimate is shown. Now we are able to execute certain tasks with the robot, for example mowing the lawn. Thereby, the robot has to move away from the boundary and the pose estimate becomes more uncertain as shown in Figure 18. After reaching again the boundary line the particle filter is able to enhance the pose estimate based on the new information as depicted in Figure 19.

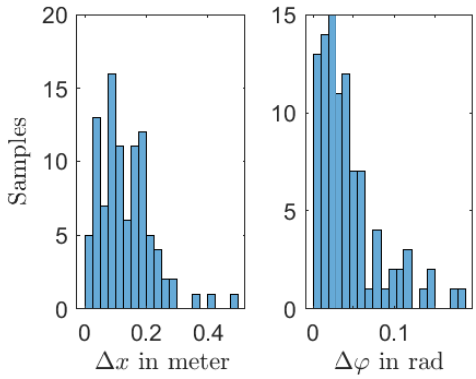

Figure 14: Histogram of the difference between position and orientation estimate and true position and orientation after the systematic search using the particle filter with $\mu_{\Delta x} = 0.13$, $\mu_{\Delta\varphi} = 0.04$, $\sigma_{\Delta x} = 0.086$, $\sigma_{\Delta\varphi} = 0.04$.

We also evaluated the performance of the proposed systematic search. Therefore, we simulated the differential drive robot 100 times sampling the particles as proposed in Equation (16). We then followed the boundary line until the standard deviation of the orientation has been below 0.2. Again, we evaluated the difference between position and orientation estimate and true position and orientation. Here, 3 times the particle filter has not found a sufficient accurate pose estimate. If this happens, we simply can restart the procedure by finding a pose estimate using land navigation. In Figure 14 a histogram depicting the results are shown. The systematic search for the pose of the robot using a particle filter improves the orientation estimate compared to the initial estimate presented in section 3.2.

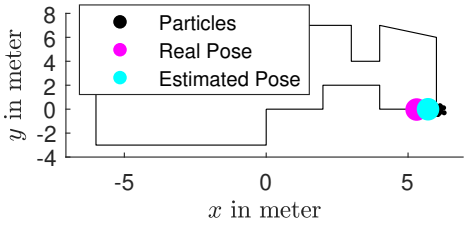

Figure 15: Randomly sampled particles around the position estimate $\boldsymbol{x}_{\text{est}}$.

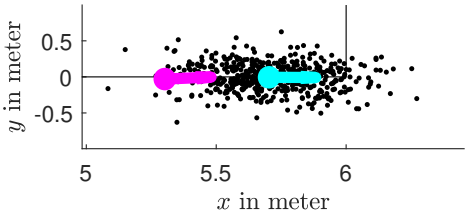

Figure 16: Randomly sampled particles around the position estimate $\boldsymbol{x}_{\text{est}}$ in close up view.

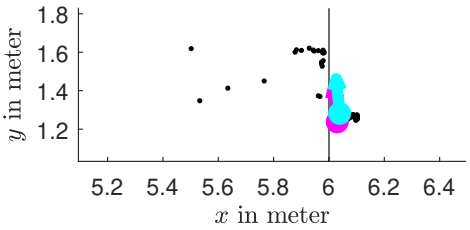

Figure 17: Pose estimate of the robot after the systematic search.

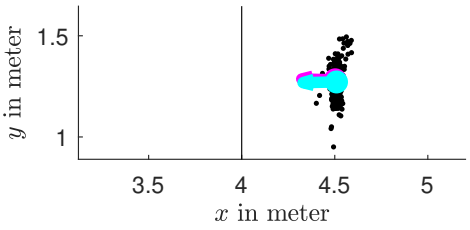

Figure 18: Moving within the map increases the uncertainty of the pose estimate.

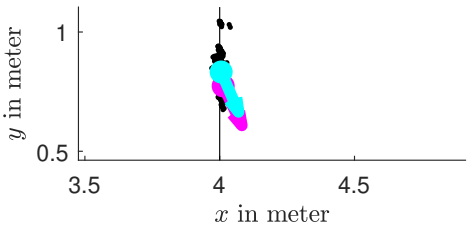

Figure 19: New information collected at the boundary leads to an improvement in pose estimation.

4 CONCLUSION

We have presented a method for global localization based only on odometry data and one binary sensor. This method, inspired by insect localization strategies, is essential for efficient path planning methods as presented in (Hess et al., 2014) or (Galceran and Carreras, 2013) which require an accurate estimate of the robots pose. Our approach combines ideas of path integration, land navigation and systematic search to find a pose estimate for the robot.

Our method relies on a stable and accurate wall following algorithm. We showed that the proposed algorithm is robust even under 40 % sensor noise and, if convenient parameters are chosen, leads to an accurate wall following behavior.

The proposed approach for finding a first pose estimate is robust and accurate. Given this pose estimate we are able to sample particles around this position for starting a systematic search with a particle filter algorithm. Therefore, we maintain the wall following behavior until the uncertainty of the pose estimate of the particle filter is below a certain threshold. The proposed search method improves the orientation estimate intensively while maintaining the accuracy of the position estimate. As mentioned in the Introduction, it is not possible to compare our approach directly with other approaches since different types of sensors are used. However, comparing our localization results with the results presented in (Stavrou and Panayiotou, 2012) for a particle filter with a single low-range sensor, we reach a similar accuracy for the position estimate by only using a binary sensor. Thereby, the accuracy is in the range of around 0.1 m.

Using the generated pose estimate we are now able to execute given tasks by constantly relocating the robot at the boundary lines. For example, using bioinspired neural networks for complete coverage path planning (Zhang et al., 2017). Open research questions in regard to the proposed approach are which methods can be used to learn the algorithm parameters L_{min}, e_{max}, c_{min}, U_{min} on the fly. Also, the algorithm has to be evaluated on a real robotic system in a realistic garden setup.

REFERENCES

Bernini, F. (2009). Lawn-mower with sensor. US Patent 7,613,552.

Bobzin, C. (2013). Cataglyphis nodus. CC BY-SA 3.0, https://de.wikipedia.org/wiki/Cataglyphis.

Carr, M. F., Jadhav, S. P., and Frank, L. M. (2011). Hippocampal replay in the awake state: a potential substrate for memory consolidation and retrieval. *Nature neuroscience*, 14(2):147.

Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999). Monte carlo localization for mobile robots. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1322–1328. IEEE.

Erdem, U. M. and Hasselmo, M. (2012). A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience*, 35(6):916–931.

Erickson, L. H., Knuth, J., O'Kane, J. M., and LaValle, S. M. (2008). Probabilistic localization with a blind robot. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1821–1827. IEEE.

Feng, L., Bi, S., Dong, M., Hong, F., Liang, Y., Lin, Q., and Liu, Y. (2017). A global localization system for mobile robot using lidar sensor. In *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 478–483. IEEE.

Foster, D. J. and Wilson, M. A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680.

Galceran, E. and Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, 61(12):1258–1276.

Hess, J., Beinhofer, M., and Burgard, W. (2014). A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5600–5605. IEEE.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Ito, S., Endres, F., Kuderer, M., Tipaldi, G. D., Stachniss, C., and Burgard, W. (2014). W-rgb-d: floor-plan-based indoor global localization using a depth camera and wifi. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 417–422. IEEE.

Johnson, A. and Redish, A. D. (2007). Neural ensembles in ca3 transiently encode paths forward of the animal at a decision point. *Journal of Neuroscience*, 27(45):12176–12189.

Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., and Wehner, R. (2000). A mobile robot employing insect strategies for navigation. *Robotics and Autonomous systems*, 30(1-2):39–64.

Lee, H. and Jung, S. (2012). Balancing and navigation control of a mobile inverted pendulum robot using sensor fusion of low cost sensors. *Mechatronics*, 22(1):95–105.

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., and Moser, M.-B. (2006). Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience*, 7(8):663.

Miller, D. P. and Slack, M. G. (1995). Design and testing of a low-cost robotic wheelchair prototype. *Autonomous robots*, 2(1):77–88.

Nebot, E. and Durrant-Whyte, H. (1999). Initial calibration and alignment of low-cost inertial navigation units for land vehicle applications. *Journal of Robotic Systems*, 16(2):81–92.

O'Kane, J. M. and LaValle, S. M. (2007). Localization with limited sensing. *IEEE Transactions on Robotics*, 23(4):704–716.

O'keefe, J. and Nadel, L. (1978). *The hippocampus as a cognitive map*. Oxford: Clarendon Press.

Pfeiffer, B. E. and Foster, D. J. (2013). Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497(7447):74.

Redish, A. D. et al. (1999). *Beyond the cognitive map: from place cells to episodic memory*. MIT press.

Rueckert, E., Kappel, D., Tanneberg, D., Pecevski, D., and Peters, J. (2016). Recurrent spiking networks solve planning tasks. *Nature Publishing Group: Scientific Reports*, 6(21142).

Samsonovich, A. and McNaughton, B. L. (1997). Path integration and cognitive mapping in a continuous attractor neural network model. *Journal of Neuroscience*, 17(15):5900–5920.

Stavrou, D. and Panayiotou, C. (2012). Localization of a simple robot with low computational-power using a single short range sensor. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 729–734. IEEE.

Su, Z., Zhou, X., Cheng, T., Zhang, H., Xu, B., and Chen, W. (2017). Global localization of a mobile robot using lidar and visual features. In *Robotics and Biomimetics (ROBIO), 2017 IEEE International Conference on*, pages 2377–2383. IEEE.

Tanneberg, D., Peters, J., and Rueckert, E. (2019). Intrinsic motivation and mental replay enable efficient online adaptation in stochastic recurrent networks. *Neural Networks - Elsevier*, 109:67–80. Impact Factor of 7.197 (2017).

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Tribelhorn, B. and Dodds, Z. (2007). Evaluating the roomba: A low-cost, ubiquitous platform for robotics research and education. In *ICRA*, pages 1393–1399.

Wehner, R. (1987). Spatial organization of foraging behavior in individually searching desert ants, cataglyphis (sahara desert) and ocymyrmex (namib desert). In *From individual to collective behavior in social insects: les Treilles Workshop/edited by Jacques M. Pasteels, Jean-Louis Deneubourg*. Basel: Birkhauser, 1987.

Zhang, J., Lv, H., He, D., Huang, L., Dai, Y., and Zhang, Z. (2017). Discrete bioinspired neural network for complete coverage path planning. *International Journal of Robotics and Automation*, 32(2).