Map-based Motion Generation for Localization with Particle Filters*

Svetlana Sodol[1]

Abstract— **This work attempts to solve the kidnapped robot problem with an available map by using a particle filter. The major contribution of the work is the comparison between four approaches to generating motion for the filter. The approaches include random motion, pre-computed best action, online maximization of best action for sensor reading change and combined pre-computed and online optimized trajectory formation. The proposed approaches utilize the information provided in the 2D map of the grid world and are evaluated on speed and accuracy of their ability to disambiguate the belief in location between the particles. Results indicate that the pre-computed best action approach is the most accurate, albeit not being an improvement in time over the random motion approach. All other approaches are significantly slower as indicated by the results of the t-tests performed at the 99% confidence level. Further study is warranted with more trials, other parameter settings for sensing and motion noise models and larger maps.**

I. INTRODUCTION

Localization is a complex task, especially in the presence of ambiguous environmental landmarks, even for humans with an available map. Dudek et al. have shown that finding an optimal trajectory that will allow for localization with multi-modal is NP-hard, even in deterministic noise-free settings [1]. A possibility to overcome this problem is to use particle filtering, often used in robotics for localization, as it is able to solve problems that were previously untractable [2] and can be more general than other approaches as it can be applied to non-Gaussian settings [3].

The classical particle filter utilizes random motions to move the agent around in the world to gain new sensor readings. However, in the presence of the map, there are more resources to aid the robot in its localization – specifically the ability to pre-compute sensory readings for possible and future locations, calculate trajectories for motion or lead the agent towards areas of the map where localization is easier. Roy et al. show that navigating in areas with high information gain results in less likelihood of getting lost [4].

Our work proposes to compare three approaches of map-based motion generation in combination with a particle filter for propagating multi-modal location belief against the random motion model. The proposed setting for simulations is a toy 2D grid world with obstacles that will allow for easy and fast implementations of various approaches. The proposed approaches will include both pre-computed and online optimization components.

Section II describes some of the related works with Section III outlining the details of our proposed solution approaches. In Section IV we present the evaluation procedure and the statistical results, with concluding remarks and outlines for future work in Section V.

II. RELATED WORK

Particle filters have been used for a variety of applications, including localization [5, 6], surveillance applications [7], target tracking [8, 9, 10] and fault detection [3]. Nunez et al. use a particle filter to find best locations for sensor placement to use in visual target search [9]. Calafiore and Fagiano use an online optimization approach with particle filtering for control strategy choosing based on randomized set of disturbance scenarios [11].

Berntrop et al. [6] use particle filtering for autonomous driving with a map, albeit with a known starting location. An interesting aspect of their work is that the driving requirements are formulated ahead of time - like the requirement of staying on the road. Each decision during the online optimization procedure is evaluated on these pre-formulated requirements. Another aspect is that the search for motion trajectories are only done in the areas of the map that are likely to result in good trajectories.

Aragwal et al. [5] present an extended Kalmann filter approach to multi-modal disambiguation of global position. Albeit not using a particle filter, the approach uses an online receding horizon procedure to optimize over trajectories resulting in maximum information gain about the location, based on pre-computed states from the map. The map is used to pre-compute a uniqueness graph of configurations that depend on landmark information from the map. Candidate paths are calculated iteratively online for each belief mode in form of trajectories that lead to best supporting sensor information for the mode. The best trajectory is picked based on maximum expected information gain – confirming a mode or rejecting a large number of incorrect modes. One of our approaches will be most similar to this work, although simplified and applied to a discrete particle filter system.

*This work was done as part of the CPSC 515 Computational Robotics course. The code for this project is available online at https://github.com/Caliatdrimme/ParticleFilterWithMapUsage

[1]Svetlana Sodol is with Department of Computer Science, University of British Columbia `sodols@cs.ubc.ca`

III. SOLUTION

A. Problem Formulation

The problem we are solving is a version of the kidnapped robot problem. A robot is placed in the 2D grid world without any prior knowledge on its location. The robot is provided with a map, and the question we are attempting to answer is how long it takes for the robot to localize itself using a particle filter and how accurately. The goal of the task is formulated in terms of all the particles agreeing on the location of the robot, with the accuracy defined as localizing to the real location, or being close enough with the Manhattan distance of 1 to the real location - which means that the real location is reachable in one more action.

The robot has a map provided with obstacles and valid cells for movement, along with a sensor that can count the number of nearby walls. The sensor has a probability of detecting the correct number of walls to account for the possible noise in the measurement and is used in weighting the particles.

The robot has no other means of localizing itself other than moving around in the 2D grid and using its sensor. There are 4 actions available for the robot: up, down, left and right. There is a probability of the controller executing the action correctly to represent the noise in the motion model. In the case of the action execution going wrong, another action out of the 3 is randomly chosen or the robot remains in place. If the action leads to running into a wall the robot stays in the same grid cell. The motion model is applied in the same way to the real location in the simulation as well as for the particles in the filter.

The simulations are performed in a 2D grid world as shown in Fig.1, with 25 total cells and 6 obstacles. The obstacles are seen as walls from all the cells surrounding it. This leaves 19 valid cells for the robot to occupy.

B. Particle Filter

Each simulation trial starts off with 15 particles – each particle is a randomly generated guess for a location in the map. Only valid cells (19 in total) in the map are possible, however multiple particles can have the same guess. The following algorithm is repeated until we are left with one particle:

1) Get sensor reading (based on real location)
2) Weight each particle as probability of seeing the sensor reading in the particle location
3) Resample particles given the weights
4) Pick action (using one of the motion generation approaches)
5) Update the simulated real and all of the particles' locations
6) Delete particles that are agreeing on location

Grid World Map

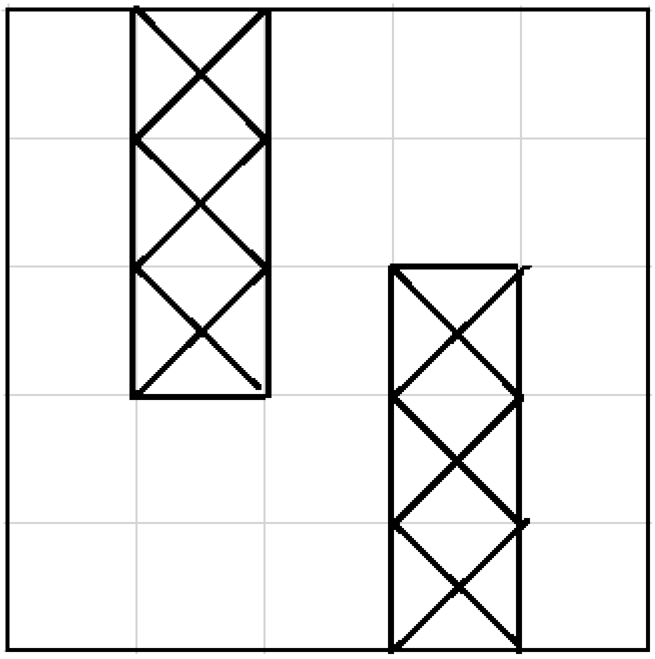

Fig. 1. 2D grid world map with marked obstacle locations

The weight of the particle j at loop $i + 1$ in step 2 is calculated as follows:

$$w_{i+1,j} = \frac{w_{i,j} * p(s)}{\sum_{k=1}^{n} w_{i,k} * p(s)}$$

where n is the total number of current particles and $p(s)$ is the probability that the sensor reading is correct.

Once the action is picked by step 4, the same action is applied to all the locations of the particles in the filter, as well as the simulated real location. The resulting action performed for each however might differ due to the noise factor introduced with the motion model, which is applied independently for each location that needs updating.

C. Motion Generation

This work will be looking at four approaches to motion generation for the particle filter - random motion, using pre-computed best actions, optimizing the action online and a combined approach where the robot optimizes online over pre-computed trajectories. There are eight strategies to test and compare in total, as two of the approaches can utilize a number of different objective functions. Each of the eight approaches will be compared on execution time, number of steps taken and accuracy of the final location guess when ran on the same starting locations.

1) Random Motion: This is a classic approach to motion generation for the particle filter. At each step of the particle filter execution the robot uniformly samples a number between 1 and 4 to pick the action to perform next.

2) Pre-computed Best Action: This is a very simplistic approach to utilizing the map information prior to any action taken in the world. For each open cell in the map, the robot pre-calculates the best action to take if its location was in that cell. The best action is defined as the one that results in the maximum change in the sensor reading after correct execution of the action. Once all of the best actions are calculated, the particle filter algorithm is executed. Step 4 for each loop of the execution picks the most popular action among the best actions for the current locations of the particles in the filter.

3) Online Best Action: This approach is similar to the pre-computed approach, but all of the calculations happen online during step 4 and are thus only ever done for locations that are guesses in the particle filter in the current loop. The action is picked as the one that maximizes the value of the objective function that focuses on change in the sensor reading as a result of execution of the action accounting for the motion model noise. There are two objective functions we are comparing for this approach, resulting in two strategies. The first objective counts the number of particles that will witness a change in the sensor readings as the result of of the action. The second objective is as follows:

$$obj = max \sum_{k=1}^{n} (s(k) - s(a))^2$$

where $s(k)$ is the sensor measurement of current guess of location for particle k and $s(a)$ is the sensor measurement for the resulting location if the action is correctly executed for the particle.

4) Pre-computed Trajectories Optimized Online: The last approach includes both a pre-computation and an online component. The pre-computation finds the best target cell for each open cell in the map. We define a maximum horizon of the trajectory length to a target state, and the target cell is thus a reachable cell at most horizon correctly executed actions away from the original cell, that results in maximum change in the sensor reading.

The online component in step 4 of the particle filter algorithm then picks the best trajectory to follow out of the pre-computed ones for the current particles. The trajectory is then followed until completion, which might take over multiple loops. If the locations have not converged a new trajectory is picked given the current particles.

The trajectory is picked by its corresponding finishing cell, as the one that maximizes one of the four objective functions. The first two are the same as the two objectives for the online planning approach above. The other two are the number of particles that are will be moved closer to their target cells as a result of the correct execution of the trajectory, and the last one is as follows:

$$obj = max \sum_{k=1}^{n} (dist(k,t) - dist(k,a))^2$$

where $dist(k,t)$ is the distance of the current particle location to its pre-computed target, and $dist(k,a)$ is the distance between the current particle location and the resulting location of executing the trajectory.

The distance function $dist(s1, s2)$ is defined as a heuristic Manhattan distance, since the actions are discrete and grid-based. The distance will be an admissible heuristic to the number of actions needed as it does not account for the obstacles present in the map.

D. Hypothesis

We hypothesize that the different approaches to motion generation will result in a statistically significant difference between the times of execution. The main result that we hope to see is that the usage of the map information will be beneficial to the task at hand - better accuracy in resulting localization, faster execution time and less steps taken.

IV. Evaluation and Results

A. Evaluation

The evaluation was performed as running the eight strategies on 10 distinct starting locations in the map out of the 19 valid cells. The trajectory horizon for the last approach was set to 5. The probabilities of the sensor and motion controller correctness were set to 0.9. Each trial of the simulation was performed as if the robot is executing for the first time in the map, with the full reset of all variables and the pre-computed components.

B. Accuracy and Number of Steps Results

The results on accuracy did not show a clear trend, with generally not great results - there were more failed trials overall. The pre-computed approach showed the best accuracy performance with only 1 failed trial and 4 successful trials. The random motion approach was close second, with all the other approaches resulting in worse performance.

The number of steps taken varies greatly from 40 to over a 1400 steps. The last approach with the objective that

TABLE I

P VALUES OF THE T-TEST BETWEEN ALL PAIRS OF MOTION GENERATION APPROACHES (WITH THE PRE-COMPUTATION TIME INCLUDED)

	0.76	0.0051	0.000984	$9.83e-11$	$3.7e-13$	$2.21e-14$	$1.23e-10$
0.76		0.00607	0.00113	$9.6e-11$	$3.2e-13$	$1.66e-14$	$1.23e-10$
0.0051	0.00607		0.252	$2.81e-08$	$6.49e-10$	$2.9e-10$	$9.4e-09$
0.000984	0.00113	0.252		$8.05e-07$	$4.02e-08$	$3.48e-08$	$1.39e-07$
$9.83e-11$	$9.6e-11$	$2.81e-08$	$8.05e-07$		0.598	0.939	0.0903
$3.7e-13$	$3.2e-13$	$6.49e-10$	$4.02e-08$	0.598		0.429	0.147
$2.21e-14$	$1.66e-14$	$2.9e-10$	$3.48e-08$	0.939	0.429		0.0452
$1.23e-10$	$1.23e-10$	$9.4e-09$	$1.39e-07$	0.0903	0.147	0.0452	

TABLE II

P VALUES OF THE T-TEST BETWEEN ALL PAIRS OF MOTION GENERATION APPROACHES (WITH THE PRE-COMPUTATION TIME EXCLUDED)

	0.811	0.0051	0.000984	0.0478	0.00183	0.00295	0.00149
0.811		0.00569	0.00108	0.0531	0.00199	0.00313	0.00162
0.0051	0.00569		0.252	0.959	0.999	0.471	0.0981
0.000984	0.00108	0.252		0.319	0.229	0.0686	0.458
0.0478	0.0531	0.959	0.319		0.956	0.639	0.132
0.00183	0.00199	0.999	0.229	0.956		0.424	0.0889
0.00295	0.00313	0.471	0.0686	0.639	0.424		0.0322
0.00149	0.00162	0.0981	0.458	0.132	0.0889	0.0322	

maximizes the sum of movement towards targets was the worst performing on average on the number of steps.

The summary of the results on accuracy and number of steps can be seen below in Table 3. The approaches are in the following order: random, pre-computed, online with objective 1, online with objective 2, combined with objective 1 through combined with objective 4. The pass and fail indicate the number of trials out of 10 that were either successfully localized for pass, or localized with a distance of larger than 1 for fail. The trial by trial results tables are included in the Appendix.

TABLE III

SUMMARY OF STEPS TAKEN AND ACCURACY OF TRIALS

max of steps	min of steps	average of steps	pass	fail
337	40	123	5	3
345	52	130	4	1
480	53	173	1	8
699	81	270	2	5
1472	40	380	2	4
641	67	363	3	5
501	107	279	0	7
1431	60	696	3	6

C. Time of Execution Results

A statistical t-test was performed to evaluate the time of execution results and all of the values are presented in Table 1. Usage of different objectives did not result in a statistically significant difference in the time of execution. The random motion and the pre-computed best action approaches were also not statistically significantly different.

The last two approaches were significantly slower than both the random and the pre-computed approaches, with the combined approach also significantly slower than the online approach. All of these differences are statistically significant at the 99% confidence level with p-values less than 0.01.

The trial by trial timing results are included in the Appendix.

Based on the same trials, the t-tests were also performed on timing of execution which did not account for the pre-computed components timing. This compares the execution time with no restarts between the trials. It is a different take on the timing of the approaches, as it also allows us to judge the timing of the approaches utilizing pre-computation as intended - doing it once for the map and if multiple restarts are necessary, reusing the stored information. The statistical results are presented in Table 2 with the trial by trial timing information included in the Appendix.

When not including the pre-computation time, usage of different objectives again was not found significantly different in timing. Moreover, the difference between the online and the combined approach was also not found to be statistically significant. The random motion approach and the pre-computed approach still showed no significant difference in timing between themselves, but differing with all other approaches at the 99% confidence level.

An interesting change was that the last approach with the objective that focuses on the sum of sensor reading changes is now not significantly different from the timing of the random motion and the pre-computed approaches.

V. CONCLUSIONS

Usage of the different objective functions did not result in a significant difference in the timing of the approaches.

The pre-computed approach did not provide a benefit in the timing over the random motion approach, albeit being a bit better in the accuracy. This finding somewhat agrees with our expectations, and the pre-computed approach is promising for further research, as being no more slower and yet having better accuracy is a benefit to the solution.

The online and the combined approaches were found to be significantly slower in general than the random motion and the pre-computed approaches, even when not including the time of the pre-computation. The findings between the approaches go against our predictions somewhat, however, the slower execution time could be contributed to the complexity of the calculations performed.

Poor performance can also be attributed to the map set up used, as there is not a lot of variability in the sensor readings for different locations in the map. Using a larger map or a different sensor model (e.g. colouring walls and detecting colours instead of simply number of them) could result in easier solutions. Setting the horizon and the noise parameters were also done arbitrarily, as well as the starting number of particles. This is a big limitation of the current study, along with the limited number of trial runs.

Besides addressing the limitations, further work could include comparing other approaches to motion generation and improving on the implementations of the approaches presented here. Another aspect could be working on more realistic problems with larger maps, 3D space, and more complex motion and sensor models.

.

APPENDIX

Tables IV through IX hold the detailed results of the simulations. The rows are the motion generation approaches in the following order: random, pre-computed, online with objective 1, online with objective 2, combined with objective 1 through combined with objective 4.

TABLE IV
SUMMARY STATISTICS ON THE TIME OF EXECUTION RESULTS (WITH THE PRE-COMPUTATION INCLUDED)

max	min	average
0.050	0.006	0.018
0.046	0.008	0.020
0.152	0.019	0.058
0.179	0.030	0.081
0.388	0.201	0.249
0.347	0.210	0.260
0.291	0.205	0.247
0.397	0.223	0.297

TABLE V
SUMMARY STATISTICS ON THE TIME OF EXECUTION RESULTS (WITH THE PRE-COMPUTATION EXCLUDED)

max	min	average
0.050	0.006	0.018
0.046	0.008	0.020
0.152	0.019	0.058
0.179	0.030	0.081
0.204	0.008	0.057
0.112	0.014	0.058
0.080	0.020	0.048
0.210	0.012	0.101

TABLE VI
TRIAL BY TRIAL RESULTS ON ACCURACY - 1 MEANS PASS, 0 FAIL, 2 MEANS THE FINAL LOCATION WAS AT A DISTANCE OF ONE CELL TO THE REAL LOCATION

0	1	2	0	2	1	1	1	1	0
1	2	0	1	2	1	2	1	2	2
0	0	0	0	0	1	2	0	0	0
2	0	1	0	0	0	0	2	1	2
1	0	2	2	2	1	0	0	2	0
0	1	1	0	2	1	2	0	0	0
0	0	2	0	0	2	0	2	0	0
0	2	0	1	1	0	0	1	0	0

TABLE VII

TRIAL BY TRIAL TIME OF EXECUTION RESULTS (WITH THE PRE-COMPUTATION INCLUDED)

0.050	0.008	0.021	0.028	0.006	0.014	0.009	0.014	0.017	0.017
0.016	0.025	0.010	0.020	0.046	0.020	0.008	0.019	0.013	0.022
0.067	0.062	0.029	0.044	0.061	0.069	0.019	0.152	0.037	0.039
0.030	0.050	0.130	0.116	0.058	0.179	0.105	0.052	0.041	0.048
0.255	0.213	0.271	0.228	0.201	0.223	0.388	0.260	0.225	0.225
0.272	0.254	0.347	0.244	0.231	0.210	0.279	0.292	0.226	0.247
0.281	0.291	0.261	0.205	0.267	0.227	0.220	0.213	0.236	0.273
0.242	0.223	0.226	0.363	0.357	0.342	0.397	0.229	0.271	0.321

TABLE VIII

TRIAL BY TRIAL TIME OF EXECUTION RESULTS (WITH THE PRE-COMPUTATION EXCLUDED)

0.050	0.008	0.021	0.028	0.006	0.014	0.009	0.014	0.017	0.017
0.016	0.024	0.010	0.020	0.046	0.019	0.008	0.018	0.013	0.022
0.067	0.062	0.029	0.044	0.061	0.069	0.019	0.152	0.037	0.039
0.030	0.050	0.130	0.116	0.058	0.179	0.105	0.052	0.041	0.048
0.062	0.015	0.069	0.042	0.008	0.035	0.204	0.070	0.034	0.029
0.069	0.044	0.112	0.045	0.032	0.014	0.087	0.098	0.035	0.042
0.076	0.080	0.059	0.020	0.074	0.032	0.021	0.023	0.042	0.050
0.040	0.012	0.037	0.165	0.164	0.141	0.210	0.043	0.069	0.130

TABLE IX

TRIAL BY TRIAL RESULTS ON NUMBER OF STEPS TAKEN

337	45	133	171	40	103	56	102	127	124
108	111	69	126	345	138	52	107	87	159
232	170	74	133	134	200	53	480	126	134
81	145	357	381	185	699	423	159	106	165
417	69	442	260	40	201	1472	506	209	186
468	260	628	289	213	67	604	641	234	232
501	452	373	107	446	193	123	120	235	240
230	60	235	1152	1176	1018	1431	272	453	939

REFERENCES

[1] Gregory Dudek, Kathleen Romanik, and Sue Whitesides. Localizing a robot with minimum travel. SIAM Journal on Computing, 27(2):583–604 (1998). doi: 10.1137/S0097539794279201

[2] Dadkhah, N., Mettler, B. Survey of Motion Planning Literature in the Presence of Uncertainty: Considerations for UAV Guidance. J Intell Robot Syst, 65: 233–246 (2012). https://doi-org.ezproxy.library.ubc.ca/10.1007/s10846-011-9642-9

[3] S. Yin and X. Zhu. Intelligent Particle Filter and Its Application to Fault Detection of Nonlinear System. IEEE Transactions on Industrial Electronics, 62(6): 3852-3861 (2015). doi: 10.1109/TIE.2015.2399396

[4] N. Roy, W. Burgard, D. Fox and S. Thrun, Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), 1: 35-40 (1999). doi: 10.1109/ROBOT.1999.769927

[5] Agarwal, Saurav, Amirhossein Tamjidi, and Suman Chakravorty. Motion Planning for Global Localization in Non-Gaussian Belief Spaces. Algorithmic Foundations of Robotics XII Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics, (2020). doi:10.1007/978-3-030-43089-419

[6] K. Berntorp, T. Hoang and S. Di Cairano. Motion Planning of Autonomous Road Vehicles by Particle Filtering. IEEE Transactions on Intelligent Vehicles, 4(2): 197-210 (2019). doi: 10.1109/TIV.2019.2904394

[7] Ángel F. García-Fernández. A track-before-detect labelled multi-Bernoulli particle filter with label switching. IEEE Transactions on Aerospace and Electronic Systems, 52(5): 2123-2138 (2016). doi: 10.1109/TAES.2016.150343

[8] Allison Ryan, J. Karl Hedrick. Particle filter based information-theoretic active sensing. Robotics and Autonomous Systems, 58(5): 574-584 (2010). https://doi.org/10.1016/j.robot.2010.01.001

[9] Jesse A. Nunez, Dashi I. Singham and Michael P. Atkinson. Particle filter approach to estimating target location using Brownian bridges. Journal of the Operational Research Society, 71(4): 589-605 (2020). DOI: 10.1080/01605682.2019.1570806

[10] D. Y. Kim, E. Yang, M. Jeon and V. Shin. Receding Horizon Estimation for Hybrid Particle Filters and Application for Robust Visual Tracking. 2010 20th International Conference on Pattern Recognition, pp. 3508-3512 (2010). doi: 10.1109/ICPR.2010.856

[11] Calafiore, Giuseppe C. and Fagiano, Lorenzo. Robust Model Predictive Control via Scenario Optimization. IEEE Transactions on Automatic Control, 58(1): 219–224 (2013). http://dx.doi.org/10.1109/TAC.2012.2203054