Energy-Aware Mobile Robot Exploration with Adaptive Decision Thresholds

Micha Rappaport, Institute of Networked and Embedded Systems, University of Klagenfurt, Austria

Abstract

This paper presents an approach for energy efficient path planning during autonomous mobile robot exploration. The idea is for a robot to efficiently explore the environment and periodically return to the starting point of the exploration for recharging the battery. The focus is to reduce the overall travel path as locomotion is the largest energy consumer. Simulations show that the path can be reduced by more than 25 %. A proof of concept is developed to demonstrate the feasibility on hardware.

1 Introduction

Mobile robots can be employed for many scenarios. These are reconnaissance, coverage, search and rescue, and planetary missions, to name a few. In most cases, robots have to explore and map the previously unknown environment before continuing their mission. This has to be done autonomously due to either physical distance, life threatening environments, or a large number of robots.

For fully autonomous exploration, a robot needs to plan the path through the environment based on the current energy level as well as interrupt exploration for recharging the batteries. The path planning is done online and behavior based since no optimal plan can be precomputed for an unknown environment. The contribution of this work is an energy-aware exploration (EA) strategy that takes into account efficient path planning based on the current battery level as well as timely recharging of batteries for larger maps.

The path planning uses frontier based exploration [1], where a robot identifies points in a two-dimensional (2D) environment that separate known and unknown space. These points are called frontiers. Selecting the frontiers in the right order is a crucial process as it defines the path on which a robot travels during exploration. This path should be as short as possible [2]. Redundancies, where the robot visits certain points in the environment more than once, should be avoided. We derive a general strategy that reduces the redundancy of the traveling path considerably.

We use an *adaptive threshold* for the robot to decide when to return to the docking station for recharging. This threshold depends on the movement of the robot, its power consumption, and the current state of the environment. A similar approach is proposed in [3] where it is shown that a static threshold is not flexible and does not make the best use of the robot's remaining energy.

We show by simulations that our strategy helps to reduce redundancy in traveling and increases the efficiency of exploring unknown environments. Additionally, a proof of concept is developed to show the feasibility on hardware.

The rest of the paper is organized as follows. Section 2 discusses the related work, Section 3 gives a description of the model, exploration strategy, and the implementation, Section 4 describes the simulation setup as well as the results, and Section 5 concludes the paper.

2 Related Work

2.1 Energy-Awareness and Recharging

Different aspects of EA have already been covered. Most importantly, when the battery state of charge (SOC) becomes low, a robot has to interrupt the current task and recharge its battery, e.g., at a docking station. One approach is to simply monitor the battery voltage and initiate a recharging procedure once it drops below a static threshold [4]. More sophisticated approaches estimate the SOC more accurately [5] while others consider additional parameters to more precisely decide on the right moment for recharging [3, 6]. In [3] the expected gain by an operation is calculated and related to the risk of running out of power. This approach is more flexible and efficient, compared to a static threshold approach. In [6] static and adaptive threshold approaches are compared to a bio-inspired rate-maximizing foraging approach which comes close to optimal. In this work we use an adaptive threshold approach.

2.2 Offline Path Planning

Other work focuses on computing optimal paths with respect to energy consumption. In [7] a method is described where mobile robots perform predefined tasks at specific locations in the environment. The robots have to complete all tasks and in between recharge their battery at a docking station to stay alive. The path planning algorithm decides

ISBN 978-3-8007-4231-8

on the best order in which to complete the tasks and the time at which to return for recharging. The authors of [8] also present a method that plans paths to specific points of interest (POIs) in the environment to gather information. The focus is not recharging though but the communication connectivity to a base station. The order in which to visit the POIs is optimized with regard to energy consumption of a mobile robot. When all information has been collected the robot has to come close enough to the base station to transmit this information. Both approaches try to reduce the energy consumption of robots by deciding on the order in which to visit the locations in the environment. The major difference to our work is that they assume an already known environment and do an offline optimization of the path planning.

2.3 Energy Saving

Besides reducing the overall operational time for saving energy, other methods have been investigated. Firstly, the consumed energy can be reduced by avoiding operations that have a high power consumption. Sometimes an operation can be carried out using a slower but more energy efficient method [9]. In the exploration scenario this could be choosing paths in a way that robots can move steadily without frequent accelerations and decelerations. Secondly, energy consumption can be reduced by reducing the frequency that certain components of robots are used. It might make sense to shut off those components, put them in sleep mode while they are not required for the mission, or simply adapt the frequency of their use to the current requirements [2]. In the exploration scenario this can be the reduction of the sensing frequency but it can slow down the exploration process if not carefully set. Finally, robots can be put to sleep mode while they have no task or while they wait for a specific event. During exploration this only makes sense if there are multiple robots competing for one docking station. If the docking station is occupied other robots might need to wait until it becomes available again.

2.4 Trajectory Planning

Opposing to theses approaches some work focuses on planning the actual trajectory to a goal. The approach described in [9] finds the route and determines the velocities of a mobile robot so that it consumes as little energy as possible. In [10] this approach is enhanced by path planning and put to the test in an exploration scenario. The focus is on enclosed indoor environments where robots do not run out of power. In our work we focus on selecting the order of the goal points in large spaces, not finding optimal trajectories. Instead we use search algorithms like Dijkstra [11] or A* [12] which perform sufficiently good for our scenario.

2.5 Frontier Selection

The main task during exploration is to select the next goal for the robot to travel to. The most common approach is introduced in [1]. The idea is to always steer robots to the closest frontier and thereby keep the traveled path relatively short. Another approach is to select the frontier according to the expected information gain [13, 14]. The information gain is a very difficult measure though since it heavily depends on the environment. The next-best-view principle takes a similar approach while trying to minimize the uncertainty of a robot's pose [15, 16, 17]. Other approaches consider the danger [18] or difficulty [19] of the path to the frontier. While there are approaches for considering the energy consumption of a path [20, 21] the focus is not on frontier selection in unknown environments. Often the communication capabilities at the goal or on the path are of importance [22, 23]. It has been shown that the orientation of a robot plays an important role for path planning [10]. This approach uses an orientation based method where the frontiers are selected in a clockwise order starting from a robot's left direction. It is especially suitable for enclosed indoor environments. In this work we combine multiple criteria into a cost function as has been proposed in [24, 25]. Thereby we can combine criteria regarding the energy as well as the exploration efficiency.

The following section describes the proposed exploration strategy that plans the path of a robot in an unknown environment in a way that the limited capacity of the robot's battery is taken into account.

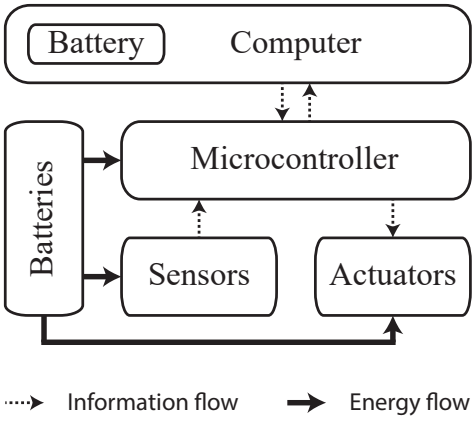


Figure 1 Components of a mobile robot.

3 Energy-Aware Exploration

3.1 Energy Model

We consider the exploration of an unknown environment with a single robot. It is assumed that the robot is powered by a battery with limited capacity so that it has to recharge periodically, e.g., at a docking station. We furthermore assume that the computer is supported by its own battery lasting at least as long as and being recharged together with the robot batteries. This allows the robot to continuously explore the surrounding environment. The robot's power consumption is modeled considering the different components of a mobile robot as depicted in **Figure 1**. The computer is responsible for all high-level computations, such as path planning or map generation; the microcontroller is

Table 1 Power consumption breakdown of the mobile robot while it is moving.

Component	Power consumption	
Sonar	0.6075 W	3 %
Lidar	8 W	36 %
Micro controller	4.6 W	20 %
Locomotion	9.17 W	41 %
Total	22.38 W	100 %

responsible for reading the sensor data as well as generating the commands for the locomotion. In the following we do not consider the computer and its battery in our model. The power consumption of computers for different workloads has been analyzed before, e.g., in [26].

To carry out realistic simulations we base our model on the Pioneer 3-DX robot by Adept Technology, which is a popular research robot. Measurements on its power consumption have been done in [2]. For simplification purposes we consider only the two discrete values P_m and P_s which represent the power consumption when the robot is moving at maximum speed of $v_\mathrm{max} = 1.2\,\mathrm{m/s}$ and when it is stationary, respectively. Using v_max to compute the moving power consumption assures that the battery charge is never overestimated. When the robot is moving, all components draw power; when it is stationary only the microcontroller and the sensors do. This is because the robot's only actuator is its locomotion. The robot is equipped with two sensors: an active sonar sensor used for collision avoidance and a laser scanner used for mapping the environment. The sonar sensor can be modeled with a constant value of $P_\mathrm{sonar} = 607.5\,\mathrm{mW}$. The laser scanner is assumed to be the UTM-30LX by Hokuyo Automatic which can be modeled with a constant value of $P_\mathrm{lidar} = 8\,\mathrm{W}$. The microcontroller uses a constant value of $P_{\mu\mathrm{C}} = 4.6\,\mathrm{W}$. The only time variant consumer is the locomotion. Its power consumption depends on the velocity v of the robot:

$$P_\mathrm{loc}(v) = 0.29\,\mathrm{W} + 7.4\,\frac{\mathrm{W}}{\mathrm{m/s}} \cdot v \tag{1}$$

The individual power consumers are listed in **Table 1**.

This yields the total power consumption of the moving and stationary robot, respectively:

$$P_\mathrm{m} = P_\mathrm{sonar} + P_\mathrm{lidar} + P_{\mu\mathrm{C}} + P_\mathrm{loc}(v_\mathrm{max}) = 22.38\,\mathrm{W} \tag{2}$$

$$P_\mathrm{s} = P_\mathrm{sonar} + P_\mathrm{lidar} + P_{\mu\mathrm{C}} + P_\mathrm{loc}(0) = 13.50\,\mathrm{W} \tag{3}$$

Using these values, the battery's charge E_l is updated at a rate of 0.5 Hz. This leads to the battery's SOC

$$SOC = \frac{E_\mathrm{l}}{E_\mathrm{t}} \tag{4}$$

with the battery's maximum charge E_t. E_l can also be used to compute the distance d_l that the robot is still able to travel

$$d_\mathrm{l} = \frac{E_\mathrm{l}}{P_\mathrm{m}} \cdot \bar{v} \tag{5}$$

where $\bar{v}$ is the average speed of the robot. It becomes obvious that the robot has the limited range of

$$r = \frac{E_\mathrm{t}}{P_\mathrm{m}} \cdot \bar{v}. \tag{6}$$

With the requirement that the robot should not run out of power it can therefore only explore the area that is at most $\frac{r}{2}$ away from the docking station. In free space the maximum explorable area therefore is a circular area with radius $\frac{r}{2}$.

The objective of the energy-aware exploration is to fully explore the area defined by (6) using as little energy as possible. We developed the exploration strategy described in the following subsection to achieve this.

3.2 Exploration Strategy

The main objective of the exploration strategy is to save energy. This means to finish the exploration as fast as possible while traveling on the shortest possible path. To explore unknown space, the robot moves to frontiers [1]. Our approach consists of a cost function in the global path planner of the robot. With this cost function the planner selects the goal which the robot navigates to. The selection of the goal follows a strategy that considers the battery charge of the robot E_l and its orientation as well as the location of the robot and the frontiers in the environment. The planner relies on no further information as the environment is only partially observable. The strategy employed by the planner consists of two steps described in the following subsections: (1) Check Remaining Energy and (2) Select Goal.

3.2.1 Check Remaining Energy

As a first step the planner checks the remaining battery charge E_l of the robot. With (5), it is able to compute the distance d_l that the robot is able to travel before running out of power. Next, the planner iterates over all frontiers and computes their distances from the robot's current position d_g and their distances from the docking station d_gb. All frontiers that satisfy the condition

$$d_\mathrm{l} > d_\mathrm{g} + d_\mathrm{gb} \tag{7}$$

are considered as goal candidates. If there are no frontiers satisfying (7), the robot returns to the docking station for recharging. This approach lets the planner adaptively decide when to return for recharging, depending on the battery charge and the exploration opportunities in the current neighborhood of the robot. This adaptive threshold gives the planner more flexibility compared to the naive approach where the robot returns for recharging once the battery charge drops below a static threshold. It assures that the robot keeps exploring as long as possible with one battery charge before returning for recharging. The robot finished the exploration by completely exploring the surrounding environment if there are no frontiers in reach after recharging.

3.2.2 Select Goal

The second step is to select the goal from the list of frontiers. This is a very important step as it defines the path

ISBN 978-3-8007-4231-8

the robot travels. As the robot should travel on the shortest possible path, this path needs to have as little redundancy as possible. To achieve this we propose the cost function

$$f = w_1 \cdot d_\mathrm{g} + w_2 \cdot d_\mathrm{gb} + w_3 \cdot d_\mathrm{gbe} + w_4 \cdot \theta_\mathrm{rel} \qquad (8)$$

which is evaluated for each frontier. The path planner selects the frontier as goal that minimizes the cost function f.

The cost function consists of four parameters that are weighted by the weights $w_i,\ i \in [1,4]$. The weights are used to tune the parameters for specific environments. The first parameter d_g is the distance from the robot to the frontier; it ensures that the planner prefers short paths. The second parameter d_gb is the distance from the frontier to the docking station, which keeps the robot close to the docking station. The third parameter d_gbe is very similar but it depends on the current battery charge E_l. It is defined by

$$d_\mathrm{gbe} = \begin{cases} -d_\mathrm{gb} & E_\mathrm{l} > 0.5 \cdot E_\mathrm{t} \\ d_\mathrm{gb} & E_\mathrm{l} \leq 0.5 \cdot E_\mathrm{t} \end{cases}. \qquad (9)$$

This ensures that in the beginning, the planner prefers frontiers which are further away from the docking station. Once the robot's battery starts to deplete, the planner prefers frontiers closer to the docking station. Eventually, when the robot has to recharge, it is close to the docking station and and does not have to travel a long distance through already known space. The final parameter θ_rel relates the robot's orientation to the direction of the frontier. It is defined by

$$\theta_\mathrm{rel} = \frac{1}{\pi} \cdot \left(\pi - \left|\left|\theta_\mathrm{r} - \theta_\mathrm{g}\right| - \pi\right|\right) \qquad (10)$$

where $\theta_\mathrm{r} = \mathrm{atan2}(\Delta x_\mathrm{r}, \Delta y_\mathrm{r})$ is the direction the robot came from and $\theta_\mathrm{g} = \mathrm{atan2}(\Delta x_\mathrm{g}, \Delta y_\mathrm{g})$ is the direction from the robot to the frontier. By minimizing θ_rel the planner prefers frontiers that are straight ahead and avoids frontiers where the robot has to turn around and go back the way it came. The exploration strategy is summarized in **Algorithm 1**. The positions p_r, p_rp, p_f, p_b, and p_g are always specified as x and y coordinates. Four functions are defined: `frontiers()` retrieves a list of the currently available frontiers, `battery_charge()` reads the current charge of the robot's battery, and `avg_speed()` computes the average speed at which the robot traveled so far. The implementation of the `distance()` function depends on the environment: in free space it computes the Euclidean distance between two points, otherwise it needs to implement a search algorithm like Dijkstra [11] or A* [12].

Algorithm 1: Exploration Strategy

Global parameters:

- Docking station position p_b
- Battery capacity E_t
- Weights $w_i,\ i \in [1,4]$
- Power consumption P_m

$p_\mathrm{r} \leftarrow p_\mathrm{b}$ `// initial position of robot`
$p_\mathrm{rp} \leftarrow p_\mathrm{b}$ `// prev. robot position`
while true **do**
  $P_\mathrm{f} \leftarrow$ `frontiers()` `// list of frontiers`
  $E_\mathrm{l} \leftarrow$ `battery_charge()`
  $\overline{v} \leftarrow$ `avg_speed()`
  $f_\mathrm{min} \leftarrow \infty$
  $p_\mathrm{g} \leftarrow p_\mathrm{r}$
  $d_\mathrm{l} \leftarrow \frac{E_\mathrm{l}}{P_\mathrm{m}} \cdot \overline{v}$
  foreach p_f **in** P_f **do** `// loop all frontiers`
    $d_\mathrm{g} \leftarrow$ `distance(`$p_\mathrm{r}, p_\mathrm{f}$`)`
    $d_\mathrm{gb} \leftarrow$ `distance(`$p_\mathrm{f}, p_\mathrm{b}$`)`
    if $d_\mathrm{l} > d_\mathrm{g} + d_\mathrm{gb}$ **then** `// frontier in reach`
      if $E_\mathrm{l} > 0.5 \cdot E_\mathrm{t}$ **then**
        $d_\mathrm{gbe} \leftarrow -d_\mathrm{gb}$
      else
        $d_\mathrm{gbe} \leftarrow d_\mathrm{gb}$
      end
      $\theta_\mathrm{rel} \leftarrow \frac{1}{\pi} \cdot \left(\pi - \left|\left|\mathrm{atan2}\left(p_\mathrm{r} - p_\mathrm{rp}\right) - \mathrm{atan2}\left(p_\mathrm{f} - p_\mathrm{r}\right)\right| - \pi\right|\right)$;
      $f \leftarrow w_1 \cdot d_\mathrm{g} + w_2 \cdot d_\mathrm{gb} + w_3 \cdot d_\mathrm{gbe} + w_4 \cdot \theta_\mathrm{rel}$
      if $f < f_\mathrm{min}$ **then** `// select frontier`
        $f_\mathrm{min} \leftarrow f$
        $p_\mathrm{g} \leftarrow p_\mathrm{f}$
      end
    end
  end
  if $p_\mathrm{r} = p_\mathrm{g}$ **then** `// no frontier in reach`
    if $p_\mathrm{r} = p_\mathrm{b}$ **then** `// robot at dock. station`
      break `// end of exploration`
    else
      $p_\mathrm{g} \leftarrow p_\mathrm{b}$ `// go recharging`
    end
  end
  $p_\mathrm{rp} \leftarrow p_\mathrm{r}$
  $p_\mathrm{r} \leftarrow$ `move_robot(`p_g`)`
end

3.3 Implementation

A first implementation of Algorithm 1 is done in MATLAB. The exploration is carried out on 2D maps represented by an array, where each array element can be either free or occupied. The robot explores the map by moving to frontiers and marking all array elements within its sensor range that are not occluded by obstacles as explored. Frontiers are defined as array elements where at least one neighbor is unknown. Adjacent frontiers are grouped into a single frontier up to a length corresponding to the sensor range. This reduces the computational complexity and the traveling overhead as visiting this single frontier also explores the adjacent frontiers that are in sensor range. For simplification the battery charge is computed by only considering the traveled distance of the robot. This speeds up the simulations as no timer is required to keep track of the stationary periods of the robot. The paths for the robot as well as their distance are computed using the A* algorithm.

A second implementation is done in the robot operating system (ROS) using the Stage simulator [27]. ROS provides a very realistic environment and even allows to exe-

cute the code used for simulations on an actual robot hardware platform. This can then be used to verify the more abstract simulations in MATLAB. For the exploration process the *explorer* node described in [28] is used as a basis. It is extended by the energy-aware strategy. To take care of the battery simulation the package *energy_mgmt* is implemented. It includes the *battery* node that keeps track of the current state of charge of the battery. It frequently publishes the remaining battery percentage as well as the distance the robot is able to travel. During simulations this node decreases the battery charge according to the model described in Section 3.1. When the robot returns to the initial position of the simulation the *battery* node increases the battery charge and frequently publishes the remaining charging time. Once the charging is finished the *explorer* node instructs the robot to continue the exploration. During physical experiments the *battery* node reads the battery charge from the actual battery. This has been implemented for the TurtleBot robot platform. A SOC estimation for other hardware platforms is left for future work. A central part of exploration is simultaneous localization and mapping (SLAM). There are several SLAM implementations for ROS which offer different levels of functionality. To avoid any disturbing effects of the SLAM components during simulations, SLAM has been replaced by two nodes: the *fake_localization* node and the *fake_mapping* node. The *fake_localization* node substitutes for a localization system as a method to provide perfect localization in a computationally inexpensive manner. Similarly, the *fake_mapping* node, which is an adaption of the *map_server* node, simply reads the map information from the map file and publishes the part of it which has been explored. This avoids any mapping errors and allows to focus solely on the performance of the exploration. Finally, the *move_base* node is used by the global planner to steer the robot through the environment. The implementation is tested experimentally on the TurtleBot platform using a docking station to demonstrate the feasibility for real world applications.

The details of the simulations are described in the following section.

4 Simulation and Results

4.1 Performance Metric

The main objective of the EA strategy is to explore as much of the unknown environment as possible while using as little energy as possible. The exploration progress can best be measured by keeping track of the already explored area in m^2. The energy consumption mainly depends on the distance the robot traveled as well as the time used for exploring. For evaluating the MATLAB and the ROS simulations equally we therefore choose the *explored area as function of the traveled distance* as the performance metric.

4.2 Setup

The environment must be sufficiently large to experience the effects of energy limitation. The floor plan of a building is usually not large enough since EA is designed to work continuously for several hours or even days. Therefore, the simulations are carried out on excerpts of three city maps. The first one is simply an empty map representing free space. This allows an evaluation of the exploration strategy independent of environment specifics. The second one is a regular map resembling the Manhattan street map. The third one is an excerpt of an old town street map exhibiting the typical features of an old town with narrow, winding streets and some open places.

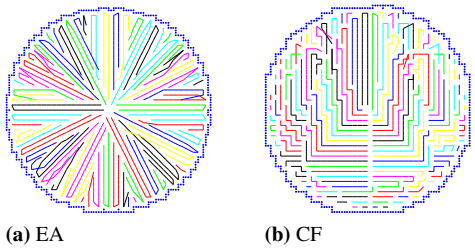
The robot specifics are mentioned in Section 3.1. The laser scanner used is the Hokuyo UTM-30LX with a range of 30 m.

The next step is to fix the four weights $w_i,\ i \in [1,4]$. By assigning values the importance of the parameters of the cost function can be defined. Different weight configurations correspond to different robot movement patterns. An exhaustive search is conducted to find the best performing weight configuration. All weights are varied in the range $[0,250]$. For each permutation the exploration process is simulated in MATLAB to measure the performance. This process is repeated for different environments as each environment might have a different set of optimal weights. First, the search is carried out in free space, i.e., the robot operates on an empty map without obstacles. If the robot is to explore 100 % of the environment the best strategy would steer the robot straight away from the docking station until it used up half of its energy and then back to the docking station. Simulations confirm this assumption as the best weight configuration for free space ($w_1 = 10$, $w_2 = 0$, $w_3 = 9$, $w_4 = 20$) leads to a star shaped movement pattern of the robot. This is depicted in **Figure 2a** where the docking station is placed in the center, the circle of blue dots represent frontiers, and the colored lines represent the robot's traveled path. This weight configuration assigns zero to w_2 which is the weight for the d_{gb} parameter. This is consistent with the definition of the parameters as the d_{gbe} parameter is an energy aware adaption of d_{gb} rendering d_{gb} obsolete. Another typical approach is to select the frontier closest to the robot's current position, named closest-frontier exploration (CF). It is a simpler approach that focuses on reducing the traveled distance but it does not consider the actual energy consumption. This approach is defined by the weights $w_1 = 1, w_2 = w_3 = w_4 = 0$. Its movement pattern can be seen in **Figure 2b**.

Next, the search for an optimal weight configuration is carried out on the Manhattan map. The weights found are $w_1 = 220$, $w_2 = 21$, $w_3 = 104$, $w_4 = 100$.

4.3 Simulation

With the weights defined, the exploration is first simulated in free space. **Figure 3** shows the explored area over the traveled distance for the EA and CF approaches simulated



(a) EA **(b)** CF

Figure 2 Exemplary robot movement patterns in free space.

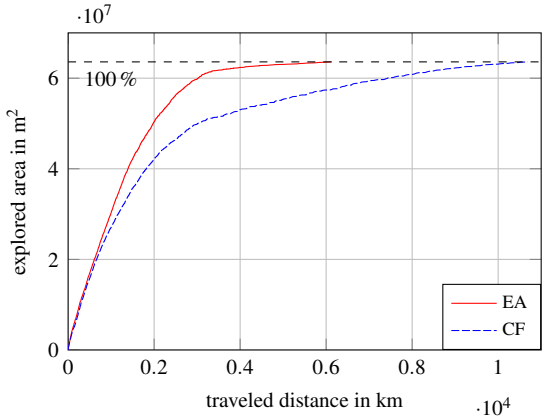


Figure 3 Exploration progress in free space (MATLAB).

in MATLAB. The robot explored 100 % once it cannot reach unexplored space anymore without running out of power. For both approaches the exploration progress was recorded for one single run with the optimized weights. As these are MATLAB simulations, the battery charge is defined as the range the robot can travel with one charge. This range is set to 9000 m. It can be seen that EA shows an improvement of more than 40 % in traveled distance to explore the same area.

To show that the results also hold for more realistic scenarios the exploration of free space is also simulated using ROS. The battery charge is set to 10 Wh. As the simulations run much less stable in ROS, 50 runs are recorded and averaged. The results in terms of average exploration progress and 5 % / 95 % quantiles can be seen in **Figure 4**. They show that the EA approach reduces the required traveling distance by 25 %. The results differ from the ones shown in Figure 3 as they are based on a more realistic setting. Furthermore, the total explored area is much smaller because ROS simulations consist of many components that tend to crash after several hours of operation.

As a next step, simulations are carried out on the Manhattan map using the weight set found above. **Figure 5** shows the explored area over the traveled distance for the EA and CF approaches simulated in MATLAB. The range of the robot is set to 1200 m. In this simulation the advantage of the EA approach is less evident but the traveled distance is still more than 25 % shorter than with the CF approach.

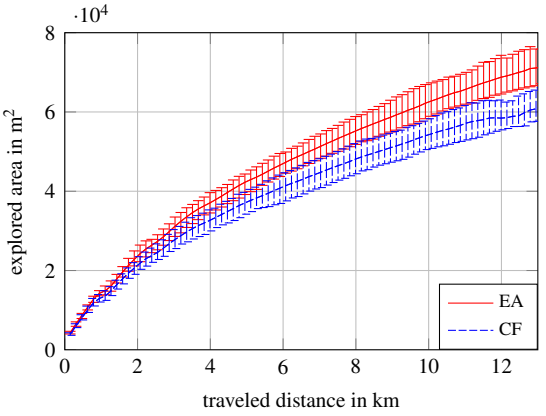


Figure 4 Exploration progress in free space with average and 5 % / 95 % quantiles (ROS).

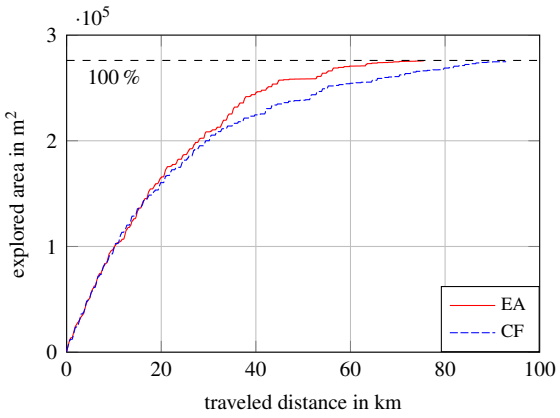


Figure 5 Exploration progress on Manhattan map (MATLAB).

Therefore, EA is also useful for environments containing many obstacles. The exploration progress curve exhibits many dents that reflect the recharging cycles of the robot. They are visible because the explored area as well as the traveled distance is much smaller compared to the previous simulation.

Finally the exploration is also simulated on the old town map. Since the weights are not optimized for this kind of map, all three weight configurations are simulated. This simulation is meant to show how the EA approach performs when the weights are not optimized for the underlying map. The exploration progress is shown in **Figure 6** where EA free corresponds to the EA approach with weights optimized for free space and EA Manhattan corresponds to the EA approach with weights optimized for the Manhattan map. The range of the robot is set to 2000 m. Figure 6 illustrates that EA free outperforms both other approaches in the beginning but the other two approaches catch up later on. This is due to the irregular structure of the map. However, during most of the exploration the EA approach outperforms the CF approach; only towards the end the advantage of EA decreases. Hence, the EA approach also works

ISBN 978-3-8007-4231-8

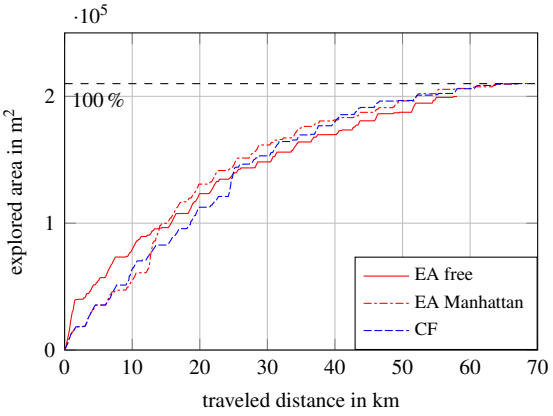


Figure 6 Exploration progress on old town map (MATLAB).

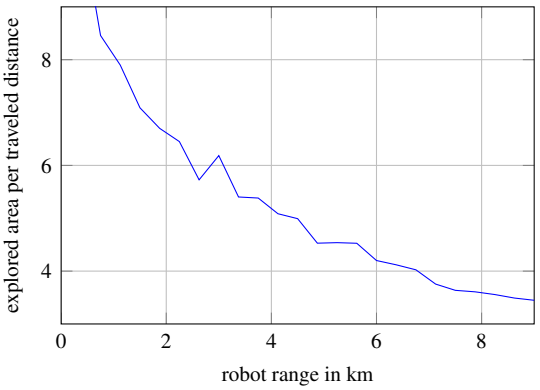


Figure 7 Exploration performance for varying robot ranges.

well for scenarios where the structure of the environment is unknown prior to the exploration but it cannot reveal its full potential. In many cases the exact map of the environment is not known but a rough idea of the structures can already help to improve the parameter weighting to increase the exploration performance.

All results show that the exploration progresses fast in the beginning and slows down towards the end. This phenomenon is further analyzed by recording the performance in explored area per traveled distance for varying robot ranges. The result of this simulation is plotted in **Figure 7**. It can be seen that the performance monotonically decreases with increasing range of the robot. This means that the less charge the robot's battery can carry the higher the exploration performance. Of course this also means that the absolute area explorable by the robot is smaller. The reason is that the robot explores only areas close to the docking station and spends less time traveling through already explored areas. This result confirms the observations from above as in the beginning of an exploration there are many areas yet to explore and towards the end most areas are explored and the robot needs to travel very far each time after recharging to reach those areas.

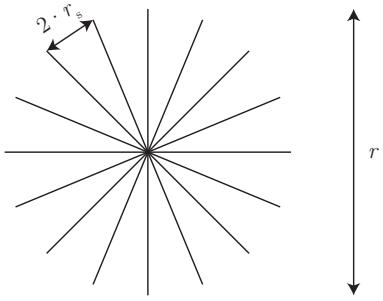


Figure 8 Star pattern for optimal exploration of free space.

4.4 Competitive Analysis

Another method for evaluating the performance of EA is to compare it to the optimal solution. To compute the optimal solution, full knowledge of the environment is required and the computation is done offline. For the exploration scenario the algorithm for computing the optimal solution depends on the map type. For free space it can be computed using the star pattern described in Section 4.2. It is depicted in **Figure 8** where r is the robot range from (6) and r_s is the sensor range. To completely explore the reachable environment the robot has to visit $n = \frac{2\pi \cdot r/2}{2 \cdot r_s}$ points at distance $r/2$. This yields a total path length of

$$d = n \cdot r = \pi \frac{r^2}{2 \cdot r_s}. \tag{11}$$

In the simulation setup described above r is set to 9000 m and r_s to 30 m. This yields a total path length of $4.24 \cdot 10^6$ m in the optimal case. In the simulations the robot has to travel $6.14 \cdot 10^6$ m to finish the exploration. This path is only 45 % longer than the optimal path which is a reasonable good result considering that the robot has no prior knowledge about the environment.

5 Conclusion

In this paper we presented a strategy for energy-aware exploration (EA) of unknown environments with a mobile robot. With this strategy the robot can explore an environment more efficiently and with higher autonomy. The strategy implements a cost function that considers the robot's battery charge to select frontiers during the exploration. Furthermore this strategy uses an *adaptive threshold* to decide when to return to the starting point of the exploration for recharging at a docking station. Thereby the robot can autonomously explore large environments without the need for a human operator. A proof of concept on the TurtleBot platform implemented in ROS shows the feasibility on actual hardware.

Future work will focus on coordinating multiple robots to extend the range that poses a hard limit on the explorable area by a single robot.

ISBN 978-3-8007-4231-8

6 Literature

[1] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. on Comp. Intel. in Robotics and Automation (CIRA)*, Jul. 1997, pp. 146–151.

[2] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques," in *Proc. Int. Conf. on Advanced Robotics (ICAR)*, Jul. 2005, pp. 492–497.

[3] V. Berenz, F. Tanaka, and K. Suzuki, "Autonomous battery management for mobile robots based on risk and gain assessment," *Artificial Intelligence Review*, vol. 37, no. 3, pp. 217–237, Mar. 2011.

[4] M. C. Silverman, B. Jung, D. Nies, and G. S. Sukhatme, "Staying alive longer: Autonomous robot recharging put to the test," Center for Robotics and Embedded Systems (CRES), Tech. Report 15, 2003.

[5] P. Shi and Y. Zhao, "Application of unscented kalman filter in the soc estimation of li-ion battery for autonomous mobile robot," in *Proc. IEEE Int. Conf. on Information Acquisition (ICIA)*, Aug. 2006.

[6] J. Wawerla and R. T. Vaughan, *Advances in Artificial Life: Proc. European Conf. (ECAL)*. Springer Berlin Heidelberg, 2007, ch. Near-Optimal Mobile Robot Recharging with the Rate-Maximizing Forager, pp. 776–785.

[7] H. Wei, B. Wang, Y. Wang, Z. Shao, and K. C. C. Chan, "Staying-alive path planning with energy optimization for mobile robots," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3559–3571, 2012.

[8] Y. Yan and Y. Mostofi, "To go or not to go: On energy-aware and communication-aware robotic operation," *IEEE Trans. on Control of Network Systems*, vol. 1, no. 3, pp. 218–231, Sep. 2014.

[9] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "Energy-efficient motion planning for mobile robots," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 5, Apr. 2004, pp. 4344–4349.

[10] Y. Mei, Y.-H. Lu, C. S. G. Lee, and Y. C. Hu, "Energy-efficient mobile robot exploration," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2006, pp. 505–511.

[11] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[12] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[13] B. Si, K. Pawelzik, and J. M. Herrmann, "Robot exploration by subjectively maximizing objective information gain," in *Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, Aug. 2004.

[14] J. Vallve and J. Andrade-Cetto, "Potential information fields for mobile robot exploration," *Robotics and Autonomous Systems*, vol. 69, pp. 68–79, 2015.

[15] H. H. Gonzales-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, Oct. 2002.

[16] C. Stachniss and W. Burgard, "Exploring unknown environments with mobile robots using coverage maps," in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, Aug. 2003.

[17] C. Potthast and G. S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment," *Visual Communication and Image Representation*, vol. 25, no. 1, pp. 148–164, 2014.

[18] J. C. Espino, B. Steux, and O. E. Hamzaoui, "Safe navigating system for indoor environments," in *Proc. Int. Conf. on Automation, Robotics and Applications (ICARA)*, Dec. 2011.

[19] S. Wirth and J. Pellenz, "Exploration transform: A stable exploring algorithm for robots in rescue environments," in *Proc. IEEE Int. Workshop on Safety, Security and Rescue Robotics (SSRR)*, Sep. 2007.

[20] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Trans. on Robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[21] C. C. Ooi and C. Schindelhauer, "Minimal energy path planning for wireless robots," *Mobile Networks and Applications*, vol. 14, no. 3, pp. 309–321, 2009.

[22] J. Yuan, Y. Huang, T. Tao, and F. Sun, "A cooperative approach for multi-robot area exploration," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2010.

[23] P. Abichandani, H. Y. Benson, and M. Kam, "Multi-vehicle path coordination in support of communication," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2009.

[24] J. Butzke and M. Likhachev, "Planning for multi-robot exploration with multiple objective utility functions," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Sep. 2011.

[25] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *Proc. Int. Symp. on Robotics (ISR) and German Conf. on Robotics (ROBOTIK)*, Jun. 2010.

[26] A. Mahesri and V. Vardhan, *Power-Aware Computer Systems: Int. Workshop (PACS)*. Springer Berlin Heidelberg, 2005, ch. Power Consumption Breakdown on a Modern Laptop, pp. 165–180.

[27] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, no. 2, pp. 189–208, 2008.

[28] T. Andre, D. Neuhold, and C. Bettstetter, "Coordinated multi-robot exploration: Out of the box packages for ROS," in *Proc. Int. Workshop on Wireless Networking and Control for Unmanned Autonomous Vehicles (Wi-UAV)*, Dec. 2014, pp. 1457–1462.

ISBN 978-3-8007-4231-8