I. LANDMARK BASED LOCALIZATION UTILIZING MONTE CARLO

Localization

Anthony Calandra
Electrical And Computer Engineering
Oakland University
Rochester Hills, Michigan
acalandra@oakland.edu

***Abstract*— Human generated maps are effective tools at capturing a robot's starting position and providing a partially reliable apriori map to aid in effective localization and mapping for an autonomous robot. Since concise starting maps are not always accessible to mobile robots, this paper investigates the use of a human created starting map containing landmarks, as well as, information on the robot's current pose to aid in localization. This paper focuses on the utilization of image processing to extract features and robot data from provided maps and generate map data that can be consumed by our navigation stack for effective localization. The utility of this method is investigated through simulation-based experiments.**

II. INTRODUCTION

Localization and mapping are heavily studied fields in mobile robotics [2]. Localization refers to estimating the robot's pose. Then, mapping refers to constructing a map of the robot's environment as well as utilizing a map of environment data, such as landmarks, to localize and navigate. [4] The large base of research in this field has yielded highly effective methods of localization[4], however these methods are often dependent on having accurate maps of the robots environment provided [2,3]. Some methods address the reliance on accurate maps by having humans draw maps of the robot's environment and provide it to the robot. However, these methods are tested in indoor environments and the robot's tend to either preform with high success or be completely ineffective in localizing based on the environment it is placed in. This shows their algorithms have a strong dependence on environments layout.[2], our technique would ideally address by including the robots starting position in the drawn map, to provide additional context to our system to start, as well as to, concisely capture relevant data from the drawn map, such that the system receives minimal erroneous data.

In many cases robots will not have the luxury of producing or being provided a perfect map to start, such as in the case of search and rescue in natural disaster areas, as the environment would be heavily damaged and any existing maps of the area would no longer be accurate, at which point methods utilizing the aforementioned maps would struggle to operate effectively[2].

This paper address those issues, by looking for an effective method of receiving a user generated map. However this method should be simple to minimize burden on the user , such that strong artistic ability isn't required. However the hand drawn map should still be able to provide the robot much needed environmental context when it isn't possible for itself to acquire it from the start. We assume the user's map will be inaccurate, in that it will be non-proportional, may have landmarks drawn out of place positionally, or contain landmarks that are overlapping or drawn twice. We try to minimize this by creating an intuitive human machine interface to allow for the person to create as accurate of a map as possible.. We also leverage image processing to generate as dependable of a map as possible for the robot and to correlate the pixel distances to metical distance in the robot's actual environment. To localize our robot we utilize the Monte Carlo Localization(MCL) algorithm, with that said, we had to augment MCL to fit our application, as we utilize pixel based coordinates in our system to represent landmarks and our robot's pose.

III. RELATED WORK

With the large body of research in mobile robot localization a number of different methodologies for solving the localization problem have been developed [2]. In regards to my paper I'm going to split localization methods into a couple categories, localization algorithms that require prior maps and static environments, then methods that use hand drawn maps. These methods include Markov Localization[10] Monte Carlo Localization(MCL)[3] and Multiple Hypothesis Localization[11][2]. One of the most popular being MCL. As described in Dellaert et al MCL is a localization algorithm that operates in two phases, prediction and update. The prediction phase that starts with a set of particles which is computed every time this algorithm runs. Then the robot's motion model is applied to each particle. A motion model being a model representing the way the robot moves and calculating it's end location after a movement command is ran. Second, is the update phase which utilizes measurements from the measurement model. Measurements being the readings from a sensor, such as a camera or rangefinder, and the measurement model, being the logic that processes measurements to create usable data for the rest of MCL. The weights of the particles, which represent the probability of a particle prediction given a

measurement, are also used. Continuing with the update phase we perform resampling, where we select the highest probability particles to then be set into the new set of particles. A limitation of this work is sample impoverishment, which is the selection a high weight particle multiple times, creating a lack of particle diversity. This is an issue as it can lead to suboptimal solutions or states where the robot will need to move to regain particle diversity.

Continuing we have methods of localization that utilize “hand drawn” maps the idea of using a hand drawn map instead is that it is a dynamic and quick solution to providing a robot a map when a complete map like a floor plan isn’t available or no longer accurate, such as if the building was damaged in a natural disaster. The area of hand drawn maps hasn't been as thoroughly researched, as stated in Behzadian et al. most works that use hand drawn maps are not using them as a tool for enhancing a localization algorithm but as a means of communication between human and robot.[2] One work that does use hand drawn maps for localization, Behzadian et al. does so by pairing a hand drawn map with Monte Carlo Localization. In this paper a augmented version of MCL(described earlier) is utilized. This algorithm is modified in two ways the robot’s state space is modified to have another variable to represent local deformation in the drawn map, such as having waviness in a drawn wall. As well, the robot is localized in a pixel coordinate frame meaning the coordinate system uses distance based on the amount of pixels separating two points, instead of a world coordinate frame. This implementation however, showed dependence on its operating environment for successful localization. Such that the robot will either be highly successful in localizing or will perform poorly based on the layout of where it is operating. This is shown in the experimentation results, as the robot’s success rate was over 80% in 2 of the 9 room layouts and a success rate of under 40% in 5 of the room layouts. Showing a trend for either strong or poor performance based on its environment.

This paper looks to build on this work, by expanding on hand drawn maps to capture more information, including an initial robot pose, and by generating more consistent map data based on the provided drawn map.

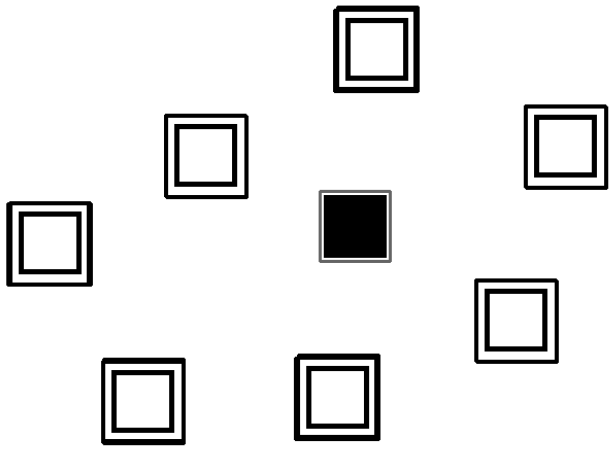
IV. Human Machine Interface

In this section we explain our system design and methodologies. Our system is split into two major components, the human machine interface (HMI) which will receive and process the map the user will create and our Localization system which will be used to allow the robot to find its location in space. The HMI receives a drawn map as an image from a digital design tool, this image is then sent to the system for processing. The maps will contain 2 major pieces of data, landmarks that will be drawn as unfilled squares. These landmarks can include data such as trees, cones, bins, etc. As well, the robots starting location will be drawn using a filled black square. The first thing we do is perform template matching algorithm, which will find all the relevant landmark and robot data, using the normalized cross-correlation technique. [5] (eq 1)

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

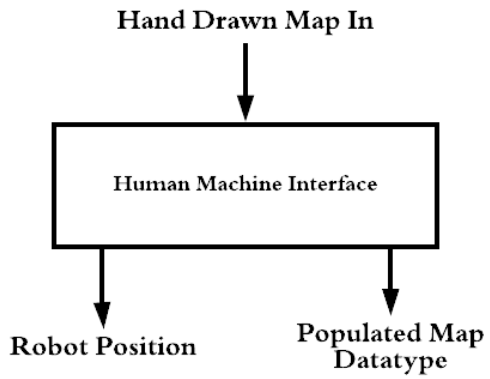
Equation 1

We then draw bounding boxes around these obstacles to capture location data. We then perform a second round of template matching to find the robot and again draw its own bounding box, utilizing the same template matching algorithm (eq 1). This data is then used to populate a map that the robot will use in its localization algorithm. This map consists of the detected positions of the objects and the robots position from the template matching and bounding algorithms



Example Hand drawn map with extracted objects and robot

we determine the size by doing simple arithmetic on the boxes subtracting the x and y values in the top left and bottom right corners to get the object size, and then track the corners of the box to get it’s positions. As well, we perform the same steps to capture the robot’s starting position. This map and robot data is then passed along. We chose to do this as it is computationally inexpensive, and will minimize adding any extra size to the users drawn landmarks as we can create a tight frame around their drawn squares.



V. LOCALIZATION

The way we designed our system is broken down into multiple parts, including the motion model, measurement model, and the core MCL algorithm, which as mentioned earlier in the paper, model the robots movement, translates its sensor measurements to useful map data, and then determines the robots location in space. These are based on the implementation in of the following algorithm MCL[6] with adjustments made to interface with our HMI, such as designing our map to use pixels coordinates as our hand drawn map is process using pixels coordinates.
Beginning we have our MCL algorithm. This is implemented to use a feature-based map designed to be effective at operating with outdoor environments, as the map contains only the shape of the environment at the specific location. Particularly the locations of objects in the map, and allows for effective updating of positions with additional sensing.[4] This map contains landmarks with range, the distance the object is from the robot. Bearing, the direction the object is in, with respect to the robots position. And signature, which is a preselected defining feature of the object this can be something such as color or size. [7]All of these aid in the robot's localization. The following algorithm was the base for our implantation

Algorithm MCL$(\mathcal{X}_{t-1}, u_t, z_t, m)$**:**

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ to M do
  $x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
  $w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$
  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
endfor
for $m = 1$ to M do
  draw i with probability $\propto w_t^{[i]}$
  add $x_t^{[i]}$ to $\mathcal{X}_t$
endfor
return $\mathcal{X}_t$

Equation 2

This algorithm works in two major parts the first part were we create particle predictions, determining each particles pose and weight, with motion and measurement models. First, we calculate the pose of the particle with the motion model(equation 4). This model takes in control, or the movement the robot is instructed to perform, and a particle. The model then will update the particle based on the control. Secondly, we run the measurement model(equation 3) on the particle. This model takes in a particle, the most recent sensor measurements and the robots map. The measurement model then determines a weight to be assigned to the particle based on the predicted new landmark locations after the robot moves. The algorithm takes the probability of the range, bearing and signature predictions being accurate and then assigns an associated weight. The second phase being the resampling phase were we draw particles from the existing particle set to populate a new refined set of particles. Resampling is done based on the particles weight, particles with a higher weight have the highest likelihood of accurately representing the robots true position as, such we want to resample those particles and not the ones that are outliers with low weights. We do not simply take all the particles with the highest weights as that can result in premature convergence which can result in a sub-optimal localization.

Algorithm landmark_model_known_correspondence(f_t^i, c_t^i, x_t, m)**:**

$$j = c_t^i$$
$$\hat{r} = \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2}$$
$$\hat{\phi} = \text{atan2}(m_{j,y} - y, m_{j,x} - x)$$
$$q = \textbf{prob}(r_t^i - \hat{r}, \sigma_r^2) \cdot \textbf{prob}(\phi_t^i - \hat{\phi}, \sigma_\phi^2) \cdot \textbf{prob}(s_t^i - s_j, \sigma_s^2)$$
return q

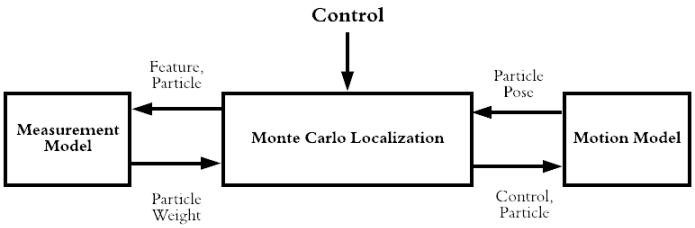
Equation 3

As mentioned earlier above is the measurement model algorithm, you can see the equations utilized to assign the particles weight. Specifically you can see the methodology for calculating the predicted landmark position. Following that is the Motion Model that is also referenced earlier this section. Our model is a simple velocity motion model, represented by the following equations, calculating the predicted position of the robot after the movement is complete. In this equation movement is based on giving the robot a translational and rotational movement control.

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x_c + \frac{v}{\omega} \sin(\theta + \omega\Delta t) \\ y_c - \frac{v}{\omega} \cos(\theta + \omega\Delta t) \\ \theta + \omega\Delta t \end{pmatrix}$$

Equation 4

Below you can see a system design for our localization system and some of the data flow between modules.



VI. EXPERIMENTATION

We evaluated our system in simulation by testing the system's ability to generate accurate maps from the human created drawings as well as the robot's ability to localize accurately within a human generated map. Testing was done utilizing the Gazebo simulator, which is a tool for creating a simulated robot and simulated world environments for the robot to operate in. These simulations are also capable of simulated sensor data as well as track a robot's odometry. Our testing procedure consists of creating three separate worlds for the robot to operate in, each world has three environment configurations, these configurations would include reorienting and slightly repositioning objects in the world, as well as, moving the robot into a new location. and then the robot will attempt to localize 3 times in each configuration. Each localization attempt will consist of the robot taking a sensor snapshot of its environment, then we will measure its ability to localize with that data and the context provided by the drawn map, the test will be considered a failure if the robot fails to localize after 20 seconds. This is done with the human drawing a map for each configuration. Variables we will be controlling in each world and adjusting each configuration to test localization will be lighting in the world, the types of obstacles present, and density of obstacles, meaning that we will attempt to localize with a large amount of objects, as well as, sparse environments.

This will result in 15 distinct environments the robot will need to localize in, with a total of 45 trials. This will showcase not only the system ability to localize in various environments but also its ability to localize consistently and duplicate any successes.

[1] *Conference on Intelligent Robots and Systems (IROS)*, 2015.

[2] **1.** B. Behzadian, P. Agarwal, W. Burgard, G. D. Tipaldi, "Monte Carlo localization in hand-drawn maps", *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[3] Frank Dellaert, Dieter Fox, Wolfram Burgard, Sebastian Thrun, "Monte carlo localization for mobile robots", *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

[4] Sebastian Thrun , Wolfram Burgard , Dieter Fox, Probabilistic Robotics, The MIT Press, 2006, 191-237

[5] Open CV "Object Detection. Internet: https://docs.opencv.org/3.2.0/df/dfb/group__imgproc__object.html#ga586ebfb0a7fb604b35a23d85391329be, Dec. 23, 2016[Nov. 21,2018]

[6] Sebastian Thrun , Wolfram Burgard , Dieter Fox, Probabilistic Robotics, The MIT Press, 2006, 250-263

[7] Sebastian Thrun , Wolfram Burgard , Dieter Fox, Probabilistic Robotics, The MIT Press, 2006, 176-177

[8] Sebastian Thrun , Wolfram Burgard , Dieter Fox, Probabilistic Robotics, The MIT Press, 2006, 127

[9] Sebastian Thrun , Wolfram Burgard , Dieter Fox, Probabilistic Robotics, The MIT Press, 2006, 237

[10] Sebastian Thrun , Wolfram Burgard , Dieter Fox, Probabilistic Robotics, The MIT Press, 2006, 197-205