NERC 2015 - Gazebo simulation implementation

Hannan Ejaz Keen, Adil Mumtaz
Department of Electrical Engineering
SBA School of Science & Engineering, LUMS, Pakistan
{14060016, 14060037}@lums.edu.pk

ABSTRACT

This paper presents autonomous NERC robot which localize and do path planning using a given map. National Engineering Robotics Contest (NERC) is a project conducted in National University of Science and Technology, which organize competitions among students to bring their robots, which can localize themselves, in an unknown environment, and also determine the location of colored boxes to pot balls and plan their motion accordingly with the help of a map. The robot, thus designed, will enter the arena from the designated entrance decided at the start of each match. Each robot will be provided with a map (.bmp file of the top view of the arena) from which it will determine the locations of the obstacles, the boxes and the location of the gates. It will further use the map to plan its motion from the entrance gate to the colored boxes. It will put the correct colored ball in each box and then move on to the exit gate while avoiding obstacles on the way. The location of the obstacles, the boxes and the entry and exit gate are all random. This paper presents a Bayesian filter algorithm for localizing a robot that is equipped with low cost ultrasonic sensor and color sensor. Q-learning is used for path planning. Q-learning is a model-free reinforcement learning technique.

RELATED WORK

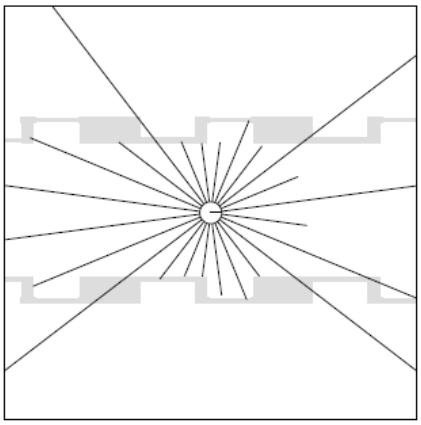
"Localization of multiple robots with simple sensors" has been done before by Mike Peasgood and Christopher Clark. This paper presents a distributed particle filtering algorithm for localizing multiple mobile robots that are equipped with low cost power sensors. This method is applicable to multi-micro robot systems, where size limitations restrict sensor selection. Localization problem is formulated as estimating the global position and orientation of a single triangle, where corners of the triangle represent the positions of robots. Each robot uses an identical particle filter algorithm to estimate the global position of the triangle. The best estimates determined by each particle filter are distributed among the robots for use in the following iteration, using a compass and two range finders.
This method can globally localize the robot team in a simulated indoor environment.

YOUR APPROACH

We have used the Bayesian Filter approach for localization and for path planning we have used Q-learning. In Bayesian Filter there are two categories Measurement Model and Robot motion model.

Measurement model

Measurement model comprise the domain specific model in probabilistic robotics. This model describes the sensor measurements generated in physical world. We have used ultrasonic sensors and color sensors. Probabilistic robotics also models the noise in sensor measurement formally the measurement model is defined as conditional probability distribution $p(z_t|x)$ where x_t is the robot pose, $z_t$is the measurement at time t. Figure given below shows how the ultrasonic sensor gives error.



Sensor model follows the Bayesian theorem and uses the formula

$$p(x_t|z_t) = \frac{p(z_t|x_t)p(x_t)}{\sum p(z_t|x_t)p(x_t)}$$

Sensor errors can encounter as they do not sense the object in front of them, may be the sensor goes faulty, or short readings caused by cross-talk between different sensors. We have to incorporate these errors in our measurement model.

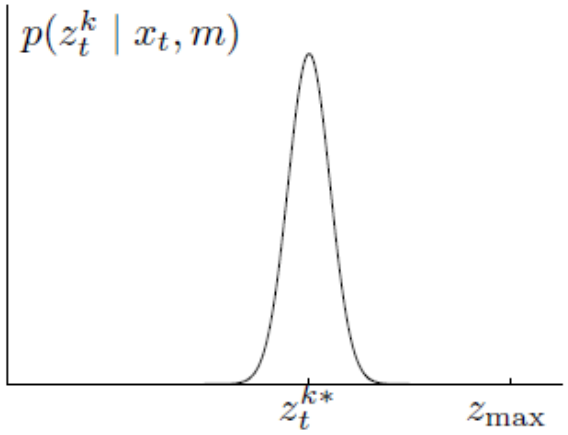
In our model we use four ultrasonic sensors paced at ninety degrees from each other. Our Probability distribution becomes

$$p(z_t \mid x) = \prod_{k=1}^{K} p(z_t^k \mid x_t)$$

All the individual probabilities will be multiplied to get the universal reading. Our model incorporates four types of measurement errors, all of which are essential to making this model work.

1. Correct range with local measurement noise.

In measurement noise, if the sensor correctly measures the range to the nearest object, the value it returns is subject to error. This error arises from the limited resolution of range sensors, atmospheric effect on the measurement signal, and so on. This noise is usually modeled by a narrow Gaussian with mean zkt and standard deviation phit. We will denote the Gaussian by $\boldsymbol{p_{hit}}$. Figure below illustrates this density p_{hit}.



In practice, the values measured by the range sensor are limited to the interval [0; z_{max}], where z_{max}denotes the maximum sensor range. Thus, the measurement
probability is given by

$$p_{\text{hit}}(z_t^k \mid x_t, m) = \begin{cases} \eta\, \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) & \text{if } 0 \le z_t^k \le z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Where

$$\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) = \frac{1}{\sqrt{2\pi\sigma_{\text{hit}}^2}}\, e^{-\frac{1}{2}\frac{(z_t^k - z_t^{k*})^2}{\sigma_{\text{hit}}^2}}$$

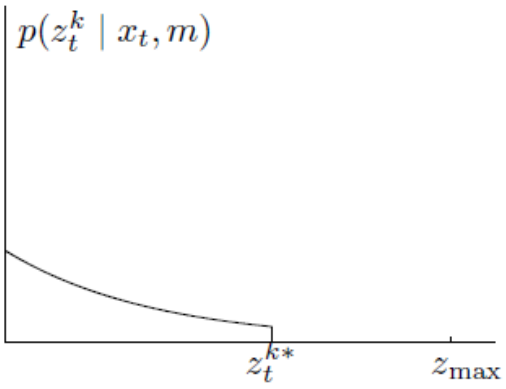
The normalizer η evaluates to

$$\eta = \left(\int_0^{z_{\max}} \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2)\, dz_t^k \right)^{-1}$$

The variance p_{hit} is an intrinsic noise parameter of the measurement model.

2. Unexpected objects.

Environments of mobile robots are dynamic as a result, objects not contained in the map can cause range finders to produce surprisingly short ranges when compared to the map. We treat this as sensor noise. The likelihood of sensing unexpected objects decreases with range. Mathematically, the probability of range measurements in such situations is described by an exponential distribution. The parameter of this distribution,λ_{short}, is an intrinsic parameter of the measurement model. The following equation for $p_{short}(x_t|z_t)$



$$p_{\text{short}}(z_t^k \mid x_t) = \begin{cases} \eta\, \lambda_{\text{short}}\, e^{-\lambda_{\text{short}} z_t^k} & \text{if } 0 \le z_t^k \le z_t^{k*} \\ 0 & \text{otherwise} \end{cases}$$

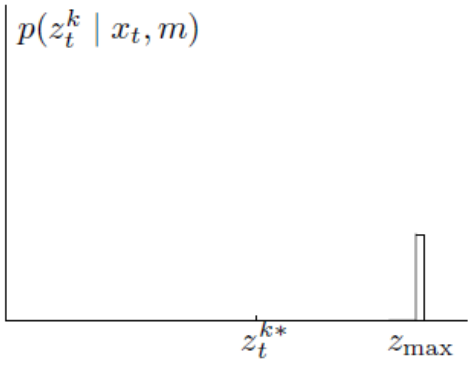
Where

$$\eta = \frac{1}{1 - e^{-\lambda_{\text{short}} z_t^{k*}}}$$

Figure given below shows the exponential distibution in sensor model

3. Failures.

Sometimes, obstacles are missed altogether. For example, this occur with laser range finders when sensing black, light-absorbing objects, or when measuring objects in bright light. A typical result of sensor failures are max-range measurements: the sensor returns its maximum allowable value z_{max}. The probability for max range is

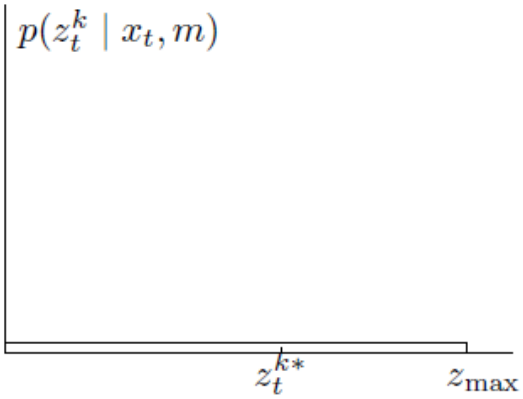


$$p_{\max}(z_t^k \mid x_t, m) = I(z = z_{\max}) = \begin{cases} 1 & \text{if } z = z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Figure given below shows the Uniform distribution
Here I denote the indicator function that takes on the value 1 if its argument is true, and is 0 otherwise.

4. Random measurements.

Range finders occasionally produce entirely unexplained measurements. For example, they are subject to cross-talk between different sensors. To keep things simple, such measurements will be modeled using a uniform distribution spread over the entire sensor measurement range[0; z_{max}] :



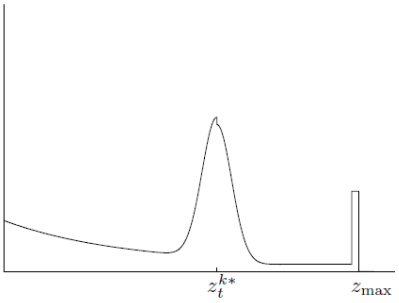
$$p_{\text{rand}}(z_t^k \mid x_t) = \begin{cases} \frac{1}{z_{\max}} & \text{if } 0 \leq z_t^k < z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Figure below shows the density of the uniform distribution p_{rand}

These four different distributions are now mixed by a weighted average, defined by the parameters z_{hit}, z_{short}, z_{max}, and z_{rand} with z_{hit}+ z_{short}+ z_{hit}+ z_{rand}= 1.

$$p(z_t^k \mid x_t, m) = \begin{pmatrix} z_{\text{hit}} \\ z_{\text{short}} \\ z_{\max} \\ z_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\text{hit}}(z_t^k \mid x_t, m) \\ p_{\text{short}}(z_t^k \mid x_t, m) \\ p_{\max}(z_t^k \mid x_t, m) \\ p_{\text{rand}}(z_t^k \mid x_t, m) \end{pmatrix}$$

A typical density resulting from this linear combination of the individual densities is shown in Figure below



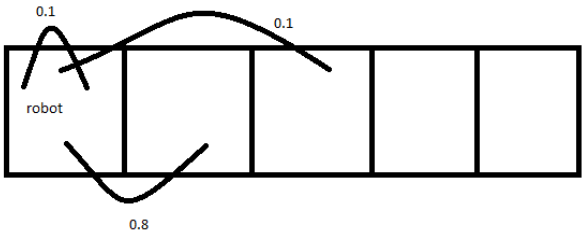
"Pseudo-density" of a typical mixture distribution $p(z_t^k \mid x_t)$.

Algorithm beam_range_finder_model(z_t, x_t, m):

$q = 1$
for $k = 1$ *to* K *do*
  compute z_t^{k*} *for the measurement* z_t^k *using ray casting*
  $p = z_{\text{hit}} \cdot p_{\text{hit}}(z_t^k \mid x_t, m) + z_{\text{short}} \cdot p_{\text{short}}(z_t^k \mid x_t, m)$
    $+ z_{\text{max}} \cdot p_{\text{max}}(z_t^k \mid x_t, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k \mid x_t, m)$
  $q = q \cdot p$
return q

Motion Model:

Motion model is achieved by using a simple technique. Using the above probability distribution we do a motion of 1 unit and allotting the probability to the error, that whether the robot have moved for 1 unit or it has left behind or have moved forward than 1 unit. This method is than distributed at every pose of each state.



In figure above the 0.1 and 0.8 are is the probability allotted to the error and 0.8 to the original movement.

Q-learning:

The Q-Learning algorithm goes as follows:
1. Set the gamma parameter, and environment rewards in matrix R.
2. Initialize matrix Q to zero.

For
{
  Select a random initial state.

  Do
  {
    While the goal state hasn't been reached.

- Select one among all possible actions for the current state.
- Using this possible action, consider going to the next state.
- Get maximum Q value for this next state based on all possible actions.
- Compute**:**
  Q (state, action) = R (state, action) + Gamma * Max [Q (next state, all actions)]

Set the next state as the current state.
}

}

The algorithm above is used by to learn from experience. In each training session, we explore the environment (represented by matrix R), receives the reward (if any) until it reaches the goal state. The purpose of the training is to enhance our location, represented by matrix Q. More training results in a more optimized matrix Q. In this case, if the matrix Q has been enhanced, instead of exploring around, and going back and forth to the same state, this will find the fastest route to the goal state.

Algorithm to utilize the Q matrix:

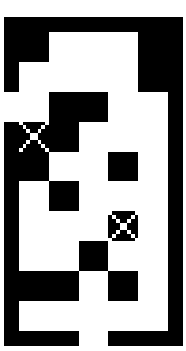
1. Set current state = initial state.
2. From current state, find the action with the highest Q value.
3. Set current state = next state.
4. Repeat Steps 2 and 3 until current state = goal state.

The algorithm above will return the sequence of states from the initial state to the goal state.

The Gamma parameter has a range of 0 to 1 (0 <= Gamma > 1). If Gamma is closer to zero, the state will tend to consider only immediate rewards. If Gamma is closer to one, the state will consider future rewards with greater weight, willing to delay the reward.

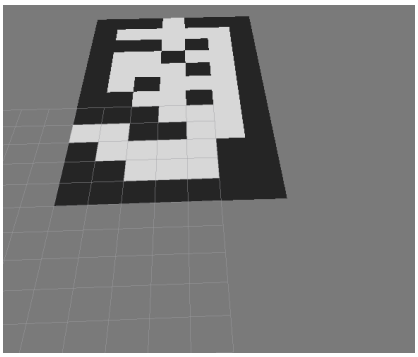
EXPERIMENTS & ANALYSIS

In NERC a map will be given to us in .bmp file as shown in the figure given below
Now by this image we did the image processing and place 0's to the occupied block, 1's where no object is detected and 2 gives the colored block where we have to place the particular color ball in it . By computing the resolution of map we can apply our probabilistic filtering, the sensor model and the motion model. Our image processing code can create any map given in .bmp file by NERC.

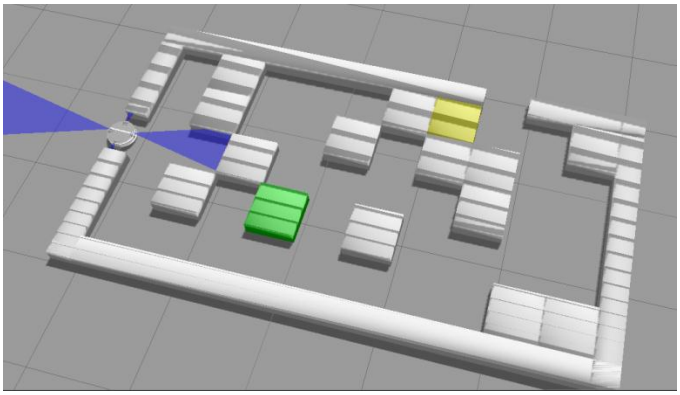


```
0 0 1 1 1 0 0
0 1 1 1 1 0 0
1 1 0 0 1 1 0
0 2 0 1 1 1 0
0 0 1 1 0 1 0
0 1 0 1 1 1 0
0 1 1 1 2 1 0
0 1 1 0 1 1 0
0 0 0 1 0 1 0
0 1 1 1 1 1 0
```

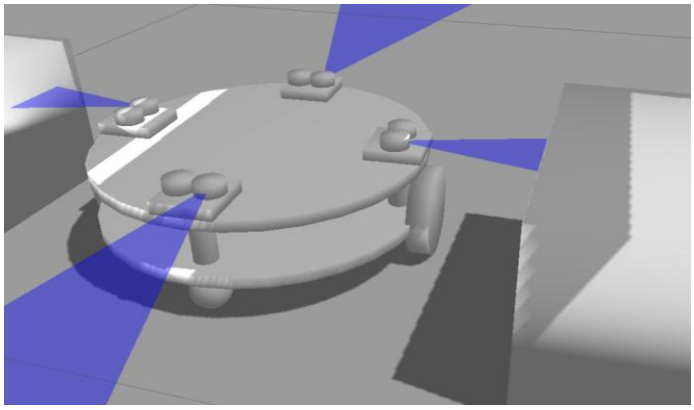
Now this figure shows this matrix on Rviz



From this we create the map on gazebo as describe in the NERC rules by making the block size of 0.5m * 0.5m with height 0.15m. As shown



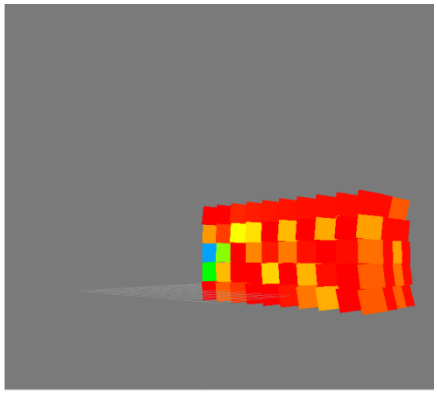
Likewise the robot in gazebo is also build. These two are save in sdf model.



Now placing all these on gazebo creates the topics as shown below

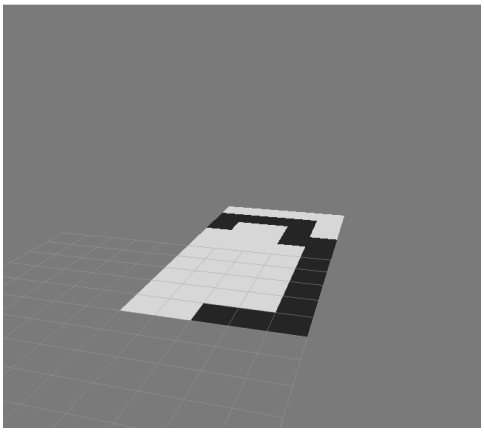
```
hannan@KeenDesktop:~$ rostopic list
/NERC/front_cam/front_image_raw/compressed
/NERC/front_cam/front_image_raw/compressed/parameter_descriptions
/NERC/front_cam/front_image_raw/compressed/parameter_updates
/NERC/front_cam/front_image_raw/compressedDepth
/NERC/front_cam/front_image_raw/compressedDepth/parameter_descriptions
/NERC/front_cam/front_image_raw/compressedDepth/parameter_updates
/NERC/front_cam/parameter_descriptions
/NERC/front_cam/parameter_updates
/NERC/front_scan
/NERC/left_cam/left_image_raw/compressedDepth
/NERC/left_cam/left_image_raw/compressedDepth/parameter_descriptions
/NERC/left_cam/left_image_raw/compressedDepth/parameter_updates
/NERC/left_cam/parameter_descriptions
/NERC/left_cam/parameter_updates
/NERC/left_scan
/NERC/odom
/NERC/rear_cam/parameter_descriptions
/NERC/rear_cam/parameter_updates
/NERC/rear_cam/rear_image_raw/compressedDepth
/NERC/rear_cam/rear_image_raw/compressedDepth/parameter_descriptions
/NERC/rear_cam/rear_image_raw/compressedDepth/parameter_updates
/NERC/rear_scan
/NERC/right_cam/parameter_descriptions
/NERC/right_cam/parameter_updates
/NERC/right_cam/right_image_raw/compressedDepth
/NERC/right_cam/right_image_raw/compressedDepth/parameter_descriptions
/NERC/right_cam/right_image_raw/compressedDepth/parameter_updates
/NERC/right_scan
/clicked_point
/clock
/cloud
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/initialpose
/move_base_simple/goal
/rosout
```

In sensor model we have calculated all the states with pose; this creates the 3-dimensional grid, as we have a state with four poses.



Red shows the low probable location of the robot and the colored is the one with greater probability of robot location. After this step we do a motion of 0.5m as explained in the motion model and place the probability to it. This will create new probabilities of robot position. Now we apply the Bayesian

theorem. This will multiply the probabilities of sensor model and the motion model, this will concise the location of the robot position and after few computations it will give one location which have highest probability and all other will become zero. When we have known the exact location, we apply the Q-learning which calculates the shortest path to the first colored block. Now the robot moves to it, the robot will dropped the same colored ball. Now this Q-learning will manage to create a new path to other colored block and after dropping the ball it creates it path to the out gate and the game finishes. Figure on right shows the shortest path from start point to the out gate. The whole code is ***generic*** which can compute the shortest path of any given map in .bmp image file



FUTURE WORK & CONCLUSION

There are various forms of manipulating this kind of experiment. We have put many constraints for example we have placed the sensor 90 degrees to one another but we can create more efficiency by placing more sensors at limited angles. This can reduce the measurement model errors. Also in the motion movement, the rotation of robot is done in 90 degrees. We can reduce this angle and step size, which will create more states and hence will increase the precision for robot localization.

Various techniques can be used for optimal path planning, as we have used Q-learning instead we can use Dijkstra's Algorithm, A and A star algorithm.

REFERENCES

[1] Peasgood, Mike, Christopher Clark, and John McPhee. "Localization of multiple robots with simple sensors." *Mechatronics and Automation, 2005 IEEE International Conference*. Vol. 2. IEEE, 2005.

[2] Dellaert, Frank, et al. "Monte carlo localization for mobile robots." Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Vol. 2. IEEE, 1999.

[3] Probabilistic Robotics by SK. THRUN, W. BURGARD and D. FOX

[4] Thrun, Sebastian. "Learning metric-topological maps for indoor mobile robot navigation." Artificial Intelligence 99.1 (1998): 21-71.

[5] Arana-Daniel, Nancy, et al. "Reinforcement Learning-SLAM for finding minimum cost path and mapping." World Automation Congress (WAC), 2012. IEEE, 2012.