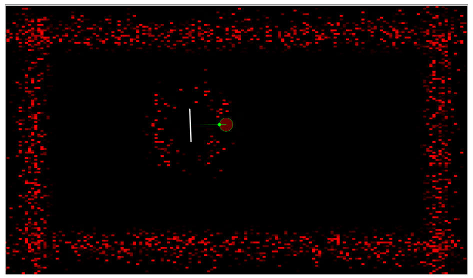
Performance Analysis of Monte Carlo Localization Algorithm

Introduction

One of the key problems of swarm robotics is how the mobile robots can navigate accurately in a given environment. To achieve this, the mobile robots need to accurately determine where they are globally, or relative to other robots and landmarks. This paper is going to be an investigation of the Monte Carlo Localization algorithm in an environment containing 3 anchors.

Localization and Particle Filter



- Robot localization is the process of determining where a mobile robot is located with respect to its environment.
- Monte Carlo Localization is a particle filter based localization algorithm where the sampling distribution is randomized.

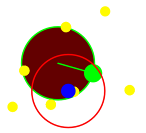
Objective

- Have different noise behavior for sensors
- The objective is to analyze how particle filter behaves in a 3 anchored environment.
- Then find out how the percentage error scales as the step size increases.
- See how particle size affect the localization performance
- Use a different approach to localize a robot in a smaller number of steps.

Methodology

- Creating a simulation
- Have different noise behavior for sensors
- Implement a virtual IMU for the robot
- Implement a virtual LIDAR for the robot
- Implement a particle filter to the simulator

Modified Monte Carlo Localization

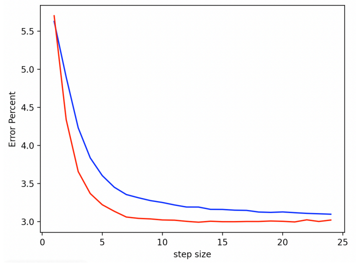


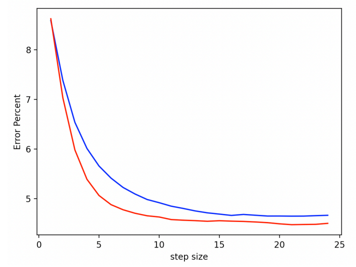
- Get position estimate
- Get direction estimate
- Re-sample portion of the particles for the next step of localization

Algorithm 2 Modified MCL Algorithm

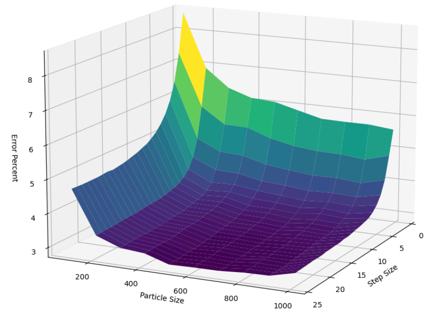
```
**MMCL**($X_{t-1}, u_t, z_t, m, g_{t-1}$):
    $\bar{X}_t = X_t = \emptyset$
    $j = 1$
    **while** $j \neq J$ **do**
        $x_t^{[j]} \sim p(x_t | x_{t-1}^{[j]}, u_t, m)$                ▷ sample particle
        $w_t^{[j]} = p(z_t | x_t, m)$                ▷ calculate weight
        $\bar{X}_t = \bar{X}_t + \langle x^{[j]}, w^{[j]} \rangle$
        $j = j + 1$
    **end while**
    $w_t = w_t / J$                ▷ Normalize the weights
    $j = 1$
    **while** $j \neq J$ **do**                ▷ Importance Sampling
        draw sample $x_t^{[j]}$ with probability $\sim w_t^{[j]}$
        $X_t = X_t + x_t^{[j]}$
        $j = j + 1$
    **end while**
    $X_t = sort(X_t)$                ▷ Sort using weights (high to low)
    $j = J * k$                ▷ $k$ is a constant, from 0 to 1
    $X_t = X_t[0 : j]$                ▷ disregard $k$ portion of $X_t$
    $g_t = centerPoint(X_t)$
    $\theta = g_t - g_{t-1}$                ▷ Estimated Direction
    **while** $j \neq J$ **do**                ▷ Generate new particles
        $n_t = g_t + Gaussian(g_t, lidarNoise)$
        $n_t.dir = \theta + Gaussian(g_t, rotationNoise)$
        $X_t = X_t + n_t$
        $j = j + 1$
    **end while**
    return $\langle X_t, g_t \rangle$
```

Results & Analysis

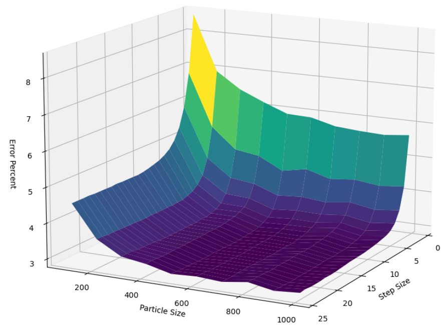




Monte Carlo Localization Algorithm Performance



Modified Monte Carlo Localization Algorithm Performance



Conclusion

- MMCL approaches its limit error percentage than MCL
- MMCL has a lower error rate when used with lower particle sizes.
- MMCL is optimized for 3 anchors.
- MMCL needs to be compared with other advanced Monte Carlo Algorithms.

EEMCS, Delft University of Technology, The Netherlands
Supervisors: Ashutosh Simha, Suryansh Sharma
Responsible Professor: Ranga Rao Venkatesha Prasad