Localization in Part 1

What localization technique did you use to answer the question in part one? Why?

For Part 1 I stored all the measurements and computed the means for distance and turning while heading is computed with the ‘atan’ trigonometric formula using deltas of current and previous measurements .Finally the X and Y are returned as the next position within the circular motion by using basic trigonometric formula:

```
new_x = old_x + cos(turning+heading) * distance;

new_y = old_y + sin(turning+heading) * distance;
```

Localization in Part 2

In part two, the target robot reported noisy position measurements. Did this require you to change your localization technique? If so, how did you change it? If not, do you think the technique you used is optimal for noisy measurements AND noiseless measurements of circular motion?

Starting with part 2 I used a KalmanFilter with both heading and distance within the initial state vector, but leaving x,y and turning outside the filter.

Again I stored all the measurements and computed the mean for the distance while for turning I did the delta between previous and current heading to properly assess the turning correction required within the circular motion and finally appended the obtained value to the current heading estimation.

X and Y were computed with the same formula used above but now I’m using the KF predicted heading and distance to estimate the next position.

Note ideal though for extremely noisy measurements. However it passes all the tests up to Part 5.

Planning in Parts 3 and 4

In parts three and four, you actually had to track down the runaway robot. Describe the planning algorithm you used in each part. Did the planning algorithm you used in part three also work for part four? If not, why not and what did you have to change?

For Part 3 I used a similar approach to Part 2 and in the end I computed the distance between the hunter and the target x,y estimates, while the turning angle was obtained by subtracting the heading of the hunter from the current heading of the target with respect to the hunter.

For Part 4 I added a section where the distance between the hunter and the target is calculated while estimating incrementally the current target position, using the KF predicted heading and distance. In the end the same approach as in Part 3 is used for computing the turning.

Problems Encountered and Lessons Learned

Think back to the problems you encountered while solving this problem. What did you learn about localization, path planning, or robot motion while working on this project?

I mainly learned that KFs are increasing exponentially and thus by increasing the number of elements within the initial state vector, the prediction will get worse when the measurement streams become noisy. Therefore I chose to use only heading and distance as inputs for the filter while computing the rest of the streams with means or circular motion corrections. However this has a limit and if the noise becomes significantly larger, (see part 5) the estimated position becomes unreliable while turning is harder to assess by applying corrections with respect to the previous values.

[Optional] Reflection on Part 5

Only answer this question if you chose to solve Part 5.
Did the solution you used for part 4 also work for part 5? If not, why not? How did the increased measurement noise impact your revised solution? This was a hard problem, and if you completed it we are sincerely interested in your honest reflection!

Tried a UKF model but couldn't make it work just yet.