”Multi-Character Tracking in Ms. Pac-Man Using Particle Filters”

1st Jesús Ortega
jgop@kth.se

2nd Aryan Dolas (partner)
aryand@kth.se

***Abstract*—This project was chosen to showcase one of the many tracking approaches studied during the length of the course EL2320 Applied Estimation at KTH Royal Institute of Technology. Of the different Bayesian filtering methods discussed, the Particle Filter (or Sequential Monte Carlo) was chosen for this project given its wide use on robotics tracking problems and to better understand the challenges of its implementation. For demonstrating the problem, we applied the algorithm to a pre-recorded video of the video game Ms. Pacman, due to the simplicity of both the tracking grid and the ghosts' dynamics. Throughout the paper, it is shown that a Particle Filter approach is suitable for the task of multi-tracking on a 2D map.**

***Index Terms*—Ms. Pacman, NES, Bayesian filters, Particle filters, Tracking problem, Kidnapping problem, Gaussian mixtures.**

I. Introduction

Tracking single or multiple objects is one of the main and most challenging tasks in robotics. When designing mobile robotic systems it is crucial to take into consideration that the environment they are traversing is more often than not dynamic, and therefore it's important to be able to detect and keep track of any possible obstacles or points of interest. The way we do this is by analyzing video input looking for these points and applying one or more different algorithms to track them in the space of interest. In this project, we approach the task of tracking multiple agents in a single frame, in this case, the videogame character Ms. Pacman and the ghosts that follow her. Common challenges that arise from this task in real-life applications are occlusions in the video frames and the special case of a ”kidnapped” robot, which can also be simulated into the video game. In this project, we use a particle filter, which takes a probabilistic (or Bayesian) approach in addition to a measurement model to estimate the pose of the tracked agents in time.

Ms. Pacman is a 1982 arcade video game, released as a sequel to the 1980 video game Pacman. The game consists of the player controlling a yellow ball-eating character throughout a maze while avoiding being eaten by 4 colored ghosts. Several different Nintendo Entertainment System games were proposed, given the simplicity of the screens at the time. Ms. Pacman was chosen given the simple and well-defined motion model of the characters (up, down, left and right), as well as their very distinctive and unique colors, which simplify greatly the measurement model of the algorithm. This game also simulates well the case of kidnapping, as one special path in the videogame takes the characters from one side of the screen to the other.

In this paper, we define the motion and measurement models implemented into the algorithm, which was inspired by the particle filter used during the laboratory tasks using Matlab. We then comment on the observed behavior of the algorithm to draw conclusions.

This paper is structured as follows: Section II gives a quick introduction to the state-of-the-art of Particle Filters and a background of their use in robotics. Section III shows the steps taken and an explanation of their relationship to our task. Sections IV and V introduce our results and conclusions.

II. Background

A. Background

The particle filter is a type of Bayesian filter, which is used to estimate the state of a dynamic system based on probabilistic models [1]. It uses the Sequential Monte Carlo approach, which involves representing the posterior distribution of the state with a set of M weighted particles, which serve as samples that approximate the true distribution. In the context of tracking, its goal is to iterate and update the belief about the system's current state in time.

B. State of the Art and related approaches

Particle filter approaches started gaining popularity in the late 90s as a solution to challenges that arose from non-linearity and non-Gaussianity in state estimation problems. Different techniques like the Unscented Particle Filter (UPF) and Gaussian Mixture Models (GMM) have been proposed to address these challenges, gaining them wide use today in robotic systems.

A very common approach is to combine the particle filter with a different filtering method like the Kalman filter, combining the strengths of each method for different regions of the state space and comparing their efficiency for complex tasks. In [2], the authors dive into the efficiency of unscented particle filters for approaching non-linear state spaces, while in [3] the authors describe instead the breakthroughs that have been developed on the unscented Kalman filter to improve its performance against traditional Gaussian approaches. [4] Explains how a hybrid approach can easily outperform them based on Gaussian mixture representations.

Particle filters are also widely used on tasks like simultaneous localization and mapping (SLAM). A good example of this is the TIAGO robot simulations, easily accessible to any ROS user. [5] Explains how to easily simulate a localizing problem using the particle filter with a TIAGO (Take It And Go) robot, while the ROS wiki [6] gives a full explanation of the coding behind the simulation. Even though the localizing problem is different from the tracking problem, the simulation can be useful for understanding the mechanics of the algorithm in real-time.

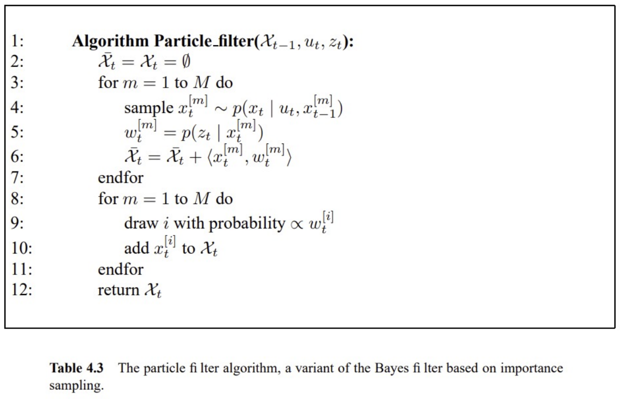
With the current surge of machine learning techniques, recent developments include their integration to facilitate online learning and adaptation within particle filters, allowing them to adapt to changing system dynamics. The authors in [7] explain how to use hybrid machine learning and particle filter approaches for multi-vehicle detection, similar to our multi-ghost tracking approach.

This is not the first time particle filters have been used for the task of tracking video game objects. [8] Shows the implementation of particle filters for the tracking of a pinball ball, while [9] shows its implementation for the tracking of the Super Mario character. [10] Shows a similar approach to multi-object tracking, in this case with the Kalman filter.

III. Tracking the Ghosts

A. Particle filter

The particle filter algorithm works as follows:



```
Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):
    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
    for $m = 1$ to $M$ do
        sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
        $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    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 $\mathcal{X}_t$
```

Table 4.3 The particle fi lter algorithm, a variant of the Bayes fi lter based on importance sampling.

Fig. 1. Particle filter algorithm

We begin by selecting an M number of particles for tracking. These represent the posterior distribution of our system or the position of the ghosts. At each iteration of the algorithm, we apply a process model to the particles, describing the movements of the tracking objects. We then apply the observation model to weigh the particles according to how close they resemble the true position of the ghosts. For the last step, we use systematic resampling on the particles to choose the set of particles for the next iteration. Systematic resampling was chosen due to its increased performance over vanilla resampling, as studied during the laboratory exercises. A more in-depth explanation of the models is given below.

B. Process model

The process model (or motion model) describes the dynamics of our system, which in this case corresponds to the motion of the ghosts over time. The motion of the ghosts, strictly speaking, is always forward. Ghosts might never reverse their direction of motion (unless they are being eaten, which is a particular case). When they reach corners in the grid, the movement changes randomly to forward, left or right. The ghosts move fast enough that they change directions often, making it possible to simplify their motion process as either up, down, left or right at any particular moment in time.

To the particles, we applied a motion process described by the equation 1:

$$X_t = X_{t-1} + U(0, \sigma) \tag{1}$$

Where σ is a variable that we can change in the code. We tried values ranging from 1 to 5, representing the frame-to-frame displacement of the ghosts in image pixels.

In truth, the actual motion process of the ghosts isn't entirely random, as each has a unique AI programmed to challenge the player. Blinky chases Pac-Man directly; Pinky and Inky try to position themselves in front of him, usually by cornering him; and Clyde will switch between chasing and fleeing from him [11]. Nevertheless, a simplified motion model works in our case, and it's an example of how in real-life applications you are often forced to trade process model accuracy for computing efficiency and compensate for it.

C. Observation model

One of the advantages of choosing Ms. Pacman over other video games for tracking is that each character has a color that is very unique between them and the background.

Blinky's color code is RGB = (255,0,0)

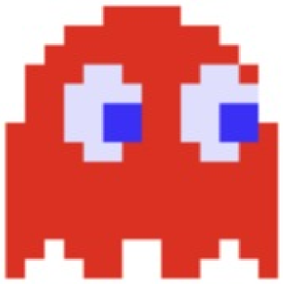


Fig. 2. Blinky - The Ms. Pacman chasing Red Ghost

Pinky's color code is RGB = (255,184,255)

Fig. 3. Pinky - The cornering Pink Ghost

Inky's color code is RGB = (0,255,255)

Fig. 4. Inky - The cyan Goofy Ghost

Clyde's color code is RGB = (255,184,82)

Fig. 5. Clyde - The sensitive Grange Ghost

Ms. Pacman's color code is RGB = (255,255,0)

Fig. 6. Ms. Pacman

It is worth stating that even though these are the actual colors given to the characters, there exists some variation in the true values of the pixels we are measuring. In our code, we apply some color thresholding to compensate for this.

Using the color information, we can apply some masks to the frames to discriminate between the different characters. From the output, it is clear that tracking the red ghost could end up being more challenging than the rest, as the maze itself also contains red in the borders. We further explore these challenges in section IV.

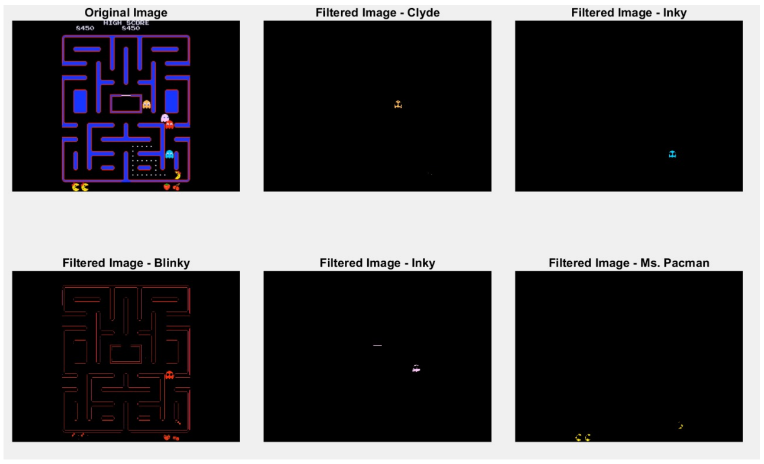


Fig. 7. Results of applying different color masks to the frame

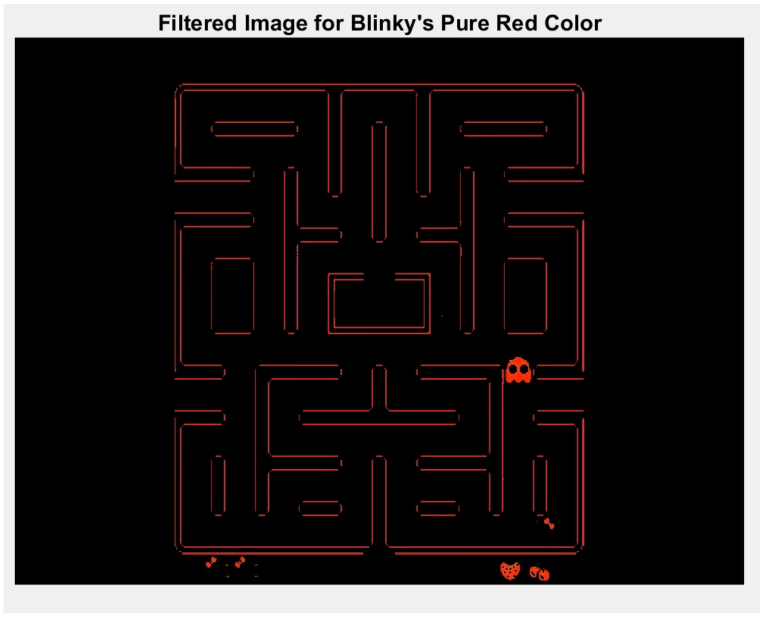


Fig. 8. Blinky mixes with the background

For tracking multiple characters simultaneously, we made some modifications to the measurement model so that we target 2-3 different colors in the frame, corresponding to the two characters of interest. After setting the target colors in the model, we calculate the Euclidean distance in color space between the characters and the particles and set different thresholds for weighting the particles. We employ a Gaussian kernel around each measured pixel to ascertain weights for resampling, utilizing the Systematic Resampling algorithm. Finally, we calculate the mean of the particles to determine the innovation in the measured pixels and particles.

D. Running the algorithm

The algorithm runs on Matlab using the standard toolboxes. In this case, as in most practical cases, different approaches can be chosen for initializing the particles. A naive approach would have been to apply a uniform distribution of the particles across the whole frame, wasting particles that are initialized in impossible locations. We decided to bind them instead to the area of the frame that corresponds to the maze. The final spread of the particles is a uniform distribution of 1000 particles across the maze.

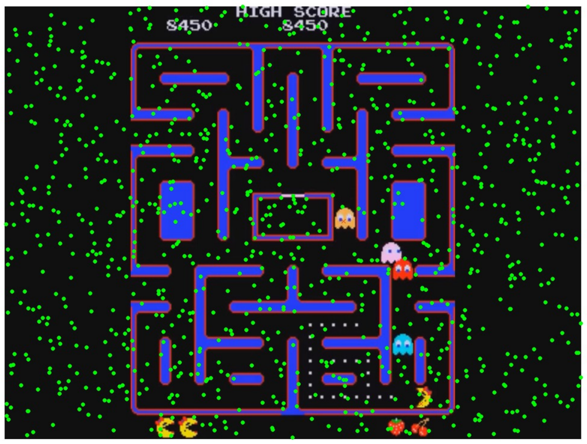


Fig. 9. Particle initialization - uniform distribution

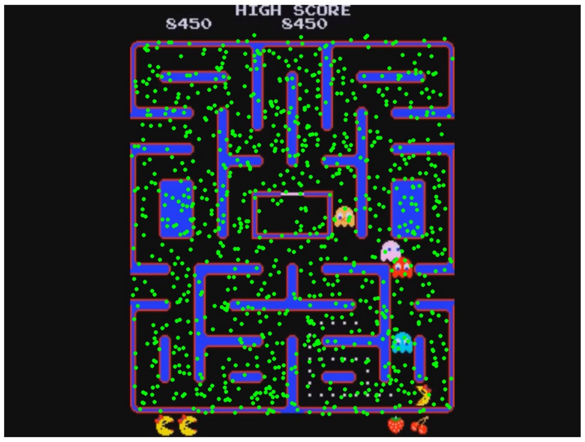


Fig. 10. Particle initialization - particles bounded to inside the maze

To test the algorithm we began tracking Ms. Pacman and the orange ghost (Clyde). With a small observation matrix, we can observe particle deprivation on the orange ghost, so we experimented with increasing the variance or increasing the number of particles. In Figure 3 we can see the particles converging to Pacman, but particle deprivation with Clyde. Setting M to 10,000 particles works to solve this issue but at the cost of computational time. The increase of computational time with the number of particles is a well-known problem for these algorithms, taking us around 5 minutes for each test to run. Instead, a larger observation model variance matrix was chosen.

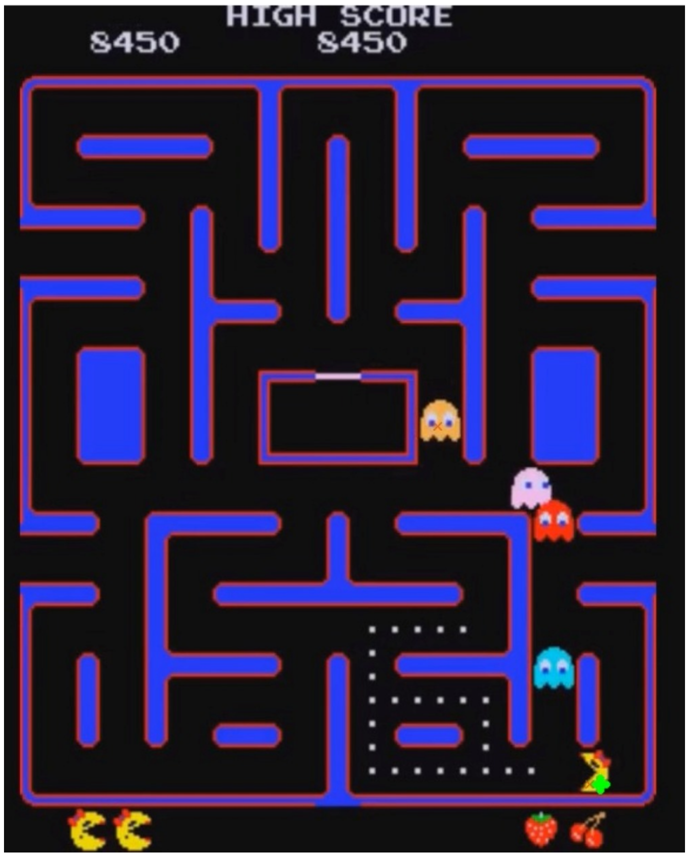


Fig. 11. Particle deprivation on Clyde

$$\Sigma_q = \begin{bmatrix} 10^2 & 0 \\ 0 & 10^2 \end{bmatrix}$$

Covariance matrix of observation model

IV. Results

A. The Ms. Pacman and Clyde problem

We discriminate between tracking two ghosts and tracking Ms. Pacman and Clyde, given that when trying to track both Ms. Pacman and Clyde, we can see the particles often switching positions between them. This could be explained by the color similarity between them. The experimental results can be seen here[12]. This implies that for the algorithm to have optimal results, the tracking objects must then be unique enough between them. Further experiments on different characters were then conducted.

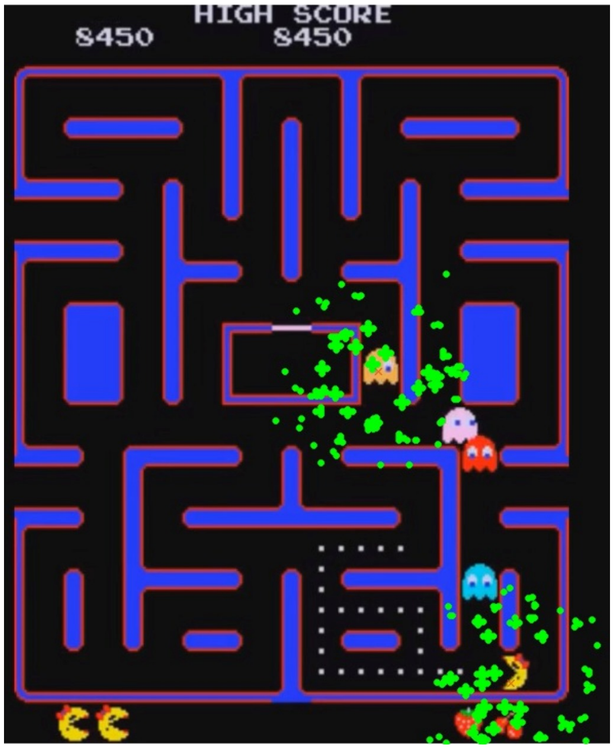


Fig. 12. Particles after a single iteration

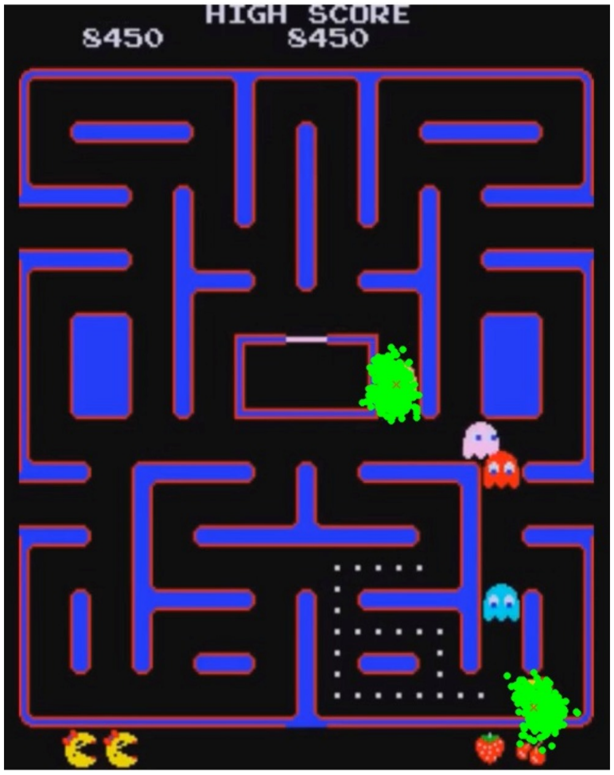


Fig. 13. Particles start to converge at the fourth iteration

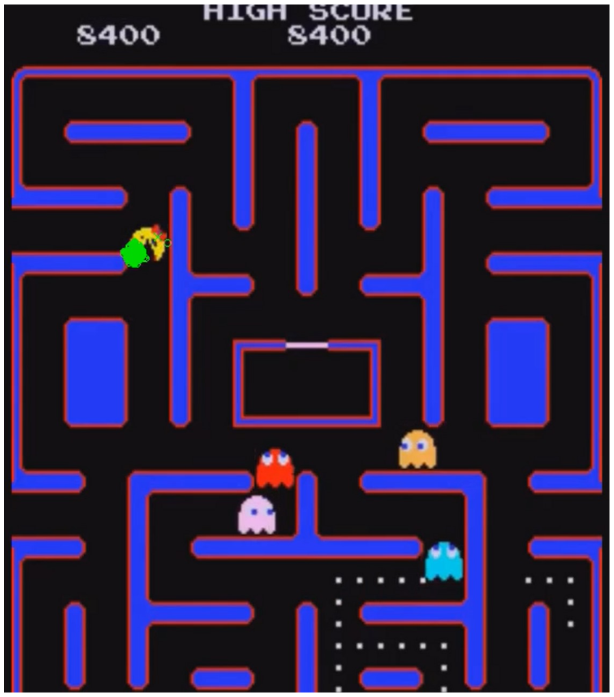


Fig. 14. We lost track of Clyde

B. *Tracking multiple characters*

A solution for this was tracking instead Ms. Pacman and a different ghost, Inky (the cyan one). In this case, we can see optimal results, shown here[13]. Given that the characters are unique enough between them, the algorithm has no problems tracking them simultaneously.

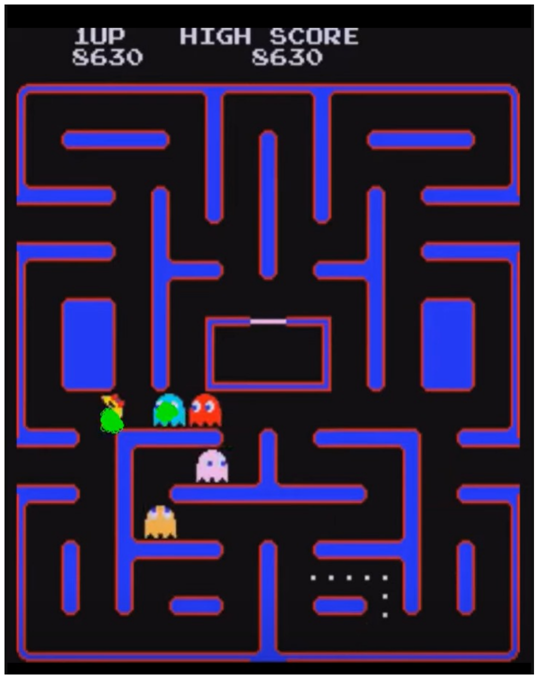


Fig. 15. Proper tracking of the two characters

Finally, we ran tests tracking only the ghosts. The motion model of Ms. Pacman and the ghosts is similar, but not the same, as Ms. Pacman has further liberties in her movement. We ran two tests on two different sets of ghosts to evaluate if the algorithm performs as well. For the first run we tracked both Inky and Pinky, the cyan and pink ghosts. The full run of the test is shown here [14]. We can observe proper tracking of both of the ghosts, without particle deprivation or errors on the tracking objects, given that the color information of the tracked characters is unique enough between both the background and themselves.

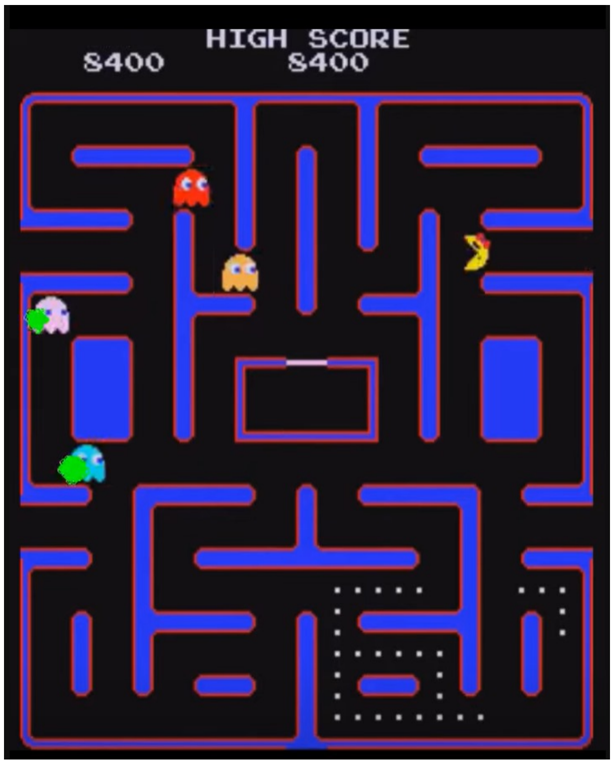


Fig. 16. Proper tracking of Inky and Pinky

For the second run, we tried tracking Blinky and Pinky, the red and pink ghosts. As stated in III, the red ghost brings an extra complication to the problem, as its color is not unique to him, but to the background as well. Running the algorithm we can see that given our measurement model, we can easily discriminate between the static red pixels and the dynamic pixels corresponding to Blinky, yielding optimal results to the tracking problem for this character as well. This showcases how the algorithm is robust enough to handle similarities between the background and the tracked characters. This second run is shown in [15].

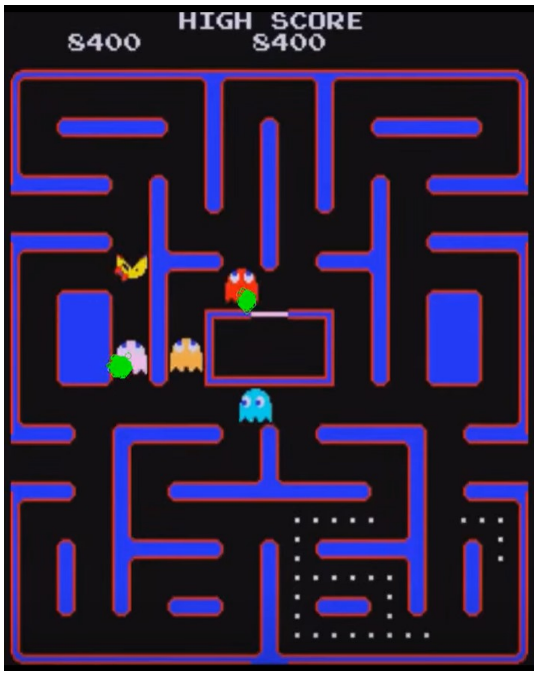


Fig. 17. Proper tracking of Blinky and Pinky

For a third and final run, we tried tracking three different ghosts, Inky, Blinky and Pinky. This final run demonstrates how the number of tracking objects can be arbitrary, as long as the tracked characters are unique enough between them. This final run is shown in [16]. Tracking all of the characters ended up bringing up the Clyde and Ms. Pacman problem, so proper discrimination between the two of them was left as further work.

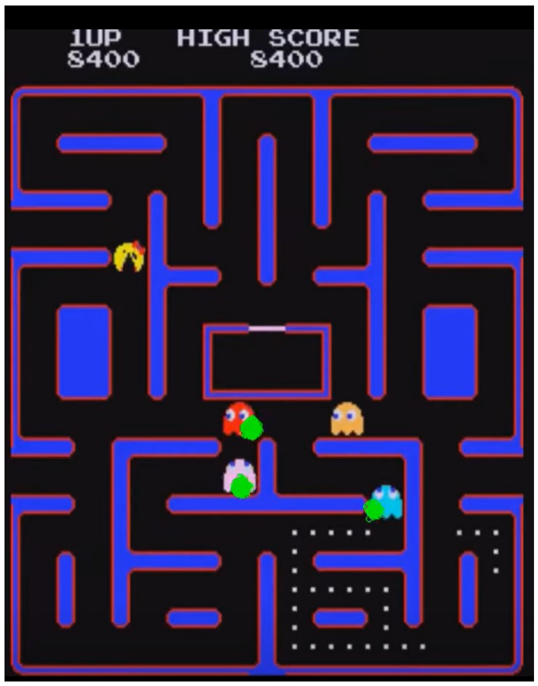


Fig. 18. Proper tracking of three different characters

C. Performance

Figure 18 displays a plot of the innovation of the particles vs time. The axes represent the mean absolute distance of the particles vs the number of iterations. That is the discrepancy between the predicted step and the measurement model (or the innovation). From the graph, taken from the run of the red and pink ghosts, we can conclude that the algorithm reaches convergence after 4 iterations. All of the runs yielded similar performances.

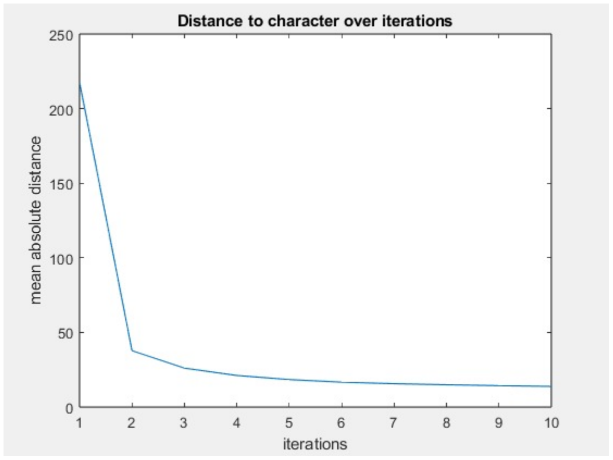


Fig. 19. Distance to character over iterations

D. The kidnapping problem

There's a special case during the Ms. Pacman gameplay where the characters can jump from one side of the maze to the other by walking through a tunnel. This is a good representation of the real-life problem of robot kidnapping, where the robot is taken from its original position to a new one, without giving that information to the system. In our case, we can see that the algorithm is robust to this problem because our measurement model takes into consideration the random nature of the motion and the incorporation of our measurements makes it so that it can quickly recover to an estimated position. As we do not define true convergence, we always keep the estimations as state probabilities, which recover quickly after the jumps. Nevertheless, we can imply the algorithm has converged after 4 iterations, as stated in IV-C This behavior is shown at the 4-second mark in [13].

V. Conclusions

The particle filter implemented during this project demonstrated good results on the desired task of tracking multiple characters of the Ms. Pacman game simultaneously. The tradeoff between the number of particles and computational time was addressed, as well as varying the values of the model parameters as an alternative solution. It was stated how the issue of inaccuracies in the motion model was addressed by compensating the observation motion model variance and how the final algorithm ended up being robust to the kidnapped robot scenario. It is shown that the measurement model is efficient at discriminating between the background pixels and the characters, and it is stated why it is important to choose characters that are different enough for optimal tracking, as shown in section IV, or compensate accordingly by modifying the measurement parameters. It is concluded then that the number of tracked characters depends mostly on the uniqueness of the parameters between them in the color space.

References

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. 2005. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press.
[2] A. K. Singh, "Major development under Gaussian filtering since unscented Kalman filter," in IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 5, pp. 1308-1325, September 2020, doi: 10.1109/JAS.2020.1003303.
[3] Katalin György, András Kelemen, László Dávid, Unscented Kalman Filters and Particle Filter Methods for Nonlinear State Estimation, Procedia Technology, Volume 12, 2014, Pages 65-74, ISSN 2212 0173, https://doi.org/10.1016/j.protcy.2013.12.457. (https://www.sciencedirect.com/science/article/pii/S2212017313006427)
[4] Hoteit, Ibrahim and Luo, Xiaodong and Pham, Dinh-Tuan and Moroz, Irene. (2010). Particle Kalman Filtering: A Nonlinear Framework for Ensemble Kalman Filters. AIP Conference Proceedings. 1281. 10.1063/1.3497823.
[5] Johannes S. Fischer, Localizing Tiago Robot with a Particle Filter in Python and ROS, January 5, 2022. https://johfischer.com/2022/01/05/localizing-tiago-robot-with-a-particle-filter-in-python-ros/
[6] ROSWiki, https://wiki.ros.org/Robots/TIAGo/Tutorials/Navigation/Localization
[7] Wang, K., Huang, Z. and Zhong, Z. Simultaneous multi-vehicle detection and tracking framework with pavement constraints based on machine learning and particle filter algorithm. Chin. J. Mech. Eng. 27, 1169–1177 (2014). https://doi.org/10.3901/CJME.2014.0707.118
[8] Lucas Rodes, Object Tracking: Particle Filter, https://www.youtube.com/watch?v=VhZmpS89QZQ
[9] Super Pikacho, (HD) Tracking Super Mario with a particle filter, SMB NES Level 1-1, https://www.youtube.com/watch?v=AGaWbOddxsE
[10] Student Dave, Multi Object Tracking Tutorial: part 2 by Student Dave, https://www.youtube.com/watch?v=lWXq12qCtkY
[11] Chris Morris (March 3, 2011). "Five Things You Never Knew About Pac-Man". www.cnbc.com. Archived from the original on October 15, 2012. Retrieved November 8, 2022.
[12] Aryan Dolas and Jesús Ortega, Particle Filter on Ms. Pacman and Clyde, https://youtube.com/shorts/lhYvMKVt3d8?feature=share
[13] Aryan Dolas and Jesús Ortega, Particle Filter on Ms. Pacman and Inky, https://youtube.com/shorts/GVfU03kPkcA?feature=share
[14] Aryan Dolas and Jesús Ortega, Particle Filter on Inky and Pinky, https://youtube.com/shorts/6O3LxDuh9uw?feature=share
[15] Aryan Dolas and Jesús Ortega, Particle Filter on Blinky and Pinky, https://youtube.com/shorts/cbNuHNZiitk?feature=share
[16] Aryan Dolas and Jesús Ortega, Tracking Inky, Blinky and Pinky, https://www.youtube.com/shorts/TfhoqogYl0E