Project Assignment - Jetbot

Samuel Nyberg[a,1] **and Tobias Mattsson**[a,2]

[a]*Blekinge Institute of Technology*

Abstract—This report will present an implementation of a simple PSO algorithm in combination with object detection to guide the movement of robots within a swarm. The report will cover the different methods applied during the project, different approaches attempted, limitations and solutions. The findings and discussion of the results will be presented at the end of the report.

Keywords—*PSO, Cluster, Robots, AI, YOLO, Optimization*

1. Introduction

The Particle Swarm Optimization algorithm (PSO) [5] is a simple yet effective algorithm that finds optimal solutions, much like birds flocking and fish swimming together. This method is useful to apply in situations where the problem cannot be solved with equations alone.

This project aims to implement a simplified version of PSO to steer multiple robots to a common target. In order for the robots to detect eachother, a object detection deeplearning algorithm will be applied.

The robots that will be used in the project are based on the NVIDIA JetBot AI Robot Kit. The robots have two wheels for output movments and a central camera as input.

2. Design and Methodology

2.1. Cv2 color range

One strategy to detect robots and the red target box was to utilize a variety of image processing tools, such as cv2 from OpenCV [1]. This allowed the server to isolate colors, looking for areas that have a larger area of a coherent color, for example, the color red for the target. This method, while simple to implement, had difficulties with detecting the other robots and only really worked with the target alone. The method of regular image processing, which involves searching for coherent areas by isolating color channels to detect positions of objects struggled greatly in different lighting conditions, making the development session dependent on the time of day. This created more problems then it solved thus it was quickly ruled out.

2.2. Jetbot Robot

During the project, robots based on NVIDIA JetBot AI Robot Kit were used for the swarm operation [4]. The robots consisted of two motorized wheels at the front, and a metal ball caster wheel at the back. The distance between the wheels were 10cm, which would be important during the movement calculations. At the top, the robot had a camera with a 160 FOV wide-lens. For processing, the robots are equipped with 128 GPU cores, 4 processing cores, and 2GB of RAM for efficient computation. To communicate with the server, the robots are equipped with a dual-band WiFi USB network card.

2.3. Swarm

The goal of the project is for the robots to be able to follow each other and the target in a strategy inspired by the PSO algorithm. The advantage in this scenario is that the robots does not rely on inputs from each-other. Each robot can navigate individually based on the algorithm and what is detected on the camera. Another advantage is that not all robots need to have a clear sight of the target, if a robot have obstacles between itself and the target, then it will follow the other robots until the target is within sight.

3. Implementation

3.1. Agent-Server Communication

The project works by having one central communication server and multiple robots. The robots, called agents, are set to send a video feed over an WIFI connection to the server. The server takes the video feed, processes it with the help of artificial intelligence as seen in figure: 3.2 and sends instructions back to the agents. To run multiple agents simultaneously, a multi-threaded server architecture was implemented. To be able to handle a potential upscale utilizing four or more agents. Communication can be split, allowing different agents to communicate with different servers. This allows for distribution the computational load since processing images with Artificial Intelligence can be resource-intensive.

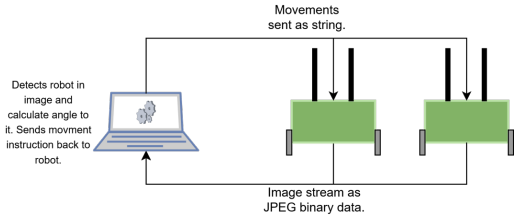


Figure 1. Agent-Server Communication

3.2. Object detection

To detect objects in the FOV of the Jetbots camera, an Artificial Intelligence algorithm named YOLO [3] was implemented, in this project the YOLO nano v8, which is one of the lightest and fastest YOLO architectures. YOLO stands for You Only Look Once and is an SSD algorithm (Single Shot Detector) developed for accurate and precise real-time object detection utilising CNN architecture (Convoluted Neural Network). Perfectly fitting for the task in this project.

The first step when implementing any Artificial Intelligence model is to gather data for training. Approximately 200 images were collected through the lens of the robot, placing it in front of the target and other robots. Following, the data was annotated through a web service named Roboflow [2], which can be seen in figure: 2 below. To get more data from the samples collected, a technique called data augmentation was applied. This allows for generating more data points to improve results and lower the chance of the model overfitting. Augmentation is a technique that duplicates an image, then changes certain things, such as flipping it, changing brightness or cropping, creating more instances which resulted in a total of 480 images.

Once the YOLO model was trained, it was implemented on the live video feed gathered from the Jetbot robots. Taking the video feed, predicting the boundary boxes for the different objects and also the class, that being the red target box or another robot.

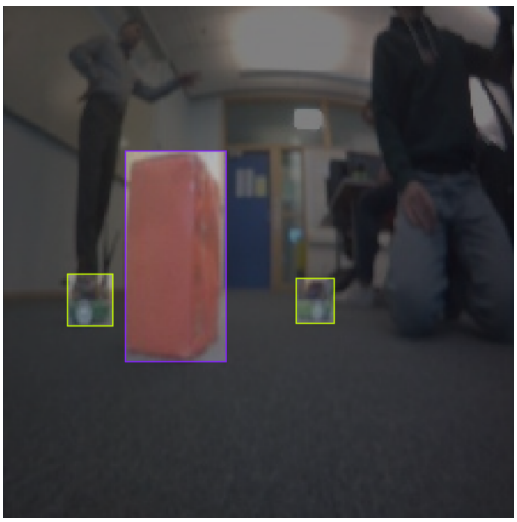
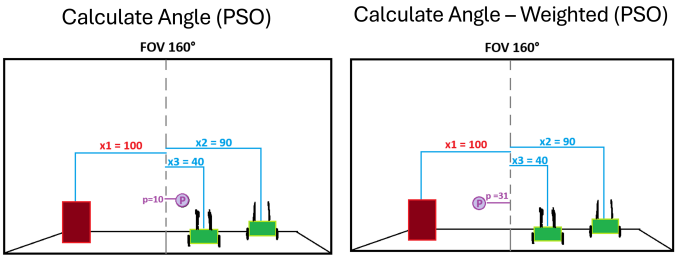


Figure 2. Annotated image, Purple box surrounds Target, yellow boxes surround Robots

3.3. PSO

3.3.1. Calculating Angle



(a) Calculate Angle (PSO) **(b)** Calculate Angle - Weighted (PSO)

Figure 3

Initially, when calculating the moving angle, only mean angle of both the target and the robots was used, figure: 3a. It was calculated using the simple formula:

$$\overline{\theta} = \frac{1}{n}\sum_{i=1}^{n}\theta_i$$

The issue with this approach is that when multiple robots are present, the calculated mean angle tends to align more with the robots positions than with the intended target. To address this, a weighted bias was added to the target to correct the alignment. The formula for the weighted mean is:

$$\overline{\theta_{\text{weighted}}} = w \cdot \theta_{\text{target}} + (1-w) \cdot \frac{1}{n}\sum_{i=1}^{n}\theta_i$$

With the modified weighted mean, the resulting average angle is closer to the target, which is beneficial figure: 3b.

3.3.2. Robot Wheel Speed

To calculate the wheel speed that the server would send as an instruction to the robot, a set of equations was used. To have consistent, smooth movement of the robot a constant speed was set at 20% of maximum velocity. Once the target angle was obtained from 3.3.1 the equation was used to turn right and vice versa to turn left:

$$V_L = V_{\max}$$
$$V_R = V_{max} * \frac{1-(\theta * 0.1)}{d_{wheelbase}}$$

If the obtained angel was less then 5 degrees in either direction. The robot would receive instructions to drive straight, running both wheels at equal speed. This was done to reduce jittery movement.

4. Results and Discussion

(a) Robots in the early stage

(b) Robots try to locate the goal

(c) Robots found the goal

Figure 4. Example figure that covers timeline of PSO algorithm **PFGPlots**.

Results show a promising implementation of a PSO algorithm using multiple agents and servers. It can be seen in figure figure: 4a that not all robots initially see the target. Instead, follow other agents as seen in figure figure: 4b until the target can be localized and reached seen in figure figure: 4c.

4.1. Limitations and Solutions

Throughout this project there was some limitations found that will be presented and discussed below.

4.1.1. Only one Camera

The Jetbot robots are built to come with only one camera. This became the only way to get information of the environment surrounding the robot since no other sensors were present. Having only one camera and nothing else meant that it became difficult to accurately calculate and predict distances between objects. Thus making navigation far more difficult and the results could have been greatly improved if one more camera had been installed to the Jetbot. The optimal solution would have been to allow the installation of some type of LIDAR/UWB sensor to allow for better depth perception.

The instance where this was the most notable was when calculating when to stop the robot. To solve this the pixel with of the boundary box retrieved from the YOLO model was used as a estimate on when the robot got to close to either to target or other robots.

4.1.2. Computing Resources

Jetbots ran older versions of Ubuntu and Python which greatly limited development since we could not execute advanced code directly on the platform, such as AI. Instead having to send the video feed to an external server where processing would be ran. This works fine when only one or two robots are being utilized, but when scaling the project to four robots the limitation quickly became noticeable. The solution found to temporarily patch this problem was to run on two different laptops, two robots each. This works since the robots in our

specific case do not communicate between each other where as in a complete implementation of the PSO algorithm they would have to.

4.1.3. Imprecise Controller

Another limitation found was that the controllers, such as drive units were imprecise. This meant that when executing the same set of instructions multiple times yielded in different outcomes. Thus forcing a solution that would not depend on precise movements, which have both pros and cons.

4.1.4. Difficult Environment

The environment in which the robots were run differed between data sampling, development and final demo. The environment was cramped, limiting us as a development team to properly test our solutions due to other students sitting in the same area, moving around. Since the environment changed multiple times, this also made lightning conditions and surrounding obstacles to constantly change. Mostly a challenge for the YOLO model.

5. Conclusion

This project has shown that a simple implementation of PSO can be implemented and successfully scaled on even limited systems such as the Jetbot Nano. It proves that PSO is a effective algorithm when trying to locate and catch a moving target in a multi agent environment. During the development of this project, several limitations were encountered. However, they led to the implementation of many interesting and innovative solutions. Overall, this project has been a success with a working algorithm with possibility to scale with more robots as long as there is computing resources available.

References

[1] OpenCV, *Opencv: Open source computer vision library*, OpenCV: Open Source Computer Vision Library, 2025. [Online]. Available: https://opencv.org/.

[2] Roboflow, *Roboflow: Organize, label, and deploy computer vision datasets*, Roboflow: Organize, Label, and Deploy Computer Vision Datasets, 2025. [Online]. Available: https://roboflow.com/.

[3] Ultralytics, *Ultralytics documentation: Yolo models, training, and deployment*, Ultralytics Documentation: YOLO Models, Training, and Deployment, 2025. [Online]. Available: https://docs.ultralytics.com/.

[4] Waveshare, *Jetbot 2gb ai kit*, JetBot 2GB AI Kit - Waveshare, 2025. [Online]. Available: https://www.waveshare.com/jetbot-2gb-ai-kit.htm.

[5] Wikipedia, *Particle swarm optimization – wikipedia, the free encyclopedia*, Particle Swarm Optimization – Wikipedia, The Free Encyclopedia, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Particle_swarm_optimization.