Coordinated Searching and Tracking of Moving Target using Multi-Agent System

A report submitted for BTP phase II

by

Sankeerth Reddy Prodduturi

(Roll No. 180108034)

Under the guidance of

Dr. Indrani Kar

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

November 2020

Abstract

Exploring an unknown environment using a single mobile agent has always been difficult and time taking. Searching and tracking the same environment using Multi-Agent System can be very fast and efficient. This can give rise for real-time applications. This report aims to describe how multi-agent systems can be used for applications such as searching an unknown environment, tracking a mobile target, target pursuit with emphasis on implementation.

Contents

List of Figures

Chapter 1

Introduction

Multi-Agent systems have been able to do much complicated tasks which are very difficult for a single agent system to do. In recent times, control of multiple robots has received significant interest. Multiple agents with the help of communication acting as a single unit are known as a multi-agent system. Searching and tracking are some of the applications where multi-agent systems can be beneficial. Searching an unknown environment has always posed threats and difficulties for humans. With multiple agents coordinating from different directions tracking a target can be more efficient. With an increased range and decreased latency of communication devices, we can achieve real-time coordination between the agents. When combined with obstacle avoidance and mapping capabilities, these agents can plan their path easily, even in a cluttered environment. These can be done when the agents know about the environment surrounding them. Computer vision provides us with inexpensive tools to estimate and localize objects in the field of view of the robot. With reinforcement learning now being used to develop the robot's decision-making process, they can make very complex decisions with excellent accuracy. Implementing searching and tracking using computer vision, target pursuit, communication and coordination between multiple agents is the aim for this project. Target pursuit and tracking were implemented in phase-1. Multi agent consensus and obstacle avoidance are implemented in phase-2.

1.1 Literature Survey

Cooperative hunting algorithm discussed in [1] uses agent in different modes to search, pursue and predict the target which can cope up with the irregularities arising due to communication delay or loss, wheel slippage and odometry errors, but with an assumption that the agents are omnidirectional with a 360° field of view. The agents search the area independently but coordinate when the target is identified. It also discusses escape strategies for the target. [2] concentrates more on the multi agent tracking algorithm dividing the area into target and clutter and then estimating the target's position using a kalman filter, which can form formations and maintain them. [3] explores the idea of leader-follower framework with fixed topology and a directed spanning tree. Agents are modelled with two control layers in [4] with one being hybrid layer which controls agent internally and other being intelligent coordination control layer which communicates with the rest of the agents and plans the path for the agent. Computer vision literature has many algorithms already developed for tracking, one such is Kernelized Correlation Filter(KCF) tracker [5] which tries to calculate the object's position by estimating the direction of motion, it tries to find the similar set of points as that of the object's, and moves the bounding box. It is very fast but comparatively less accurate. Channel and Spacial Reliability Tracker(CSRT) [6] works by training a classifier, which is used to re-detect the object and correct tracking errors. This tracker works by learning the texture, color, shape and surroundings of an object as they change over time. It is bit more accurate than KCF but slower and has high false-positive tracking rate.

Chapter 2

Problem Formulation

2.1 Statement

Considering a cluttered environment with a mobile target whose position is not known to the agents. The target and the agents are assumed to be a differential drive unmanned ground vehicle with a camera and a distance sensor like lidar,radar etc. Agents start at a random starting position and start searching for the target in different directions. When an agent of the system detects a target, all the other agents in the network know the position of the target and start to pursue the target. When the agents reach near the target, both the agents and target stop and we reach an end state.

Here we try to find solution for multi agent consensus assuming the consensus position as the average position of the agents. Then implementing obstacle avoidance to reach a stationary goal.

2.2 Preliminaries

The simulation and analysis are done using Robot Operating System(ROS) and Gazebo. The controllers and obstacle detection programs are written in Python using open-source packages. Understanding of ROS and Gazebo is required for the simulations and analysis. Skid-steer drive controller plugin from ROS is used as the lower level control for the agents. Python packages like Rospy and Numpy are used to write the program for the agents. Controller and target tracking have been implemented in phase-1 and are assumed to be known. Agent and Rover are used interchangeably to describe the robot.

Chapter 3

Simulation and Results

3.1 Multi agent simulation

3.1.1 Simulation Setup

Leo Rover (Fig. 3.1) which is available open source [8] is used to simulate and analyse the results. 3 rovers are setup using name-spaces with individual controllers. All the agents communicate to through single rosmaster as a leader-follower network. The leader being the rosmaster and the rovers as followers. As all the simulations are done on a single system, the communication is done through internal rostopics.

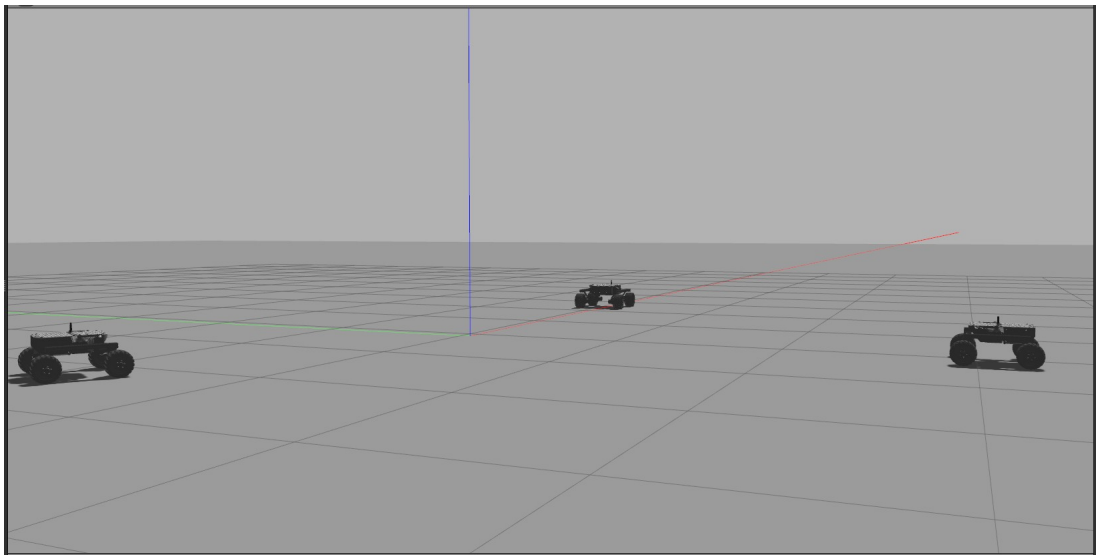

Figure 3.1: Multi agent system in Gazebo

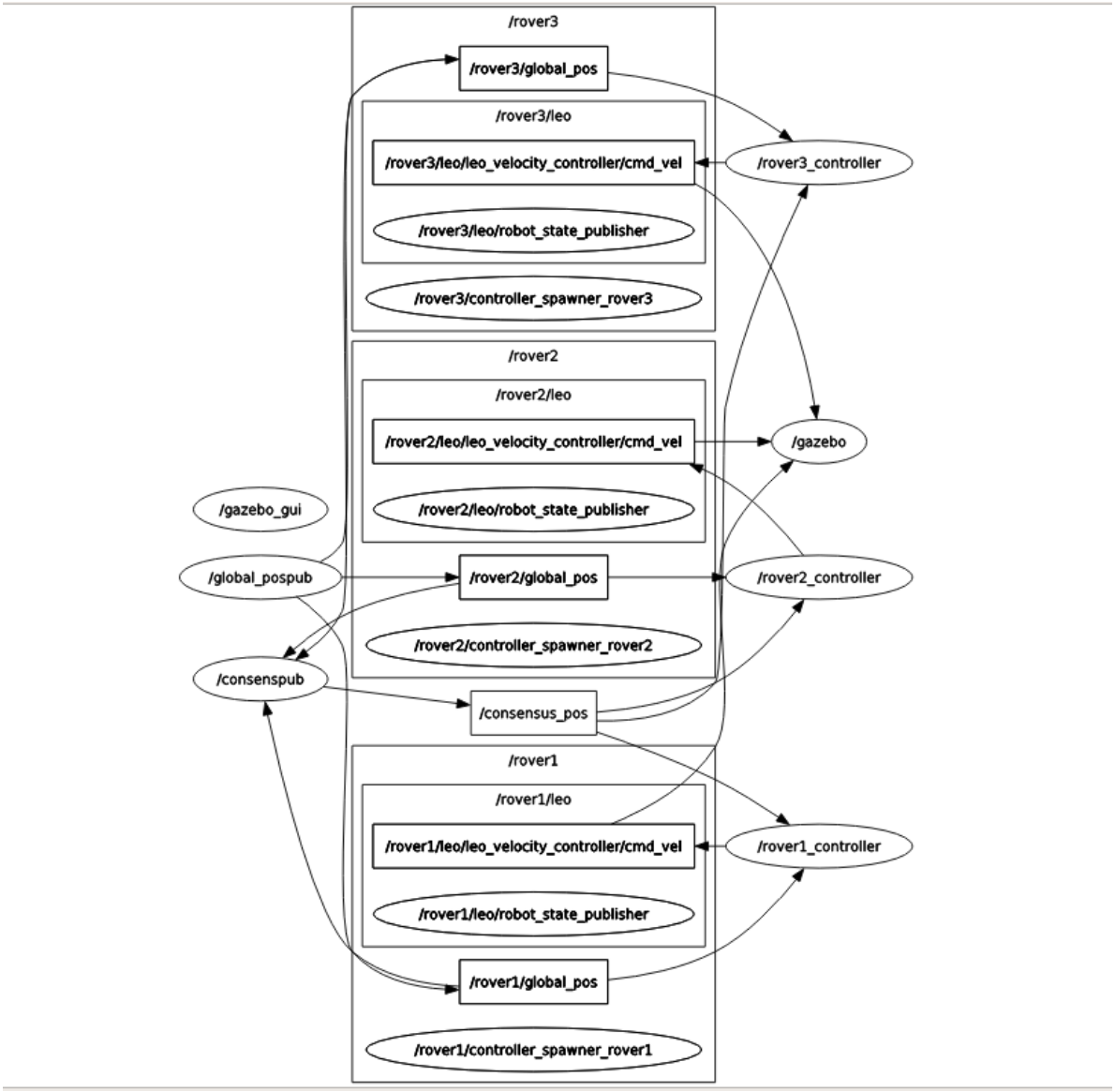

Figure 3.2: RQT graph of the system

Every agent has the knowledge of it's local position through odometry, but it is required to have a global position to let the agents know about the consensus position and track them. For this the model state from the gazebo simulator is used as the global position of the agents.

3.1.2 Consensus Position

Agents are first to reach a consensus position which is the average position of the agents at any given time. Every agent is given the consensus position as a target similar to the moving target that in phase-1. When agents reach within a radius of $1\,\mathrm{m}$ they stop. When all the agents stop we can conclude that the system has reached consensus.

3.1.3 Controller

The controller(Fig. 3.3) is used to navigate the agent to a given goal position. Distance error is the euclidean distance between the rover and the target, the yaw error is the required angle the rover needs to rotate to adjust towards the new target position. When agent reaches the threshold error range, both the agent and target stop indicating the end state.

Every agent is controlled individually without just consensus position and it's global position

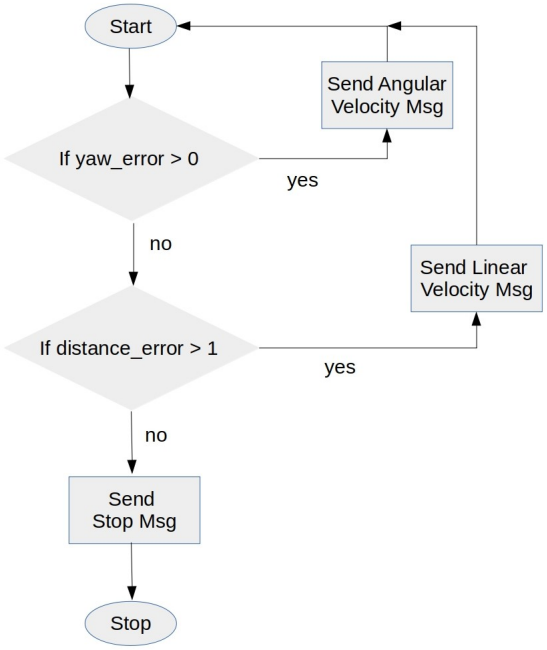

Figure 3.3: Flowchart of the controller

as the feedback. Any agent does not have the knowledge of the position of other agents. So this is a centralized multi-agent system.

3.1.4 Results

The agents are simulated to reach the consensus in 2 different conditions. One where the agents form a equilateral triangle with equal velocities, other where the speed of one of the agents is less compared to the others. The results are as follows.

For the above case(Fig. 3.4) we see the agents travelling almost identical distances to reach the consensus point. As the velocities are equal there isn't much of a variation in the consensus position. We can also observe that the agents stopped within a radius of $1\,\mathrm{m}$.

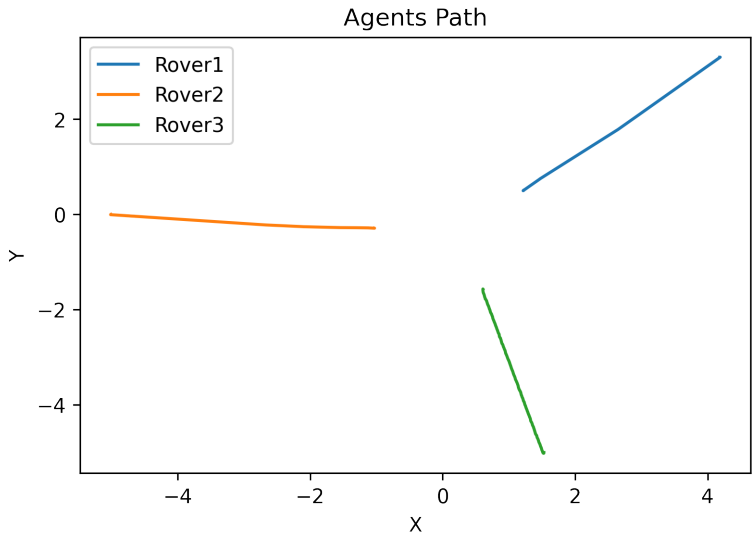

Figure 3.4: Paths when velocities are equal

For the unequal velocities case(Fig. 3.5) we see the rover3 travels less than other rovers and the agents reach consensus in nearly the same time.

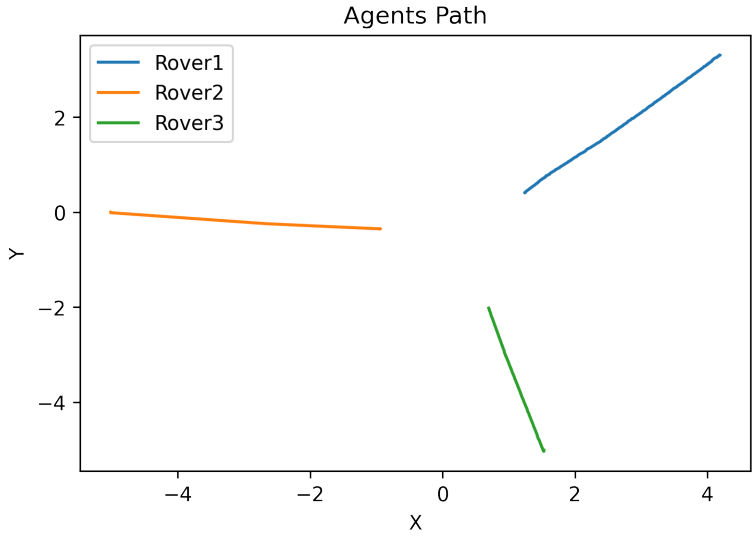

Figure 3.5: Paths when rover3 is slow

3.2 Obstacle Avoidance

3.2.1 Simulation setup

The environment considered consists of obstacle of different kinds so the agents must be equipped with algorithms which avoid obstacles and plan the path accordingly. For this a single agent equipped with a depth camera is used. The agent starts with a static goal to reach. The controller has two different modes. One mode avoids obstacles and other is the controller discussed above. When the agents comes within a given threshold of an obstacle it changes from the guided mode to avoidance mode. After completely avoiding the obstacle then the agent switches back to the guided mode.

3.2.2 Algorithm

The algorithm takes the weighted average of the depth given for every pixel and estimates the direction in which the agent has to move. Assuming origin at the centre of the depth image we calculate the weighted average of the x coordinate of the obstacle. Then a yaw message is sent to the agent proportional to the calculated value of the x_m. This way according to the weight function $f(d)$ the agent avoids obstacle when they appear in the field of view.

$$x_m = \frac{\sum x f(d)}{\sum f(d)} \tag{3.1}$$

The equation(3.1) is used to calculate the weighted obstacle position on the x-axis and then a yaw message is published in opposite direction of the x_m. When the agent is clear obstacle it switches to guided mode and reaches the destination.

3.2.3 Results

The depth camera had a range of $3\,\mathrm{m}$ and a FoV of 70°. The agent had to avoid two square blocks and stay well within the walls to reach the goal. The agent is given an endpoint which is at rest.

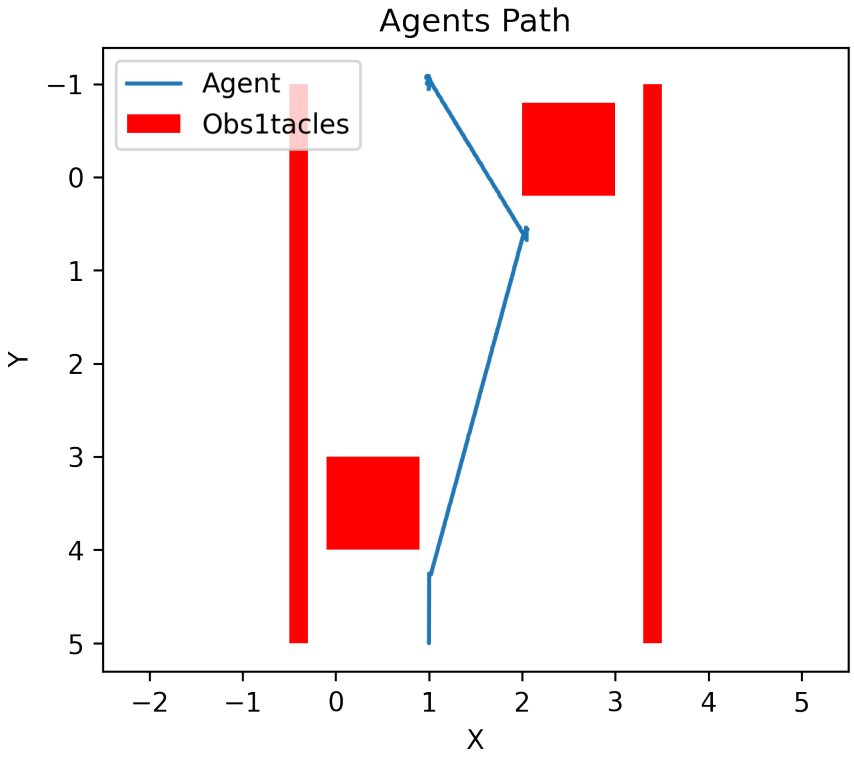

Figure 3.6: Path avoiding obstacles

The above figure(Fig. 3.6) indicates the path followed by the agent. The agent swiftly avoids the obstacles before reaching too close to them and reaches the target in near straight line path.

Chapter 4

Conclusions and Future Work

4.1 Conclusions

We have seen that the agents can reach a consensus state from any given situation. We have also seen that agent can swiftly avoid the obstacles and move accordingly. The consensus position which has been taken as the average position can be changed according to the situation and can be achieved. The obstacle avoidance algorithm is fairly simple and effective but can be improved upon using mapping and localising the environment.

4.2 Future Work

This project now has 4 completed modules. Target pursuit, Target tracking, Multi agent consensus and obstacle avoidance. Combining all these 4 modules with mapping and path planning algorithms will help us achieve the goal of coordinated search and track. Hardware implementation requires other things like the global position and odometry of the agents which were provided by the simulator. Multi agent localisation can be done using Aruco markers on each agent and a network of cameras estimating the position of the agents. So the future works remains in the hardware implementation and fusion of all the modules.

Bibliography

[1] Zhiqiang Cao, Min Tan, Lei Li, Nong Gu and Shuo Wang, “Cooperative hunting by distributed mobile robots based on local interaction,” in IEEE Transactions on Robotics, vol. 22, no. 2, pp. 402-406, April 2006, doi: 10.1109/TRO.2006.862495.

[2] X. Ru, W. Liu and Z. Tian, ”“ Target Dynamic Tracking and Hunting Based on Multi- Agent systems Control,” 2019 International Conference on Control, Automation and Information Sciences (ICCAIS), Chengdu, China, 2019, pp. 1-6, doi: 10.1109/ICCAIS46528.2019.9074569.

[3] G. Wen, Z. Duan, G. Chen and W. Yu, “Consensus Tracking of Multi-Agent Systems With Lipschitz-Type Node Dynamics and Switching Topologies,” in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 61, no. 2, pp. 499-511, Feb. 2014, doi: 10.1109/TCSI.2013.2268091.

[4] F. Karray, O. Basir, I. Song and H. Li, “A framework for coordinated control of multi-agent systems,” Proceedings of the 2004 IEEE International Symposium on Intelligent Control, 2004., Taipei, 2004, pp. 156-161, doi: 10.1109/ISIC.2004.1387675.

[5] KCF tracker Documentation - “https://docs.opencv.org/3.4/d2/dff/classcv_1_1TrackerKCF.html”

[6] CSRT Documentation - “https://docs.opencv.org/3.4/d2/da2/classcv_1_1TrackerCSRT.html”

[7] YOLOv3 Paper - “https://pjreddie.com/media/files/papers/YOLOv3.pdf”

[8] Leo Rover Source - “https://github.com/PUT-UGV-Team/leo-vanilla”