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

A report submitted for BTP phase I

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

Nomenclature

ROS	Robot Operating System
UGV	Unmanned Ground Vehicle
RL	Reinforcement Learning
CV	Computer Vision
API	Application Programming Interface
FoV	Field of View
KCF	Kernelized Correlation Filters
CSRT	Channel and Spatial Reliability Tracker
YOLO	You Only Look Once

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 have been implemented.

Chapter 2

Literature

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 3

Problem 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. The problem statement can be divided into three modules.

1. Searching and Tracking
2. Target Pursuit
3. Multi Agent Communication

Here we try to find solution for target pursuit assuming the position of target being known and tracking using computer vision, implementing them using open source tools and libraries.

Chapter 4

Simulation and Results

4.1 Target Pursuit

4.1.1 Controller

Leo Rover (Fig. 4.1) which is available open source [8] is used to simulate and analyse the results. The rover's speed has been set to $0.4\,\mathrm{m/s}$ for the analysis and observe the results for varying target speeds. Initially rover and target are at a distance of $10\,\mathrm{m}$ apart. We assume that the target location known and updated continuously. Rover takes the target position as its input and adjusts its linear velocity and yaw accordingly. The rover is controlled using linear velocity in x-direction and angular velocity along z-axis.

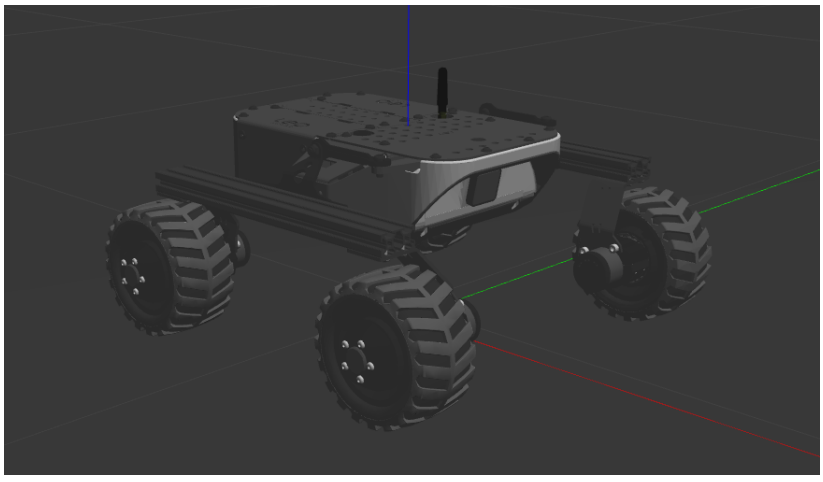


Figure 4.1: Leo Rover in Gazebo

Distance error and yaw error are used to define the controller variables (Fig. 4.2). 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.

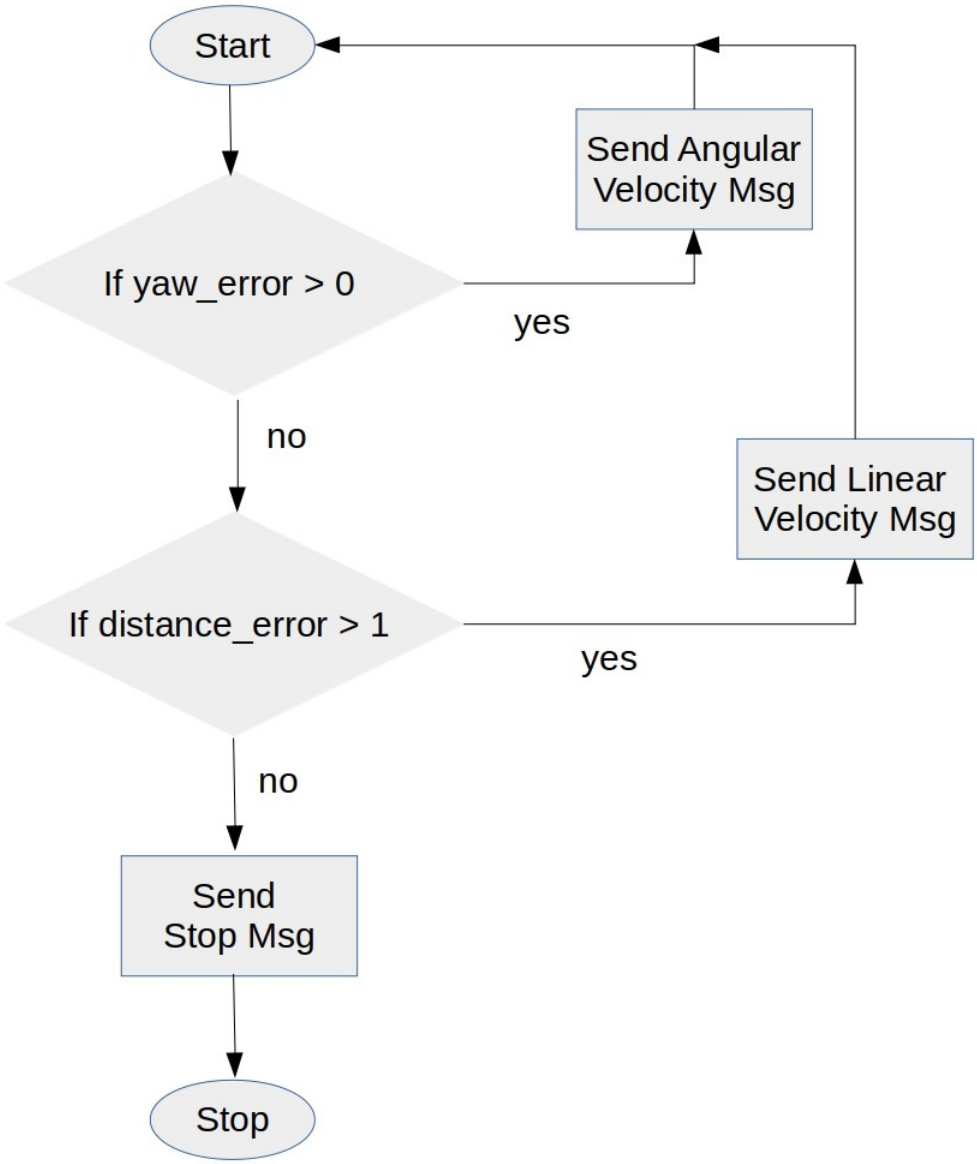


Figure 4.2: Flowchart of the controller

4.1.2 Results

Simulations have been done using ROS and Gazebo with python to test the controller with target at rest, Target moving in straight line with 50% and 80% velocity of that of the agent. When target is at rest, rover adjusts its yaw once and then moves towards the target in a straight line (Fig. 4.3) with constant velocity(Fig. 4.4).

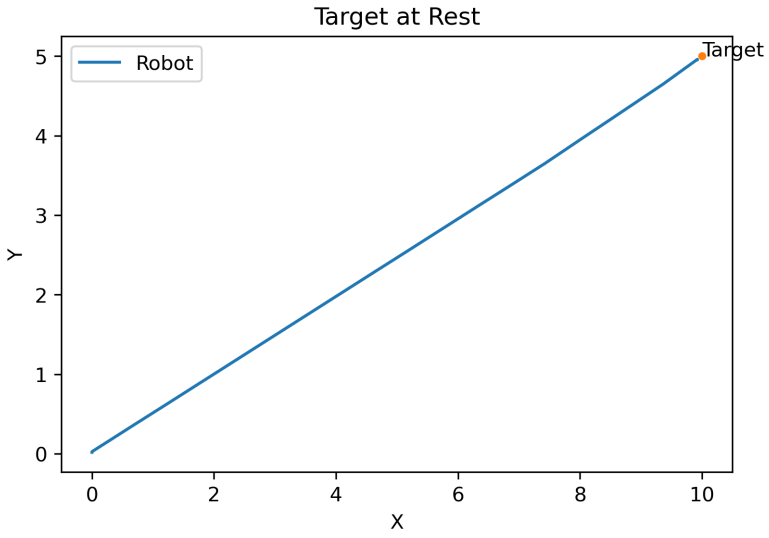


Figure 4.3: Path for Target at Rest

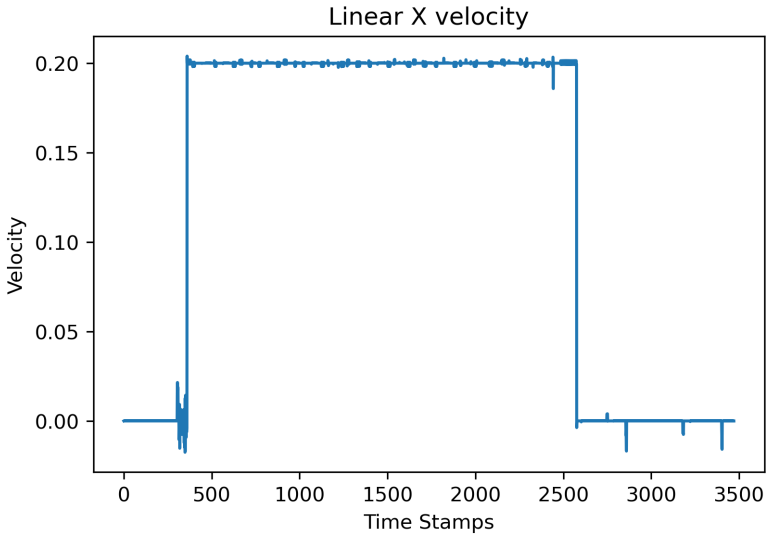


Figure 4.4: Velocity graph for Target at Rest

When the target was moving in a straight line with 50% velocity of agent, path for agent was curved (Fig. 4.5), with linear velocity almost constant (Fig. 4.6) and yaw adjusting rapidly (Fig. 4.7) and reached the target after it moved for around $5\,\mathrm{m}$.

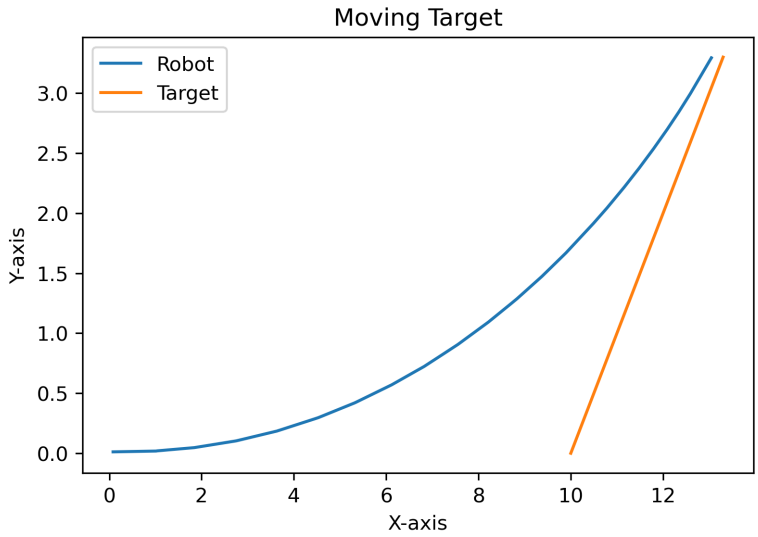


Figure 4.5: Path for Slower Target

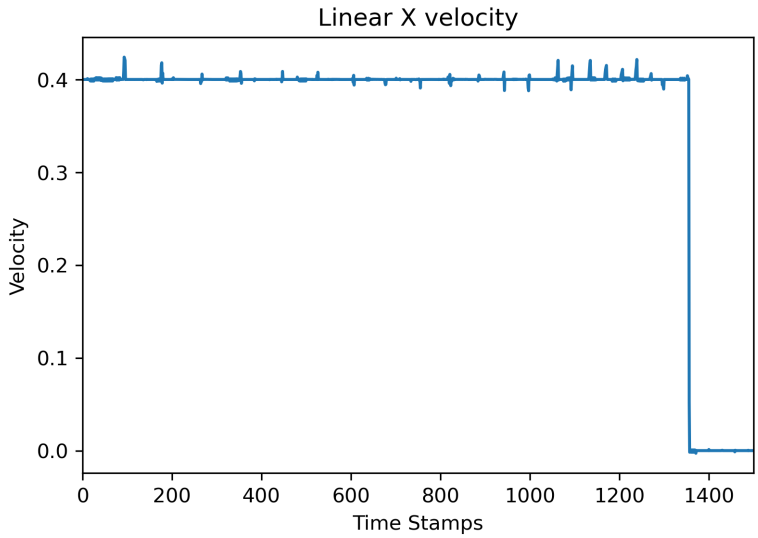


Figure 4.6: Velocity graph for Slower Target

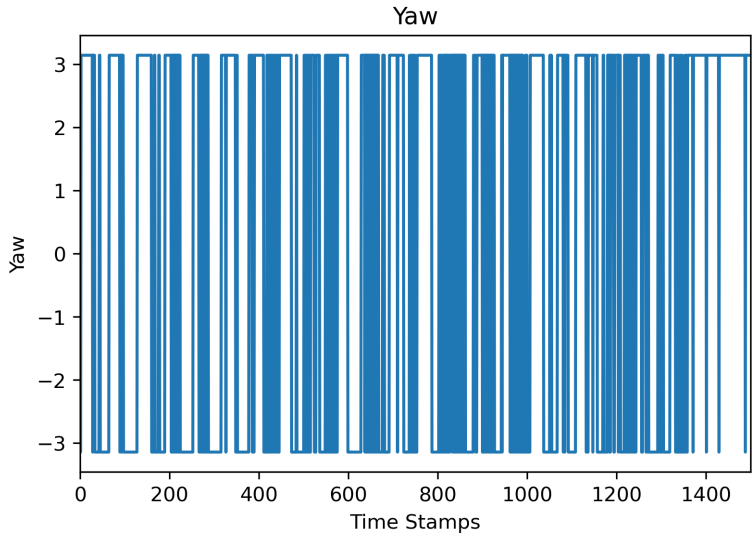


Figure 4.7: Yaw graph for Slower Target

When the Target was moving with 80% velocity of the agent, path for the agent was more curved with some linearity (Fig. 4.8) and with similar velocity (Fig. 4.9) and yaw (Fig. 4.10) changes as of previous case. It reached the target after it moved around for $20\,\mathrm{m}$.

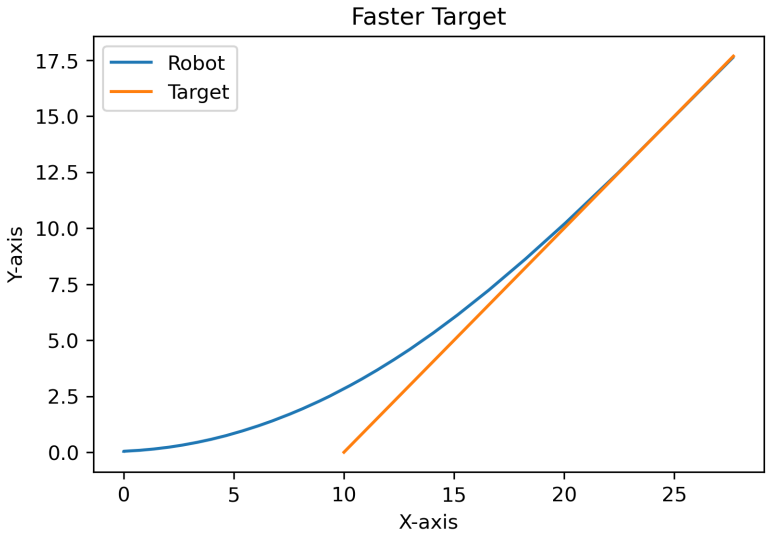


Figure 4.8: Path for Faster Target

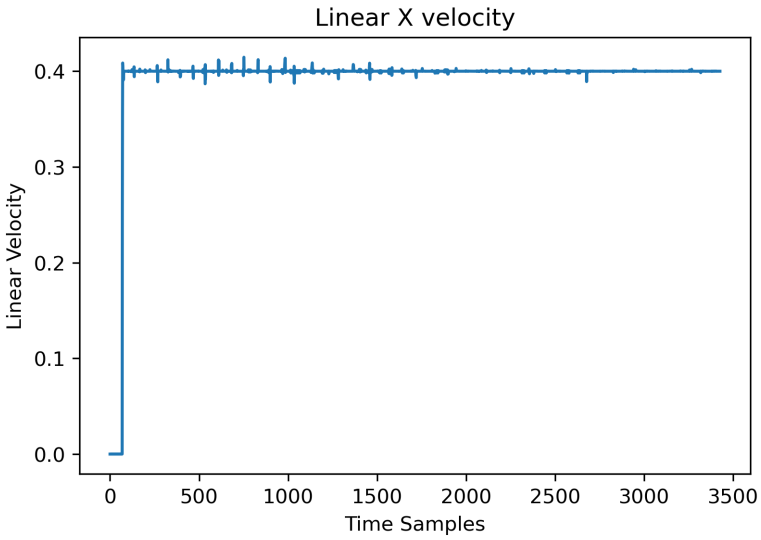


Figure 4.9: Velocity graph for Faster Target

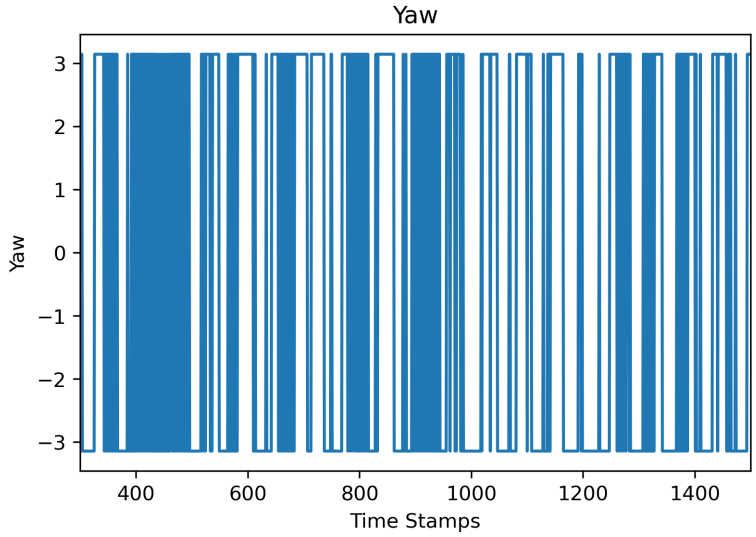


Figure 4.10: Yaw graph for Faster Target

4.2 Target Tracking

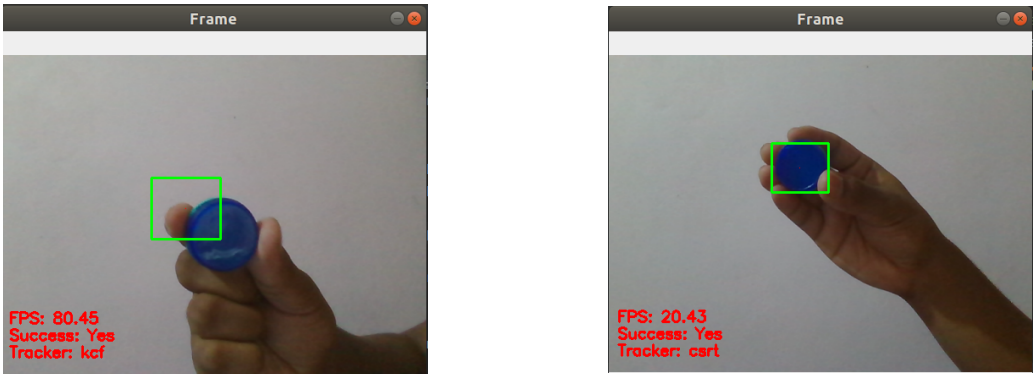
4.2.1 Computer vision algorithms

Above section on target pursuit was built upon the idea that position of target is known. But it is difficult to estimate the position of target without using expensive sensors like laser scanner and motion capture systems. However using camera along with another distance measuring sensor like lidar,depth sensors we can find the position of target with good accuracy. Computer vision is

very useful in solving these problems. Using OpenCV - an open source CV library, we try to find a solution for target tracking using its Tracker API. We mainly use two trackers KCF and CSRT. Kernelized Correlation Filters(KCF) is mainly used when speed is main requirement with a little low accuracy. Main problem with this tracker is it does not recover from full occlusion, that is when the target gets obstructed and again reappears in the FoV, the tracker will not be able to track the target. KCF is also good at reporting failure if object is not detected. Channel and Spatial Reliability Tracker(CSRT) is more accurate but slower than the KCF tracker, but it does not report failure and tracks similar objects if target is missed.

4.2.2 Results

The above mentioned trackers are implemented using OpenCV-python with a laptop's 720p webcam. As we can see from (Fig. 4.11a) KCF tracker gives more than 80fps thorough-put but being less accurate, whereas CSRT gives more accurate results (Fig. 4.11b) with just 20fps thorough-put.

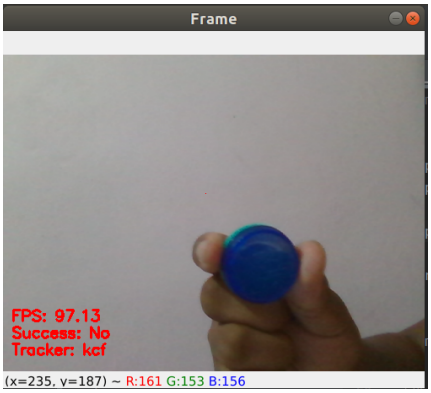


(a) KCF Success

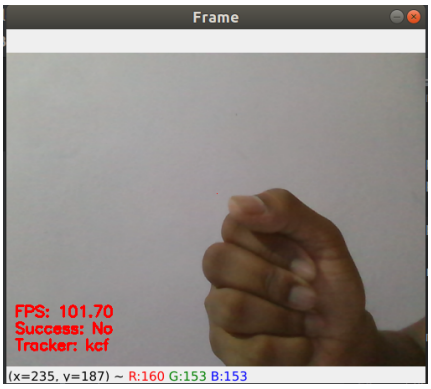
(b) CSRT Success

Figure 4.11: Success cases for two Trackers

From (Fig. 4.12) we can see that KCF tracker reports more negatives, not detecting even when the object is present (Fig. 4.12a) and reporting negative for object not present (Fig. 4.12b). CSRT tracker does not report negatives (Fig. 4.13) as much as KCF. It has high false-positive rate both when object is present (Fig. 4.13a) and when object is absent (Fig. 4.13b). CSRT fails to report no success most of the time.

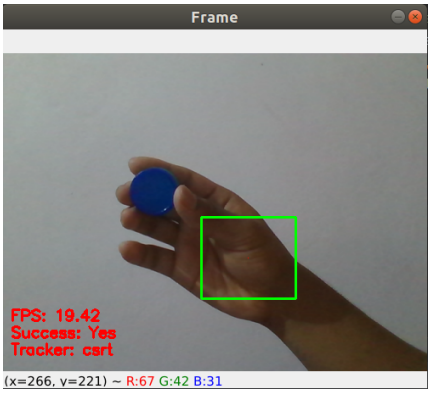


(a) False Negative

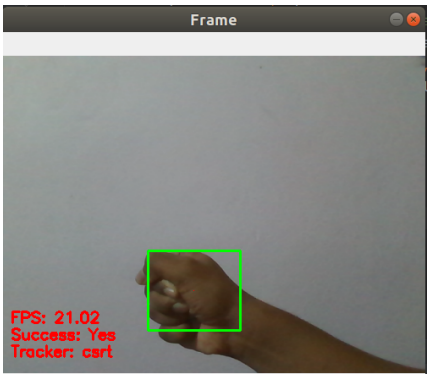


(b) True Negative

Figure 4.12: KCF Trackers Negative's



(a) Not detecting Object



(b) Detecting False Object

Figure 4.13: CSRT false-positives

The results conclude that the above mentioned limitations of the trackers are true and they need to be improved upon. As these trackers are trained online,means they require very few frames to learn about the target. We can train our custom model for our target building upon the ideas of these trackers or other real-time object detection algorithms like YOLOv3 [7].

Chapter 5

Conclusions and Future Work

5.1 Conclusions

We have observed that robot can pursue the target and reach it even when the target has speed just slightly less than to that of robot. Linear velocity of the rover is almost constant throughout the motion which gives us smooth transitions between different positions. We have also seen the efficiency and accuracy of OpenCV tracking algorithms, which needs to be improved and modified for our custom requirements.

5.2 Future Work

Future work for this project mainly lies in the coordination and communication between multiple agents. Then improving the controller to include different modes such as search,track,pursuit similar to what has been done in [1]. Then we need to improve upon the tracking algorithms and need to explore upon the ideas of real-time object detection like YOLOv3 [7]. Applying an estimating algorithm using the data from camera and sensor, localizing the objects relative position. Then if possible train a reinforcement learning model to effectively take decisions for the agents.

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”