Technical Report, DJI Robomaster AI Challenge, ICRA, 2022, Team ERA, IIT Kanpur

Ashok Kumar Chaudhary, Luvneesh Kumar, Ayush Ranjan, Sidhartha Watsa, Aaryen Mehta,
Tejas Chikoti, Aditya Jain, Nishi Mehta, Yukkta Seelam, Shubham Mittal, Sachin Bhadang, Ashish Kumar, L. Behera

***Abstract*— In this report, we describe the progress we have made so far in the DJI Robomaster AI Challenge which will be held at ICRA, 2022. In particular, we talk about the overall system design from hardware and software perspective.We describe the hardware infrastructure which we have used to cater the requirements of the challenge. In the hardware description, we discuss different kinds of sensors, computing devices and network communication. In addition, we also describe the key algorithmic components of our system which is the most crucial component in the system design. In the algorithms, we primarily discuss about the visual perception modules necessary for enemy robot detection, self-localization, navigation and planning to maximize the likelihood of winning and many more. All the algorithms mentioned in the report have been when tested and are providing great results.**

I. Introduction

The problem statement of Robomaster AI Challenge is to build an AI powered software pipeline for a given robotic platform. The key concepts involved in this pipeline are localization of the ego in the static arena, mapping, dynamic obstacle avoidance, detecting and shooting the targets, path planning and decision making. The sensors installed on the robots form a part of visual perception system while the cameras installed on the outpost give the bird's eye view to the robots and form a part of global perception system. After processing the sensor data, the pose data of ego and target is estimated and a map of the arena is created. Based on the position and orientation of the target and the game state, the ego takes a decision, plans its motion globally and goes to the destination through local path planning. Once the destination is reached, the sensors on the robot detect the armour plates of the target and prepares to shoot them. The decisions are taken based on the states of both the robots and thus a communication is established among the two robots and the sentry. The sentry helps in the global level decisions and planning.

All authors are with Department of Electrical Engineering, Indian Institute of Technology, Kanpur, India

II. Hardware

Since, the robots are not designed particularly for the challenge, therefore, it is necessary to modify them to serve in the challenge. In this regard, the robots have been modified to house several required sensors and computing devices. For sensing the robot surroundings, we endow the platform with an Intel RealSense D455 RGB-D camera, RPLiDAR A1M8 and 4K ELP cameras. For onboard computing requirements, we use Intel NUC Enthusiast with parallel computing capabilities in order to deploy neural networks and other algorithmic components. Apart from the above enhancements, we also plan to make use of sentry outpost which will consist of two 4K ELP cameras along with NXP signal boosters and an Intel Core-i7-11700 Processor to run relevant algorithms.

A. Mechanical Design

In order to house and attach several sensors and other hardware components firmly with the robot, many mechanical modifications have been made. As different components comes in different shapes and sizes, therefore, specialized mounts have been successfully integrated on the robot for installation of all sensors and placement of computing board along with a high speed USB interfacing hub. In our approach, we also have a provision for onboard visual interaction with the robot, therefore, a customized mount for the installation of the visual debugging unit has also been developed. The mounts made for fixing the ELP cameras on the outpost have been optimized to get the maximum area coverage on the map and shock absorbers

are attached to their base to ensure minimum disturbance. All the mounts are 3D printed with PLA plastic and have been fastened firmly to the body of the robots. Moreover, two ELP 4K sentry cameras have been mounted using specialised PLA plastic mounts and have been optimised to cover maximum area of the arena so that the robots can be visible in two cameras at all times.

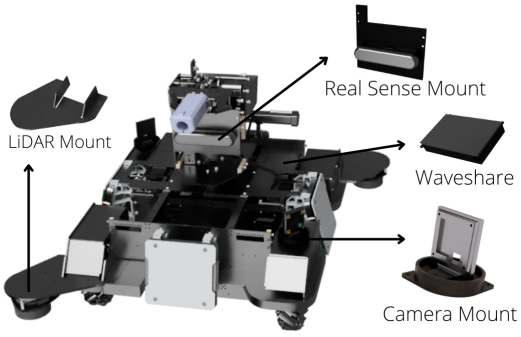

Fig. 1: Mechanical Attachments for AI Robot

B. Sensors

1) LiDAR: According to our previous experiences in the Robomaster challenge series [1], we will use a RPLiDAR A1M8 for dynamic obstacle avoidance as it is economical and has an adequate sampling rate of 8000 samples per second. Moreover, the 360 degree field of view allows one to detect any possible dynamic obstacle within the immediate range of the robot. As the size of the arena is small, the low range of 6 meters suffices for the desired purpose.

2) Onboard RGB-D Camera: The robot will be endowed with a depth camera for aiming and tracking the enemy robots. The depth data will enable us to precisely calculate the gimbal yaw and pitch to hit the target armor module. Keeping all these requirements in mind, we choose to use the Intel RealSense D455. The sensor can provide the RGB-D frames at a minimum of 30FPS and a field-of-view (FOV) of $86^\circ \times 57^\circ$ which is sufficient for the robot to continuously perceive the immediate environment with utmost accuracy. The sensor can operate in different lighting conditions, which is vital due to the dynamic nature of the battlefield. The range of depth measurement is suitable for our purpose as per the dimensions of the arena.

a) Sentry RGB Camera: The ELP USB4K03-V100 camera has been chosen as the outpost zone camera for detection of vision markers, which is required for overhead detection and localization. Since the sentry is required to have high resolution, uninterrupted and efficient view of the entire battlefield, we have chosen a 4K camera with a resolution of 1920×1080 at 30 fps. It has 100 FOV and the feed format is MJPEG. As the camera is mounted externally and is far from the ground station, therefore, we have used PTN24361B-NXP Signal Booster to connect the cameras with the ground station and to prevent noise in the USB bus, USB extension cables provided by RMOC will be used.

3) Onboard Monocular Cameras: Two monocular cameras will be mounted on the two diagonally opposite corners of each robot for localization purposes. The camera feeds shall be used to detect visual markers present in the arena. We have chosen this setup to ensure that atleast one marker is always visible irrespective of the position and orientation of the robot in the arena. To fulfill the purpose, we choose to use ELP Monochrome camera which can provide a 720p video feed at 60fps and has a 100° FOV and feed format of MJPEG.

C. Onboard Computing Device

The onboard computing device that we have used is Intel NUC 11 Enthusiast Kit NUC11PHKi7C. We have used its Tensor cores for matrix multiplication required in convolution operator in Convolutional Neural Network. This comes with a 11th Gen Intel Core i7-1165G7 2.8GHz Processor and NVIDIA RTX 2060 Graphics. The AI performance and 4-core 64bit ARM CPU facilitate real-time execution of computer vision and deep learning tasks. It can run multiple neural networks in parallel which is desirable for our purpose and would help in processing large amounts of high-resolution sensor data.

D. Sentry Computing Device

The computing device placed in the operator zone consumes the live feed coming from the two aforementioned sentry ELP 4K cameras. By processing the respective video feeds, the relevant algorithm relays the pose and refree system data. Sentry computing device consists of Intel i7 CPU,

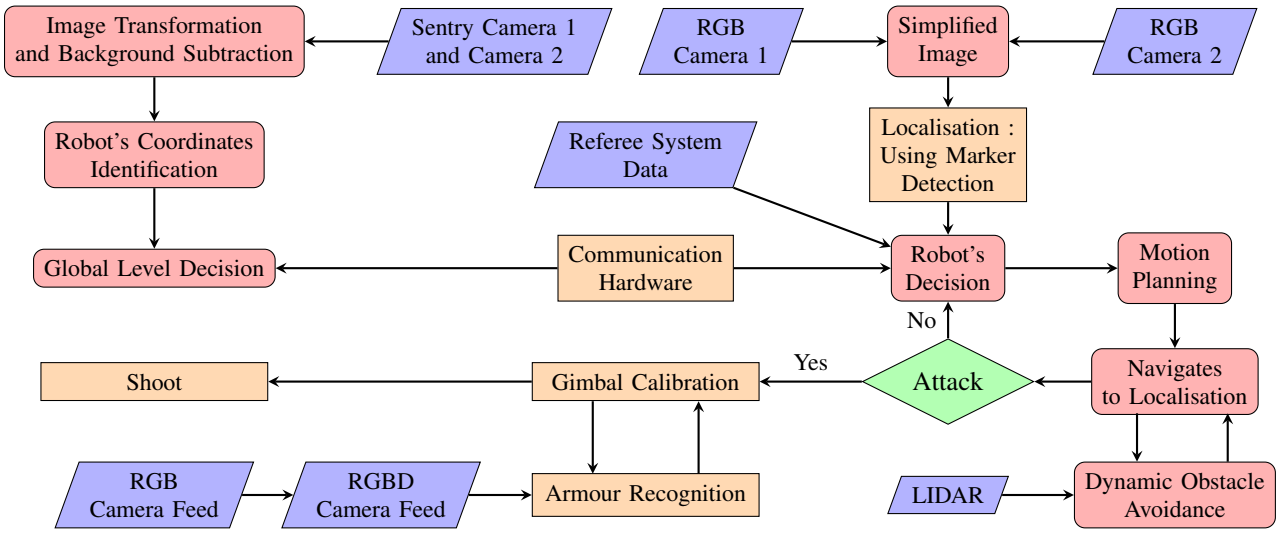

Fig. 2: Overall Software System

32GB RAM and an NVIDIA RTX3080Ti Graphics which would fulfill our requirements of several visual perception algorithms, handling ROS communication, and $\mathcal{Q}$-learning based action selection.

E. Communication Hardware Link Analysis

We divide the overall communication process into two modules i.e. intra-robot communication and inter-robot communication. To achieve the former, we utilize the four USB ports provided by NUC Enthusiast. One of the port has been connected to Quantum Zero USB hub which is independently powered and caters the bandwidth requirements of RealSense D455, visual debugging unit, the central microcontroller and the LiDAR. The electrical power for USB hub has been drawn from one of the auxiliary power connectors available on the main power unit.

On the other hand, for inter-robot communication, we have exploited star topology to establish a communication link between the sentry computing device and both the robots and the referee system server. Based on our previous experiences [2], we have used Netgear Nighthawk XR500 Pro Gaming Wi-Fi router for a stable connection in highly noisy environment. The router allows an uninterrupted data transfer over 2.4GHz at a speed of 800Mbps. 2.4GHz bandwidth is necessary for the connection of refree system. Router has 5GHz bands at an aggregated rate of 1700Mbps which can be used for inter-robot and sentry connections. The high speed is sufficient to relay large amount of data streams at higher frame rates between the robots and the operator computer.

III. Software System

The overall software system is undeniably complex. Fig. 2 depicts the minimal software framework. We have discussed every component of the software in detail below.

A. Autonomous Recognition

Recognition of the armor plate of the enemy robot at the robot level and detection of the enemy robot from the sentry are one of the most important milestones achieved towards success in this challenge. Armor plate detection and their classification on the basis of the pattern into 8 different classes was crucial for efficient decision making as well as estimating the pose of the enemy robot and this task has been well performed by deploying YOLO (you only look once) based architecture (as shown in Fig 3.) on the feed of Realsense 455 camera. With a backbone specifically customized for increasing the accuracy of detection, the light-weighted model achieved a high detection rate of 60 Hz. YOLO, unlike CNN or RCNN or even faster-RCNN, relies on the intersection over union (IoU) technique, which makes the detection faster and more accurate. The model achieved mean average precision (mAP) of 85.68 % and that too at 60 fps which made it an irreplaceable choice. The

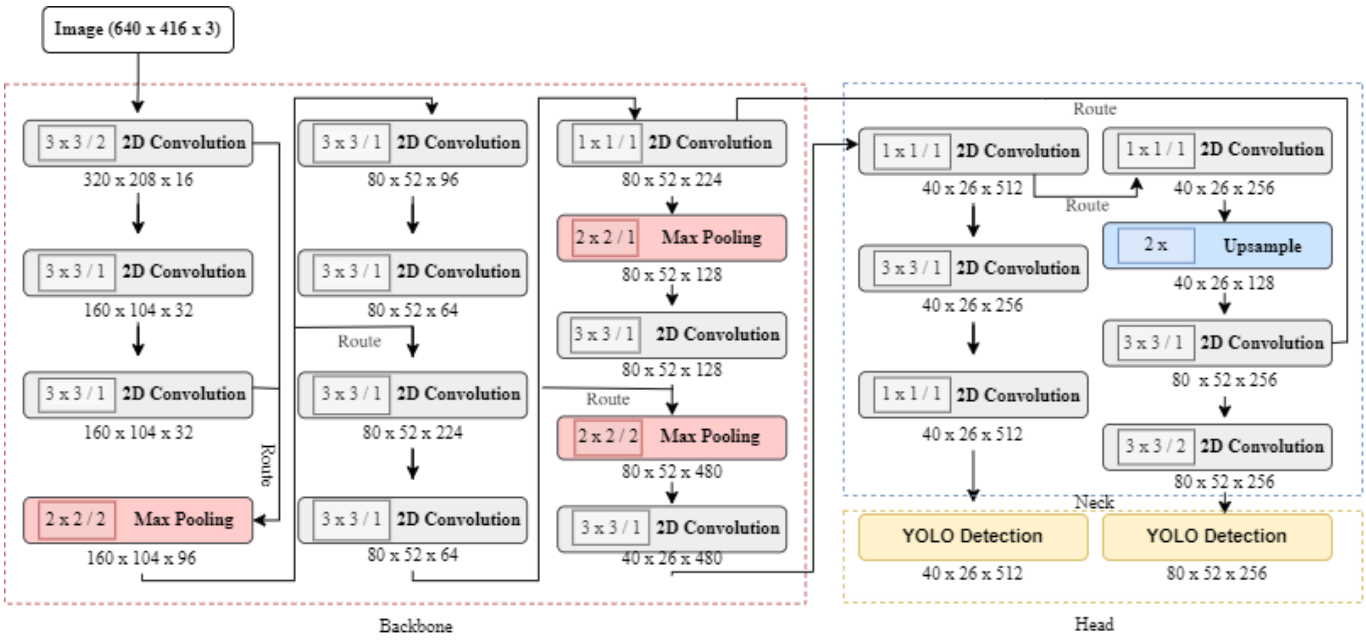

Fig. 3: Neural Network Architecture

Fig. 4: Dataset Augmentation

dataset for training the model was created by using 2000 images for each class, which were captured at different exposure. Images were further augmented by adding gaussian noise to them. Moreover, a few images were rotated and sheared at different angles and finally mosaics were created.

At the sentry level, a model based on YOLO version 4 has been deployed to accurately detect all the robots in the arena and provide their coordinates, which, after perspective transform, returns real-world coordinates (with one of the sentry as the origin) which are further processed and used by decision making module. With mean average precision (mAP) of 82.45 % at 60 fps, the model is capable of accurately detecting the robots moving even at the highest allowed speed, from the cameras situated at the outposts.

To further increase the speed of training and detection, the Darknet framework, which is completely based on C and C++ language has been preferred over Tensorflow or PyTorch based frameworks.

B. Positioning

1) Visual Marker Detection Strategy: For extracting the absolute pose of the robot, we detect the corners of the given visual markers in the pixel space and map them to camera coordinates using pin-hole model which takes inputs as intrinsic and distortion matrices of the camera and then transform these coordinates to real world since we know the real world pose of visual markers. This is done by using the two ELP camera's mounted on two opposite ends of the robot. In order to detect corners of these visual markers we generated our custom dictionary and tuned the hyperparameters of the algorithm such as BitsCorrectionRate and adaptive thresholding kernel window size and used AprilTag corner refinement to give maximum detection accuracy. To avoid the noise in the pose estimation using markers, we have set a threshold on the distance of the detected tag from the robot and on the angle between camera and the marker normal.

The marker 'X' has an indistinguishable symmetry across both x and y axes. To detect it successfully, we divided the image into four quadrants with marker's center being the origin and reordered them according to their quadrants to get the correct orientation of axes. Moreover, the markers 'H' and 'I' appear same after 90° rotation and thus we distinguished them using their color. A situation may arise when no markers are being detected in few frames or the detected axes may fluctuate due to blurry image feed as a result of high speed robot movement. These issues bias the actual pose of the robot to some extent.

2) Laser and Wheel Odometry: To overcome both these issues, we have used rf20 laser odometry(this is obtained from the A1M8 Lidar mounted on the robot.) which uses scan alignment based on scan gradients and happens to work best when the lidar is confined to move in a plane.We have additionally used the wheel odometry and performed sensor fusion using the extended Kalman Filter which updates the measurements using co-variance matrices associated with each sensor and helps in increasing the accuracy of the measurement of the Pose estimated in real time.

We have tried and tested the Localisation module and it can handle the problem of random initial locating with the visual marker odometry which gives it its absolute pose in the world coordinate system.The high speed locating problem was resolved by fusing the Lidar odometry and the wheel odometry with the absolute pose obtained from the marker.

We have a rate of about 40Hz at publishment of pose and an accuracy of 10cm assuming that the centre of the robot is the tail of the robot and the head denoting its orientation.

C. High Speed Target Tracking

After detecting the enemy robot, a KCF tracker is deployed on the detected window to determine the direction of the travel of the robot to allow recovery from full occlusion is needed because the robots are subject to leaving the frame of the camera now and then owing to the obstacles present. The application of a tracker simultaneously in synchronization with the detection also speeds up the tracking process.

D. Motion Planning

Our global planner is an implementation of the A* search algorithm in ROS Navigation Stack. Although A* is a simple and efficient planning algorithm, the re-planning of the global path when

Fig. 5: Visual Marker Detection

the map is updated with dynamic obstacles (The enemy robots and debuff zones) becomes computationally very heavy, and so we have planned to use the local path planner to deal with obstacles which the robots will continuously detect in the arena using the LiDAR sensor mounted for dynamic obstacle avoidance. The algorithm to be employed for local path planning is Timed Elastic Bands trajectory optimization. The nodes containing parameters that contribute to the same objective function are connected by a corresponding multi-edge. Each objective function depends on a subset of TEB-states (configurations and time differences), and a hyper-edge represents the objective function and connects nodes that correspond to the configurations and time differences that occur as parameters in its evaluation. To finally optimize this graph, the general (hyper)graph optimization "g2o" algorithm is used, which will yield the ultimate optimized local path that the robot will follow. The obstacles in the path will be circumvented exactly, meaning the extra time it takes to deviate from the planned global path will be optimized.

E. Automatic Firing

Orienting the gimbal to aim at the recognized armor plate is crucial for imparting damage to the enemy robot. The exact pixel coordinates of the centre of the armor module have been computed by the armor recognition module.For executing the motion of the gimbal of the robot relative yaw and the absolute pitch angles have been computed. For calculating the relative yaw the pixel coordinates have been mapped with the depth obtained from the Realsense camera and thus yaw have been computed from the following equations.

$$x = (X - cx) * z/fx$$

$$yaw = atan(x/z)$$

The pitch motion of the gimbal is executed when the yaw is aligned within a threshold of yaw.The pitch angle is computed from the trajectory equations as shown below.

$$H + Ztan(\theta)t - \frac{1}{2}\frac{gZ^2}{V^2} - \frac{1}{2}\frac{gZ^2}{V^2}tan(\theta)^2 = 0$$

where, H,Z are height of projectile launcher from target level and horizontal distance between launcher and target respectively.
Now, to compensate for the effect of air drag, theta is further modified as

$$\theta_{final} = \theta + aZ^2 + bZ + c$$

where a,b and c are experimental constants.

All these calculations give exact parameter values and a PID controller orients the gimbal accordingly. But, the above procedure yielded desired results only when both robots moving at a slow speed, which is a rare case. This challenge, of aiming moving target, has been effectively tackled using ANN by learning the relationship between successive coordinate points involved in a generic

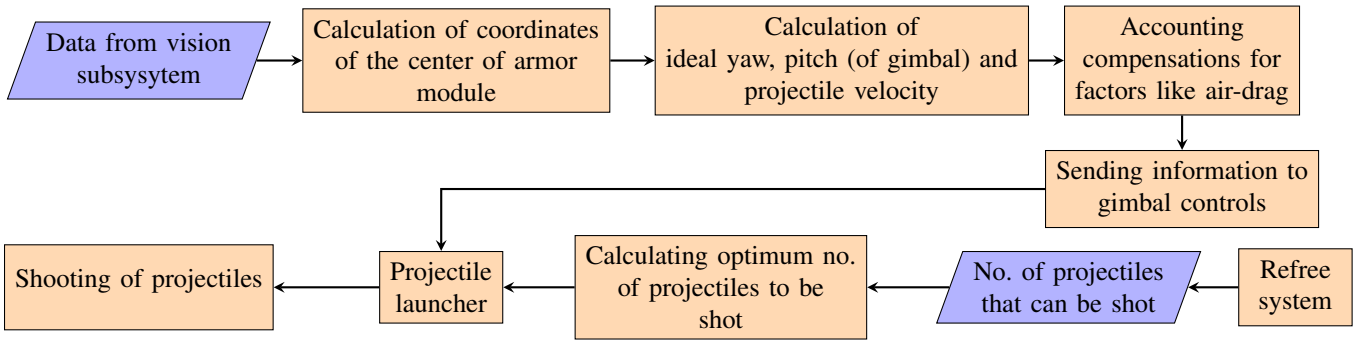

Fig. 6: Aiming and Shooting

continuous cubic movement and separated by one sampling period, T. Once the ANN is trained, the position, velocity, and acceleration of the target is predicted and the projectile is fired [1]. Fig. 6 summarizes the entire projectile firing process.

We have tested and deployed the firing Module with a shooting precision of shoot 90 percent and accuracy gradually fading after the distance of 3.5m which can be thought of farthest strike distance.

F. Global Perception System of the Sentry

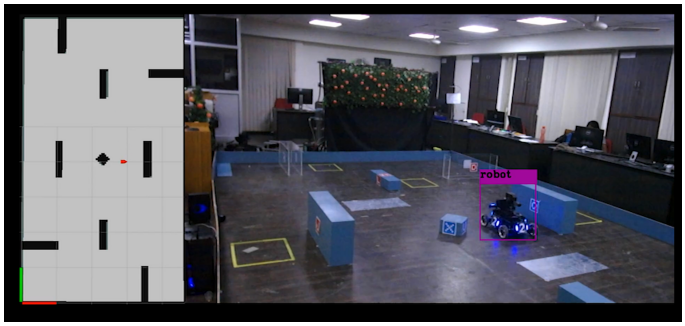

Fig. 7: Sentry Camera Coordinate Transformation

The sentry's camera feed gives a bird's eye view of the arena. It helps in relaying the enemy coordinates for the decision-making stack in the post-lurking phase of the competition. The two cameras used are HD, and the combined field of view of both the cameras covers the whole arena. .The coordinates obtained from the sentry are subtracted from the coordinates obtained from the localization of our bots to recognize the enemy robot's coordinates.

The sentry pose estimation is done using perspective transformation as there is a unique linear transformation from the pixel space of the camera to the x-y plane of the bot's base.

The transformation is calculated by specifying four corresponding points in the pixel space and the real world. We have chosen the reflective mirror and the debuff-zones pixel coordinates depending on the sentry's view and occlusion.

The matrix obtained using this perspective transform is linearly multiplied by the bounding box pixel coordinates obtained from the detector to get the real-world coordinates of the robot.

We have opted for this method over pixel coordinates triangulation. It is more robust in cases where the bot is visible through one sentry and has a better frame rate as we don't have to extract features from the robot for checking correspondences for triangulation.

As the bot's center plane is slightly elevated in the upward z-direction, we have chosen to take the pixel coordinates of (x+w/2,y+(3h/4)) so that we are close to the robot's base, which is near about on the ground plane and its distance will be estimated from the global origin. Fig. 8 depicts the pipeline of sentry system.

1) Communication between the Robots and Sentry: We have set up three independent rosmasters, one on each robot and one on the sentry system computing device. We set up a communication link between the three rosmasters for information transmission and retrieval across robots and sentry outpost using fkie Multimaster package from ROS stack. The main advantage of Multimaster communication is the guarantee of a robust communication system which is immune to unexpected communication break and other cases of a robot going offline as when the health of a robot goes down or one robot ending up in damage unexpectedly. This approach also handles the limited bandwidth

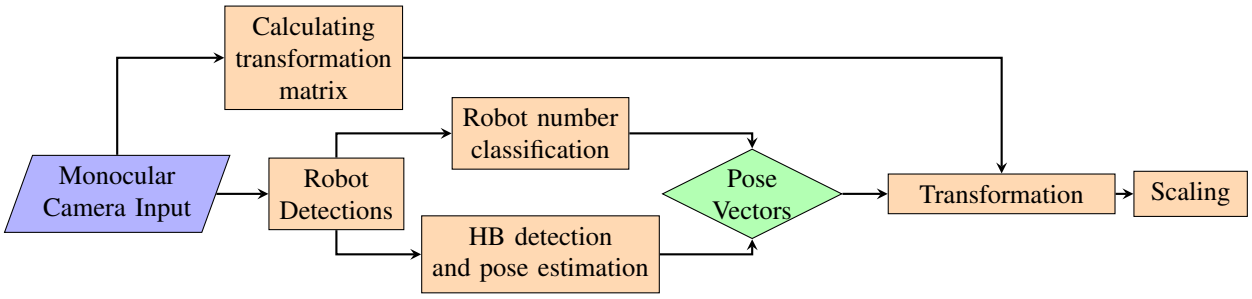

Fig. 8: Sentry System Pipeline

of the communication system and almost removes latency. One approach which could be taken is to establish a single central rosmaster that controls the advertisement and subscription of topics and services in both the robots as well as the sentry outpost. This way, there would have been a central brain of the whole system which collects and controls all the information regarding the active nodes, topics, and services. This approach adds congestion to the network and accounts for added latency, which decreases the performance of each robot. Since our robot has to respond fast to the surrounding situation, added latency may lead to falling prey to enemy attack.

G. Intelligent Decision-Making

1) QL-BT Approach: The competition requires skill, coordination and the ability to analyse different situations with utmost precision and timing. Due to the increased state space and the numerous factors involved, it could not be modelled by a case-based approach, so we implemented a Behaviour tree, refer Fig.9, accelerated using Q-learning. Our approach is advantageous as the robot is exposed to a variety of states during training which makes it more robust and efficient. The RL agent decides actions based on its state, while many of its actions are defined it takes decisions that are best suited to the scenario that cannot be foreseen by conventional algorithms. We defined two hierarchy levels namely the agent and the sentry as some decisions are better made at the sentry level like coordination of the robots for attack, and at the agent level such as retreat to the buff zone as an added advantage it allows for easy debugging of the different modules used at the robot level and the sentry level.

2) RL Agent Level Autonomy:

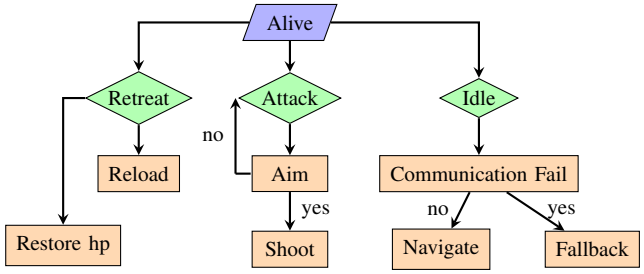

Fig. 9: Behaviour Tree

Health	Ammo	Enemies	Action
L	L	0/1/2	Restore Health
M/H	L	0/1/2	Restore Ammo
L/M/H	M/H	1 or 2	Attack
M/H	M/H	0	Search

Fig. 11: State - Action Table

The robot level addresses functions that are central to the robot such as attacking when the enemies are in the field of view, retreating when we are having low health and restoring ammo when there are low bullets. The state of the RL-agent is defined by three variables namely enemies (in field of view), health, ammo as these are easily accessible from the referee system and can be perceived by the robot even when communication is lost. The health and ammo are divided into three different levels low, medium, high.

The action space agent is divided into five routines:

- Restore health: Activated when the agent has low health and low ammo. Makes the agent go to the nearest buff zone irrespective of the enemies in its field of view. If the routine has been used before, escape routine will be activated.
- Restore ammo: Activated when the agent has low ammo with above medium health. Makes the agent go to the nearest buff zone. If the routine has been used before, search routine

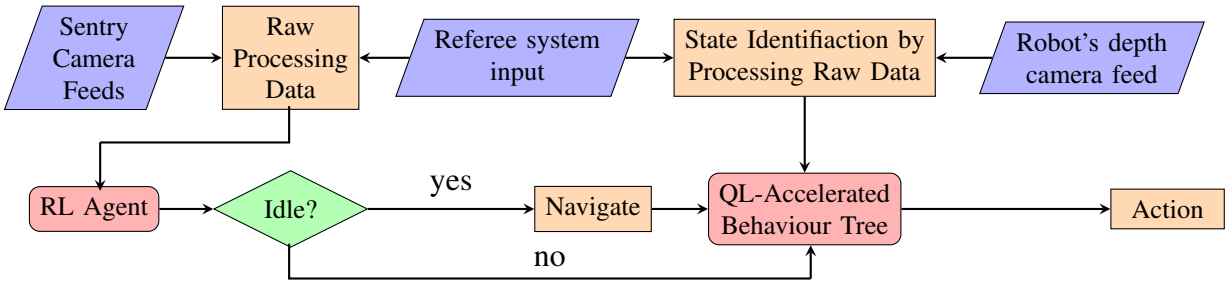

Fig. 10: Decision System

will be activated.

- Attack: Activated when there is more than medium ammo and more than one enemy in the field of view. Moves the gimbal so that it can aim at the enemy robot's armour plate after detection to inflict damage. This routine is usually done after the robot comes into the attack zone after the help from the sentry.
- Navigate: This routine is a simple mechanism where it follows a path to the waypoint published by the sentry. In case communication is lost it follows a simple search and shoot mechanism till it restores communication.
- Escape: This is a type of subroutine where the low health agent tries to avoid enemy robots and goes into self-preservation till completion.

The q-table played a crucial role in the training phase of the agent where it was allowed to search randomly and update Q-values of the states visited. These values were used in deciding the priority order of the behaviour tree which was used in deciding the actions of the agent. Fig. 10 summarizes the entire decision-making system.

3) Global Level Autonomy: This level is primarily dedicated to deciding the best possible attack strategy for the robots. The decision-making system at the outpost is well equipped with the data from the sentry camera feeds and referee system, and therefore aware of the location of all the robots as well as the debuff and buff zones. Trained with reinforcement learning, mainly Deep Q learning or DQN, the system commands the robots to navigate to the optimal location for attacking the enemy robots. The model uses distances between the robots as states, and each time an attack routine is performed, the strategy followed is rewarded a certain score based on its effectiveness. Therefore,

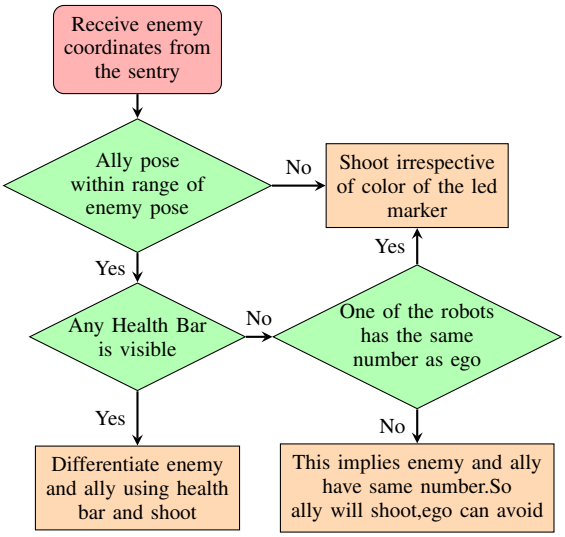

Fig. 12: Post-Lurking Engage Pipeline

the system implements the best possible strategy for the present state. Also, we have refrained from keeping the robot HP and ammunition as states in the model, this not only reduces the complexity to a great extent but also makes the decision-making process much faster and efficient which is a necessity for challenges like this. The decision of moving to buff zone comes under robot level autonomy to ensure that this action does not interfere with the attack action of the other robot which could have been the case if health and ammunition were considered as states for the decision-making process at the global level. Thus, the offensive gameplay of the robots is strategically planned and implemented. Moreover, it enables the robots to impart critical damage on the enemy robots by using strategies such as capture and attack.

H. Enemy recognition

Before the lurking phase the enemy robot is detected using its presence of blue/red armor modules. In the lurking phase the enemy robot is detected using the health bar and elimination of coordinates of our own robots using their localization data. The robots will provide their coordinates and orientation to the sentry based on the robot level localization module. On comparing the coordinates we can identify the enemy robots. The pose is also estimated at this point, using the patterns made on the armor plate. The images detected will be passed through a pre-trained Neural Network and the resulting output tells us the corresponding Armor plate number. Through this the orientation of the robot can be determined. Thus the pose and location of an enemy robot as reduced to a set of 2 points in order.

I. Visual Interaction and Debugging

The system design under consideration is quite complex and consists of several key modules including hardware and software components. Therefore, we have installed a Waveshare LCD Display Module. All the actions at the robot level are performed with the help of this screen, which includes launching nodes, checking the robot response and verification of proper working of all the major modules in the robot.

The importance of this module lies in the fact that being a complex system, the pilot must be able to see the overall status. It helps reducing the development, testing and debugging time drastically.

The Waveshare screen is connected with the onboard jetson processor via HDMI. The approach used for real-time debugging of the overall system is based on tracepoint selection. Tracepoints are tagged at particular points in the code, which is executed upon the successful running of an execution sequence. We use tracepoints at all our running nodes and display the tags in the Waveshare display. The appropriate conversion between different frames is ensured by checking the rqt_tf_tree. It puts the coordinate frames in a tree structure, which makes it easier to debug in case of any error.

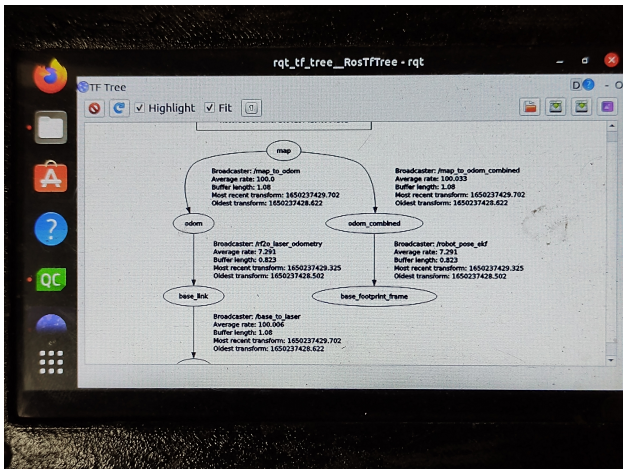

Fig. 13: Visual Debugging using On-Bot Display

J. Video report submission

Here is the link of video display submission uploaded on official youtube channel of team ERA-IITK :
https://youtu.be/U3kv0PN-7x0

IV. Acknowledgement

We would like to express our gratitude to our supervisor Prof. Laxmidhar Behera, Department of Electrical Engineering, IIT Kanpur for his guidance and support. Prof. Behera is a very renowned professor of Indian Institute of Technology Kanpur. He specializes in the field of intelligent systems and autonomous robotics with over 50 published journals, and over 150 conferences and 2 books. The team works in his Intelligent Systems Lab (ISL) which is a lab of international eminence. Prof. Behera participated in the Amazon Robotics Challenge 2017 (ARC17) held in Japan, Nagoya where his team IITK-TCS was ranked 3rd, 4th and 5th worldwide in a pick, stow-pick and stow task respectively.

References

[1] Ashok Kumar Chaudhary, Suraj Hanchinal, Suryansh Agarwal, and Laxmidhar Behera. Chasing and aiming of a moving target.

[2] Ashish Kumar, Mohit Vohra, Ravi Prakash, and Laxmidhar Behera. Towards deep learning assisted autonomous uavs for manipulation tasks in gps-denied environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1613–1620. IEEE, 2020.