

Design and Development of AI Robot

Technical Proposal: IIT Kanpur ERA ICRA AI Challenge

Ashok Kumar Chaudhary, Suraj Hanchinal, Suryansh Agarwal, Soumya Ranjan Dash, Karan Vaish, Shobhit Jagga, Aviral Khare, Muskan Agarwal, Luvneesh Kumar, Sourish Mondal, Haren Bhandari, Vaibhav Vardhan, Dipak Miglani, Sparsh Agarwal, Harshit Sinha, Ashish Tiwari

I. INTRODUCTION

The problem statement of the AI Challenge is to build the software pipeline of an AI powered autonomous robot. The robot should be able to localize itself in the static arena with the enemy robots as dynamic obstacles and create a map of the surrounding in order to use it to detect obstacles and for path planning which is explained later in the proposal. The sensor data obtained is used for localization and mapping. The computer vision system acts as the visual cortex of the robot to help it identify enemy robots and their armor modules. Sentry system, comprising two diagonally placed monocular cameras, acting as a god's eye, identifying enemies and detecting visual markers on the arena to estimate the location information of the opposite team robots. The brain of the robot relies on a reinforcement learning-based decision making process to formulate strategies of actions in the arena. The whole pipeline has to be established in both the robots after which a communication link needs to be set up between the individual robots and the sentry system. This communication link establishes transfer of information across multiple computation centers.

II. SENSOR TYPE AND USAGE

The arena is sized at $(5.1\text{m} \times 8.1\text{m})$, the maximum distance would be the diagonal distance from one corner to the opposite corner $\sim 9.43\text{m}$ however, this diagonal is obstructed with multiple obstacles. The total weight of the robot should be less than 25 Kg. Each round is 3 minutes long. Comparing the dimensions of the arena to the dimensions of the robots, the number of robots and their motions, the arena can be considered to be relatively small. This requires the sensors to measure at a relatively high frequency and with high accuracy and precision. The mechanical addition to these sensor installation will include the mounts to be built and set on the robot.

A. LiDAR Sensor

We will use a Light Detection and Ranging (LiDAR) sensor for obstacle avoidance and path planning. These

functionalities can be achieved using a 2D LiDAR sensor. A 2D sensor has been traded off vs. a 3D one to avoid higher computational time and data bandwidth. A 3D lidar would provide more accurate data at the cost of computation time, Thus reducing the rate of update. We have chosen the Slamtec RPLIDAR A2 as the best fit for our purpose, and its depth range is 0.15m-12m with a resolution of 0.5 mm. It also features a high refresh rate of 4000 Hz. The LiDAR is placed in the front of the robot and is low enough on the body to detect obstacles accurately.

B. Cameras

The robot will be equipped with a depth camera to provide accurate depth information in addition to the 2D image information to aid in the shooting algorithm. For this, we have chosen the Intel RealSense D435i camera. The range of depth is suitable according to the dimensions of the arena. The depth camera is placed just below the gimbal for accurate depth data for the shooting algorithm. The camera will be mounted such that it rotates along with the gimbal in the horizontal plane. This has been decided to increase the effective field of vision of the camera while keeping it sufficiently stable.

To efficiently track the enemy robot, we have chosen a higher resolution camera to get better visual details. With its frame rate, the robot is visually always in sync with the environment, which is essential since the arena is relatively small. It can operate in different lighting conditions, which is essential due to ambient light changes during the match. We have therefore decided to use Logitech C615 cameras for the detection of the enemy robots and the vision markers. Two of these monocular cameras along with the Intel RealSense depth camera are installed on the chassis symmetrically to enable us have to a 360 degree vision at all times.

Sentry cameras are required to provide high resolution visual feed so as to detect the vision markers placed in the arena. It is also required to provide a wide field of view so as to cover the whole arena. The Sentry system is also responsible for detecting



Hardware	Specification
 <p>Fig. 1. Slamtec RPLIDAR A2</p>	<ul style="list-style-type: none"> • Depth Range: 0.15m-12m • Depth Resolution: 0.5mm • Horizontal Field-of-View: 360° • Measurement frequency: 4000 Hz
 <p>Fig. 2. Intel RealSense D435i Camera</p>	<ul style="list-style-type: none"> • Resolution: 1280 × 720 • Frame rate: 30 frames per second • Operating Range (Min-Max): 0.11m - 10m • Depth Field of View: 85.2 × 58
 <p>Fig. 3. Logitech C615</p>	<ul style="list-style-type: none"> • Resolution: 720p • Frame rate: 30 frames per second • Field-of-View: 78°
 <p>Fig. 4. Amcrest IP8M-2496EB-28MM</p>	<ul style="list-style-type: none"> • Resolution: 2K (3840x2160) • Frame rate: Upto 30 frames per second • Field-of-View: 112°
 <p>Fig. 5. Netgear XR500</p>	<ul style="list-style-type: none"> • 2600 Mbps Speed • Frequency: 2.4 GHz, 5 GHz

TABLE I
Sensor specifications

robot and thus supporting high frame rate is also an essential parameter. Continuous change in lighting conditions requires a variable aperture lens with a controlled change in aperture. The module also has to have a sufficiently large resolution as described above; thus we have chosen the Amcrest IP8M-2496EB-28MM. It has a high field of view of 112 degrees. The sensor is Sony Starvis, thus solving our purpose. It has a high frame rate and resolution for a robust vision algorithm.

III. COMPUTING DEVICE

A. Onboard the robot

The *NVIDIA Jetson TX2* is a full Linux computer designed to help make robots, drones and other devices

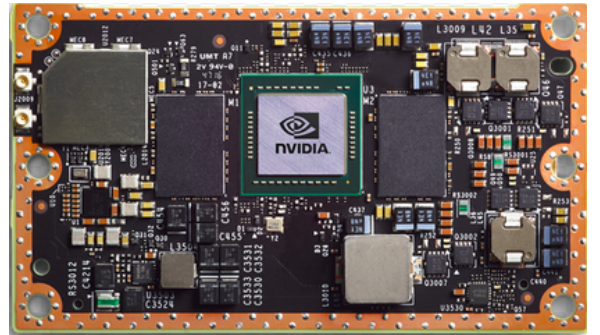


Fig. 6. NVIDIA Jetson TX2

that rely on computer vision applications. The board's

main attraction is a GPU based on *NVIDIA*'s latest Pascal architecture. The *Jetson TX2* can run Ubuntu and ROS (Robot Operating System), which is built on top of the Linux OS. It has 32GB of storage, 8GB of LPDDR4 memory which is sufficient to store learning models and 802.11ac Wi-Fi which will be used to set up communication between the 2 robots in the arena. The *TX2* has two *Denver 2* CPU and four Cortex-A57 CPUs. *Jetson TX2* accelerates cutting-edge deep neural network (DNN) architectures using the *NVIDIA* cuDNN and TensorRT libraries. Its dual-CAN bus controller enables autopilot integration to control robots and drones that use DNNs to perceive the world around them and operate safely in dynamic environments. We prefer to use *Jetson TX2* because it is twice as energy efficient for deep learning inference than its predecessor, *Jetson TX1*, and offers higher performance than an *Intel Xeon Server CPU*.

B. Operator Zone



Fig. 7. Intel NUC 8i5INH

The Processor in the operator zone will be processing continuous input from the 2 Sentry cameras and will be appending the positions and other data in a virtual map. The RL model and Decision system will also be running on the Operator zone Computer. Hence there is need of a heavy computing device at this zone which is capable of processing computations faster and minimize latency in the processing. We will use Intel NUC 8i5INH System. The NUC Supports Quad-core 8th Generation Intel® Core™ processors and Radeon® 540X discrete graphics. It has 32GB of storage, 8GB of LPDDR4 memory. This will be connected to the netgear XR500 for efficient communications at all times.

C. Overall weight calculation

The weight of the Slamtec RPLIDAR A2 is 190g. The weight of the Intel RealSense Camera is 24g. The weight of the *NVIDIA* Jetson TX2 is 85g. The weight of the Logitech C615 is 103g. The weight of the router is 762g. The total added weight of the equipment is 1593g. The cumulative weight is under 25kg.

IV. COMMUNICATION HARDWARE LINK ANALYSIS

For maintaining a continuous link of communications between the operator zone computing device, as well as both the robots and the referee system server, we plan to use a robot that ensures stable connection. Asus Netgear XR500 has been chosen because it provides an uninterrupted link even in high traffic and noise situations. It uses flexible radio assignment in the 5GHz and 2.4 GHz bands. It also provides 5 GHz Ethernet ports (4 LAN + 1 WAN) for maximized wired speeds. It supports 4 external antennas for better WiFi coverage, faster speeds and less interference. High-speed links upto 2600 Mbps allow us to relay large data streams at high frame rates between the robots and the operator computer.

Additionally, wifi cards will be installed on both robots. It is required because the wireless communication system on Jetson Tx2 has limited bandwidth and is susceptible to low performance over noisy spaces.

V. SOFTWARE SYSTEM

Sentry system

The sentry system uses visual algorithms to generate a real-time dynamic map of the arena in a back-end server. It uses vision markers information to localize and position the enemy robot in the arena map. The two sentry cameras on each designated outpost at the diagonally opposite corner would allow the system to cover the entire arena at once. Also, since the sentry cameras are placed diagonally, there would not be any blind spots to the system. However, since the cameras are placed at a height on each outpost, the frames obtained by the system will be perspective in nature. To solve this problem and get the top view of the arena, we will be using geometric transform functions. These functions do not change the image content, but reform the pixel grid and map the deformed grid to the destination image. Once the perspective view is converted to an top view, the 2D map can be obtained from the visual feed. The transformed view of the arena contains various visual markers symmetrically aligned in the map. We would use visual marker detection algorithms to detect the unique identity of each marker. As the visual markers are not standard AR tags, To detect the Vision tag around the main play zone, the team has decided to use a custom Marker dictionary. Our algorithm will consist of three main steps: feature detection or extraction, feature description, and feature matching. The feature extraction and description steps are similar to any standardized marker detection algorithm, e.g. ArUco Tags, or FAST tags. We have not

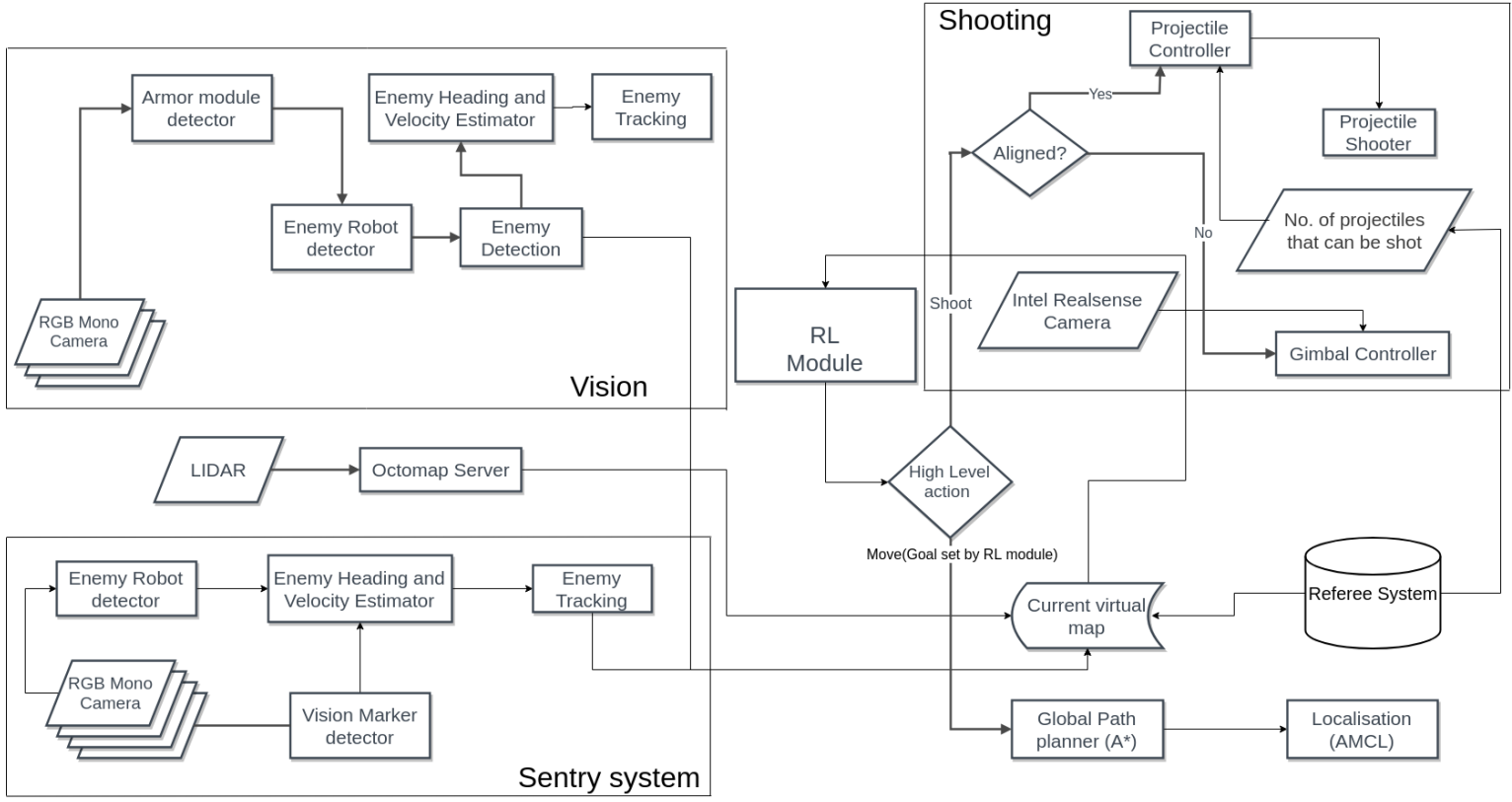


Fig. 8. Overall Software System

found the 5x5 vision tags to be an instance of any common tag library. Thus we propose the following pipeline. We first look in both the reference and target images for features e.g. lines, corners. These describe part of the object to be recognized. These features can be later used to find the reference object in the target image. We will assume we have found the object when a certain number of positive feature matches are found between the target and reference images. The descriptors obtained will be matched to find valid match on the reference frame. Since we have already found a set of matches between both images we can certainly find directly by transferring from a standard AR-tag detection library. The demo of detection of the custom AR tags have been attached

AI Robot Perception System

The robot will be using computer vision algorithms to detect the armor modules and to determine their real-world position and orientation relative to the robot. This information regarding the pitch, yaw, the velocity, the distance and the angle with which the launching mechanism needs to align itself to be able to shoot the respective armor module will be passed to the gimbal control. Apart from this, the other two major tasks under this section are to provide the real-time pose of the enemy robot in-order to determine its direction of travel for future reference and to track the enemy robots after

detecting them.

The detection of the enemy robot will be primarily done by using the color and the position of the LED markers that are present on the armor modules of the robot. Apart from emitting bright light, the LED panels possess some significant geometry attributes including width-height ratio, area, and orientations, making them a very reliable source of detection of the armor modules in a noisy environment. We refrain from using other detection methods here like neural networks because of the low accuracy and high computational complexity, as well as a requirement of large labeled data. We shall tune some internal camera parameters such as exposure, saturation, brightness, and contrast using V4L2-CTL API. This would solve the problem of overexposure near the center region of the armor plate and would allow for faster detection using a more sparse input. After obtaining the above parameters, they are sent to the gimbal control which accordingly adjusts the projectile shooter with compensations accounted for the acceleration due to gravity, air drag, and motion, increasing the accuracy of the shooting process. The enumeration of armor plates will be done by observing the wheels of the bot. A circular view will indicate lateral plates while a horizontal view that is rectangular will indicate the front and back armor plates

After detecting the enemy robot a KCF(Kernelized Correlation Filter) tracker [1] is deployed on the de-

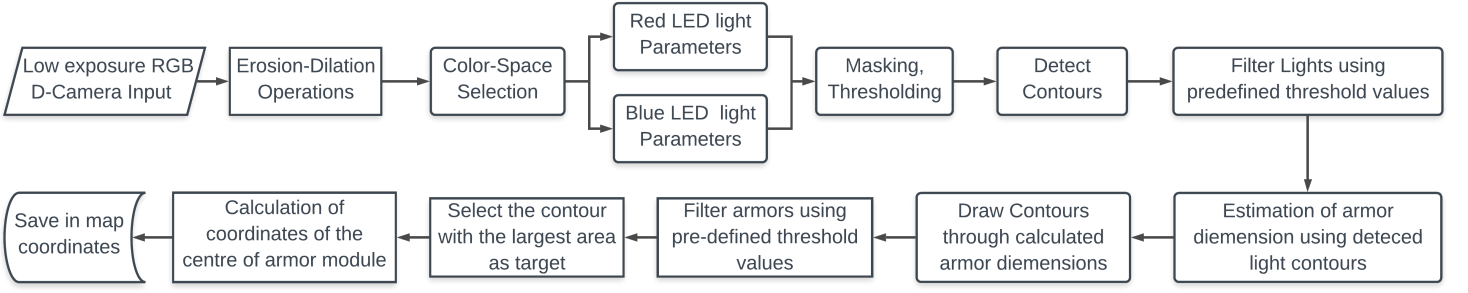


Fig. 9. Armor Detection Pipeline

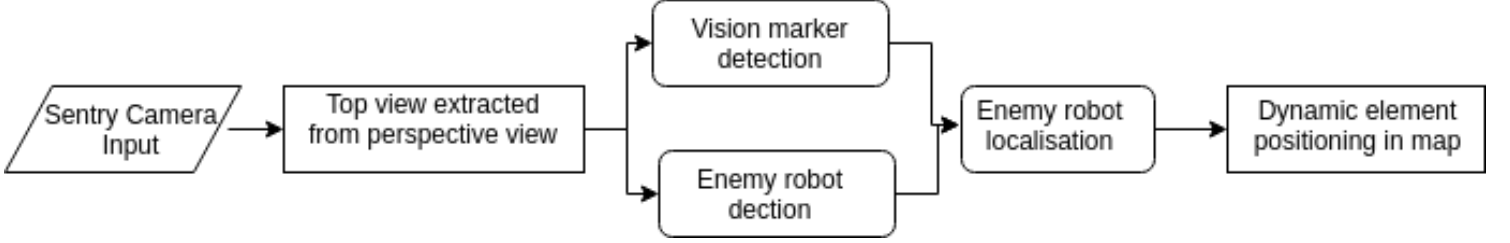


Fig. 10. Sentry System Pipeline

tected window to determine the direction of travel of the robot. Recovery from full occlusion is of paramount importance because the robots are subject to leaving the frame of camera every now and then. After losing a dummy target due to some obstacle in the path, the KCF tracker tracks the object in both forward and backward directions in time and measures the discrepancies between these two trajectories. The application of a tracker simultaneously in sync with detection also speeds up the computation, as once the object is detected the tracker stores these set of points and tracks their movement between two consecutive frames until they go out of the frame, after which the frame is again detected for enemy robots and so on the process continues and reduces the need for applying the more time consuming detection algorithm on every frame. Some other tracking algorithms are Multiple Instance Learning (MIL), Kernelized Correlation Filters (KCF) and Tracking, Learning and Detection (TLD). MIL does not report and recover from full occlusion. KCF builds on the idea presented by BOOSTING and MIL, giving better speed and accuracy than both. Like the other two, it too cannot recover as reliably as the median flow tracker from full occlusion. TLD cannot be used for our task because it gives false positives which is a big disadvantage. Therefore, median flow is the best tracker for this task.

Gimbal Control

The main task of the gimbal would be to align the launching mechanism in the direction of the shooting

trajectory calculated using a non-linear model. For the purpose of regulating a projectile trajectory through the synthesis of appropriate control laws, the availability of a sufficiently accurate nonlinear projectile model is of crucial importance. During flight there are two kinds of forces acting on the Projectile motion, the weight and resultant body aerodynamic forces. We first get the center coordinates of the enemy's armor module through the tracking algorithm with the help of the depth sensing camera, after which we can easily get the yaw angle by aligning the line of sight of the frame of the camera with the x coordinates of the target. For calculation of the pitch angle, let Z be the distance of the target for the robot and H be the relative height between the shooting module and the target.

$$Z = V \cos(\theta_{calc})t$$

$$H + V \sin(\theta_{calc})t - \frac{1}{2}gt^2 = 0$$

Solving these simultaneous equations in θ and t by eliminating t , we get a quadratic equation in $\tan(\theta)$.

$$H + Z \tan(\theta_{calc})t - \frac{1}{2} \frac{gZ^2}{V^2} - \frac{1}{2} \frac{gZ^2}{V^2} \tan^2(\theta_{calc})^2$$

We get the value of θ from this equation. However the angle has been calculated assuming an absence of air resistance. To get the actual angle of launch required to hit the target we use a quadratic polynomial approximation for calculating the offset angles to compensate for

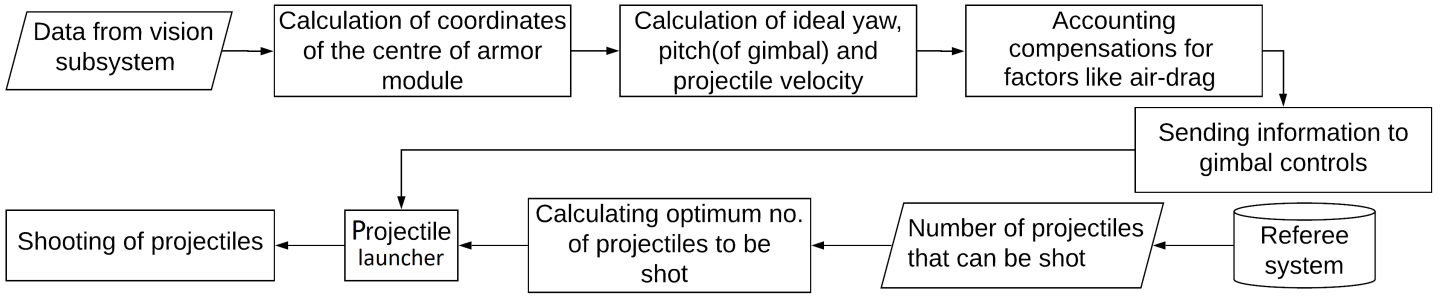


Fig. 11. Shooting Mechanism pipeline

the air resistance. Hence the final expression of pitch is:

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

where a,b and c are experimental constants.

Then obtained pitch and yaw angles would be fed to a simple PID controller responsible for actually handling the control of the gimbal actuators along with the response and settling time of the gimbal. After calculating the orientation of the target, the PID controller would then instruct the micro-controller to align itself to the target in the least possible time and reducing the oscillations, thus achieving accurate shooting precision. Till this point, calculations done assume both the target and shooter to be static. But this won't be the case most of the time so when we consider the motion of enemy robot, location forecasting of enemy's armor is required. An ANN would accomplish this by learning the relationship between successive coordinate points involved in a generic 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.

Motion planning

Localization

Localization is necessary for the robot to be aware of its position in the map; the pose is determined using the LiDAR sensor data. Particle filters will be used for this purpose. The robot will generate many random guesses of where it is going to be next. Each particle contains a full description of possible future states. Now the robot observes the environment and discards all particles inconsistent with the observation and generates more particles close to what appears consistent. Finally, the particles converge to where the robot is.

The localization algorithm we are using is Adaptive Monte Carlo Localization (AMCL). Given a map of the environment, it tries to match laser scans to the map, thus detecting if there is any drift occurring in the pose estimate based on odometry. This drift is then

compensated by publishing a transform between the map frame(it is a global frame in which the pose of the robot makes discrete jumps based on sensor data) and the Odometer frame.

Obstacle Avoidance

The robots will continuously detect obstacles in the arena using the LiDAR sensor. Since the layout of the arena has already been provided, the map will be initialized with these static obstacles. Mapping is needed because the robots need to know the location of dynamic obstacles in the arena (the other robots).

Since the robots will be in a moving state more often than in a stationary state, the algorithm should be able to register changes in the map (that is, allow dynamic mapping) and should follow a probabilistic approach, rather than a deterministic approach, to account for those changes. To provide more memory to the more resource-intensive decision-making process, memory usage in the mapping process should be optimum.

Concluding from the requirements above, we will be using OctoMap [2] as our mapping framework. Octomap is a mapping approach based on octrees and uses probability occupancy estimation while keeping memory usage at its minimum. The primary advantage of Octomap is the very small memory footprint. Experimental observations of Octomap have led to the conclusion that the size of the maps never exceed more than a few MBs. The static obstacles will be initialized in Octomap using a pre-saved map based on the layout of the arena. During a match, the dynamic obstacles (the other robots) will be mapped on the same map based on the laser scans of the LiDAR sensor. This occupancy map will be used by the path planner for motion.

Path Planning

The whole arena can be divided into grid cells assigning one cell as the 'start' and another as the goal. Global and local cost maps of the arena will be built from the probabilistic occupancy map of the arena, and these cost maps will be used to plan the global and

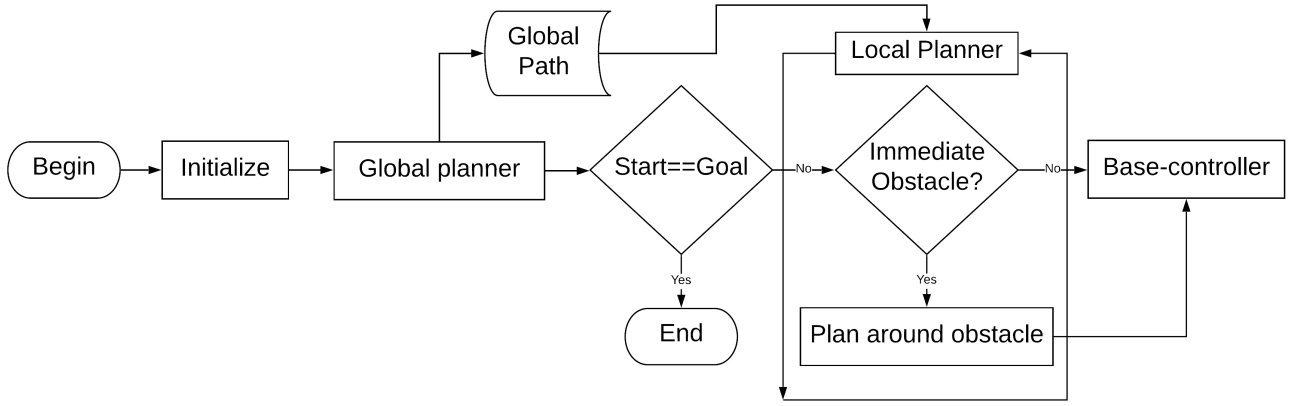


Fig. 12. Path Planning Structure

local path, respectively. The global path will define the path through which the robot has to go to reach the destination, assuming a static map. The local path will define the exact trajectory that the robot has to go through, in real-time depending on the obstacles in its immediate locality. Since the map has static obstacles, the only obstacles that may obstruct the robot during the traversal of the global path is an enemy robot crossing its path.

We will use a path planning strategy involving the use of both a global and a local path planner. The global planner uses a *priori* information of the environment to create the best possible path, and the local planner recalculates the initial plan to incorporate dynamic obstacles. The global planner we will be using is an implementation of the A* search algorithm in ROS Navigation Stack. The algorithm gives us the cost of many paths and the length of the shortest path after complete exploration of all possible paths without considering the requirement of any local changes. The actual sequence of edges leading to the path is extracted by keeping track of the predecessor of every node encountered. Although A* is a simple and efficient planning algorithm, the re-planning of the global path when the map is updated with obstacles becomes computationally very heavy, and so we have planned to use the local path planner to deal with obstacles.

The local path planner that we will be using is the Dynamic-Window Approach (DWA) path, planner [3]. In the dynamic window approach, the search for commands controlling the robot is carried out directly in the space of velocities. The velocity space is restricted by taking into account dynamic constraints. In addition to this restriction, only velocities are considered, which are safe considering the obstacles. This pruning of the search space is done in the first step of the algorithm. In the second step, the velocity maximizing the objective function is chosen from the remaining velocities. In the

current implementation, such a cycle is performed every 0.25 seconds. Now given the velocity space, we have to choose the velocity V_{max} for which we get the largest value of the objective function.

The target heading (v, ω) measures the alignment of the robot with the target direction. It is given by $180 - \theta$, where θ is the angle of the target point relative to the robot's heading direction.

Communication between the Robots and Sentry

In the given scenario, the two robots in the arena have some tasks to be operated individually and some tasks which are communicated across robots. To tackle this, we plan to set up three independent rosmasters, one on each robot and one on the sentry system computing device. In this way, some topics like the pose of the robot, referee system data and other such individual topics and services are handled by the individual robot's rosmaster.

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. This way there will be a central brain of the whole system which will collect and control 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 surrounding situation, added latency may lead to falling prey to enemy attack.

To avoid this, we would setup a communication link between the three rosmasters for information transmission and retrieval across robots. The communication system is broadly known as multimaster communication. In this we would implement algorithms as in foreign relay and wifi comms with slight modifications to suit to the problem statement. The former implementation is based on TCP communication protocol while the latter is based on UDP communication protocols. The main

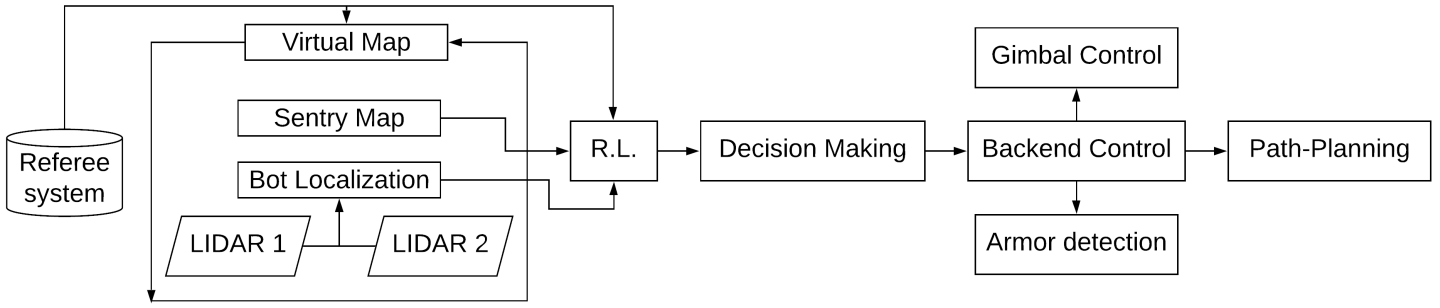


Fig. 13. Reinforcement learning module

advantage of multimaster communication is the guarantee of a robust communication system which is immune to unexpected communication break and other cases of robot going offline as when the health of a robot goes down or one robot ending up in a damage unexpectedly. This approach also handles the limited bandwidth of the communication system and almost removes latency. The central rosmaster in this case will only be responsible to transmit the enemy robot pose as estimated by the sentry system camera.

The hardware to be used supports a wide bandwidth and guarantees a stable connection. The algorithm in itself being robust enough ensures proper communication among the robots and sentry station.

Decision-Making

Reinforcement Learning

We will implement reinforcement learning algorithms for strategy planning and taking intelligent decisions in the arena. For the purpose of this competition, conventional supervised learning methods are not suitable because the actual arena conditions and scenarios that the bot would face will vary from the training zone for the robot. The strategies used by the opponent team will be unknown to us and the bot will have to make decisions on the spot. The number of strategies that can be followed by the enemy robot is practically huge and it is not feasible to account for all possible strategies possible. But the strategies used by the robots can be generalized in several ways. Hence, we plan to simulate dueling of robots using Q learning to discover and implement other such strategies. The reason for narrowing down to using RL for the decision-making subsystem is to account for the endless possible states in the arena during the match, something very difficult to implement in comparison to traditional methods such as behavior trees. The RL algorithm will form the backbone of the entire library of functions and will be responsible for decision making according to the dynamics of the environment. The RL will encompass obstacle avoidance for motion planning, motion strategy

for Buff/ DeBuff zone and will also call the automated shooting algorithm.

Algorithm Choice

For the core decision making part, behaviour trees and RL algorithms including the Q-learning methods and the policy gradient (PG) ones were some options. Due to the enormous number of possibilities that the arena can offer (despite its centre symmetry), it would prove to be extremely difficult to account for each and every situation arising during a match. From the RL implementations, PG algorithms allow for the modeling of the infinite state space and alongside provide an infinite action space output. Q-learning algorithms, on the other hand, can either be modeled as to provide for a discrete state space (using Q-tables) or a continuous infinite space (using the deep Q-networks) [4], but only provide for a discretized action space, which would ultimately result in discrete state space. With the arena having center symmetry, a stochastic policy would be preferential, since a deterministic policy would lead to actions being taken following one of the symmetrical situations. PG algorithms, due to their direct learning of the policy consume a lot of computation which may lead to increased latency. Additionally, since the localization and path planning modules are being utilized separately with the ROS environment, our task at hand would be restricted to decision making, for which the discrete action-state space of Q learning would be beneficial. Since, using deep Q-learning models, the infinite state space characteristics can be added, an architecture based upon that is to be implemented. For adding stochasticity to our model, which is essential due to reasons explained above, a controlled random selector would be implemented at the end of the DQN model, which takes into account a few of the Q-A pairs, depending upon the differences among them, and selects one randomly.

Simulation

Currently, we are using pygame to build the simulation environment. The environment includes all the

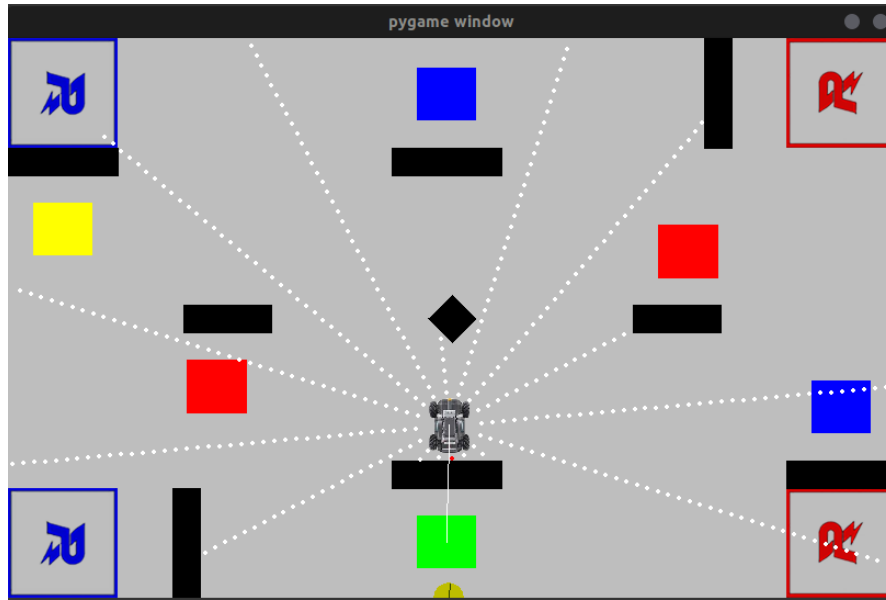


Fig. 14. Reinforcement learning module

obstacles. The reward function is defined to avoid the obstacles and navigate through the arena and reach the goal optimally. The reinforcement learning module is trained in the built environment. According to the problem statement, there are buff zones and debuff zones apart from the obstacles and enemy robots. To handle debuff zones in the simulation, the robot is given punishment equally towards going to obstacles and debuff zones. Currently, once a start and goal point is decided, the model effectively avoids obstacles in the path and chooses safe locations to navigate. As a demo, the model starts and finds the way to a buff zone. The demo video of the simulation for 300000 iterations is attached to the proposal.

Motion strategy for buff/debuff zones

In the arena, we will be treating debuff zones as obstacles that will be avoided by the above path planners and the decision-making model will be rewarded when in buff zones. The incentive mapping and the obstacle mapping for the complete software module will be updated every one minute, thus accounting for the changing locations of buff/debuff zones. When landed upon buff or debuff zones, the message from the referee system will account for the corresponding penalty or bonus as per the rules. The present reinforcement learning decides buff zone as one of the goal states and avoids debuff zones.

VI. DETAILS OF TEAM

All the team members are based at the Indian Institute of Technology, Kanpur.

- 1) **Prof. Laxmidhar Behera:** Faculty Advisor
Professor, Department of Electrical Engineering

Prof. Laxmidhar Behera is a very renowned professor of Indian Institute of Technology Kanpur. Prof. Behera 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.

- 2) **Ashok Kumar Chaudhary:** Founder and Embedded System Group
Senior Undergraduate, Electrical Engineering
Ashok is one of the Ex-coordinators of Electronics Club IITK. Ashok has experience in machine-learning, embedded systems, designing the Shooting Mechanism and Gimbal control, working of Main Control Board and AHRS and, mechanical design of required attachments.
- 3) **Suraj Hanchinal:** Founder and Algorithm Group
Senior Undergraduate, Electrical Engineering
Suraj is the Ex-coordinator of Electronics Club IITK. He has worked in the field of FPGAs, computer vision and machine learning. In ERA IITK, he works on computer vision and reinforcement learning-based strategy for the robot.
- 4) **Suryansh Agarwal:** Founder & Algorithm Group
Senior Undergraduate, Electrical Engineering
Suryansh has been the former team captain of the Team AUV-IITK, a team for autonomous underwater robotics at IITK. He has experience as an intern in a multinational company in the trending field of

Artificial Intelligence and Internet of Things. He is well-versed in developing the software pipeline of the robot and analyzing each aspect in great detail.

- 5) **Soumya Ranjan Dash:** Captain, Algorithm Group
Junior, Electrical Engineering
Soumya is one of the Coordinators of Electronics Club IITK. In ERA IITK, Soumya works on ROS and implementing simulations of the arena. He also works on path planning algorithms and decision making module.
- 6) **Karan Vaish:** Captain, Algorithm Group
Junior, Mechanical Engineering
Karan is an ex-member of Electronics Club IITK. He has gained experience in implementing image processing techniques by working on various projects. In ERA IITK, he works on object detection algorithms through CNN architectures.
- 7) **Shobhit Jagga:** Algorithm Group
Junior, Computer Science and Engineering
Shobhit is an ex-member of Electronics Club IITK. In ERA IITK, he has works on ROS, implementing the trigger for shooting mechanism after detection and tracking of target by CNN architectures.
- 8) **Aviral Khare:** Captain, Business Group
Junior, Aerospace Engineering
Aviral is one of core member of Techkriti, IIT Kanpur, one of Asia's largest technical festivals. He is responsible for the management and outreach of the team.
- 9) **Sourish Mondal:** Algorithm Group
Sophomore, Mechanical Engineering
Sourish has experience in Algorithms and micro-controllers. In ERA IITK, he works on Image processing, ROS and Designing.
- 10) **Sparsh Agarwal:** Embedded Systems Group
Sophomore, Mechanical Engineering
Sparsh has experience in image processing and Machine Learning techniques. In ERA IITK, he works in sensor data acquisition and processing.
- 11) **Dipak Miglani:** Embedded Systems Group
Sophomore, Aerospace Engineering
Dipak is an ex team member of the team Aerial Robotics, IITK. He has experience in micro-controllers and works on the same in ERA IITK.
- 12) **Luvneesh Kumar:** Embedded Systems Group
Sophomore, Biological Sc. and bioengineering
Luvneesh is an ex team member of Team Robocon IITK. As a part of the team he has gained experience in implementing various Image processing and Reinforcement learning algorithms.
- 13) **Vaibhav Vardhan:** Algorithm Group
Sophomore, Electrical Engineering

Vaibhav, from his previous projects has gained experience in Reinforcement learning and machine learning applications. He has also worked on ROS and image processing as part of the team.

- 14) **Muskan Agarwal:** Algorithm Group
Sophomore, Electrical Engineering
Muskan has experience in Image Processing, and Reinforcement Learning. In ERA IITK, she works on ROS and object detection through CNN.
- 15) **Haren Bhandari:** Algorithm Group
Sophomore, Aerospace Engineering
Haren is an ex team member of Team Robocon IITK. He has experience in image processing, deep learning and ROS.
- 16) **Harshit Sinha:** Business Group
Sophomore, Materials Science and Engineering
Harshit is a Senior Executive at Techkriti, IITK and a member of media council, IITK. He is looking after marketing and outreach of the team.
- 17) **Ashish Tiwari:** Business Group
Sophomore, Mechanical Engineering
Ashish is a member of media council, IITK. He is looking after Design and Marketing of the team.

VII. RELEVANT WORK DONE

- Simulation of RL decision making : <https://www.youtube.com/watch?v=x2HjnA480pI&feature=youtu.be>
- Vision marker detection : <https://youtu.be/Zp4DDKI-zuk>
- Elements of Intelligent robot : <https://youtu.be/o8hk3Alfqcg>

References

- [1] Andrew G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: (Apr. 2017).
- [2] Armin Hornung et al. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees". In: *Autonomous Robots* (2013). Software available at <http://octomap.github.com>. doi: 10.1007/s10514-012-9321-0. url: <http://octomap.github.com>.
- [3] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "The Dynamic Window Approach to Collision Avoidance". In: *Robotics Automation Magazine, IEEE* 4 (Apr. 1997), pp. 23–33. doi: 10.1109/100.580977.
- [4] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].