2024 Tenth Indian Control Conference (ICC)
9 - 11 December, 2024
IISER Bhopal, India

A Modular Training Approach for Multi-Robot Cooperative Search in Dynamic Environments

Yogesh Dasgaonkar, Radhe Shyam Sharma, Raj Nayan Samant

***Abstract*— In a multi-robot system, two or more robots are trained to cooperate in their actions to complete a given task faster and with a higher probability of success than in a single-robot system. State-Of-The-Art (SOTA) cooperative multi-robot training methods are of two types: (a) Algorithmic: Greedy, Genetic, and (b) Machine Learning (ML): Regression, Reinforcement Learning (RL), Deep RL (DRL), and Inverse RL (IRL). Existing literature shows they perform well when the environmental conditions are well-known and static, for instance, in a car body part assembling task.**

To test the performance of the SOTA under dynamic conditions, we simulate a 3D fire disaster environment in a real-world urban city building with humans trapped inside it. We model the fire dynamics such that the intensity and spread rate of fire, smoke, and blocked paths increase non-linearly. The trapped humans have a high risk of losing their lives if not searched within a deadline. We state this problem as the Multi-robot Cooperative Search in Dynamic Environments (MCSDE) problem. When we applied the SOTA approaches to train multi-robots to search for human lives in this simulated environment, they failed to search for all human lives within a deadline and lacked computational efficiency.

To address the issue of multi-robot training in MCSDE, we present a modular training approach. In this approach, we train each robot locally based on their local perceptions. Our efforts in this research are to uncover these perceptions' geometric locality properties. When fused with another robot's perception, each perception reveals the true behaviour of the environment's dynamic. Each robot becomes self-aware of global dynamics by training from its local perceptions. The proposed modular approach requires just one-fourth of the computation than the RL or DL-based training approaches and performs far superior to the algorithmic approaches.

***Index Terms*— multi-robot, search and rescue, collaborative task planning, Simulation, robot programming, robot control**

I. Introduction

Background: The critical part of the search and rescue operation is making timely and accurate decisions. A wrong decision taken or delay in decision making may lead to failure in saving a human life or loss of the rescuing human or robot. Human decisions are often subjective in search and rescue use-cases. Even an experienced person cannot guarantee success of the search and rescue operation due to the environment dynamics. We define a *multi-robot system* as a group of heterogeneous robots that cooperate with one another to solve the given problem. In this paper, we confine our problem to - *Searching for humans trapped inside a fire calamity within a deadline, where the fire dynamics are non-linear*. Hereafter, we shall refer to this problem as the MCSDE problem.

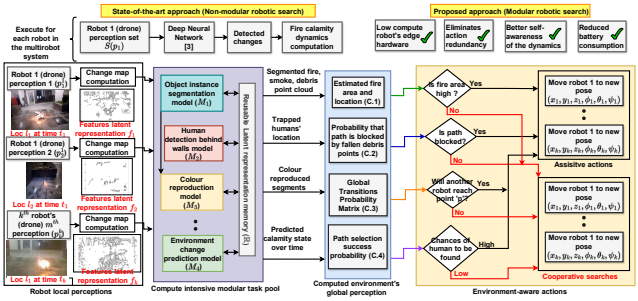


Fig. 1: *The proposed perception based modular training framework*

Existing research: Currently, there is a lack of understanding of how multi-robot systems can be efficiently deployed to solve the MCSDE task. Several State-Of-The-Art (SOTA) systems [1], [2] were proposed to solve the MCSDE problem. While these systems have proposed ML [3], RL [4], [5], DRL [6], IRL [7], regression [8], genetic [9], greedy [10] and several polynomial bound algorithmic methods, to solve the MCSDE problem, they mainly focus on the existing trivial data and their current experimental conditions. The real-world dynamics are very different from the simulation conditions. Analyzing these systems, we found the following significant limitations. The SOTA fails to address how the search and rescue dynamics, including fire, smoke, heat, and path blockage, affect the robots' real-time computation concerning sensing, decision-making, and motion. They do not address the multi-robot damage risk due to fire. Training a multi-robot system for such a dynamic system is challenging due to the limited battery energy and onboard computing resources on the robot.

This research addresses this problem by proposing a modular training approach for search multi-robots. The idea is that multiple independent perceptions provide better inference into the environment's dynamics than one. We take data from the robot's Red Green Blue - Depth (RGB-D), Channel State Information (CSI) and thermal sensors and represent them as sparse 3D perception matrices. These perception matrices exhibit the geometric locality property. We independently train multiple robots at various search locations to handle the robotic computation overhead on only one feature at a time. Robots exchange their local perceptions and inferences with each other and the control center. Fusing the inferences from multiple robots reveals the entropy of the environment's

Yogesh Dasgaonkar, Radhe Shyam Sharma are with Centre for Artificial Intelligence and Robotics at Indian Institute of technology Mandi-175005, India (d22190@students.iitmandi.ac.in, radheshyam@iitmandi.ac.in).

Raj Nayan Samant is with Nvidia Graphics Private Limited, Mumbai, Maharashtra - 400093, India (rsamant@nvidia.com).

dynamics. The revealed knowledge is transferred to each robot. The next decision taken by each robot is environment-aware and maximizes the global probability of success of the search mission, as shown in Fig. 1.

A. Problem Statement

We are given the pre-calamity map of a fire calamity-struck environment where h humans are trapped. The environment's dynamics are such that the intensity and spread of fire, smoke and blocked paths change at a non-linear rate. Due to this fire dynamics, the trapped humans have a high risk of losing their lives if not searched and rescued on time. As a part of the search and rescue team, we are given a multi-robot system containing r robots (drone, husky, quadruped). The problem is to *devise an approach for deploying the multi-robot system to search the environment such that the success rate of searching human lives within a deadline is maximum* *Constrains*: The robots have limited onboard computing capabilities and battery energy. *Assumptions*: The robots are connected over the network to share their local perceptions.

B. Contributions

The key contributions are listed below,

1) Modular training framework: We present a modular training framework for solving the MCSDE problem. The framework requires low computation, provides fast training convergence, and has a higher success rate.

2) Mathematical representation of the real-world fire calamity dynamics: We present the mathematical model to represent the real-time fire and smoke dynamics. The robots use this model to estimate the fire and smoke spread, which affects their next action to be taken.

3) Environment-aware decision-making from perceptions: As the environment's dynamics change, the robots are environment-aware to take cooperative actions that maximize the probability of success. We provide a probabilistic approach for deciding the robot's next actions where each robot is aware of the environment, the consequences of its actions, and those of the other robots.

4) Benchmarking: We benchmark the performance of our modular approach with the SOTA on seven widely used simulators by researchers: Unity, GazeboSim, PyBullet, Mathworks Robotics, CoppeliaSim, Webots and IsaacSim.

II. Proposed approach

A. *Modular training framework*

This subsection provides the details of the research and development process of the proposed modular framework.

A.1 - Generating the perception matrix P: Each robot generates a $m \times k \times n$ perception matrix P consisting of 'm' sized sensor data values (RGB-D (image size with depth), CSI (subcarrier size), thermal (1)), 'k' robot actions, and a maximum of 'n' environmental points.

A.2 - Decomposing the perception matrix in modular units: Let $(k_1, n_1) \in \mathbb{R}^{d_1}, (k_2, n_2) \in \mathbb{R}^{d_2}$ and $(k_3, n_3) \in$

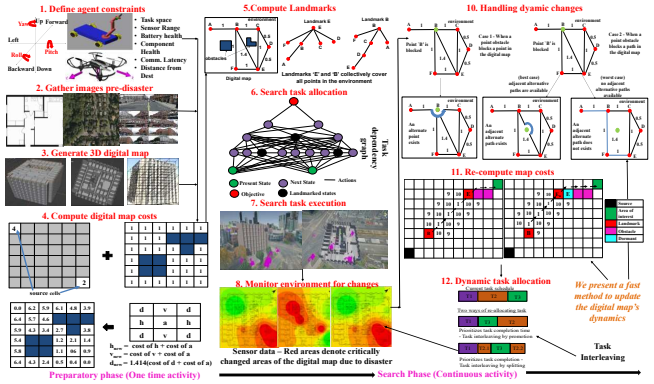


Fig. 2: *The figure summarizes the phases of the proposed approach for multi-robot cooperative search for fire dynamic environments*

$\mathbb{R}^{d_3}$ be vectors such that each pair k_i, n_i is linearly independent. Then, the perception matrix,

$$k_1 \otimes k_2 \otimes n_3 + k_1 \otimes n_2 \otimes k_3 + n_1 \otimes k_2 \otimes k_3 \quad (1)$$

has rank 3. Each modular unit of P will be of rank 3. We have the matrix $P \in \mathbb{R}^{m \times k \times n}$ for some integer $m > 1, k > 1, n > 1$. We show a way to decompose P into several modular parts such that P can be represented as $P = \sum_{j=1}^{r} \mathbf{v}_j^{\dagger} \otimes \mathcal{F}_j$, where $\mathbf{v}_j^{\dagger}$ is the complex conjugate of the canonical basis vector $\mathbf{v}_j$ in $\mathbb{R}^m$ and the $k \times n$ matrices of $\mathcal{F}_j$ are a modular partition of P. The $m_1 \times k_1 \times n_1$ matrix $\mathcal{P}_{k1}$ is a partition of P. By taking a low-dimensional matrix of arbitrary sizes, we arrange them using the Kronecker product $\otimes$. The number of decomposition are bounded by r.

A.3 - Cooperative perception sharing: We express the calamity environment as a set of points N of a graph with E as its edges. Then the j^{th} neighbour of robot 'r_i' is,

$$\mathcal{N}_{i,j} = \{p_j \in N : (p_i, p_j) \in E_i\} \quad (2)$$

A set of all neighbours $S(j)$ in the neighbourhood of at least one robot 'r_i' inside a cluster 'C' such that $C \subseteq E$ is,

$$\mathcal{N}_{i,S(j)} = \{p_j \in C : p_i \in C, (p_i, p_j) \in E_i\} \quad (3)$$

A robot that does not satisfy Eq. (3) is considered a non-neighbouring robot. For cooperative information sharing, bringing diversity to the exchanged perception module is necessary. Therefore, each robot must exchange one modular unit of its perception matrix with at least one neighbouring and one non-neighbouring robot. This exchange occurs for every modular unit of the perception matrix we decomposed.

A.4 - The modular task pool: The modular task pool has four compute intensive robot training tasks,

(1) Object instance segmentation: We have a set of input image frames N_I from the camera feed of the multi-robot search system $X = \{x_i\}_{i=1}^{N_I}$. Every image has associated labels providing an accurate location of the objects of a given class within an image. We define 43 object classes, including fire, smoke, debris, human, wall, signboards, furniture, highly, medium flammable and non-flammable. We design model M_1 (depicted in Fig. 1) to segment the fire instances and the objects near them to estimate the fire spread. Mathematically we formulate our model as, let $q = \{c_i\}_{i=1}^{N_c}$ where $N_c \in \mathbb{N}$ be a vector set containing N_c classes and the count c_i of each class to be segmented, where

$c_i \in \{\mathbb{N} \cup \{0\}\}^I$ and $I \in \mathbb{N}$. We aim to find a set $s \subset q$ such that $|s| = \lfloor b_1 N_c \rfloor$ where $b_1 \in (0, 1)$. For $i \in \mathbb{N}$, we consider x_i as a binary decision variable indicating whether $c_i \in s$.

$$\text{To find: } P_q = \frac{\sum_i c_i}{||\sum_i c_i||}, \; P_s = \frac{\sum_i x_i c_i}{||\sum_i x_i c_i||} \tag{4}$$

$$P_{q \setminus s} = \frac{\sum_i (1 - x_i) c_i}{||\sum_i (1 - x_i) c_i||}, \; \sum_i x_i = \lfloor b_1 N_c \rfloor \tag{5}$$

$$\text{To maximize: } \cos(P_s, P_q) + \cos(P_{q \setminus s}, P_q) \tag{6}$$

(2) Human detection behind walls: Model M_2, depicted in Fig. 1, is derived from our previous work [11] that estimates the position of the trapped human behind walls by recognizing the changes in the CSI patterns as the trapped human makes minor movements during the calamity.

(3) Colour reproduction: During the fire calamity, objects lose their true colour due to their burning and smoke/flame captured by the robot's camera, thereby hampering the accuracy of the actual object instance detection. We implement a colouring reproduction autoencoder model M_3 as shown in Fig. 1, trained to learn the encoding and decoding process from input discoloured images and their true coloured counterpart. The model's encoder $e(.)$ parametrized by ψ and a decoder function $d(.)$ parametrized by ω. The latent representation learned for input image X is $\mathbf{z} = e_\psi(\mathbf{X})$ and the reconstructed input is $\mathbf{X} \approx d_\omega(e_\psi(\mathbf{X}))$. The loss function is given as,

$$L_{M_3}(\omega, \psi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{X}^{(i)} - d_\omega(e_\psi(\mathbf{X}^{(i)})))^2 \tag{7}$$

(4) Environment change prediction: Predicting fire and smoke spread over time from RGBD camera images is challenging due to the varying camera alignments. We design a model M_4 as shown in Fig. 1 that takes time series image data and outputs a time-varying point cloud of the fire and smoke regions. From these point clouds, we derive (in subsection II-B) the model for predicting fire and smoke area spread dynamics. The process to generate point clouds from RGB-D cameras are,

$$x = z(p_x - m_x)/fs_x, \; y = z(p_y - m_y)/fs_y, \; z = d \tag{8}$$

where p_x and p_y are pixel coordinates, m_x and m_y are the camera's centre, s_x and s_y are the scale factors relating the width x and height y of the sensor to the pixel count respectively, f is camera focal length and d is the depth.

A.5 - The modular learning process: The below conditions apply to the gradient operator ∇, for some $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$, $\nabla(f + g)x = (\nabla f)x + (\nabla g)x, \; \forall x \in \mathbb{R}$ and $\nabla(kf)x = k(\nabla f)x, \; \forall k, x \in \mathbb{R}$. These linearity conditions restrict the gradient operator, meaning the functions are evaluated at the same point x in the domain. If the gradient is linear, so will be its derivative.

Here, we shall recall the two linearity conditions: (a) A linear map is a mapping between two vector spaces that preserves vector addition and scalar multiplication. (b) The straight line function is $f(x) = cx + k$. A function may not be a linear map; it can still be a straight line. We define a neural network as a composition of non-linear functions as,

$$y'_\theta = f^l_{\theta_l} \circ f^{l-1}_{\theta_{l-1}} \circ \ldots \circ f_{\theta_1}, \tag{9}$$

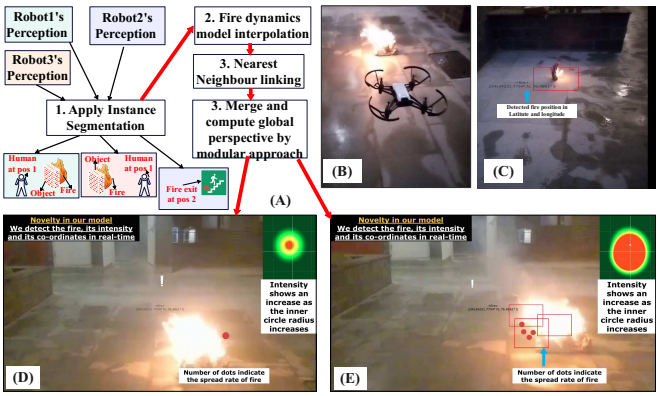


Fig. 3: *Understanding fire dynamics through perceptions: (A) Block diagram for detecting fire intensity and spread. (B) A Tello drone's perceptions with a camera to detect the fire. (C) The proposed system localize the fire. (D) The fire intensity and spread is changing. (E) The system estimates the rate of fire spread and the rate of change in fire intensity.*

where $f^l_{\theta_l}$ is the i^{th} layer of a neural network that outputs a non-linear function, θ_l is a parameter vector associated with the l^{th} layer, L denotes the number of layers, y'_θ denotes the neural network, θ is a vector containing all neural network parameters, $y'_\theta(x)$ is the output of your neural network and $\circ$ is the composition operator. We use the sum of squared error as the **loss function** for our neural network,

$$\mathcal{L}_\theta(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \mathcal{S}_\theta((\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i), \mathbf{y}_i) \tag{10}$$

$$= \sum_{i=1}^{N} (\mathbf{y}_i - y'_\theta(\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i))^2 \tag{11}$$

where $\mathbf{x} \in \mathbb{R}$ is the input vector, $\mathbf{y} \in \mathbb{R}$ is the label vector, y_i is the label for the i^{th} input x_i. We ask the question - *Why simply using a mini-batch processing model [12] does not work in cooperative multi-robot training?* Since we show that the gradient is a linear operator, is computing the gradient of the sums equivalent to computing the sum of the gradients? In our perceptions, we are summing $\mathcal{S}_\theta((\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i), \mathbf{y}_i)$ where $(\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i) \neq (\mathbf{m}_j, \mathbf{k}_j, \mathbf{n}_j)$, for $i \neq j$. S_θ is being computed at different points of the environment. However, the definition of a linear map applies only when the perceptions are to be evaluated at isomorphic points in the environment. However, in the case of the neural networks defined in Eq. (9) and loss function in Eq. (10), the properties of the linear operator for the gradient will not apply because we are evaluating the loss at different points of the environment. In the case of modular units u and r robots where $u \leq r$, this is actually a sum of the gradients. Let's assume we uniformly sample an arbitrary perception sample $(\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i)$ from the sensor dataset $\mathcal{D} = \{(\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i)\}_{i=1}^N$. Formally,

$$\nabla_\theta \mathcal{L}_\theta((\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i), \mathbf{y}) = \mathbb{E}_{((\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i), \mathbf{y}_j) \sim \mathbb{U}} [\nabla_\theta \mathcal{S}_\theta] \tag{12}$$

where $\nabla_\theta \mathcal{S}_\theta$ is the gradient of $\mathcal{S}_\theta$ for model parameters θ and $\mathbb{E}_{((\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i), \mathbf{y}_j) \sim \mathbb{U}}$ is the expectation of the random variable of sample $(\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i)$ from the uniform distribution $\mathbb{U}$. We represent Eq. (12) as,

$$\nabla_\theta \mathcal{L}_\theta(\mathbf{x}, \mathbf{y}) = \nabla_\theta \mathbb{E}_{((\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i), \mathbf{y}_j) \sim \mathbb{U}} [\mathcal{S}_\theta] \tag{13}$$

After simplification, we arrive at,

$$\nabla_\theta \mathcal{L}_\theta((\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i), \mathbf{y}) = \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{S}_\theta((\mathbf{m}_i, \mathbf{k}_i, \mathbf{n}_i), \mathbf{y}_i) \tag{14}$$

This shows that the gradient of θ of the loss function in Eq. (10) includes perceptions from all the robots and is equivalent, in terms of expectation, to the gradient of $\mathcal{S}_\theta$ which is the loss function of one modular unit. So for each module we forward propagate, calculate error and calculate all gradients independently to obtain the results. We calculate the gradient of the loss function with respect to each perception individually. We then average the gradients across subset of m perceptions at the robot's hardware or all m perceptions at the central control server to estimate a true gradient. We perform this cycle for each robot, and then average all gradients and apply the weight change. **We save significant memory** by summing gradients either at the robot's end or all of then at the central control. Once we have calculated the gradients for one perception for the weights of one layer, then we do not re-use the individual gradients again, we just keep the sum. Since we are calculating the gradients at individual robot's hardware in parallel, as each perception is independent, **we achieve a speedup in training time**, which explains how the modular approach is so effective in multi-robot training.

B. *Mathematical representation of the fire dynamics*

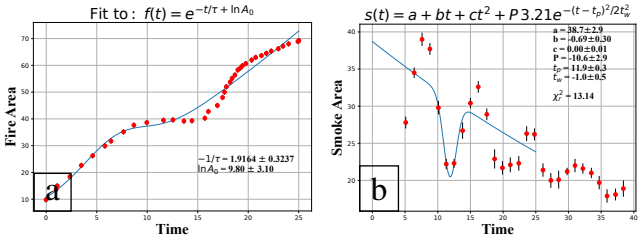


Fig. 4: *These graphs depict the non-linear (a) fire and (b) smoke dynamics in time.*

The robot perceptions capture the fire area, heat, smoke and path blockage. With the fire dynamics model one can estimate the fire intensity and the fire spread rate. We model the fire dynamics shown in Fig. 3 as,

$$f(t) = e^{-t/\tau + \ln A_0} \tag{15}$$

which is a non-linear function in time t, A_0 is the fitting parameter governing the increase in the fire area A per unit of time and τ determines the spread rate. Similarly, we model the smoke dynamics shown in Fig. 4 as,

$$s(t) = a + bt + ct^2 + E_f P e^{-(t-t_p)^2/2t_w^2} \tag{16}$$

where t is the time taken for the fire to spread, t_p is the time at which the fire area is the highest, t_w resembles the time width for which the peak remains, P is the peak fire intensity above the baseline, E_f is the environment factor that varies depending on the environment dominant object (wood, cotton, cement) and type (urban, forest), and a, b, c are the fitting parameters. Heat follows the same non-linear dynamics as fire but provides a lower damage percentage to humans or robots, as depicted in the simulation model in Table II. These parameters determine the rate at which fire and smoke will spread in the building, causing loss of lives.

C. *Environment-aware decision making from perceptions*

C.1 - Estimating the Fire Area (FA) from captured local perceptions: Let $\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_l$ denote l points in the point cloud outputted by model M_4. The sum of cross products of $k \subseteq l$ points in the cloud is given as $\vec{A_k} = \frac{1}{2}\sum_{k=1}^{n} \vec{p}_k \times \vec{p}_{k-1}$. FA depicted in Fig. 3 is given as,

$$\text{Fire Area} = \frac{1}{l} \left\| \sum_{k=1}^{k=l} C\vec{A}_k \times C\vec{A}_{k+1} \right\| \tag{17}$$

$C = \frac{1}{l}\sum A_k$ is the centre of gravity of the point cloud.

C.2 - Probability of arriving at a path blocked due to debris: Consider our search environment to be a $n \times n$ grid with one point p per grid. A point may be blocked by fallen debris with probability p and not by probability $1-p$. Let E_1 be the event that there exists a blocking path from robot R_1 to R_2 via path P_1 and E_2 be the event that there exists a blocking path from robot R_1 to R_2 via path P_2. After the cooperative perception exchange, robot R_1 obtains information regarding event E_1, and calculates the probability of event E_2 such that $P(E_2) = P(E_2 \mid E_1)$,

$$P(E_2) = \lim_{t\to\infty} P(\sigma^t \in E_2) \leq \lim_{t\to\infty} P(\rho^t \in E_2) \tag{18}$$

Here, σ and ρ are two marker classes in the map to indicate the presence or absence of debris in a given grid, respectively. σ^t and ρ^t are the coupled chains defined by Markov process [13]. They satisfy $\sigma(n) \leq \rho(n)$ for all the n grid points.

C.3 - Probability of the robot to reach a desired point from the other's robot's perspective: We define a transition matrix T of probabilities for the robot's movement between two points, $\begin{bmatrix} P_{0\to0} & .. & P_{k\to0} \\ P_{0\to1} & .. & P_{k\to1} \\ .. & .. & .. \\ P_{0\to k} & .. & P_{k\to k} \end{bmatrix}$. Let $\vec{w}^{(0)} = \begin{bmatrix} P_0 \\ P_0 \\ .. \\ P_k \end{bmatrix}$ be the initial probability matrix of the robot starting at point 0. The robots exchange T after each movement action. We have $\vec{w}^{(n+1)} = Tw^{(n)}$. The probability that the robot has visited point k is $\sum_{n=0}^{n=\infty} \vec{w}_k^{(n)}$, where $\vec{w}_k^{(n)}$ denotes the k^{th} element of w, starting from the 0^{th} element. Since we assume no backtracking in the calculations, the robots do not move in the opposite direction of the destination. We stop our algorithm when all elements of $w^{(n)}$ are zero.

C.4 - Probability of selecting a path with the highest search success: Based on the search success probability, the robot will either move towards the spreading fire areas to assist another robot or explore another path searching for trapped humans. With equal probability, the robot can move in either of the eight directions (N, S, W, E, NE, NW, SE, SW). Fire, blocked path, heat and smoke are obstacles in the robot's path. Given a set of possible path options, the goal is to compute each path's search success probability. The robot chooses the path with the highest probability of search success. We divide the search area into uniquely numbered tiles to derive the equation for this probability. Let p_k be the probability for success if the robot starts on the k^{th} tile and performs the search. To obtain the solution

we apply recurrence the function $p_k = \frac{1}{8}(p_{k-1} + p_{k+1})$ with the boundary conditions $p_0 = 0$ (source) and $p_n = 1$ (destination). Rewriting, we get $p_{k+1} - pa_k + p_{k-1} = 0$. This equation resembles the form $p_k = A\lambda^k$ where λ is the root of the quadratic equation $x^2 - 8x + 1 = 0$. Solving it, we get $p_k = a_1(4 + \sqrt{15})^k + b_1(4 - \sqrt{15})^k$. a_1 and b_1 are the parameters governed by real-time calamity conditions, yielding a reasonable estimate of searching success.

III. Results and Discussions

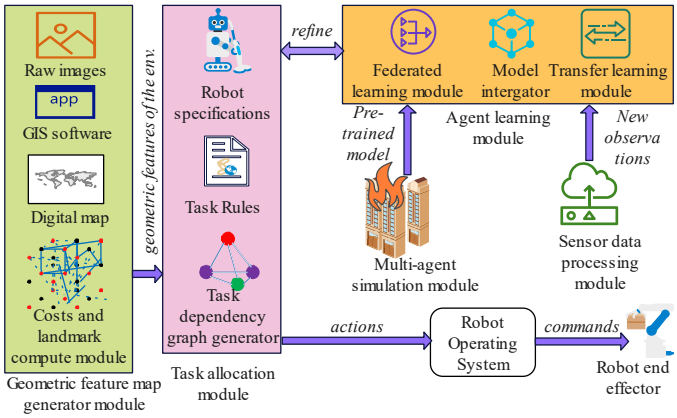


Fig. 5: *Implementation of the proposed multi-robot modular training system.*

TABLE I: *Table depicting the specification of the simulation system used for performance evaluation*

Parameter	***Specifications***
Test environment area size	The indoor environment has, on average, 40 rooms with an average area of 15 m² per room
Environment point size	Four indoor environment maps of size 50k, 100k, 200k, 300k points
Simulator test environment (Fig. 6)	Around 11287 3D objects, including walls, floors, doors, furniture, stairs, shafts, columns, and 50 multi-robots (Turtlebot)
Compute resources	Intel Xeon 4110 CPU 2.10 GHz X 16 cores, 62.5 GB of memory, Ubuntu 20.04.06/Windows 11 with Nvidia GeForce RTX 3080
Fire dynamics simulator	Our Python program for randomly generating non-linear fire dynamics from our proposed model in contribution 2
Test Plan	Ten humans are trapped at random locations in the test environment with spreading dynamics mentioned in Table II. The multi-robots must search all the humans alive.
Total Simulation run time	50 hours, including training time for the robots
Simulation frequency	Simulation was run three times and maximum values were taken

A. Performance Evaluation Metrics

Machine-independent metrics - These metrics measure the performance of the multi-robot system as they solve the MCSDE problem. These metrics do not depend of the computation hardware or the simulation application used. Task Success Ratio **TSR** is the ratio of the number of search tasks given to the number of the search tasks completed where all the trapped humans are still alive. Total number

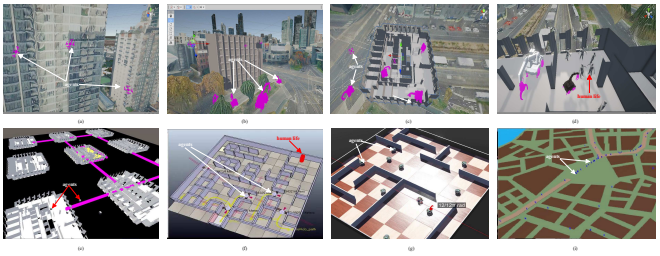


Fig. 6: *The screenshots convey the steps of the algorithm. (a) The nine-story building is the fire environment. (a), (b), (c) is the multi-robot system consisting of drones, mobile manipulators and quadrupeds are searching the environment cooperatively for trapped human lives (d) Finally the robots have successfully searched for the human lives, Multi-robot simulation run on (e) Unity, (f) Coppeliasim, (g) Webots and (h) Pybullet.*

of Human Lives Lost **HLL** is the number of deadline misses due to fire spread or smoke or falling debris leading to the loss of a human life. Total Backtracks **TB** is the total number of times the multi-robot system backtracks to an already searched path. Backtrack Path Length (meters) **BPL** is the total path length of the total number of robot backtracks. Search Task Completion time (min) **STC** is the time required by the multi-robot system to search for all trapped humans.

Machine-dependent metrics - These metrics measure the time and space resources required to train the multi-robot system for solving the MCSDE problem and depend of the computation hardware and simulation application used. Multi-robot training time is time taken to train the multi-robot system to achieve a 60% task success ratio. Maximum memory usage is the memory taken to train the multi-robot system to achieve a 60% task success ratio.

TABLE II: *This table specifies how we simulate the calamity's effects on the environment's dynamics.*

Dynamics	**Spread rate**	**Damage radius**	**Damage rate**
Fire	5%	50 points	25%
Heat	5%	50 points	5%
Smoke	5%	150 points	0.5%
Path blocked	Permanent	50 points	0%

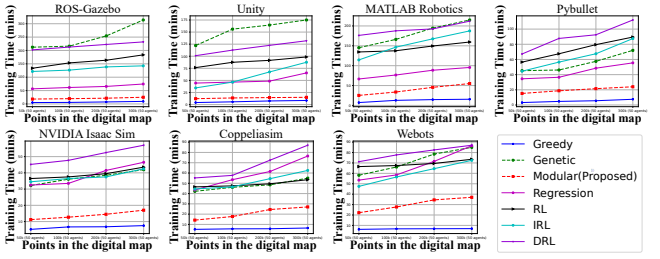


Fig. 7: *The time required in training the multi-robots on the seven simulators.*

B. Experimental Setup

For benchmarking and training the multi-robot system to be better prepared to handle the real-world fire dynamics, we create a 3D model of a real world building as shown in Fig 6 in Unity. The goal of this setup is to monitor the location for fire calamity. We add multiple robots (five drones, three quadrupeds and four mobile manipulators) equipped with

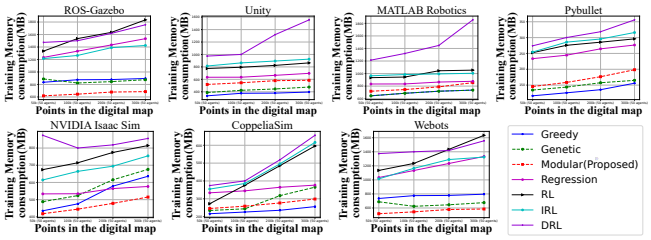


Fig. 8: *The RAM usage for multi-robot training on the seven simulators.*

RGB-D camera and a control center near this building. The recordings of our simulation runs is provided in Github [14]. Table I provides the simulation details.

C. Obtained Results and comparison

Fig. 7, 8, Tables III provide the obtained results. **TSR and HLL** - The results are given in Table III. We see that proposed modular approach of robot training has the highest TSR. HLL gives a granular understanding of how an approach performed even though it wasn't able to save all human lives but only some of them. Among the algorithmic approaches, the greedy method performed the worst. *This indicates that multi-robots trained using modular approach has the highest probability of rescuing all of the human lives.* **TB and BPL** - Wrongly assigned policy, rule, heuristics or reward functions caused the robot to search location where humans were not present. They backtracked and wasted time. *Results show that the proposed approach has the least number of backtracks and the least backtrack path length.* **STC** - Robots consumed time to process information from the RGB-D, CSI and thermal sensors and decide upon the next action. Incorrect path explorations also consumed extra search time. The results indicate that algorithmic methods have the worst search task completion time. *The results indicate that the proposed modular approach has the least search task completion time among others.*

TABLE III: *Results of the machine independent metrics.*

Method/Metric	TSR	HLL	TB	BPL	STC
Greedy [10]	60	432	1123	234	15
Genetic [9]	83	232	567	213	14
Regression [8]	85	203	332	82	14
RL [5]	87	187	276	62	13
IRL [7]	90	163	342	68	12
DRL [6]	92.9	64	147	57	8.2
Modular (Proposed)	**97**	**32**	**67**	**23**	**8**

Multi-robot training time - Results in Fig. 7 indicate that the DRL algorithms take nearly 3 hours to train compared to greedy approach which take nearly 7 min of is a decrease by 96.5%. But the greedy method performs worse than the DRL in task success ratio by 31.36%. *The proposed modular approach requires 85% lesser training time than DRL, while having 4.1% higher task success ratio than the DRL approach*. ROS-Gazebo took highest time to train the model and the Pybullet platform took the least time.

Multi-robot training time memory usage - Fig. 8 concludes ROS gazebo requiring the most ram among all the simulator platforms. The best simulator for low-memory systems is the Pybullet. Genetic algorithms performed poorly in MATLAB.

IV. Conclusion

We address a challenging problem to train multi-robot system to search human lives during the fire calamity. The obtained results and comparative analysis indicate that the existing methods are not efficient to solve this problem, since they suffer from poor search success ratio, high training time and memory requirements. We provide a novel modular approach to train the robot using their local perceptions from RGB-D, CSI and thermal sensors. We present the mathematical representation of the real-world fire dynamics from these perceptions. We propose the self-aware model for cooperative robot search task control. Together with these three contributions, the proposed approach performs better than the state of the art both in terms of human-life search task success rate and requires low compute resources. We benchmark the MCSDE problem on seven simulators and present our key observations.

References

[1] L. Xu, Q. Yang, M. Qin, W. Wu, and K. Kwak, ''Collaborative human recognition with lightweight models in drone-based search and rescue operations,'' *IEEE Transactions on Vehicular Technology*, 2024.

[2] S. Hayat, E. Yanmaz, C. Bettstetter, and T. Brown, ''Multi-objective drone path planning for search and rescue with quality-of-service requirements,'' *Autonomous Robots*, 2020.

[3] Y. Dasgaonkar and R. S. Sharma, ''Through the rubble: Discovering the invisible,'' *IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2024.

[4] Z. Li, M. Lanctot, K. R. McKee, L. Marris, I. M. Gemp, D. Hennes, K. Larson, Y. Bachrach, M. P. Wellman, and P. Muller, ''Search-improved game-theoretic multiagent reinforcement learning in general and negotiation games,'' in *Adaptive Agents and Multi-Agent Systems*, 2023.

[5] A. Colom and C. Torras, ''Ordinal inverse reinforcement learning applied to robot learning with small data,'' *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[6] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, ''Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments,'' *IEEE Robotics & Automation Letters (RAL)*, 2019.

[7] F. Shkurti, N. Kakodkar, and G. Dudek, ''Model-based probabilistic pursuit via inverse reinforcement learning,'' *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[8] C. Ye, J. Yang, and H. Ding, ''Bagging for gaussian mixture regression in robot learning from demonstration,'' *Journal of Intelligent Manufacturing*, 2020.

[9] D. R. Maddi, A. F. Sheta, A. M. Mahdy, and H. Turabieh, ''Multiple waypoint mobile robot path planning using neighborhood search genetic algorithms,'' *International Conference on AI, Robotics and Control*, 2019.

[10] M. Corah and N. Michael, ''Efficient online multi-robot exploration via distributed sequential greedy assignment,'' in *Robotics: Science and Systems*, 2017.

[11] Y. Dasgaonkar and R. S. Sharma, ''System for real time human detection & counting in a closed room environment and a method thereof,'' *IN Patent 549539*, 2024.

[12] K. Lee, A. N. Cheng, C. Paquette, and E. Paquette, ''Trajectory of mini-batch momentum: batch size saturation and convergence in high dimensions,'' *International Conference on Neural Information Processing Systems (NIPS)*, 2022.

[13] F. Nawaz and M. Ornik, ''Multiagent, multitarget path planning in markov decision processes,'' *IEEE Transactions on Automatic Control*, 2022.

[14] Github. (2024). [Online]. Available: https://github.com/CAIRIITMandi/MultiRobotSimulatorBenchmarking.git