MTSO: Multi-Target Search Optimisation based on Probability Map

Aman Virmani
Department of Electrical Engineering
Delhi Technological University
Delhi, India
virmaniaman4@yahoo.com

Vayam Jain
Department of Applied Physics
Delhi Technological University
Delhi, India
jainvayam@gmail.com

Nilesh Aggarwal
Department of Applied Physics
Delhi Technological University
Delhi, India
nilesh.ggoo@gmail.com

Arjun Gupta
Department of Electronics Engineering
Delhi Technological University
Delhi, India
arjun222gupta@gmail.com

Anunay
Department of Mechanical Engineering
Delhi Technological University
Delhi, India
anunay2608@gmail.com

Dr. Anup Kumar Mandpura
Department of Electrical Engineering
Delhi Technological University
Delhi, India
kanup@dtu.ac.in

***Abstract*—This report presents an optimized, decentralized way of searching for targets using swarms of rotary based unmanned aerial vehicles with information available from onboard EO/IR sensors. The are three main objectives of the proposed algorithm: time-optimized multi-target search, maximum and optimized payload drop, and maximum area coverage. The real-time application of these UAVs is in Human Aid and Disaster Relief scenarios mostly where the survivors have to be identified and given relief material. The proposed controller is inspired by the multi-target particle swarm optimized method(MTPSO) and its limitations of convergence towards Pbest and Gbest which may not be the best option in HADR scenarios. The algorithm involves the target probability distribution of the cells present in the target area which keeps on updating dynamically as the UAVs dive deeper into the target area. The deep learning-based model for target detection is deployed on each UAV using a dual camera with a limited field of view and its target discriminability varies as a function of the Class of the target, environmental conditions, etc. These parameters are given input to the probability distribution method for generating a probability map using which the UAVs optimize their path. The algorithm is evaluated on Ardupilot's SITL platform for parameter tuning and simulation in various scenarios which are later compared with existing search methods.**

Keywords—Swarm, Unmanned Aerial Vehicles, Multi-target Search, Multi-agent systems, Particle Swarm Optimization, Probability map, k-means, Hungarian algorithm.

I. INTRODUCTION

The demand for Unmanned Aerial vehicles has increased in the past years, as they are capable of rapidly covering areas, and providing better surveillance and efficient monitoring areas which gives them an edge over other alternatives. Because of these characteristics, the role of UAVs in civilian and military applications has become popular. The ground target search problem is one of the most important and popular applications of UAVs. Over time, the problem has become more complex as the number of targets and the search area has grown; to solve these huge UAV swarms are used. In this type of mission, it is essential to provide aid quickly and find multiple targets as quickly as possible.

There have been multiple solutions[1] put forward by various researchers for the problem of multi-target acquisition. The most common is a simple exhaustive search, i.e. scanning the complete area using pre-defined trajectories. Although used successfully, this method has certain limitations. First, as the trajectories are pre-fed, they fail to adapt according to the dynamic situations, resulting in more time taken. Second, as the decisions are not being made autonomously, there is a need for the operations team to look after the mission which is both expensive and time-consuming.

There are also unique approaches based on heuristic methods like multiple target search area optimization [2] which have proven to give good results during operations. The algorithm begins with an exhaustive search, identifies the global best and personal best during the search, and changes their behavior according to them. This algorithm satisfies all three main objectives but has its limitations like the guidance of UAVs away from multiple targets if the global best is identified somewhere else. Also, the percentage of the area covered can be improved if additional lines of UAVs are added.

There are also other centralized approaches similar to our algorithm which involves dividing the target area into small n-cells [3],[4] and assigning a probability to each cell. As the UAVs move further into the search area the probability of the cells keeps on updating depending on the targets found by them. This creates a dynamic updating probability map of the search area. This method adapts according to the situation but has its limitations. First, the centralized controller stores data from all agents and this requires a robust communication architecture.

In this report, we propose a decentralized swarm controller which is based on the probability method which has been proven to give good results but its application in multi-target searches has been limited due to a lack of adequate structure, multi-UAV collisions, and poor navigation in unfamiliar search locations. The proposed controller has a defined structure to optimize search and payload drops and incorporate inter-UAV collision avoidance. The algorithm is scalable, and fault-tolerant but is computationally expensive and hence requires a strong OnBoard Computer. In our case, we chose the NVIDIA Jetson Nano as the OBC as the requirement was to deploy a CPU-intensive algorithm for search and GPU intensive deep learning-based model for object detection. Also, for the communication architecture requirement, we integrated a software-defined Radio with mesh topology to test the controller in a real-world environment.

The limitation of the probability-based method was the need for a centralized controller so the probability map is the same for all the UAVs, but the current controller estimates velocity based on the weight assigned to the cells making the process dynamic and real-time hence eliminating the issue of uniformity. There are two main terms in the velocity vector term, first the inertial term which controls the factor exploration vs exploitation, and the second the derivative of the probability distribution curve. The weights of these were tuned visually such that there was no saturation in the local minima of the probability distribution curve. The exploration and exploitation characteristics are also analyzed over different functions which have given different performances and can be used for different missions. [5]-[11].

II. PRELIMINARIES

2.1 *Problem Formulation*

We envision a swarm of quadcopter UAVs moving across a search region specified by a starting waypoint W_{start}, an ending waypoint W_{end}, and a breadth B, seeking numerous targets strewn throughout the landscape. Because all of the UAVs are at the same altitude, they all travel in the same direction. The position vector $p_i^t = [x_i,y_i]^T$ and the velocity vector $v_i^t = [x_i,y_i]^T$ where N is the number of UAVs, may be employed for the UAV i (i = 1, 2,..., N) at time t.

Using an onboard software stack. Each UAV has an onboard camera that can identify and locate objects inside its field of vision (FOV). Each UAV is also equipped with a single payload that may be dropped at one of many different target locations. The ultimate purpose of UAVs is to scan the surveillance area, find as many targets as possible, drop as many payloads as possible, and reach the end of the search area while avoiding inter-UAV collisions.

A partially connected ad-hoc network [12] is used to mimic the network architecture of all the agents. The model can be represented mathematically by an undirected graph G = (V, E), where V = 1,2,..., N is the set of nodes and $E = \{(i, j): i, j \in V, \|pi - pj\| \leq DC\}$ is the edge set. A single UAV directly connects only to its neighboring agents denoted by $NG = \{j \in V, (i,j) \in E\}$

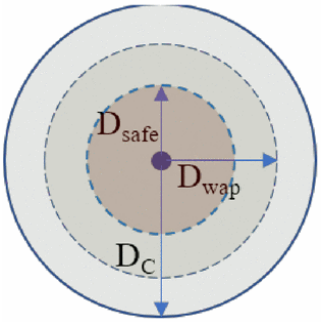

Fig.1: Pictorial representation of D_{safe}, $D_{wap,}$ and D_C

Agents only share the bare minimum of data, such as their location data for inter UAV-collision avoidance, the GPS location and probability of detected targets for cooperative convergence on target clusters, and their payload status for smart and cooperative payload delivery a pictorial representation of the radii are given in fig.1..
The following bounding requirements (in order of priority) confine the algorithm:

- Avoid colliding with other UAVs.
- Identify every target in the search area
- Drop as many cargoes as feasible as fast as possible.

2.2 Probability Map Formulation

Initially the search area is divided into N-cells, where the dimension of each cell is chosen on the basis of the computational capability of the Onboard Computer of the UAV. In our case, the NVIDIA Jetson nano can support up to 30,000-40,000 grid cells, hence according to this limitation the cell dimension is decided.

An object detection algorithm has been used to execute aerial detection of the objects using the images from the EO/IR camera. After deploying and testing the leading models for object detection, YOLOV5 was chosen for its high accuracy while also maintaining decent FPS when deployed on the Jetson Nano. You Only Look Once(YOLO), takes in the images from the EO/IR sensor and predicts the classes and the bounding boxes along with each detection's confidence value. The object is subsequently geotagged using the field of view of the EO/IR sensor.

The probability and area of influence are the two inputs that are estimated for the construction of a probability map. The probability of a target being at the specific cell can be specified by the confidence level that the YOLOv5 model returns and has a range of (0.7,1). The area of influence on the other hand has various factors which need to be accounted for and have been discussed below briefly.

- **Class of the Object:**- There are multiple types of targets that we took in our test case e.g. humans, fuel dumps, type-A vehicles, and type-B vehicles. Certain classes such as Type A vehicles are physically larger than other objects such as Humans and easily detected. Moreover, they should have a much higher area of influence than humans as the presence of vehicles implies a greater concentration of possible targets.

- **Altitude and Size of the object**:- For a given class the relationship between the altitude of the UAV and the size of the object in the image plane (in pixels) is directly proportional and can be calculated using the field of view of the camera system and the resolution of the camera using basic trigonometry.

- **Cluster of targets**:- In real-world environments targets tend to stick together and travel in clusters, hence the probability of a specific grid cell needs to be determined increased if the other targets are also dedicated in its vicinity. This can be represented mathematically just by adding the probability over the cells again.

III. ALGORITHM DESIGN

3.1 Initialization of UAVs

We begin the target-search method in a thorough manner, with UAVs placed in a line at the start of the search region, to optimize the initial exploration. The overlap between each UAV's FOV is limited to a bare minimum and remains consistent (Omin). For repeated runs, if the search region is large enough, it can be divided into many portions.

3.2 Inertial Term

The inertial velocity term, as its name implies, aids in maintaining the swarm unit's original course and prevents abrupt changes in each UAV's trajectory. It's the result of multiplying the current velocity by the inertial function. The algorithm's exploration vs. exploitation features is controlled by the inertial function, which is generally calibrated to give better exploration at first and exploitation as the swarm reaches deeper into the area and has already collected sufficient information about the area.

3.3 Probability Calculation

The concept is based on the fact that targets stick together and travel in clusters, hence a probability distribution is proposed. The mathematical model used to estimate the probability distribution is the Gaussian / Normal distribution curve which is represented below:-

$$f(x) = e^{\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \div \sigma\sqrt{2\pi}$$

where,
σ = Standard Deviation

μ = Mean

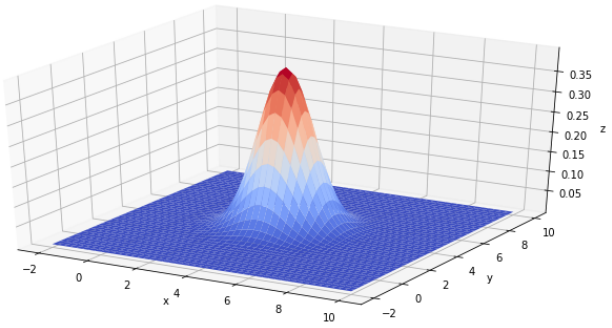

Fig.2: Graphical representation of gaussian curve

σ represents the area of influence which as specified in the previous section has to be selected according to the class of targets identified during the search. The below table represents the area of influence for each class:-

S.No	Type of target	Range of Influence
1	Human	20-40 meters
2	Fuel Dumps	40-60 meters
3	Type-B vehicles	60-100 meters
4	Type-A vehicle	100-140 meters

Table 1: Limits of σ for the type of class

The reason for taking the limits was the second factor which corresponds to increase in probability if the number of pixels occupied by that class of target is more. For YOLOV5 there is a relationship between the number of pixels occupied by the target and the accuracy of the detection given as:

$$m(x) = \frac{1}{1+e^{s(x-m)}}$$

Where:-
f(x) = Accuracy of the detection
x = Size of the object
s = Steepness
m = Shift

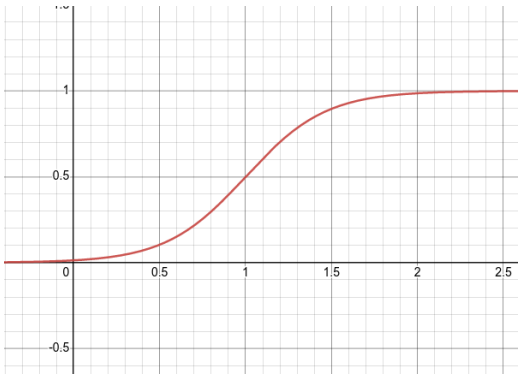

Fig.3: The S-shaped curve of m(x)

The graph for the above function m9x0 is shown in fig.3.
The YOLOV5 model was tested and the value of the hyperparameters came out be:
S = -0.1
M = 16

These parameters represent that the YOLOV5 model can only detect targets with good accuracy if they occupy more than 16 pixels in the input image. Now, this behavior of the graph is imposed between the limits to increase the σ if the number of pixels of the same target class increases, hence making the algorithm more operationally robust.

$$\sigma = m(x) * (Upper\ Limit - Lower\ Limit) + Lower\ Limit$$

This process is repeated for each detected object to generate an independent probability map for each target and the corresponding probability maps are added up to find the final probability map $P(x)$ countering the third point that probability needs to be increased further if more targets keep on detecting in high probable areas. This map can be visualized as a surface plot where the x and y-axis represent the search grid and the z-axis represents the probability value at that specific grid point.The map gives a representation like shown in fig.4.

$$P(x) = \sum_{i=0}^{n} f_i(x)$$

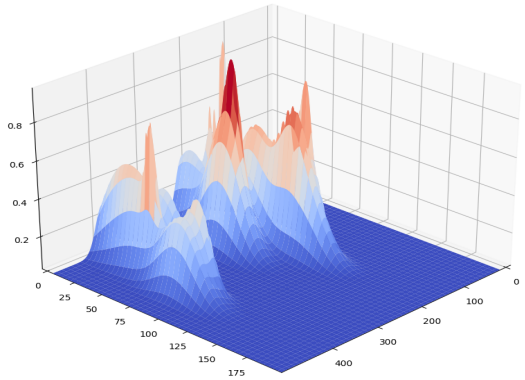

Fig.4: Gaussian-based probability distribution curve of the target area

3.4 Probability-Based Velocity

The probability-based velocity term aims to guide the UAVs towards the high probability areas using the probability map generated earlier. The partial derivative/gradient of the probability map is taken to get the following surface plots representing the partial derivative in the x and y-axis respectively as shown in fig.5.

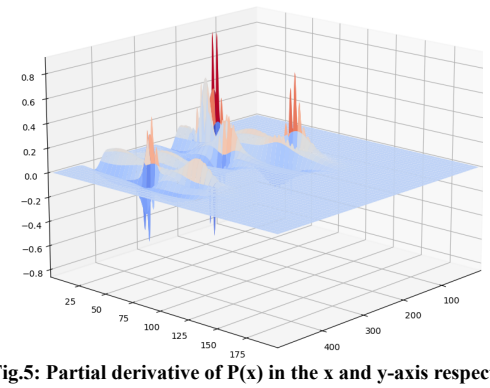

Fig.5: Partial derivative of P(x) in the x and y-axis respectively

$$D(x) = \frac{dP(\sum_{i=0}^{n} f_i(x))}{dx}, D(y) = \frac{dP(\sum_{i=0}^{n} f_i(y))}{dy}$$

The probability-based velocity term uses this gradient to guide it towards the high probability areas using the formula:-

$$V_{(x,y)} = w * D(x,y)$$

where,

w = Tunable constant

3.5 Payload Optimization

The most crucial duty in the quick reaction approach is the payload drop. For job assignment and optimal delivery, we begin by assigning the UAVs to one of three states, which aids in determining which UAVs should continue to search and which should proceed for payload drop:

State	Meaning
State '1'	Payload available
State '0'	Going to deliver the payload
State '-1'	Payload has been dropped

Table 2: Payload drop states

As long as the target is within the maximum displacement range, the locations of the identified targets are continuously relayed between the UAVs, and the payload delivery job is assigned to the UAV nearest to the target.

$$v_{drop}^{t+1} = C_3 * (p_{drop} - p^t)$$

Furthermore, if the UAV detects a target B that is closer to target A while dropping a payload on target A, it moves its drop location to target B, re-qualifying target B for payload drop. While delivering the payload, the UAV continues to seek targets and communicate with other UAVs. It does not return its steps after delivering the goods, instead of continuing the hunt from the drop spot.

3.6 Inter-UAV Collision Avoidance

To maintain the safety of any swarm system, inter-UAV collision avoidance is essential. As a result, we use the consensus equation to generate an extra velocity term that may be vectorially added to each agent's final velocity [13]. If the distance between any nodes gets lower than Dsafe, avoidance is initiated. This method is simple to include in our framework and is represented as follows:

$$D = \left\| p_i - p^t \right\|_2$$

$$v_C^{t+1} = \begin{cases} C_4 * \sum_{i=1}^{N} (p_i - p^t) * \left(1 - \frac{D_{safe}}{D}\right), D \le D_{safe} \\ 0, D > D_{safe} \end{cases}$$

This method is suited for our application since it just considers the agents coming for collision. The D_{safe} parameter is usually set to match the maximum swarm velocity, although it can be manually adjusted if necessary.

3.7 Net Equivalent Velocity

The UAVs communicate with each other to share information about the targets and allow for the decentralized construction of probability maps, such that the final velocity vector can be calculated onboard each UAV independently. Hence the final velocity vector is as follows:

$$V = v_{Inertial} + v_{Probability} + v_{Obstacle\ Avoidance} + v_{Payload\ Drop}$$

3.8 Return Mission

As the mission in MTSO progressed a lot of empty unsearched gaps in the search area were left behind to optimize the time of the mission. Thus when the MTSO mission is over another grid search mission is plotted on the unsearched regions so as to be sure that no other targets are left behind.

3.8.1 K-Means Clustering Algorithm

In the return mission the first task is to calculate the total unsearched area left behind during the search as shown in fig.6. Once this is calculated all the points in the area are segregated in clusters using the K-means clustering algorithm. The K-Means Clustering technique divides the input dataset into multiple clusters based on their distances from the centroids. It's a clustering-based approach in which each cluster has its own centroid, which is used to characterize the clusters. The input given to the Kmeans is the coordinates of the unsearched area which are represented as d-dimensional real vectors as (x1, x2,..., xn).v K means to organize the n groups to k groups where k<=n. The main equation of k means is

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

Which can be reduced to

$$S_i \left| \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2 \right.$$

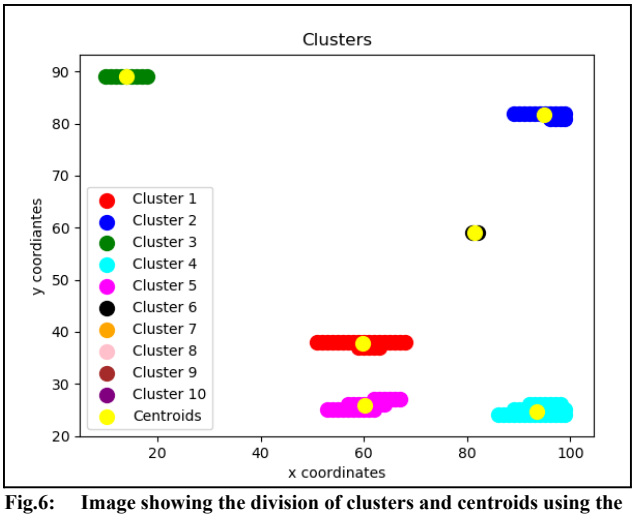

Fig.6: Image showing the division of clusters and centroids using the K-means algorithm

Because the total variance is constant, this is identical to iteratively maximizing the sum of squared deviations across points in distinct clusters.
The number of clusters in which the unsearched area is to be divided is decided by the elbow function. The elbow method runs k-means clustering on the dataset for a range of values for k and then for each value of k computes an average score for all clusters. Using these scores a graph is plotted as shown in fig.7 and the line with the highest changes in slope indicates the optimal number of clusters that need to be made.

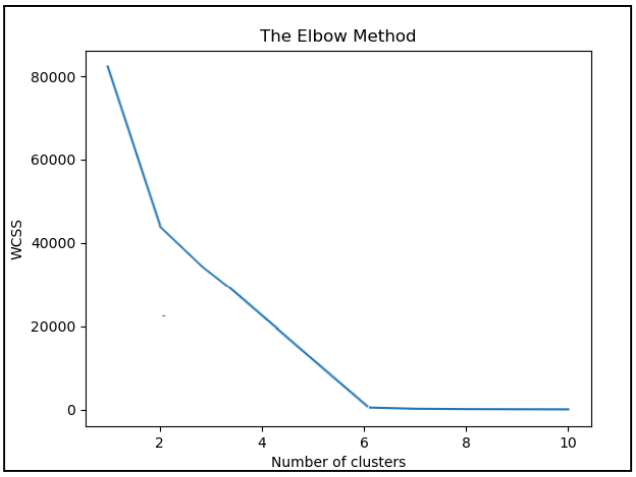

Fig.7: Elbow method showing the max slope change at 6 clusters

3.8.2 Health Function

Once the clusters are divided, UAVs are also segregated based on the priority order given below -:

- Payload dropped or not-: As the number of targets and the number of UAVs are not always equal, so in a case if the number of targets is less than the number of UAVs the UAVs that haven't dropped their payload yet are listed first for the return mission.
- Health function -: The rest of the UAVs with the highest battery percentage and proximity to targets are chosen first for the return mission.

3.8.3 Hungarian Algorithm

The clusters have been constructed, and the UAVs that are qualified for the mission have been separated; now each cluster must be assigned to a UAV that will conduct the search. The nearest possible cluster should be assigned to each UAV for mission robustness and time optimization, and the Hungarian method was utilized for this. The Hungarian Method is a technique based on the idea that if the same value is added or subtracted from every member of a matrix's row or column, the new assignment issue's optimal solution should be the same as the original problem.

3.8.4 Polygon Grid Search

The Hungarian algorithm provided each UAV its cluster. Now each UAV is to be assigned waypoints to search in its area. To assign the waypoints a self-implemented polygon grid search was invented that can work on any polygon. Our algorithm takes in the coordinate points of the boundary of the polygon and forms a perfect rectangle around the polygon completely enclosing the polygon inside it. Now the Grid search method is applied on the rectangle providing initial and final waypoints to the UAV as it continues the mission. To overcome the extra space provided by the rectangle that is not in the polygon a line intersect function is implemented to keep the UAV inside the polygon.

IV. SIMULATION AND TESTING

This section firstly describes the methods of simulations used to test our code. Then, a comparative study of the tuning of different inertial functions according to different FOVs is given. Lastly, the effectiveness of MTSO is compared with other methods in various simulated scenarios.

A. Simulation Environment

To simulate the algorithm in various scenarios and later on compare it to other algorithms, Ardupilot SITL was utilized. To visualize the targets, a self-created mavproxy module is used, which selects the targets at random and visualizes them in the SITL map as shown in fig. 8-11. If the target is detected, the probability map is updated and presented continuously using OpenCV and matplotlib libraries. Because the SITL only accepts GPS data, conversion to UTM coordinates were conducted.

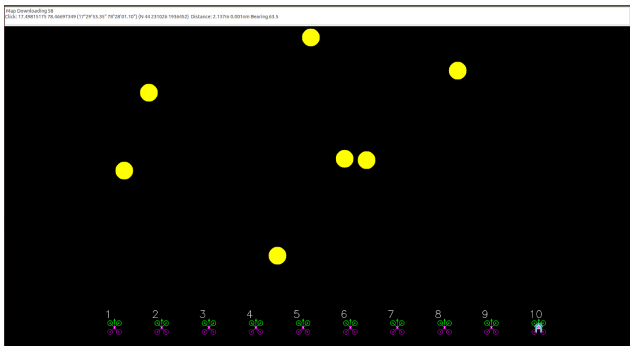

Fig.8: Simulation Environment and Initialization of UAVs

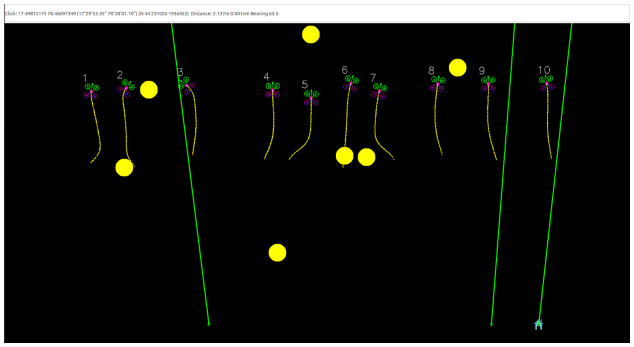

Fig.9: Convergence of UAVs on targets in search area

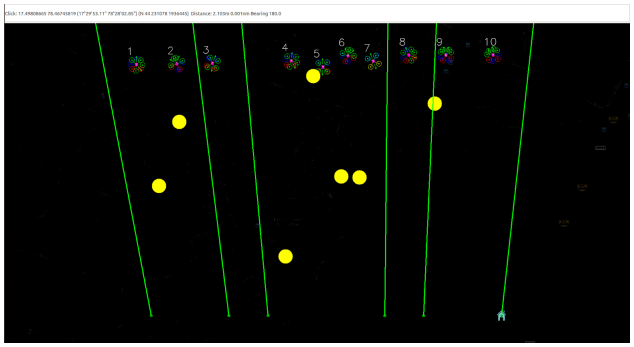

Fig.10: UAVs halting and performing clustering

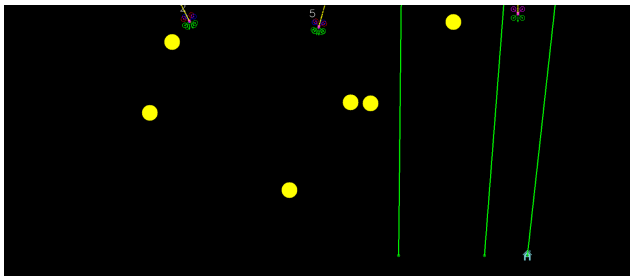

Fig.11: UAVs searching their assigned cluster according to the Hungarian algorithm

B. Path Map of the Swarm

In order to evaluate the mission later, a real-time map of each UAVs travel trajectory is created at the start of the flight. This map also shows the target that was discovered. The map is created using a straightforward openCV function and is shown with a real-time mission updates shown in fig.12.

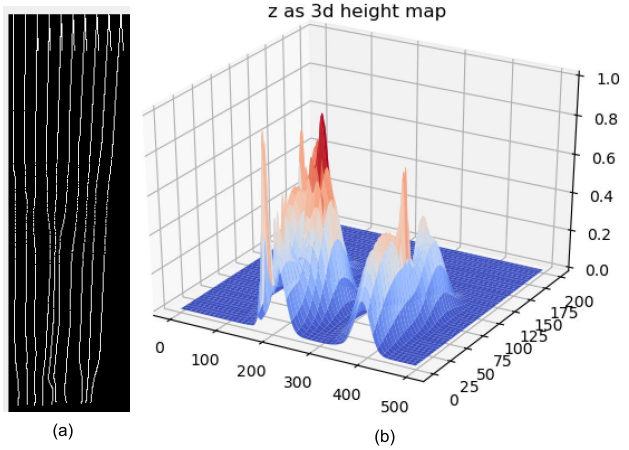

Fig.12: (a)Path map of the swarm and (b) Probability Map for the test case of FOV 140x80

C. Quantitative Comparison with Different FOVs

The algorithm of MTSO with probability map was rigorously tested against different parameters for FOV of the camera to find its performance and discrepancies if any. These tests were carried out while the camera module's FOV was changed. The FOV settings used in the experiment were chosen with the least amount of overlap in mind (O min).

Table III summarizes the outcomes of the tests. The data clearly illustrates that expanding the FOV enhances exploration since the majority of the search region is searched, resulting in an improved overall number of discovered targets but also increasing the total time needed to scan the environment. So it is essential to find an optimum balance of efficiency, time optimization and total area scanned to get the best results for the mission.

Metrices	FOV (in meters)		
	55X32	70 X40	140X80
Target Detected	15	17	20
Payload Dropped	14	17	19
Time For Mission(s)	188.14	182.24	176.89
Total Area Scanned	92.46	96.23	98.91

Table 3: Comparison between different FOVs

D. Quantitative Comparison with other approaches

This part shows the comparison between MTPSO, Exhaustive Search and our approach. Below in table.4 is a summary of the results of different qualities of the search algorithm by changing fov in different scenarios by all three algorithms.

Metrices	Method	FOV		
		55X32	70X40	140X80
Targets Detected	Exhaustive Search	19	20	20
	MTPSO	15	16	19
	New Approach	15	17	20
Payload Dropped	Exhaustive Search	18	18	19
	MTPSO	15	15	17
	New Approach	14	17	19
Time For Mission	Exhaustive Search	402.63	422.75	454.56
	MTPSO	207.86	200.01	193.45
	New Approach	188.14	182.24	176.89
Total Area Scanned	Exhaustive Search	99.99	99.99	99.99
	MTPSO	92.61	96.38	98.32
	New Approach	92.46	96.23	98.91

Table 4: Performance metrics evaluated for different approaches with different FOVs

In each of the scenarios outlined above, MTSO with probability map outperformed exhaustive search and MTPSO. The time used by MTSO was roughly half that of the exhaustive search in terms of time optimization. Because the probability map aids in swarm convergence rather than selecting a Pbest and Gbest, the convergence towards the goal is finer, resulting in a smaller scanned region. The processing power required by MTSO utilizing a probability map is significantly lower than that required by any other probability graph approach, and it is easily implemented in a decentralized version on very light hardware platforms with minimum connectivity.

Hyper parameter	Value	Hyper parameter	Value
D_{wap}	20m	v_{min}	0m/s
D_c	500m	N	20
D_{max}	2.5*FOV	D_{safe}	15m
N_s	5	w	30
α	105°	μ	0
v_{max}	15m/s	m	16

Table 5: Hyperparameters used for simulations

V. CONCLUSION

The new algorithm, unlike the earlier MTPSO algorithm, does not limit the movement of UAVs on the basis of only PBEST and GBEST, which often led to UAVs getting stuck in local minima and being unable to contribute for the rest of the mission. The new approach allows the movement of the UAVs on the basis of all the detected objects using a probability map, hence areas with a much higher concentration of objects take precedence. Moreover, the impact of each object is not constant unlike the previous approaches and various factors like Altitude, Class and Model Confidence which actually affect the ideal behavior in real-world environments are taken into account for assigning the probability which leads to better convergence of the swarm. This also allows for considerably more payloads to be dropped quickly, when compared to exhaustive search and MTPSO increasing the effectiveness of the UAV swarms in real-world disaster relief scenarios. To overcome the limitation of the new algorithm of limited coverage of the search area a return mission is proposed which can be used when the endurance of the UAVs allows, to exhaustively search the left out areas and confirm any objects missed by the initial pass of the algorithm.

The various hyperparameters used in the new algorithm have been tuned manually but instead, they can be tuned using advanced techniques like evolutionary hyperparameter tuning which can further improve the performance of the algorithm.

In future work, the research aims to improve-:

- Refining the revert mission to do an iterative MTSO.
- Further improving MTSO to decrease time and increase efficiency.

REFERENCES

[1] Cabreira, T.M.; Brisolara, L.B.; Ferreira Jr., P.R. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. Drones 2019, 3, 4.

[2] L. Bertuccelli and J. How, “Robust UAV search for environments with imprecise probability maps,” in Proc. IEEE Conf. Decision Control, Dec. 2005, pp. 5680–5685.

[3] Y. Yang, A. Minai, and M. Polycarpou, “Decentralized cooperative search by networked UAVs in an uncertain environment,” in Proc. Amer. Control Conf., Jun.–Jul. 2004

[4] J. Hu, L. Xie, K. Lum and J. Xu, "Multiagent Information Fusion and Cooperative Control in Target Search," in IEEE Transactions on Control Systems Technology, vol. 21, no. 4, pp. 1223-1235, July 2013

[5] J. Kennedy and R. Eberhart. Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks, pages 1942-1948, IEEE Press, Piscataway, NJ, 1995

[6] Y. Shi and R. Eberhart., “A modified particle swarm optimizer”, In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pages 69–73.

[7] IEEE, 2002. R.C. Eberhart and Y. Shi., “Tracking and optimizing dynamic systems with particle swarms”, In Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, volume 1, pages 94–100.

[8] IEEE, 2002. A.Nikabadi, M.Ebadzadeh , “Particle swarm optimization algorithms with adaptive Inertia Weight : A survey of the state of the art and a Novel method”, IEEE Journal of evolutionary computation , 2008

[9] R.F. Malik, T.A. Rahman, S.Z.M. Hashim, and R. Ngah, “New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight”, International Journal of Computer Science and Security (IJCSS), 1(2):35, 2007.

[10] J. Xin, G. Chen, and Y. Hai., “A Particle Swarm Optimizer with Multistage Linearly-Decreasing Inertia Weight”, In Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on, volume 1, pages 505–508. IEEE, 2009.

[11] Y. Feng, G.F. Teng, A.X. Wang, and Y.M. Yao., “Chaotic Inertia Weight in Particle Swarm Optimization”, In Innovative Computing, Information and Control, 2007. ICICIC’07.

[12] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, “Flying AdHoc Networks (FANETs): A survey,” Ad Hoc Networks, vol. 11.

[13] Anuj Agrawal, Aniket Gupta, Joyraj Bhowmick, Anurag Singh, Raghava Nallanthighal, “A Novel Controller of Multi-Agent System Navigation and Obstacle Avoidance”, Procedia Computer Science, Volume 171, 2020, Pages 1221-1230

[14] A. Gupta, A. Virmani, P. Mahajan and R. Nallanthigal, "A Particle Swarm Optimization-Based Cooperation Method for Multiple-Target Search by Swarm UAVs in Unknown Environments," 2021 7th International Conference on Automation, Robotics and Applications (ICARA), 2021, pp. 95-100, doi: 10.1109/ICARA51699.2021.9376529.

[15] N. Aggarwal, Anunay, V. Jain, T. Singh and D. K. Vishwakarma, "DLVS: Time Series Architecture for Image-Based Visual Servoing," 2023 8th International Conference on Control and Robotics Engineering (ICCRE), Niigata, Japan, 2023, pp. 101-107, doi: 10.1109/ICCRE57112.2023.10155585.

[16] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, 4, 1942-1948. DOI: 10.1109/ICNN.1995.488968.

[17] Coello Coello, C. A., Lamont, G. B., & Van Veldhuizen, D. A. (2007). Evolutionary algorithms for solving multi-objective problems (2nd ed.). Springer.

[18] Tan, Y., Zhu, J., & Ding, X. (2013). A hybrid multi-objective particle swarm optimization algorithm. Soft Computing, 17(6), 1013-1031. DOI: 10.1007/s00500-013-0983-3.

[19] Zeng, J., Chen, J., Zhang, Q., & Zhang, B. (2018). Multi-objective particle swarm optimization with decomposition for optimization problems with variable linkages. IEEE Transactions on Evolutionary Computation, 23(2), 217-231. DOI: 10.1109/TEVC.2018.2817386.

[20] López-Ibáñez, M., Paquete, L., & Stützle, T. (2010). Hybridization of evolutionary multiobjective algorithms: A survey. ACM Computing Surveys (CSUR), 42(4), 1-37. DOI: 10.1145/1830761.1830763.

[21] N.Roberto, Z.Qian, L.Yanmin H.Huayao, Y.Meilan, S.Xiaoli, 02/03/2022 "Multi-Objective Particle Swarm Optimization with Multi-Archiving Strategy", 1058-9244, doi:10.1155/2022/7372450.

[22] Raida SELLAMI, Farooq SHER, Rafik NEJI, An improved MOPSO algorithm for optimal sizing & placement of distributed generation: A case study of the Tunisian offshore distribution network (ASHTART), Energy Reports, Volume 8, 2022, Pages 6960-6975, ISSN 2352-4847,https://doi.org/10.1016/j.egyr.2022.05.049.

[23] D.Van, W.Shihua, L.Yanmin, Z.Kangge, L.Nana, W.Yaowei, 2022, "Multiobjective Particle Swarm Optimization Based on Ideal Distance", 2022, DOI:10.1155/2022/3515566

[24] Yang M, Liu Y, Yang J. A Hybrid Multi-Objective Particle Swarm Optimization with Central Control Strategy. Comput Intell Neurosci. 2022 Mar 9;2022:1522096. doi: 10.1155/2022/1522096. PMID: 35310587; PMCID: PMC8926491.