2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

# Heterogeneous UGV-MAV Exploration Using Integer Programming

Ayush Dewan Aravindh Mahendran Nikhil Soni K.Madhava Krishna

*Abstract*— **This paper presents a novel exploration strategy for coordinated exploration between unmanned ground vehicles (UGV) and micro-air vehicles (MAV). The exploration is modeled as an Integer Programming (IP) optimization problem and the allocation of the vehicles(agents) to frontier locations is modeled using binary variables. The formulation is also studied for distributed system, where agents are divided into multiple teams using graph partitioning. Optimization seamlessly integrates several practical constraints that arise in exploration between such heterogeneous agents and provides an elegant solution for assigning task to agents. We have also presented comparison with previous methods based on distance traversed and computational time to signify advantages of presented method. We also show practical realization of such an exploration where an UGV-MAV team efficiently builds a map of an indoor environment.**

## I. Introduction

With the advent of micro-air vehicles (MAV) as a popular robotic testbed [1], [2] numerous problems involving such vehicles have been approached with right earnest. These include systems modeling and control of such agents [3], collision avoidance [4], mapping and state estimation with such agents [5]. Mobile robots or unmanned autonomous ground vehicles (UGV) have been in vogue for several years and there arise several fruitful applications where such heterogeneous systems of UGV-MAV work together. For example a team of UGVs and MAVs can be asked to map a terrain consisting of water bodies inaccessible to UGV while several minute details hazy to MAVs from an aerial view can be mapped by UGVs on ground. In an equivalent indoor situation the MAV can map terrains inaccessible to UGV such as areas above desks and tables while the UGV can ably find and map regions underneath the tables and chairs and detect spaces between them.

Central to such mapping applications is the requirement of a collaborative exploration strategy. This exploration strategy involving heterogeneous agents could bring with it additional constraints not envisaged in typical multi robotic ground applications. For example, a low cost off the shelf MAV may not be able to carry depth sensors such as laser range finders or depth cameras due to payload restrictions. In such scenarios a MAV using a lightweight single camera to map the terrain would require the aid of the UGV for egomotion estimation every once in a while. It may, moreover, need the UGV for guidance in moving between frontiers that are far apart. These constraints emerge since **single camera (Monocular)** Simultaneous Localization and Mapping (SLAM) systems typically do not scale up over large areas. Till date almost all single camera SLAM systems have been manually guided [6], [7], even semi autonomous systems are rare.

K.Madhava Krishna is the head of Robotics Research Lab, IIIT-Hyderabad, India mkrishna@iiit.ac.in

Ayush Dewan is a MS Student at Robotics Research Lab , IIIT-Hyderabad ayush.dewan@research.iiit.ac.in

Aravindh Mahendran is a MS Student at Carnegie Mellon University amahend1@andrew.cmu.edu

Nikhil Soni is a MS Student at Carnegie Mellon University nikhilso@andrew.cmu.edu

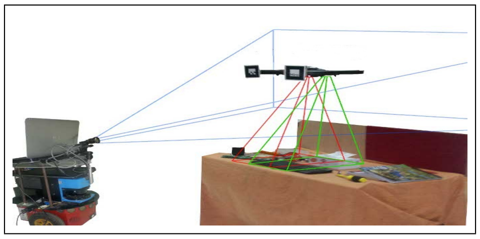
Herein we present a novel optimization formulation for coordinated exploration using heterogeneous agents(Fig. 1). We follow the next best view strategy where a set of favorable goal/frontier locations are estimated using the current map information. The problem of allotting frontier locations to agents is posed as an Integer Programming (IP) problem where allotment of agents to frontier location is specified as integer constraint. Despite the complexity of the IP solution sufficient pragmatic heuristics coupled with sophisticated relaxation techniques ensure that the performance gain due to a fully optimal formulation is maintained without compromising computational time.

The paper is best viewed in two ways. Firstly it shows how an exploration problem can be cast as an Integer Programming problem with one shot assignment of frontiers and the performance gain accrued thereof. All previous works in this area [9], [11] make use of incremental frontier assignment. The performance gain is visible in terms of reduced distances traveled by the robots for completing the exploration without tangible loss in computation times. This is shown in a generic multiagent exploration scenario wherein constraints specific to UGV-MAV heterogeneous situations are not incorporated. Secondly it also paves way for adding several practical constraints in a heterogeneous framework in a seamless fashion. Posing the problem as constrained optimization formulations that do not make explicit use of such an optimization framework hand coding of these constraints becomes rather unwieldy.

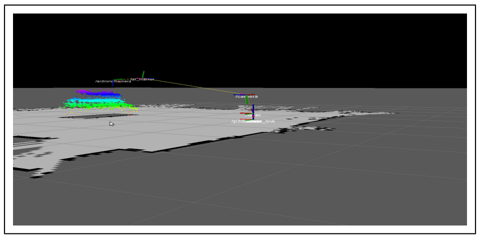
The formulation is also tested for a distributed/ decentralized systems. Distribution of agents becomes necessary in case of communication breakdown. As agents disperse into the environment constant communication between them cannot be guaranteed. We divide agents into smaller teams using graph cut partitioning where each vertex of graph is location of ground vehicle in map.

This paper goes beyond our earlier shorter version [22] by detailing the optimization formulation, posing practical requirements of visibility and MAV guidance as integer constraints, elaborating how the time complexity can be kept to manageable levels, through extensive comparisons and real-time experiments on heterogeneous set of robots.

We also bring to note based on our survey there appears



(a)



(b)

Fig. 1. (a)A pair of heterogeneous agents. (b)Mapped ground(gray) and elevated area(colored points).

no other effort that shows how a team of UGV and MAV can explore and map an environment by collaboration.

## II. Literature Survey

In recent years there has been a spurt in literature dealing with MAV systems in relation to various aspects of them. For example, [13] describes a differential equation formulation to decide the next location for exploration. This was done for a payload constrained UAV. A cooperative target tracking strategy with multiple MAV using the Kalman Bucy formulation was presented in [14] whereas in [15] a Recursive Bayes Filter formulation for incrementally estimating the state of the environment with multiple MAV was presented.

In the context of UGV-MAV collaboration, [16] present a system paper on a UGV guiding a MAV to navigate in an indoor environment making use of a LED cube structure pattern attached to MAV. In [17], the authors propose a cooperative control framework for real time task generation and allocation for a hierarchical MAV/UGV platform. Whereas [18] proposes a decentralized architecture for autonomous teams of MAVs and UGVs engaged in actively searching for and localizing ground features.

Finally from the point of view of multi agent/robot exploration there have been several works addressing various problems ranging from coordination strategies [9], [10], choice of metric [11] and presence of constraints [19]. However all such efforts in multi agent exploration have used incremental agent-frontier allotment as the quintessential strategy. This paper differs from other multi agent exploration approaches through its formulation and its application on heterogeneous team of agents/robots.

## III. Problem and System description

### A. Problem Description

Given a **workspace U** consisting of floor area and obstacles the objective is to find an efficient exploration strategy to explore U through a combination of UGV and MAV. The **floor area** denoted by **F** is the area explored by the UGV while areas above obstacles such as desks, tables are mapped by the MAV. The **MAV-explored area** is represented as **A** and is disjoint from F.

### B. Agent Mapping

The UGVs and MAVs share a global 2D occupancy grid map, updated by sensor readings from each robot.

This map is processed in each iteration to identify points at the boundary of unoccupied and unknown regions and contiguous set of such boundary points are accumulated to constitute frontiers. The remaining frontiers are then sampled to generate a set of (pose, orientation) pairs used for deciding the next best state for the heterogeneous system. MAVs follow UGVs till they detect regions that are not mappable by UGVs but are mappable by MAV's. These are identified using sonar data in simulation by tracing the path of the sonar through the occupancy grid map and its obstacles. The boundaries of such regions constitute what we call as "Passive MAV Frontiers". When a MAV, which is currently following a UGV, is assigned to explore one such frontier it formally changes to "Active" state and starts mapping the regions below it using its down facing 3D sensor. New mapped points in global coordinates are added to the original point cloud, which is again shared globally across MAV's. The point cloud is down sampled using a voxel grid to remove redundant points and processed to extract frontiers. Frontier extraction begins by converting the 3D point cloud into a binary 2D occupancy grid with large cell size. The rightmost point cloud boundary cell is identified and a 2x2 window is made to now trace a contour around the point cloud. The large cell size ensures that the point cloud projects to a continuous blob in the binary voxel grid and this makes it very easy for the 2x2 window to trace the contour (see [21] for details). Since, multiple runs of the MAV over different elevated regions can create multiple blobs, these are split into different point clouds using Euclidean clustering [23] before being projected into separate binary voxel grids for contour extraction. This contour is the required frontier and is sampled to create a set of (pose, orientation) pairs used for deciding the next best pairs for the heterogeneous system. The pose of the MAV and UGV are assumed to be known as it mapped the area beneath. The pose of all the agents are centrally maintained by the ROS framework.

### C. Optimization Formulation

As already discussed in section I, the objective is to allot frontier locations from the set of current frontiers to the set of UGV-MAV agents.

We formulate this problem in following manner,

$$\max \quad \textstyle\sum_{\texttt{Agent } i} \quad \sum_{\texttt{Frontier } j} \left( h_{ij} \times \frac{\texttt{V}(j)}{\texttt{D}\,(i,j)} \right) \qquad (1)$$

$V_j$ symbolizes visibility at $jth$ frontier and $D_{ij}$ is the amount of distance agent $i$ has to travel to reach frontier $j$ and $h_{ij}$=1 if agent $i$ is allotted frontier $j$, otherwise 0. The optimization attempts to maximize net visibility gain per unit distance($V/d$) traveled. This metric was demonstrated to give superior results [11] than metrics which used various other combinations of utility (visibility gain) and cost (distance traveled). Since, $h_{ij}$ is either 0 or 1, this problem becomes a standard 0-1 Integer Linear-Programming problem.

Determining an $exact$ solution for integer linear programming problem is in general NP-Hard. By $exact$ we mean that all the unknowns (to be solved) are well constrained to be integers. To get a better insight on solving this NP-Hard problem we first introduce $FacilityLocationProblem(FLP)$[12]problem, a well know subject of Operational Research.

Suppose there is a set $J = 1...n$ of $n$ potential sites for establishing facilities to serve $m$ customers. The objective of $FLP$ is to determine which subset of facilities should be opened in order to minimize cost of transportation and cost of opening a facility, which can be presented as

$$\min_{h,l} \sum_{\mathtt{i}\in\mathtt{C}} \sum_{\mathtt{k}\in\mathtt{J}} h_{ik} d_{ik} + \alpha \sum_{\mathtt{k}\in\mathtt{J}} l_k \tag{2}$$

such that

$$\forall ik, h_{ik} \leq l_k, \tag{3}$$

$$\forall i, \sum_{k\in J} h_{i,k} = 1, \tag{4}$$

$$\forall ik, h_{ik} \in \{0,1\}, l_k \in \{0,1\} \tag{5}$$

Here $C = 1...m$ is set of clients/customers, $l_k$ =1 if facility at $k$ is opened and $l_k$ =0 otherwise, $h_{ik}$ =1 if customer $i$ is served by facility at $k$, $d_{ik}$ is transportation cost and $\alpha$ is the cost associated with opening a facility. Comparing $FLP$ problem with problem posed in 1, we find they are similar. Both are challenging (NP Hard) linear integer programming problems. There has been many strategies proposed for tackling NP Hard problems and among them Linear Program Relaxation(LPR) is one of the most promising approaches. The key idea behind LPR is to relax 0-1 integer constraint into linear inequalities over continuous variables, thereby reducing to a form that can be solved as a set of linear programming solvers. LPR has been applied successfully in many cases, including $FLP$ problem.

We have used glpk solver [27] to obtain the solution for the IP formulation. The solver uses branch and cut method which is a combination of branch and bound and cutting plane methods to find the solution. Branch and cut methods efficiently works on LP relaxation of IP problem. It prune branches to remove non integer optimal solution from feasible space thereby preventing an exhaustive search for solving the non polynomial IP formulation.

To further improve the performance, we use a clustering heuristic to decrease the number of frontiers to be considered for allotment. This technique reduces the size of feasible solution space, hence improving the computational time required for finding solution for IP formulation.

*1) Objective:* Since our formulation involves heterogeneous agents, equation 1 is modified to following form,

$$\begin{aligned} \max \quad & \textstyle\sum_{\mathtt{UGV}\; i} \sum_{j\epsilon \mathtt{U_f}} \left( x_{ij} \times \frac{\mathtt{V}(j)}{\mathtt{D}\,(i,j)} \right) + \\ & \textstyle\sum_{\mathtt{MAV_a}\; i} \sum_{j\epsilon \mathtt{M_f}} \left( y_{ij} \times \frac{\mathtt{V}(j)}{\mathtt{D}(i,j)} \right) + \\ & \textstyle\sum_{\mathtt{MAV_p}\; i} \sum_{j\epsilon \mathtt{M_f}} \left( z_{ij} \times \frac{\mathtt{V}\,(j)}{\mathtt{D}(i,j)} \right) + \\ & \textstyle\sum_{\mathtt{UGV}\; i} \sum_{j\epsilon \mathtt{MAV_a}} \left( \frac{ha_{ij}}{\mathtt{D(i,j)}} \right) \end{aligned} \tag{6}$$

where,$U_f$ and $M_f$ are set of UGV and MAV frontiers respectively. First three terms capture the allotment of frontiers for UGV, ActiveMAV(MAV$_a$) and PassiveMAV(MAV$_p$) respectively, whereas the last term signifies the transition where an MAV goes from an active map builder (Active MAV) to a passive follower (Passive MAV), after completion of mapping or due to lack of new reachable frontiers. The integer variables $x_{ij}$, $y_{ij}$, $z_{ij}$ and $ha_{ij}$ are discussed in the section below (III-C.2).

*2) Variables:* The state of the system is captured using 5 sets of binary integer variables. The first three, $x_{ij}$, $y_{ij}$ and $z_{ij}$ denote that the $i^{th}$ robot is allotted the $j^{th}$ frontier. More elaborately, the first set $\{x_{ij}\}$ identifies the UGV:UGV frontier pairing/allocation. The second, $\{y_{ij}\}$ serves the same purpose but now for Active MAV:MAV frontier allocation. $\{z_{ij}\}$ records the pairing between Passive MAV:MAV frontiers. Apart from frontier allocation, explicitly labeling active to passive and passive to active transitions greatly helps in imposing constraints specific to the nature of these transitions. We define:-

$$ha_{ij} = \begin{cases} 1 \ \mathtt{i^{th}\,UGV\,helps\,j^{th}\,MAV_a\,become\,passive} \\ 0 \ \mathtt{otherwise} \end{cases}$$

$$hp_{ij} = \begin{cases} 1 \ \mathtt{i^{th}\,UGV\,helps\,j^{th}\,MAV_p\,become\,active} \\ 0 \ \mathtt{otherwise} \end{cases}$$

*3) Constraints:* While the complete formulation can be seen here [24], we discuss the most important constraints.

*Visibility Constraint*:- In our formulation we have introduced a visibility constraint, which requires an UGV to observe a MAV at its frontier location. Due to scale drifts[8] that occur in monocular SLAM, ego estimation of MAV by an UGV becomes inevitable. Since the UGV is more firm about its own ego estimate its observations of the MAV enhance the mapping performance of the air vehicle.

This constraint is difficult to express in closed form and is non-linear. Hence, we compute the pairs of viewer-object positions which sa qtisfy the constraint and provide this information as input to the Optimization. The following two types of viewer-object position pairs are of interest:-

*FF* - $(i,j)\,\epsilon\, FF$ if MAV at frontier j is visible to UGV at frontier i.

*CF* - $(i,j)\,\epsilon\, CF$ if MAV at frontier j is visible to $ith$ UGV at its current position.

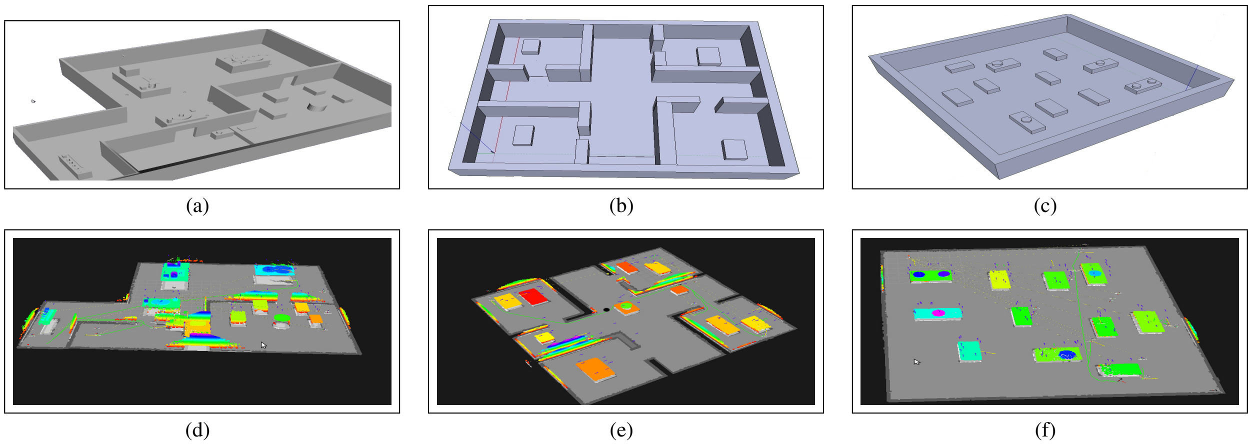


Fig. 2. (a)-(c)Simulation models for exploration.(d)-(f) Final exploration results for respective models

This way, a complicated non linear visibility constraint can be incorporated into the Integer program. This is also why frontier positions are modeled using binary variables corresponding to real valued positions instead of formulating a mixed integer program (MIP) where variables represent x,y coordinates.

One of three Visibility constraints are explained below:-

$$\forall \quad j\epsilon\mathtt{M_f}$$
$$\sum_{\mathtt{MAV_a}\ i} y_{ij} - \sum_{(l,j)\epsilon FF}\left\{\sum_{k\epsilon\mathtt{UGV}} x_{kl}\right\} + \sum_{(k,j)\epsilon CF}\left(\left(\sum_{l\epsilon\mathtt{U_f}} x_{kl}\right)-1\right) \le 0 \tag{7}$$

Equation 7 can be understood by noting the significance of each term in the equation. The first summand is $1$ if a MAV is allotted the $j^{th}$ frontier, otherwise $0$. The second summand is equal to the number of UGV frontiers that can see the $j^{th}$ MAV frontier and have a UGV assigned to reach it by the next time step. The third summand is equal to negative of the number of UGVs who in their current position can see a MAV after it reaches the $j^{th}$ frontier. We also ensure that this UGV is not allotted frontier in the present iteration by the inner summation used in this term. Put together, if a MAV is assigned the $j^{th}$ frontier, we should have at least one UGV that can see it either after reaching a UGV frontier or by staying at its current position. [1]

*Minimum Distance Between Frontiers*:-The objective function does not subtract the region of overlap corresponding to the selection of two neighboring frontiers. We compensate for this by forcing allotted frontiers to be a minimum $C$ distance apart. $x_{ij}d_{jl}x_{kl} \geq C$ suggests that if the $i^{th}$ UGV is allotted the $j^{th}$ frontier and the $k^{th}$ UGV is allotted the $l^{th}$ frontier then the precomputed distance between these frontiers $d_{jl}$ should exceed a threshold C. This formulation of the minimum distance constraint introduces non linearity.

[1]This also explains why a UGV might in some iterations choose to not move to any frontier location

To ensure that our formulation remains linear we instead impose that

$$d_{jl} + \left(2 - \left(\left(\sum_i x_{ij}\right) + \sum_k x_{kl}\right)\right) \times C \geq C$$

. This achieves the same result because, if any robot is allotted the $j^{th}$ frontier then $(\sum_i x_{ij}) = 1$. Similar is the case with $x_{kl}$. Three cases arise.

1) If none of the frontiers are allotted then the equation becomes $d_{jl} + 2C \geq C$ and we are done. The distance is not constrained as expected.

2) If only $1$ of the frontiers is allotted then the equation becomes $d_{jk} + 1C \geq C$ and we are done. Again the distance is not constrained as expected.

3) If both are allotted then the equation becomes $d_{jk} \geq C$ and the constraint is applicable as required.

*At Most One Frontier Per Agent*:-The minimum distance constraint discussed above does not ensure that two robots are not allotted the exact same frontier. This is incorporated with a simple constraint saying

$$\forall\, j\epsilon\mathtt{U_f} \sum_{\mathtt{UGV}\ i} x_{ij} \leq 1 \tag{8}$$

## IV. Distributed System

In real world situations uninterrupted communication between agents cannot be guaranteed. As agents disperse into environment, communication link between them are stretched and a breakdown becomes inevitable. This breakdown can be avoided by ensuring constant communication between agents and make them move in a pack, like in [20] a reactive virtual spring damper system was used to manipulate motion of agents to ensure constant communication.

To tackle the communication constraint problem, we use a distributed system approach. As agents disperse into environment they are divided/distributed into smaller teams. Graph $G = (V, E)$ is used for capturing the spread of UGVs.

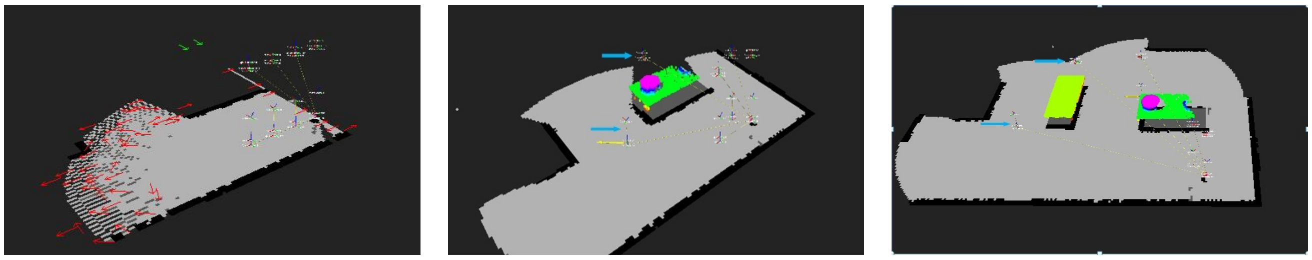

(a) (b) (c)

Fig. 3. (a)Grey area shows ground map and colored points represent point cloud. Red arrows represents UGV frontiers, green arrow represents Passive MAV frontiers.(b) and (c)Blue arrows point to UGV and MAV pair which are maintaining visibility constraint.

Each vertex $v_i \in V$ represents position of a UGV and it is connected to every other vertex through edge $e_{ij}$. Edge weight $w_{ij}$ between vertex $v_i$ and $v_j$ is calculated using Equation 9.

$$w_{ij} = e^{-d_{ij}} \tag{9}$$

$$d_{ij} = \begin{cases} ||v_i - v_j|| & \text{if } vib_{ij} \text{ is } 1 \\ p_{ij} & \text{if } vib_{ij} \text{ is } 0 \end{cases} \tag{10}$$

When a pair of UGVs are visible to each other, $vib_{ij}$ is set to 1 and $d_{ij}$ is the Euclidean distance between them. If visibility is occluded by an obstacle then $d_{ij}$ is set to $p_{ij}$ where $p_{ij}$ is the total distance that UGV at $v_i$ has to travel to reach location of UGV at $v_j$.

In every iteration total distance between UGVs (Equation 11) is computed. When $D$ exceeds a predefined threshold, the graph is partitioned using $k$ Way partitioning algorithm [26]. The objective of graph partitioning problem is to compute a k-way partitioning such that sum of the weight of the edges that straddle different partitions is minimized so that robots that are far away from each other are part of a different team. Using $d_{ij}$ we find UGVs that are farthest away from each other beyond a threshold and arrive at an initial guess on the number of partitions. Since at the end of every iteration each UGV is uniquely associated with a MAV, explicit partitioning of MAV is not required.

$$D = \sum_{ij} d_{ij} \tag{11}$$

## V. Mapping with Real Agents

For mapping a Pioneer P3DX, iRobot Turtlebot and Parrot ArDrone(MAV) is used. P3DX is equipped with a SICK laser, Turtlebot is equipped with a Kinect while MAV is equipped with a downward facing camera. For state estimation of MAV, attached AR-marker [28] is tracked by kinect.

A global 2D occupancy gound map is created using sensor attached to UGVs. A SLAM algorithm runs on UGVs for effective and accurate localization and mapping. The data from the MAV camera is input to the Parallel Tracking And Mapping (PTAM) algorithm [7] giving a sparse map of the elevated regions of the map. Since ground robot is accurately localized, tracking the MAV by Kinect results in the VSLAM based maps being accurate and up to scale. The resulting map consists of a dense occupancy grid of the ground along with a sparse aerial map of the elevated region. The pose of the MAV is given by Equation 12.

$$T^{W}_{Mcam} = T^{W}_{Ubase} \times T^{Ubase}_{Ucam} \times T^{Ucam}_{Marker} \times T^{Marker}_{Mcam} \tag{12}$$

where, $Mcam$ and $Ucam$ are camera frames w.r.t MAV and UGV respectively and $Ubase$ represents base frame of UGV. $T^{W}_{Ubase}$ is the localization of the robot provided by the GMapping[25] algorithm. $T^{Ucam}_{Marker}$ comes from [28], while $T^{Ubase}_{Ucam}$ and $T^{Marker}_{Mcam}$ are obtained from system calibration.

The PTAM algorithm runs in real time and has been effective over small workspaces. We break down our aerial region into disjoint parts running the PTAM algorithm on them individually. The PTAM algorithm outputs a set of points and the MAV camera trajectory for certain keypoints. These results are in a variable scale which has to be solved for to allow accurate map fusion. We find scale for points of the camera trajectory as well as the map using:

$$P^{w}_{i} = T^{w}_{Mcam} \times T^{Mcam}_{ptam} \times P^{ptam}_{i} \tag{13}$$

and

$$C^{w}_{i} = T^{w}_{Mcam} \times T^{Mcam}_{ptam} \times C^{ptam}_{i} \tag{14}$$

where, $P$ and $C$ represent a map point and camera trajectory point respectively. The transformation matrix $T^{world}_{MAVcam}$ has the scale factor $s$ embedded in it. We solve for $s$ using successive camera poses. The camera pose at instant k can therefore be written as

$$C^{w}_{k} = sR^{w}_{Mcam} \times C^{Mcam}_{k} + t^{w}_{Mcam} \tag{15}$$

Subtracting successive poses and and solving for $s$ we get,

$$[C^{w}_{k+1} - C^{w}_{k}]^T \times [C^{w}_{k+1} - C^{w}_{k}] = s^2 \times$$
$$[C^{Mcam}_{k+1} - C^{Mcam}_{k}]^T \times [C^{Mcam}_{k+1} - C^{Mcam}_{k}]$$

Thus the scale is a ratio of distances between two camera locations in the world frame to two camera locations in the camera frame.

Though the MAV was localized in the global frame, the error in pose estimates leads to error in point cloud position, orientation and scale. These errors are particularly large due to the instability of the quad-copter during PTAM's initialization. In particular, achieving the pure translation required for good initialization was hard on table tops. Hence, a map fusion post processing phase was used to align PTAM's point clouds with the occupancy grid map.

For map fusion we find the mean of the PTAM point cloud and find the unknown region of the occupancy grid nearest to it. Points are uniformly sampled from this region and the aerial PTAM point cloud are then projected onto the occupancy grid plane.For aligning the two point clouds we align their eigen vectors. We do so because the point clouds to be fused have complete overlap and are sampled from a close boundary. The process is non-iterative, making it fast and stable.

The scale is obtained from the ratio of corresponding eigenvectors. The rotation is obtained from the eigenvectors and translation from the vector joining the means of both the point clouds.

More formally, let $C_A$ and $C_B$ be the covariance matrices of the point clouds A and B respectively. $v_{a1}$, $v_{a2}$ and $\lambda_{a1}$, $\lambda_{a2}$ be the normalized eigenvectors and eigenvalues corresponding to $C_A$ respectively. We assume $\lambda_{a1} \geq \lambda_{a2}$. Similarly, for B. Then,

$$\begin{aligned} \theta &= \arccos(v_{a1}^T v_{b1}) \\ s^* &= \frac{\lambda_{b1}}{\lambda_{a1}} \\ t^* &= \bar{b} - \bar{a} \end{aligned}$$

Here, $\bar{b}$ and $\bar{b}$ are the means of point clouds A and B respectively.

## VI. Simulation and Comparative Results

### A. Simulation

The simulation was carried out on a Intel(R)Core(TM)i7-2600K CPU @ 3.40Ghz processor running Ubuntu version 12.04 with 8Gb RAM. The ROS framework was used to model and simulate the ground and aerial robots. The UGV was equipped with a Laser Range Finder (LRF) similar to SICK LRF while the MAV was simulated with a horizontal sonar and downward pointing 3D sensor. Various results are shown for three different maps(Fig. 2) amongst those where we have tested our formulation.

Fig. 3(a) shows a snapshot of simulation where the map constructed by the UGV is shown in grey. Passive frontiers are marked by green arrows and UGV frontier locations are marked by red arrows. Active frontier locations are not shown since as all the MAV's are in passive state at current instant. In next iteration one MAV have becomes active and blue arrows show a pair of UGV and MAV which are maintaining visibility constraint. Fig. 3(b) & 3(c) shows the situation for next couple of iterations, after all agents have reached their allotted positions.

In simulations however the MAV pose is known to a good degree of accuracy, however in a real implementation these observations are of prime importance.

### B. Comparative Results

We present here multiple sets of comparisons of the IP formulation vis-a-vis earlier ones. The first set of comparisons are shown for the heterogeneous UGV-MAV scenario where constraints specific to such scenarios discussed before are incorporated. The comparisons are shown from the point of view of the net distance traveled by agents as well as the computational time for every iteration of optimization. The second set of comparisons depict performance gain when constraints are relaxed thereby establishing that IP formulation being apt for any generic exploration situation.

Graphs of Fig. (4) shows comparisons between IP formulation and incremental frontier allocation method [9], [11]. The incremental method first finds the best agent-frontier allotment based on an objective function. Once the allotment is made the process of finding the best agent frontier pair continues till all agents have been allotted the frontiers. We use the same objective function of Equation 6 for all the methods except that in the incremental formulation the integer terms denoting the allotment of an agent to frontier are not part of the objective function. Thus, the allotment is computed extrinsically as the one that evaluates to the best value of the function. Evident from Fig. 4(a)-4(c), IP formulation outperforms incremental frontier allocation method in terms of distance traveled. The average distance traveled by an agent reduced by 10-15% in case of IP formulation. Distance values are averaged over several runs for a given number of agents in the map. Each run corresponds to different starting pose of agents and the average is computed over ten such runs. The process is repeated for varying number of agents and plotted.

Fig. 4(d)-4(f) shows computational time for IP formulation and incremental frontier location remains comparable. Computational time is time taken for one iteration of exploration and it includes the time taken by the path planning module for computing paths to frontiers from agent locations, time taken by the optimization/incremental formulation in respective methods and time taken for graph partitioning for distributed method. Amongst these the computational cost/time of finding paths between all possible agent frontier pairs is found to be maximum and dominates over all other modules. Beside, the advantages of clustering that have been mentioned in section III-C also helps in reducing computational time required for finding path between agent frontier pair. Fig.4(g) shows how the computational time falls as the number of frontier clusters reduces for a given number of agents.

Fig. 4(h) shows comparison of the IP formulation when the constraints were relaxed, compared with result when constraints were kept intact. Distance traveled to complete the exploration is notably lesser. This is along expected lines for in the absence of constraints there is more freedom to choose the best allotment that optimizes the objective function. Further performance gain in terms of reduced distances with the corresponding incremental formulation sans constraints is shown in fig. 4(i). This vindicates the IP

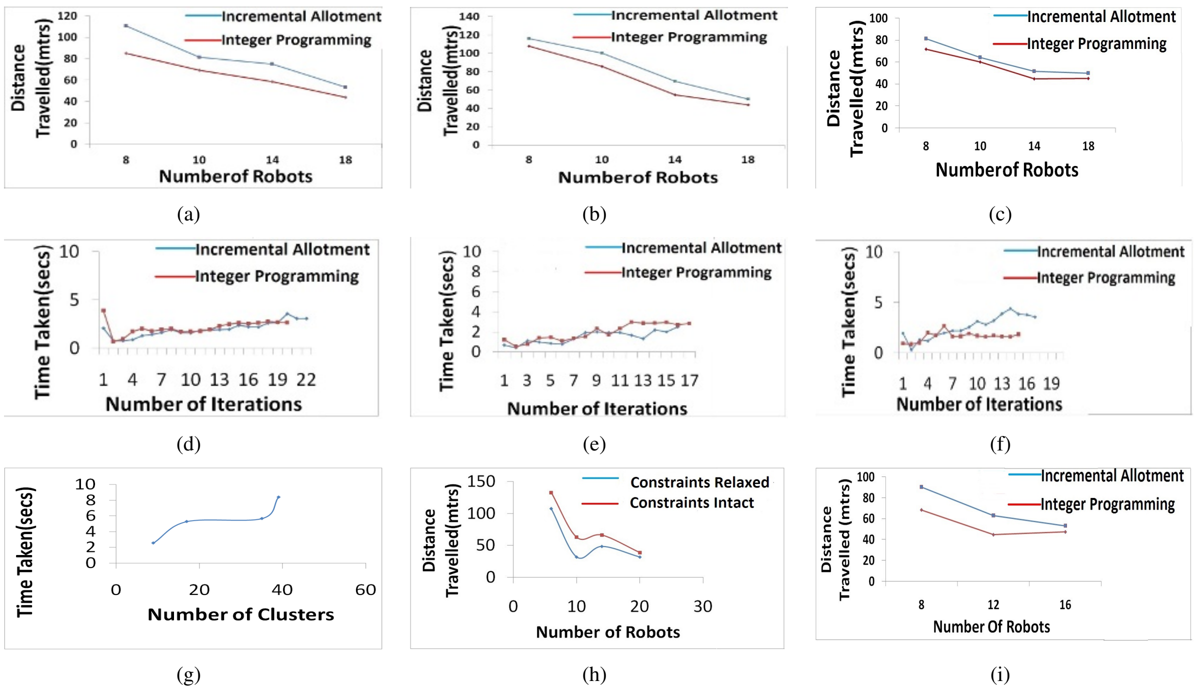


Fig. 4. Comparison between incremental frontier allotment(blue) and IP(red) on the basis of (a)-(c)Distance (d)-(e)Computation time(secs) across different iterations for map 1, map2 and map3(L-R).(g) Comparison of time taken as number of clusters are changed (h)Comparison when constraints are intact (red) and relaxed(blue) for IP Method.(i)Comparison between IP(red) and incremental allotment(blue) when constraints are relaxed.

formulation as an alternative that advances the state of the art and can be applied for a generic exploration situation.

Further results when the agents get partitioned into multiple teams and the optimization is run distributively on each team is shown in fig. (5). Unlike the original formulation due to breakdown of communication or other such contingent situations the teams are unable to share information related to the map structure amongst them. This results in an area being explored multiple times and can be seen in increased distances when compared with original formulation in fig. 5(a). Fig. 5(b) shows comparison on the basis of computational time. The motivation behind this effort of exploring in multiple teams is more than anything else, to achieve the goal of completing the exploration when communication and other contingent hardware failures occur.

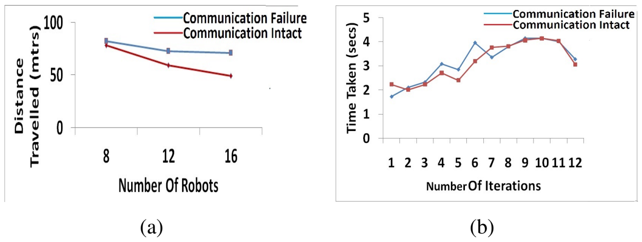


Fig. 5. Blue depicts the performance when multiple teams are formed versus original formulation (red). Comparison on the basis of (a)Distance (b) Time.

## VII. Experimental Results

Results are presented for experimental setup discussed in section V. Figures 6(a) & 6(d) show starting configuration, grey area in figure represents mapped ground area, red markers represent ground frontiers. Yellow arrows mark goal position for agents. UGV1 starts exploration whereas UGV2 stays at current position to maintain visibility with MAV at its goal position. Fig. 6(h) shows tracking of ArMarker using Kinect mounted on UGV2. Figures 6(b) & 6(e) show updated ground map, point cloud and next set of goal positions. As mentioned in section V pointcloud is accurate up to scale. Figures 6(c) & 6(f) show the state of system at next iteration. Fig. 6(g) shows image of scene from MAV's camera and fig 6(j) shows final mapping for environment showed in fig. 6(i). Movement of MAV was autonomous but at times was teleoperated to ensure stability in flight.

## VIII. Conclusions

The paper presented an Integer Programming based multi robotic exploration framework as an alternative that advanced the state of art when compared with previous incremental formulations. The formulation provided performance gain both in the presence of constraints that arise in a heterogeneous team of UGV-MAV and also in the absence of such constraints. LP relaxation techniques coupled with practical heuristics such as clustering contain computational times to levels comparable with previous formulations. Experimental results using couple of ground robots and Parrot drone based on this formulation were presented. Future scope of the work include dense aerial monocular mapping for generation of accurate terrain profiles and aerial vehicle stabilization for sustained exploration.

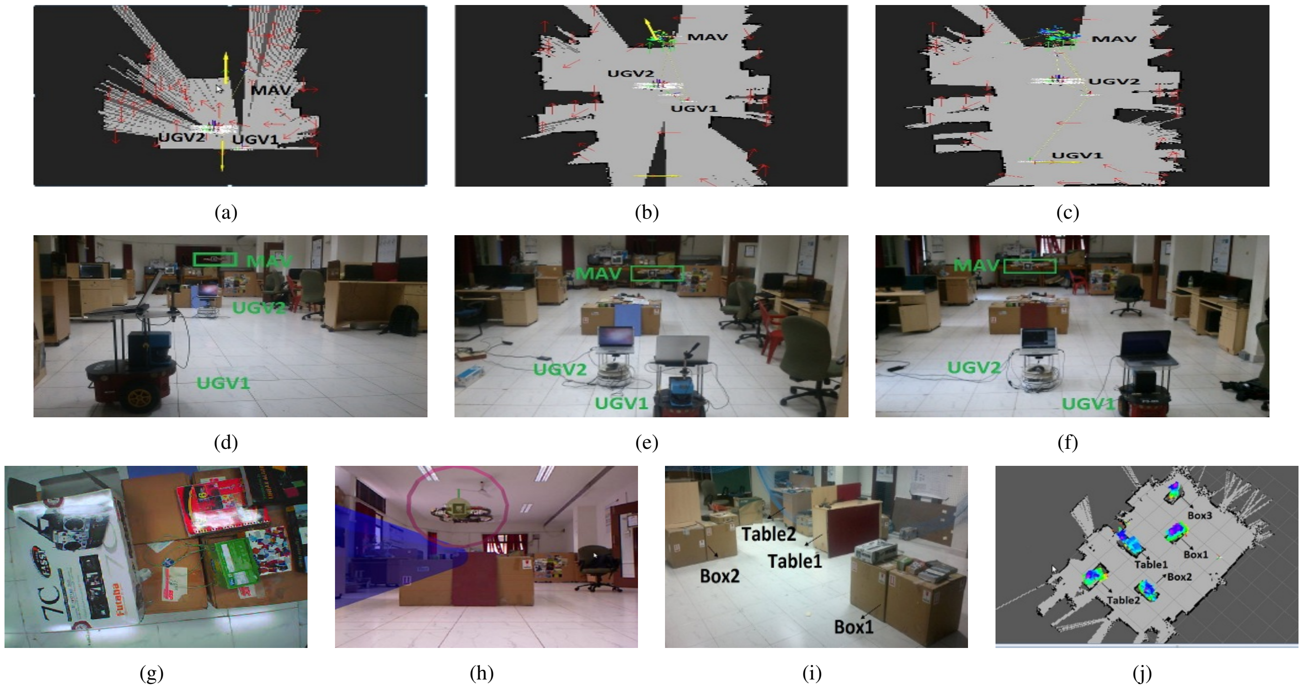


Fig. 6. Experimental setup for consecutive iterations (a)-(c)Yellow arrows represent goal position for agents, red arrows shows ground frontiers, mapped ground area is shown in gray and colored points represent pointcloud. (d)-(f) State of real system at multiple instants. (g) Snapshot of scene from MAV's camera (h) Tacking of ArMarker using Kinect (j)Final mapping result showing ground map(grey) and pointcloud over obstacles for environment(i)

## IX. Acknowledgment

This work was supported by DeITY, NPPE, Phase 2 Project.

## References

[1] T. Krajnik, V. Vonasek, D. Fiser, and J. Faigl, "AR-Drone as a Platform for Robotic Research and Education". In Research and Education in Robotics: EUROBOT 2011, Heidelberg, 2011. Springer

[2] www.astec.de

[3] S. Bouabdallah, "Design and Control of Quadrotors with Application to Autonomous Flying", Theses 3727, EPFL, 2007

[4] Stefan Hrabar, "Reactive Obstacle Avoidance for Rotorcraft UAVs", IEEE/RSJ Intl Conf. on Robots and Systems, pp 4967-4974, 2011

[5] S. Shen and N. Michael and V. Kumar, "3D Estimation and Control for Autonomous Flight with Constrained Computation", International Conference of Robotics and Automation, 2011, Shanghai, China

[6] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Realtime single camera SLAM, PAMI, vol. 29, no. 6, pp. 10521067, 2007

[7] George Klein and David W. Murray, "Parallel Tracking and Mapping", International Symposium on Mixed and Augmented Reality, 2007

[8] H. Strasdat, J. Montiel, and A. Davison, "Scale Drift-Aware Large Scale Monocular SLAM", in RSS, 2010.

[9] W. Burgard, M. Moors, C. Stachniss, and F. Schneider,"Coordinated Multi Robot Exploration", IEEE Transactions on Robotics, 21(3), 2005.

[10] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun and Hakan Younes. "Coordination for Multi-Robot Exploration and Mapping", In Proc. of AAAI, 2000.

[11] Rahul Sawhney, K Madhava Krishna and K Srinathan, "On Fast Exploration in 2D and 3D Terrains with Multiple robots", AAMAS 2009 .

[12] Pirkul, H., Jayaraman, V. (1998). A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution. Computers and Operations Research, 25(10), 869-878.

[13] Shaojie Shen, Nathan Michael, and Vijay Kumar, "Autonomous Indoor 3D Exploration with a Micro-Aerial Vehicle", ICRA 2012, St. Paul, Minesotta

[14] Fabio Morbidi and Gian Luca Mariottini, "On Active Target Tracking and Cooperative Localization for Multiple Aerial Vehicles", IROS 2011, San Francisco

[15] B J. Julian, M Angermannz, M Schwagerx, and Daniela Rus, "A Scalable Information Theoretic Approach to Distributed Robot Coordination", IROS 2011

[16] P Rudol, M Wzorek, G Conte and P Doherty, "Micro Unmanned Aerial Vehicle Visual Servoing for Cooperative Indoor Exploration", IEEE Aerospace Conference, 2008

[17] C. Phan and H. Liu, "A cooperative uav/ugv platform for wildfire detection and fighting". In System Simulation and Scientific Computing, 2008.

[18] B Grocholsky, S Bayraktar, Vijay Kumar and George Pappas, "UAV and UGV Collaboration for Active Ground Feature Search and Localization", AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit, 2004

[19] Piyoosh Mukhija and K Madhava Krishna, "A Two Phase Recursive Tree Propagation based Multi-Robotic Exploration Framework with Fixed Base Station Constraint", IROS 2010

[20] Mosteo, A.R.; Montano, L.; Lagoudakis, M.G., "Multi-robot routing under limited communication range," Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on , vol., no., pp.1531,1536, 19-23 May 2008 doi: 10.1109/ROBOT.2008.4543419

[21] R B Rusu and S Cousins, "3D is Here: Point Cloud Library", ICRA 2011, Shanghai

[22] A Dewan,A Mahenderan,N Soni and K Madhav Krishna "Optimization Based Coordinated UGV-MAV Exploration for 2D Augmented Mapping", AAMAS 2013, Poster

[23] Radu Bogdan Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments", Computer Science department, Technische Universitaet Muenchen, Germany October-2009

[24] Report no: IIIT/TR/2012/63,`http://researchweb.iiit.ac.in/~ayush.dewan/report.pdf`

[25] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters", IEEE Transactions on Robotics, 23:34-46, 2007.

[26] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

[27] http://www.gnu.org/software/glpk/

[28] http://www.hitl.washington.edu/artoolkit

[29] http://www.openscenegraph.org/projects/osg