Collaborative Air-Ground Target Searching in Complex Environments

Changsheng Shen*, Yuanzhao Zhang*, Zimo Li, Fei Gao and Shaojie Shen

***Abstract*— Collaboration between heterogeneous robots can greatly improve the overall robot system by obtaining capabilities that each single robot is unable to achieve. In this paper, we present a collaborative robot system designed for search and rescue missions in an unknown environment with obstacles. The system consists of an aerial robot and a ground robot. An extended Kalman filter (EKF) is used for robot pose estimation, and an online trajectory generation algorithm is implemented for dynamic obstacle avoidance of the ground robot. The aerial robot first surveys an area of interests and sources a number of targets. The ground robot is then guided by the aerial robot to reach the target location while at the same time avoids obstacles along the way using a laser range finder. The system is entirely autonomous, achieves maximum efficiency and releases the human operator from all low-level types of operations. A centralized EKF is implemented with the flexibility of easily being modified into a distributed EKF.**

I. Introduction

Robots have proven to be increasingly important to search and rescue missions given the improvement they can bring to both efficiency and quality. However, most of the tasks are undertaken by a single-robot or a centralized multi-robot system where there is no direct cooperation between individuals. In recent years, multi-robot cooperation has gain increasing attention from researchers, mainly because of the advantages it has over traditional single-robot systems. Unmanned aerial vehicle (UAV) and unmanned ground vehicle (UGV) are two of the most widely-used type of robots and the technologies associated with them are also relatively mature. Quadrotor UAVs are becoming more efficient and lower-cost solutions adopted in various industrial applications, especially in search and rescue missions. However, the quadrotor UAV has two non-negligible disadvantages: small payloads and limited flight time. Meanwhile, UGVs also suffer from a relatively diminished field-of-view.

In order to overcome the shortcomings mentioned above, we introduce a system that can maximize the capability of each single robot and improve the overall situational awareness. The essential framework of our method is an extended Kalman filter (EKF) that fuses sensor data from both the aerial and ground vehicle. A typical scenario is as follows: there is a target in an unknown environment which needs to be reached by the UGV in order to carry out search and rescue tasks. The UAV first surveys the environment in a lawn-mower fashion and records the target coordinates with respect to a global frame and sends it back to the UGV. The UAV then returns to the starting point and signals the UGV to start moving towards the target following a collision-free trajectory generated from the point clouds from the LiDAR. While the UGV is moving towards the target by following the trajectory, the UAV follows the UGV autonomously in a loosely constrained manner and keeps the UGV within its camera view. The EKF updates the robots' coordinates every time the UAV observes a reference marker. Our system is fully autonomous and requires no human intervention once the UAV is in the air.

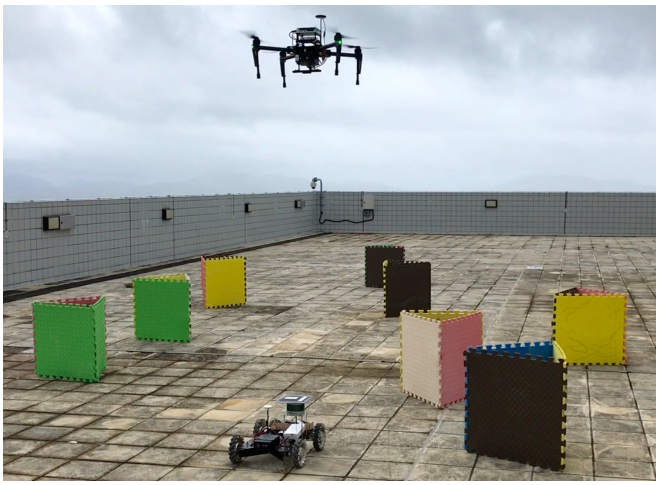


Fig. 1. Test in an outdoor environment with randomly placed obstacles. Video is available at https://youtu.be/3yQpnE65ZSA

The remaining sections of the paper are organized as follows. In section II we discuss related work on collaborative UAV-UGV robot systems, and give an overview of our system architecture in section III. The EKF-based pose estimation, system logic design, online path planning algorithm and UGV control scheme are detailed in sections IV, V, VI and VII, respectively. Finally the test results are evaluated in section VIII and we conclude the paper as well as propose future directions for work in section IX.

II. Related Work

There have been a great deal of research conducted on multi-robot systems, and here we provide a literature overview of some of the latest air-ground collaborative robotic systems.

A UAV-UGV system for object transportation is presented in [1] where a team of ground robots are guided by a drone and a human operator. The system has several follower robots, which automatically follow the leader robot using a

*These authors contributed equally to this work.

All authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. cshenab@connect.ust.hk, yzhangcg@connect.ust.hk, zlibc@connect.ust.hk, fgaoaa@connect.ust.hk, eeshaojie@ust.hk

vision-based tracking method. However, the system requires a human operator to determine the waypoints of the ground robot's trajectory.

In [2] a collaborative 3-D mapping system is introduced. The ground vehicle is tele-operated while the aerial robot can navigate through a complex multistory environment autonomously. Both robots are equipped with sensing devices, such as laser scanners, that can generate 3-D maps. The system shows the feasibility of UAV-UGV collaborative mapping in a search and rescue mission after disasters, and it also shows how heterogeneity in robot system design can overcome the weakness of homogeneous robots.

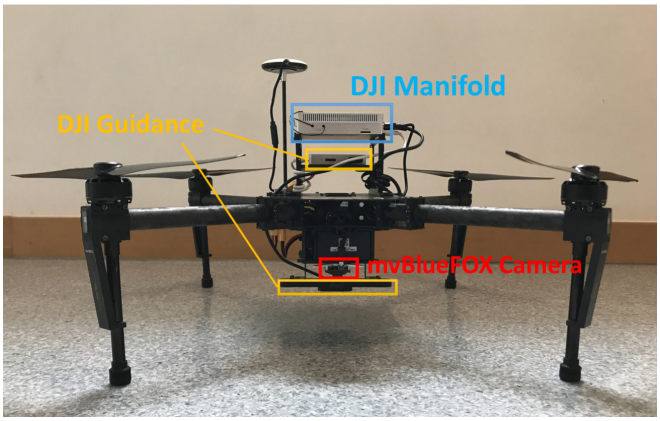
An aerial-ground robotic system is presented in [3]. The system utilizes a monocular camera on the UAV to measure the pose of obstacles relative to the UGV and obtain the global pose of the UGV using an EKF. A global map containing both UGV and obstacles can then be built. Similarly, our system uses an EKF to update the robot's pose in a global reference frame. However, our EKF also includes the pose of the UAV. We utilize one of the five sensor units of Guidance [4] to obtain a visual odometry during the propagation phase of the EKF.

The system in [5] uses a mock-up disaster scene to demonstrate the capability of a UAV-UGV system in search and rescue tasks. The aerial robot provides a bird's-eye view for the ground robot to plan its path to a target location in a previously unknown environment based on the A* algorithm, while removing obstacles along the way. However it requires the use of AprilTag to recognize obstacles, and all obstacles are assumed to be stationary. In contrast our system focuses more on real-life scenarios where obstacles are difficult for the UAV to distinguish, and they may also be moving around. We use a laser range finder to generate a trajectory directly on point clouds [6], and our system is capable of dynamic obstacle avoidance.

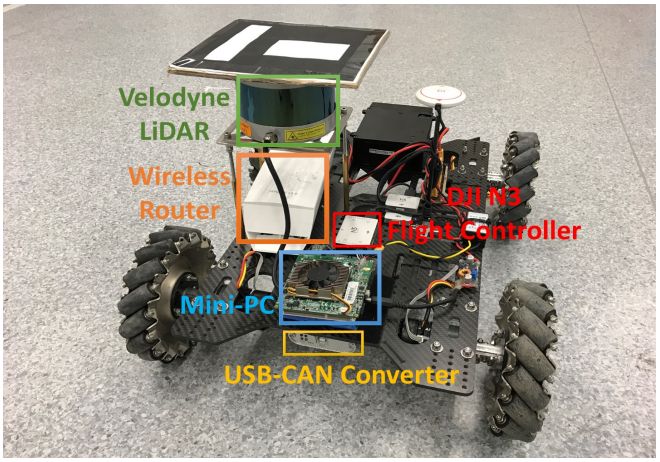
Other systems like [7] and [8] focus on global localization using an elevation map. A particle filter or an EKF is used for position tracking in the reference map. Our system differs from these in that we use the observation from the UAV's downward-facing monocular camera to update the pose estimation of both robots. Due to the complexity of a typical real-life search and rescue mission, we believe it is essential for the UAV to keep the UGV in its camera view at least in a loosely constrained manner so that the human operator can detect a hazardous situation timely and override the system control in case of an emergency that can not be handled by system autonomously.

III. System Overview

We begin by presenting the hardware architecture setup of our UAV-UGV system. The overall system block diagram is illustrated in Fig. 3:



(a) UAV hardware system setup



(b) UGV hardware system setup

Fig. 2. Hardware system architecture

A. Aerial Robot

We choose an off-the-shelf DJI Matrice 100[1] quadrotor as our UAV platform. This quadrotor platform has a highly-integrated propulsion system and flight controller. The quadrotor is powered by one 4500mAh DJI TB47D Li-Po battery, allowing a 15min flight time. It has various expansion I/Os available, such as UART serial ports. This platform also supports DJI onboard software development kit[2] (SDK) that we utilized for the basic attitude control of the quadrotor.

We installed the DJI Guidance[3] module with a downward-looking sensor unit consisting of a set of stereo cameras and an ultrasonic module. The Guidance module can provide a velocity and height measurement relative to the ground surface obtained from its embedded optical flow algorithm and the ultrasonic sensor, respectively. It is also compatible with the Matrice 100 quadrotor platform. When Guidance is connected to the Matrice 100, the quadrotor's flight controller will utilize its data to fuse with the onboard IMU's measurement, enabling a more stable and robust attitude control.

In order to perfrom search and rescue missions more efficiently, we also installed a downward-looking MatrixVi-

[1] http://www.dji.com/matrice100
[2] https://developer.dji.com/onboard-sdk/
[3] http://www.dji.com/guidance

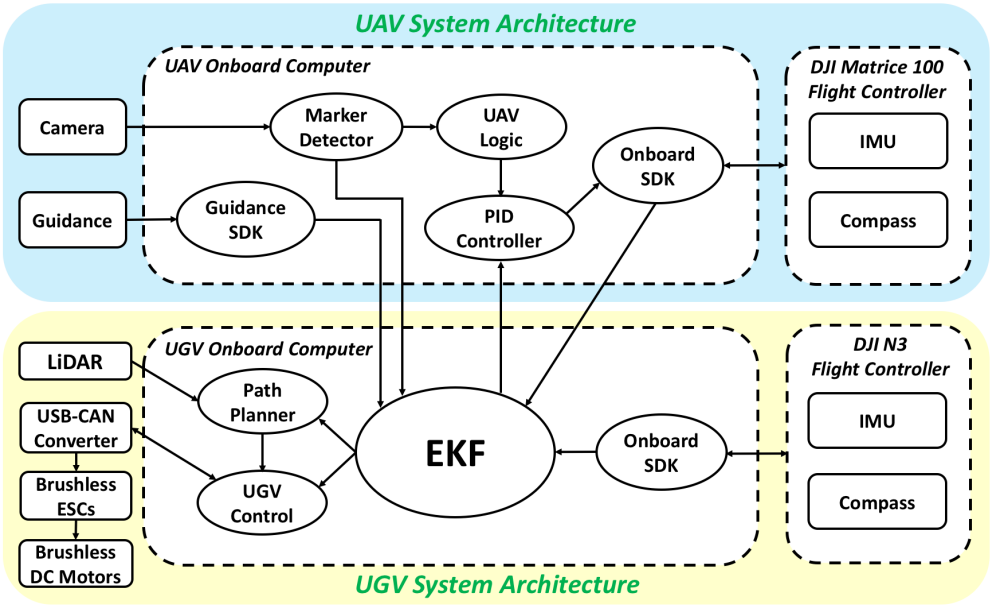


Fig. 3. System Block Diagram

sion mvBlueFOX-MLC200w[4] global shutter camera with 752x480 resolution on the UAV. A 90-degree lens is outfitted to provide a wide field-of-view.

A DJI Manifold[5] mini-PC is installed as the UAV's onboard computer. It runs Ubuntu 14.04 and all our programs utilize the Robot Operating System[6] (ROS) infrastructure. The Manifold communicates with the flight controller through a serial port with the DJI onboard SDK interface.

B. Ground Robot

Our UGV platform is designed and built completely from scratch. The chassis frame consists of a piece of carbon fiber plate with four aluminum motor mounts installed. The chassis is equipped with four Mecanum wheels driven by DJI RM3510 brushless DC motors with 19:1 gearboxes, such that the UGV can perform omni-directional movements with 2-DOF on the ground surface.

An mini computer powered by a dual-core Intel Core i7-5500U[7] processor is installed as the onboard computer of UGV. All the programs run on Ubuntu 14.04 with ROS, and communicate with other peripheral devices through its ethernet and USB ports.

We installed a DJI N3 flight controller[8] on the UGV, utilizing its IMU and compass module. Since N3 is compatible with DJI onboard SDK libraries, all the sensor data can be easily retreived through ROS. It transfers data to the onboard computer through its serial port with a USB-TTL converter.

A Velodyne-VLP16[9] LiDAR is placed on the UGV, and it communicates with the onboard computer through an ethernet port.

IV. EKF-Based State Estimation

In this section we present our implementation of the EKF for robot pose estimation. The basic framework and data flow is shown in Fig. 3. We primarily use a centralized EKF schema to fuse the sensor data retrieved from the different robots and estimate the pose of both of them. However, this centralized EKF is also designed in such a way that it can be decomposed into two separate EKFs running on the two robots easily. For instance, when the computational overhead is too much for a single processing unit (in our case, the computer on the UGV) to handle, only a small modification is needed in order to distribute the computational tasks to multiple robots while still maintaining data interdependencies as if it is running on a single master machine.

A. Propagation

The state vector and process model are designed as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{P}^g_{UAV} \\ \mathbf{P}^g_{UGV} \end{bmatrix} = \begin{bmatrix} x^g_{UAV} \\ y^g_{UAV} \\ z^g_{UAV} \\ x^g_{UGV} \\ y^g_{UGV} \\ z^g_{UGV} \end{bmatrix} \quad (1)$$

[4] https://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html
[5] http://www.dji.com/manifold
[6] http://www.ros.org/about-ros/
[7] https://ark.intel.com/products/85214/Intel-Core-i7-5500U-Processor-4M-Cache-up-to-3_00-GHz
[8] http://www.dji.com/n3
[9] http://velodynelidar.com/vlp-16.html

$$\dot{\mathbf{x}} = f(\mathbf{x}, u, n) = \begin{bmatrix} \mathbf{v}_{UAV}^{g} \\ \mathbf{v}_{UGV}^{g} \end{bmatrix} = \begin{bmatrix} v_{x_{UAV}}^{g} \\ v_{y_{UAV}}^{g} \\ v_{z_{UAV}}^{g} \\ v_{x_{UGV}}^{g} \\ v_{y_{UGV}}^{g} \\ v_{z_{UGV}}^{g} \end{bmatrix} = \begin{bmatrix} R_{g,UAV} \times \mathbf{v}_{UAV}^{b} \\ R_{g,UGV} \times \mathbf{v}_{UGV}^{b} \end{bmatrix}, \tag{2}$$

where $\mathbf{P}_{UAV}^{g}$ represents the position vector of the UAV in a global reference frame, and $\mathbf{v}_{UAV}^{g}$ represents the velocity vector of the UAV in the same global frame. $R_{g,UAV}$ represents the matrix denoting the rotation from the UAV body frame to the global frame, and $\mathbf{v}_{UAV}^{b}$ represents the velocity vector of the UAV in its own body frame.

The velocity data of the UAV and UGV in the process model are obtained from the DJI Guidance module and the DJI N3 flight controller, respectively. The Guidance module is a hybrid version of an inertial-assisted stereo visual odometer and an inertial-assisted monocular visual odometer [4] where an internal EKF is already implemented to fuse visual and inertial measurements. Its velocity measurement accuracy can achieve 0.04 m/s when around 2 meters above the ground. Similarly, we can obtain the velocity of the ground vehicle from the onboard N3 flight controller with relative high accuracy and reliability. Thus, it is unnecessary to include sensor bias in the state vector $\mathbf{x}$.

Compass modules are accompanied by the N3 flight controller on the UGV and the N1 flight controller on the UAV. They provide the vehicles' orientation information in quaternion form with respect to a North-East-Down world frame. The global frame is different from the world frame in that it follows the right-hand rule with its origin denoted by an ArUco marker [9] and its x-axis aligned with the vehicle heading. To start the search and rescue mission, an initial angle of the UGV's heading with respect to the world frame is recorded and it will be used as the x-axis of the global frame in the following process. The rotation matrices $R_{g,UAV}$ and $R_{g,UGV}$ can then be calculated from the quaternions published in real time by the SDK.

The propagation step is as follows:

$$\overline{\mu}_{\mathbf{t}} = \mu_{\mathbf{t-1}} + \delta t \times f(\mu_{\mathbf{t-1}}, u_t, 0) \tag{3}$$

$$\overline{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + V_t Q_t V_t^T, \tag{4}$$

where the matrices are calculated as:
linearization:

$$A_t = \frac{\partial f}{\partial x}|_{\mu_{\mathbf{t-1}}, u_t, 0} \tag{5}$$

$$U_t = \frac{\partial f}{\partial n}|_{\mu_{\mathbf{t-1}}, u_t, 0} \tag{6}$$

discretization:

$$F_t = I + \delta t A_t \tag{7}$$

$$V_t = \delta t U_t, \tag{8}$$

At time t, $\overline{\mu}_{\mathbf{t}}$ denotes the estimated mean of the state vector and $\overline{\Sigma}_t$ denotes the estimated covariance matrix.

The resulted 6×6 covariance matrix $\overline{\Sigma}_t$ and the system noise matrix Q_t are always diagonal due to our EKF design:

$$\overline{\Sigma}_t = \begin{bmatrix} \overline{\Sigma}_{UGV} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \overline{\Sigma}_{UGV} \end{bmatrix}_t \tag{9}$$

$$Q_t = \begin{bmatrix} Q_{UAV} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & Q_{UGV} \end{bmatrix}_t \tag{10}$$

B. Observation

The observation step is threefold:

1) When the downward-facing monocular camera on the UAV detects the marker on the UGV, the EKF will modify the observation model as:

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{0}_{3\times1} \\ T_{g,UGV} \end{bmatrix} + \mathbf{v}, \tag{11}$$

where $T_{g,UGV}$ is a 3-element vector denoting the translation from the origin to the UGV center in the global frame. It is calculated as:

$$T_{g,UGV} = T_{g,UAV} + R_{g,UAV} \cdot (T_{UAV,c} + R_{UAV,c} \cdot T_{c,UGV}), \tag{12}$$

$T_{c,UGV}$ denotes the translation from the camera to the UGV and it is obtained using the ArUco library. $R_{UAV,c}$ is set to be $\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $T_{UAV,c}$ equals to $\begin{bmatrix} 0 \\ 0 \\ 0.12 \end{bmatrix}$ by measurement. $R_{g,UAV}$ is calculated using compass data, and $\mathbf{v}$ represents observational Gaussian noise. Note that here $T_{g,UAV}$ is set to be the estimated position vector of the UAV in the global frame due to the robustness of the Guidance module. We approximate its velocity data to be the ground truth.

2) The downward-facing camera detects the marker placed at the target location:

$$\mathbf{z}_t = \begin{bmatrix} T_{g,UAV} \\ \mathbf{0}_{3\times1} \end{bmatrix} + \mathbf{v}, \tag{13}$$

where $T_{g,UAV}$ is a 3-element vector denoting the translation from the origin to the UAV center in the global frame. It is calculated as:

$$T_{g,UAV} = T_{g,tar} - R_{g,UAV} \cdot (T_{UAV,c} + R_{UAV,c} \cdot T_{c,tar}) \tag{14}$$

where $T_{g,tar}$ is recorded during the initial survey flight before the UGV begins moving. $T_{c,tar}$ is also obtained through the calibrated camera with the help of the ArUco library.

3) The downward-facing camera sees both markers, and the observation model will be modified as:

$$\mathbf{z}_t = \begin{bmatrix} T_{g,UAV} \\ T_{g,UGV} \end{bmatrix} + \mathbf{v}, \tag{15}$$

with all parameters calculated the same way as the illustration above.

C. Update

The update step follows the following equations:

$$\mu_{\mathbf{t}} = \overline{\mu}_{\mathbf{t}} + K_t(\mathbf{z}_t - g(\overline{\mu}_{\mathbf{t}}, 0)) \tag{16}$$

$$\Sigma_t = \overline{\Sigma}_t - K_t C_t \overline{\Sigma}_t, \tag{17}$$

where matrices are calculated as linearization:

$$C_t = \frac{\partial g}{\partial x}|_{\overline{\mu}_t, 0} \tag{18}$$

$$W_t = \frac{\partial g}{\partial v}|_{\overline{\mu}_t, 0} \tag{19}$$

and

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + W_t R W_t^T)^{-1} \tag{20}$$

$$\mathbf{z}_t = g(\mathbf{x}_t, v_t), \tag{21}$$

Here C_t equals $\begin{bmatrix} I_1 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & I_2 \end{bmatrix}$ and both I_1 and I_2 are 3×3 matrices. When the camera only detects the target marker, I_1 is set to be an identity matrix and I_2 is set to be a zero matrix; when the camera only detects the marker on the UGV, it is the opposite; when the camera sees both, C_t becomes an identity matrix. Such manipulation again allows the EKF to be decentralized.

Note that once the target position is recorded, it will be regarded as ground truth and acts as an absolute reference during the update phase. The reason for this is that our ultimate goal is to guide the UGV to the target. Even if the target position is inaccurate, the UGV can always reach the target using our EKF and path planning algorithm.

V. Mission Logic

The mission logic is illustrated in Algorithm. 1 and Algorithm. 2, which run on the UAV and UGV, respectively.

The UAV starts its mission by first going through INIT and TAKE-OFF modes. It then surveys an area in a lawn-mower pattern in order to maximize the chances of discovering the target. Currently the target is marked by an ArUco marker which can be recognized by the UAV when it flies near it. The UAV then moves back to the origin and begins to follow the UGV to reach the target.

The UGV's mission algorithm is triggered by the UAV after it flies back to the origin. The UGV is then controlled by a path planning algorithm to reach the target in a collision-free route.

The onboard computers are synchronized and both robots enter their last mode after the UGV reaches the target.

Algorithm 1 UAV Flight Logic

```
Enter INIT mode
while UAV control has not been obtained do
    Delay 0.5 second
end while
Enter TAKE-OFF mode
while UAV has not reached 2-meter height do
    Increase thrust via PID control
end while
Enter SEARCH-FOR-TARGETS mode
while waypoints P_i in a lawn-mower pattern have not all been reached do
    Continue to survey the area while recording the position of target markers T_{g,tar} ← T_{g,UAV} + T_{UAV,tar}
end while
Signal UGV to move
Enter UGV-TRACKING mode
while UGV has not completed task do
    UAV keeps tracking
    if Camera detects UGV marker then
        Update UGV position
    end if
    if Camera detects target marker then
        Update UAV position
    end if
end while
Enter LANDING mode
```

Algorithm 2 UGV Mission Logic

```
Enter INIT mode
while UGV has not received signal from UAV do
    Delay 0.5 second
end while
Enter GOING-TO-TARGETS mode
while Targets have not been fully traversed do
    while There is a feasible path to the next target P_i do
        Receive desired velocity v_x, v_y from path planning algorithm
        Velocity control via PID controller
    end while
end while
Enter STAND-BY mode
```

VI. Path Planning

Considering the safety of the navigation of the ground vehicle in complex environments, we adopt our previous work [6] to generate safe and smooth trajectories onboard the UGV in real-time. In our previous work [6], we directly extract free 3D space based on the unordered point cloud from the range measurement, following a optimization-based trajectory generation which confines the trajectory entirely within the free space. In this paper, we degenerate the method from being used in 3D space to 2D cases. We will briefly review the complete pipeline of the trajectory generation method and highlight some modifications in our applications

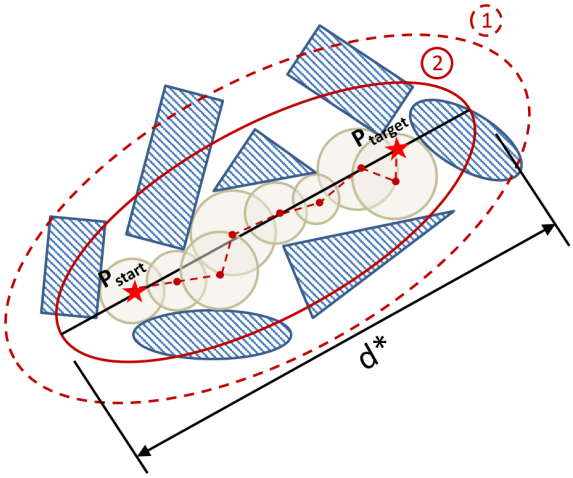


Fig. 4. Heuristic sampling domain updated in path finding. Red dashed ellipsoid with marker 1 is the last hyper-ellipsoid domain for generating samples, while red solid ellipsoid is the updated heuristic sampling region. Red stars indicate the start point and the target point for the path finding mission. Gray spheres are safe regions found by sampling and nearest neighbor query.

on ground vehicles.

Our path planning module includes the front-end path searching and the back-end trajectory generation modules. In complex environments, we follow the sampling-based path finding method to find a collision-free path that consists of simple convex shapes. The collision-free path extracts free space in the environments and trajectory can be generated within it to ensure the safety.

For the front-end path searching, we generate random samples in 2D space and evaluate the safe volume of each sample using nearest neighbour query. We keep samples which have enough safe volume for the vehicle to travel through as nodes , and add vertex between samples when they can be connected. After the generation of the graph, A* is used to search a minimum distance path on it. By controlling the minimum radius of the ball-shaped safe regions, which will be added into the graph as nodes, the minimum clearance of the path can be adjusted easily. And we also adopt a informed sampling heuristic [10] to help accelerate the convergency of the sampling procedure, as is shown in Fig. 4. We refer readers to [11] for details of the path finding details.

Having the path consists of safe circle regions, we can formulate the trajectory generation as a typical convex quadratic constrained quadratic program (QCQP). The basis of the trajectory we choose is simple monomial basis piecewise polynomial. And we write the cost function as a typical minimum snap formulation:

$$J = \sum_{\mu\in\{x,y\}} \int_0^T \left(\frac{d^{k_\phi} f_\mu(t)}{dt^{k_\phi}} \right)^2 dt, \tag{22}$$

The objective function can be written in a quadratic formulation $\mathbf{p}^T\mathbf{Q}_o\mathbf{p}$, where $\mathbf{p}$ is a vector containing all polynomial coefficients, and $\mathbf{Q}_o$ is the Hessian matrix of the objective function. We must also enforce a number of constraints to ensure the smoothness and safety of the trajectory:

1) Waypoint Constraints: First, we have to fix start and target positions and their first k_ϕ - 1 derivatives d_{ik}. If a fixed waypoint exists at the time of T_i, we have:

$$f_\mu^{(k)}(T_i) = d_{ik}, \tag{23}$$

2) Continuity Constraints: The trajectory must be continuous at all the k^{th} derivatives at the connecting point between two polynomial segments, where $0 \le k \le k_\phi - 1$:

$$\lim_{t\to T_i^-} f_\mu^{(k)}(t) = \lim_{t\to T_i^+} f_\mu^{(k)}(t), \tag{24}$$

3) Corridor Constraints: At the end time of each polynomial segment, the position of the trajectory must be inside the connected ball-shape safety regions:

$$\begin{cases} \sum_{\mu\in\{x,y\}} (f_\mu(T_i) - \mathbf{c}_i^\mu)^2 & \le r_i^2 \\ \sum_{\mu\in\{x,y\}} (f_\mu(T_i) - \mathbf{c}_{i+1}^\mu)^2 & \le r_{i+1}^2, \end{cases} \tag{25}$$

where $T_i (i = 1, 2, ..., M)$ are the end time of the piecewise polynomial, $\mathbf{c}_i$ is the ball center and r_i is the radius.

In this way, the trajectory generation problem is reformulated as a convex QCQP problem, which can be solved in polynomial time:

$$\begin{aligned} \min \quad & \mathbf{p}^T\mathbf{Q}_o\mathbf{p} \\ \text{s.t.} \quad & \mathbf{p}^T\mathbf{Q}_i\mathbf{p} + \mathbf{a}_i\mathbf{p} \le \mathbf{b}_i, \ \ i = 0, \cdots, M-1 \\ & \mathbf{A}_{eq}\mathbf{p} = \mathbf{b}_{eq}, \end{aligned} \tag{26}$$

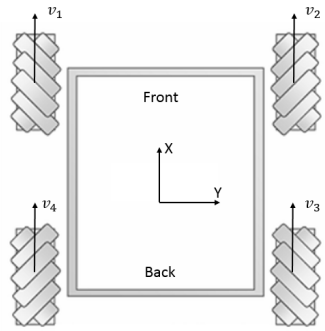


Fig. 5. Configuration of the UGV chassis.

VII. UGV Control

The UGV is equipped with four Mecanum wheels and is capable of performing omni-directional movements. We choose to keep the vehicle's heading unchanged by controlling its yaw angle. The complete control scheme consists of 4 PID controllers: yaw angle control, velocity v_x control, velocity v_y control and motor speed control. The vehicle chassis is shown in Fig. 5. The control algorithm on the UGV receives the desired target speed v_x and v_y generated from the path planning algorithm. The desired speed of each individual motor is calculated as:

$$\begin{cases} v_1 = v_x + v_y + \omega \\ v_2 = v_x - v_y - \omega \\ v_3 = v_x + v_y - \omega \\ v_4 = v_x - v_y + \omega, \end{cases} \tag{27}$$

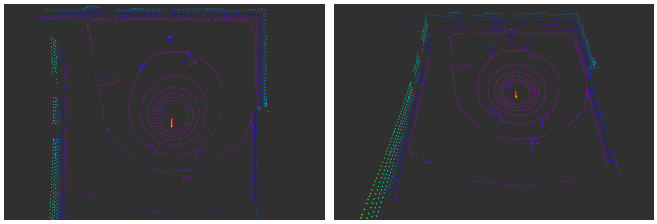
where ω represents the angular velocity offset in order to keep the yaw angle unchanged. The calculated $v_{1\ldots4}$ are then sent to the PID controllers to generate the actual control signals.

VIII. Results

Experiment results and the overall system performance are evaluated in this section. The system was tested in multiple outdoor locations and achieved the same robustness.

A. EKF-based Pose Estimation

We test our EKF-based robot pose estimation by first running a centralized EKF on the UGV. The EKF on the UGV receives all the necessary sensor data and propagates the position of both robots. With the UGVs odometry information relative to our defined global frame, we can project the original point cloud back into our global frame and obtain a steady global map shown in Fig. 6(a) and Fig. 6(b). All the subsequent tests are visualized using this global map.



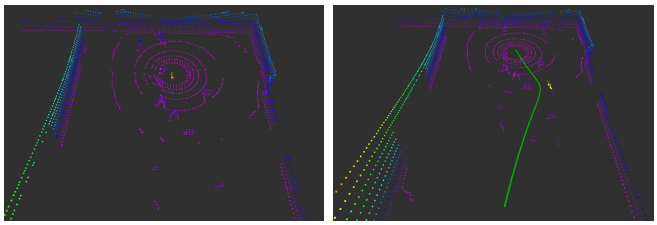
(a) A global map in 2-D view (b) A global map in 3-D view

Fig. 6. A global map generated by projecting point clouds back into the global frame

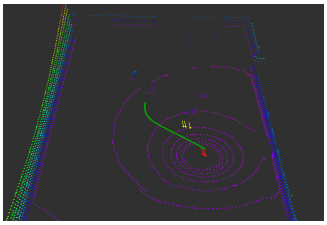
We assume the yaw angle of the UGV remains unchanged and the state vector $\mathbf{x}$ does not contain the orientation information of the robots, thus the global map sometimes experiences undesired rotation. However test results show that its effect on the overall performance is minimal. Also, the update step can always reduce the drift in position estimation to almost zero. This can be observed when the global map suddenly shifts when the UAV sees the marker on the UGV.

In a search and rescue mission, the starting state, mission execution state and ending state can be seen in Fig. 7(a), Fig. 7(b) and Fig. 7(c).

A distributed EKF scheme is also tested. The EKFs on the two robots have the same structure except that each of them only retrives information that is relevant to itself. The results prove the feasibility of our scheme and we believe it reduces the computation burden of the UGV when comparing it with the centralized scheme. We expect the difference to be more obvious when a greater number of robots or more complex algorithms are introduced into the system.



(a) The initial state of the system. Both of the robots are standing-by. (b) During mission execution state, the UGV follows the current trajectory while the UAV tracks the UGV.



(c) The UGV arrives at the target location.

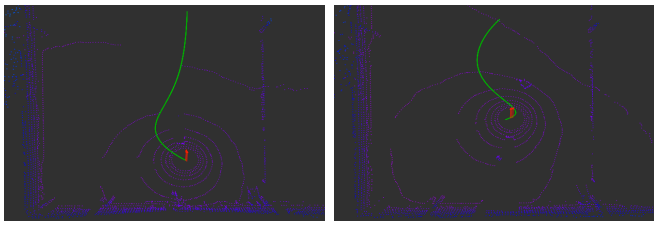
Fig. 7. An illustration of the three steps in a mock-up search and rescue mission. The red and yellow arrow denotes the UGV and the UAV, respectively.

B. Dynamic Obstacle Avoidance

The dynamic obstacle avoidance function is tested both independently and together with the whole system.

1) We first test the UGVs obstacle avoidance performance independently without the visual feedback of the UAV. The test fields are an open space next to the road and a standard tennis court, both with obstacle blocks randomly placed on the ground. When obstacles are stationary, our mission planning algorithm generates an optimal trajectory in the global frame and publishes the real-time target speed for the UGV to follow. The trajectory remains the same because there is no change in the environment. If the environment changes, a new trajectory starting from the UGVs current location may be generated depending on the exact circumstances. The algorithm constantly checks the clearance of the unexecuted trajectory, to see if obstacles exist within the minimum clearance 0.3 m in our tests. There are three cases where the trajectory changes: first case happens when a previous stationary obstacle starts to move and it will block the UGV if the trajectory remains unchanged; the second case happens when the UGV moves forward and detects a previously undetectable obstacle due to its location and the limitation of the laser ranger finder; and the third case is when the first and second cases happen simultaneously. Notably, in the second case, though the obstacle does not move, a new trajectory is still needed in order to avoid it. An example of the trajectory change is shown in Fig. 8(a) and Fig. 8(b).

The testing results show a small drift in the location estimation, which results in an offset between the UGVs



(a) Initial trajectory

(b) Re-generated trajectory after an obstacle is detected along the previous path

Fig. 8. Trajectory change where a moving obstacle is present

(a) Test in an open space near the road

(b) Test on a rooftop with a complex environment where obstacles are present

Fig. 9. Several outdoor tests are carried out in different environments.

actual path and the desired trajectory in visualization, and the trajectory sometimes also changes even when there is not a newly introduced obstacle that can affect the planned trajectory. This type of error can be greatly reduced when the visual observation from the UAV is integrated into the system and the update step is executed in the EKF.

2) The complete system is tested on a rooftop of size 20m*10m (shown in Fig. 9(b)) and several other open spaces such as Fig. 9(a). With the help of downward-facing camera, the UGVs initial position can overcome drift and stay around $(0, 0, 0)$ with high accuracy. This helps to reduce the offset in the position estimation before the UGV starts to move, and the generated trajectory is more accurate and thus less likely to change due to drift.

IX. Conclusion and Future Work

In this paper, we presented a fully-autonomous collaborative UAV-UGV robotic system explicitly designed for search and rescue missions in unknown outdoor environments where random obstacles may be present. An EKF-based collaborative localization algorithm and a dynamic path planning and obstacle avoidance algorithm are implemented and tested in multiple mock-up disaster scenarios where the targets are marked by ArUco markers.

However, there are several assumptions made in our system. First, the targets are mocked by markers whose information is known. We plan to introduce learning-based object detection algorithms in the future. Another direction is to attach special sensors to the UAV, such as infrared cameras, enabling the system to be deployed in practical applications such as searching for survivors or detecting fire in real-world disaster scenarios. Second, though in principle our system does not require GPS signal for localization, it still relies on the compass module whose performance can be significantly affected when a magnetic interference is present. We plan to implement a monocular visual-inertial state estimation algorithm [12] to obtain an accurate odometry of the robots instead of relying on the magnetometer.

References

[1] E. H. C. Harik, F. Gurin, F. Guinand, J. F. Breth, and H. Pelvillain, "Uav-ugv cooperation for objects transportation in an industrial area," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, March 2015, pp. 547–552.

[2] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *J. Field Robotics*, vol. 29, pp. 832–841, 2012.

[3] M. Garzón, J. Valente, D. Zapata, and A. Barrientos, "An aerial-ground robotic system for navigation and obstacle mapping in large outdoor areas," *Sensors*, vol. 13, p. 1247, 2013. [Online]. Available: http://www.mdpi.com/1424-8220/13/1/1247

[4] G. Zhou, L. Fang, K. Tang, H. Zhang, K. Wang, and K. Yang, "Guidance: A visual sensing platform for robotic applications," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2015, pp. 9–14.

[5] E. Mueggler, M. Faessler, F. Fontana, and D. Scaramuzza, "Aerial-guided navigation of a ground robot among movable obstacles," in *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, Oct 2014, pp. 1–8.

[6] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation," in *Proc. of the IEEE Intl. Symp. on Safety, Security, and Rescue Robot.*, Lausanne, Switzerland, Oct. 2016.

[7] R. Kslin, P. Fankhauser, E. Stumm, Z. Taylor, E. Mueggler, J. Delmerico, D. Scaramuzza, R. Siegwart, and M. Hutter, "Collaborative localization of aerial and ground robots through elevation maps," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct 2016, pp. 284–290.

[8] P. Fankhauser, M. Bloesch, P. Krsi, R. Diethelm, M. Wermelinger, T. Schneider, M. Dymczyk, M. Hutter, and R. Siegwart, "Collaborative navigation for flying and walking robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2859–2866.

[9] S. G. Jurado, R. M. Salinas, F. J. M. Cuevas, and M. J. M. Jimenez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320314000235

[10] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Chicago, IL, Sept 2014.

[11] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Vancouver, Cananda, Sep. 2017.

[12] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, pp. n/a–n/a. [Online]. Available: http://dx.doi.org/10.1002/rob.21732