# Autonomous Navigation of Unmanned Aerial Vehicle (UAV)

Aashi Manglik, Krishnraj Singh Gaur, Rachit Agarwal

***Abstract*— This work describes a system for autonomous navigation of an unmanned aerial vehicle in a known environment. The system requires a knowledge of location of the UAV(Unmanned Aerial Vehicle) within the map which is accomplished by applying Kalman filter on the data provided by optical flow sensor and IMU(Inertial Measurement Unit) sensor. Our grid-based path planning algorithm use discrete state transitions that allows the robot's motion in particular headings (0, $\frac{\pi}{2}$ and $\frac{\pi}{4}$). The presented work is implemented on a high performance quadrotor which autonomously followed the optimised path in the known domain.**

## I. INTRODUCTION

In recent years there has been a rapid development of autonomous unmanned aircraft equipped with autonomous control devices called UAVs. At present the major UAV applications are in defense and the main investments are driven by future military scenarios [1]. Most military unmanned aerial vehicles are primarily used for intelligence, surveillance, reconnaissance and strikes. The next generation of UAVs will execute more complex missions such as air combat, electronic attack and network node relay. However, it can be said that the infinite possibilities of utilizing their outstanding characteristics for civil application remain hidden. Potential applications of UAVs capable of autonomous navigation include inspection of terrains, aerial mapping and meteorology, environmental monitoring, disaster and crisis management and research by university laboratories.

In mobile robot navigation, the environment is mostly represented in the form of grid and then the aim is to plan a path from some initial robot location to desired goal position. The grid-based map of the environment can be represented in two ways. In binary representation, each grid cell either represents an obstacle or free space. The other way is to assign a cost to each grid cell which is in proportion to the difficulty of traversing that particular area of environment.

The path planning algorithms are extensively discussed in the robotics research literature [2], [3]. A* search is the most common and efficient algorithm that uses a heuristic to compute the shortest path from a given location to goal [4]. A* is efficient when the environment is static and fully-observable and the action model is fully-deterministic. Since many applications domains do not provide the luxury of an a priori map, algorithms such as D*, incremental A* and D* Lite were proposed [5]. These algorithms are the extensions of A* which repairs the solution path when a change is detected in the map. Hybrid A* is another famous algorithm, particularly in the context of self-driving cars. The car cannot turn around like a human and thus there arises a need to apply hybrid A* to compute a drivable path to the goal [6]. As shown in Fig.1, it is a variant of the A* algorithm with a modified state-update rule that captures continuous-state data in the discrete search nodes of A*. Therefore, it associates with each grid cell a continuous 3D state of the vehicle. Hybrid A* is computationally more intensive than A* and would not be of much use in case of quadrotor which can easily rotate about its axis. While doing the path planning for 2D grid, the common approach is to plan from the center of each grid cell and the transitions are allowed only between the centers of adjacent grid cells. Applying the linear interpolation between the two centers, the computed paths are optimal and effective in practical applications of robotic systems. In Section II, we have provided the detailed description of the two versions of A* search algorithm implemented. However, Ferguson and Stentz [7] presented a Field D* algorithm, shown in Fig.1, in which instead of assigning nodes to the centers of grid cells, they have assigned nodes to the corners of each grid cell, with edges connecting nodes that reside at corners of the same grid cell.

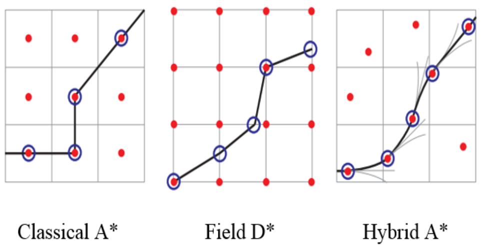

Fig. 1. Illustrations of Classical A*, Field D* and Hybrid A* Search.

To achieve autonomous navigation, the robot needs to know where is it currently located in the global map. This can be facilitated by equipping the robot with sensors suitable for localisation throughout the path. Sensors have inherent limitations such as drift and noise and thus, no independent sensor could estimate the pose of the robot accurately. To overcome this limitation of sensors, two or more sensors' measurements are fused to estimate the robot's pose applying Kalman filter algorithm [8]. The details of the sensors used and data fusion algorithm implemented are described in section III.

The localisation and path planning algorithms presented in this work are applicable for the known environment and were implemented on a UAV, in particular, a quadrotor platform. The architecture and working of the on-board system of the quadrotor is explained in section IV.

## II. A* Path Planning

A* algorithm is the most commonly used algorithm for path planning. The secret to its success is that it combines the two pieces of information. First, the paramter $g(n)$ is favoring vertices that are close to the starting point (similar to Dijkstra's algorithm explained in [9]) and other parameter $h(n)$ is favoring the vertices that are close to the goal (Greedy Best-First-Search Algorithm). In the standard terminology used, when talking about A*, $g(n)$ represents the exact cost of the path from the starting point to any vertex $n$, and $h(n)$ represents the heuristic estimated cost from vertex $n$ to the goal. A* balances the two as it moves from the starting point to the goal. Each time through the main loop, it examines the vertex $n$ that has the lowest $f(n)$ given by Eq.1.

$$f(n) = g(n) + h(n) \tag{1}$$

The classical A* algorithm, as provided in section II-A, is stated for two headings, i.e, 0 and $\frac{\pi}{2}$. Thus, the action model is restricted to four directions and using the manhattan distance as heuristic, we are able to get the optimised path. The manhattan distance is defined as the distance between the two points in the grid based on a strictly horizontal and/or vertical path. When the classical A* was applied, the expected path to be taken by UAV is visualised as shown in Fig.2.

Since a quadcopter can easily turn about its axis, while hovering at a point, and take the heading of $\frac{\pi}{4}$, it can easily follow a diagonal path. So, the A* search Algorithm was modified as described in II-B, in the action model was changed to allow movement in 8 directions and Euclidean distance was used to compute the heuristic. Hence, implmenting this A* search Algorithm, the quarotor is no longer expected to move in redundant steps as shown in Fig.2 and chooses the optimized path which constitutes diagonal motion,as shown in fig.3.

### A. Classical A* Algorithm

```
ComputePath()
01: initialise grid
02: initialise goal
03: initialise open list
04: initialise closed list
05: place starting node in open
06: while open is not empty
07:     pop the node with least $f$ off the open list, call it $p$
08:     generate $p$'s 4 successors and set their parent to $p$
09:     for each successor $s$
09:         if $s$ is goal, stop
10:         $s.g = p.g + 1$
11:         $s.h$ = manhattan distance from goal to $s$
12:         $s.f = s.g + s.h$
13:         if node with same position as $s$ is in open list
             and has a lesser $f$ than $s$, skip this $s$
14:         else add $s$ to open list
15:     end for
16:     push $p$ on closed list
17: end while
```

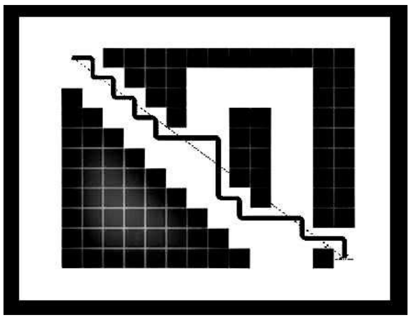
Fig. 2. Path given by Classical A* Algorithm.

### B. Modified A* Algorithm

The modification lies in line 8 where the number of successors generated is eight, instead of four as shown in section II-A, and in lines 10 and 11, where transition cost and heurisric are assigned as the euclidean distance between node $n$ and its parent and euclidean distance between node $n$ and goal respectively. In other words, the transition cost is $\sqrt{2}$ when the UAV heads diagonally and it remains to be 1 as in II-A while moving horizontally/vertically.

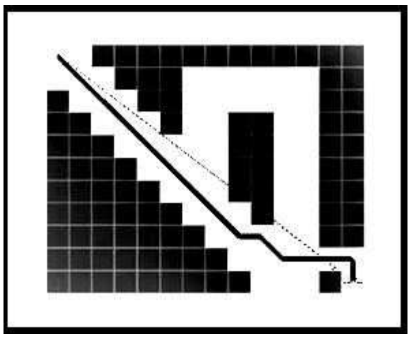
Fig. 3. Path given by Modified A* Algorithm.

```
ComputeModifiedPath()
01: initialise grid
02: initialise goal
03: initialise open list
04: initialise closed list
05: place starting node in open
06: while open is not empty
07:     pop the node with least $f$ off the open list, call it $p$
08:     generate $p$'s 8 successors and set their parent to $p$
09:     for each successor $s$
09:         if $s$ is goal, stop
10:         $s.g = p.g$ + euclidean distance between $s$ and $p$
11:         $s.h$ = euclidean distance from goal to $s$
12:         $s.f = s.g + s.h$
13:         if node with same position as $s$ is in open list
             and has a lesser $f$ than $s$, skip this $s$
```

```
14:         else add $s$ to open list
15:     end for
16:     push $p$ on closed list
17: end while
```

## III. Localisation

To implement the path planning algorithm on UAV, a precise localisation is required. Many sensors such as Inertial Measurement Unit (IMU), laser and cameras are frequently used in solving the UAV localisation problem. Most of the times, the robot is equipped with IMU sensor for spatial navigation since IMUs provide high frequency acceleration and rotation rate data that can be used independent of vehicle models. However, using a low cost IMU gives a large amount of error in the acceleration data which can not be used alone for the localisation. The optical flow cameras provide the velocity estimate but cannot be used to compute pose since they suffer from drift due to accumulation in error during flight. Therefore, the IMU data is fused with the optical flow to precisely estimate the UAV's pose. The use of Kalman filter for the data fusion of sensors' measurements is a well-established technique and is presented in III-B.

### A. Sensor Models

- Inertial Measurement Unit (IMU):
  The IMU is used as the core sensing device. It measures the acceleration ($a_x$, $a_y$, $a_z$) and the rotation rates of the quadrotor platform at high update rates, which can be transformed to get the attitude of the platform.

- Optical Flow Camera (Px4flow Smart Camera):
  The Optical Flow Camera measure the velocity ($v_x$ and $v_y$) by sensing the apparent change in the visual scene caused by relative motion of object and camera.

### B. Kalman Filter

The Kalman filter has been applied along $x$ and $y$ axis separately. The variance in the values of acceleration $a_x$ and $a_y$ are represented by $\sigma_x$ and $\sigma_y$ respectively. For the px4flow smart camera, the variance in $v_x$ and $v_y$ are represented by $R_x$ and $R_y$ respectively. The variance values were found to be as follows:
$\sigma_x$ = 0.007868
$\sigma_y$ = 0.00871
$R_x$ = 0.006017
$R_y$ = 0.00995

Algorithm Kalman Filter ($\mu_t - 1$,$\Sigma_t - 1$):
$x_t$: the position estimate at time $t$
$v_t$: the velocity estimate at time $t$
$dt$: delta time between two states
$K_t$: Kalman gain at time $t$
$z_t$: sensor data at time $t$
$z_v$: velocity data from the Optical flow camera

$$\begin{aligned}
\mu_t &= \begin{bmatrix} x_t \\ v_t \end{bmatrix} \\
A &= \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \\
B &= \begin{bmatrix} \frac{dt^2}{2} \\ dt \end{bmatrix} \\
H &= \begin{bmatrix} 0 & 1 \end{bmatrix} \\
z_t &= \begin{bmatrix} 0 \\ z_v \end{bmatrix}
\end{aligned}$$

*1) Prediction Step:*

$$\overline{\mu}_t = A\mu_{t-1} + Bu_t \tag{2}$$
$$\overline{\Sigma}_t = A\Sigma_{t-1}A^T + B\sigma^2 B^T \tag{3}$$

*2) Update Step:*

$$K_t = \overline{\Sigma_t} H^T (H\overline{\Sigma_t} H^T + R) \tag{4}$$
$$\mu_t = \overline{\mu_t} + K(z_t - H\overline{\mu_t}) \tag{5}$$
$$\Sigma_t = (I - K_t H)\overline{\Sigma_t} \tag{6}$$

return $\mu_t$, $\Sigma_t$

## IV. SYSTEM ARCHITECTURE

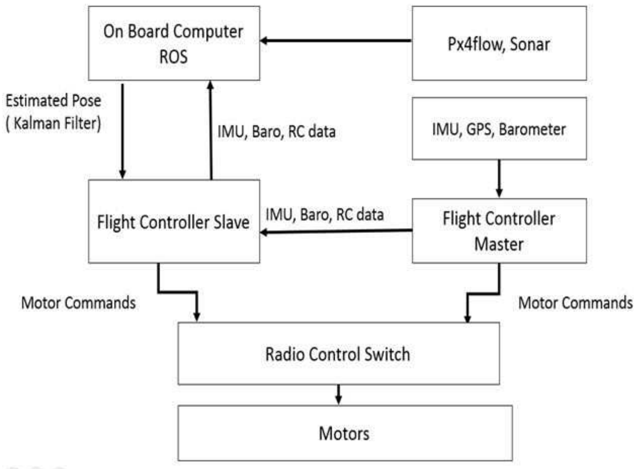

Fig. 4. Hardware Architecture.

This section describes the UAV's system architecture. A flow chart of the hardware architecture is shown in Fig.4. There are three Radio Frequency communication channels. The first is a bidirectional link between the OBC(onboard computer) and the PC ground station using wifi. The ground station uplinks the ROS(Robot OPerating System) commands to run the code on OBC. The second RF channel is a link between OBC and a high level processor which uses mavros drivers to communicate at 20 Hz using mavlink protocol. The third RF link is a 2.4 GHz RC uplink to the UAV which is used to bypass the autopilot [10]. The

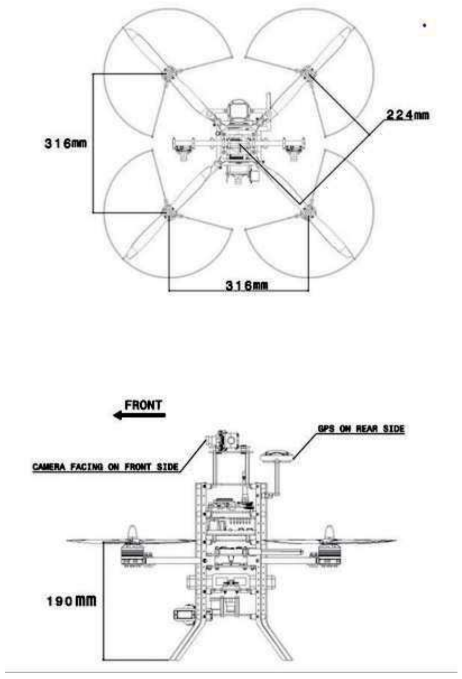

Fig. 5. Quadrotor Airframe

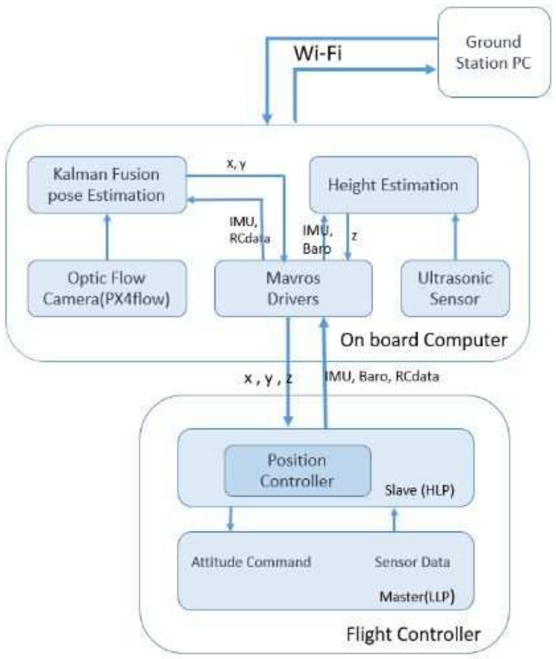

Fig. 6. Programming Overview Flowchart

bypass capability is used as a fail-safe mode that allows new algorithms to be rapidly tested and debugged. There are two flight controllers in the system- Master and Slave. Flight controller runs a PID controller to fly, stabilize and navigate the UAV. It takes data from various sensors, processes them and gives commands to motor for desired control. Master controller extracts data from sensors like IMU, GPS and barometer to compute motor commands. In slave controller, we get estimated pose from OBC which is computed by Kalman filter fusion of optical flow data from Px4flow smart camera and acceleration data from IMU sensor. Using this estimated pose, the slave controller also computes the motor commands. Then, using the third 2.4GHz RF link we can switch between the master and slave processor.

## A. *UAV Hardware*

The quadrotor platform used is a high performance system with multiple subsystems working together to collaboratively deliver flexible output. The UAV has multi-stack made up of carbon fiber structure. This makes a light rugged and configurable design and helps in integrating various other subsystems swiftly and efficiently. The graphics shown in Fig.13 provides overall idea of airframe.

The UAV has lithium polymer battery which powers brushless DC motors. It has a high performance single board computer running Linux(Ubuntu 14.04) with ROS pre-installed. This OBC can communicate with flight controller via USB to receive flight controller data and send control commands or sensor outputs. It also has following sensors :

- Inertial Measurement Unit (IMU)
- Barometer
- GPS
- Px4flow optical flow smart camera
- Matrix vision Blue Fox camera

The hardware architecture provides a flexible experimental test bed to explore a variety of algorithms that enable autonomous behavior for the UAVs.

## B. *Programming Overview*

The On-board computer holds all the ROS packages and does the processing of all the ROS nodes. The two main ROS packages responsible for navigation are discussed here. First package contains the Kalman fusion and A* path planning algorithms which outputs $x$ and $y$ pose estimate of the quadrotor along with the target location $(x_t, y_t)$. The Kalman filter uses the accelerometer data, optical flow data and the altitude information. The altitude information is required to transform the data in body fixed frame to the earth fixed frame.The second package contains the height estimation fusion algorithm which outputs the height ($z$) estimate. The ROS mavros drivers are responsible for the commutation and transfer of required data to-and-from the high level processor. The target location and the current location are then processed in the high level processor where a PID position controller runs to provide final PWM (Pulse Width Modulation) signals to the motors as output, as shown in Fig.4.

## V. RESULTS

In this section, the results of real time flight test are presented which demonstrate the performance of A* path

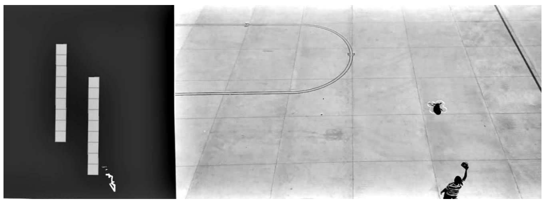

Fig. 7. Flight test at time t=1s

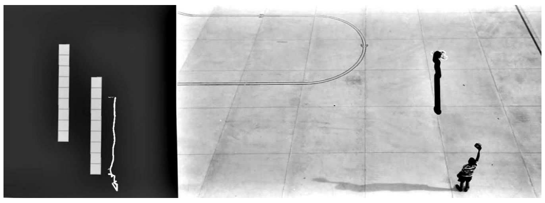

Fig. 8. Flight test at time t=5s

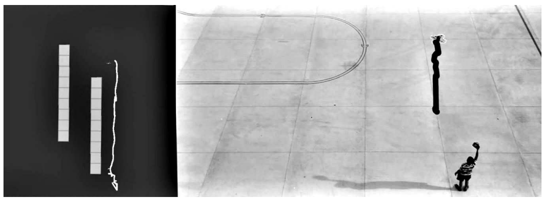

Fig. 9. Flight test at time t=10s

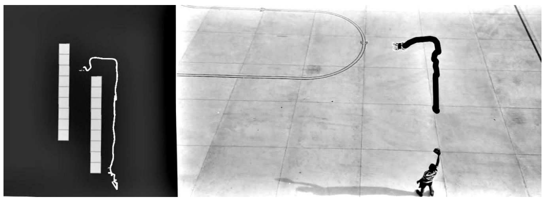

Fig. 10. Flight test at time t=15s

planning algorithm and Kalman based localization when implemented on the UAV. The UAV is provided with the prior information of map in the form of 2D matrix. There are two straight walls in the area, each considered as an obstacle. The UAV is restricted to avoid the wall by a distance of 0.5 metres and restricted to the speed of 0.5m/s. The obstacles are depicted as solid grey grid and the path followed by UAV as a thin, solid white line in the rviz visualisation tool as shown in left hand side of the Figure 7 through Figure 14. The flight test with the UAV was initialised after taking it to a height of about 2 metres above its starting position. The measurement uncertainty for the location of the obstacle aircraft is set to 0.1 m. UAV is then switched to attitude hold and is allowed to autonomously follow the path to the goal location avoiding the obstacles. Figures 7 through Figure 14 show the results of the test at various time intervals.

## VI. CONCLUSIONS

This report has described an approach to autonomous navigation for UAVs. In particular, the UAV hardware, low-cost sensors, and a computationally efficient approach to path planning, trajectory generation and localisation are presented. The algorithms developed for 2D path planning and localisation are tested on UAV. The UAV is initially launched by the operator manually and once the UAV has reached an operating attitude, the path planning starts directing the UAV via waypoint and heading commands.

The two versions of A* path planning algorithm were tested as a solution to robot motion planning problem in the static workspace. The classical A* algorithm allowed 4 directional movement and used manhattan distance for the heuristic calculation. To overcome the limitations of restricted movement in 4 directions, an improved version of A* algorithm was proposed which allowed 8 directional movement and used Euclidean distance for heuristic computation and thus allowed the UAV to choose the most optimized path. The presented work on path planning can be applied in general to any type of robot.

TABLE I
STANDARD DEVIATION IN ESTIMATED POSE

| Standard deviation | SVO | Kalman Filter |
|---|---|---|
| $x$(in m) | 0.1300 | 0.5301 |
| $y$(in m) | 0.1436 | 0.7436 |

Further, a Kalman filter based method was presented for mobile robot localisation and compared with the already established semi-visual odometry based localization [11]. The results were then compared in hovering condition of the UAV and standard deviation has been presented in table I for a flight time of 5 minutes. The Kalman filter was also compared with the stand alone optical flow data. The Kalman filter was found to perform much better as compared to the pose estimated via stand alone optical flow sensor.

## ACKNOWLEDGMENT

We would like to thank Aarav Unmanned Systems for providing us with the quadrotor platform(UAV) and Dr. Gaurav Pandey for the esteemed guidance.

## REFERENCES

[1] A. M. Samad, N. Kamarulzaman, M. A. Hamdani, T. A. Mastor, and K. A. Hashim, "The potential of unmanned aerial vehicle (uav) for civilian and mapping application," in *System Engineering and Technology (ICSET), 2013 IEEE 3rd International Conference on*, Aug 2013, pp. 313–318.

[2] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi, "An evolution based path planning algorithm for autonomous motion of a uav through uncertain environments," in *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, vol. 2, 2002, pp. 8D2–1–8D2–12 vol.2.

[3] A. T. Stentz , "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '94)*, vol. 4, May 1994, pp. 3310 – 3317.

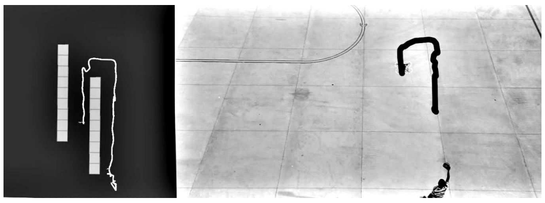
Fig. 11. Flight test at time t=20s

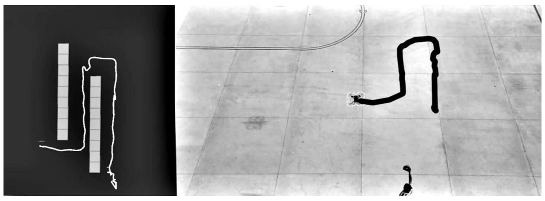
Fig. 12. Flight test at time t=25s

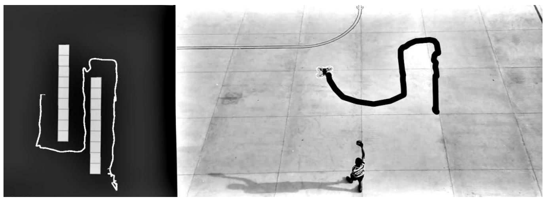
Fig. 13. Flight test at time t=30s

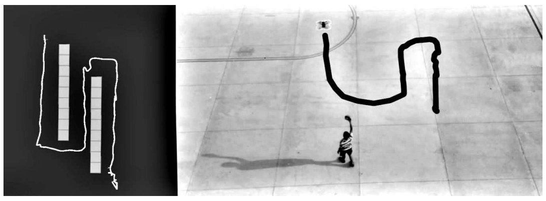
Fig. 14. Flight test at time t=35s

[4] N. Sariff and N. Buniyamin, "An overview of autonomous mobile robot path planning algorithms," in *Research and Development, 2006. SCOReD 2006. 4th Student Conference on*, June 2006, pp. 183–188.

[5] S. Koenig and M. Likhachev, "D* lite." in *AAAI/IAAI*, 2002, pp. 476–483.

[6] D. Doglov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," vol. 29, April 2010, pp. 485–501.

[7] D. Ferguson and A. Stentz, "Field d*: An interpolation-based path planner and replanner," in *Robotics research*. Springer Berlin Heidelberg, 2007, pp. 239–253.

[8] I. Ashokaraj, A. Tsourdos, P. Silson, and B. White, "Sensor based robot localisation and navigation: using interval analysis and extended kalman filter," in *Control Conference, 2004. 5th Asian*, vol. 2, July 2004, pp. 1086–1093 Vol.2.

[9] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *Mechanic Automation and Control Engineering (MACE), 2011 Second International Conference on*, July 2011, pp. 1067–1069.

[10] R. W. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich, "Autonomous vehicle technologies for small fixed-wing uavs," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 1, pp. 92–108, 2005.

[11] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.