Proceedings of the 4th
International Conference on Robotics and Mechatronics
October 26-28, 2016, Tehran, Iran

Autonomous Flight and Obstacle Avoidance of a Quadrotor By Monocular SLAM

Omid Esrafilian and Hamid D. Taghirad *Senior Member IEEE,*
Advanced Robotics and Automated Systems (ARAS), Industrial Control Center of Excellence (ICCE),
Faculty of Electrical Engineering, K. N. Toosi University of Technology,
esrafilian.omid@email.kntu.ac.ir, taghirad@kntu.ac.ir.

Abstract*—In this paper, a monocular vision based autonomous flight and obstacle avoidance system for a commercial quadrotor is presented. The video stream of the front camera and the navigation data measured by the drone is sent to the ground station laptop via wireless connection. Received data processed by the vision based ORB-SLAM to compute the 3D position of the robot and the environment 3D sparse map in the form of point cloud. An algorithm is proposed for enrichment of the reconstructed map, and furthermore, a Kalman Filter is used for sensor fusion. The scaling factor of the monocular slam is calculated by the linear fitting. Moreover, a PID controller is designed for 3D position control. Finally, by means of the potential field method and Rapidly exploring Random Tree (RRT) path planning algorithm, a collision-free road map is generated. Moreover, experimental verifications of the proposed algorithms are reported.

***Keywords*—Autonomous navigation; obstacle avoidance; autonomous quadrotor; monocular SLAM**

I. Introduction

It goes without saying that in such a sophisticated world the prominent role of robots, and especially aerial robots, may not be dismissed and autonomous obstacle avoidance is the key component for the success of these micro aerial vehicles (MAVs). When it comes to the MAVs, controversy surrounds the issue how to reduce the weight of robot, because of the highly constrained payload. As a result, cameras are the appropriate and attractive lightweight sensors to provide rich information. So far, many researches have been conducted in the autonomous flight of quadrotor using GPS-denied in unknown environments but this field is still an open research area.

Garcia Carrillo et.al, work may be considered as a representative work on autonomous navigation of MAVs using on-board image processing in GPS-denied environments, [1]. The aim of this study is the road following using a quadrotor with an onboard image processing system. Their approach generally consists of two parts, namely estimating and tracking the road map without any prior information about the environment and the desired path for tracking. Also, a switching controller is designed for stabilizing the lateral position of the Unmanned Aerial Vehicle (UAV) with respect to the road. This approach is limited to segmented straight line which exists in the image and they have not considered obstacle avoidance. Reference [2], presents a solution for enabling a quadrotor to autonomously navigate, explore and locate specific objects in unstructured and unknown indoor environments using a laser scanner and a stereo camera. In order to accomplish that, a quadrotor platform is used, which is capable of carrying a laser scanner, stereo camera and an onboard computer. However, the aim of our research is to develop a lightweight system for autonomous flight and obstacle avoidance using only a monocular camera and on-board sensors. Reference [3], proposed an autonomous navigation system on a quadcopter in an unknown environment. In this work a monocular SLAM is proposed that calculates the 3D position of the robot and fuses the output with the measured navigation data, which sent to the ground station using EKF. However, no solutions for autonomous obstacle avoidance are proposed in this paper. Sven, et al. [4], report an Autonomous corridor flight of a UAV using an RGB-D camera, in which SLAM by RGB-D camera is an accurate real time method. Compared to the color camera, RGB-D camera has more weight and requires higher power consumption, which is very restrictive for our application. Despite the development of visual SLAM, it is still very challenging to use a monocular SLAM for autonomy. The major concern is the limited computational capability of MAVs, for that reason often data are communicated to a ground station using Wi-Fi connection link. So, the visual SLAM must be robust to the data loss and probable measurement noise in an unstructured environment.

In this paper, an autonomous flight and obstacle avoidance system for a low cost commercial quadrotor called AR.Drone is presented. The robot sends video data stream to the ground station using Wi-Fi connection link. On the ground station the data are processed by ORB-SLAM in order to determine the 3D position of the robot and the 3D sparse map of the environment. The scalar factor is calculated by linear fitting and then visual odometry data are fused to the navigation data measured by robot on-board sensors to obtain more accurate measurements by using Kalman Filter. A PID controller is design for the position control of the drone in three dimensions. Since the reconstructed 3D map is sparse and cannot be used for obstacle avoidance, an algorithm is proposed in this paper to make the output map more dense and precise. Finally, by applying some path planning methods an appropriate and collision-free path is generated.

The remainder of the paper is organized as follows. Section 2 gives an overview of the monocular ORB-SLAM and the proposed vision-base pose estimation including the KF and scaling factor. Section 3 describes our algorithm for enrichment of the reconstructed 3D map. We generate the safe and collision-free trajectory for autonomous flight of the robot by utilizing some path planning methods which is elaborated in Section 4. Finally, Section 5 provides a performance evaluation, presents experimental results and in Section 6 concluding remarks are given. To verify the proposed method in a series of experiments with a real quadrotor, many experimental scenarios are followed, whose videos are available online at this link.

II. Vison-Based Odometry and Mapping

A. Monocular ORB-SLAM

ORB-SLAM is a sparse and feature-based monocular SLAM system that operates in real time and in wide assortment of indoor and outdoor environments and is robust to rapid motions [5]. The system makes the pose estimated more accurate by utilizing loop closing and re-localizing as well as this system is initialized automatically. The system consists of three threads that run in parallel, namely tracking, local mapping and loop closing thread. FAST corner detection and ORB extractor is used for feature tracking. The tracking and feature matching are necessary for camera localization which are performed with every frame and decides when to insert a new *keyframe*. After track the extracted features in two consecutive frames, the first estimation of camera pose performed and then it is made more precise by utilizing the Bundle Adjustment (BA) optimization. The place recognition is conducted by the Bag of Word (BoW) when tracking is lost.

Processing new *keyframes*, Performing Local Bundle Adjustment and optimal reconstruction achievement in the surroundings of the camera pose are done in local mapping thread. Once a new *keyframe* is added, the loop closing thread searches for loops with other ones. When a loop is detected, both side of the loop are aligned and duplicated points are fused. The key point is that the optimization is performed over the Essential Graph.

B. Scale Calculation

One of the major problems in using the monocular SLAM is the scale difference between visual SLAM output and the real world. Therefore, before applying the SLAM measurements, the scale factor $\lambda \in \mathbb{R}^+$ of a monocular SLAM system shall be calculated. Since the scale is almost identical in three dimensions, it is enough to calculate the scale factor in one dimension. For this mean, if we consider the robot coordinate as given in Fig. 1, then the scale factor may be computed by measuring the SLAM output in the Z-axis ,which is consistent with the height of the robot, in two different points (P_{1_Z}, P_{2_Z}). Thanks to the robot on-board sonar sensor, the actual altitude is known, and therefore, the scale factor is calculated using a linear fitting:

$$P_{1_Z} = (Z_{1_{sonar}}, Z_{1_{SLAM}}) \quad , \quad P_{2_Z} = (Z_{2_{sonar}}, Z_{2_{SLAM}}) \tag{1}$$

$$\lambda = \frac{Z_{2_{sonar}} - Z_{1_{sonar}}}{Z_{2_{SLAM}} - Z_{1_{SLAM}}} \tag{2}$$

Hence, the modified robot position in the Z-axis is equal to:

$$Z = \lambda(Z_{SLAM} - Z_{1_{SLAM}}) + Z_{1_{sonar}} \tag{3}$$

Other axes may be computed similarly.

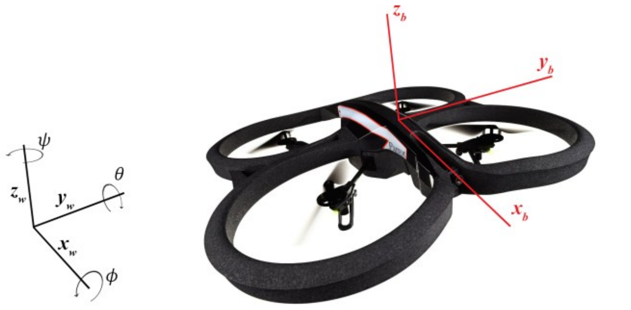

Fig. 1. AR. Drone (body) coordinate system and world coordinate system.

C. Pose Estimation

In this section, we introduce the state space of KF, including the observation and prediction model. The state space consists of the 3D positions and the one degree of freedom attitude angle. Therefore, state variables are set as follows:

$$X_k = (x_k, y_k, z_k, \psi_k)^T \tag{4}$$

where (x_k, y_k, z_k) denotes the 3D position of the robot in meter that represented in world coordinate and ψ_k is the rotation around Z-axis (yaw) in degrees. The prediction model is given as:

$$X_{k+1} = A\,X_k + B\,U_k + w_k \tag{5}$$

The aim of KF is to compensate and fuse the 3D position of robot in world coordinate and yaw angle with the measured navigation data. Therefore, the measurement model has the form of:

$$Y_k = (\hat{x}_k, \hat{y}_k, \hat{z}_k, \hat{\psi}_k)^T \tag{6}$$

$$Y_k = C\,X_k + D\,U_k + v_k \tag{7}$$

$$C = I_{4\times 4} \quad , \quad D = 0 \tag{8}$$

where w_k, v_k are the process and measurement noise, respectively. To find the system matrices, consider the kinematic equations that lead to:

$$x_{k+1} = 0.5\, a^w{}_{x_k} \Delta t^2 + v^w{}_{x_k} \Delta t + x_k \tag{9}$$

$$y_{k+1} = 0.5\, a^w{}_{y_k} \Delta t^2 + v^w{}_{y_k} \Delta t + y_k \tag{10}$$

$$z_{k+1} = 0.5\, a^w{}_{z_k} \Delta t^2 + v^w{}_{z_k} \Delta t + z_k \tag{11}$$

$$\psi_{k+1} = \omega_{\psi_k} \Delta t + \psi_k \tag{12}$$

where $(a^w{}_x, a^w{}_y, a^w{}_z)$ and $(v^w{}_x, v^w{}_y, v^w{}_z)$ are acceleration $\left(\frac{m}{s^2}\right)$ and velocity $\left(\frac{m}{s}\right)$ of robot in the world coordinate, respectively, and ω_ψ is the yaw angular velocity $\left(\frac{deg}{s}\right)$ and Δt is the system time step in seconds (s). If we consider U as follows:

$$U_k = \left(v^w{}_{x_k}, v^w{}_{y_k}, v^w{}_{z_k}, a^w{}_{x_k}, a^w{}_{y_k}, a^w{}_{z_k}, \omega_{\psi_k}\right)^T \tag{13}$$

Consequently, according to (9) to (12) the system matrices are given as follows:

$$A = I_{4\times 4} \tag{14}$$

and

$$B = \begin{bmatrix} \Delta t & 0 & 0 & 0.5\,\Delta t^2 & 0 & 0 & 0 \\ 0 & \Delta t & 0 & 0 & 0.5\,\Delta t^2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 & 0 & 0.5\,\Delta t^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \Delta t \end{bmatrix} \tag{15}$$

The horizontal velocity $(v^b{}_x, v^b{}_y)$ is measured by the robot which is in the robot frame. Therefore, there is a need to compute them in the word frame. Use rotation matrix for transformation:

$$(v^w{}_x, v^w{}_y) = \begin{bmatrix} cos(\psi) & sin(\psi) \\ -sin(\psi) & cos(\psi) \end{bmatrix} \cdot (v^b{}_x, v^b{}_y) \tag{16}$$

Furthermore, $v^w{}_z$ is equal to the derivative of the robot height:

$$v^w{}_z = \frac{h_k - h_{k-1}}{\Delta t} \tag{17}$$

On the contrary, robot cannot directly compute the acceleration in three dimensions and sum of both dynamic and static acceleration is accessible. While, we need just dynamic acceleration which can be computed as follows:

$$\vec{A}^b = \left(a^b{}_x, a^b{}_y, a^b{}_z\right)^T \tag{18}$$

$$\vec{A}^b{}_{measured} = \vec{A}^b{}_{dynamic} + \vec{A}^b{}_{static} \tag{19}$$

According to [6], $\vec{A}^b{}_{static}$ is equal to:

$$\begin{bmatrix} a^b{}_x \\ a^b{}_y \\ a^b{}_z \end{bmatrix} = \begin{bmatrix} sin(\phi)\, cos(\theta) \\ -sin(\theta) \\ \sqrt{g^2 - \left[(a^b{}_x)^2 + (a^b{}_y)^2\right]} \end{bmatrix} \tag{20}$$

where, ϕ is the rotation around X-axis (Roll), θ is the rotation around Y-axis (Pitch) and g is gravity of earth $(9.8\ \frac{m}{s^2})$; therefore,

$$\vec{A}^b{}_{dynamic} = \vec{A}^b{}_{measured} - \vec{A}^b{}_{static} \tag{21}$$

Finally, acceleration in the world coordinate may be computed similarly:

$$\vec{A}^w{}_{dynamic} = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \vec{A}^b{}_{dynamic} \tag{22}$$

Moreover, ω_ψ is directly available in the robot measurements.

D. PID Controller

After prediction of the robot 3D position, we can control the position of robot. For this, ordinary PID controller is utilized for each axis and robot Yaw angle. All the PID coefficients are determined and tuned during some experiments. The general structure of controller is shown in the Fig. 2 using block diagram. In this structure, the estimated 3D robot position and robot Yaw angle calculated in the last section are used as feedback.

III. Map Enrichment

As mentioned before, the environment map which is reconstructed by ORB SLAM is sparse, and hence, it is not suitable for autonomous flight and obstacle avoidance. To realize this aim, there is a need to make the output map more precise and dense. For this reason, we proposed an algorithm which is mainly based on plane fitting, clustering and classifying methods. In what follows, we will elaborate the proposed approach.

First, a part of sparse map that is around the robot (e.g. points that are inside of a sphere with a finite radius) is selected and split up into cubes with specific side length. Then, the mean of points of all boxes is computed. Any box in which the number of containing points is lower than the mean point is ignored. After that, for each remaining box a plane is fitted.

Thanks to K-Means [7] clustering algorithm, the determined planes are categorized into five groups, namely top, right, left, front, and rear directions, with respect to their normal vectors. Then, for each cluster a mean direction vector is computed. In every cluster, the angle deviation between each normal plane and mean vector is computed and if the deviation is more than 15 degrees, the corresponding plane will consider as outlier. By utilizing the K-nearest neighbors (K-NN) classifier method, [8], with K = 3 the outliers are classified to the right cluster. As

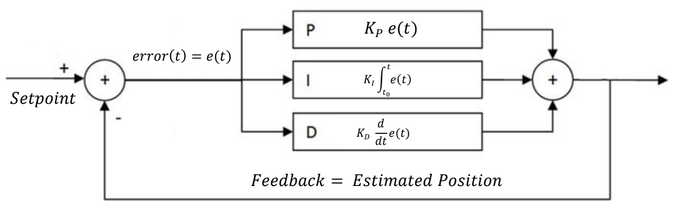

Fig. 2. General structure of PID controller for 3D position and Yaw angle control

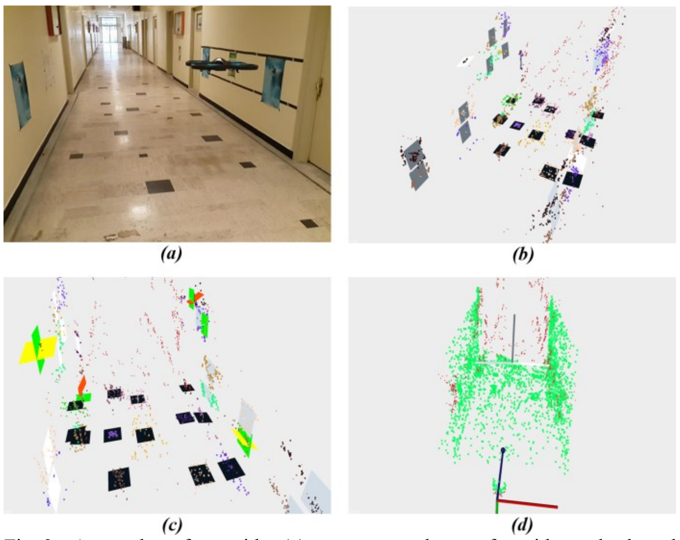

Fig. 3. A snapshot of a corridor (a); reconstructed map of corridor and selected around point cloud of robot, the selected point cloud divided into several boxes and each box exhibited in different colors, the fitted planes are distinguished in different clusters and displayed in different colors (b); outlier planes determined and highlighted, the modified planes are in green color (c); according to corrected normal vector for each cluster, the existed points are lined up and some extra points added (d).

the outliers are corrected, the mean vectors for each cluster are altered.

Finally, to enrich the map, we line up points in each cluster with respect to the average normal vector. Moreover, some extra random points are added according to the mean normal vector in every separate box, in order to enrich the environment map. In Fig. 3 it is shown the map enrichment algorithm for a corridor (Fig. 3-a), as you can see in Fig.3-b the reconstructed map using only ORB SLAM in really sparse and not suitable for autonomous flight; therefore it should be made more dense.

First, the point cloud area of robot is selected and is divided into several boxes in order to fit a plane to each box. Then, these boxes are distinguished in different clusters (Fig. 3-b). In the next step the outlier planes are determined and corrected according to the normal vector of each cluster (Fig. 3-c). Finally, the existed points are lined up and some extra points are added to make it more dense (green points in Fig. 3-d).

IV. Path Planning

By performing the previous steps, the data are ready to be used for autonomous flight and obstacle avoidance. The autonomous flight consists of two steps. The first step is to determine the next appropriate motion set point and the second step is generating a collision-free trajectory to reach to this set point. To aim this, we used some path planning methods such as potential field and RRT algorithms. In what follows, we will explain the proposed approach.

A. Set Point Determination

The first step of autonomous flight is to determine an appropriate set point that is far enough from the obstacles. To this end, the next set point is specified using potential field method [9]. Usually, in potential field method robot is considered as a point in a potential field that combines attraction to the goal and repulsion from obstacles. However, in our approach we do not consider goal point and the potential field only consists of repulsive field which made up of two different sources. The field is a combination of the repulsion from obstacles and the current position of robot. The repulsive field of both sources is computed by the following equation:

$$U_{rep}(q) = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 , & \rho(q) \leq \rho_0 \\ 0 \quad , & \rho(q) \geq \rho_0 \end{cases} \tag{23}$$

where $\rho(q)$ is the Euclidean distance between the query point q_{query} and new set point q, k_{rep} is a constant coefficient which determines the intensity of field. Hence, the repulsive force acting on robot is proportional to the gradient of repulsive field:

$$F_{rep}(q) = -\nabla U_{rep}(q) \tag{24}$$

$$F_{rep}(q) = \begin{cases} k_{rep} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(q)} \frac{q - q_{query}}{\rho(q)} , & \rho(q) \leq \rho_0 \\ 0 \quad , & \rho(q) \leq \rho_0 \end{cases} \tag{25}$$

Therefore, the affecting force is equal to:

$$F(q) = F_{rep|obst} + F_{rep|\, current\ robot\ pos} \tag{26}$$

where each of two forces has different constant coefficients (k_{rep}, ρ_0). Thus, to determine the new set point for the robot we consider the initial position for new set point with the same height of the robot and a bit further than the current robot horizontal position. Then, the potential field algorithm is executed and at each iteration $F(q)$ is applied to the new set point to converge to a steady state. When $F(q)$ is lower that ε then the algorithm is terminated and the computed point is considered as the new set point.

B. Path Generation

To reach to new set point we must determine a collision-free path. To this end, there is a need for a real time path planning algorithm, and in this research we used RRT path planning method [10]. RRT is an algorithm designed to efficiently search non-convex, high-dimensional spaces by randomly building a space-filling tree. The tree is incrementally constructed from random samples generated in admissible search space and grows towards large unsearched areas of the environment. Tree starts growing from the current robot position and tries to find a feasible path toward the new set point by using random samples from the search space. As each sample is drawn, a connection is attempted between it and the nearest state in the tree. If the connection is feasible, and the new connection pass through admissible space and keeps enough distance from the obstacles, the addition of the new state is inserted to the tree. This process continues until a collision-free path can be found between the goal and the start point.

After a path is found, path smoothing may be applied to make a smoother and more direct route to the goal. This may be done iteratively by sampling points between nodes on the overall path and then checking if two points could be connected to reduce the overall path length. In Fig. 4 it is shown the RRT path generation for a 2D environment. The dashed-dotted lines are the RRT Tree which consists of some branches. As you can see, the tree has randomly grown in the admissible space to find a feasible path between start point and goal point. Upon finding a feasible path, the tree growth terminates (solid lines). Since the generated path is not smooth enough for navigation, consequently a smooth path is fitted to it which is shown in dashed line.

Another noteworthy aspect of the proposed path planning is to consider an admissible area in both set point determination and path generation. Due to the sparse reconstructed map, sometimes the floor and the ceiling cannot be recognized, and therefore, to avoid wrong path generation, the admissible area shall be determined. Thanks to the proposed algorithm for map enrichment, we exactly know the bounds of reconstructed map and we consider these bounds as admissible area. In Fig. 5, new set point and a generated collision-free path between robot (red wired sphere) and set point (green solid sphere) are illustrated. As it is seen in this figure in order to prepare the point cloud for collision-free and successful autonomous flight, first we line up and enrich the reconstructed point cloud (green points). Then by means of RRT path planning method a feasible path generated which is rough and not appropriate for navigation (red solid lines). Therefore, the smooth path is generated which is distant enough from the obstacles (violet dashed lines).

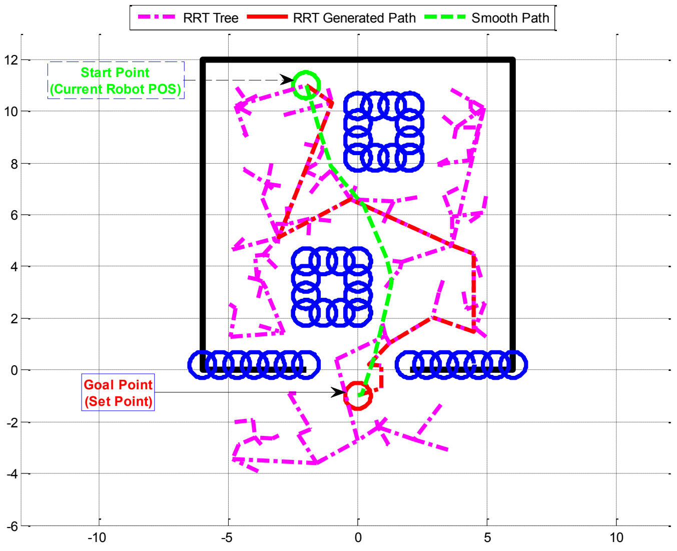

Fig. 4. RRT path generation algorithm, for the ease of understanding the 2D path generation is shown. The dashed-dotted lines are the RRT tree, the feasible generated path is in solid lines and the dashed lines denote the final smooth path. Blue circles and the black lines are obstacles.

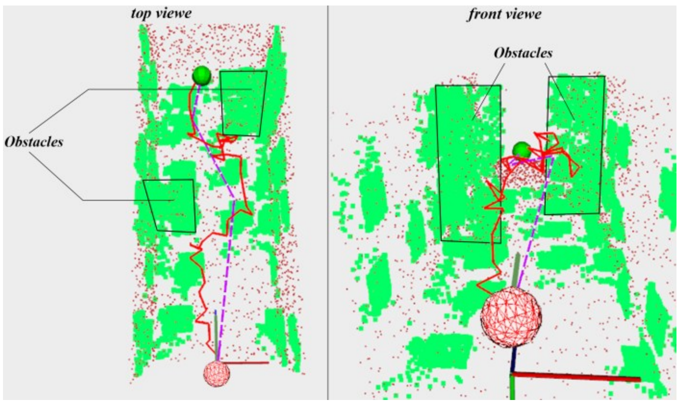

Fig. 5. Generating 3D collision-free path between current robot point (red wired sphere) and determined new set point (green solid sphere). The red solid lines are RRT feasible generated path and violet dashed lines are final smooth path.

V. Experiments and Results

To verify the effectiveness of the proposed approach a series of experiments is conducted in different environments. For all tests, we used a low cost AR. Drone quadrotor which is released in 2010 and has been used in several researches [11, 12, 13, 14]. This Drone is equipped with a navigation system operating in 200 Hz that includes an inertial measurement unit (IMU) and vision based horizontal velocity system estimator. Its IMU consists of a 3-axis gyroscopes and accelerometer. The height of robot is measured by an ultrasonic sensor with a refresh rate of 25 Hz. Moreover, there is a 720p frontal camera sensor with 93° lens that is recording up to 30 frames per second. All navigation data and video stream are sent to the ground station laptop using Wi-Fi connection link to be processed real-time. All programs are designed in Robot Operating System (ROS-indigo) and will be published as open source software in near future.

A. POS Estimation Evaluation

In current section, we evaluate the performance of the proposed Kalman filter. After scale factor calculation, Kalman filter provides directly the robot 3D position and the yaw angle. To verify the KF output data, the estimated position of robot in Z-axis and yaw angle with the actual values that obtained from the accurate external sensors are compared. Fig. 6, shows the results; in which the solid lines denote the actual measurements, the dashed-dotted lines are the estimated values and the SLAM outputs are drawn with dotted lines. In the both graphs it can be seen that the KF estimated values are close to the actual measurements. Moreover, the delay is well compensated .In samples 800 to 900 the SLAM is lost for a moment and the outputs are not reliable while the KF is capable to modify and estimate the correct values. In short, the KF outputs are more reliable, precise and in general far better than the raw SLAM outputs.

Moreover, we evaluated the position control system by defining a $2m \times 2m$ square path as reference trajectory that the results of this experiment are shown in Fig. 7. In this figure the reference path is shown in the dashed-dotted red lines and the blue solid lines denote the robot position. As you can see, robot can suitably track the reference trajectory, although the aim of this research is not path following.

B. Complete System Performance

To appraise the reliability of our approach, we arranged increasingly challenging environments which the robot had to fly autonomously. The aim of these experiments is to evaluate the success rate of autonomous flight and avoiding the probable obstacles. Fig. 8 shows the three different environments which have been tested in our approach. In Fig. 8-a the robot fly only in the blocked corridor, while Fig. 8-b illustrates a blocked corridor with an extra obstacle. In Fig. 8-c a more challenging environment is set up in which a blocked corridor with two extra obstacles are installed while there are no additional features added to the environment. Columns in Fig. 8 represent, respectively, a snapshot of the environment, the ORB-SLAM output image which consists of the robot view and extracted and tracked corner features of environment and the last picture is a view of the developed software that exhibits all steps of our proposed algorithm (ORB-SLAM reconstructed map, enriched point cloud and collision-free generated path). In all experiments robot starts the mission from the beginning of corridor and moves along it. The mission will be finished when robot faces an obstacle that cannot pass through, while the robot lands at the end of mission. It is noteworthy that all experiments have been conducted successfully in real-time without any collisions; this verifies the effectiveness and reliability of the proposed approach. The videos of experiments with more details are available online at this link.

VI. Conclusion

In this paper, a visual navigation for autonomous flight and obstacle avoidance of a low-cost quadrotor is presented. The proposed system enables the robot to flies autonomously in unknown environment and avoids colliding obstacles. The proposed algorithm generally consists of two parts. Firstly, we obtain the 3D position of robot. For this, the 3D position of robot is estimated using Kalman Filter which fuses the monocular ORB-SLAM outputs and navigation data measured by on-board sensors of drone. Regarding the autonomous flight and obstacle avoidance, robot needs to have a perception of its environment. To fulfill this aim, we use the surrounding map of robot which is reconstructed by monocular ORB-SLAM. But, this map is sparse and not appropriate for autonomous applications. Therefore, we represented an algorithm that lines up and enriches the reconstructed map. In the next step we determine the motion next set point and generate a collision-free path between specified set point and current robot position. For this, a dynamic trajectory generation algorithm is proposed to fly and avoid the probable obstacles autonomously in an unknown but structured environment by utilizing some path planning methods such as potential field and RRT. The algorithm has been evaluated in real experiments and the flight variables are compared with some external precise sensors. In the experiments, it is illustrated that robot can perform reliable and robust autonomous flight in different scenarios while avoiding obstacles. Moreover, the proposed system can be easily applied to other platforms, which is being extended and implemented in our future plans.

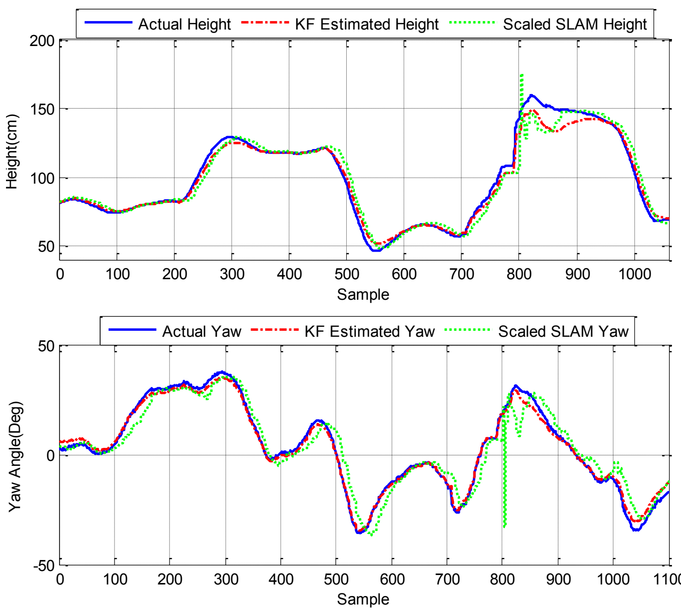

Fig. 6. Evaluation of estimated KF states, top graph is comparison between estimated and actual height, in bottom graph the actual and estimated Yaw angle compared.

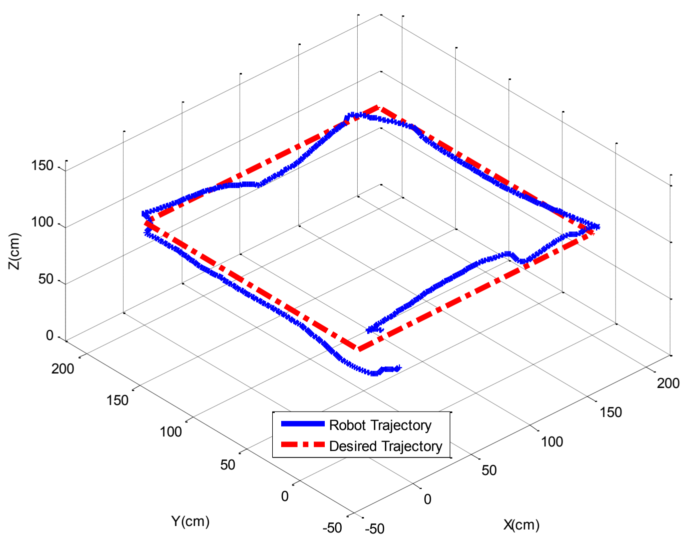

Fig. 7. Square path following of a quadrotor in three-dimensions. The solid lines are the 3D position of robot and the reference path is shown in dashed-dotted lines.

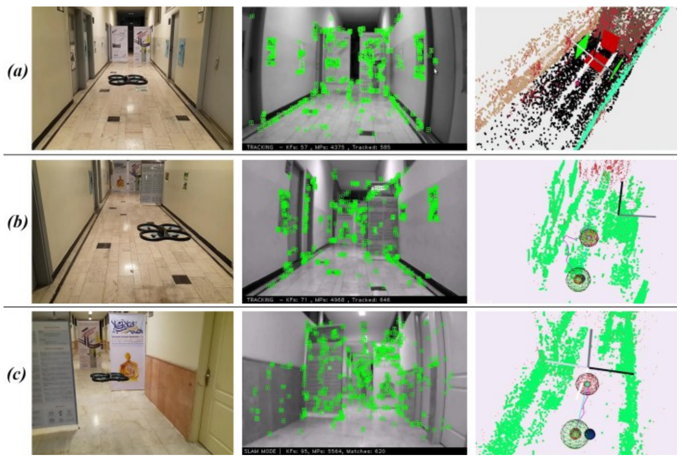

Fig. 8. Verification of autonomous flight and obstacle avoidance approach on three different environments. From top to bottom: blocked corridor, blocked corridor with an extra obstacle, blocked corridor with two extra obstacles and without any additional features.

Acknowledgement

We are sincerely thankful to the staff of ARAS Laboratory and KN2C robotic team for helping us in carrying out the experiments and collecting data.

References

[1] L. R. G. Carrillo, G. Flores Colunga, G. Sanahuja, and R. Lozano, “Quadrotorcraft switching control: An application for the task of path following,” Control Systems Technology, IEEE Transactions on, vol. 22, no. 4, pp. 1255–1267, 2014.

[2] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, “Autonomous navigation and exploration of a quadrotor helicopter in gpsdenied indoor environments,” in First Symposium on Indoor Flight, no. 2009, 2009.

[3] Huang, Rui, Ping Tan, and Ben M. Chen. "Monocular vision-based autonomous navigation system on a toy quadcopter in unknown environments." Unmanned Aircraft Systems (ICUAS), 2015 International Conference on. IEEE, 2015.

[4] Lange, Sven, et al. "Autonomous corridor flight of a UAV using a low-cost and light-weight RGB-D camera." Advances in Autonomous Mini Robots. Springer Berlin Heidelberg, 2012. 183-192.

[5] Mur-Artal, Raul, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system." Robotics, IEEE Transactions on 31.5 (2015): 1147-1163.

[6] Engel, Jakob, Jürgen Sturm, and Daniel Cremers. "Camera-based navigation of a low-cost quadrocopter." Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012.

[7] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.

[8] Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician 46 (3): 175–185.

[9] Hwang, Yong K., and Narendra Ahuja. "A potential field approach to path planning." Robotics and Automation, IEEE Transactions on 8.1 (1992): 23-32.

[10] LaValle, Steven M., and James J. Kuffner Jr. "Rapidly-exploring random trees: Progress and prospects." (2000).

[11] Bills, Cooper, Joyce Chen, and Ashutosh Saxena. "Autonomous MAV flight in indoor environments using single image perspective cues." Robotics and automation (ICRA), 2011 IEEE international conference on. IEEE, 2011.

[12] Krajník, Tomáš, et al. "AR-drone as a platform for robotic research and education." International Conference on Research and Education in Robotics. Springer Berlin Heidelberg, 2011.

[13] Ng, Wai Shan, and Ehud Sharlin. "Collocated interaction with flying robots." 2011 RO-MAN. IEEE, 2011.

[14] Alvarez, H., et al. "Collision avoidance for quadrotors with a monocular camera." Experimental Robotics. Springer International Publishing, 2016.