SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Autonomous Navigation of A Multirotor Aerial Robot in GNSS-denied Environments For Search and Rescue

HOU XiaoLei[1*], LI ZhuoYi[1] & PAN Quan[1]

[1]*School of Automation, Northwestern Polytechnical University, Xi'an* 710000*, China*

Appendix A Self-localization System

Technical details on the multiple sensor information fusion odometry calculation, as well as loop closure processing in self-localization are presented and analyzed comprehensively in this Appendix.

Appendix A.1 Fusion Odometry and Loop Closure

Odometry is the fundamental component of a self-localization system, where measurements of stereo camera, IMU and Lidar are fused using a local optimizer to facilitate a high-accuracy and robust fusion odometry. We follow OKVIS [1] to obtain visual-inertial odometry, which includes IMU pre-integration, visual feature matching and tracking, local non-linear optimization, and marginalization. The VIO calculation in OKVIS is formulated as a joint optimization of a cost function $J(\boldsymbol{x})$ consisting both reprojection errors $\boldsymbol{e}_r$ and temporal error from the IMU $\boldsymbol{e}_s$ as follows:

$$J(\boldsymbol{x}) = \sum_{k=1}^{K} \sum_{j \in \boldsymbol{z}_k} \boldsymbol{e}_r^{j,k\,T} \boldsymbol{W}_r^{j,k} \boldsymbol{e}_r^{j,k} + \sum_{k=1}^{K-1} \boldsymbol{e}_s^{k\,T} \boldsymbol{W}_s^{k} \boldsymbol{e}_s^{k}, \tag{A1}$$

where k and j represent the index of image frame and landmark, respectively. The set of landmarks observed in kth image frame is denoted by $\boldsymbol{z}_k$. $\boldsymbol{W}_r^{j,k}$ is the information matrix of the landmark measurements, and $\boldsymbol{W}_s^k$ is the information matrix of kth IMU error.

The state estimation of VIO is provided as a motion prediction for laser scan-to-map registration to calculate fusion odometry, since ICP algorithm can quickly converge with an accurate initial value. Denote optimized fusion odometry at time-step $k+1$ as $\boldsymbol{L}_{k+1}$ and the initial estimation of $\boldsymbol{L}_{k+1}$ as $\boldsymbol{L}_{k+1}^*$, we have:

$$\boldsymbol{L}_{k+1}^* = \boldsymbol{L}_k \boldsymbol{T}_t^L, \tag{A2}$$

where $\boldsymbol{L}_k$ is the optimized fusion odometry at time-step k, and $\boldsymbol{T}_t^L$ is motion prediction provided by VIO. In laser scan-to-map matching, the residual $\boldsymbol{r}_E$ of an edge feature point and residual $\boldsymbol{r}_U$ of a surface point in the current scan can be written as,

$$\begin{aligned} f_E(E_{(c,i)}^L, \mathbf{L}_{k+1}^*) &= \mathbf{r}_E, \\ f_U(U_{(c,i)}^L, \mathbf{L}_{k+1}^*) &= \mathbf{r}_U, \end{aligned} \tag{A3}$$

where $E_{(c,i)}^L$ and $U_{(c,i)}^L$ are the 3D position of the ith dewarped feature point in the body coordinate frame. Levenberg-Marquardt optimizer is used to solve this non-linear optimization problem for calculating $\mathbf{L}_{k+1}$, formed by stacking the cost functions for all feature points.

It is worth noting that the advantage of fusion odometry is that state estimation can still be calculated when one kind of sensor fails. The diagram of the fusion odometry data processing pipeline is shown in Fig. A1. When the number of tracked visual features is lower than a threshold, the laser scan registration will be initialized based on IMU motion prediction instead of VIO shown as the blue line. Similarly, if there are not enough geometric features in the environment, the VIO outputs will be passed directly to the back-end shown as the green line.

Loop closure and proximity detection are critical to any self-localization system as errors accumulate over long-term operations. Once a loop closure or proximity detection is triggered, a 6-DOF pose graph based optimization is used to correct global pose. DBoW2 [2] and ORB [3] features are used for visual loop closure detection. Tf-IDF [4] method is

* Corresponding author (email: hou.xiaolei@nwpu.edu.cn)

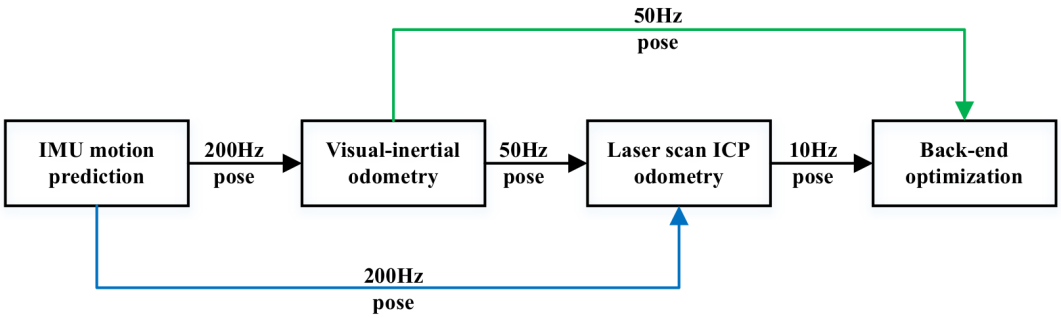


Figure A1 Diagram of fusion odometry data processing pipeline.

used to update a Bayes filter estimating the loop closure probability. However, visual loop closure detection is limited by the camera's FOV, we thus introduce laser-based proximity detection [5] to assist visual loop closure detection. To reduce redundant constraints and improve the efficiency of loop detection, we set the priority of the visual DBoW2 method higher than laser proximity detection, that is, if the visual loop closure is detected between the current frame and the keyframe in the database, laser proximity detection will no longer be performed.

During the flight, the number of keyframes in the database could increase rapidly, which would have a significant impact on real-time performance and accuracy of loop closure and proximity detection. RTAB-Map's memory management approach [6] runs on top of graph management modules to limit the number of keyframes to be detected in the database, which ensures long-term online SLAM can be achieved in large-scale environments. Although DBoW2 uses feature descriptions to distinguish similar locations as much as possible, it still has a small probability of false detection when the current location is too similar to the revisited location. To avoid the impact of this situation on the global pose map, a threshold is introduced to the validation of loop closure or proximity detection. If a link's transformation in the graph after non-linear optimization has changed more than this threshold, all links added by this keyframe are rejected, keeping the optimized graph as the previous graph. When there is a reasonable loop closure or proximity detection triggered, a link between the current frame and the corresponding keyframe is added to the pose graph, then iSAM2 implementation in GTSAM library [7] is used to minimize errors in the global pose graph, so as to decrease the accumulated drift of fusion odometry. The diagram of the proposed self-localization system is shown in Fig. A2.

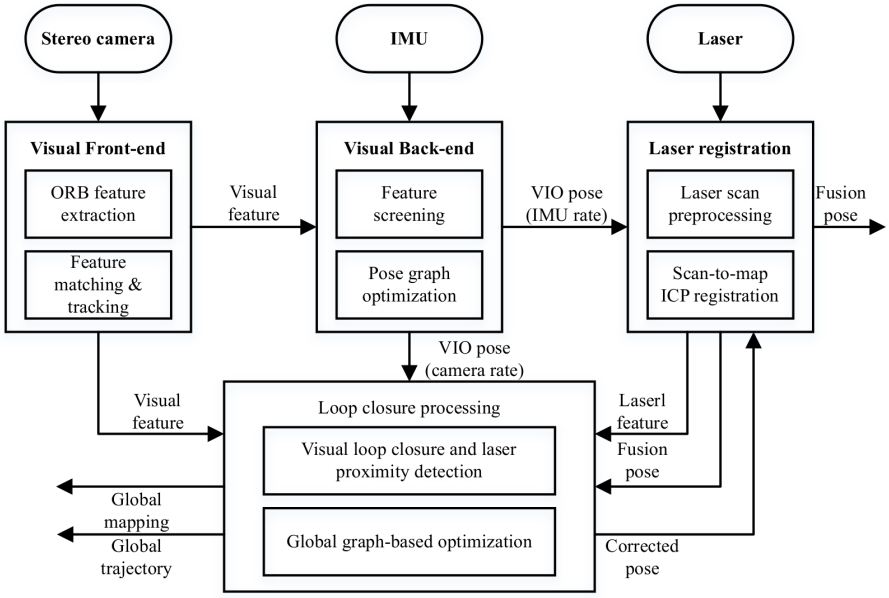


Figure A2 Diagram of multiple sensor information fusion self-localization system.

Appendix B Path planner

In Appendix B, the details of the proposed autonomous navigation algorithm are illustrated.

Provided the limited on-board computational resources, it is challenging for a path planner to guarantee both feasible solutions and real-time performance in complex and unknown environments. Therefore, it is almost impossible to plan a

globally optimal path in the unknown large-scale environments, while only a global path could be found by incrementally searching for a locally optimal path based on the local navigation histogram. The proposed path planner is extended based on VFH* [8] and 3DVFH+ [9], which removes the global navigation map construction process and only use local navigation histogram to minimize the planning time and impact of absolute self-localization accuracy. Moreover, A* heuristic search is used to search for a locally optimal path based on a local navigation histogram. By iterating the above path planning process, a feasible globally sub-optimal path could be generated in real-time.

Most of the following definitions and mathematical terms are related to the VFH+ [9] and A* [10]:

- ***Map*** Ω_{nav}: A local navigation histogram includes the azimuth, elevation, and distance of the obstacles with respect to the UAV's current position p_{cur}.

- ***Free space*** S_{free}: A set of free grids, i.e. the grids that satisfy '$\Omega_{nav}(row, col)_1 = 1$ and $\Omega_{nav}(row, col)_2 \geqslant d_{safe}$' or '$\Omega_{nav}(row, col)_1 = 0$' are defined as *free*, where d_{safe} is the radius of UAV's safety envelope.

- ***Occupied space*** $S_{occupied}$: A set of grids occupied by obstacles, i.e. the grids that satisfy $\Omega_{nav}(row, col)_1 = 1$ and $\Omega_{nav}(row, col)_2 \leqslant d_{safe}$.

- ***Unknown space*** $S_{unknown}$: A set of grids outside the field of Ω_{nav}, that is $S_{unknown} = S \backslash (S_{free} \cup S_{occupied})$.

- ***Node*** n: A node n includes the 3D position coordinate, the parent node, and the objective function value. In A* search algorithm, these nodes are connected to one another (its parent node) by the edges.

- ***Edge*** e: An edge is the branch/link connecting two nodes (a node and its parent node) with a fixed step-size α.

- ***Tree*** T: Both nodes and edges form a search tree T, $T = (N, E)$, where $N = \{n_1, \cdots, n_i\}$, $E = \{e_1, \cdots, e_j\}$.

- ***OpenList***: A set stores the nodes of T that are not evaluated.

- ***CloseList***: A set stores the nodes of T that are evaluated.

After the local navigation histogram being updated, the VFH+ planner considers all unoccupied grids as candidate nodes. Then, the VFH+ planner evaluates all candidate nodes according to the cost function C and selects the nodes with lower cost in order. With respect to p_{cur}, two aspects $c^{p_g}_{p_{can}}$ and $c^{p_{sel}}_{p_{can}}$ are mainly considered including how close the candidate node p_{can} is to the goal position p_g and motion smoothness between the candidate node and the previously selected node p_{sel}. Therefore, the cost function C is determined as the weighted sum of the goal and smoothness costs as follows:

$$C = w_g \cdot c^{p_g}_{p_{can}} + w_s \cdot c^{p_{sel}}_{p_{can}}, \tag{B1}$$

where w_g and w_s are the weight factors corresponding to $c^{p_g}_{p_{can}}$ and $c^{p_{sel}}_{p_{can}}$, respectively. Furthermore, $c^{p_g}_{p_{can}}$ and $c^{p_{sel}}_{p_{can}}$ can be calculated as follows:

$$\begin{aligned} \Delta_\phi(p_g, p_{can}) &= \phi^{p_g}_{p_{cur}} - \phi^{p_{can}}_{p_{cur}}, \\ \Delta_\theta(p_g, p_{can}) &= \theta^{p_g}_{p_{cur}} - \theta^{p_{can}}_{p_{cur}}, \\ c^{p_g}_{p_{can}} &= \Delta_\phi(p_g, p_{can}) + \Delta_\theta(p_g, p_{can}), \end{aligned} \tag{B2}$$

$$\begin{aligned} \Delta_\phi(p_{sel}, p_{can}) &= \phi^{p_{sel}}_{p_{cur}} - \phi^{p_{can}}_{p_{cur}}, \\ \Delta_\theta(p_{sel}, p_{can}) &= \theta^{p_{sel}}_{p_{cur}} - \theta^{p_{can}}_{p_{cur}}, \\ c^{p_{sel}}_{p_{can}} &= \Delta_\phi(p_{sel}, p_{can}) + \Delta_\theta(p_{sel}, p_{can}), \end{aligned} \tag{B3}$$

A* is a heuristic path planning algorithm based on graph searching, which is widely used in UAV's path planning missions. By evaluating the reachable nodes in the *OpenList* using an objective function, heuristic search extends the path towards the node with the highest score, and an optimal path can be found by iteratively repeating the node evaluation and heuristic search. A* objective function $F(n)$ usually consists of a cost function $C(p_{cur}, n)$ and an heuristic function $H(n, p_g)$ as follows:

$$F(n) = C(p_{cur}, n) + H(n, p_g), \tag{B4}$$

where n is the node being evaluated, $C(p_{cur}, n)$ is calculated in the same way as the cost function in 3DVFH+, and $H(n, p_g)$ is obtained by calculating the Euclidean distance between n and p_g as follows:

$$H(n, p_g) = \|n - p_g\|_2. \tag{B5}$$

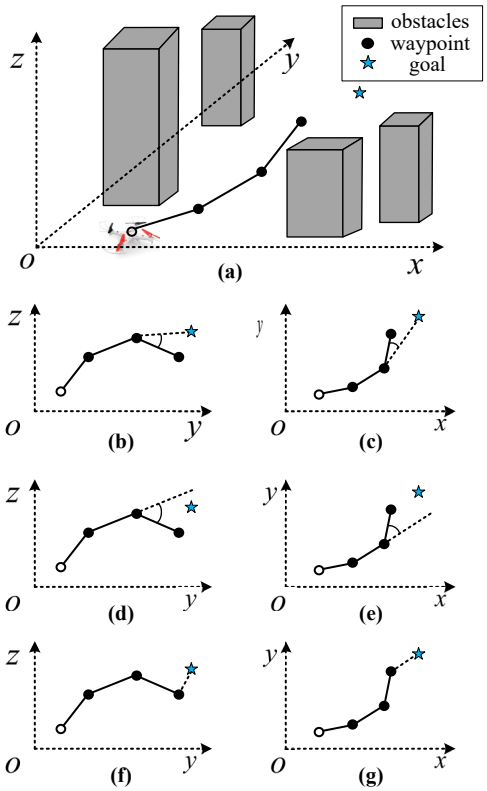


Figure B1 The schematic diagram of local path planner combining VFH+ and A*. From (b) to (e), there are the difference of elevation and azimuth in goal cost, the difference of elevation and azimuth in smoothness cost, respectively. (f) and (g) show the calculation of heuristic function value. Here, the side and top views of a 3D trajectory are projected on the frame $o-yz$ and $o-xy$, while the obstacles are omitting.

The construction of A* search tree is shown in Figure B1. Denote the current evaluating node by γ_i, the local planner evaluates all valid candidate nodes relative to γ_i based on objective function $F(n)$. A number of the top-ranked low-cost nodes $\{n_1, \cdots, n_j\}$ are pushed into *OpenList* and γ_i is pushed into *CloseList* for search tree updating. Then, the optimal node n^* with the smallest objective function value $F(n^*)$ is selected as γ_{i+1} for next search step. Path searching starts at γ_{i+1} iteratively until the number of the node in *CloseList* reaches threshold η or the target position is contained by the evaluated path $p_g \in \rho_k$. Finally, the locally optimal path ρ_k can be obtained by back-tracing from γ_i to γ_1. The pseudo code of the local path planner is described as Algorithm B1.

Appendix C Experimentation

The details of Softwere-In-The-Loop (SITL) experiments in a virtual environment and experiments on a robotic platform are illustrated in this Appendix Appendix C.

Appendix C.1 Experiments in a virtual environment

In the SITL experiments, the px4 firmware with proper parameter setting for a quadrotor aerial robot is used to achieve close dynamics approximation. A virtual environment with a maze of vertical brick walls and a grove of trees (shown in Fig. C1) is established using Gazebo[1)] and Robot Operating System (ROS).

Appendix C.1.1 *Experimental Scenarios:*

The UAV is initially placed at the starting point outside the woods, and then commanded to navigate through the woods and maze by approaching a series of predefined waypoints in UAV's local coordinate frame, and eventually comes back to the starting position. A successful flight is expected to have no collisions with the environment. The entire flight is fully automated using the proposed approach, and flight data including flight time, path length and number of collisions is recorded for analysis. In search for the optimal combination of parameter settings, different combinations of heuristics and memory settings (as in Table C1) were tested 20 times in experiments, where heuristics and memory are denoted as H and M, L and S denote long and short respectively. In addition, the original 3DVFH+ path planner is also implemented as

1) http://gazebosim.org/

Algorithm B1 Proposed Path planner

Input: UAV's current position p_{cur}, goal position p_g, and local navigation histogram Ω_{nav} at time-step k. Maximum number η of nodes in *ClostList* , maximum number m of the child nodes from a node, minimum distance ξ between two child nodes.

Output: Path $\rho_k = \{\gamma_1, \gamma_2, \cdots, \gamma_i\}$.

Initialization: $i = 1$, $\gamma_i = p_{cur}$, $\rho_k = \varnothing$, $OpenList = \varnothing$, $CloseList = \varnothing$.

while $Num(ClostList) \leqslant \eta$ and $p_g \notin \rho_k$ **do**

Sample j ($j \leqslant m$) child nodes $\{n_1, \cdots, n_j\}$ of γ_i based on Ω_{nav}, which satisfies $\forall\ k, l \in [1, j], n_k \in S_{free}, \|n_k - n_l\|_2 \geqslant \xi$.

Calculate $F(n_i) = C(\gamma_1, n_i) + H(n_i, p_g), i \in [1, j]$ according to $C = w_g c_{p_{can}}^{p_g} + w_s c_{p_{can}}^{p_{sel}}$, and $H(n, p_g) = \|n - p_g\|_2$.

$OpenList = OpenList \cup \{n_1, \cdots, n_j\}$.

$ClostList = ClostList \cup \{\gamma_i\}$.

Find node n^* by sorting *OpenList* that $\forall\ n_i \in OpenList$, $F(n^*) \leqslant F(n_i)$.

$i \leftarrow i + 1$, $\gamma_i \leftarrow n^*$.

end while

Obtain ρ_k by back-tracing from γ_{i-1} to γ_1 in *CloseList*.

the baseline algorithm for comparison, which has no heuristics or memory, thus denoted as NHNM. The basic experiment setting and waypoint coordinates are listed in Table C2.

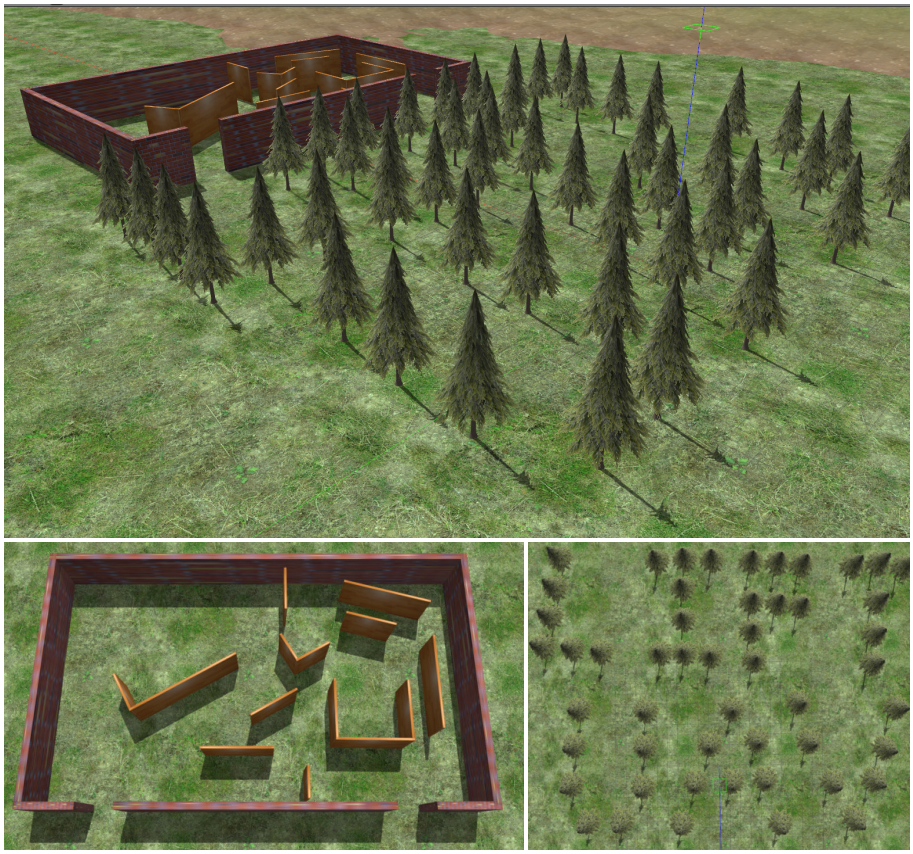


Figure C1 The simulated flight environment for the SITL experiments.

Table C1 Contrast experiment setting.

Type	VFH+	Memory(s)	Heuristic(m)
NHNM	✓	\	\
LHNM	✓	\	8
SHNM	✓	\	4
NHLM	✓	15	\
LHLM	✓	15	8
SHLM	✓	15	4
NHSM	✓	5	\
LHSM	✓	5	8
SHSM	✓	5	4

Table C2 Setting of planner parameter and waypoint sequence.

Parameter	value
Sensor range	0.2m-10.0m
Pitch cost	30.0
Yaw cost	3.0
Heuristic weight	30.0
Safety envelope radius	0.6m
Max linear speed	2m/s
Max yaw speed	30deg/s
Waypoint[1]	(30, 10, 1.5)
Waypoint[2]	(40, 7, 1.5)
Waypoint[3]	(47 ,4.5 ,1.5)
Waypoint[4]	(38.7 ,-7.5, 1.5)
Waypoint[5]	(0, 0, 1.5)

Appendix C.1.2 *Results*

The experimental results are presented in Fig. C2 and Tab. C3. The successful flight paths with different parameter settings are illustrated using different colors and the waypoints are depicted as circles in Fig. C2. In Tab. C3, the flight time, path length and success rate are provided.

From the results, we have following intuitive observations:

- the proposed path planner with memory can significantly improve the flight performance;
- the longer the memory is, the shorter the time and distance the UAV will take, and the higher the success rate will be;
- with the same memory setting, flights with longer heuristics seem better than those with shorter heuristics;

Based on the above observations, two-sample T-Test with heterogenous variations is applied to pairwise performance comparisons of flight data to achieve quantitative analysis on the system performance with different parameter settings. The equally good performance hypothesis is assumed for each pairwise T-Test.

T-Test between `NHNM` and `NHLM` was carried out first and the result is shown in Tab. C4. Flight time and length of the system with `NHLM` exhibit the significance α of 0.000001 and 0.006044 to reject the hypothesis. Hence, we claim that, the system with long-term memory will lead to better performance in flight time, flight length and success rate.

Provided the outcomes above, we conducted T-Tests among `NHLM`, `SHLM` and `LHLM`, and the results are shown in Tab. C5. The system with `LHLM` completely outperforms that with `NHLM`, and exhibits the significance level of 0.004165 and 0.002458 to reject the hypothesis of equally good performance. Therefore, we claim that long-distance heuristics could result in better performance.

At last, considering the limited computational resource onboard the UAV, we would like to evaluate the necessity of long-term memory. A T-Test between `LHSM` and `LHLM` was performed, and the results are provided in Tab. C6. From the compelling T-Test outcomes, it is clear that although it is computationally expensive to have long-term memory, the performance of the system with `LHLM` is superior in all three different flight data.

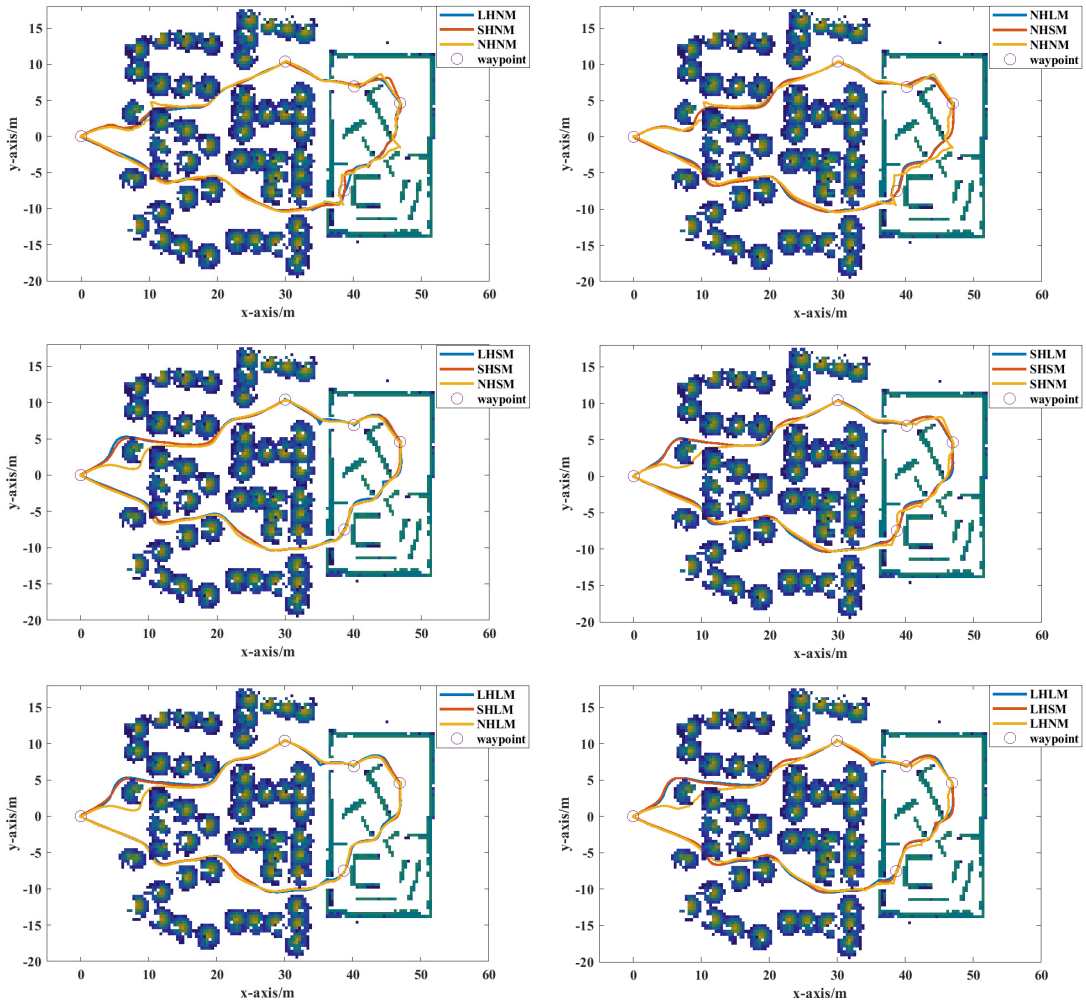


Figure C2 Successful flight demonstration of the local planer with different parameter setting as shown in Table. C1. To facilitate the representation of obstacle information in the environment, we merge point cloud into a global map based on Octomap. The global map is only used for flight environment visualization, not for autonomous navigation of UAV.

Table C3 Experimental results of UAV flight based on different memory duration and heuristic distance conditions.

Type	Time(s)	Length(m)	Success rate(%)
NHNM	144.5	159.5	20
SHNM	141.7	158.4	20
LHNM	140.3	156.7	25
NHSM	93.6	142.4	80
SHSM	89.9	138.8	85
LHSM	86.2	130.0	85
NHLM	82.1	138.5	90
SHLM	79.9	133.7	95
LHLM	75.3	128.2	95

Table C4 T-Test results of UAV flight data with NHNM and NHLM.

Type	NHNM	NHLM	α
Time(s)	144.5	82.1	0.000001
Length(m)	138.5	133.7	0.006044
Success rate(%)	20	90	N/A

Table C5 Pairwise T-Test results of UAV flight data with NHLM, SHLM and LHLM.

Type	NHLM	SHLM	α	NHLM	LHLM	α	SHLM	LHLM	α
Time(s)	82.1	79.9	0.000047	82.1	75.3	0	79.9	75.3	0.004165
Length(m)	138.5	133.7	0.000494	138.5	128.2	0	133.7	128.2	0.002458
Success rate(%)	90	95	N/A	90	95	N/A	95	95	N/A

Table C6 T-Test results of UAV flight data with LHLM and LHSM.

Type	LHLM	LHSM	α
Time(s)	75.3	86.2	0
Length(m)	128.2	130.0	0.131388
Success rate(%)	95	85	N/A

Figure C3 The indoor and outdoor experiment environments.

Appendix C.2 Experiments on a robotic platform

In this section, experiments were conducted on a multirotor aerial robot in both indoor and outdoor complex and GNSS-denied environments to evaluate the self-localization capability and the practicality of the proposed approach in search and rescue missions.

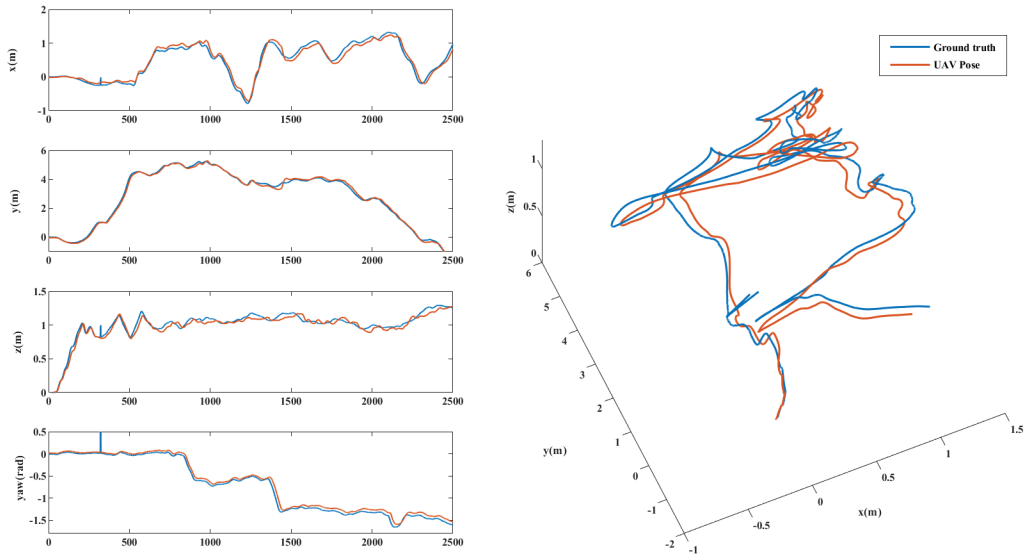
Appendix C.2.1 *Experimental Scenarios*

Indoor experiments were carried in two stages. An aggressive manual flight is performed to assess the self-localization performance at the first stage by comparing the robot states with the ground truth data. It is expected that, the proposed self-localization approach is capable of offering accurate measurements. Then at the second stage, the aerial robot is commanded to navigate through the indoor environment using the proposed system. In the outdoor experiment, the UAV takes off outside the dense woods, and flies through the woods towards a predefined position in UAV's local coordinate frame. No collision with obstacles is expected for both indoor and outdoor navigation experiments.

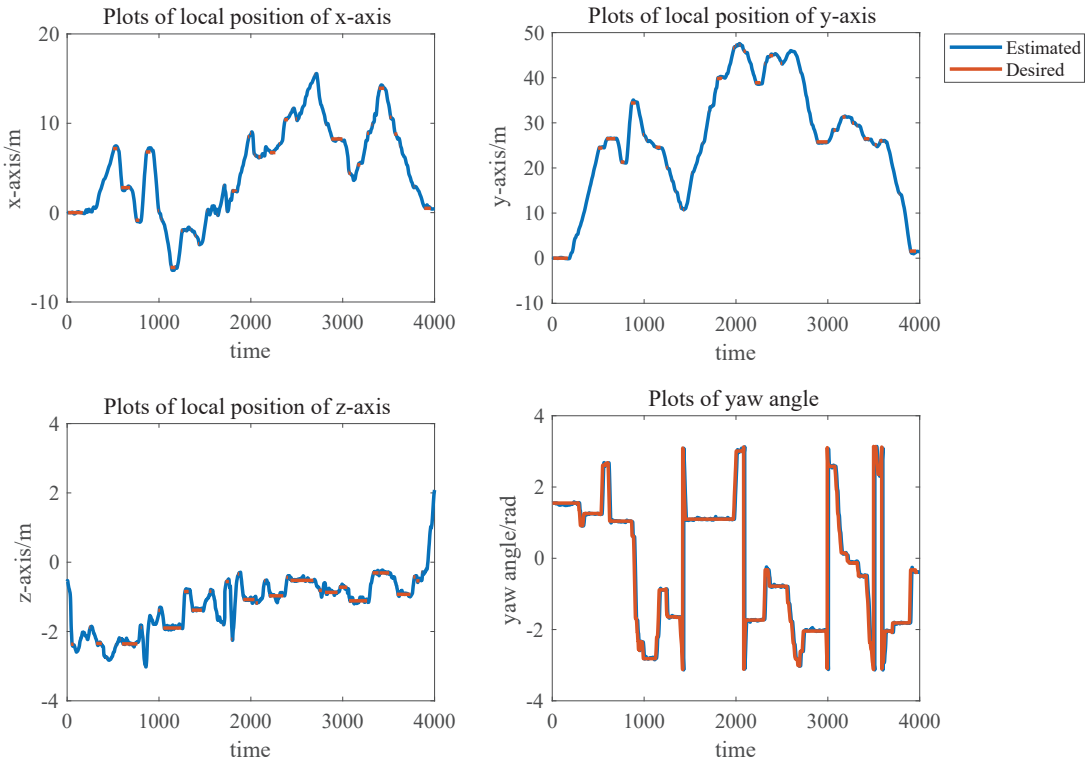
Appendix C.2.2 *Results*

Experimental results are illustrated in Fig. C4, Fig. C5 and Fig. C7. Screenshots of local navigation map in the indoor and outdoor experimental environment are shown in Fig. C5 and Fig. C7. The complete flight trajectories are shown in Fig. C6 and Fig. C6 respectively, the real-time path planned by navigation algorithm is shown as the red line, while the actual flight trajectory is shown as the blue line.

The result of the proposed self-localization system is compared with ground truth as shown in Fig. C4(a), where the mean self-localization error of x-axis, y-axis and yaw angle are less than 2%. The desired and estimated states during the indoor navigation flight are depicted in Fig. C4(b), where the red curves denote the desired positions and yaw angles and the blue curves denote the estimated states during flight. During the entire flight, the mean self-localization error is 0.25m, and the minimum distance from the obstacle is 0.4m. Both experimental results demonstrate that the proposed system is capable of offering accurate and robust state estimation in various motion conditions.



(a)



(b)

Figure C4 C4(a) Position and yaw angle of UAV's state estimation and Opti-Track. C4(b) Plots of desired and estimated position and yaw angle of UAV in the indoor navigation experiment.

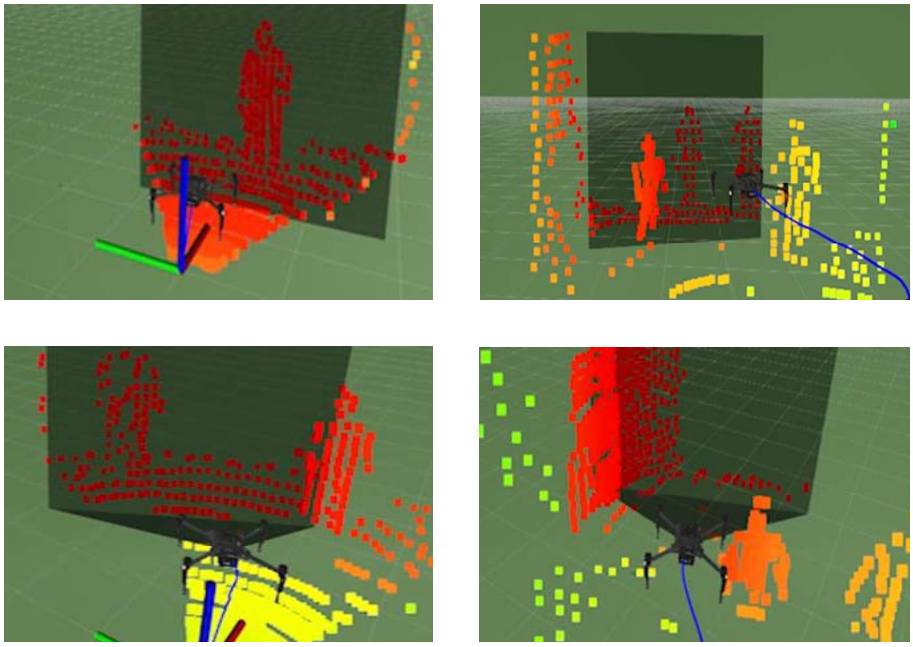


Figure C5 Screenshots of the local navigation map corresponding to the indoor experimental environment, where the different colors represent the chronological order of incoming point cloud. Flight trajectory is colored by blue.

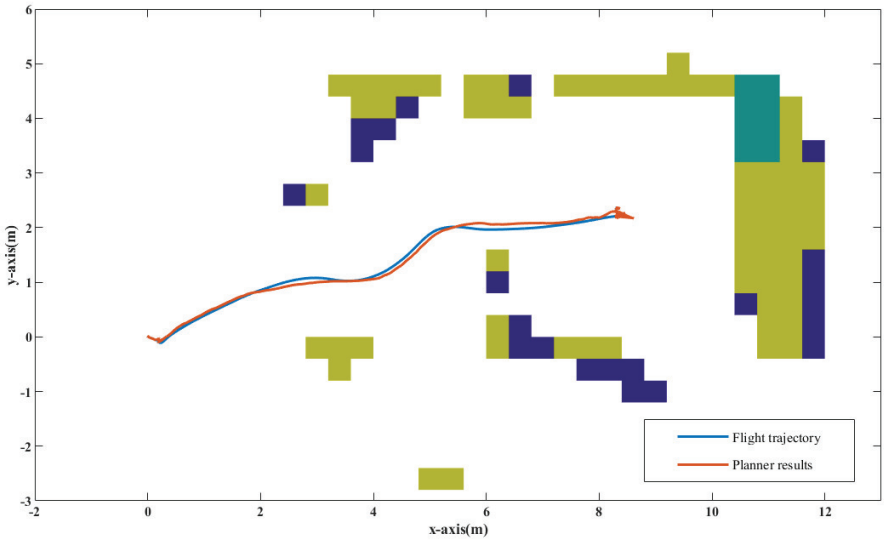


Figure C6 Flight trajectory and planner results of indoor self-localization and navigation experiment.

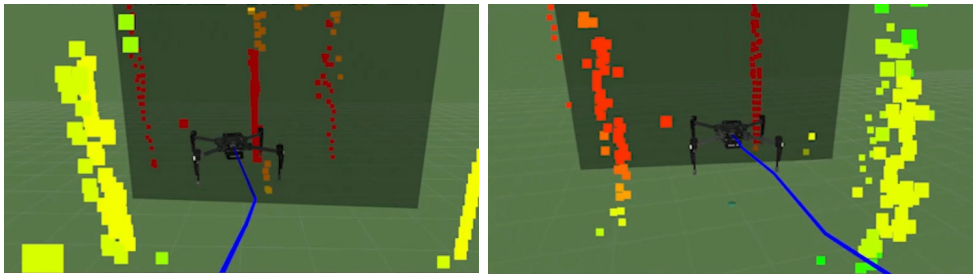


Figure C7 Screenshots of the local navigation map corresponding to the outdoor experimental environment.

In the outdoor experiment, the planned path and flight trajectory of the outdoor navigation are shown in Fig. C8, the total length of which is approx. 170m, including a 120m long path through a dense woods. During the experimental flight, there was no collision between the UAV and the environment. Authors believe that, experimental results have provided convincing evidence that the proposed approach is capable to facilitate the multirotor aerial robot's navigation system achieving safe flight in complex and GNSS-denied environments.

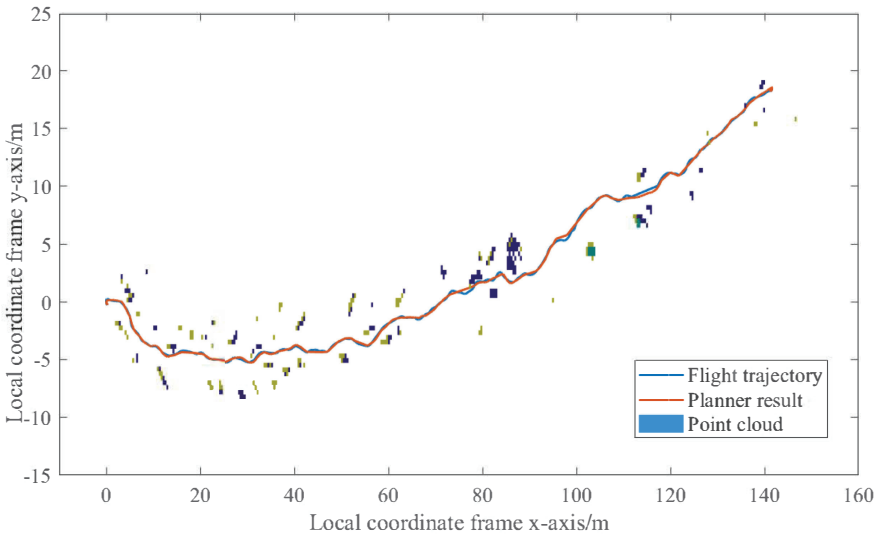


Figure C8 Planned path and flight trajectory of an outdoor navigation experiment in a dense woods.

Appendix C.3 Discussion

From the results and analysis of the experiments in a virtual environment, the navigation histogram with long-term memory is beneficial to the path planner to calculate a relatively optimal result, which represents a larger area of environment than short-term. Meanwhile, given the same memory for the navigation histogram, the heuristic function with long-distance can make better use of the navigation histogram to calculate a locally optimal path than the short-distance. It is clear that the planned path will approach closely to the globally optimal path as both memory and heuristics becoming longer. However, it is worth noting that both storing more memory and increasing the distance of heuristic function are computational expensive. Since reaching the destination safely is more important in navigation, the navigation histogram with memory has higher priority than that of the heuristic function. Therefore, limited computational resources should first be allocated to the construction of navigation histogram with memory.

In the experiment on the robotic platform, the path smoothness weight needs adjustments to adapt with different curvatures of environments, which may help the UAV to find better directions to fly through. Nevertheless, the elevation weight of the path planner should also be tuned carefully to avoid collision with over-the-head obstacles and acquire sufficient visual features for VIO computation.

Considering the limited computational resource onboard the UAV, all algorithms and softwares are carefully tuned and optimized. CPU usage of each module during the outdoor navigation experiment is provided in Table C7. It can be observed that there is still spare CPU computational power left to secure computer system stability.

Table C7 CPU usage during flight test.

Module	CPU usage(%)
`Sensor drivers`	7
`Fusion odometry`	22
`Back-end optimization`	10
`Path planning`	10
`Local mapping`	15
`Controller`	6
`Communication`	15

References

1 Leutenegger S, Lynen S, Bosse M et al. Keyframe-based visual-inertial odometry using nonlinear optimization. *International Journal of Robotics Research* 2014; 34(3): 314–334.

2 Gálvez-López D and Tardós J. Dbow2: Enhanced hierarchical bag-of-word library for c++, 2012.

3 Rublee E, Rabaud V, Konolige K et al. Orb: An efficient alternative to sift or surf. In *ICCV*, volume 11. Citeseer, p. 2.

4 Ramos J et al. Using tf-idf to determine word relevance in document queries. In *Instructional Conference on Machine Learning*, volume 242. Piscataway, NJ, pp. 133–142.

5 Zhang J and Singh S. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2. p. 9.

6 Labbe M and Michaud F. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics* 2013; 29(3): 734–745.

7 Dellaert F. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

8 Ulrich I and Borenstein J. Vfh*: Local obstacle avoidance with look-ahead verification. In *IEEE International Conference on Robotics and Automation.*, volume 3. IEEE, pp. 2505–2511.

9 Vanneste S, Bellekens B and Weyn M. 3dvfh+: Real-time three-dimensional obstacle avoidance using an octomap. In *International Conference on Software Technologies*. 1319, pp. 91–102.

10 Hart PE, Nilsson NJ and Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 1968; 4(2): 100–107.