Autonomous Flight of Micro Air Vehicles 2015 - Group 6

M. Bevernaegie[*], C. Cheung[†], Y.I. Jenie[‡], S.H. Lee[§], P. Lu[¶], M. Siddiquee[‖]
Delft University of Technology, Delft, The Netherlands

ABSTRACT

This should be done at the end

1 INTRODUCTION

1. Overall introduction of the project
2. Equipment - Opti-track, AR.Drone 2.0
3. Software - Paparazzi

2 VISION ALGORITHM

2.1 *Vision-based Navigation in the Literature*

Here we talk about the vision algorithm used in the literature

2.2 *Own Vision Algorithm*

In Section 2.1, a number of the vision-based navigation methods used in literatures were briefly introduced and discussed. Although these methods have shown considerably good results in certain work domains, our team decided to come up with more simple method for the obstacle detection and avoidance. The rationale behind this is as follows:

- Lucas-Kanade optic flow generation from Harris corner detector or FAST feature detector will not work well unless the obstacles have many "features" on their surface. In the competition, monotonic orange poles and a black wall will be used, and hence we anticipate that there will not be many features available on the texture of the obstacles.
- For the same reason, it will be very difficult to compute the Time-to-Contact because it primarily uses the optical flow vectors.
- Object recognition or image classification method can be difficult for tuning and other practical reasons. For example, in order to avoid the orange poles, one cannot simply judge from the overall color of the image as to whether the obstacle is close enough or not. This is because there are cases where multiple poles are visible in the image, although none of them are close enough to the drone. Also, the poles appearing from the "blind spot" of the camera's view are more likely to be hit than the poles located straight ahead (These are verified during the tests). In comparison to other simpler methods, the object recognition will not allow for a flexible adaption to account for all these various situations.
- As the drone has only one front camera, the mapping techniques, such as SLAM, will take considerably more computational time and memory. This technique does not go along well with the objective of the competition which is to enable the drone travel as much as possible while minimizing the collisions.

For these reasons, it was necessary to come up with much simpler vision algorithm which can be implemented and tested easily. The proposed vision algorithm uses a simple color thresholding method for the detection of specified obstacles, which is done in the following steps:

1. Take only the bottommost row of the pixels in the image as input. Note that the floor will be visible until the distance between the drone and an obstacle standing on the floor becomes lower than a certain threshold distance, assuming that the drone moves slowly in a hovering flight at a fixed altitude. For the visualization, refer to Section 4.
2. By setting a intensity (i.e. Y color channel) threshold, convert the bottommost pixel row into a binary row which has the elements of value 1 and 0 (i.e. 1 if the pixel intensity value is lower than the threshold, and 0 other wise). Therefore, if the image of the black wall is captured at the bottommost row, it will be indicated by the continuous sequence of 1's.
3. Count the number of continuous 1's, and if the sequence is longer than a certain threshold, namely the "black wall width-threshold", then mark the position as the position of a black wall. For each mark, give the weight based on the detected width. Determine either to turn left or right, depending on the weighted position of the black wall.
4. If there is no black wall detected (i.e. no continuous sequence of 1's exceeding the "black wall width threshold"), evaluate the Cr chroma value to detect the orange poles. Generate a binary row in the similar manner as for the black wall detection by setting an appropriate Cr threshold.

[*]M.Bevernaegie@student.tudelft.nl
[†]C.Cheung@student.tudelft.nl
[‡]Y.I.Jenie-1@tudelft.nl
[§]S.H.Lee-2@student.tudelft.nl
[¶]P.Lu-1@tudelft.nl
[‖]M.Siddiquee-1@student.tudelft.nl

5. In order to react more quickly and sensitively towards the poles appearing from the sideways, evaluate the left- and right-end of the Cr binary row to see if there is any shorter continuous sequence of 1's appearing from the sides. This necessitates a different type of width-threshold on the sides, namely the "orange pole sideway threshold". If there is no detection of an orange pole on the sides, continue to the next step.

6. Using the analogous procedure as for the black wall detection, determine either to turn left or right, depending on the weighted position of the orange pole which can be computed using another width-threshold, namely the "orange pole width-threshold".

7. If there is still no detection of the continuous sequence of 1 which exceeds any of the above-mentioned width-thresholds, then yield a command to fly forward.

3 Flight plan

3.1 Flight Plan I - Moving Waypoints

Seong's flight plan

3.2 Flight Plan II - Fixed Waypoints

Peng's flight plan

3.3 Comparison and Choice of Flight plan

Comparison and discussion.

4 Simulation

4.1 MATLAB & Paparazzi Simulation

Seong's method and results

5 Competition Result and Discussion

After tuning the control and vision threshold parameters through trial-and-error in real tests, it was confirmed that both vision algorithm and flight plan works very well with adequate sensitivity towards the distance from the orange poles and black wall. Also, it was observed that the drone could maintain a stable flight during the maneuvers, such as stopping and turning of the heading. The slow speed of the drone, together with the intermittent turning strategy, allowed the drone to stop and turn with only a marginal overshoot whenever an obstacle is detected or it goes outside the flight area. Regrettably, however, the tuning was done in the environment where the obstacles were rather spaced out widely, and in particular, the black wall was positioned near and parallel to one of the boundaries of the flight area.

During the actual competition, the black wall was positioned adjacently near one of the corners, and there was another orange pole next to it. What happened in the first half of the competition was: the drone moved and made a few turns, and eventually went to near the corner. After it realized that it was out of bounds, it made 180^o turn and attempted to come back inside. However it saw the adjacent black wall, and turned to the other side. Then it saw the orange wall and turned back to the previous heading. And it saw the black wall again, and so on. After drifting slowly towards the black wall, it touched the black wall and had to be restarted due to the low battery.

In the second half of the competition, the drone was started after being repositioned in the home position, so it did not go into the corner and did not get trapped again. As a result, the drone covered the traveled distance of well more than 100 m. Although this is not quite a high result compared to the other teams, our drone had the fewest collisions with the obstacle, and our team won the 6th place out of 13 teams in total.

In the following list, the main reasons are addressed as to why it could not be foreseen that the drone may get trapped near the boundaries/corners:

- In the Paparazzi simulation, the vision algorithm was "mimicked" by a random number generator. Hence, there was never a possibility of a drone getting trapped in a place.

- During the real tests for tuning, the black wall was located parallel to one of the boundaries and further way from the corner. So, the drone could not be possibly trapped by the "blocking" of the wall.

To solve this problem in the future, the following strategies may be used to improve the flight plan:

- "Breakthrough Maneuver": Incorporate a maneuver that when a drone makes more than two consecutive turns back and forth, turn the heading only 45^o and fly forward for a certain period of time while suppressing command from the vision result.

- "Temporary Blinding": In case the displacement of the drone is too low during a fixed time period, flexibly adjust the vision threshold parameters (e.g. the width threshold for the pole detection) such that the drone becomes less sensitive towards the obstacles nearby.

- "Circular Flight Area": Define the flight area as a circle, instead of a square. This will prevent the drone from going too close into a corner.

6 Conclusion

References

Appendix A: Data

Appendix B: More data