Indoor mapping with an autonomous quadcopter

Karel Serruys

Supervisor(s): Joni Dambre, Francis wyffels and Jonas Degrave

Abstract—**This article reports how a custom low-cost indoor quadcopter is optimized and extended to accommodate *Simultaneous Localization and Mapping* (SLAM). First the starting state of the platform is evaluated. Several optimizations have to be made to enable stable autonomous flight behaviour. Optical flow is used to detect drift, PID controllers are used to control the flight behaviour. When the platform has the desired stability, it is extended with a laser range scanner to perform SLAM. Several SLAM methods are considered. Eventually BreezySLAM is chosen. The performance of this algorithm is also evaluated. Finally a platform is obtained both capable of both flying autonomously and mapping its environment.**

Keywords—**autonomous quadcopter, indoor, optical flow, SLAM**

I. Introduction

The use of quadcopters is increasing in both private and commercial applications. They benefit from a simple mechanical design and thanks to the advances in electronics of the past few years, it can be made easy to control them. In comparison to ground robots, quadcopters can cope well with rough terrain as they only need to fly over it. Quadcopter can be used for tasks that are too dangerous, expensive or infeasible to be done by man. Examples are inspecting a bridge for micro ruptures [1], package delivery [2], etc. This article focusses on quadcopters for search and rescue missions. Here, quadcopters are used to map a damaged building and to search for survivors.
In this article an existing quadcopter platform is evaluated and optimized. When the platform is capable of stable autonomous flight, it is extended to accommodate *Simultaneous Localization and Mapping* (SLAM). With SLAM a map can be made of the quadcopter's environment.

II. Platform

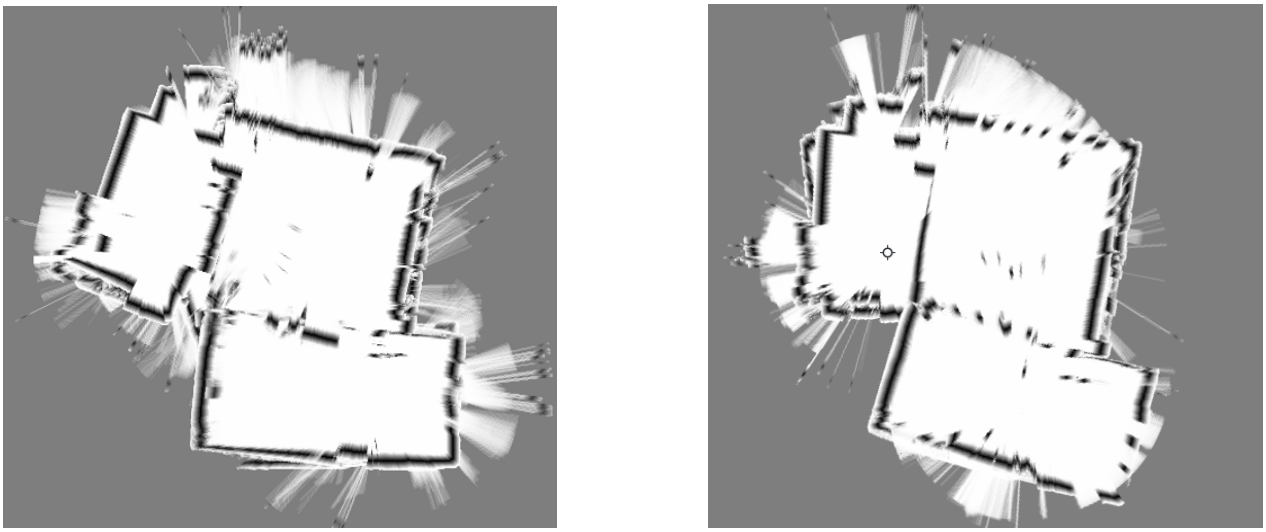
The quadcopter platform of W. De Gucht is the starting point of this work [3]. As autopilot, the ArdupilotMega 1.4 is used. The platform also uses a Raspberry Pi to calculate drift based on optical flow. Once drift is detected, the autopilot can make sure it is compensated. The Raspberry Pi is also employed to pass user commands to the autopilot. This platform however lacks the ability to allow safe autonomous flight. To accommodate SLAM, the platform must be equipped with a laser range scanner. The platform is optimized and subsequently extended to allow mapping of its surroundings.

A. Optimizations

The Raspberry Pi has a very limited processing power. Optical flow is calculated on the Pi, this leaves very little processing power for anything else. The Raspberry Pi is relieved from the computation of optical flow by using a PC instead. When using the PC the resolution over which optical flow is calculated can be doubled while keeping the same frame rate. This benefits the accuracy of the calculated optical flow. Detecting drift with higher accuracy leads to more accurate drift compensation. As a result the platform will be more stable.
Optical flow is calculated by searching features of one frame in a second frame. The accuracy of optical flow is highly dependent on the features selected in the first frame. To further improve the accuracy of drift detection, a different feature detector is proposed. As computational power is no problem on a modern day laptop, the more computationally expensive GFTT detector [4] is implemented. This detector finds optimal features by construction. When comparing optical flow with the GFTT detector to the original FAST detector [5], the use of GFTT clearly performs better, especially when the platform is moving fast and frames become blurry.
Next is the height PID controller. The current height is a function of the throttle of the platform. The assumption is made that there exists a throttle which allows the platform to hover. In hover the platform height is constant. Finding this throttle value is not easy as this value is dependent on the battery voltage. The lower the voltage, the higher the throttle value is. The ideal throttle value is estimated during take off by measuring the vertical acceleration of the platform. The error on this estimation needs to be compensated by the integral term of the PID controller. When the estimation error is large, this will take a long time. The correct estimation of the throttle value leads to better height control just after take off. Optimizing the PID parameters according to Ziegler et al. [6] leads to better height control in later flight.
Height control is already a lot better, but there is still quite a bit overshoot. Furthermore fast ascending and descending is bad for stability. This is caused by the fact that optical flow detects drift during ascending and descending that is not actually there. This fake drift leads to an unwanted compensation by the autopilot. To solve this issue, a function generator is proposed. The maximum vertical speed of the platform is limited. When the platform moves slower, the fake drift detected by optical flow is smaller, causing a smaller unwanted compensation and a more stable flight behaviour. The slower vertical movement also reduces the overshoot of the platform.
Finally the position controller is looked into. A PI controller is used to control the horizontal speed of the platform. The platform is kept at a certain position by keeping the setpoint of the controller at zero. To move to a specific location in space the velocity in this direction can be augmented until the platform has reached the desired position. The maximum horizontal speed of the platform is also limited to prevent overshoot when the quadcopter moves from one location to another.
The platform is now able to fly autonomously and ready to be extended.

B. Extensions

The platform is extended with a URG-04LX-UG01 laser range scanner. This type of laser range scanner can be con-



(a)Map generated without the use of odometry.

(b)Map generated using odometry.

Fig. 1. This figure show a map of three rooms generated without the use of odometry (a) and with the use of odometry (b). The rooms are more correctly aligned when odometry is used.

nected to one of the USB ports on the Raspberry Pi. However the Raspberry Pi alone is not able to deliver the required current to operate the laser range scanner. To counter this problem the scanner receives extra current from the ArdupilotMega. Now everything is in place to implement a fitting SLAM algorithm.

III. SLAM

For SLAM a lot of algorithms exist. The algorithms can be divided into two paradigms [7]. There are those that use extended Kalman filters to estimate the position of the platform and the map points or landmarks. The second paradigm uses particle filters for this estimation. It is essential to know the principles of both paradigms to make a good choice between the two.

A. Extended Kalman filters

When using extended Kalman filters, the positions of the platform and all the landmarks are assumed to be jointly gaussian distributed. They can be represented by a gaussian variable. This variable is described by its mean vector and a by a covariance matrix. When a new scan is available, the entire mean vector and covariance matrix have to be updated. This is computationally very expensive [8]. Moreover, it has been proven that this paradigm can leads to inconsistencies for larger maps [9].

B. Particle filters

The second paradigm uses a number of particles to represent the possible states in which the platform can be. Also, the platform position is assumed to be independent of the landmarks. Consequently landmarks are also independent of each other. The distribution of the landmarks is still assumed to be gaussian. For a 2D map this means that a landmark is described by a mean vector of length two and a 2x2 covariance matrix. The reader can already sense this greatly reduces computational complexity. By grouping the landmarks in a binary tree, the complexity is even further reduced [8].

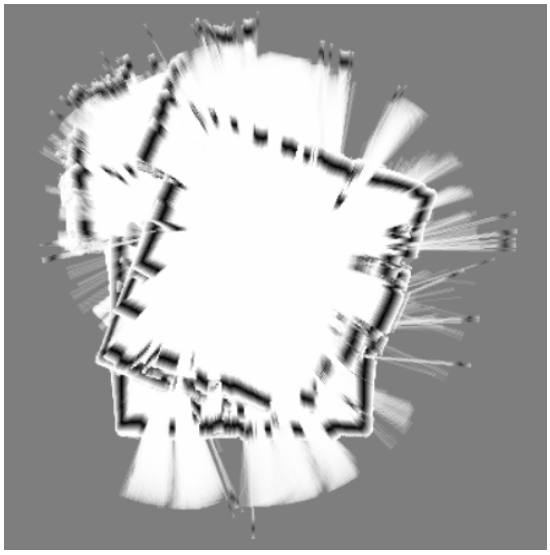


Fig. 2. This figure illustrates the effect of visiting the same location twice. A new map is redrawn above the map that was already there. This is called the loop closing problem.

C. BreezySLAM

From the previous two sections, one can conclude that when looking at computational complexity alone, an algorithm with particle filters needs to be used.
First HectorSLAM [10] is looked into. This algorithm uses odometry of all six degrees of freedom a quadcopter has. Implementing it however has failed. Next an attempt has been made to implement TinySLAM [11], without success, due to its complex interface. Finally BreezySLAM [12] is implemented, with success. This algorithm is based on TinySLAM. BreezySLAM can also use odometry, but only forward displacement and rotation around the vertical axis are taken into account. This should however not pose any problem as the platform ensures stable flight behaviour.

IV. Integration of SLAM

Once the SLAM algorithm is chosen, it has to be implemented in the existing architecture of the platform. BreezySLAM can be used both without and with odometry. When mapping multiple rooms, using odometry helps to generate a better map as can be seen in figure 1. There are still some problems as the algorithm does not perform well when a single location is visited twice. This is illustrated in figure 2. In SLAM literature, this is called the loop closing problem [7].
Despite the loop closing problem, the platform is capable of generating a map. This illustrates that the platform is suitable for exploration of unknown environments.

V. Conclusion

First of all the quadcopter flight behaviour has been optimized. Drift detection is now more accurate because of the higher resolution of the frames and because of the better GFTT feature detector. By implementing a better height and position control the platform is even more stable. I conclude that the platform is now capable of safe autonomous flight.
Second, the platform has been extended with a laser range scanner. By using BreezySLAM the platform is now able to make a relatively accurate map of three rooms. My conclusion is that the platform is capable of mapping a small unknown environment.

References

[1] Z. Yin, "A Quadcopter with Heterogeneous Sensors for Autonomous Bridge Inspection," 2014.

[2] M. R. Haque, M. Muhammad, D. Swarnaker, en M. Arifuzzaman, "Autonomous Quadcopter for Product Home Delivery," *2014 International Conference on Electrical Engineering and Information and Communication Technology (ICEEICT)*, 2014.

[3] W. De Gucht, "Ontwerp van een autonome quadcopter," Thesis, Universiteit Gent, 2014.

[4] J. Shi en C. Tomasi, "Good Features to Track," *Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[5] E. Rosten en T. Drummond, "Machine learning for high-speed corner detection," *European Conference on Computer Vision*, vol. 1, pp. 430–443, 2006.

[6] J. G. Ziegler en N. B. Nichols, "Optimum settings for automatic controllers," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, no. 2B, pp. 220–222, 1993.

[7] S. Thrun en J. J. Leonard, *Springer Handbook of Robotics, and Technology*. Springer, 2008.

[8] M. Montemerlo, S. Thrun, D. Koller, en B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," 2002.

[9] T. Bailey, J. Nieto, J. Guivant, M. Stevens, en E. Nebot, "Consistency of the EKF-SLAM Algorithm," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3562–3568, 2006.

[10] S. Kohlbrecher, J. Meyer, O. von Stryk, en U. Klingauf, "A Flexible and Scalable SLAM System with Full 3D Motion Estimation," November 2011.

[11] B. Steux en O. E. Hamzaoui, "SLAM algorithm with parallel localization loops: TinySLAM 1.1," *2011 International Conference on Automation and Logistics*, 2011.

[12] S. Bajracharya, "BreezySLAM: A Simple, efficient, cross-platform Python package for Simultaneous Localization and Mapping," Thesis, 2014.