Autonomous Indoors Navigation using a Small-Size Quadrotor

Slawomir Grzonka Giorgio Grisetti Wolfram Burgard

Autonomous Systems Lab, Department of Computer Science
University of Freiburg, D-79110 Freiburg, Germany
{grzonka, grisetti, burgard}@informatik.uni-freiburg.de

Abstract. Recently there has been increasing research on the development of autonomous flying vehicles. Whereas most of the proposed approaches are suitable for outdoor operation, only a few techniques have been designed for indoor environments. In this paper we present a navigation system for an indoor quadrotor. Our system adapt techniques which have been successfully applied on ground robots to our flying platform. We validate our system with real-world experiments.

1 Introduction

Low-cost and small-size quadrotors are becoming broadly available. Some of these vehicles are able to lift relatively high payloads and provide an increasingly broad set of basic functionalities. This enables even unexperienced pilots to control these vehicles and allows them to be equipped with autonomous navigation abilities. Most of the proposed approaches for autonomous flying [15, 6, 15] focus on systems for outdoor operation. Vehicles that can autonomously operate in indoor environments are envisioned to be useful for a variety of applications including surveillance and search and rescue. In such settings and compared to ground vehicles, the main advantage of flying devices is their increased mobility.

As for ground vehicles, the main task for an autonomous flying robot consists in reaching a desired location in an unsupervised manner, i.e. without human interference. In the literature, this task is known as *navigation*. To address the general task of navigation one requires to tackle a set of problems ranging from state estimation to trajectory planning. Most of these tasks have been successfully addressed by using ground robots.

Whereas the general principles of the navigation algorithms, which have been successfully applied on ground robots, could in principle be transferred to flying vehicles, this transfer is not straightforward for several reasons. First, due to their limited payload and size an indoor flying robot cannot carry the variety of sensors which can be easily mounted on a mobile robot. Second, the additional degrees of freedom of the vehicle prevents the direct use of well known and efficient 2D algorithms for navigation. Third the dynamics of a flying robot is substantially more complex than that of ground-based vehicles which makes them harder to control. Finally, one has to consider the increased risk of damaging the platform

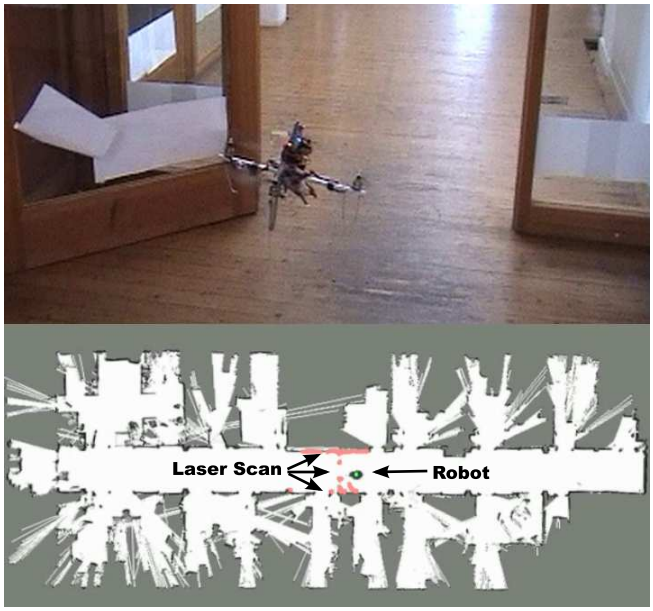

Fig. 1. Our Autonomous quadrotor during a mission (top) and position of the vehicle estimated on-line during the flight by our localization algorithm.

during testing. Whereas a failure in the stabilization of the pose of a ground robot may result in a crash, the user can typically stop the system in time. In case of a flying vehicle this operation is not possible: stopping causes its fall on the ground.

In this paper, we describe an autonomous quadrotor based on an open-source hardware project, namely the Mikrokopter [1]. Our system is a result of an integrated hardware/software design which meets several of the challenging constraints imposed by the limited payload of the platform while preserving large degree of flexibility for future extensions. Figure 1 shows the Mikrokopter equipped with our navigation system during a mission.

The remainder of this paper is organized as follows. In Section 2 we give an overview of the related literature. Subsequently, we present our system in sections 3 and 4. We conclude with a set of experiments which illustrate the functionalities currently implemented on our platform in Section 5.

2 Related Work

In the last decade, flying platforms received an increasing attention from the research community. Many authors focused on the modeling and on the control of these vehicles [11, 13, 14, 2], with a particular emphasis on small helicopters.

Hoffmann *et al.* [10] present a model-based algorithm for autonomous flying with their STARMAC-quadrotor. Their system flies outdoors and utilizes GPS

Fig. 2. The quadrotor platform used to evaluate the navigation system includes a Mikrokopter (1), Hokuyo laser range finder (2), an XSens IMU (3), a Gumstix computer (4), and a laser mirror (5).

and IMU measurements. Bouabdallah *et al.* [4, 5] developed a complete model of their quadrotor platform and a set of different control strategies. Recently [3] they discussed the requirements of a flying platform for indoor navigation. Ng and colleagues [6] have developed algorithms for learning controllers for autonomous helicopter navigation. Their approach allows helicopters to perform impressive maneuvres in outdoor environments. Tournier *et al.* [16] used monocular vision to estimate and stabilize the current pose of a quadrotor. Thrun *et al.* [15] used a remotely controlled helicopter to learn large-scale outdoor 3D models.

There also has been some work that addressed the navigation of flying vehicles in indoor environments and in absence of GPS signal. Roberts *et al.* [12] used ultrasound sensors for controlling a flying vehicle in a structured testing environment, while He *et al.* [9] presented a system for navigating a small-size quadrotor without GPS. The pose of the vehicle is estimated by an unscented Kalman filter. Whenever the robot has to reach a given location, a path which ensures a good observation density is computed from a predefined map. These highly dense observations minimize the risk of localization failures.

In contrast to this approach, our quadrotor is suitable to be used on less structured environments which can be effectively represented by grid maps. We focus on adapting a set of algorithms which have been validated on ground robots to indoor flying platforms.

3 Hardware Architecture

The hardware of our quadrotor is similar to the one proposed by He *et al.* [9] and is shown in Figure 2. We equipped the platform with the following devices:

– a Linux-based Gumstix embedded PC with USB interfaces and a WiFi network card,

– an Hokuyo-URG miniature laser sensor for localization and obstacle avoidance,
– an XSens MTi-G MEMS inertial magnetic unit (IMU) for estimating the attitude of the vehicle, and
– a mirror which is used to deflect some of the laser beams along the z direction to measure the distance from the ground.

The Gumstix communicates with the microcontroller on the quad-rotor via an RS-232 interface and reads all the sensors. We use the laser range finder for both measuring the distances to the obstacles in the surrounding of the robot and the distance from the ground. The IMU provides accurate estimates of the roll and the pitch of the vehicle, which are directly used for localization.

For safety reasons, the user can always control the vehicle via a remote control (RC) and our system mixes the user and the program commands. During our experiments, we allow the programs to perturb the user commands by $\pm 20\%$. In this way, if one of the control modules fail the user still has the possibility of safely land the vehicle without loosing time of pressing a button first.

4 Navigation system

In this section, we present the functionalities currently implemented in our quadrotor. It is based on a modular architecture in which the different components communicate via the network using a publish-subscribe mechanism. At the current state, all the device drivers and some time-critical modules are executed on-board. The more computing-intensive algorithms for localization as well as the user interface are executed on a remote PC that communicates over wireless network with the platform.

The roll ϕ and pitch θ measured by the IMU are typically accurate up to 1°, which is sufficient for localization. In practice, we therefore calculate only four of the six components of the vehicle pose vector $\mathbf{x} = (x\ y\ z\ \phi\ \theta\ \psi)^T$, namely the 3D position $(x\ y\ z\ )^T$ and the yaw ψ.

The only sensor used for measuring the distances of nearby objects is the laser range finder. Based on known calibration parameters and on the attitude estimated by the IMU, we project the endpoints of the laser in the global frame. We address the problems of controlling and stabilizing the platform along different partitions of the state space separately. From the projected laser beams, we estimate the $x - y$ position and the yaw ψ of the vehicle in a 2D map. To compensate for the lack of odometry measurements we estimate the incremental movements by 2D scan matching. Finally, we control the altitude of the vehicle by fusing the IMU accelerometers and the distance from the ground as measured by the laser.

In the remainder of this section, we first discuss the projection of the laser data and the estimation of the relative motion between subsequent laser scans. Subsequently, we present our localization module and conclude by discussing the control algorithms.

4.1 Projection of the Laser Data

In this section, we explain how we project the laser data in the global frame of the helicopter, given a set of known calibration parameters. The laser range finder measures a set of distances b_i along the $x-y$ plane, in its own reference frame. Each of these distances can be represented by a homogeneous vector $\mathbf{b}_i$ in the 3D space $\mathbf{b}_i = (b_i \cos\alpha_i \;\; b_i \sin\alpha_i \;\; 0 \;\; 1)^T$, where α_i is the angle of the individual beams. Let $T_{\text{IMU}}^{\text{laser}}$ be the homogeneous transformation matrix from the IMU reference frame to the laser frame, known from a calibration procedure and let $T_{\text{world}}^{\text{IMU}}$ be the time dependent transformation from the world to the IMU. Note that $T_{\text{world}}^{\text{IMU}}$ is computed only from the estimated pitch and roll. We can compute the position of a laser endpoint $\mathbf{b}'_i$ which is *not* deflected by the mirror by the following equation:

$$\mathbf{b}'_i = T_{\text{world}}^{\text{IMU}} \cdot T_{\text{IMU}}^{\text{laser}} \cdot \mathbf{b}_i \tag{1}$$

Conversely, if a beam is deflected by the mirror, we obtain the point $\mathbf{h}'_i$ in the world frame by the following chain of transformations:

$$\mathbf{h}'_i = T_{\text{world}}^{\text{IMU}} \cdot T_{\text{IMU}}^{\text{mirror}} \cdot \mathbf{b}_i \tag{2}$$

Here, $T_{\text{IMU}}^{\text{mirror}}$ represents the transformation between the IMU and the *virtual* laser position which accounts for the effect of the mirror.

4.2 Incremental Motion Estimation

Some tasks, like pose stabilization, do not require to know the absolute location of the vehicle in the environment. Conversely, they rely on an accurate local pose estimate. We can estimate the relative movement of the robot between two subsequent scans by means of a scan matching procedure. Since the attitude is known from the IMU, this procedure can be carried on in 2D. In our implementation, we use an approach similar to [8]. This algorithm estimates the most likely pose of the vehicle $\hat{\mathbf{x}}_t$ given the previous pose $\mathbf{x}_{t-1}$, the current projected laser measurements $\mathbf{b}'_t$ and the previous one $\mathbf{b}'_{t-1}$, as follows

$$\hat{\mathbf{x}}_t = \underset{\mathbf{x}:=(x,y,\theta)}{\operatorname{argmax}} \; p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{b}'_{t-1}, \mathbf{b}'_t), \tag{3}$$

In our implementation we use a constant velocity model to compute the initial guess for the search.

4.3 Localization

We estimate the 2D position of the robot in a given grid-map by Monte-Carlo Localization [7]. The idea is to use a particle filter to track the position of the robot. Whenever the robot travels over certain distance, we sample the next generation of particles based from a proposal distribution according to

$$\mathbf{x}_t^{[i]} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{[i]}, \mathbf{v}_t, \Delta\mathbf{x}) \tag{4}$$

where $\mathbf{x}_t^{[i]}$ is the generated sample, $\mathbf{x}_{t-1}^{[i]}$ is the previous sample, $\mathbf{v}_t$ are the velocities computed by integrating the IMU accelerations, and $\Delta\mathbf{x}$ is the relative movement estimated by the scan matcher. Subsequently, we sample a new set of particles proportional to likelihood

$$p(\mathbf{b}'_t|\mathbf{x}_t^{[i]}, \mathbf{m}) \tag{5}$$

of the measurement. Here $\mathbf{b}'_t$ is the current projected laser beam, $\mathbf{x}_t^{[i]}$ is the pose of the particle, and $\mathbf{m}$ is the known map. Note that whenever we use a scan for computing the odometry, the same scan is excluded from the evaluation of the likelihood. This prevents us from reusing the same information, which ultimately would result in overly confident estimates.

4.4 Control

The altitude is controlled by a PID controller which utilizes the current height estimate z and the velocity v_z respectively. The height control C_h can be summarized as

$$C_h = K_p \cdot (z - z^*) + K_i \cdot e_z + K_d \cdot v_z, \tag{6}$$

with K_p, K_i and K_d being the constants for the P, I, and D part respectively. Here z^* denotes the desired height and e_z denotes the integrated error.

The yaw is controlled by a proportional controller which computes the yaw command C_ψ as

$$C_\psi = K_p \cdot (\psi - \psi^*). \tag{7}$$

Here ψ and ψ^* are the measured and desired yaw.

5 Experiments

In this section we present experiments for each of our modules described above, namely localization, altitude, and yaw control. During the experiments, altitude and yaw control were executed on-board, while scan matching, localization was executed off-board on a standard laptop computer.

5.1 Localization

Using 2D grid maps for localization enables our system to operate with maps acquired by different kind of robots and not necessarily built by the flying vehicle itself. In this section we present an experiment in which we perform global localization of the flying vehicle in a map acquired with a ground-based robot. This robot was equipped with a Sick Laser range scanner. The height of the scanner was $80\,cm$. Throughout this experiment, the particle filter algorithm

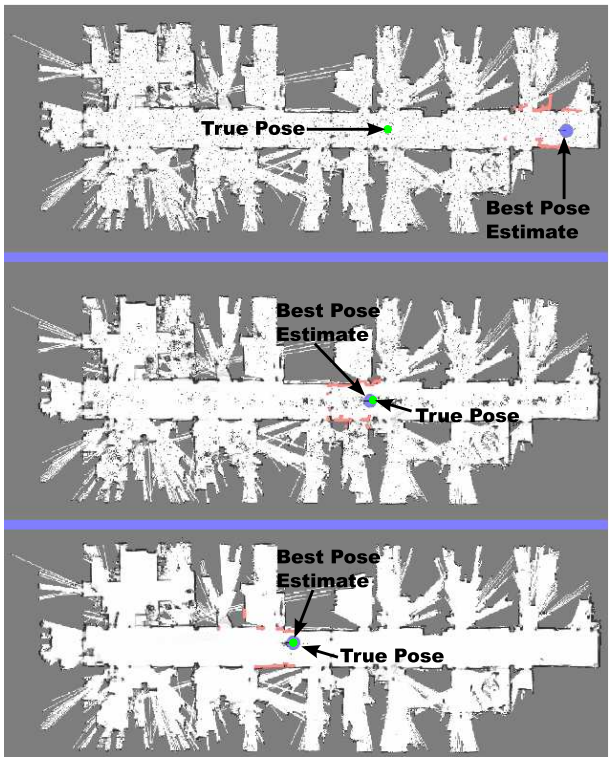

Fig. 3. Global localization of our quadrotor in a map, previously acquired by a ground-based platform. The blue circle highlights the current best estimate of the particle filter. The green circle marks the true pose of the vehicle. All potential robot poses are visualized as small black dots within the free (white) space of the environment. Top: initial situation. Middle: after about $1\,m$ of flight. Bottom: after approximately $5\,m$ of flight the quadrotor is localized.

employed 5,000 particles. Given this number of particles, our current implementation requires $5\,ms$ on a Dual-Core 2 GHz laptop, while scan matching requires 30 ms on average. Figure 3 shows three snapshots of the localization process at three different points in time. The top image depicts the initial situation, in which the particles were separated uniformly over the free space. After approximately $1\,m$ of flight (middle image), the particles start to focus around the true pose of the vehicle. After approximately $5\,m$ the quadrotor was globally localized (bottom image). The blue circle highlights the current best estimate by the filter. A full video of a localization run is available on the Web under `www.informatik.uni-freiburg.de/ ~grzonka/localization.avi`.

5.2 Altitude and Yaw Control

In this final experiment, we show the capabilities of our yaw and altitude control modules. The yaw controller receives as input the yaw estimate coming from the

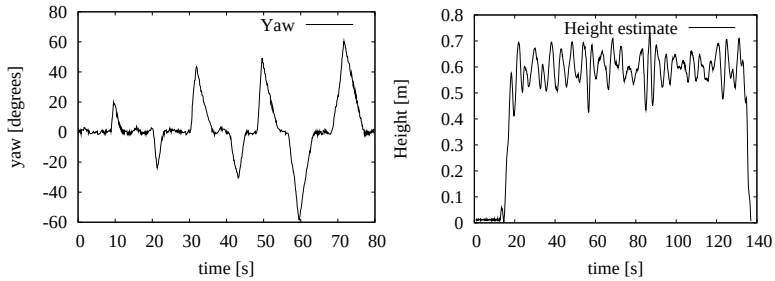

Fig. 4. Experiments for the autonomous stabilization of yaw (left) and height (right). During the yaw stabilization experiment, the quadrotor was required to rotate to 0°. From time to time, the user manually changed the yaw . After the user released the remote control, the quadrotor autonomously rotated back to the desired yaw angle. During the height experiment (right) the quadrotor was required to maintain height of $60\,cm$. The resulting error in height was $\pm 10\,cm$.

scan matcher. For testing the yaw controller, we set a desired yaw of 0° and once in a while, we turned the helicopter via the remote control. When the user released the rc, the vehicle always returned back to its desired yaw with an error of $\pm 2^\circ$. Figure 4 (left) plots the outcome of a typical run for yaw stabilization.

In a subsequent experiment, we tested the altitude stabilization. The designated altitude was $60\,cm$. In the beginning the vehicle was hovering over the ground. After enabling the stabilization the vehicle started climbing to the desired altitude. The desired height was kept by the vehicle up to an error of ± 10 cm. The results are shown in Figure 4 (right).

6 Conclusions

In this paper, we proposed an autonomous quadrotor which can operate indoors. Our current system includes major relevant state estimation modules for localization and attitude estimation. We furthermore implemented a yaw and altitude control and an effective user interaction approach which allows to reduce the risk of collisions. Our system adapts a set of techniques which have been validated with ground robots, and it can also operate with data acquired by such platforms. We furthermore implemented some control strategies for yaw and altitude stabilization which can be further improved by incorporating a vehicle-specific model.

7 Acknowledgments

This work has been supported by the EC under contract number FP6-IST-034120 Micro/Nano based Systems

References

1. Mikrokopter, http://www.mikrokopter.de/.
2. E. Altug, J.P. Ostrowski, and R. Mahony. Control of a quadrotor helicopter using visual feedback. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.
3. S. Bouabdallah, M. Becker, and R. Siegwart. Autonomous Miniature Flying Robots: Coming Soon! *IEEE Robotics and Automation Magazine*, 13(3), September 2007.
4. S. Bouabdallah, P. Murrieri, and R. Siegwart. Towards Autonomous Indoor Micro VTOL. *Autonomous Robots*, 18(2), March 2005.
5. S. Bouabdallah and R. Siegwart. Full Control of a Quadrotor. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.
6. A. Coates, P. Abbeel, and A.Y. Ng. Learning for Control from Multiple Demonstrations. *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.
7. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.
8. D. Hähnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.
9. R. He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a GPS-denied environment. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.
10. G. Hoffmann, DG Rajnarayan, SL Waslander, D. Dostal, JS Jang, and CJ Tomlin. The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC). *The 23rd Digital Avionics Systems Conference (DASC).*, 2, 2004.
11. P. Pounds, R. Mahony, and P. Corke. Modelling and Control of a Quad-Rotor Robot. *Proceedings of the Australasian Conference on Robotics and Automation (ACRA)*, 2006.
12. J.F. Roberts, T. Stirling, J.C. Zufferey, and D. Floreano. Quadrotor Using Minimal Sensing For Autonomous Indoor Flight. *European Micro Air Vehicle Conference and Flight Competition (EMAV)*, 2007.
13. A. Tayebi and S. McGilvray. Attitude stabilization of a four-rotor aerial robot. *43rd IEEE Conference on Decision and Control (CDC)*, 2, 2004.
14. A. Tayebi and S. McGilvray. Attitude stabilization of a VTOL quadrotor aircraft. *Control Systems Technology, IEEE Transactions on*, 14(3):562–571, 2006.
15. S. Thrun, M. Diel, and D. Hahnel. Scan Alignment and 3-D Surface Modeling with a Helicopter Platform. *Field and Service Robotics (STAR Springer Tracts in Advanced Robotics)*, 24:287–297, 2006.
16. G.P. Tournier, M. Valenti, J.P. How, and E. Feron. Estimation and Control of a Quadrotor Vehicle Using Monocular Vision and Moire Patterns. *AIAA Guidance, Navigation and Control Conference and Exhibit*, pages 21–24, 2006.