Assisted Teleoperation of Quadcopters Using Obstacle Avoidance *

João Mendes, Rodrigo Ventura

Abstract:

Teleoperation of unmanned aerial vehicles often demands extensive training. Yet, even well trained pilots are prone to mistakes, resulting frequently in collisions of the vehicle with obstacles. This paper presents a method to assist the tele-operation of a quadrotor using an obstacle avoidance approach. In particular, rough map of the nearby environment is constructed using sonar sensors. This map is constructed using FastSLAM to allow tracking of the vehicle position with respect to the map. The map is then used to override operator commands that may lead to a collision. An unknown and GPS denied environment is considered. Experimental results using the USARsim simulator are presented.

Keywords: Collision avoidance, 3D FastSLAM, Quadcopter, Occupancy Grid Mapping

1. Introduction

A quadcopter is an aircraft propelled by four rotors. This type of vehicle fits in the Vertical Take Off and Landing (VTOL) category as they can successfully perform vertical take offs, landings and hovering despite being heavier than air. The advantages of VTOLs to other flying mechanisms are notorious, as shown in [1].

However, flying a quadrotor is not a simple task: for a pilot to have a safe flight and control the vehicle as desired, significant experience is required. Even extensively trained pilots may face situations where it is difficult to guarantee a safe flight due to, for instance, loss of visual contact. A method to automatically avoid collisions is, therefore, a major necessity.

The presented solution is an assistive algorithm for tele-operation of a quadrotor in order to avoid collisions based on a FastSLAM [2] approach using an occupancy grid map [3]. Since the purpose of this work is neither a detailed map of the environment nor a precise measurement of obstacles positions, the problem can be efficiently addressed by knowing the relative position of the quadrotor in relation to the object. After knowing the vehicle's position and map, a decision making process based on danger assessment, performed by a classifier, is applied. This classifier overrides the user's inputs whenever they compromise the quadcopter's physical integrity in the near future. Overriding may range from simple velocity reduction to, in extreme cases, an evasive maneuver. Unknown areas are, for the sake of safety, always considered as occupied. The main difference between our approach and a simple reactive algorithm is memory. Unlike a purely reactive methodology, if the map is kept in memory it is possible to avoid crashes in sonar's blind spots. The full architecture is presented in Figure 1.

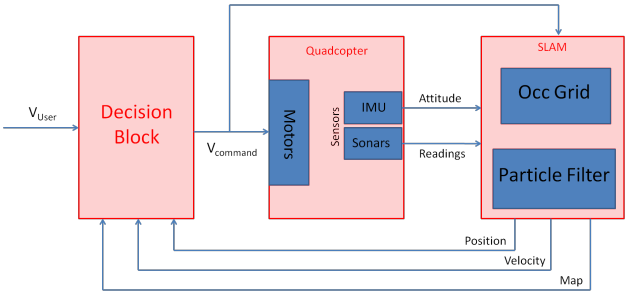

Fig. 1. Full architecture of the proposed approach

3D Simultaneous Localization and Mapping (SLAM) in Unmanned Aerial Vehicles (UAV) using lasers has been studied but typically including techniques, such as loop closure algorithms [4]. As for obstacle avoidance methodologies for UAVs, literature mostly addresses for path replaning topics [6]. This paper differs from the above in that we aim at a rough and low complexity map, and thus more time efficient.

We consider as a simulation test bed a quadrotor equipped with an Inertial Measuring Unit (IMU), an altimeter, and six sonars: one above each of the propellers pointing sideways, one above and one below the main body of the quadcopter.

2. Methodology

2.1. FastSLAM

Correct attitude is assumed to be given at all times by an IMU since accurate attitude estimations can be provided by a commercial solution, thus the 6D problem (position and attitude) is reduced to a 3D problem (position only). The objective of SLAM is to estimate the position and the map of a robot simultaneously. Let $x_{1:t}$ denote the path of the robot, m the map, $z_{1:t}$ all measurements and $u_{1:t}$ all control inputs where $1:t$ represents the time step from 1 to t. To solve the SLAM problem we use the FastSLAM approach proposed by Motermerlo et al. [5]. By performing a factorization of the posterior:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) \quad (1)$$

decomposing SLAM into a path estimation problem and a mapping problem (2) hereby solved by a combination of a Particle Filter with an occupancy grid mapping algorithm.

$$p(x_{1:t}|z_{1:t}, u_{1:t})p(m|z_{1:t}, x_{1:t}) \quad (2)$$

*This work was supported by the FCT projects [PEst-OE/EEI/LA0009/2011] and [PTDC/EIA-CCO/113257/2009].

The occupancy grid mapping algorithm uses a straightforward application of the inverse sensor model described in [7] where the posterior probability of the map is approximated as the product of the probability of each cell (3). This probability represents the belief on the occupancy of each individual cell.

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \tag{3}$$

Each cell not being updated is subject to a slow decay towards the value of the prior $p(m_{prior})$, in order to introduce a forgetness factor to all long time non-observed cells.

Localization estimation is provided by a bootstrap Particle Filter [8]. It is a Bayesian state estimation method which approximates the posterior $bel(x_t) = p(x_t|z_t, u_t)$ by a set of weighted particles S_t

$$\begin{aligned} S_t &= \{s_t^{[1]}, s_t^{[2]}, ..., s_t^{[N]}\} \\ &= \{[x_t^{[1]} w_t^{[1]} m^{[1]}], [x_t^{[2]} w_t^{[2]} m^{[2]}], ..., [x_t^{[N]} w_t^{[N]} m^{[N]}]\} \end{aligned} \tag{4}$$

where each particle s_t contains a different hypothesis

$$\tilde{x}_t^{[n]} = [X^{[n]} Y^{[n]} Z^{[n]}]^T \tag{5}$$

of the state to estimate $x_t = [X_r Y_r Z_r]^T$. Multiple particles, considering a total number equal to N, are used and to each one is associated a weight, w_t, representing the importance of that specific hypothesis.

For each iteration of the particle filter a predict and an update step are performed. The predict step models the effects of the control inputs u_t on each particle of S_{t-1} by sampling from the motion model distribution. The referenced motion model is identical to the dynamic model applied by the USARSim.

The particle weights are computed in the second step. In this phase, a measurement model is used to evaluate the weight of each particle based on sensors information. This weight is updated by the likelihood of the sensor measurements z_t given the prediction $\bar{x}_t$ and the map m.

$$p(z_t|m, \bar{x}_t) \tag{6}$$

The weight of each particle results from the joint likelihood of all measurements, given the map and the path. These measurements are given by (1) sonars and (2) the altimeter. The sonar measurements are modeled with a Gaussian distribution:

$$P(z_t^i|m^{[n]}, \bar{x}_t^{[n]}) = \frac{1}{\sigma_{dist}\sqrt{2\pi}} e^{-\left(z_t^i - \bar{d}_t^{[n]}\right)^2 / 2\sigma_{dist}^2} \tag{7}$$

where z_t^i stands for measurement of sonar i, m for the map and $\bar{d}_t^{[n]}$ is the Euclidean distance between position hypothesis $\bar{x}_t^{[n]}$ and the first occupied cell in the map of the n-th particle. Note that equation (7) is applied for each sonar.

The altimeter measurements are modeled with another Gaussian

$$P(h_t|m^{[n]}, \bar{x}_t^{[n]}) = \frac{1}{\sigma_{alt}\sqrt{2\pi}} e^{-\left(h_t - \bar{Z}_t^{[n]}\right)^2 / 2\sigma_{alt}^2} \tag{8}$$

where h_t is the altimeter measurement. By using the altimeter readings it is possible to significantly reduce the uncertainty along the vertical axis. The final weight $w_t^{[n]}$ of each particle is equal to the multiplication of all involved likelihoods, assuming conditional independence given the robot position and map

$$w_t^{[n]} = P(h_t|m^{[n]}, \bar{x}_t^{[n]}) \cdot \prod_i P(z_t^i|m^{[n]}, \bar{x}_t^{[n]}) \tag{9}$$

In order to determine a single position estimation we choose the particle with the highest weight from the set

$$\hat{x}_t = \bar{x}_t^{[n^*]}, \qquad \text{where} \quad n^* = \underset{i \in \{1,...,N\}}{\arg\max}\ w_t^{[i]} \tag{10}$$

2.2. Decision Block

The inputs for the Decision Block are the position estimation $\hat{x}_t$ and a binarized version of the map m. The threshold adopted for the binarization is $p(m_{occupied}^i) < 0.5$ in order to consider the never observed cells as occupied. By analysis of the map and the vehicle estimated velocity it is possible to predict how far away the object is and how long it takes — if we maintain the current speed — to collide with it. This concept is known as Time To Collision (TTC) and is a crucial step in the classification of the danger levels. The global flow chart of the Decision Block is presented in Figure 2.

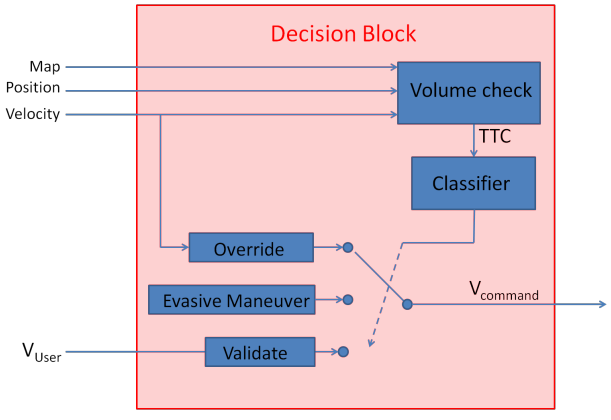

Fig. 2. Flow chart of the Decision Block.

To successfully avoid crashes, the position of obstacles has to be known as well as the direction to which the robot is flying. The volume check corresponds to the extrusion of a square centered on the quadcopter's position, along the velocity vector, as illustrated in Figure 3. The size of this square, b, encompasses the quadrotor volume while a is a visibility bound. Defining this volume enables the algorithm to find which is the position of the closest occupied cell.

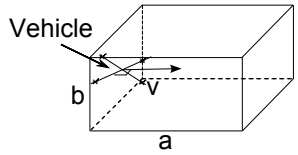

Fig. 3. Graphical definition of the volume check

Upon having the velocity estimation and the distance to the closest cell it is easy to compute the TTC and use it as a danger assessment. The classifier block acts as a multiplexer by choosing an input, and forwarding it to the vehicle, given a certain TTC. The threat levels, together with the contingency actions considered, are presented in Figure 4 and explained below.

– **No Action**: For the given TTC the algorithm considers that no threat exists and no action is performed on the inputs, meaning that the vehicle is fully controlled by the user.

– **Slow**: If the TTC falls into the given interval the quadcopter is considered to be in medium danger and user's inputs will be limited. In order for the vehicle to increase its TTC to a safe value, thus causing the decision block leave the Slow threat level, the velocity applied to the vehicle, denoted by $v_{command}$, is

$$v_{command} = \frac{d_{obstacle}}{TTC_{TH_{SLOW}}} \tag{11}$$

where $d_{obstacle}$ represents the distance between the vehicle and the obstacle, and $TTC_{TH_{SLOW}}$ the threshold between the Slow and No action stage — Figure 4.

– **Stop**: At this level, threat is considered to be high and velocity is immediately set to zero making the vehicle hover.

– **Evasive Maneuver**: At this level the threat is considered to be extremely high and the solution to avoid collision in this situation is to give a velocity in the opposite direction of the current movement. By doing so, the distance while decelerating is much lower than in the STOP stage.

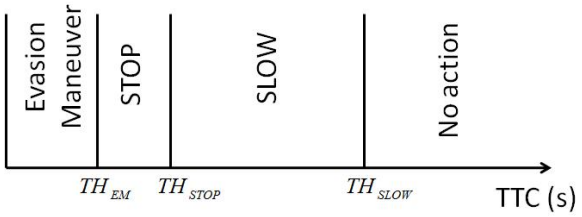

Fig. 4. Threat levels

Consider as an example that the vehicle is ordered to move towards an unknown part of the map. Since no information regarding those cells is yet known, the algorithm will consider them as occupied for safety reasons. Therefore, the above mentioned levels are equally applied. A problem occurs whenever the attitude of the vehicle is higher than half the field of view of the sonar, i. e., the sonar will not be able to check whether the volume in the desired direction is occupied or not as it is not pointing there. By direct application of the computation and selection of the danger level, it is possible to conclude that a Slow order is given and the velocity will lower. The sonar will then point back towards the direction of the movement, due to velocity reduction, and the distance to the object is updated. The algorithm will cause the vehicle to alternate between acceleration and deceleration in order to maintain knowledge of where it is heading. The emergence of this behavior can be seen in our experiments.

3. Results

The full architecture was implemented and tested in the USARSim environment. A total number of particles of 10, together with $\sigma_{dist} = 2$, $\sigma_{alt} = 2$, $FoV = 30°$, maximum sonar range equal to 5 meters and a square cell with length equal to 1 meter was considered. The examples inputs are the user's commands, attitude and sonars readings. Neither the map nor the position is provided by the simulator. Note that the FastSLAM and the decision block are running online during real time simulation on USARSim.

In the first example the robot was ordered a full speed movement towards a wall out of sonars range. As it is observable from Figure 5, the vehicle is, up until around 7 seconds, consecutively lowering and increasing its distance to the obstacle. Since this distance keeps oscillating it means that no real obstacle is there but, as it is the first time the area is mapped, the algorithm is updating the cells to free while moving. After 7 seconds the distance starts to lower meaning a real obstacle was found.

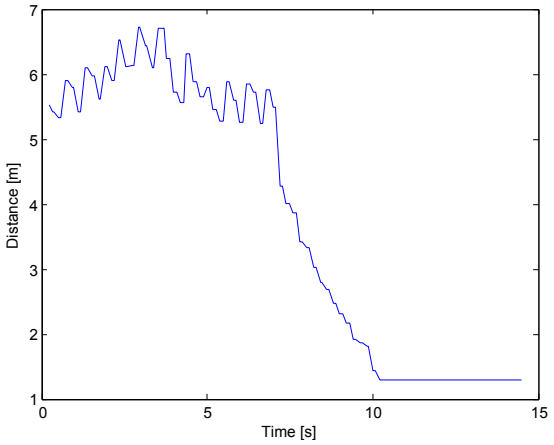

Fig. 5. Distance between the best particle to the first obstacle

Figure 6 shows the estimated velocity of the vehicle along time. Note that the velocity is also oscillating. The user is ordering a full speed movement ($5m/s$) but the classifier applies a Slow override. When the obstacle is sensed to be further away from the robot, the inputs are once again given to the user. When the algorithm perceives that the vehicle is moving towards an obstacle, once again after 7 seconds, an override is imposed and the velocity fully limited by the classifier and lower till zero. By fusing the information from both figures it is possible to see that the vehicle stopped its movement at, approximately, 1.3 meters from the wall despite being constantly ordered by

the user to move in that direction. If, at this point, the robot were ordered to fly in the opposite direction the vehicle would fly at full speed since those cells are known to be free according to the map.

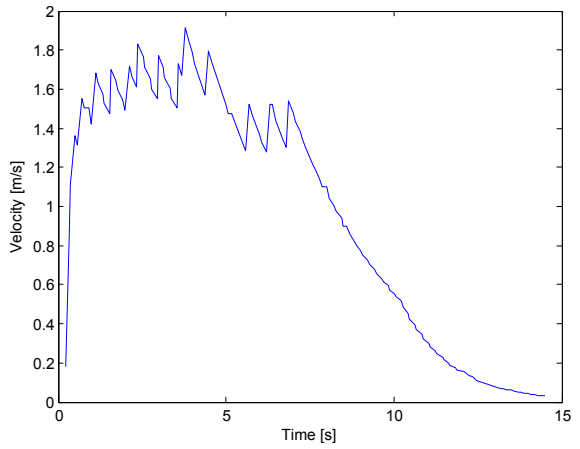

Fig. 6. Estimated velocity of the quadrotor corresponding to Fig. 5

In the second example the trajectory performed has three distinct phases: the robot was initialized far from a wall and a velocity imposed towards it; a movement along the wall; a separation and re-approximation to the wall. With this experiment the main benefit of our approach facing a purely reactive method is shown. That distinction is proven useful in the final part of the movement where the vehicle is expected to keep a memory of the obstacle previously seen. All movements were performed at full speed. Results are presented in Figures 7 and 8.

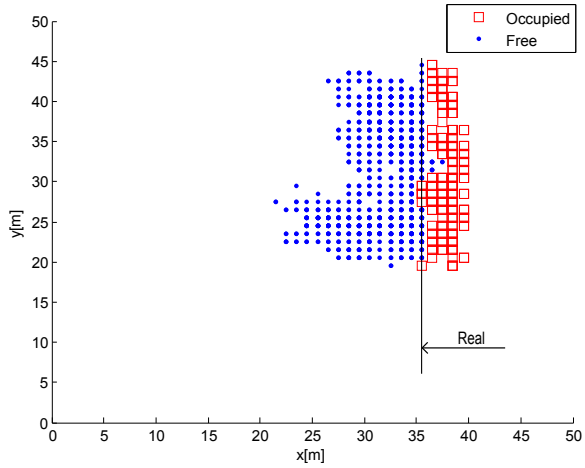

Fig. 7. XY cut of the 3D map of the best particle at the end of the movement. The Real labeled arrow indicates the limit of the groundthruth wall location

During this experiment, the algorithm faces two near collision situations. In both cases, the program managed to fulfill its objective and avoid the crash despite the orders from the user to continue its trajectory towards the wall. Although both collisions were avoided a major difference arises between them. The second time the algorithm was moving towards the wall it managed to avoid the crash at a higher distance from the wall. After sensing the obstacle

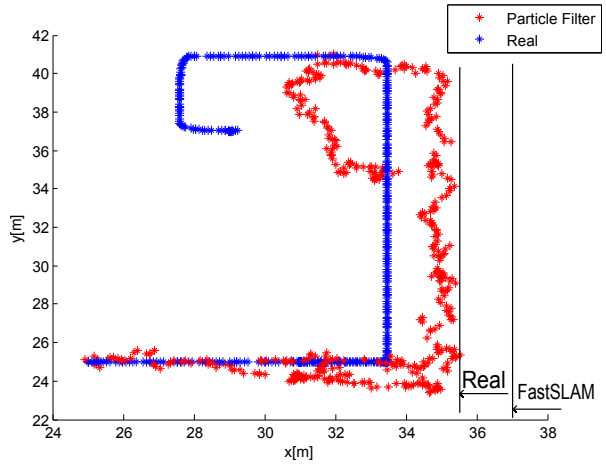

Fig. 8. Localization of the best particle at the end of the movement. The arrows represents the limits of the groundthruth wall and the obtained map.

once, it is able to keep a localization in relation to it and it is also able to prevent crashes more effectively.

It is noticeable that, despite having drift away from the real position, the algorithm managed to build a map, Figure 7, according to the belief of its position and localize itself on it. Since our goal is to localize the robot relatively to our map, we argue that the algorithm's performance is not directly compromised by the error between the real and the estimated position.

The obtained results match our expectations and can be seen as a proof of concept towards a real world implementation.

4. Conclusions

The main objective of this work is to develop an assisting tele-operation algorithm for quadcopters with the purpose of avoiding collisions with obstacles. The desired solution considered as head objective the safety of the quadcopter even in situations where user's commands were contrary. Whenever confronted with an unknown area, the algorithm overrides the inputs in order to reduce uncertainty. The main objective was successfully achieved in simulation using a FastSLAM approach for simultaneous localization and mapping combined with a danger classification methodology, in order to classify and act correspondingly in any situations. As for future work, we are currently working at a real world implementation of the proposed method.

AUTHORS

João Mendes – Institute for Systems and Robotics, Instituto Superior Técnico, Portugal, e-mail: mendes.joao.p at gmail.com

Rodrigo Ventura – Institute for Systems and Robotics, Instituto Superior Técnico, Portugal, e-mail: rodrigo.ventura at isr.ist.utl.pt

References

[1] S. Bouabdallah, Design and Control of an Indoor Micro Quadrotor. Proceedings of International Con-

ference on Robotics and Automation, 2004.

[2] A. Stentz, D. Fox and M. Montemerlo, FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association. Proceedings of the AAAI National Conference on Artificial Intelligence, 2003.

[3] A. Elfes, Occupancy grids: A probabilistic framework for robot perception and navigation. PhD thesis, Carnegie Mellon University, 1989

[4] C. Stachniss, D. Hähnell, W. Burgard, G. Grisetti, On Actively Closing Loops in Grid-based FastSLAM. Advanced Robotics, 2005.

[5] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. Proceedings of the AAAI National Conference on Artificial Intelligence, 2002.

[6] Z. He, R. Iyer, P. Chandler, Vision-Based UAV Flight Control and Obstacle Avoidance. Proceedings of the American Control Conference, 2006.

[7] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, 2005

[8] N. Gordon, D. Salmond, A. Smith, Radar and Signal Processing. IEEE Proceedings,1993