Safe Teleoperation of a Quadrotor Using FastSLAM

João Mendes and Rodrigo Ventura
Institute for Systems and Robotics
Instituto Superior Técnico
Av. Rovisco Pais, 1; Lisbon, PORTUGAL
jmendes@isr.ist.utl.pt, rodrigo.ventura@isr.ist.utl.pt

Abstract — Unmanned Aerial Vehicles (UAV) provide many advantages in Search and Rescue (SaR) scenarios, such as the capacity for remote inspection over areas that are difficult to reach by ground vehicles. Moreover, it can carry small payloads, such as first aid equipment, over large distances. However, the teleoperation of UAVs often demands extensive training, since 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 teleoperation of a quadrotor using an obstacle avoidance approach. The target scenario is SaR operation in unknown, unstructured, GPS-denied environments, such as warehouses or other buildings. A short-term 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 (1) override operator commands that may lead to a collision, and (2) perform evasive maneuvers whenever collision is imminent. A simple active perception routine is used to orient one of the sensors to an unknown area, in case the UAV is ordered to move towards an unmapped area. Experimental results using the USARsim simulator are presented. Further testing was conducted in a real quadcopter, allowing a preliminary validation of the proposed methods.

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

I. Introduction

Situation assessment is a key component on any Search and Rescue (SaR) operation. As human lives might be at risk, an efficient and effective assessment of the situation is paramount. However, in many such situations, the target areas are often difficult to reach by human teams, either because they present a risk to human presence (e.g., a building close to collapse), or because they are physically unreachable in due time. Therefore, the usage of robots present an interesting potential to perform inspection operations without risk to humans.

Among the variety of existing robots, Unmanned Aerial Vehicles (UAV) have several characteristics that make them well fit for SaR operations. In particular, since they do not rely on a terrain to move, it is possible to cover large areas airborne. Remote inspection of large areas can be carried out by using cameras onboard the UAV. However, the teleoperation of UAV, particularly in confined environments, is far from trivial. It usually demands extensive training and expertise to be able to safely teleoperate such vehicles.

In this paper we address the problem of *Assisting Teleoperation* to safely pilot a UAV. We define assisted teleoperation of a UAV as the process of overriding the operator input, either by modulation, inhibition, or replacement with a different input. Our goal is to use sensor data to determine an appropriate assisted teleoperation in order to guarantee, to the best of sensor capabilities, a safe flight.

The UAV targeted here are quadcopters: 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]. Its high agility, small size and VTOL capacity make quadcopters a powerful tool not only for situations where hovering is crucial but also for indoor usage.

We focus on unknwon, unstructured, GPS-denied, and indoor environments. We assume that the main threat to the UAV safety is collision with obstacles. Thus, distance sensors are used to detect and map these obstacles. The approach is based on the FastSLAM [2] algorithm, together with 3D occupancy grid mapping [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 based on danger assessment, performed by a classifier, is applied. This classifier overrides the user's inputs if they compromise the quadcopter's physical integrity in the near future. Overriding may extend from simple velocity reduction to, in extreme cases, an evasive maneuver. As our aim is to ensure the vehicle's safety at all times, an active perception methodology is applied to address map uncertainty whenever it is necessary. 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.

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 [1] or laser scan matching [2]. 6D SLAM resorting to vision is yet a different approach widely used by the community [3]. Different SLAM approaches resorting to particle filters with an EKF for mapping [4] can be found in the literature, but applied in wheeled robots. As for obstacle avoidance methodologies for UAVs, literature mostly addresses for path

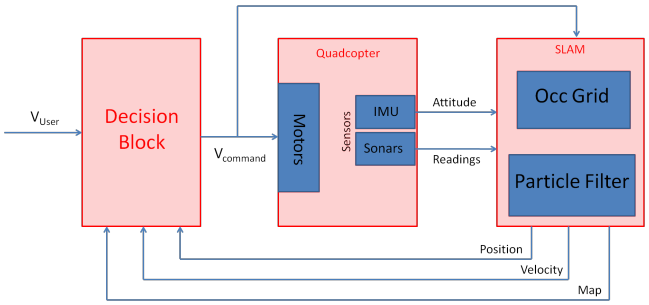

Fig. 1. Full architecture of the proposed approach

replanning topics [5] or to vision based solutions [6] [7], while [8] addresses the problem of collision avoidance for multiple robot systems. This paper differs from the above in that building an accurate map is not our objective. Instead, 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. Simulations were performed using USARSim simulator.

II. Methodology

A. FastSLAM

Correct attitude is assumed to be given at all times by the onboord 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 hereby solved by the FastSLAM approach proposed by Montermerlo et al. [4]. 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 on each particle 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.

$$p(z_t|m, \bar{x}_t) \tag{1}$$

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:

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

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 (2) is applied for each sonar.

The altimeter measurements are modeled with another Gaussian

$$P(h_t|m^{[n]}, \bar{x}_t^{[n]}) = \frac{1}{\sigma_{alt}\sqrt{2\pi}} e^{-\left(h_t - \bar{Z}_t^{[n]}\right)^2 / 2\sigma_{alt}^2} \tag{3}$$

where h_t is the altimeter measurement. By using the altimeter readings efficiently 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

$$w_t^{[n]} = P(h_t|m^{[n]}, \bar{x}_t^{[n]}) \cdot \prod_i P(z_t^i|m^{[n]}, \bar{x}_t^{[n]}) \tag{4}$$

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

$$\hat{x}_t = \bar{x}_t^{[n^*]}, \qquad \text{where} \quad n^* = \underset{i \in \{1,\ldots,N\}}{\arg\max}\ w_t^{[i]} \tag{5}$$

B. Decision Block

The inputs for the Decision Block are the position estimation $\hat{x}_t$ and a map m. Each cell i is classified in one of 3 states, $c^i \in \{F, U, O\}$, corresponding to Free, Unknown, Occupied. The classifier is based on thresholding the occupancy probablities, according to:

$$c^i = \begin{cases} F & \text{if } P(m^i_{occupied}) < 0.5 \\ U & \text{if } P(m^i_{occupied}) = 0.5 \\ O & \text{if } P(m^i_{occupied}) > 0.5 \end{cases} \tag{6}$$

All cells are initialized with $P(m^i_{occupied}) = 0.5$ thus classified as Unknown. After being mapped, they become classified as either Free or Occupied, apart from the extremely unlikely event of being numerically equal to 0.5.

The global flow chart of the Decision Block is presented in Figure 2. When the user inputs a desired velocity to the quadcopter, the Active Perception block validates if the commands will be applied to the vehicle without any constraint. To do so, these inputs are firstly used to compute the desired direction for the vehicle and the distance to the first non-Free cell. If the closest cell is Occupied, the inputs are subject to a confirmation that they do not compromise the vehicle's safety in a near future and then applied to the vehicle. If the closest cell is Unknown and if the distance to it is higher than a certain threshold (TH) the algorithm checks whether there is any sonar aligned with the desired direction. Otherwise, the algorithm will autonomously rotate the vehicle and then apply the same speed the user demanded. By doing so, the algorithm is able to avoid flying into unknown areas as the sonar is, after this active perception routine, pointing towards the velocity direction while still performing the movement the user requested. The flowchart of the Active Perception Block is presented in Figure 3. With the application of the Active

Perception it is possible to guarantee that the vehicle is not allowed to fly towards unknown areas and, therefore, the cells in the future path of the robot are classified only as Occupied or Free.

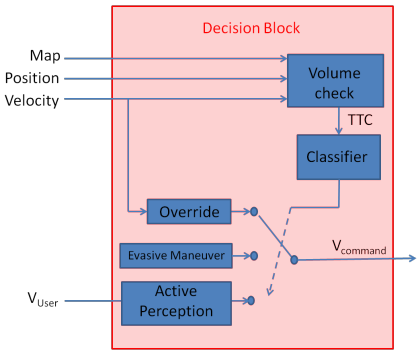

Fig. 2. Flow chart of the Decision Block.

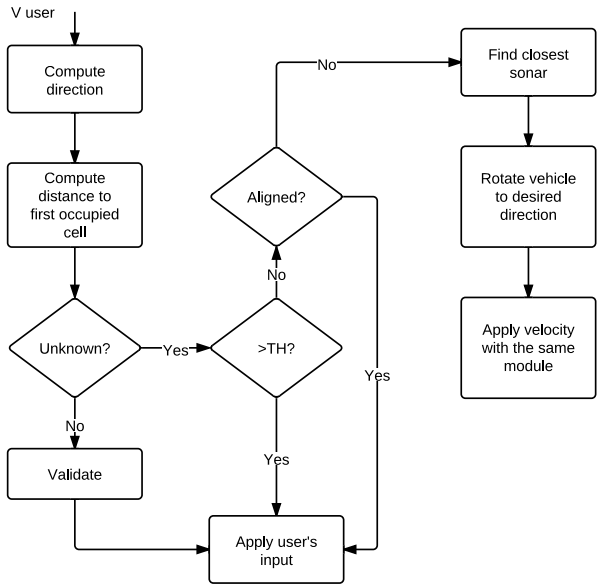

Fig. 3. Flow chart of the Active Perception block.

To successfully avoid collisions, the position of obstacles has to be known as well as the direction to which the robot is currently flying. The *volume check* corresponds to the extrusion of a square centered on the quadcopter's position, along the velocity vector (obtained from the predict step of the particle filter), as illustrated in Figure 4. The size of this square, b, encompasses the quadrotor volume while a is a visibility bound. Defining this volume enables the algorithm to know which grid cells are in the near-future path of the vehicle, find which is the position of the closest occupied cell, and compute the distance between that cell and the vehicle. It is then possible to predict 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 classifier block acts as a multiplexer by choosing

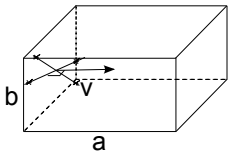

Fig. 4. Graphical definition of the volume check

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 5 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. Within this level there are two relevant cases; the first one takes place if there are no objects inside the volume checked; the second occurs when the TTC is high enough for the algorithm to consider that the robot is not in danger.
- **Slow**: If the TTC falls into the given interval the quadcopter is considered to be in medium danger and the user inputs will be limited. The vehicle's current velocity is multiplied by a factor $k(TTC)$, where $k \in [0, 1[$, making $v_{command} = k(TTC) \cdot V_{current}$ where $v_{command}$ stands for the next velocity to be ordered to the vehicle. Since $k < 1$ it will slow down the vehicle until the TTC becomes greater than a safe value, thus causing the decision block to leave the Slow threat level.
- **Stop**: At this level, threat is considered to be high and vehicle is put in a hover position thus making it decelerate. Like in the previous level, this action will increase the TTC.
- **Evasive Maneuver**: At this level the threat is considered to be extremely high. If TTC falls within this interval, it means that if the vehicle is ordered to hover the distance traveled before stopping is superior to the distance to the object and crashing is inevitable. The solution to avoid the collision in this extreme situation is to give a velocity in the opposite direction of the current movement for a very brief period. By doing so, the distance while decelerating is much lower than in the STOP stage. This stage is applied for a short duration, just to guarantee that the vehicle does not hit the object.

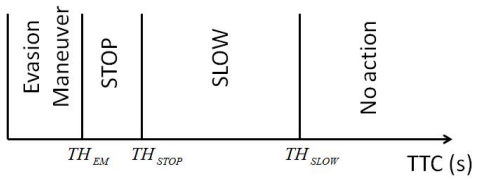

Fig. 5. Threat levels

III. Simulation 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 its sensor range. As shown in Figure 6, the distance to the "obstacle" is oscillating until about 7 seconds simulation time. This is due to the fact that Unknown cells trigger the active perception routine to orient the closest sonar towards the movement direction. When doing so, unmapped areas become Free, and the distance to obstacle increases. This proceeds until the real obstacle is found, which makes triggers the Slow order until eventually stopping it.

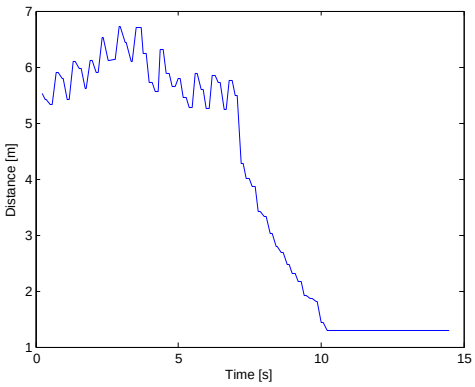

Fig. 6. Distance between the best particle and the first obstacle

In Figure 7, the estimated velocity of the vehicle along time is shown. 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.

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 8 and 9.

During this experiment, the algorithm faces two near collision situations. In both cases, the program managed to fulfill its objective and avoid crashing 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 collision at a higher distance from the wall. After sensing the obstacle once, it is able to keep a localization in relation to it and it is also able to prevent crashes more effectively.

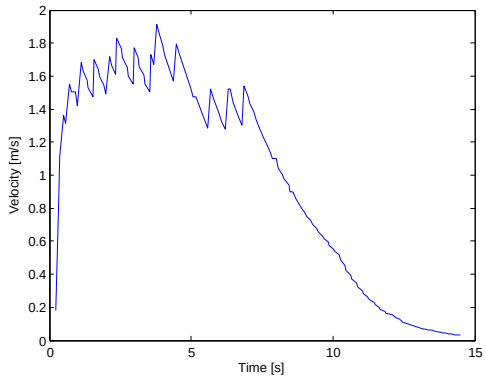

Fig. 7. Estimated velocity of the quadrotor corresponding to Fig. 6

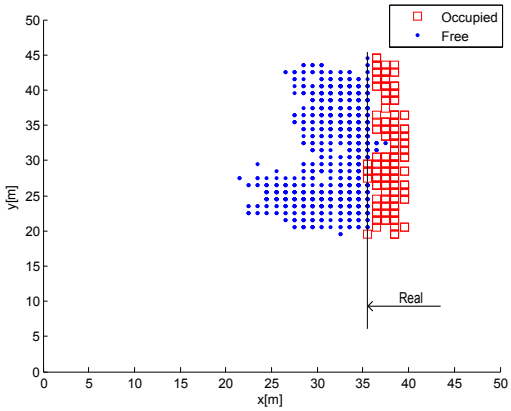

Fig. 8. XY cut of the 3D map of the best particle at the end of the movement. The arrow indicates the limit of the groundthruth wall location

Note that, despite having drift away from the real position, the algorithm managed to build a map, Figure 8, 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.

IV. Real world results

In this section, preliminary results obtained in several real world experiments are presented. The available quadrotor is equipped with an IMU capable of providing filtered values of yaw, pitch and roll. The communication between the quadrotor and the computer — for all telemetry parameters — is supported by a Xbee module. Unlike in simulation, only 4 sonars were used and placed above each of the propellers. The lobe was modeled by a 30° FoV cone and maximum range considered was three meters. An Arduino was used to perform

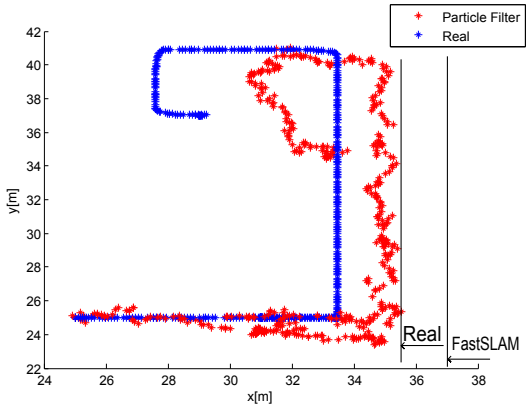

Fig. 9. Localization of the best particle at the end of the movement. The arrows represents the limits of the groundthruth wall and the obtained map.

the readings and all necessary transformations to meters. All readings were recorded with the quadrotor's motors off, for safety issues, and all experiments were performed in offline mode. The quadrotor was manually placed in the different poses throughout the tests.

A. *FastSLAM*

In this experiment the FastSLAM, combining localization and mapping, was applied. The particles are all initialized in the same point since it does not compromise the results. No initial map is provided nor any information concerning the localization. The total number of particles used was 10. No knowledge about the motion model is yet experimentally acquired and, therefore, the predict set of the particle filter becomes (7) where ω and Q_t are defined in (8) with $\sigma_x = \sigma_y = 0.3$ and $\sigma_z = 0$.

$$\bar{x}_t^{[n]} = \tilde{x}_{t-1}^{[n]} + \omega \tag{7}$$

$$\omega = N(0, Q_t), \qquad Q_t = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \tag{8}$$

The z axis is limited as no information is given about the height of the vehicle — no sonars pointing top or bottom nor altimeter readings — and, as a consequence, the weights of the particles in the observation model become only dependent on sonar readings (9).

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

For better comparison, in Figure 10 is presented the map obtained with ground truth positions. The sonars' values used were the same for both situations but, in the FastSLAM case, no localization was provided. The points where sonar values were recorded are presented in red. The trajectory was performed from the right side of the image to the left. In Figure 11 is presented the best particle's map at the end of the movement.

The algorithm is being able to successfully map the environment while localizing itself. Although the obtained map

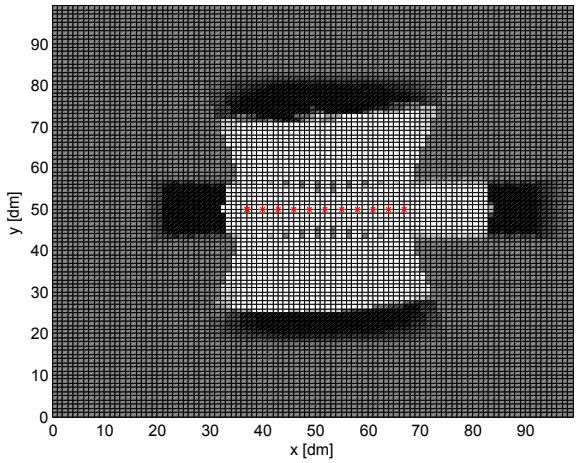

Fig. 10. xy cut of the Map obtained with GT localization at the end of the test.

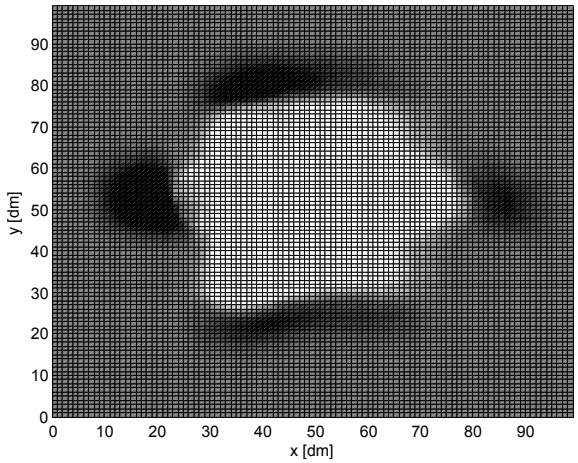

Fig. 11. xy cut of the best particle's map at the end of the test.

diverges from the ground truth, it is possible to check that the dimensions of the room were roughly preserved and a coarse object identification was performed. In Figure 12 is plotted the Euclidean error between the best particle and the real position along the positions.

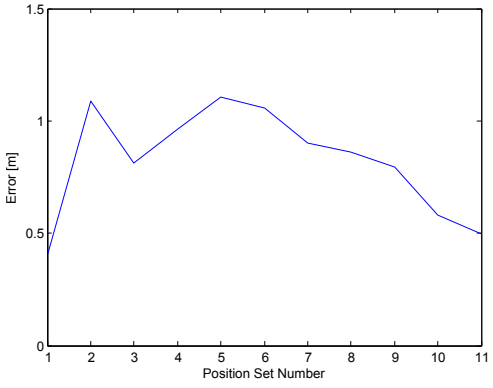

Fig. 12. Error obtained at the end of the 10^{th} iteration in each position. Each iteration corresponds to one run of the full FastSLAM algorithm.

Although the error experience in localization is substantially high — considering that the vehicle is operating indoor — no motion model is being used for now. The inclusion of one would significantly improve the particle filter performance and hence the map obtained. However, the existence of error to the real position is admissible as a localization in relation to the obstacles is intended in opposition to a global localization.

B. Decision Block

In this section, real world tests to an architecture combining the FastSLAM and the decision block are presented. The presented example considers 5 different stages illustrated in Figure 13. It is considered that the user is ordering a constant movement towards the wall through the entire test. With this set-up, we aim at a possible and common movement where a purely reactive methodology would not manage to detect the obstacle and consequently would loose its purpose. Due to the distance between positions, around 1 meter, a motion model was introduced. The difference between the real positions is, therefore, being fed to the predict model of the particle filter.

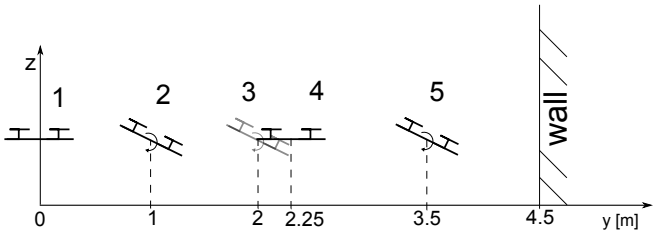

Fig. 13. Position and attitude in each of the five phases.

By analysis of Figure 13 it is possible to see that in position 1 the vehicle has $pitch = 0°$ and is able to map the area where it is going to be in position 2. The vehicle then moves to the previously mapped and now known to be free area. Since the distance to the nearest obstacle — in this case an unknown position — is still high enough (Table I) the algorithm allows the user to maintain control of the quadcopter. When the vehicle reaches position 3, the distance between its position estimation and the first obstacle in its own map will lower and the TTC decreases. As a consequence, the user inputs will be overriden by the Decision Block and the vehicle ordered to slow its movement. When the velocity is lowered — position 4 — the vehicle regains sight of its moving direction and re-updates the map. When the distance to the obstacle is updated, the algorithm allows the user to input commands once again. This situation can occur multiple times until the vehicle reaches a real obstacle.

In Table I the comparison between the distance from the center of mass of the vehicle to the closest obstacle in each position of Figure 13 is presented. The first line shows the distance reported by the reactive method while the second one shows the distance between the position estimation and the first obstacle — unknown or occupied area inside the volume defined in Figure 3 — in its own map. While the reactive method is unable to infer the distance between the vehicle and the obstacle in the direction of the movement, the proposed is solution keeps a constant track of this distance allowing the classifier to choose the best action in each situation. In the last two rows it is possible to see the classification C given to the closest non-free cell i and the consequent action performed on the command inputs.

TABLE I

COMPARISON BETWEEN THE DISTANCE TO THE FIRST OBSTACLE IN BOTH METHODS AND CONSEQUENT CLASSIFIER DECISION.

	1	2	3	4	5
$d_{reactive}$	3	–	–	2	–
$d_{mapping}$	3.25	2.34	1.38	2.28	1.11
C^i	U	U	U	O	O
Classifier decision	User	User	Override	User	Override

V. CONCLUSION

This paper presented an assisted teleoperation method for UAVs, based on short-term, rough mapping of the nearby environment. In particular, we targeted quadcopter vehicles operating in unstructured, GPS-denied, and dynamic environments, such as the ones often encountered in SaR scenarios. By mapping the environment, the method is capable of overriding user's inputs in order to guarantee a safe flight of the UAV. Whenever confronted with an unknown area, the active perception routine forces the vehicle to point a sensor towards that area. 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 in a full real world implementation of the proposed method.

ACKNOWLEDGMENT

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

REFERENCES

[1] C. Stachniss, D. Hhnel, W. Burgard, and G. Grisetti, "On actively closing loops in grid-based fastslam," *ADVANCED ROBOTICS*, vol. 19, p. 2005, 2005.

[2] A. Diosi and L. Kleeman, "Fast laser scan matching using polar coordinates," *I. J. Robotic Res.*, vol. 26, no. 10, pp. 1125–1153, 2007.

[3] J. Artieda, J. M. Sebastian, P. Campoy, J. F. Correa, I. F. Mondragón, C. Martínez, and M. Olivares, "Visual 3-d slam from uavs," *J. Intell. Robotics Syst.*, vol. 55, no. 4-5, pp. 299–321, Aug. 2009.

[4] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *In Proceedings of the AAAI National Conference on Artificial Intelligence*. AAAI, 2002, pp. 593–598.

[5] Z. He, R. V. Iyer, and P. R. Chandler, "Vision-based uav flight control and obstacle avoidance," *2006 American Control Conference*, pp. 2166–2170, 2006.

[6] G. L. Barrows, J. S. Chahl, and Y. V. Srinivasan, "Biomimetic visual sensing and flight control," in *Proc. Bristol UAV Conf*, 2002, pp. 159–168.

[7] O. Call, Y. Beard, C. Taylor, and B. Barber, "Obstacle avoidance for unmanned air vehicles using image feature tracking," in *AIAA Guidance, Navigation, and Control Conference*, 2006.

[8] N. Ayanian and V. Kumar, "Abstraction and control for groups of robots in environments with obstacles," in *International Conference on Robotics and Automation*, IEEE, Ed., 2010.