An elevation map from a micro aerial vehicle for Urban Search and Rescue

Nick Dijkshoorn and Arnoud Visser

UvA Rescue - RoboCup Rescue Simulation League
Universiteit van Amsterdam, Science Park 904, Amsterdam, The Netherlands

Keywords: Quadrotor, visual SLAM, elevation map

Abstract. The developments in unmanned aerial vehicles make it possible to use this platform on a much larger scale. The current challenge is to use a team of flying robots to explore a city block, place lookouts at strategic points and if possible to enter some of the buildings in the block, to search for hazards and/or people. This challenge is still quite ambitious, but allows researchers to explore some of the aspects in simulation. This paper describes how to build a visual map of the environment including height information. This is an essential step towards more extensive applications, both in simulation and for a real platform.

1 Introduction

Nowadays, small quadrotors with on-board stabilization like the Parrot AR.Drone can be bought off-the-shelf. These quadrotors make it possible to shift the research from basic control of the platform towards applications that make use of their versatile scouting capabilities. Possible applications are surveillance, inspection and search & rescue. Still, the limited sensor suite and the fast movements make it quite a challenge to fully automate the navigation for such platforms. One of the prerequisites for autonomous navigation is the capability to make a map of the environment.

Once such a map exists, a team of micro aerial vehicles could be used to explore an area like a city block. The map is needed to coordinate the actions between the team members. After a disaster one could not rely on prior satellite maps, part of the job of the rescue team is to do a situation assessment and an estimation of damage (roads blocked, buildings on fire, locations of victims visible from the sky).

In this paper is described how a visual map can be build. This visual map consists of a feature map which is built based on storing the most distinguishable SURF features on a grid. This map can be used to estimate the movement from the AR.Drone on visual clues only, as described in previous work [1]. In this paper the focus on an extension of the previous method; an experimental method [2] to create an elevation map by combining the feature map with ultrasound measurements. This elevation map is combined with textures stored on a canvas and visualized in real time. An elevation map is a valuable asset when the AR.Drone has to explore unstructured terrain, which is typically the case after a disaster (an Urban Search and Rescue scenario).

The paper proceeds as follows. In Section 2 we give a short overview of related work. Section 3 describes our method to build a feature map. More details how this feature map can be used to localize the AR.Drone can be found in [1]. In this paper the 2D feature map is extended with a method to build an elevation map based on sonar measurements. Experiments and results for both the feature map and elevation map are presented in Section 4. Conclusions and future work are covered in Section 5 and 6.

2 Related work

Our approach was inspired by Steder *et al.* [3], who presented a system that allows aerial vehicles to acquire visual maps of large environments using comparable setup with an inertial sensor and a low-quality camera pointing downward. In their approach the inertial sensor was used to estimate a number of parameters in the spatial relation between two camera poses, which reduces the dimensionality of the pose to be estimated.

Equivalent with our approach, Steder uses Speeded-Up Robust Features (SURF) [4] that are invariant with respect to rotation and scale. By matching features between different images, one can estimate the relative motion of the camera and construct the graph that serves as input to the TORO-based network optimizer [5].

While Steder uses a Euclidean distance measure to compare the correspondences of the features in two images, Caballero *et al.* [6] indicate that this is a last resort. They are able to make a robust estimation of the spatial relationship on different levels: homogeneous, affine and Euclidean. First an attempt is made to estimate complete homogeneous transformation (8 degrees of freedom). When the number of correspondences is too low an affine transformation (6 degrees of freedom) is estimated. Only when the features in the image are really noisy a Euclidean transformation (4 degrees of freedom) is estimated. Because the AR.Drone is a quite recent development, the number of studies based on this platform is limited. A recent publication is from Cornell University [7], where an AR.Drone is used to automatically navigate corridors and staircases based on visual clues.

3 Methods

First the method to build a visual map is explained, followed by a section how this map is extended with height measurements.

3.1 Visual mapping

Feature map For human navigation a texture map is essential. Such a texture map is built by using a pose estimate to calculate 2D world coordinates for the pixel corners (rays), and uses this information to compute a perspective transformation. This transformation is used to warp the image on the canvas, as described in [1]. Yet, for a robot a feature map is more useful. Such a feature map can be built with a grid of image features (feature map). The inertia measurements provide frequent velocity and acceleration estimates, which can be aggregated to position estimates. However, drift will decrease the accuracy of this position estimate over time. If the AR.Drone is able to relate a video frame to a position inside the feature map, the vehicle is able to correct this drift long enough to build a map as large as an indoor gym (as used in the experiments of Section 5).

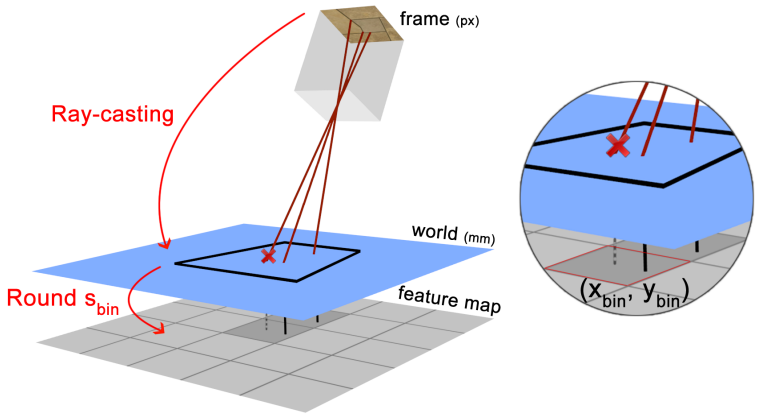


Fig. 1: Feature map: adding visual features to a feature grid. Each feature found in the camera frame is mapped to the corresponding 2D world coordinates. The 2D world coordinates are mapped to the grid cell the feature belongs to. For each cell, only the best feature is stored.

For the gym a 2D grid with a fixed resolution of $100 \times 100mm$ per cell is used. From each camera frame we extract Speeded-Up Robust Features (SURF) [4] that are invariant with respect to rotation and

scale. Each feature is an abstract description of an "interesting" part of an image (e.g., corners). A feature is described by a center point in image sub-pixel coordinates and a descriptor vector that consists of 64 floats.

Each feature that is detected in a camera frame is mapped to the corresponding cell of the feature map. A feature's center point (x_f, y_f) is transformed to its corresponding position in 2D world coordinates (x_w, y_w), visible in the top of Figure 1. This is done by casting a ray from the features pixel coordinates in the frame, similar to the method used to create the texture map [1]. Finally, the 2D world position (x_w, y_w) of each feature is transformed to the corresponding cell indices (x_{bin}, y_{bin}), visible in the bottom of Figure 1, by applying the follow equation:

$$\begin{bmatrix} x_{bin} \\ y_{bin} \end{bmatrix} = Round(s_{bin} \cdot \begin{bmatrix} x_w \\ y_w \end{bmatrix}) \tag{1}$$

where $s_{bin} = 0.01$.

For each cell, only the best feature (e.g. with the highest response) is kept and the other features are dropped. If a cell already contains a feature descriptor ($grid_{x,y} \neq \emptyset$), the cell is ignored.

$$grid_{x,y} = \begin{cases} \arg\max_{d \in D_{x,y}} response(d) & \text{if } grid_{x,y} = \emptyset \\ grid_{x,y} & \text{else} \end{cases} \tag{2}$$

where $D_{x,y}$ is the set of features that is mapped to cell x_{bin}, y_{bin}.

The remaining (best) feature descriptors, including the corresponding world coordinates (x_w, y_w), are added to the corresponding grid cells. This feature map can be used for localization purposes, as extensively described in [1]. In this paper the focus is on the extension of this feature map with height measurements, as described in the next section.

4 Elevation mapping using an ultrasound sensor

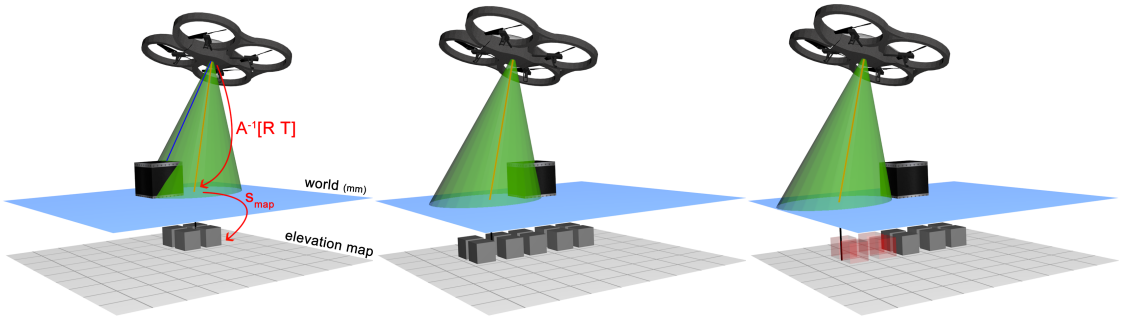
An elevation map can be used to improve navigation capabilities of both aerial and ground robots. For example, ground robots can use elevation information to plan routes that avoid obstacles. To build such an elevation map, the altitude z of the vehicle should be estimated. When a MAV is flying above a perfectly flat floor, the measured altitude z_{sensor} mean should be close the true altitude z_{true} (white noise assumption). However, when an obstacle comes in range of the ultrasound sensor, an edge effect becomes visible. The measured distance z_{sensor} decreases in advance and is not equal to the true altitude z_{true} directly under the vehicle, meaning that z_{true} cannot be derived from the ultrasound sensor directly. Sound propagates in a cone-like manner. Due to this property, the sensor acquires entire regions of constant depth instead of discrete depth points. So, the ultrasound sensor can only tell there is an elevation at the measured distance somewhere within the measured cone. This makes it hard to accurately determine the contours/borders of obstacles.

The elevation information is stored in a grid that is similar to the feature map described in Section 3.1. For each ultrasound distance measurement, elevation δ_t is computed and stored in the grid cell that corresponds to the world coordinates where a line perpendicular to the AR.Drone body intersects the world plane. These world coordinates are the position where the center of the ultrasound sensor's cone hits the floor. Because the exact size of an object is unknown, the elevation is written to all grid cells within a radius $\gamma_{elevationRadius}$ around the intersection point. This process is visualized in Figure 2a and 2b.

This approach may lead to cases where the size of an obstacle is overestimated in the elevation map, as can be seen in Figure 2b. Therefore, an additional refinement step was added to the elevation mapping methods. If no elevation is measured ($\delta_t \approx 0$), it can be assumed there is no obstacle inside the cone of the ultrasound sensor. Using this assumption, all grid cells within the cone can be reset to zero elevation and locked to prevent future changes. This refinement step is visualized in Figure 2c. The radius of the cone is computed using the following equation:

$$r = tan(\alpha_{ultrasound} \times z_{sensor}) \tag{3}$$

where r is the radius of a cone of height z_{sensor} and $\alpha_{ultrasound}$ is the opening angle of the ultrasound sensor.



(a) Obstacle enters range (b) Obstacle in range (c) Obstacle out of range

Fig. 2: Overview of the elevation map updates. The green cone indicates the range of the ultrasound sensor. The red line inside the cone represents the center of the cone, perpendicular to the AR.Drone body. In 2a and 2b an elevation is measured. All grid cells within a radius $\gamma_{elevationRadius}$ around the center of the cone (red line) are updated to store the measured elevation. 2c Describes the refinement step. When no elevation is measured, all grid cells within the cone (red cubes) are reset to zero elevation.

5 Results

5.1 Mapping

An experiment has been carried out to evaluate the mapping approach. This experiment is performed both with the AR.Drone and a simulated AR.Drone. The AR.Drone simulation model, including a sensor and motion model, is extensively described in [1]. Special care has been taken to reproduce the circumstances for the image processing as good as possible; the camera images in simulation are post-processed (decreased saturation, increased brightness, down-sampled resolution, variations in the white balance) to mimic the behavior of the AR.Drone's camera as close as possible.

Fig. 3: 3D model of a gym (left) with realistic ground and wall textures which represents the circumstances during the Micro Air Vehicle Pylon challenge (right).

A sports hall is used as indoor testing environment. This environment closely resembles the environment of the IMAV2011 indoor competitions (see Figure 3). Six magazines are laid out on the floor at the locations A, B, C, D, F, G in Figure 4. These magazines are used as groundtruth markers that are easily recognizable in the map. The distances between these markers are measured before the experiment was performed, providing ground truth. The AR.Drone flew in a 8-shape above the floor to capture an intermediate loop (point A and E in Figure 4). In addition to the landmarks, the lines on the floor provide visual clues about the overall quality of a created map. Care has been taken to mimic the gym floor in the simulated environment.

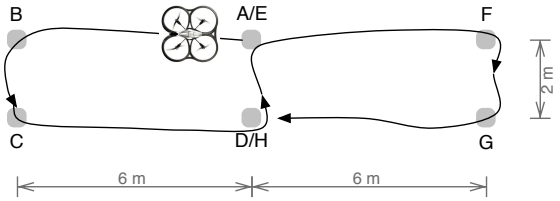


Fig. 4: Top view of the experiment's floorplan. The AR.Drone flew an 8-shape. The labels A-G represent six landmarks that provide ground truth to evaluate the accuracy of a map.

The mapping method is performed on the floor as described in Section 3.1. The distances between the landmarks inside the generated map (red lines) are compared to the know ground truth to compute the error of the map.

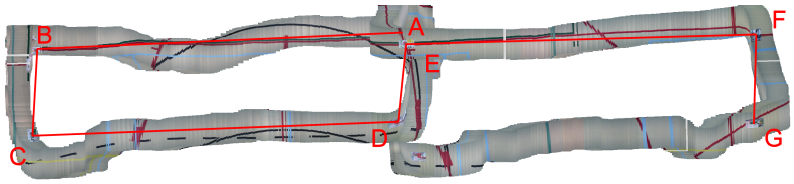


Fig. 5: Visual map created by the mapping method. Camera images are taken by the AR.Drone, flying at approximately 70cm altitude.

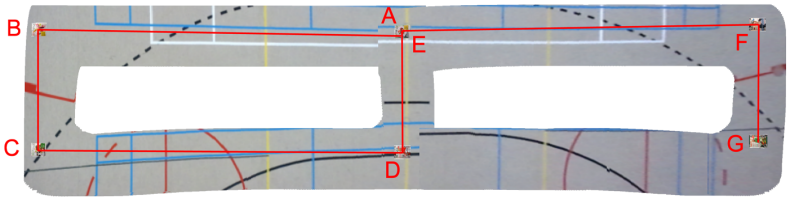


Fig. 6: Visual map created by the mapping method. Camera images are taken by the simulated AR.Drone, flying at approximately 85cm altitude.

The results of this experiment can be found in Table 1 and Figures 5 (real AR.Drone) and 6 (simulated AR.Drone). Both the simulated and real AR.Drone produce a map with enough quality for navigation in the IMAV Pylon challenge. The visual map could be further optimized by a method equivalent to TORO [5]. The visual map created by the simulated AR.Drone contains fewer errors than the map of the real AR.Drone. This is partly due to the optimistic noise level of the simulator's inertia measurements. Still, there are small errors visible in the map, which have the same origin for both the simulated and real AR.Drone. These errors are due to integrating the acceleration measurements to velocities and positions. The largest errors are in the long stretches of the trajectory, because these parts of the trajectory are longer affected by integration errors due to the larger traveling times.

The position of the real AR.Drone was estimated using the AR.Drone's velocity estimates produced by the proprietary onboard filter. This estimate is based on inertia measurements and optical flow obtained from the relative motion between camera frames. These measurements suffer from noise, resulting in larger map errors. The AR.Drone's proprietary optical flow algorithm is probably having difficulties with the gym floor. When flying along a single line or parallel lines (e.g. path B-C in Figure 5) no apparent motion can be observed from the camera frames. These circumstances mainly occurred when flying along the short

landmarks	A-B	B-C	C-D	D-E	E-F	F-G	B-G	C-F
real AR.Drone								
mean error (m)	0.042	0.653	0.087	0.726	0.350	0.579	0.491	0.388
error (%)	0.71	32.65	1.44	36.28	5.84	28.95	4.03	3.19
simulated AR.Drone								
mean error (m)	0.280	0.095	0.319	0.010	0.172	0.021	4.566	0.496
error (%)	4.66	4.74	5.31	4.98	2.85	1.1	3.75	4.1

Table 1: Accuracy of the visual map by measuring the distance between landmarks. The maps are created using the AR.Drone (Figure 5) and the simulated AR.Drone (Figure 6) .

stretches in the trajectory, producing the largest errors in this direction. This artifact is not observed for the simulated AR.Drone, because it lacks an optical flow method to estimate the velocity.

5.2 Elevation Map

Another experiment has been carried out to evaluate the elevation mapping approach. As shown in Fig. 7, the AR.Drone is able to estimate the height and length of two obstacles (a grey and blue box). The brown box is not detected due its limited height. Therefore, the measured (ultrasound) acceleration is insufficient to trigger an elevation event. As expected, the ramp was not detected. The gradual elevation change does not produce a significant acceleration.

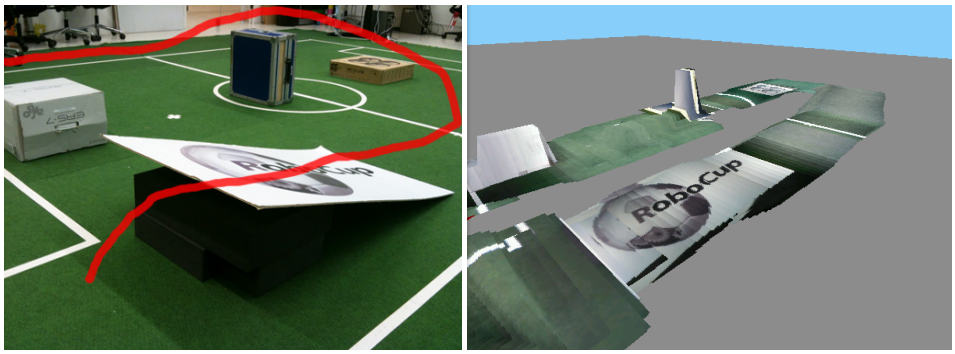


Fig. 7: Elevation map of a flight over several obstacles.

Elevation changes are detected using a filtered second order derivative (acceleration) of the sonar measurement z_{sensor}, as illustrated in Figure 8.

Obstacles that enter or leave the range of the ultrasound sensor result in sudden changes in ultrasound distance measurements. These changes are detected when the second order derivative exceeds a certain threshold $\gamma_{elevationEvent}$ and an elevation event is triggered. The threshold $\gamma_{elevationEvent}$ was carefully chosen such that altitude corrections performed by the AR.Drone altitude stabilization are not detected as being elevation events. An elevation event ends when the sign of the second order derivative switches. This happens when the AR.Drone altitude stabilization starts to change the absolute altitude to compensate for the change in measured altitude. Now, the elevation change can be recovered by subtracting the measured distance at the end of the elevation event from the measured distance before the elevation event was triggered, as illustrated in Figure 9.

In a final experiment, the AR.Drone flew with a constant speed over a large stair (Figure 10). The depth of each step is $30cm$ and the height of each step is $18cm$. The total height of the stair is $480cm$. After the stair is fully traversed by the AR.Drone, the estimated elevation is compared against the actual height of the stair.

After fully traversing the stair, the measured elevation is $313cm$ and the error is $\frac{480-313}{480} \times 100 = 35\%$. The shape of the measured elevation corresponds with the shape of the stair. However, the approach underestimates the elevation. When traversing the stair, the AR.Drone's altitude stabilization increases the altitude smoothly, which causes a continuous altitude increase. Therefore, the observed altitude difference within an elevation event is smaller than the actual altitude different caused by an object. Another explanation of the underestimation is the error in the triggering of elevation events (due to thresholds). When an elevation event is triggered at a suboptimal timestamp, the full altitude difference is not observed.

6 Conclusion

This paper demonstrates what is possible for rapid development when a realistic simulation environment for the AR.Drone is available. The simulation model of the AR.Drone is made publicly available [1] inside the USARSim environment. The validation of this model was described in in [1]. We hope that the availability of such a model inside the infrastructure of the RoboCup Rescue Simulation League will contribute in attracting researchers to develop advanced algorithms for such a system.

i

The mapping method described in this paper is able to map areas visually with sufficient quality for both human and artificial navigation purposes. Both the real and simulated AR.Drone can be used as source for the mapping algorithm. The visual map created by the simulated AR.Drone contains fewer errors than the map of the real AR.Drone. The difference can be explained by the measurement noise produced by the real AR.Drone. The optical flow algorithm in the AR.Drone's firmware should improve the accuracy, but is probably having difficulties with the gym floor.

Earlier work [1] shows that the visual map can be used to localize the AR.Drone and significantly reduce the error of the estimated position when places are revisited. The lines on the gym floor provide sufficient information for robust localization, although longitudinal movement along long straight lines is underestimated. Visual mapping is an important capability to scale up the robot team to the level where

[1] http://sourceforge.net/apps/mediawiki/usarsim/index.php?title=Aerial_Robots#AR.Drone

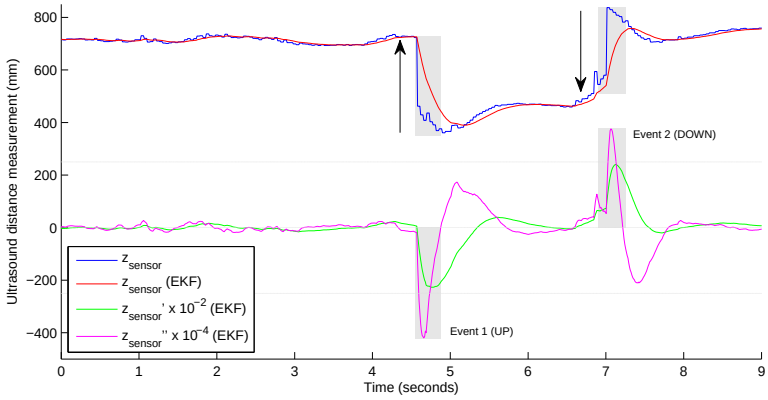


Fig. 8: Response of the ultrasound sensor when flying over an object of approximately $(60, 60, 40)mm$. The light-gray lines indicate the threshold $\gamma_{elevationEvent}$ and null-line. When the second order derivative (magenta line) exceeds the threshold, an event is started (lightgrey rectangle). An event ends when the derivative swaps sign. Each arrow indicates the change in elevation caused by the event. The first event increases the elevation when entering an object and the second event decreases the elevation when leaving the object. Between both events, the AR.Drone performs an altitude correction, as can be seen by the relatively slow increasement of distance. This increasement is not informative about the elevation and is ignnored by the elevation mapping method.

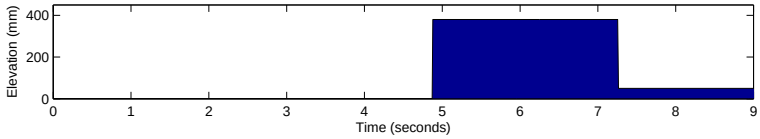


Fig. 9: Elevation δ below the AR.Drone over time. The elevation increases to approximately $40cm$ when flying above an obstacle. The elevation is decreased when the obstacle is out of the ultrasound sensor's range. There is a small error between both elevation events, resulting in a small false elevation ($\pm 50mm$) after the AR.Drone flew over the obstacle and is flying above the floor again.

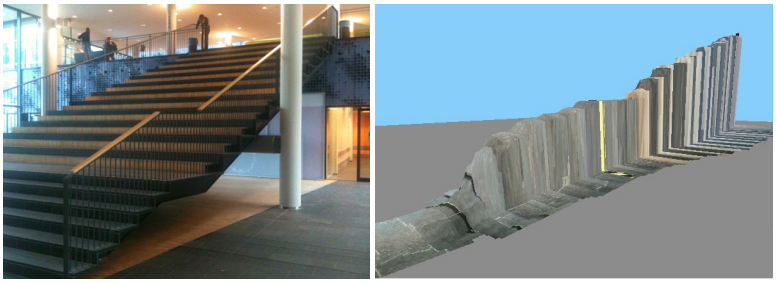


Fig. 10: Photo and map of a large stair at our university which is traversed by the AR.Drone. The depth of each step is $30cm$ and the height of each step is $18cm$. The total height of the stair is $480cm$.

they can explore a city block, which is close the current challenge in the Agent competition of the RoboCup Rescue Simulation league.

7 Future work

Future work will use additional data sources to improve the accuracy of the estimated position. The vehicle's motion can be computed from consecutive video frames. Our previous paper [8] describes a method for this. In addition a loop-closure method can be used to optimize the map when places are revisited.

A further improvement would be to include the images of the high-resolution front camera into the process.

Acknowledgements: This research is partly funded by the EuroStars project 'SmartINSIDE'.

References

1. Dijkshoorn, N., Visser, A.: Integrating sensor and motion models to localize an autonomous ar.drone. International Journal of Micro Air Vehicles **3**(4) (December 2011) 183–200
2. Dijkshoorn, N.: Simultaneous localization and mapping with the ar.drone. Master's thesis, Universiteit van Amsterdam (June 2012)
3. Steder, B., Grisetti, G., Stachniss, C., Burgard, W.: Visual SLAM for flying vehicles. Robotics, IEEE Transactions on **24**(5) (2008) 1088–1093
4. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (surf). Computer Vision and Image Understanding **110**(3) (2008) 346 – 359
5. Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., Burgard, W.: Efficient estimation of accurate maximum likelihood maps in 3d. In: IROS, San Diego, CA (USA) (2007) 3472–3478
6. Caballero, F., Merino, L., Ferruz, J., Ollero, A.: Unmanned Aerial Vehicle Localization Based on Monocular Vision and Online Mosaicking. Journal of Intelligent and Robotic Systems **55**(4) (2009) 323–343
7. Bills, C., Chen, J., Saxena, A.: Autonomous mav flight in indoor environments using single image perspective cues. In: International Conference on Robotics and Automation (ICRA). (2011)
8. Visser, A., Dijkshoorn, N., van der Veen, M., Jurriaans, R.: Closing the gap between simulation and reality in the sensor and motion models of an autonomous ar.drone. In: Proceedings of the International Micro Air Vehicle Conference and Flight Competition (IMAV11). (September 2011)