Closing the gap between simulation and reality in the sensor and motion models of an autonomous AR.Drone

Arnoud Visser, Nick Dijkshoorn, Martijn van der Veen and Robrecht Jurriaans*

ABSTRACT

This article describes a method to develop an advanced navigation capability for the standard platform of the IMAV indoor competition: the Parrot AR.Drone. Our development is partly based on simulation, which requires both a realistic sensor and motion model. This article describes how a visual map of the indoor environment can be made, including the effect of sensor noise. In addition, validation results for the motion model are presented. On this basis, it should be possible to learn elevation maps, optimal paths on this visual map and to autonomously avoid obstacles based on optical flow.

Keywords: Quadrotor, visual SLAM, monocular vision, SURF features, noise models

1 INTRODUCTION

Small quadrotors with on-board stabilization which can be bought off-the-shelf, like the Parrot AR.Drone, 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 platform. One of the prerequisites for autonomous navigation is to capability to make a map of the environment.

2 RELATED WORK

Our approach was inspired by Steder *et al.* [1], who presented a system that allows aerial vehicles to acquire visual maps of large environments using comparable setup with an inertia sensor and low-quality camera pointing downward. In their approach the inertia 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) [2] 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 thus, construct the graph that serves as input to the TORO-based network optimizer [3].

Prior to 2011 no standard platform existed in the Micro Air Vehicle competition[1], which allowed to participants to construct optimal sensor configurations. For instance, the team from MIT [4] installed a Hokuyo laser scanner on their UAV and where able apply many techniques developed for ground robotics on their flying robot. Another example is the approach of the team from Canterbury [5], where an omni-directional camera was mounted on the platform, which was used to acquire an heading estimation from the optical flow. At the latest edition of the Micro Air Vehicle competition[2] an Parrot AR.Drone was extended with an infrared camera to be able to follow people [6]. Because the AR.Drone is a quite novel 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.

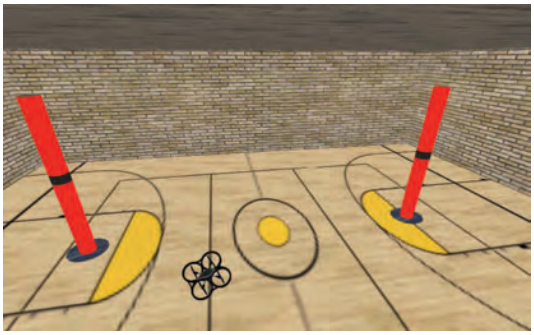

Figure 1: 3D model of a gym with realistic ground and wall textures which represents the Micro Air Vehicle pylon challenge.

As indicated in [8]; an accurate simulation of a quadrotor is a valuable asset, which allows safe and efficient development of control algorithms. Additionally, it gives direct access to ground truth values and allows to design repeatable experiments. To be used in such a way, not only the models of the actuators and sensors have to be validated. Another key feature is that the simulation environment is equipped with an editor which allows detailed modifications to the simulated

*Intelligent Systems Lab Amsterdam, Universiteit van Amsterdam

[1] http://www.emav2009.org, http://www.imav2010.org
[2] http://www.springimav2011.com

http://www.imavs.org/

http://www.imavs.org/

environment so that the details needed for the experiment can be easily added as indicated in Figure 1. The environment selected is USARsim [9], which allows physical realistic simulations and a versatile environment editor.

3 METHODS

3.1 Map stitching

Our approach uses image stitching to build a visual map of an indoor environment. The frames from the AR.Drone's low-resolution down-looking camera are aligned together and drawn on a canvas object. A set of matches between two camera frames is used to estimate a homographic model for the apparent image motion. This model can be composed with the estimated motion of the rest of images in order to build a mosaic. The sonar sensor is used to detect obstacles and mask these obstacles in the camera image to make the image stitching more robust.

The first camera frame $\mathbf{I}_0$ is added at the center of the canvas object without any image transformation. This image determines the scale of the map, because consecutive images are related to this image. Each next image is related to the previous image by calculating a local perspective transformation $\mathbf{H}_{(i-1)i}$. This perspective transformation $\mathbf{H}_{(i-1)i}$ is used to transforms image $\mathbf{I}_i$ such that it matches the scale, orientation and translation of image $\mathbf{I}_{i-1}$.

$\mathbf{H}_{(i-1)i}$ is computed by matching point features from two consecutive images. A certain degree of image overlap is required in order to find features that are present in both images. We use Speeded-Up Robust Features (SURF) [2] that are invariant with respect to rotation and scale. Each feature is represented by a descriptor vector and its position, orientation, and scale in the image. Each features from image $\mathbf{I}_i$ is matched with a feature from image $\mathbf{I}_{i-1}$ that has the shortest Euclidean distance.

The local perspective transformation $\mathbf{H}_{(i-1)i}$ is calculated by minimizing the back-projection using a least-squares algorithm. However, if not all of the point pairs fit the rigid perspective transformation (i.e. there are some outliers), this initial estimate will be poor. We use Random Sample Consensus (RANSAC) [10] to filter the set of matches in order to detect and eliminate erroneous matches. RANSAC tries different random subsets of four corresponding point pairs. It estimates the homography matrix using this subset and then computes the quality of the computed homography by counting the number of inliers.

Now that image $\mathbf{I}_i$ is related to image $\mathbf{I}_{i-1}$, it can be related to the first image $\mathbf{I}_0$ (canvas) by multiplying the local perspective transformation with the local perspective transformations from all previous matched frames.

$$\hat{H}_{j0} = \prod_{k=j}^{1} H_{(k-1)k} \tag{1}$$

The resulting transformation $\hat{\mathbf{H}}_{j0}$ is used to warp image $\mathbf{I}_j$. Finally, the warped image $\mathbf{I}_j^w$ is added to the canvas.

The local transformations between consecutive images can be composed to obtain the global transformation of the current frame, but local errors lead to a progressive drift in the transformation of future frames. One faulty local transformation can have disastrous impact on the image stitching. Therefore, we included several methods to detect faulty local transformations. Once a faulty local transformation is detected, the corresponding frame is dropped.

The first method to detect faulty transformations is counting the percentage of outliers. If the percentage exceeds threshold γ_1 the frame is being dropped (ignored). All feature points from image $\mathbf{I}_j$ are transformed with local transformation $\mathbf{H}_{(i-1)i}$. Then, the Euclidean distance between each transformed point from image $\mathbf{I}_i$ and the matched point from image $\mathbf{I}_{i-1}$ is measured. If the distance is larger then threshold ϵ the point is marked as outlier.

Another method to detect faulty transformations is by calculating the relative change of the global transformation when a new local transformation $\mathbf{H}_{(i-1)i}$ is added. Large relative changes indicate unlikely local transformations, because consecutive frames are likely to have only small differences in translation, rotation and scaling. If the relative change introduced by a frame's local transformation exceeds threshold γ_2 the frame is being dropped (ignored). The relative change $\delta\mathbf{H}$ introduced by frame $\mathbf{I}_j$ and its local transformation $\mathbf{H}_{(i-1)i}$ is calculated as following:

$$\delta\mathbf{H} = \hat{H}_{(i-1)0} ./ \hat{H}_{i0} \tag{2}$$

where $./$ is a per-element division. The frame is dropped if $\langle | \delta\mathbf{H} | \rangle > \gamma_2$.

Processing a single frame and merging it into the visual map requires approximately 150ms. The AR.Drone's framerate is fixed at 15fps, which is too high to process each single frame in realtime. In order to achieve realtime map stitching, frames that are receiving while another frame is still being process, are dropped. However, this reduces the vehicle's maximum speed that provides enough overlap between consecutive frames. For example, when flying at $1m$ altitude, the camera (64 degree FOV) perceives $1.24m$ floor. The maximum horizontal speed to achieve 50% overlap at 15fps is $9.3m/s$, which is nearly twice the actual maximum speed of the AR.Drone ($5m/s$). With reduced framerate (6.67fps instead of 15 fps), the maximum horizontal speed is $4.13m/s$, 83% of the actual maximum speed.

3.1.1 Obstacle masking

The map stitching method just described assumes that the terrain is approximately flat. However, this assumption does not hold when flying at low altitude. The parallax effect results in faulty local transformations and errors in the map.

A method to prevent the parallax effect is by removing obstacles from the camera frames, such that a flat terrain re-

http://www.imavs.org/

mains. The sonar sensor is used to create an elevation map. This elevation map is being generated simultaneously with the visual map and has the same scale and size. When image $\mathbf{I}_i$ is received, the corresponding piece from the elevation map is extracted by transforming the elevation map with $\hat{H}_{i0}^{-1}$. Now, the extracted elevation map has the same size and scale as the received image. All pixels from image $\mathbf{I}_i$ with an elevation greater then γ_3 are masked and not being used for feature detection. The method is currently under study, details will be published elsewhere.

3.1.2 Inertia

The motion was in the previous sections purely estimated on visual clues only. The AR.Drone is equipped with a number of additional sensors which give constants updates about the motion. The AR.Drone's inertial sensor data (body acceleration and attitude) can be used to estimate the current position. To get a robust estimate an Extended Kalman Filter is applied. The state vector comprises a position vector p^W, velocity vector v^W, acceleration vector a^W and attitude vector q^W.

$$x = [p^W v^W a^W q^W] \tag{3}$$

The resulting position estimate can be used as input for the map stitching algorithm. This method is applied in one experiment to study if the map stitching algorithm could benefit from this additional information.

3.2 Simulation model

The AR.Drone is a stabilized system (see Figure 2). When no control signals are given the quadrotor hovers on the same location, which is accomplished by a feedback loop which uses the sonar (for altitude) and the bottom camera (for horizontal position). The simulation makes use of this assumption. When no control signal is given, the AR.Drone stays at the same location. When a control signal for an longitudinal or lateral velocity is given, it calculates the force needed to reach that velocity (and assuming that the drag force D_b increases linearly with the velocity). When the control signal stops, the drag force D_b slows the quadrotor down until it hovers again. The USARSim quadrotor model uses the Karma physics engine (part of the Unreal Engine [11]) to simulate the force and torque acting on the aircraft. Yet, only the overall trust is calculated, the differential trust is not used. When moving in the horizontal plane, a real quadrotor changes its angle of attack (which is the defined as the angle between direction of motion e_V and the body frame e_N [12]). The Karma physics engine does not need this angle to calculate the resulting horizontal movement. Yet, this angle of attack has direct consequences for the viewing directions of the sensors, so the roll and the pitch should be adjusted in correspondence with horizontal movements.

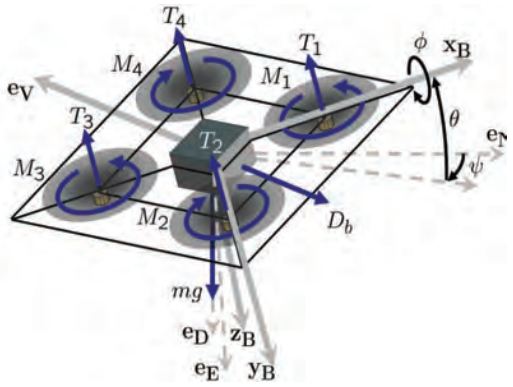

Figure 2: Free body diagram of a quadrotor helicopter (Courtesy Hoffman *et al.* [13]). Note that a right-handed orthogonal coordinate system is used with the z-axis pointing down. Each of the 4 motors has a trust T_i and momentum M_i. Together the motors should generate sufficient vertical trust to stay airborne, which is indicated by the gravity force mg in the direction e_D. Differential trust between the motors can provide roll ϕ and pitch θ torques, which lead to an angle of attack α. This can result in fast movements of the helicopter (e.g. in the horizontal plane) in the direction e_V which a resulting drag force D_b.

Control signals for vertical and rotational movements (around the z-axis) are calculated in the same manner. For vertical movements not only the drag force D_b is taken into account. In this case also the gravitational force mg is included in the equation. Rotations around the z-axis stop quite quickly when no control signal is given. For this rotational movement a 20x larger drag force D_r is used to model the additional inertia.

Figure 3: 3D model of the Parrot AR.Drone. This is a simplified model, based on the highly detailed model provided by Parrot SA.

The result is a simulation model (see Figure 3), which maneuvers close to the actual AR.Drone. Both the simulated and real system have the same dimensions $(0.525, 0.515, 0.115)m$. The principal elements of inertia are calculated correspondingly to $(0.0241, 0.0232, 0.0451)kg \cdot m^2$, assuming a homogeneous distribution of the mass.

http://www.imavs.org/

4 RESULTS

4.1 Map stitching

Three types of experiments have been carried out to evaluate the map stitching approach. Each experiment is performed with the AR.Drone and the simulated AR.Drone using USARSim.

The first experiment measures the performance of the method without obstacle masking. Four large puzzle pieces are lead out on the floor within a square of $1.6m$ x $1.6m$. These puzzle pieces are used as landmarks that are easily recognizable in the map. Each landmark has a label (A, B, C, etc). The distance between the centers of the landmarks is $1.3m$, except for the distance B-D ($\sqrt{1.3^2 * 1.3^2} = 1.84m$). Smaller puzzle pieces are lead out inside the square to provide enough texture for feature detection.

The realtime map stitching method is performed on the floor as described above. The distances between the landmarks inside the generated map (red lines) are compared to the know ground truth to compute the error of the stitched map.

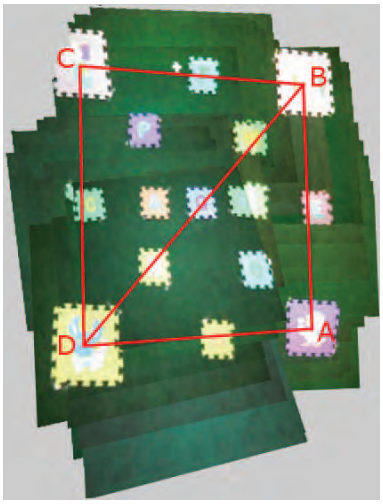

Figure 4: Map created by the map stitching method. Camera images are taken by the AR.Drone flying at approximately 0.85m.

The results of this experiment can be found in Table 1 and Figures 4 (real AR.Drone) and 5 (simulated AR.Drone). Both for the simulated and real AR.Drone a visual map is created with enough quality for human navigation purposes. The visual map created by the simulated AR.Drone contains fewer errors than the map of the real AR.Drone. This is not an intended result. Care has been taken to reproduce the circumstances in simulation as good as possible; the camera images in simulation are post-processed (decreased saturation, increased brightness, down-sampled resolution) to mimic the real images as close as possible. The difference between real and simulated visual map could also be explained by

Figure 5: Map created by the map stitching method. Camera images are taken by the simulated AR.Drone flying at approximately 0.80m.

landmarks	A-B	B-C	C-D	D-A	B-D
AR.Drone					
mean error (m)	0.385	0.146	0.608	0.156	0.445
error (%)	29.6	11.2	46.8	12.0	24.1
USARSim simulator					
mean error (m)	0.019	0.047	0.026	0.075	0.028
error (%)	1.46	3.62	2.00	5.77	2.15

Table 1: Accuracy of the realtime stitched map by measuring the distance between landmarks. The maps are creating using the AR.Drone (Figure 4) and the USARSim simulator (Figure 5) .

smoother movements between the frames in simulation. Yet, also here care has been taken to mimic the dynamics of the AR.Drone as close as possible (as described in Section 4.2). Visual inspection of the video stream shows that there are equivalent changes in the movements between frames. Our hypothesis is that the remaining difference between simulation and reality are due to the effect of automatic white balancing of the real camera. In the next experiment this hypothesis will be further studied.

A second experiment is performed to study our hypothesis that the real AR.Drone produces less accurate maps caused by the automatic white balancing of the camera. The post-processing step from the images in simulation is extended with an additional filter, which changes the brightness randomly (including a non-linear gamma correction). With this additional error source the map of the simulated AR.drone (Figure 6) close resembles the map of the real AR.Drone (Figure 4). Also the quantitive comparison of Table 1 and 2 show now that the errors in simulation increased from maximal $7.5cm$ to $25.4cm$, which means an improvement in realism. The maximum error in the real visual map is still a bit larger ($60.8cm$), but this can be attributed to a scaling error (a systematic error in the attitude measurement of the sonar sensor).

http://www.imavs.org/

Closer inspection shows that the white balance variations reduce the stability of the detected features, i.e., less of the same features are found between consecutive frames. This finding is supported by a statistical measure: the average Euclidean distance of features that are matched across consecutive frames. The post-processing increases the average feature distance from $22.1px$ to $32.9px$. This resembles the average feature distance of the real AR.Drone ($32.7px$).

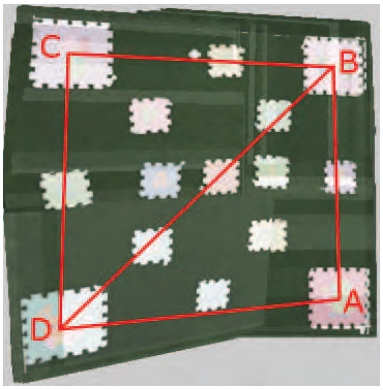

Figure 6: Map created by the map stitching method. Camera images are taken by the simulated AR.Drone that mimics the real AR.Drone's white balance variations.

landmarks	A-B	B-C	C-D	D-A	B-D
USARSim simulator (white balance variations)					
mean error (m)	0.031	0.181	0.215	0.254	0.190
error (%)	2.21	12.93	15.36	18.14	10.27

Table 2: Accuracy of the realtime stitched map by measuring the distance between landmarks. The map is creating using the USARSim simulator (Figure 6) with post-processing of the camera images to increase realism.

The floorplan from the first and second experiment is repeated 3 times in both directions. Now, the texture on the floor covers $4.8m$ x $4.8m$. The AR.Drone flew in a 8-shape above the floor to capture an intermediate loop (point A and E from Figure 7). This experiment is performed on the simulated AR.Drone to show how this method scales up under favorable conditions (without additional white balance noise).

The third experiment shows the limit of the current approach without loop-closure or information from other sources (inertia measurements and controls). Figure 7 and Table 3 reveal the accumulative error propagation of the current stitching method, which was not clearly visible from first experiment. The error at point A-E can only be reduced with a global optimization routine. In section 6 an indication is given how this accumulative error propagation can be battled.

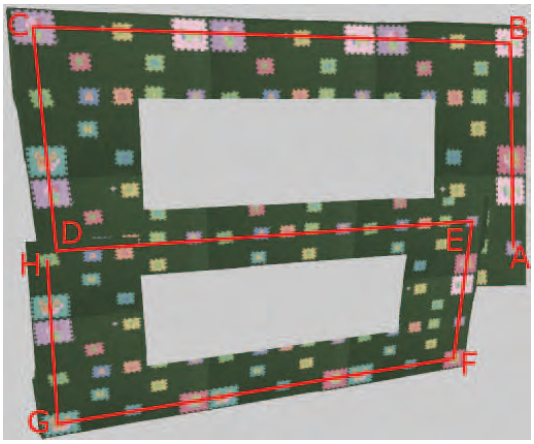

Figure 7: Map created by the map stitching method. Camera images are taken by the simulated AR.Drone flying at approximately 0.80m.

landmarks	A-B	B-C	C-D	D-E	E-F
mean error (m)	0.220	0.87	0.579	0.220	0.523
error (%)	10.48	19.33	27.57	4.89	24.90
landmarks	F-G	G-H	H-A	B-G	
mean error (m)	0.011	0.244	0.788	0.14	
error (%)	0.24	11.62	17.51	2.20	

Table 3: Accuracy of the realtime stitched map (Figure 7) by measuring the distance between landmarks. The map is creating using the USARSim simulator.

To show the effect of additional information, the last experiment is repeated with data from the AR.Drone's inertial sensor. The information from this sensor (body acceleration and attitude) is used in an Extended Kalman Filter to estimate the current position, as described in Section 3.1.2.

landmarks	A-B	B-C	C-D	D-E	E-F
mean error (m)	0.029	0.689	0.049	0.565	0.013
error (%)	1.38	15.31	2.33	12.56	0.62
landmarks	F-G	G-H	H-A	B-G	
mean error (m)	0.596	0.080	0.720	0.243	
error (%)	13.24	3.81	16.0	3.83	

Table 4: Accuracy of the EKF-based stitched map (Figure 8) by measuring the distance between landmarks. The map is creating using the USARSim simulator.

Figure 8 clearly shows that the AR.Drone flew a 8-shaped trajectory. Also the quantitative comparison showed that the relative error drops from maximum 27.57% in Table 3 is reduced to 16.0% in Table 4.

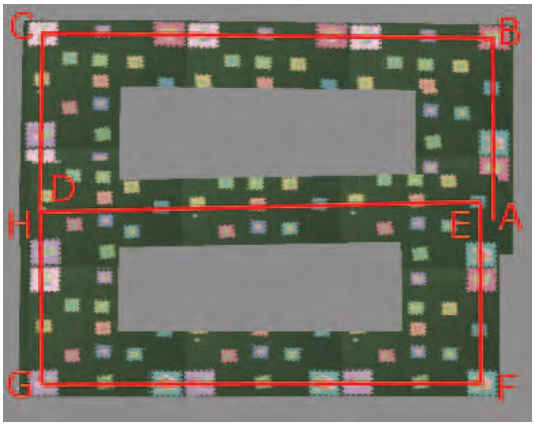

Figure 8: Map created by an Extended Kalman Filter (EKF). Camera images are taken by the simulated AR.Drone flying at approximately 0.80m.

4.2 Validation of the simulation model

To evaluate the USARSim quadrotor model, a set of maneuvers is flown with the actual AR.Drone and simulated AR.Drone. The differences between the maneuvers are studied in detail. To enable multiple repetitions of the same maneuver it is described as a set of time points (milliseconds since initialization) each coupled to a movement command. We wrote wrappers for the AR.Drone programming interface and for USARSim interface which read these scripts and output a control signal, using the system clock to manage the timing independently from the game engine and the AR.Drone hardware. Orientation, altitude and horizontal speed are recorded at a frequency of 200Hz during the maneuvers. These are gathered through the AR.Drone's internal sensors and the build-in algorithms, which are also used by the controller to operate the drone. The filtered output of the MEMS gyroscope is used for estimating orientation. The filtered output of the ultrasound distance sensor is used for estimating altitude. The optical flow algorithm using the bottom camera is used for estimating the horizontal (linear and lateral) speeds. The simulator has equivalent sensors. In addition, simulation can provide ground-truth data. Also for the real maneuvers an attempt was made to generate ground truth via an external reference system; the movements were recorded with a synchronized video system consisting of with four firewire cameras, capturing images at 20 frames per second at a resolution of 1024 x 768 pixels. The position of the AR.Drone in each frame has been annotated by hand.

Corresponding to NIST guidelines [14] a set of experiments of increasing complexity was performed. For the AR.Drone four different experiments were designed. The first experiment is a simple hover, in which the drone tries to maintain its position (both horizontal and vertical). The second experiment is linear movement, where the drone actuates a single movement command. The third experiment is a small horizontal square. The last experiment is a small vertical square.

4.2.1 Hovering

Quadrotors have hovering abilities just like a helicopter. The stability in maintaining a hover depends on environmental factors (wind, underground, aerodynamic interactions) and control software. If no noise model is explicitly added, the USARSim model performs a perfect hover; when no control signal is given the horizontal speeds are zero and the altitude stays exactly the same.

For the AR.Drone, this is a good zero-order model. One of the commercial features of the AR Drone is its ease of operation. As part of this feature it maintains a stable hover when given no other commands, which is accomplished by a visual feedback loop. So, the hovering experiment is performed indoors with an underground chosen to have enough texture for the optical- flow motion estimation algorithm.

As experiment the AR Drone maintains a hover 35 seconds. This experiment was repeated 10 times, collecting 60k movement samples for a total of 350 seconds. Over all samples the mean absolute error in horizontal velocity (the Euclidean norm of the velocity vector) is $0.0422m/s$ with a sample variance of $0.0012m^2/s^2$. From the samples we obtain the distribution of the linear and lateral velocity components.

From the velocity logs the position of the AR Drone during the 35 second flight was calculated. The mean absolute error of the horizontal position is $0.0707m$ with a sample variance of $0.0012m^2$.

4.2.2 Horizontal movement

In this experiment the drone is flown in a straight line. It is given a control pulse with a constant signal for 5 different time periods: 0.1s, 1s, 2s, 3s, and 5s. Each pulse is followed by a null signal for enough time for the drone to make a full stop and a negative pulse of the same magnitude for the same period, resulting in a back and forth movement. In Figure 9 the red line shows the control signal, the blue line the response of the AR.Drone. The experiment was repeated for 5 different speeds. The control signal s specifies the pitch of the drone as a factor (between 0 and 1) of the maximum absolute tilt θ_{max} which was set to the default value[3]. The trails were performed with the values of 0.05, 0.10, 0.15, 0.20, 0.25 for the control signal s.

Robots in USARSim are controlled with a standardized interface, which uses SI units. A robot in USARSim expects a DRIVE command with a speed in m/s and not the AR.Drone native signal s. Thus in order to fly comparable trials the relation between the drone's angle of attack α and the corresponding velocity v has to be investigated. When flying

[3] ARDrone firmware (1.3.3)

http://www.imavs.org/

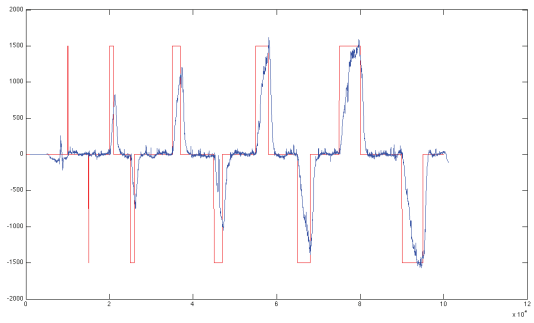

Figure 9: Response of the real AR.Drone on a number of pulses with a amplitude of $s = 0.15$.

straight forwards, the angle of attack α is equivalent with the pitch θ. In order to do this the samples from the logs where the drone has achieved maximum velocity has to be selected. Closer inspection of the velocity logs show that in each trial there is still constant increase of velocity for the first three pulses. For the last two pulses there is obvious plateauing, which indicates that the last two seconds of the five-second pulses is a good indication for the maximum velocity. Therefore the velocity at those last two seconds was used to compute mean absolute speeds $\bar{v}$, which are combined with the mean absolute pitch $\bar{\theta}$ as measured by the MEMS gyroscope. The estimates for $\bar{v}$ and pitch $\bar{\theta}$ are presented in Table 5 together with their standard deviation. Extrapolating the mean pitch $\bar{\theta} \simeq 7.5^o$ at control value $s = 0.25$ to the maximum control signal gives an indication of the drone's maximum pitch $\theta_{max} \simeq 30^o$ value. For typical usage, the angle of attack never exceeds 12^o degrees.

	Control signal s				
	0.05	0.10	0.15	0.20	0.25
$\bar{v}$ (m/s)	0.4044	0.6284	1.4427	1.7587	2.2094
σ_v (m/s)	0.096	0.226	0.070	0.126	0.165
$\bar{\theta}$ (deg)	1.4654	2.9025	4.1227	5.7457	7.4496
σ_θ (deg)	0.455	0.593	0.482	0.552	0.921

Table 5: Averaged velocity $\bar{v}$ measured at the end of a 5 seconds pulse of the control signal s, including the corresponding pitch $\bar{\theta}$ as measured by the gyroscope.

To convert the drone's control signal s to USARSim commands v a least-squares fit through the points of Table 5 is made for the linear function $v = c \cdot \theta$, which gives us $c = 0.2967$. Equation 4 gives the final conversion of a control signal s to a velocity v in m/s given the drone's maximum pitch θ_{max} in degrees.

$$v = 0.2967 \cdot s \cdot \theta_{max} \quad (4)$$

The USARSim model has a parameter P_θ for calculating the angle in radian given the velocity, which is the value $\hat{P}_\theta = 0.057$, as used in subsequent simulations.

The next experiment checks the acceleration of the real and simulated AR.Drone. First we give an estimate of how quickly the drone's controller changes its pitch to match the commanded pitch and how well it can keep it. For this we select all samples from 100ms after the start of the 2s, 3s, and 5s pulses till the first sample at which the commanded pitch has been reached. This corresponds to the time-span between which the drone has started to act on the change in the control signal until it reaches the commanded pitch. The result is illustrated in Figure 10.

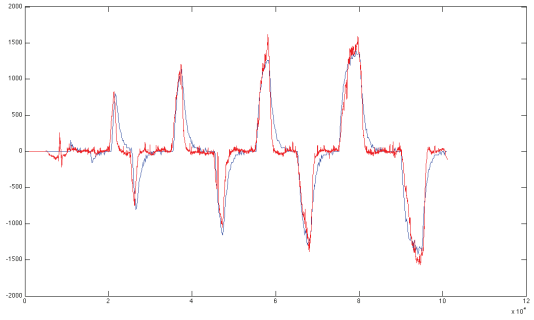

Figure 10: Response of the real (red) and simulated AR.Drone (blue) on the same pulses as shown in Figure 9.

As one can see, the acceleration has for the real and simulated AR.Drone nearly the same slope. The deceleration of the simulated AR.Drone is slightly slower. In the real AR.Drone the feedbackloop based on the optical flow of the ground camera actively decelerates the system. Overall, the dynamic behavior of the simulator closely resembles the dynamic behavior of the real system. Additionally, tests with more complex maneuvers (horizontal and vertical square) have been recorded, but unfortunately not yet analyzed in detail.

5 CONCLUSION

The current map stitching method is able to map small areas visually with sufficient quality for human navigation purposes. Both the AR.Drone and USARSim simulator 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. An experiment showed that the difference can be explained by the effect of automatic white balancing of the real camera.

The validation effort of the hovering and the forward movement shows that the dynamic behavior of the simulated AR.Drone closely resembles the dynamic behavior of the real AR.Drone. Further improvement would require to include the characteristics of the Parrot's proprietary controller into the simulation model.

http://www.imavs.org/

http://www.imavs.org/

6 FUTURE WORK

Future work will use visual slam to build the map and estimate the position of the robot. Such methods can handle loop-closures events when places are revisited and corrects for drift. Other future work is the integration of graphic framework to display a visual elevation map similar to [1].

A further improvement would be to include the images of the high-resolution front camera into the process. Those images can be used to calculate the optical flow from monocular stereo vision, which serves as the basis for both creating a disparity map of the environment. The disparity map can be used to detect obstacles ahead, which can be used in autonomous navigation. When combined with a time to contact method, the measurements can be used for a crude 3D reconstruction of the environment and another source for estimating the egomotion.

Once the visual map exists, this map can be extended with range and bearing information towards landmarks (as the pylons from the IMAV challenge). On close distance the bearing information could be derived directly from perception. By modeling this as an attractive force, the correct heading could be spread over the map by value iteration. In addition, obstacles visible in the disparity map of the front camera could be used as basis for a repulsive force. Both forces could be combined to a force field that guides a robot on the map. The availability of a realistic simulation model will add in the training of these machine-learning approaches.

ACKNOWLEDGEMENTS

We like to thank Parrot S.A. for providing an AR.Drone for the competition. This research is partly funded by the EuroStars project 'SmartINSIDE'. We like to thank Carsten van Weelden for his experiments to validate the motion model of the AR.Drone.

REFERENCES

[1] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard. Visual SLAM for flying vehicles. *Robotics, IEEE Transactions on*, 24(5):1088–1093, 2008.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008.

[3] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. pages 3472–3478, San Diego, CA (USA), 2007.

[4] Abraham Bachrach, Ruijie He, and Nicholas Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, December 2009.

[5] John Stowers, Andrew Bainbridge-Smith, Michael Hayes, and Steven Mills. Optical flow for heading estimation of a quadrotor helicopter. *International Journal of Micro Air Vehicles*, 1(4):229–239, December 2009.

[6] E. Jovanov, M. Milosevic, R. Tilly, M. Truex, and C. Jones. Autonomous Personnel Tracking Help AR.Drone. In *International Micro Air Vehicle Conference*, May 2011.

[7] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[8] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro-uav testbed. *Robotics Automation Magazine, IEEE*, 17(3):56 –65, sept. 2010.

[9] Workshop on robots, games, and research: Success stories in usarsim. In Stephen Balakirky, Stefano Carpin, and Mike Lewis, editors, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2009)*. IEEE, October 2009.

[10] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] Stefano Carpin, Jijun Wang, Michael Lewis, Andreas Birk, and Adam Jacoff. High fidelity tools for rescue robotics: Results and perspectives. In Ansgar Bredenfeld, Adam Jacoff, Itsuki Noda, and Yasutake Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*, pages 301–311. Springer Berlin / Heidelberg, 2006.

[12] T. Yechout, S. Morris, D. Bossert, and W. Hallgren. *Introduction to Aircraft Flight Mechanics: Performance, Static Stability, Dynamic Stability, and Classical Feedback Control*. American Institute of Aeronautics and Astronautics, Reston, VA, 2003.

[13] Gabriel M. Hoffmann, Haomiao Huang, Steven L. Wasl, and Claire J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proc. of the AIAA Guidance, Navigation and Control Conference*, 2007.

[14] Adam Jacoff, Elena Messina, Hui-Min Huang, Ann Virts, and Anthony Downs. Standard Test Methods for Response Robots. ASTM International Committee on Homeland Security Applications, January 2010. subcommittee E54.08.01.