IEEE Copyright Notice

This is an author version of the paper:

Tomáš Krajník, Matías Nitsche,Sol Pedre, Libor Přeučil, Marta E. Mejail
A simple visual navigation system for an UAV,
In 9th International Multi-Conference on Systems, Signals and Devices (SSD), 2012. doi: 10.1109/SSD.2012.6198031.

The full version of the article is available on IEEE Xplore or on request.

A simple visual navigation system for an UAV

Tomáš Krajník*, Matías Nitsche†, Sol Pedre†, Libor Přeučil*, Marta E. Mejail†,
*Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague
tkrajnik@labe.felk.cvut.cz, preucil@labe.felk.cvut.cz
†Departamento de Computacion, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires
spedre@dc.uba.ar,mnitsche@dc.uba.ar,marta@dc.uba.ar

***Index Terms*—autonomous navigation, UAV, image processing**

***Abstract*—We present a simple and robust monocular camera-based navigation system for an autonomous quadcopter. The method does not require any additional infrastructure like radio beacons, artificial landmarks or GPS and can be easily combined with other navigation methods and algorithms. Its computational complexity is independent of the environment size and it works even when sensing only one landmark at a time, allowing its operation in landmark poor environments. We also describe an FPGA based embedded realization of the method's most computationally demanding phase.**

I. Introduction

The field of micro aerial vehicles is nowadays a popular and fast evolving one. Current technology allows the μUAVs to perform quick and complex maneuvers [1], cooperate in manipulation and transportation tasks [2], navigate autonomously in structured [3] and unstructured [4] environments. However, the μUAVs can carry only a limited payload, which is a substantial obstacle in achieving their full autonomy. The payload limits not only the sensory equipment, but also constrains the computational power needed to perform localization and navigation algorithms.

Outdoors, the problem of localization can be solved simply by using the GPS [5]. Indoors, one can either rely on expensive external localization systems such as the ViCon [6], [2], [1] or use artificial, easily distinguishable landmarks [7]. Another way to deal with the limited payload issue is to transmit the UAV sensory readings to a ground station which performs the computationally demanding sensor processing and localization algorithms [8], [4], [9], [10]. Recently, a successful realization of a fully autonomous quadcopter capable of indoor operation has been reported in papers [11], [12], [13]. However, the authors of these articles report not only issues with algorithm speed, but also problems caused by occasional landmark deficiency.

In this paper, we present a simple and robust monocular camera-based navigation system for an autonomous quadcopter. The basic idea of the presented system is a simple method of control input computation based on monocular vision and odometry. Contrary to the traditional localization methods, which use advanced mathematical methods to determine vehicle position prior to control input computations, our method uses a more direct approach. We base the position estimation only on dead-reckoning techniques, and use the information from the camera image to determine only the quadrotor yaw and vertical speed. Focusing on estimation of the yaw and vertical speed from the camera image allows to use a simple histogram voting scheme instead of more complicated localization methods. The simplicity of the computation results not only in the method swiftness, but also its robustness to outliers and landmark deficiency. We also present an FPGA-based module, which implements the method's most computationally demanding phase. Due to its small size and power requirements, this FPGA module would allow the method deployment even on μUAVs.

A. Paper structure

In the following chapter, we describe the method working principle and depict in which aspects it differs to a similar method for ground robots described in [14]. After that, we outline a formal mathematical proof, which shows that the localization error of a robot guided by this method does not diverge. Then, we introduce the embedded solution for the SURF extraction, which is the most computationally demanding part of the method. Later on, we describe the experiments that verify the method functionality. Finally, we discuss the method performance and possible future work.

II. Method principle

The presented navigation method extends one used for a ground robot already presented in article [14]. The method is based on a "record and replay" technique. The mapping (or recording) phase is carried out by guiding the UAV along (polyline shaped) paths, which are supposed to be traversed autonomously later. During this phase, the navigation algorithm recognizes and tracks salient features [15] in an image from the quadrotor on-board camera. The resulting map consists of a sequence of straight segments, each described with its length, relative orientation and a set of salient features. Each feature description contains its SURF descriptor, image coordinates and distance from the segment start in two instants: when its tracking started and finished. The segment length is measured by the UAV dead-reckoning system and its orientation by the quadrotor gyroscope.

The autonomous flight is initiated at a position of the learned path start. The UAV turns in the direction of the first path segment and starts to move forwards until its dead-reckoning reports that it has traveled a distance greater than the segment length. During its flight, the drone corrects its altitude and

heading by comparing the mapped landmarks to the features in its field of view. Based on its distance from the segment start, the drone retrieves relevant landmarks from the segment map and computes their expected image coordinates by means of linear interpolation. These landmarks are paired to the features retrieved from the current image based on their descriptor similarity. A difference of horizontal and vertical coordinates of each pair is calculated and stored. Once all the features are processed, modes of the horizontal and vertical differences are computed by histogram voting and passed to the quadrotor yaw and altitude controllers respectively. These controllers adjust the yaw and vertical speed to keep both mode values close to zero, effectively suppressing the drone displacement from the learned path. A typical view from the drone camera with detected features, established pairs and position difference histograms are shown on Figure 1.

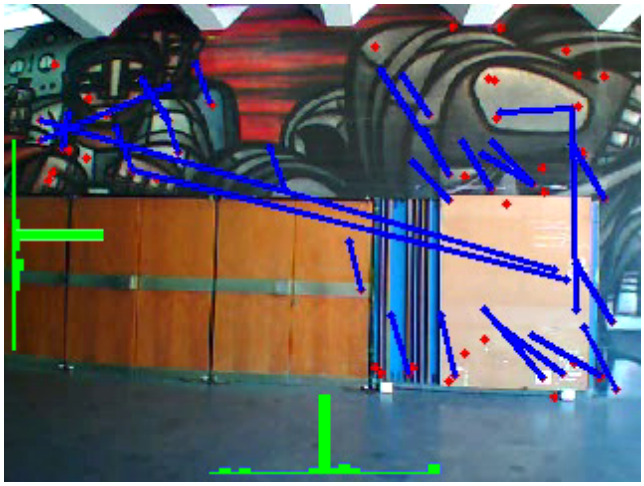

Fig. 1: UAV GUI during autonomous flight.

It should be noted that, unlike for ground robots, the forward speed of the quadrotor depends on its pitch. Also, changing direction of the drone movement just by changing its yaw is rather inefficient, because the centrifugal forces move the drone away from the desired path, see Section V. Therefore, the centrifugal force resulting from flying a curved path needs to be compensated by drone roll. Since the roll and pitch of the drone vary during the autonomous flight, the image feature coordinates are not the same as during the mapping phase. To deal with this, the current drone roll and pitch should be taken into account when computing the expected feature image coordinates. However, recalculation of the coordinates requires calibration of the forward camera.

III. Navigation Stability

In this section, we show how the aforementioned visual servoing method compensates the dead reckoning error. At first, we will describe the idea by means of geometrical terms, which should help the reader to interpret the mathematical formalism used in the proof. Then we will outline equations describing the evolution of the UAV position error as it travels one path segment. Based on these equations, we will mathematically prove that the position error does not diverge if the drone travels closed polygonal paths repeatedly.

A. Geometrical interpretation

Let us assume, that the UAV flies autonomously along a square shaped path and its starting position is a random variable with a 2D Gaussian distribution. We will represent the position uncertainty by a circle, in which the robot is found with some confidence. As the drone moves along a particular segment, it corrects its heading towards the segment axis and therefore its lateral position error is decreased. Larger displacement from the learned path causes higher value of the heading correction and therefore the lateral error decreases by a factor h each time the drone completes a segment. However, the longitudinal position error increases due to dead-reckoning error. The dead-reckoning error is proportional to the segment length and it can be modeled as an additive error o.

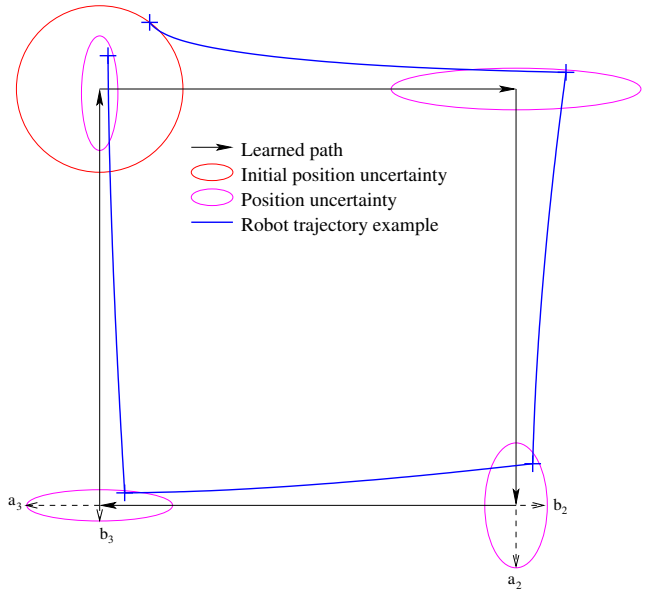

Fig. 2: Position uncertainty evolution for a square path.

If we depict the longer and shorter semiaxis of the i^{th} error ellipse as a_i and b_i respectively, we can prove that for $h < 1$,

$$\begin{aligned} a_\infty &= o/(1-h) \\ b_\infty &= ho/(1-h)\,. \end{aligned} \tag{1}$$

This means that as the robot travels the square path repeatedly, its position uncertainty converges to a finite value. Now, we will leave this simple symmetric case and describe the model of position uncertainty more rigorously.

B. A Model of UAV Movement

For the sake of simplicity, assume that the robot has learned a straight path consisting of one segment, which lies on the x axis of a 3D coordinate system. Let the learned path start at coordinate origin and end at point $(s, 0, 0)^T$. Let the drone initial and actual position be $(a_x, a_y, a_z)^T$ and $(x, y, z)^T$ respectively, and $|a_x| \ll s$, $|a_y| \ll s$ and $|a_z| \ll s$. Suppose

that the UAV associates a nonempty set of landmarks from the map to a set of landmarks it detects in its camera image.

If the aforementioned conditions are fulfilled and the robot horizontal and vertical controllers work the way described in the previous section, a larger horizontal/vertical displacement from the learned path causes a larger output of these controllers. Therefore, one can state that $\dot{y} \approx -k_y y$ and $\dot{z} \approx -k_z z$. Since the forward speed controller maintains a constant speed until the robot has traversed a path longer or equal to the segment length, we can state that $\dot{x} = k_x$. Solving these equations allows to estimate the robot position (b_x, b_y, b_z) at the end of the segment by the following equation:

$$\begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{-k_y s} & 0 \\ 0 & 0 & e^{-k_z s} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} + \begin{pmatrix} x_\xi \\ y_\xi \\ z_\xi \end{pmatrix},$$

where x_ξ, y_ξ and z_ξ are random (normally distributed) variables with variances sx_ϵ, y_ϵ and z_ϵ respectively. A compact form of the previous equation is

$$\mathbf{b} = \mathbf{M}\mathbf{a} + \mathbf{s}. \tag{2}$$

To apply (2) for an arbitrarily oriented segment, the coordinate system can be rotated by the matrix $\mathbf{R}$ and then back by $\mathbf{R^T}$. Thus, (2) can be rewritten as

$$\mathbf{b} = \mathbf{R^T}\mathbf{M}\mathbf{R}\mathbf{a} + \mathbf{R^T}\mathbf{s} = \mathbf{N}\mathbf{a} + \mathbf{R^T}\mathbf{s}. \tag{3}$$

Using (3), the UAV position at the end of the segment can be computed from its starting position. However, the absolute position is not our concern, we care only about the position error. Assuming that $\mathbf{a}$ and $\mathbf{b}$ are random variables with Gaussian distributions and covariance matrices $\mathbf{A}$ and $\mathbf{B}$ respectively, we can rewrite (3) in terms of covariance matrices

$$\mathbf{B} = \mathbf{N}\mathbf{A}\mathbf{N^T} + \mathbf{R^T}\mathbf{S}\mathbf{R} = \mathbf{N}\mathbf{A}\mathbf{N^T} + \mathbf{T}. \tag{4}$$

Equation (4) allows determination of the robot position uncertainty after traversing one segment.

C. Traversing multiple segments

Let the robot path be closed and consisting of n chained segments, each denoted by $i \in \{0, \ldots, n-1\}$. Since the segments are joined, $\mathbf{A_{i+1}} = \mathbf{B_i}$ and we can write

$$\mathbf{A_{i+1}} = \mathbf{B_i} = \mathbf{N_i}\mathbf{A_i}\mathbf{N_i^T} + \mathbf{T_i}.$$

The robot position uncertainty after traversing whole path consisting of the n segments will be

$$\mathbf{A_n} = \breve{\mathbf{N}}\mathbf{A_0}\breve{\mathbf{N}}^\mathbf{T} + \breve{\mathbf{T}},$$

where

$$\breve{\mathbf{N}} = \prod_{j=n-1}^{0} \mathbf{N_j}$$

and

$$\breve{\mathbf{T}} = \sum_{\mathbf{j=0}}^{\mathbf{n-1}} \left(\left(\prod_{\mathbf{k=n-1}}^{\mathbf{j}} \mathbf{N_k} \right) \mathbf{N_j^{-1}} \mathbf{T_j} \left(\mathbf{N_j^T} \right)^{-1} \left(\prod_{\mathbf{k=j}}^{\mathbf{n-1}} \mathbf{N_k^T} \right) \right). \tag{5}$$

If the robot traverses the entire path k-times, its position uncertainty $\mathbf{A_{kn}}$ can be computed in a recursive way by

$$\mathbf{A_{kn}} = \breve{\mathbf{N}}\mathbf{A_{(k-1)n}}\breve{\mathbf{N}}^\mathbf{T} + \breve{\mathbf{T}}. \tag{6}$$

D. Convergence proof

Since (6) is a Lyapunov discrete equation, its limit for $k \to +\infty$ is finite if all eigenvalues of $\breve{\mathbf{N}}$ lie within a unit circle and $\breve{\mathbf{T}}$ is symmetric.

Since matrix $\mathbf{S_i}$ is constructed as diagonal, $\mathbf{T_i} = \mathbf{R_i^T}\mathbf{S_i}\mathbf{R_i}$ is symmetric. The product $\mathbf{X}\mathbf{T_i}\mathbf{X^T}$ is symmetric for any $\mathbf{X}$ and therefore all addends in equation (5) are symmetric. Addition preserves symmetry and therefore the matrix $\breve{\mathbf{T}}$ is symmetric.

As every $\mathbf{N_i}$ equals to $\mathbf{R_i^T}\mathbf{M_i}\mathbf{R_i}$, its eigenvalues are equal to the $diag(\mathbf{M_i})$ and eigenvectors are columns of $\mathbf{R_i}$. Therefore, each matrix $\mathbf{N_i}$ is positive-definite and symmetric. Since the product $\mathbf{XY}$ of a positive definite matrix $\mathbf{X}$ and a symmetric positive definite matrix $\mathbf{Y}$ is positive definite, the matrix $\breve{\mathbf{N}} = \mathbf{N_{n-1}}\mathbf{N_{n-2}}\ldots\mathbf{N_0}$ is positive definite. Since the dominant eigenvalue of every $\mathbf{N_i}$ is one, eigenvalues of $\breve{\mathbf{N}}$ are smaller or equal to one.

The dominant eigenvalue of $\breve{\mathbf{N}}$ is equal to one if and only if the dominant eigenvalue of products $\mathbf{N_{i+1}}\mathbf{N_i}$ equals 1 for all i. However, the dominant eigenvalue of a product $\mathbf{N_{i+1}}\mathbf{N_i}$ equals 1 only if the dominant eigenvectors of both matrices are the same. This means, that the dominant eigenvalue of $\breve{\mathbf{N}}$ equals 1 only if all segments are rotated in the same direction.

Therefore, if the path is not a straight line, all eigenvalues $\breve{\mathbf{N}}$ are smaller than one and equation (6) has a finite solution. This means that if the robot travels the trajectory repeatably, its position uncertainty $\mathbf{A_{kn}}$ will not diverge. □

IV. Embedded solution

The computationally most intensive part of the navigation algorithm is the calculation of Speeded up Robust Features [15]. This method needs to be accelerated in order to get real-time performance of the whole system. An usual approach for accelerating image processing algorithms is to exploit their inherent parallel sections, building implementations for them on parallel architectures such as GPUs or FPGAs. The appearance of the CUDA [16] framework allowed to implement image processing methods on GPUs (graphical hardware of common PCs) with relative small coding effort resulting in their significant speedup [17], [18]. Although these implementations are based on affordable computational hardware, small robotic platforms such as μUAVs can not carry entire PC systems.

An alternative solution is to use Field Programmable Gate Arrays. FPGAs are devices made up of thousand of logic cells and memory. Both the logic cells and their interconnections are programmable using a standard computer. Their highly

parallel architecture with low power consumption, small size and weight provide an excellent platform for achieving real-time performance on this type of applications. Several authors [19], [20], [21], [22], [23], [24] reported successful implementation of robotic vision algorithms on FPGA-based hardware. Despite existing guidelines, methods and tools for FPGAs aimed at shortening the design cycle [25], [26], [27], [28], the main drawback of FPGA-based methods is time consuming implementation.

We present an FPGA-based implementation to massively accelerate the SURF algorithm. The detailed description of the implementation and complete embedded module can be found in [29] and [30]. The SURF algorithm has 4 broad stages: 1) Integral image generation, 2) Fast-Hessian detector (interest point detection), 3) Descriptor orientation assignment (optional) and 4) Descriptor generation. Considering the FPGA architecture, the first two stages were implemented in hardware using FPGA logic, while the rest is implemented in software on a PowerPC440 embedded processor with floating point arithmetics. Extra hardware modules were implemented to connect the FPGA logic to the embedded processor and to the camera. The block diagram of the solution is shown in Fig. 3.

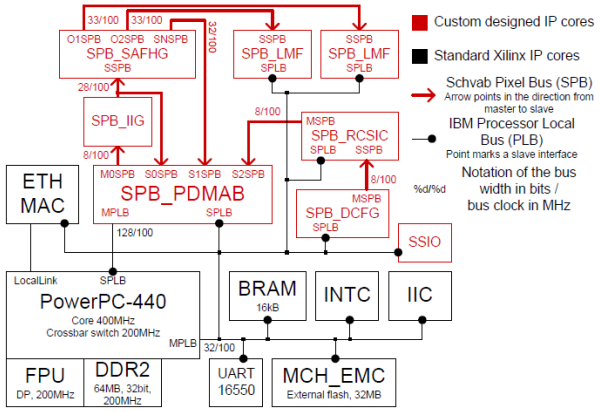

Fig. 3: Block diagram of the high-level SoPC architecture.

Here we shortly describe each of the custom designed IP cores. The SPB (Schvab Pixel Bus) is a single-master, multi-slave unidirectional custom bus with zero latency and a tiny logic overhead. The bus is designed to transfer naturally ordered image data, chaining image processing blocks to form a pipeline-like structure. The SPB_PDMAB core is a multi-channel DMA-capable bridge between SPBs and one PLB (the PowerPC peripheral bus). The SPB_DCFG core adapts a CMOS camera interface to the SPB in order to capture image frames. The SPB_RCSIC core subsamples and/or crops the captured images. The SPB_IIG core generates the integral image. The SPB_SAFHG is the key component of the SURF accelerator processing chain. It calculates the Fast-Hessian responses (the determinants) from the integral image and forms the entire scale space used by the detector. Finally, the SPB_LMF core performs thresholding and non-maxima suppression.

Maching method		Platform		
Limited Search	Laplacian Used	CPU	GPU	FPGA
Y	Y	0.62	-	0.53
Y	N	0.61	0.54	0.53
N	Y	0.43	-	0.27
N	N	0.43	0.27	0.26

TABLE I: Comparison of SURF, GPU-SURF and FPGA-SURF distinguishability.

The complete embedded system is implemented using the Avnet MiniModule Plus AES-MMP-V5FXT70-G solution that includes a Xilinx Virtex-5 XC5VFX70T FPGA with a PowerPC440 embedded processor. This module provides all key memories and communication interfaces for the application. It is originally designed to be attached to Avnet's MiniModule Plus baseboard. Instead, a tailored small baseboard was designed, that is significantly smaller and optimized for computer vision applications.

Although this implementation follows the original definition as closely as possible, optimizations for hardware implementation were necessary. These affect the precision of the detector and therefore repeatability and distinguishability of the whole algorithm. By distinguishability, we understand the chance of the algorithm to establish a valid correspondence between features from two images of the same scene with a viewpoint change. Correspondences are established solely by their descriptor Euclidean distance and then checking these against geometric constraints of multiple view geometry [31]. The ratio of valid correspondences to the number of correspondence candidates is a measure of the SURF algorithm distinguishability. The higher ratio means better algorithm performance.

Table I shows the results for four modifications of the tentative correspondence creation. Since each two consecutive images are taken with a quite small viewpoint change, it is possible to limit the correspondence creation by a pixel distance upper limit (200 pixels, first column in the table I). The second possible choice is whether to consider the Laplacian sign while establishing the correspondence. These results were obtained for our 1500 image dataset collected in a park-like environment.

Performed experiments show that the repeatability and robustness of the FPGA-SURF have not been severely affected and are comparable to the original and GPU-SURF implementation. However, the FPGA-SURF is faster than the original - it processes approximately 10 images (1024x768 pixels) per second - (considering about one hundred descriptors per image), consumes aproximatelly 7 Watts, occupies less space and weighs considerably less than a GPU-based system. Reasonable speed of feature extraction together with its lower size, weight and power consumption opens possibilities for applications in small embedded devices as the presented μUAV application described in this work.

V. Experiments

The assumptions formed in the previous sections have been verified in two experimental scenarios. The first scenario aims at evaluating the position uncertainty evolution as the UAV travels a single segment. The second experiment examines the UAV position uncertainty as it travels a closed path.

A. The experimental platform

The experiments have been performed with the AR-Drone platform [32], which is an electrically powered quadcopter originally intended for augmented reality games. Its sensory equipment consists of a three-axis accelerometer and gyroscope, sonar-based altimeter and two cameras. The first camera has a $75^\circ \times 60^\circ$ field of view, is aimed forwards and provides a 320×240 pixel color image. The second one is mounted on the bottom, provides color image with 176×144 pixels and its field of view is $45^\circ \times 35^\circ$. The drone can achieve speeds over $5 \mathrm{ms}^{-1}$ and its batteries provide enough energy for a 15 minutes flight. However, its turbines are not powerful enough to keep a steady position in windy conditions, which limits its outdoor use.

B. External localization system

During experiments, a ground robot can be stopped and its position relative to the path can be measured by hand by an ordinary ruler. This is not possible for an UAV, which cannot just stop and hold its position without any drift. Moreover, the first experimental scenario needs a continuous measurement of the quadrotor position. Therefore, we have created a simple localization system comprising of an off-the-shelf USB camera and a circular pattern, which was attached on the quadcopter. The videos from experiments were processed afterwards by a fast roundel detection algorithm, which is able to localize the aforementioned pattern with a few centimeter precision.

C. Position error evolution for a single segment

The purpose of this scenario is to test the ability of the navigation system to reduce the altitude and lateral displacement relative to the learned path. Moreover, it is possible to have a rough estimation of the UAV position uncertainty model constants and use these to predict the drone position deviation in the following scenario.

The drone is taught one approximately 3 m long segment. After that, it is requested to autonomously navigate the segment four times. In the first autonomous flight, its starting position is the same as the starting position of the taught path. The second and third autonomous run start with a horizontal displacement from the learned segment start. The demonstrate the effect of the centrifugal forces acting on the quadrotor, the roll compensation is switched off during the second flight. The fourth autonomous flight is started above the reference path start. During both training and autonomous navigation flight, the drone has been in the localization camera field of view, and therefore, it was possible to establish its position. This allowed to calculate the UAV displacement from a reference path for each moment of its autonomous flight.

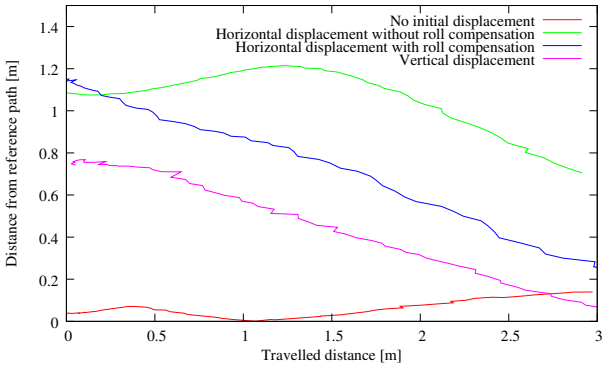

Fig. 4: Position error evolution during a single segment flight.

The recorded positions, shown on Figure 4, indicate that as the UAV travels the segment, its distance from the reference path slowly diminishes. The rate at which the lateral and vertical error decreases depends on the environment characteristics. In an indoor environment, the actual error decay rate depends on the distance of the segment end to the nearest group of landmarks, which are typically located on a wall [33]. In this particular case, we can see that both the horizontal and vertical error is diminished approximately by a factor of 0.8 per meter of the path.

D. Position error evolution for an entire path

This experimental setup is similar to the original two-dimensional version of the navigation algorithm [14]. The UAV is taught a four segment, 28 m square-shaped path in an indoor environment. After that, it is placed at the path start and requested to autonomously fly the entire path three times consecutively. Once again, the test is repeated with a (approx.) 1 m initial position displacement both in vertical and horizontal direction. The localization system takes a snapshot of the drone each time it completes the entire path and computes the UAV distance from the path start.

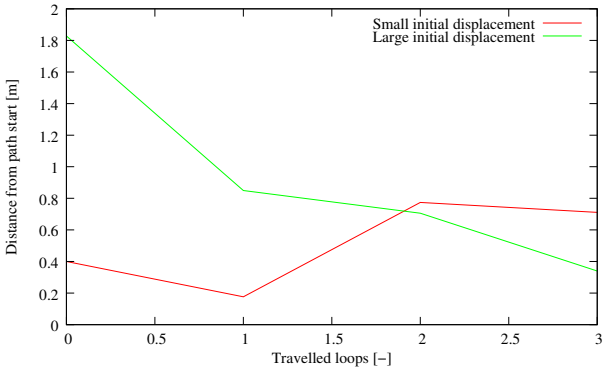

Fig. 5: Position error relative to the path start.

The measurements taken by the localization system show, that the position error stabilizes around 0.7 m, see Figure 5. Since the expected error value computed by Equation (6) is also 0.7 m, the measured results are in good accordance with the theoretical model introduced in Section III.

VI. Conclusion

We have presented a visual navigation system for a quadrotor helicopter. Contrary to the traditional localization methods, which use advanced mathematical methods to determine vehicle position prior to control input computations, our method uses a more direct approach. We base the position estimation only on dead reckoning, and use the information from the onboard camera image to determine only the UAV yaw and vertical speed. This allows to use a simple histogram voting scheme, which makes the method swift and robust to outliers and landmark deficiency situations.

Moreover, we presented an FPGA-based realization of the feature extraction algorithm. By using this hardware implementation, fully autonomous operation can be achieved.

With the experiments performed we showed how, during replay, the system diminishes an initial deviation from its previously learnt path, stabilizing with a relatively small error.

Although the method achieves to navigate the UAV with sufficient precision using only its onboard sensors, it has several limitations. It can operate only along paths it has traveled during a human-guided training run. Moreover, these paths can be composed only from straight line segments with a limited length.

In the future, we would like to overcome these limitations by using more sophisticated map building methods like the ones described in [34], [35]. Moreover, we would like to use publicly available maps instead of the human guided training run. Finally, we expect to physically integrate our FPGA-based implementation to a Microdrone quadrotor, in order to perform testing with a fully autonomous setup.

Acknowledgments

The work has been supported by the Ministry of Education of the Czech Republic by projects MEB111009, 216240, 7AMB12AR022 and by EU project No. 7E08006. The Ministry of Science of Argentina also supported this work by projects RC/08/09 and ARC/11/11.

References

[1] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *International Symposium on Experimental Robotics*, Delhi, India, Dec 2010.

[2] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, Jan 2011.

[3] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *IEEE International Conference on Robotics and Automation*, May 2011.

[4] M. Blöandsch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 21 –28.

[5] "Ascending technologies." [Online]. Available: http://www.asctec.de/

[6] "Motion capture systems." [Online]. Available: http://www.vicon.com/

[7] S. Lange, N. Sünderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments," *Electrical Engineering*, 2009.

[8] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," in *SPIE Unmanned Systems Technology XI*, 2009.

[9] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "AR-Drone as a Robotic Platform for Research and Education," in *International Conference on Research and Education in Robotics*. Prague: Springer, 2011.

[10] C. Yuan, F. Recktenwald, and H. Mallot, "Visual steering of UAV in unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 3906–3911.

[11] M. Achtelik et al., "Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[12] A. Huang et al., "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *International Symposium on Robotics Research (ISRR)*, Flagstaff, Arizona, USA, Aug. 2011.

[13] S. Erhard, K. Wenzel, and A. Zell, "Flyphone: Visual Self-Localisation Using a Mobile Phone as Onboard Image Processor on a Quadrocopter," in *Selected papers from the 2nd International Symposium on UAVs*. Springer, 2010, pp. 451–465.

[14] T. Krajník et al., "Simple, yet Stable Bearing-only Navigation," *Journal of Field Robotics*, October 2010.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. Vangool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[16] NVIDIA. (2012, Jan.) CUDA: Parallel Programming made Easy. [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html

[17] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)." [Online]. Available: http://cs.unc.edu/~ccwu/siftgpu

[18] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *CVPR 2008 Workshop (June 27th)*, Anchorage Alaska, June 2008.

[19] J. Diaz, E. Ros, F. Pelayo, E. Ortigosa, and S. Mota, "FPGA-based real-time optical-flow system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 274–279, Feb 2006.

[20] V. Bonato et al., "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1703–1712, Dec 2008.

[21] M. Tornow et al., "Hardware approach for real time machine stereo vision," in *WMSCI 2005: 9th World Multi-Conference on Systemics, Cybernetics and Informatics, Jul, 2005, Vol 5*, 2005, pp. 111–116.

[22] J. Diaz, E. Ros, S. Mota, and R. Rodriguez-Gomez, "FPGA-based architecture for motion sequence extraction," *International Journal of Electronics*, vol. 94, no. 5, pp. 435–450, May 2007.

[23] D. Cardon, W. Fife, J. Archibald, and D. Lee, "Fast 3D reconstruction for small autonomous robots," in *Thirty-First Annual Conference of the IEEE Industrial Electronics Society, Vols 1-3*, 2005, pp. 373–378.

[24] S. Pedre, A. Stoliar, and P. Borensztejn, "Real Time Hot Spot Detection using FPGA," in *14th Iberoamerican Congress on Pattern Recognition*. Springer, 2009, pp. 595–602.

[25] B. Bailey and G. Martin, *ESL Models and their Application. Electronic System Level Design and Verification in Practice*. Springer, 2010.

[26] S. Pedre, T. Krajník, E. Todorovich, and P. Borensztejn, "A co-design methodology for processor-centric embedded systems with hardware acceleration using FPGA," in *Proceedings of the VIII IEEE Southern Programmable Logic Conference*, 2012.

[27] J. Gaisler. (2011, Oct.) A structured VHDL Design Method. [Online]. Available: http://www.gaisler.com/doc/structdes.pdf

[28] A. J. Virginia, Y. D. Yankova, and K. L. M. Bertels, "An empirical comparison of ANSI-C to VHDL compilers : SPARK, RORCC and DWARV," in *Anual Workshop on Circuits Systems and Signal Processing ProRISC*, 2007, pp. 388–394.

[29] J. Šváb, T. Krajník, J. Faigl, and L. Přeučil, "FPGA-based Speeded Up Robust Features," in *IEEE International Conference on Technologies for Practical Robot Applications*. Boston: IEEE, 2009, pp. 35–41.

[30] M. Šváb, "FPGA-based Computer Vision Embedded Module," Master Thesis, Dept. of Cybernetics, CTU, 2011.

[31] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[32] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The Navigation and Control Technology Inside the AR.Drone Micro UAV," in *18th IFAC World Congress*, Milano, Italy, 2011, pp. 1477–1484.

[33] T. Krajník and L. Přeučil, "A Simple Visual Navigation System with Convergence Property," in *European Robotics Symposium*. Heidelberg: Springer, 2008, pp. 283–292.

[34] T. Vintr et al., "Batch FCM with volume prototypes for clustering high-dimensional datasets with large number of clusters," in *World Congress on Nature and Biologically Inspired Computing*, 2011, pp. 427–432.

[35] T. Vintr, L. Pastorek, and H. Rezankova, "Autonomous robot navigation based on clustering across images," in *International Conference on Research and Education in Robotics*, Prague, 2011, pp. 310–320.