Autonomous Quadrotor Flight Using Onboard RGB-D Visual Odometry

Roberto G. Valenti, Ivan Dryanovski, Carlos Jaramillo, Daniel Perea Ström, Jizhong Xiao*
* *Senior Member, IEEE*

***Abstract*— In this paper we present a navigation system for Micro Aerial Vehicles (MAV) based on information provided by a visual odometry algorithm processing data from an RGB-D camera. The visual odometry algorithm uses an uncertainty analysis of the depth information to align newly observed features against a global sparse model of previously detected 3D features. The visual odometry provides updates at roughly 30 Hz that is fused at 1 KHz with the inertial sensor data through a Kalman Filter. The high-rate pose estimation is used as feedback for the controller, enabling autonomous flight. We developed a 4DOF path planner and implemented a real-time 3D SLAM where all the system runs on-board. The experimental results and live video demonstrates the autonomous flight and 3D SLAM capabilities of the quadrotor with our system.**

I. INTRODUCTION

Micro aerial vehicles such as quadrotors are popular platforms often used by researchers because of their agility, high maneuverability, simple mechanicale design and compact size. Their applications range from surveillance, search and rescue, to 3D mapping and photography. In order to perform such tasks, they require a set of sensors suited to the particular use and context, and the capability of ensure stable and autonomous flight. Global Positioning System (GPS) is one of the most common sensors used for outdoor flight. Such solution is not always reliable, particularly when the signal's reception or precision might be unacceptable, as in the case of dense or indoor environments. Furthermore, in tasks like exploration and autonomous navigation in cluttered environments, global position information of the MAV is not sufficient, so a perception of the surroundings is needed. When external motion capture systems can not be deployed, the vehicle needs to rely only on onboard sensors such as laser scanners and cameras. Laser scanners provide range information with high precision and odometry can be derived by scan matching technique. However, laser scanners are not an optimal solution because of their weight and high power consumption. Therefore, for more complex environment, The need of lightweight 3D perception sensors is fulfilled by cameras. For this reason, visual odometry algorithm has become very common for flying robots.

Fig. 1. The CityFlyer MAV equipped with an Asus Xtion Pro Live RGB-D camera.

An RGB-D camera is a device which provides RGB (red, green, blue) color and depth information for each pixel of the image. Depth is retrieved through the conjuction of an infrared projector and an infrared receiver. Recently new RGB-D cameras such as the Microsoft Kinect and the Asus Xtion, have become popular in the robotics community due to their reduced size, low weight and affordable cost.

In this paper we present our approach for autonomous quadrotor flight and navigation by means of pose data from an RGB-D visual odometry algorithm, which relies on a frame-to-model registration technique, mantaining a low computation complexity (without any GPU acceleration), while reducing considerably the drift error of a typical frame-to-frame approach.

II. PREVIOUS WORK

Autonomous MAV flight using data from visual odometry has been achieved in the past by several researcher groups with different approaches.

Algorithms based on stereo vision have been used by Johnson et al. [13] and Yu et al. [26] to control the altitude of a UAV. Park et al. [19] and Zamudio et al. [27] used a stereo camera to control a quadrotor. In [19] a stereo vision system is used to perform collision avoidance. Other

*This work is supported in part by U.S. Army Research Office under grant No. W911NF-09-1-0565, U.S. National Science Foundation under grants No. IIS- 0644127 and No. CBET-1160046, Federal HighWay Administration (FHWA) under grant No. DTFH61-12-H-00002 and PSC-CUNY under grant No. 65789-00-43 and by ACIISI program "Apoyo al Personal Investigador en Formación 2010".

Roberto. G. Valenti and Jizhong Xiao are with the Electrical Engineering Department, City College of New York, Convent Ave & 140th Street, New York, NY 10031 rvalent00@citymail.cuny.edu, jxiao@ccny.cuny.edu, corresponding author

Ivan Dryanovski and Carlos Jaramillo are with the Dept. of Computer Science, The Graduate Center, The City University of New York, 365 Fifth Avenue, New York, NY 10016 idryanovski@gc.cuny.edu, cjaramillo@gc.cuny.edu

Daniel Perea Ström is with Dept. of Ingeniera de Sistemas y Automtica y Arquitectura y Tecnologa de Computadores, Universidad de La Laguna, Calle Astrofisico Fco. Sanchez s/n Edificio Fisica/Matematicas, San Cristbal de La Laguna, Santa Cruz de Tenerife, Spain 38271 dani@isaatc.ull.es

strategies using monocular vision, which better meet the needs of a limited payload of a MAV, have been adopted. Achtelick et al [1] controlled a quadrotor for both indoor and outdoor flight by using visual information coming from a single camera pointing down, where depth information was recoverred by fusing a pressure sensor and an accelerometer through an Extended Kalman Filter. A similar system is presented by Blosch et al. [3] to perform indoor exploration with a MAV (with the difference that it does not rely on any other exteroceptive sensor). Conroy et al. [5] and Zingg et al. [28], present a biologically inspired vision system for safe and stable corridor navigation of a MAV based on optical flow from an omnidirectional camera. Rondon et al. [22] present a monocular vision system to estimate and control altitute, lateral position and forward velocity of a MAV via optical flow.

In order to perform several tasks with the same platform, both monocular and stereo vision techniques have been adopted. For instance Hrabar et al. [11] combine stereo vision from a forward-facing camera pair and optical flow from two sideways-looking monocular cameras in order to avoid obstacles and navigate inside a canyon with maximum clearance. Meier et al. [17] and Fraundorfer et al. [9] integrate a down-facing monocular camera with a forward-facing stereo camera for flight control and obstacle avoidance. For a more robust navigation in different environments, visual odometry is sometimes fused with other exteroceptive position sensors. In the work by Tomic et al. [14] and Bachrach et al. [2], a quadrotor is equipped with a laser range-finder and a stereo camera to enable stable flight and SLAM in a number of large-scale, GPS-denied environments, such as an urban canyon.

Unlike visual odometry from standard cameras, RGB-D visual odometry has not been widely used by researchers to control a flying robot. Stowers et al. [24] presented one of the first results of using RGB-D camera for real-time robot-control application. They use a Kinect pointed towards the ground to estimate and control the height of a quadrotor. The work presented by Huang et al. [12] is the most closely related to ours. They use a RGB-D visual-odometry (based on standard stereo visual odometry) to control a MAV in unknown indoor environments and do mapping. The pose estimation runs on the onboard computer while mapping and loop closure runs on an off-board computer.

In our proposed system, we use a recently developed visual odometry algorithm, able to reduce the drift error thanks to a fast frame-to-model approach. Further, all the components of the system (including mapping and loop closure) run onboard on the same kind of computer as the system proposed by Huang et al. [12].

III. SYSTEM ARCHITECTURE

The platform we use for our experiments is an AscTec Pelican quadrotor [10], on which we mounted an Asus Xtion Pro Live. The quadrotor is equipped with a 1.86 GHz Core2Duo processor with 4GB of RAM and a Flight Control Unit (FCU) board with 2 ARM7 microcontrollers,

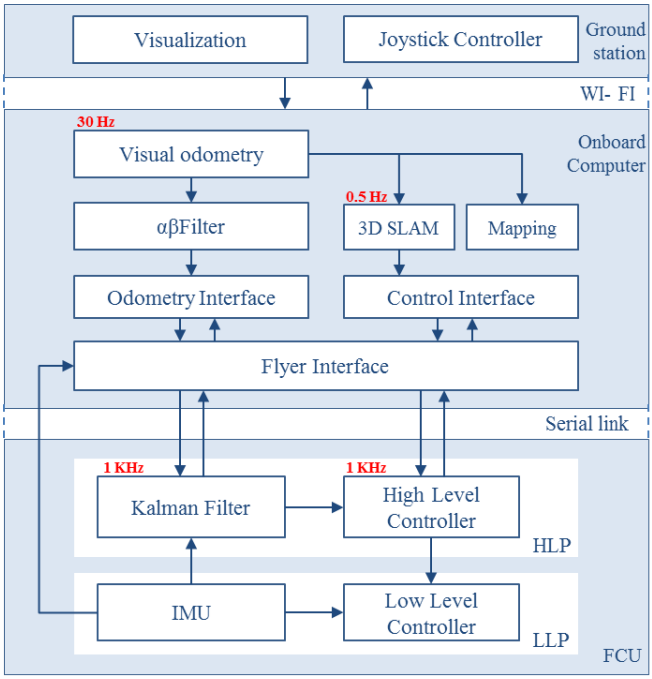

Fig. 2. System diagram. All the software running on the on-board computer and on the FCU are our contribution except mapping and the Low Level Controller provided by Asctec Technology.

an Inertial Measurement Unit (IMU) and a pressure sensor. The system architecture is shown in Fig. 2 which is described in detail in our previous work [6]. We use one of the two microcontrollers, the so-called High Level Processor (HLP), to run our custom firmware that handles sensor fusion and control, while the Low Level Processor (LLP) is responsible for attitude control, IMU data fusion and hardware communication. The powerful on-board computer is able to manage visual odometry, 3D SLAM and motion planning. The entire framework is distributed between a ground station, the on-board computer and the FCU. The ground station is only used for visualization and teleoperation. Our framework uses ROS [21] as middleware, allowing communication among all the different components of the software (implemented as *nodelets*, a ROS mechanism for zero-copy message transport). The HLP and the onboard computer communicate with each other through the serial interface, where the `Flyer Interface` sends ROS messages and services traslated into packets. Communication between the two ARM7 microcontrollers (HLP and LLP) of the FCU board is via an I^2C bus.

IV. STATE ESTIMATION

A. Visual Odometry

The visual odometry adopted in this paper uses a frame-to-model registration approach to compute the transformation between two consecutive camera poses. This approach allows us to considerably decrease the drift in the pose as demonstraded in [7]. We first detect the features in the

captured scene by using Shi-Tomasi [23] algorithm and their 3D coordinates in the camera frame, then we align these features against a global model of 3D features (previously detected). We perform data association and filtering using a probabilistic method, which employs a novel uncertainty estimation based on a Gaussian mixture model (described in our previous work [7]). We use this 3D normal distribution model for each feature detected in the incoming RGB-D image. This set of 3D features (with mean and covariance matrix $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$), is expressed with respect to the camera reference frame. We refer to this set as *Data*. We have a similar set that we call *Model* and is expressed in the fixed frame. We use ICP [4] to align *Data* against *Model* and then it is transformed into the fixed frame. The alignment produces the transformation *T*, composed by a Rotation matrix *R* and a translation vector $\mathbf{t}$. We also need to express the mean and the covariance matrix of each feature in *Data* with respect to the fixed frame. We can do so according to:

$$\boldsymbol{\mu}^{'} = \mathbf{R}\mu + \mathbf{t} \tag{1a}$$

$$\boldsymbol{\Sigma}^{'} = \mathbf{R}\boldsymbol{\Sigma}\mathbf{R}^{T}. \tag{1b}$$

Once we have *Data* expressed in the fixed frame we generate correspondences adopting the following steps. First we build a Kd-tree [18] of the *Model*, and then for each feature of *Data* we find *k* nearest Euclidean neighbors from the *Model*. Next, for each point $\mathbf{d}_i^{'}$ of the transformed *Data*, we compute the Mahalanobis distance between the point and its nearest neighbor in the *Model*, $\mathbf{m}_j$.

$$dist(\mathbf{d}^{'}, \mathbf{m}) = \sqrt{\Delta_{\mathbf{d}'\mathbf{m}}(\Sigma^{[M]} + \Sigma^{[D']})^{-1}\Delta^{T}_{\mathbf{d}'\mathbf{m}}} \tag{2}$$

If this distance is lower than a certain threshold, the two points are associated establishing the correspondence. All the points which cannot be associated are inserted in the *Model*. The update is performed by a Kalman Filter, which takes the *Model* and its covariance matrix $\boldsymbol{\Sigma}$ as prediction and updates it with the new features and their covariances (for more details refer to [7]). In order to guarantee costant-time performance, we constrain the model's maximum size. If the model grows beyond a certain upper bound, the oldest features are discarded and overwritten with the new ones.

The images are streamed at QVGA resolution and processed in the on-board computer. The visual odometry runtime is shown in Fig. 3, consists of two parts: feature extraction and motion estimation. The average processing time is 12.3 ms with a maximum of 43 ms and a standard deviation of 2.5 ms.

B. Sensor Fusion

The output pose of the visual odometry is sent through the serial interface to the HLP, where is fused with IMU data at a rate of 1 Khz. The high frequency KF's output are fed into the controller, enabling stable flight. As in [6], we cascade an Alpha Beta Filter $(\alpha\beta F)$ and a Kalman Filter (KF). The $(\alpha\beta F)$ runs on the on-board computer and provides a smoother evaluation of its input data (without an actual probabilistic analysis). We use it to reduce the

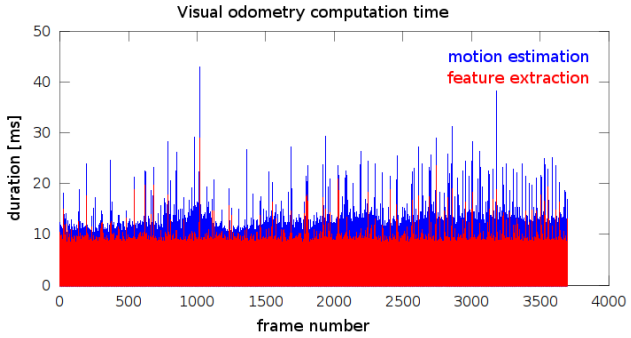

Fig. 3. Onboard computer Processing duration for each incoming QVGA image.

noise in the first estimation of the velocity, which is a simple derivative of the visual odometry position data. Hence, the output of the $(\alpha\beta F)$ is sent over serial interface and serves as correction in the KF. In this indoor application we assume that the quadrotor moves with low velocity and following quasi-rectilinear path and in-place rotations. This assumption allows us to decouple the axis in the KF design and to compute the linear acceleration relative to the fixed frame, from the IMU reading, as explained in [1]. We use three smaller KF's for each position axis as well as a KF for yaw. The discrete state space linear model of the KF is:

$$\boldsymbol{x_k} = \boldsymbol{A}_k \cdot \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \cdot \boldsymbol{u}_k \tag{3a}$$

$$\boldsymbol{z}_k = \boldsymbol{H}_k \cdot \boldsymbol{x}_k \tag{3b}$$

where the state, input and measurement for x (and similary y and z) are:

$$\boldsymbol{x} = [x\ v_x]^T \quad \boldsymbol{u} = [a_x] \quad \boldsymbol{z} = [x_{vo}\ v_{x_{vo}}]^T \tag{4}$$

while for yaw we have:

$$\boldsymbol{x} = [\psi] \quad \boldsymbol{u} = [\omega_z] \quad \boldsymbol{z} = [\psi_{vo}] \tag{5}$$

where a is the linear acceleration detected by the IMU, expressed with respect to the fixed frame and ω, the angular velocity. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{H}$ of the system in (3) for x, y and z are:

$$\boldsymbol{A} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} \frac{\Delta T^2}{2} \\ \Delta T \end{bmatrix} \quad \boldsymbol{H} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{6}$$

and for yaw:

$$\boldsymbol{A} = [1] \quad \boldsymbol{B} = [\Delta T] \quad \boldsymbol{H} = [1] \tag{7}$$

Fig. 4 shows the ouput result of the KF for the $x-$component of the linear velocity.

V. Control

The control system provides position and velocity control separately for each axis. It is based on a cascade structure of two loops, where the inner loop is provided by the Low level Controller (LLC) implemented in the LLP. It controls roll, pitch, yaw-rate and thrust (RPYT). The outer controller generetes RPYT commands to the inner loop controller. Roll

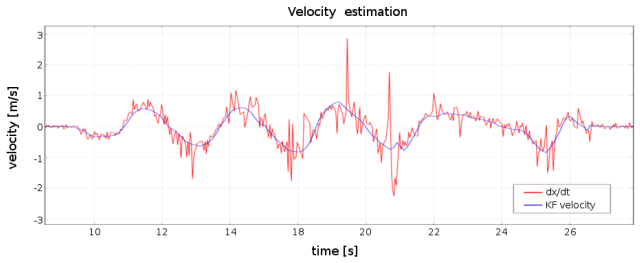

Fig. 4. Linear x velocity estimation output of the Kalman Filter compared to the rough estimation by position derivative. We obtained similar results for y and z.

and pitch commands are generated from the outer loop $x-$ $y-$ controllers as reference to the inner attitude controller. Similarly, thrust and yaw rate commands are generated from the height and yaw controller, respectively. The position controller is based on a modified PID, while velocity and yaw controller are PI and P controllers, respectively, as explained in [6].

VI. REAL-TIME VISUAL SLAM

We developed a visual keyframe-based SLAM system. The algorithm runs in real time on-board the quadrotor in a separate thread. The SLAM algorithm takes as input the pose of the quadrotor provided by the visual odometry and generates a sequence of RGB-D *keyframes*. Each keyframe consists of a RGB and Depth image pair, together with the pose of the camera at that instant and a set of SURF features detected in the RGB image. A new keyframe is generated once the angular or linear distance traveled between the current pose and the pose of the latest keyframe exceeds a given threshold (for example, 0.3 meters or 20 degrees). Incoming keyframes are tested for associations against previous keyframes. An association between two keyframes occurs when they are observing the same scene. This is accomplished in three steps. First, for the incoming keyframe, we build a set of *candidates* from the set of previous keyframes. Candidates are keyframes whose poses are close enough to be associated with the new keyframe. We use a liberal pruning threshold (for example, 5 meters and 90 degrees). Next, we train a descriptor matcher from all the SURF (Speeded Up Robust Features) keypoints in the candidate frames. The descriptor matcher is based on a FLANN (Fast Library for Approximate Nearest Neighbors) search tree [18]. We use the tree to further limit the candidate keyframes, based on the number of nearest neighbors each feature in the incoming keyframe has in each of the candidate keyframes. We keep only the k top candidates. For each of the remaining candidates, we perform robust RANSAC (RANdom SAmple Consensu) [8] matching of the SURF features. If the RANSAC algorithm finds enough geometric inliers, we assume there is an association between the two keyframes. The association observation is the transformation which best aligns the inliers.

Once the associations are established, we build a graph whose nodes are keyframe poses and whose edges are association observations. For consecutive keyframes, the observation comes from the visual odometry. Additional associations are generated through the RANSAC matching described above. Using g^2o [15], we find the configuration of poses which minimizes the observation error accross the whole graph.

The procedure runs at a rate between 1 Hz and 2 Hz onboard the quadrotor.

The keyframes are used to build a dense Octomap [25] which can be used for path-planning.

VII. 4DOF PATH PLANNING

This section introduces a quadrotor path planner in x, y, z and yaw directions. This implementation is based on a search approach where the state space is discretized using a state lattice of motion primitives [20] and an incremental and anytime version of the A* algorithm with Euclidean distance heuristic. This module has been tested in simulation in real-time in combination with the rest of the systems presented in this paper, on an identical computer as the CPU onboard the MAV.

A. State space discretization

The quadrotor state space is discretized following a state lattice, a graph search space that integrates motion planning constraints within state exploration. In this case, the state space is four-dimensional, combining the quadrotor position in Euclidean space (x, y, z) with the yaw orientation ψ. State space exploration is executed following a set of motion primitives. Motion primitives are short, kinematically feasible path segments, that can be combined together to produce longer and more complex paths. Any combination of motion primitives yield a path that complies to the non-holonomic constraints imposed by the motion planning problem. Motion primitives are pre-computed, and their traversal cost is multiplied by a user selected weight to obtain the motion cost. Weights are assigned to each motion primitive, in order to model preferences of one primitive over the others, e.g., penalizing changes in altitude while moving forward, in order to keep next positions centered within sensors field of view. Collision checking is performed online while exploring the search graph.

Planning results are greatly affected by motion primitives selection, in terms of planner times, planner completeness, and resulting path quality. The planner can not obtain a feasible path if it cannot be produced by a combination of available motion primitives. For example, backwards paths cannot be generated if backwards motion primitives are not pre-computed and made available in the set. A richer set of motion primitives improves state space coverage adding flexibility to the planner, but there is a trade-off in computation time, as each new motion primitive increases the branching factor at each state.

B. Search algorithm

The described state lattice is explored using a graph search algorithm. This algorithm is a variant of the A*

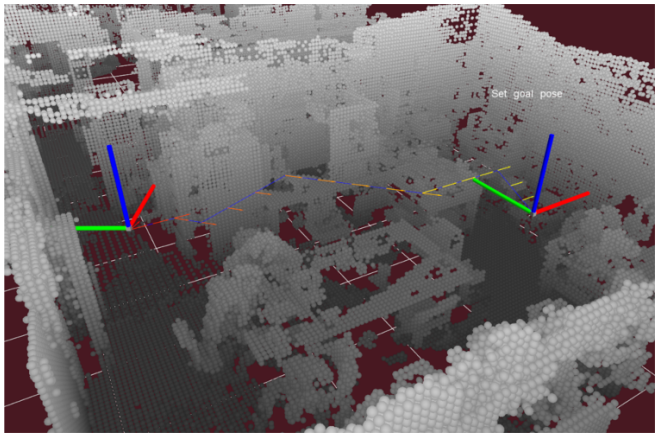

Fig. 5. Four-dimensional path (blue) in a cluttered indoor environment. Path starts from actual quadrotor pose (left reference frame) to a user selected goal pose (right reference frame). Intermediate quadrotor poses are shown along the path (colored arrows).

search extended with anytime and incremental capabilities called ARA* (Anytime Repairing A*) [16]. ARA* anytime capability is obtained by executing a series of A* searches where the heuristic is inflated by a factor $\epsilon > 1$, and reducing this factor on each execution. With an inflated heuristic, A* search gives more relevance to the heuristic estimation. This results in a faster algorithm by means of losing optimality, but it has been shown that the computed path sub-optimality is bounded to ϵ times the cost of the optimal solution [16]. ARA* starts with a high ϵ value in order to obtain a feasible path very fast. If time is available, ϵ is decreased and a search is executed again reusing computation from previous search. If enough time is available to reduce ϵ to 1, the heuristic is not inflated anymore, and the last search returns the optimal solution.

The motion primitives used in our implementation complies with a state lattice discretization of 0.25m per cell of the 3D Euclidean space, and $\pi/4$rad for yaw orientation θ. A typical path query takes 283ms average in a single core (maximum time allowed is 500ms) until the optimal path is obtained, in an indoor environment 30x30x5m in size at 0.25m resolution. For larger environments, more motion primitives, or finer space resolution, obstacle free paths can still be obtained before reaching the optimal path (ϵ = 1) within the 500ms time budget. An example of the 4D path obtained in a cluttered indoor environment is shown in Fig. 5.

VIII. EXPERIMENTAL RESULTS

To evaluate the functionality of our system, we performed several experiments in automous flight where waypoints were sent through an off-board workstation. In the first experiment shown in Fig. 6, we commanded the quadrotor to hover in place for a time of 100 seconds. In the experiment of Fig. 7, the quadrotor changed its x and y position after sending a sequence of waypoints. Both experiments prove the effectiveness of state estimation and control with a maximum error of 20 cm or less. Fig. 8 demonstrates the 3D SLAM

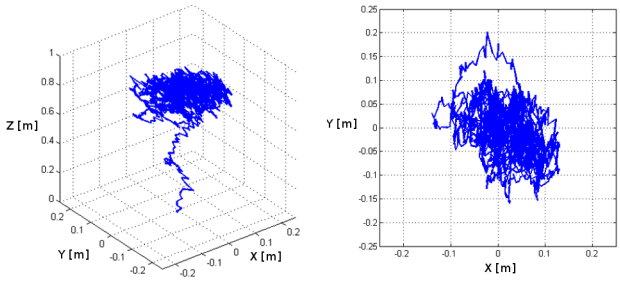

Fig. 6. Control performance in a hovering experiment over 100s. 3D view (left) and top view (right)

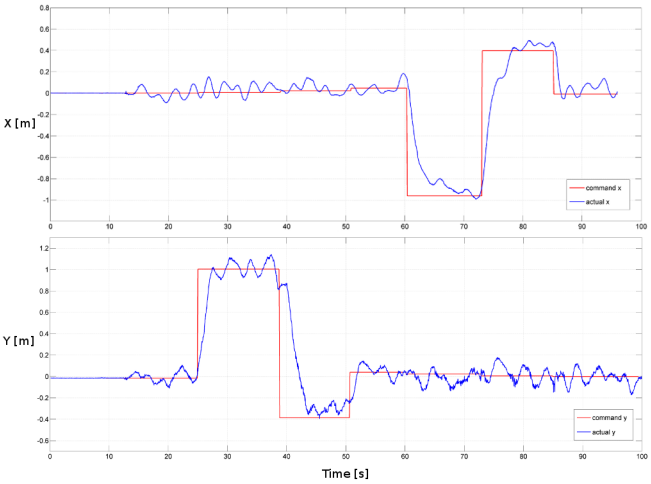

Fig. 7. Position control with a sequence of waypoints, varying x and y.

capability of the system. The quadrotor flew autonomously in a large room, with all the computation carried out on-board. The 3D SLAM algorithm receives pose data from the visual odometry and generates a sequence of RGB-D keyframes as explained in Sec. VI. The SLAM algorithm tests association between the incoming keyframe and the previous ones to provide correction of the quadrotor trajectory while building a 3D map. The video is provided as a real-time demonstration of the autonomous flight.

IX. CONCLUSIONS

In this paper, we describe an autonomous navigation system for a quadrotor based on our recently developed RGB-D visual odometry algorithm. We show how the use of the RGB-D camera as the only exteroceptive sensor enables 3D SLAM in autonomous flight in indoor environments. The powerful on-board computer is able to run all the components in real time. We also developed a 4DOF path planner whose functionality has been verified by simulation and it will be implemented on-board the quadrotor as future work. In addition we will tackle some challenging problems related to RGB-D sensors which have limited range and improve the visual odometry algorithm in featurless environment.

Fig. 8. Results of the real-time onboard SLAM experiment. Left: orthogonal projection of the recovered point cloud map. The path generated by the visual odometry and the corrected SLAM path are shown. Middle: side view of the point cloud map. Right: side view of the octomap.

REFERENCES

[1] Markus Achtelik, Michael Achtelik, Stephan Weiss, and Roland Siegwart. Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 3056–3063, May 2011.

[2] Abraham Bachrach, Samuel Prentice, and Nicholas Roy. RANGE - Robust Autonomous Navigation in GPS-denied Environments. *Journal of Field Robotics*, 28:644–666, 2011.

[3] M Blosh, S Weiss, D Scaramuzza, and R Siegwart. Vision based MAV navigation in unknown and unstructured environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 21–28, May 2010.

[4] Andrea Censi. An ICP variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19–25, Pasadena, CA, May 2008. Ieee.

[5] Joseph Conroy, Gregory Gremillion, Badri Ranganathan, and J. Sean Humbert. Implementation of wide-field integration of optic flow forautonomous quadrotor navigation. *Autonomous Robots*, 27(3):189–198, August 2009.

[6] Ivan Dryanovski, Roberto G Valenti, and Jizhong Xiao. An Open-Source Navigation System for Micro Aerial Vehicles. *Autonomous Robots*, pages 1–12, 2013.

[7] Ivan Dryanovski, Roberto G Valenti, Jizhong Xiao, and Senior Member. Fast Visual Odometry and Mapping from RGB-D Data. In *IEEE Int. Conf. on Robotics and Automation*, volume 10031, 2013.

[8] Martin A. Fischler and Robert 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.

[9] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor MAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4557–4564. Ieee, October 2012.

[10] Ascending Technologies GmbH. http://www.asctec.de.

[11] S. Hrabar, G.S. Sukhatme, P. Corke, K. Usher, and J. Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a UAV. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3309–3316. Ieee, 2005.

[12] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Dieter Fox, and Nicholas Roy. Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. *International Symposium on Robotics Research (ISRR)*, pages 1–16, 2011.

[13] Andrew Johnson, James Montgomery, and Larry Matthies. Vision Guided Landing of an Autonomous Helicopter in Hazardous Terrain *. In *International Conference on Robotics and Automation*, number April, 2005.

[14] Michael Kassecker, By Teodor Tomi, Korbinian Schmid, Philipp Lutz, Elmar Mair, Iris Lynne Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Research Platform for Indoor and Outdoor Urban Search and Rescue. *Robotics & Automation Magazine, IEEE*, 19(SEPTEMBER):46–56, 2012.

[15] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613, 2011.

[16] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *in advances in neural information processing systems 16: proceeding of the 2003 conference (NIPS-03*. MIT Press, 2004.

[17] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, pages 1–19, 2012.

[18] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP09),*, pages 331–340, 2009.

[19] Jongho Park and Youdan Kim. Stereo Vision Based Collision Avoidance of Quadrotor UAV. In *International Conference on Control, Automation and Systems*, pages 173–178, 2012.

[20] Mihail Pivtoraiko and Alonzo Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3231–3237. IEEE, 2005.

[21] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, number Figure 1, 2009.

[22] Eduardo Rondon, Luis-Rodolfo Garcia-Carrillo, and Isabelle Fantoni. Vision-based altitude, position and speed regulation of a quadrotor rotorcraft. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 628–633. Ieee, October 2010.

[23] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[24] John Stowers, Michael Hayes, and Andrew Bainbridge-Smith. Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor. In *IEEE International Conference on Mechatronics*, pages 358–362. Ieee, April 2011.

[25] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, USA, May 2010.

[26] Zhenyu Yu, Demian Celestino, and Kenzo Nonami. Development of 3D Vision Enabled Small-scale Autonomous Helicopter. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2912–2917, October 2006.

[27] Z Zamudio, R Lozano, J Torres, and V Rosas. Stereo vision for the stabilization of a quadrotor. In *System Theory, Control and Computing (ICSTCC), 2012 16th International Conference on*, pages 1–6, 2012.

[28] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. MAV navigation through indoor corridors using optical flow. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3361–3368. IEEE, 2010.