Vision-based localization, mapping and control for autonomous MAV: EuRoC challenge results

Julien Marzat[1*], Julien Moras[1], Aurélien Plyer[1], Alexandre Eudes[2], Pascal Morin[2]

1. ONERA – The French Aerospace Lab, F-91123, Palaiseau, France
2. ISIR-UPMC CNRS UMR 7222, Paris, France
*Corresponding author: julien.marzat@onera.fr +33180386650

Abstract

This paper presents vision-based localization, mapping and control solutions for autonomous navigation of a micro-air vehicle (MAV) in GPS-denied environments. The proposed algorithms have been evaluated in the simulation contest of the FP7 project EuRoC (European Robotics Challenges) dedicated to "Plant Servicing and Inspection", where our team composed of members from ONERA and ISIR-UPMC ranked 2nd over 21 European research laboratories and has been selected for the next stages of the project.

I. Introduction

The EU-funded FP7 project EuRoC aims at enhancing the use of robotics in industry via realistic demonstrations. In particular, one of the challenges focuses on "Plant Servicing and Inspection" using micro-air vehicles (MAV). Three stages are scheduled from 2014 to 2017. The first one, which took place in 2014, was a simulation competition open to European teams so as to select challengers for the next stages. The second stage will involve five selected teams to perform realistic MAV experiments related to the challenge topics in 2015 and 2016. Finally, the third stage will take place in 2017 at an actual end-user industrial site to demonstrate the MAV capabilities for future large-scale deployment.

This paper describes the solutions proposed by our joint team ONERA/ISIR-UPMC (*Eiffel Team*), to address the simulation challenge. This challenge was divided into the following tasks, covering the whole MAV estimation and control loop.

- Task 1: vision-based localization using only stereo images and IMU data (Section II).
- Task 2: environment 3D model reconstruction (occupancy grid) using stereo data and true camera localization (Section III).
- Task 3: state estimation and control with disturbance rejection (wind) (Section IV).
- Task 4: waypoint guidance with obstacle avoidance (Section V).

In what follows, the strategies developed for each task are summarized and illustrations from the challenge evaluation results are provided. Final scores are reported in Table I. The evaluation was based on accuracy, computational efficiency and robustness to uncertainty.

II. Vision-based ego-localization

A. Related Works

Vision-based ego-localization has reached a high level of maturity in the last decade. From a methodological point of view, two approaches are often opposed despite recent convergent trends: Visual Odometry (VO) and Visual Simultaneous Localization and Mapping (V-SLAM). Basically, VO estimates the relative motion of the camera between t_k and t_{k+1} by the camera pose at t_{k+1} with respect to 3D reference data, for instance a cloud of 3D points, recorded in the camera frame at t_k. From relative motion information, the full trajectory can be estimated by simple dead-reckoning or by fusion with inertial measurements. In contrast, V-SLAM addresses the problem of self-localization through the building of a globally-consistent map of the environment. This map is usually a sparse representation made of a limited number of landmarks, often 3D points. Camera positioning parameters (position and orientation) and positions of 3D landmarks are estimated jointly according to a criterion based on 2D projection error by filtering techniques like Extended Kalman Filter (EKF, [2]) or by multi-view optimization methods like Bundle Adjustment (BA, [10], [12]). The two estimation strategies differ notably in the frequence of map updating: at each frame for the EKF-based solutions or at key-frames for the optimization-based solutions.

B. Embedded Visual Odometry

Task "Vision based localization" is addressed by our vision-only algorithm eVO [14] (for "embedded visual odometry"). Designed to process image pairs acquired by a calibrated stereorig, eVO combines characteristics of both approaches described above. Indeed, eVO builds a map of isolated landmarks, as in VSLAM approach, which is updated in a keyframe-scheme as proposed in [10], [12]. Oriented towards low computational cost rather than optimality, some simplifications concerning the

TABLE I
EUROC SIMULATION CONTEST: EVALUATION RESULTS

Task 1: vision-based localization				
Scenario difficulty	Average translational drift (%)	Average processing time (ms)	Score	
			our	max
Low	0.8	31	7	8
Medium	1.6	36	9	10
High	2.1	33	6	8

Task 2: 3D environment mapping				
Scenario difficulty	Matthews correlation coefficient	Corrected processing time (ms)	Score	
			our	max
Low	0.816	0	7	8
Medium	0.771	3.2	7.5	10
High	0.823	4	7	8

Task 3: state estimation and stabilization					
Scenario	Position error(m)	Angular velocity error (rad/s)	Reenter time (s)	Score	
				our	max
Hovering	0.018	0.013	x	7.5	9
Constant wind	0.03	0.09	0.0	3.5	4.5
Wind gust	0.046	0.018	6.43	7	9

Task 4: waypoint navigation					
Scenario	Settling time (s)	Position error (m)	Actuator work (Ws)	Score	
				our	max
Obstacle free	15.103	0.037	2173	4.5	6
Switching sensors	9.530	0.031	1735	6	9
With obstacles	49.635	0.048	6889	10.5	15

map updates have been made. (1) As the generation of a long-term consistent map - useful to detect vehicle visiting again a previously seen area - is computationally very costly, we adopt the standard solution consisting in pruning the landmarks when they leave the sensor field of view (as in [12]). (2) In standard approaches, the positions of visible landmarks are refined at each key-frame by minimizing a multi-view reprojection criterion by bundle-adjustment methods. This step was dropped because of the limited computational capacity of the embedded PC. The landmarks are then localized once, the first time they are seen, in the global frame thanks to the current estimated pose.

In practice, eVO is structured by two macro-functions: state estimation and map management (Figure 1). For more details, please refer to [13].

- The **localization thread** provides the current pose (position and attitude) by tracking the landmarks stored in the map through the images acquired by the left camera. Feature tracking is done thanks to an efficient implementation of the KLT algorithm [16]. The pose is robustly inferred from the subsequent 2D-3D matching in a two-stage process. (1) RANSAC combined with the Perspective-3-Points algorithm [4] estimate the most acceptable pose in regard to the 2D-3D matchings and a set of inliers. (2) The pose is refined by minimizing the reprojection error over the inlier matches. This nonlinear least-squares optimization is solved using the motion-only optimization functions provided in Lourakis SBA code [11].
- The **Mapping thread** is in charge of adding/pruning landmarks in the 3D map. Adding new landmarks requires to detect interest points in the left image, based on Harris points [16]. These image features are matched with the right image. Inspired by dense stereovision algorithms, this operation is done by exhaustive search along the epipolar lines in a multi-scale way. The relative-to-sensor state is computed by triangulation. Finally, the localization in the global frame[1] is obtained by combination of the relative positioning and the current estimated pose. A pruning mechanism takes into account the inlier/outlier classification of RANSAC. A counter records how many times a landmark is successively classified as outlier. When the counter is higher than a threshold (3 by default), the landmark is pruned.

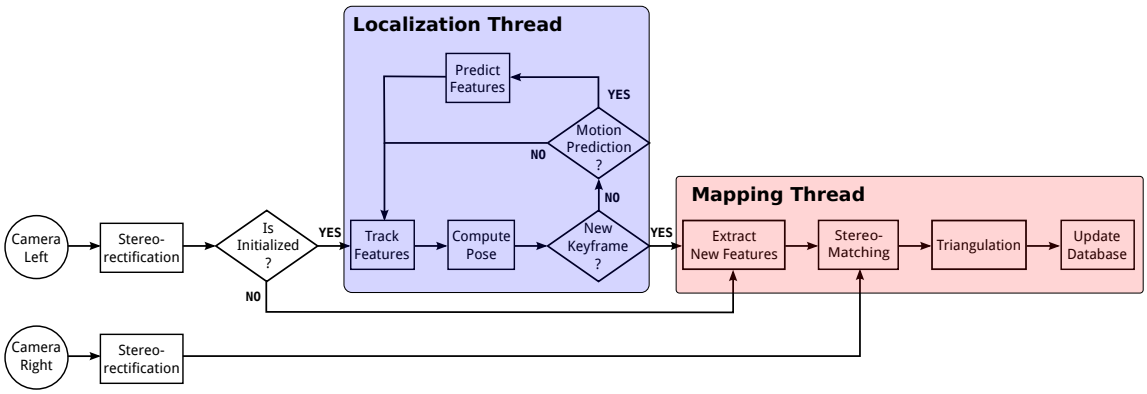

Fig. 1. Architecture of eVO algorithm where the two cooperative threads are highlighted by the two colored blocks. Each step is described in details in [14]

[1]defined as the frame of the reference sensor at the launch of the algorithm. The reference sensor is either the left camera in vision-only scenario or IMU if available.

C. Evaluation

The performance of eVO was originally evaluated on various datasets, including the Kitty benchmark [5]. Here we present the results obtained on the EurRoC dataset. The localization function is evaluated by drift indicators in translation (expressed in percentage by meter). Scores are averaged on all possible sub-sequences with length ranging from 2 m to 75 m. Figure 2 shows the evaluation and the estimated trajectory on one of three available datasets. Datasets differ each others in the number of features, illumination conditions, motion blur strength, rotational velocity. Processing times are computed on Intel i7 820QM but the evaluation virtual machines use only 2 cores.

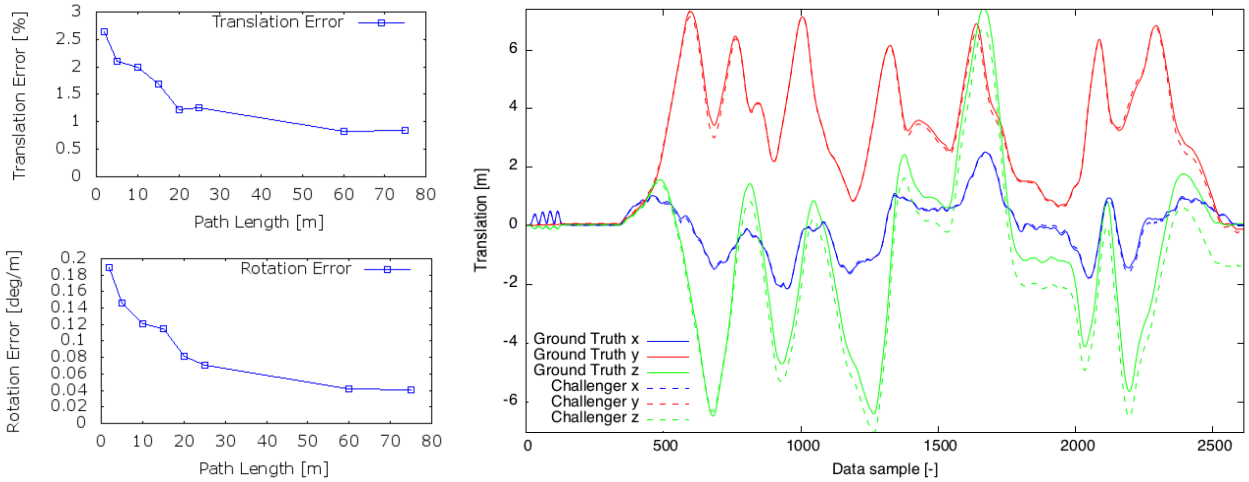

Fig. 2. EuRoC Task 1: vision-based localization

III. 3D ENVIRONMENT MODELING

On-line environment modeling is an essential capability for robotic applications in cluttered environments, since this information is used to detect and avoid obstacles. The environment model is incrementally built by adding instantaneous sensor-to-scene distance measurements, using the current vehicle state estimate to convert local measurements into global ones. In order to evaluate the sole environment mapping capability independently from the vision-based localization one, the stereo sequences were provided by the organizers of the EuRoC challenge with the true camera localization.

A. Depthmap computation

Depthmaps are computed from rectified stereo pair by a state-of-the-art dense stereovision algorithm, ELAS [2], proposed by Geiger et al. [6]. This algorithm proceeds in two steps: (1) the robust matching of image features, called support points (2) the computation of a dense disparity by probabilistic propagation of depth estimated on support points. We use the standard "Robotics" parameters profile.

B. 3D volumetric modeling

Proposed in the seminal work of Elfes [3], the occupancy grid have become the standard for 2D and 3D environment modeling in mobile robotics. In case of 3D modeling, a standard approach consists in subdividing the workspace in cubic volume elements of equal size called voxels.

We use a well-known open-source implementation of a volumetric occupancy grid using an octree data structure, called Octomap [7]. Each element of this data structure contains two probabilities, the occupancy one and the free one. These quantities are updated by a ray-tracing technique emulating a depth sensor: the ray between the sensor and a 3D point increases the free probability of intersected voxels while the occupancy probability of the end-point voxel is increased. Using octree offers a very efficient memory structure but the update strategy is very computationally intensive. Only low framerate (near 2Hz, with depthmaps at VGA resolution) are available due to this limitation. Figure 3 shows the 3D model obtained with our solution.

[2]available online at http://www.cvlibs.net/software/libelas/

C. Processing optimization and evaluation

The evaluation is relative to two criteria: (1) the completeness of the environment model, measured with Matthews correlation coefficient of correctly/incorrectly detected free and occupied voxels, and (2) the computational time. Three optimizations have been made in order to minimize this last criterion. First, a key-frame selection mechanism was added to reduce the number of depthmap computations by avoiding points of view too close to each other. The selection criterion mixes angular and linear distances between the current view and the set of (previously identified) key-frames. After depthmap conversion to a 3D-point cloud (in sensor frame), the 3D points which are too close to each other are suppressed by applying a voxel filter with a resolution of 5 cm (half the voxel grid resolution imposed for map evaluation). This operation reduces drastically the number of insertions in the Octomap with a minimal impact on modeling precision. Finally, Octomap is configured with a maximal distance of 5 m in order to avoid insertions of noisy data and reduce the number of updated voxels per insertion.

The evaluation scenarii differed in illumination conditions, motion blur strength and scene depth variation. The impact of these parameters are limited as shows the completeness results in Table I. The processing times are corrected by taking into account latencies caused by evaluation process (using two virtual machines).

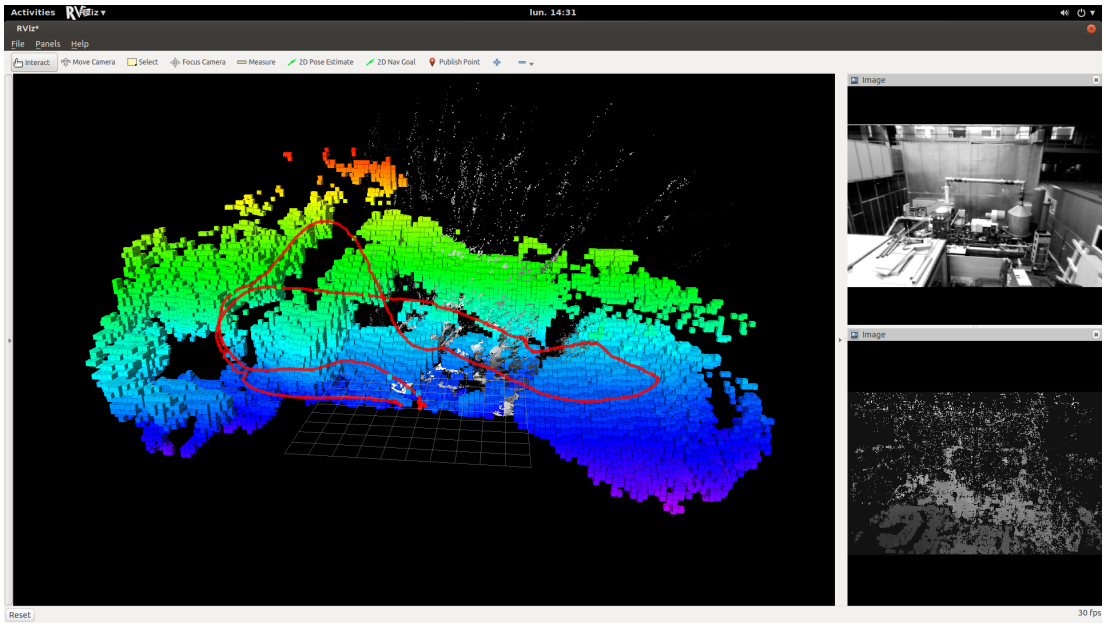

Fig. 3. EuRoC Task 2.1: occupancy grid construction

IV. State estimation and control with disturbance rejection

The aim of this task is to demonstrate MAV stabilization capabilities with visuo-inertial input in different wind condition (no wind, constant wind and wind gust). This task is performed in a simulated environment defined in Gazebo. The wind model is a single force applied to the MAV center of mass. As the evaluation of localization and mapping was done in the previous tasks, the visual sensor is simulated here.

A. Modeling

This section provides the MAV and sensor models as well as mathematical notation used to describe the estimator and controller in the following sections.

We denote $\mathcal{I}$ the inertial frame, $\mathcal{B}$ the body frame (same as the IMU frame), $\mathcal{C}$ the visual sensor frame, $O_\bullet$ is the origin of $\bullet$ frame. The position and velocity in $\mathcal{I}$ are $\{p, v\}$, $R = {}^{\mathcal{I}}R_{\mathcal{B}}$ is the rotation from body to inertial frame, ω the angular velocity in body frame, $a_s = R^T(\dot{v} + ge_3)$ the specific acceleration.

Let $S(a)$ be the matrix of cross product associated with vector a and vex the inverse operation such that $a \times b = S(a)b$ and $a = vex(S(a))$ and $P_a(M)$ be the anti-symmetric part operator of a matrix M such that $P_a(M) = \frac{1}{2}(M - M^T)$.

For control and estimation purpose, the MAV is considered as a rigid body of mass m and inertia $\mathbb{I}$ under the action of three forces and a torque: gravity($-ge_3$ in $\mathcal{I}$), disturbance(mF) in inertial frame and the propellers force and torque ($TmRe_3, \Gamma$), all of them applied at the center of mass. With this notation, one then obtains the following kinematic model:

$$\left\{ \begin{array}{rcl} \dot{R} & = & RS(\omega) \\ \mathbb{I}\dot{\omega} & = & -S(\omega)\mathbb{I}\omega + \Gamma \end{array} \right. \qquad \begin{array}{rcl} \dot{p} & = & v \\ \dot{v} & = & -ge_3 + TRe_3 + F \\ & = & Ra_s - ge_3 \end{array} \tag{1}$$

The visual sensor provides a pose measurement at 10Hz in its own frame: ${}^{\mathcal{I}}R_{\mathcal{C}}, {}^{\mathcal{I}}p_{O_{\mathcal{I}},O_{\mathcal{C}}}$. Since the pose of the MAV is known before takeoff in this simulation contest, we use this extra information to estimate the missing change of frame [3] between $\mathcal{I}$ and $\mathcal{C}$. In order to simplify notation, the pose of the MAV estimated by the visual sensor (from $\mathcal{B}$ to $\mathcal{I}$) is noted R_v, p_v thereafter. The IMU gives us three measurements: the angular velocity $w_m = w + w_b$, the specific acceleration $a_m = a_s + a_b$ and a measurement of the orientation $R_i = {}^{\mathcal{I}}R_{\mathcal{C}}$ at high frequency (100 Hz). Figure 4 gives an overview of the proposed solution.

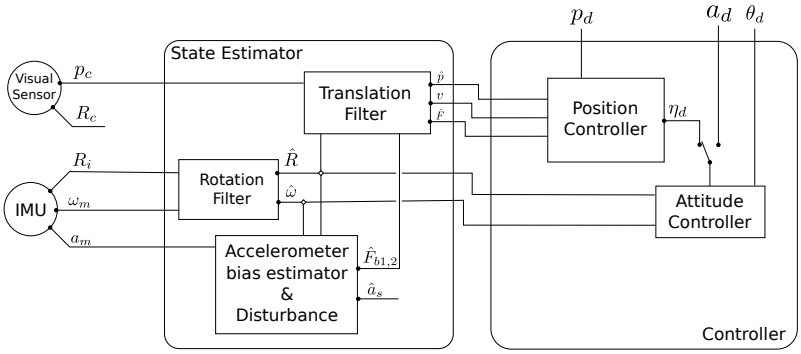

Fig. 4. Architecture of the state estimator and controller.

B. Controller

The multi-rotor is an under-actuated vehicle with only 4 control inputs (one force and three torques). The control of such a vehicle consists in changing its orientation to control position and velocity. For more information on the definition of the controller, please refer to [8].

The controller is a standard two-stage controller with attitude and stabilization loop. This separation is valid here especially since the simulator gives us a fast and reliable estimation of the attitude and angular velocity. The first loop is the attitude loop and is responsible for the stabilization of the thrust direction. This loop needs measurements of the attitude and the angular velocity. It also allows switching to an external mode where the input of desired acceleration in inertial frame is used instead of the stabilization loop for point-to-point navigation with obstacle avoidance (see Section V).

The second loop is the stabilization loop, which regulates the position at a given reference. This control loop needs the estimation of position, velocity and disturbance as an input and provides a desired thrust orientation.

C. State estimation

The purpose of this state estimator is to recover the pose and velocity at a high enough frequency for the controller and also estimate the disturbance force. This estimator should also take into account accelerometer and gyrometer biases. This state estimator is based on a complementary filtering approach derived from the one defined in [15]. Figure 4 sums up the estimation process and shows inputs and outputs of the three stages (rotation filter, accelerometer bias and disturbance estimator, translation filter).

First, we describe the attitude estimation part. In this simulation, two sensors provide direct measurements of the orientation of the MAV. On the one hand, the orientation provided by the simulated IMU is a high quality signal (high frequency, very slow drift, reasonable noise). On the other hand, the simulated visual sensor is very slow and has a large amount of noise, with a larger variance than the drift of the IMU orientation measurements. Therefore, we use the IMU (R_i) as our main source of attitude and only filter it for noise reduction purposes. This part of the state estimation is also in charge of the gyro bias estimation which is achieved by a complementary filter with a down-sampling of R_i at 10Hz as measurement of R. The following system (Eq.2) summarizes this first estimator:

$$\begin{cases} \dot{\hat{R}}_i &= k_R \hat{R}_i P_a(\hat{R}_i^T R_i)) & \dot{\hat{\omega}}_b &= -k_{\omega_b} vex(P_a(\hat{R}^T R)) \\ \dot{\hat{R}} &= \hat{R}(S(\omega - \hat{\omega}_b) + k_R P_a(\hat{R}^T R)) & \hat{\omega} &= \omega_m - \omega_b \end{cases} \tag{2}$$

Before describing the two other observers, let us introduce how the disturbance is estimated from accelerometers. To achieve a fast response when a disturbance occurs, the disturbance is evaluated directly from accelerometer measurements. From

[3] In real application, this unknown change of frame could be estimated with a calibration procedure as in [15]

kinematic equations of the model and accelerometer $\dot{v} = -ge_3 + TRe_3 + F = Ra_s - ge_3$, one deduces that the accelerometer measurement is equal to $a_s = Te_3 + F_b$ with $F_b = R^T F$ and that the accelerometers directly measure the first two components of the disturbance in the body frame $F_{b,12}$. This disturbance is used directly in the control loop, to counteract external forces. The last component $a_{m,3} = T + F_{b,3}$ cannot be directly used in the controller because T is one of our command. Therefore, we rely on the state estimator for the evaluation of $F_{b,3}$.

The second part of the state estimator is related to accelerometer filtering. Here we recover the accelerometer bias with a complementary filter (Eq.3) with the visual position p_v as measurement of p. We also add a first-order filter on the accelerometer measurements to limit the amount of noise propagated in other stages for disturbance evaluation ($\hat{F}_{b,12}$).

$$\begin{cases} \dot{\hat{p}}_a &= \hat{v}_a - k_p(\hat{p}_a - p) & \dot{\hat{a}}_b &= -k_b(I_3 + \frac{1}{k_p}S(\omega))\hat{R}^T(\hat{p}_a - p) \\ \dot{\hat{v}}_a &= -ge_3 + R(a_s - \hat{a}_b) - k_v(\hat{p}_a - p) & \dot{\hat{F}}_{b,12} &= -k_f(\hat{F}_{b,12} - a_{s,12} + \hat{a}_{b,12}) \end{cases} \tag{3}$$

The last part of this state estimator focuses on estimating the translation motion (p,v) at a higher rate and the last component of the disturbance. This estimator (Eq.4) is a complementary filter based on the model of the multi-rotor and the position p_v estimated by vision as a measurement of p. For the estimation of the last component of the disturbance, a model of constant disturbance in inertial frame is used($\dot{F} = 0$ or $\dot{F}_b = -S(w)\dot{F}_b$) and estimated in the complementary filter.

$$\begin{cases} \dot{\hat{p}} &= \hat{v} - k_p(\hat{p} - p) & \dot{\hat{F}}_{b,3} &= e_3^T(-S(w)\hat{F}_b - k_F\hat{R}^T(\hat{p} - p)) \\ \dot{\hat{v}} &= -ge_3 + T\hat{R}e_3 + R\hat{F}_b - k_v(\hat{p} - p) & \hat{F} &= \hat{R}\hat{F}_b \end{cases} \tag{4}$$

D. Evaluation results

The simulation contest evaluated results of each challenger proposal with an automatic scoring procedure on three different scenarii. The first scenario is standard hovering at 1 m. The two others scenarii test the rejection of disturbances, one with static wind (Heaviside step) and the other with a wind gust (rectangle). Three metrics are used for this task. Position and attitude accuracy are evaluated by the position error defined as $e_p = rms(p_{12}) + rms(p_3)$ and the angular velocity error ($e_\omega = rms(\omega)$) with $rms()$ the root mean square function. The last criterion is used only on scenarii with disturbance and measures the time after which the MAV reenters a ball of radius 1 m around the stabilization position after the disturbance occurred. The scoring is established with respect to the validation of criteria, for example to obtain the maximum score on the first scenario, the solution must have a $rms(\omega) < 0.1$ rad/s and a residual error on position $e_p < 0.1$ m. Figure 5 gives an example of the system behavior during a wind gust. The proposed framework achieves a position RMS error of less than 5 cm and an angular velocity RMS under 0.1 rad/s in hovering mode and reject disturbance on less than 10 s.

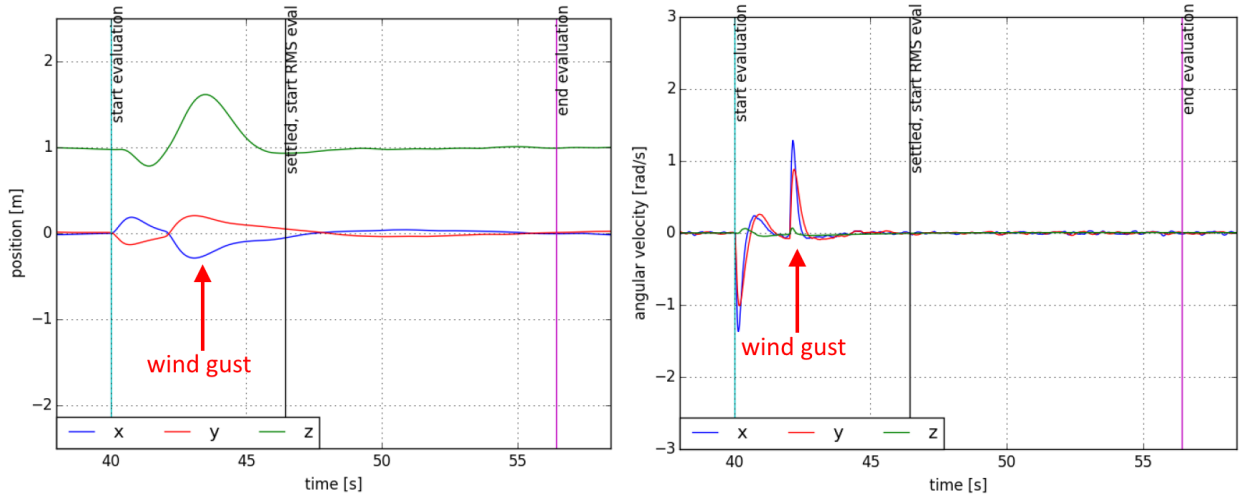

Fig. 5. EuRoC Task 3: state estimation and control with disturbance rejection

V. Waypoint guidance with obstacle avoidance

This task addresses waypoint navigation with obstacle avoidance in a cluttered environment, simulated in Gazebo. Waypoints were not known in advance, thus the trajectory had to be determined on-line with a limited computational cost. An occupancy map of the environment similar to the one built in Task 2 was provided to detect collisions (Figure 6). In this context, we used a model predictive control (MPC) scheme to autonomously define the safe exploration trajectories [1].

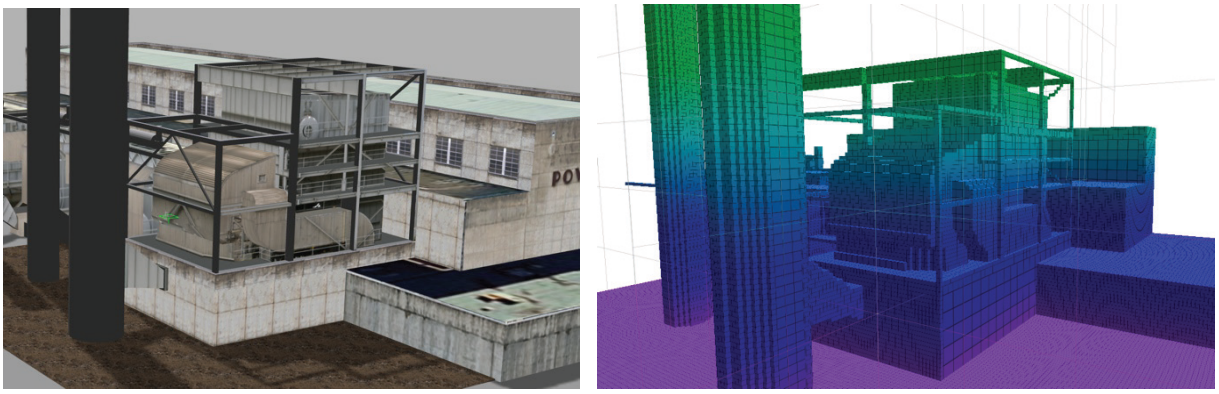

Fig. 6. EuRoC Task 4: Gazebo simulated environment and corresponding occupancy map

A. MPC principles

Model Predictive Control (MPC) is a usual method for the guidance of autonomous vehicles in complex environments, taking into account differential constraints [17]. Convergence results for this receding horizon strategy can be found in [9]. A dynamical model of the system is used to predict its future state on a finite time horizon and a multi-modal performance criterion is optimized at each time step for computing control inputs. Here, the prediction model is the following discrete-time three-axis kinematic model in inertial frame

$$\begin{array}{rcl} p_{k+1} & = & p_k + \Delta t \cdot v_k \\ v_{k+1} & = & v_k + \Delta t \cdot u_k \\ u_k & = & -g e_3 + T_k R_k e_3 \end{array} \tag{5}$$

where $u_k \in \mathbb{R}^3$ is the acceleration control input to be computed and Δt the guidance timestep (0.5 s in the simulation). Considering the current estimated MAV state $x_k = [p_k^T, v_k^T]^T$, a sequence $\mathbf{U}_k$ of H_c control inputs is defined as well as the resulting sequence $\mathbf{X}_k$ of H_p predicted states using this model.

$$\mathbf{U}_k = \{u_k, u_{k+1}, \ldots, u_{k+H_c-1}\} \tag{6}$$

$$\mathbf{X}_k = \{x_{k+1}, x_{k+2}, \ldots, x_{k+H_p}\} \tag{7}$$

Finite control horizon $H_c = 2$ and prediction horizon $H_p = 20$ were considered for tractability. Since $H_c < H_p$, control inputs at timesteps larger than H_c were considered null. Each control input vector u_k is bounded within the compact set $\mathcal{U}$ as $u_{\min} < u_k < u_{\max}$ and $\mathbf{U}_k \in \mathcal{U}^{H_c}$. A cost function $J(\mathbf{U}_k, \mathbf{X}_k)$ is defined to quantify the mission requirements and constraints. The following optimization problem is then solved at each timestep k to find the optimal control sequence.

$$\begin{array}{c} \mathbf{U}_k^* = \arg \min_{\mathbf{U}_k \in \mathcal{U}^{H_c}} J(\mathbf{U}_k, \mathbf{X}_k) \\ \text{with } x_i \text{ satisfying (5)}, \\ \forall i \in [k+1; k+H_p] \end{array} \tag{8}$$

The first component u_k^* of this sequence is then applied on the MAV and the procedure is repeated at the next timestep using the updated state estimate.

B. MPC costs

The main cost function is defined as

$$J = w_u J_u + w_{\text{nav}} J_{\text{nav}} + w_{\text{obs}} J_{\text{obs}} \tag{9}$$

where J_u limits the acceleration requirements, J_{nav} is the waypoint navigation cost and J_{obs} the obstacle avoidance cost. These sub-costs $J_\bullet$ are all smaller than the unit norm and the weights $w_\bullet$ are chosen to reflect their relative importance: w_{nav} is an order of magnitude larger than w_u, while w_{obs} is several orders of magnitude larger than the other weights to prevent any collision.

The control cost penalizes large accelerations on the control horizon: $J_u = \dfrac{1}{H_c} \displaystyle\sum_{i=k}^{H_c-1} \|u_k\|$.

The reference trajectory between the current position p_k and the next waypoint p_w is decomposed into reference points $p_{k,w}^i$ ($i \in [k+1, k+H_p]$). They correspond to positions that the MAV would reach at timestep i if moving along a straight line to p_w at nominal velocity v_{nom},

$$p_{k,w}^i = p_k + (i-k)\Delta t v_{\text{nom}} \frac{p_k - p_w}{\|p_k - p_w\|}. \tag{10}$$

The navigation cost is then given by

$$J_{\text{nav}} = \sum_{i=k+1}^{k+H_p} \|p_i - p^i_{k,w}\| \tag{11}$$

where p_i represents the MAV position at timestep i predicted using (5). The length of the prediction horizon is chosen so as to compensate for the observed distance to the next waypoint. When the distance to the waypoint falls below a pre-defined distance (1 m), the MAV switches to the position controller described in Section IV.

The obstacle avoidance cost penalizes the intersection of each predicted position p_i with existing obstacles in the current occupancy grid. This is achieved by considering a set of N_{obs} discretized test positions (54 in our simulation) located on three concentric spheres of radius d_s, $2d_s/3$, $d_s/3$ centered on p_i, with d_s a safety distance related to the MAV size. The occupancy grid is tested at each of these locations and a penalty is added each time an obstacle is encountered. The obstacle avoidance cost is computed as

$$J_{\text{obs}} = \frac{1}{H_p N_{obs}} \sum_{i=k+1}^{k+H_p} \sum_{j=1}^{N_{obs}} \mathcal{O}\left(p_i + p_j\right), \tag{12}$$

where p_j is the tested location with respect to the center of the spheres and $\mathcal{O}(p)$ returns 1 if an obstacle is present at p, otherwise 0. To reduce the computational cost, it is possible to interrupt the test loops when a single obstacle is found and to apply a binary penalization to the entire trajectory.

C. Online computation of control inputs

As the computational cost should be reduced to cope with the embedded resources, we limit the search to a finite set $\mathcal{S}$ of candidate control sequences. The distribution of the candidate control sequences is chosen so as to limit their number while providing a good coverage of the control space, as follows: (i) the extremal control inputs are included to exploit the full vehicle potential, (ii) the null control input is included and (iii) candidates are distributed over the entire control space with an increased density around the null control input for precise local control. More details on how $\mathcal{S}$ is built for this three-axis acceleration model can be found in [1]. This resulted in a total of 640 possibilities to be evaluated at each timestep.

D. Evaluation

The evaluation comprised three subtasks of growing complexity: waypoints without obstacles, with a switch in pose measurements and finally navigation in the cluttered environment from Figure 6. The evaluation metrics were similar to the previous task: settling time to reach the waypoint, accuracy of hovering once settled, as well as the workload of the motors. No collision occurred in the last subtask, which was mandatory otherwise a null score would have been attributed. Examples of trajectories with obstacle avoidance are shown in Figure 7.

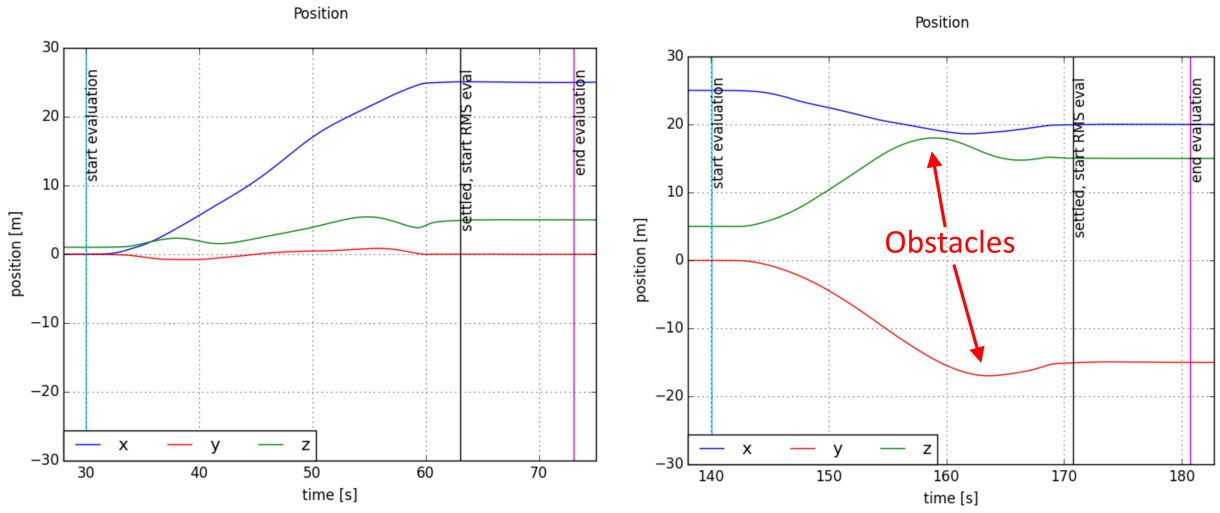

Fig. 7. EuRoC Task 4: waypoint guidance with obstacle avoidance

VI. Conclusions and perspectives

This paper has reported the localization, mapping and control algorithms developed by the *Eiffel Team* between ONERA and ISIR-UPMC to address EuRoC challenge. Accurate results were obtained for all tasks within required computation time, which confirmed the interest of the proposed algorithms. The final score combining all tasks was equal to 82.5 points (for a maximum of 104.5, see Table I), which allowed us to rank 2nd over the 21 competing teams. Following these simulation results, the ONERA-ISIR team, in association with RTE, has been selected as one of the five teams to compete in the next stage of the EuRoC project. This will involve real-scale demonstrations of the vision-based localization, mapping and control loop described in this paper, which is currently being embedded on actual multi-rotor MAVs.

Acknowledgements

The authors would like to thank the other participants to this research project for their contribution: Martial Sanfourche, Sylvain Bertrand, Hélène Piet-Lahanier, Bruno Hérissé, Guy Le Besnerais and Minh Duc Hua.

References

[1] S. Bertrand, J. Marzat, H. Piet-Lahanier, A. Kahn, and Y. Rochefort. MPC strategies for cooperative guidance of autonomous vehicles. *AerospaceLab Journal*, (8):1–18, 2014.

[2] A. Davison, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(6):1052–1067, 2007.

[3] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

[4] 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.

[5] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354 – 3361, Providence, RI (USA), June 2012.

[6] A. Geiger, M. Roser, and R. Urtasun. Efficient large-scale stereo matching. In *Proceedings of the 10th Asian Conference on Computer Vision (ACCV), Queenstown, New Zealand, November 8-12*, pages 25–38, 2010.

[7] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[8] M.-D. Hua, T. Hamel, P. Morin, and C. Samson. Introduction to feedback control of underactuated VTOL vehicles: A review of basic control design ideas and principles. *IEEE Control Systems Magazine*, 33(1):76–88, 2013.

[9] A. Jadbabaie, J. Yu, and J. Hauser. Unconstrained receding-horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 46(5):776–783, 2001.

[10] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *International Symposium on Mixed and Augmented Reality*, Nara, Japan, November 2007.

[11] M. A. Lourakis and A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

[12] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 363–370, 2006.

[13] M. Sanfourche, A. Plyer, A. Bernard-Brunel, and G. Le Besnerais. 3DSCAN: Online ego-localization and environment mapping for micro aerial vehicles. *AerospaceLab Journal*, (8):1–17, 2014.

[14] M. Sanfourche, V. Vittori, and G. Le Besnerais. evo: A realtime embedded stereo odometry for mav applications. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, November 3-8*, pages 2107–2114, 2013.

[15] G. G. Scandaroli, P. Morin, and G. Silveira. A nonlinear observer approach for concurrent estimation of pose, IMU bias and camera-to-IMU rotation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, USA, September 25-30*, pages 3335–3341, 2011.

[16] J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593 – 600, 1994.

[17] L. Singh and J. Fuller. Trajectory generation for a UAV in urban terrain, using nonlinear MPC. In *Proceedings of the American Control Conference, Arlington, VA, USA*, volume 3, pages 2301–2308, 2001.