IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2014)
September 18, 2014, Chicago, Illinois, USA

Embedded vision-based localization and model predictive control for autonomous exploration

Hélène Roggeman, Julien Marzat, Martial Sanfourche, Aurélien Plyer

***Abstract*— This paper presents a complete mobile robot architecture for autonomous exploration in a GPS-denied unknown environment. The platform considered is equipped with wheel encoders, stereo-vision and depth sensors, the measurements of which are fused within an extended Kalman filter for robust localization. An occupancy grid of the environment is built on-line for environment reconstruction and obstacle detection. Based on this map, a model predictive control scheme autonomously defines safe exploration trajectories, while taking into account interaction with the imaging sensors. Experimental results demonstrate the embedded computational capability of this vision-based control loop.**

***Index Terms*— autonomous robot, exploration, model predictive control, robust estimation, visual odometry**

I. Introduction

Autonomous robots are interesting contenders to carry out surveillance and exploration missions in unknown or dangerous environments. The algorithms embedded on these platforms should be able to simultaneously achieve mapping, localization, trajectory definition and control. Much research has been conducted on mobile robots equipped with either global localization sensors (GPS, Vicon systems) or relative ones such as a laser range finder, which is expensive and consumes a lot of energy. An alternative choice is to rely mainly on visual sensors, since recent progress in embedded computational capabilities now allows fast image processing and three-dimensional environment modeling. Such a vision-based architecture has been shown to be practical on a Micro Air Vehicle for an exploration mission in [1].

Vision-based ego-localization has reached a high level of maturity in the last decade with many applications in aerial robotics or mobile robotics. From a methodological point of view, two approaches are often opposed despite recent convergent trends: Visual Odometry (VO) ans V-SLAM (Visual Simultaneous Localization and Mapping). Basically, VO estimates the relative motion of the camera between two successive images [2], [3] (monocular, stereo or depth image) while V-SLAM resolves the global localization problem by building a globally-consistent maps of the environment [4], [5]. However, in a recent movement of convergence, VO algorithms tend to be like V-SLAM algorithms which the constraint of globally consistent would be released. For a more detailed review, we advise the reading of the recent two-parts tutorial of D. Scaramuzza and F. Fraundorfer [6], [7].

H. Roggeman, J. Marzat, M. Sanfourche, A. Plyer are with ONERA – The French Aerospace Lab, F-91123 Palaiseau, France, firstname.lastname@onera.fr

Model Predictive Control (MPC) is an appealing control strategy to build and follow trajectories in unknown environments. It uses a dynamical model of a system to predict its future state on a time horizon. Using this prediction, a possibly multi-modal performance criterion is optimized at each timestep for computing control inputs that achieve the required goals [8]. Unlike most path planning methods, this scheme is able to take into account an accurate system model as well as environment changes, since new control inputs are computed on the basis of measurements acquired in real time. This is why it has been used in a few recent works dealing with exploration by mobile robots [9]–[12].

This paper details an embedded vision-based MPC coupled loop to address autonomous exploration of an unknown area by a mobile robot (see model in Section II), with obstacle avoidance. The algorithmic architecture is presented in Figure 1. The visual odometry algorithm (eVO) is described in Section III-A, an EKF filter for fusion of wheel and visual odometries in Section III-B and the environment mapping strategy in Section IV. The MPC autonomous guidance scheme is explained in Section V and experimental results are reported in Section VI.

II. Robot architecture and model

The experimental platform considered is a four-wheel Robotnik Summit XL (Figure 2) equipped with a Kinect sensor and a stereo rig composed of two electronically synchronized USB cameras separated by a 18cm long baseline equipped with 5.5mm S-mount lens. The cameras are configured to capture VGA frames at 20Hz. The stereo image flow is processed to estimate the robot trajectory and to provide depth maps in outdoor environments while Kinect gives depth maps in indoor environment. These sensors are linked to an embedded PC with an Intel quad-core i7 processor and a Nvidia GPU (GT640) in charge of data processing.

The state vector for this system is $\mathbf{x} = [x, y, \theta]^T$, where (x, y) is the robot position and θ its heading angle. The control input vector is $\mathbf{u} = [v, \omega]^T$, where v is the robot linear speed and ω its angular speed. These are related to the controlled rotation speeds of the wheels by

$$\begin{cases} v = \frac{r}{2}(\omega^l + \omega^r) \\ \omega = \frac{r}{2L}(\omega^r - \omega^l) \end{cases} \tag{1}$$

where L is the half-axis length and r the wheel radius. The discrete-time dynamical model $\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$ can be

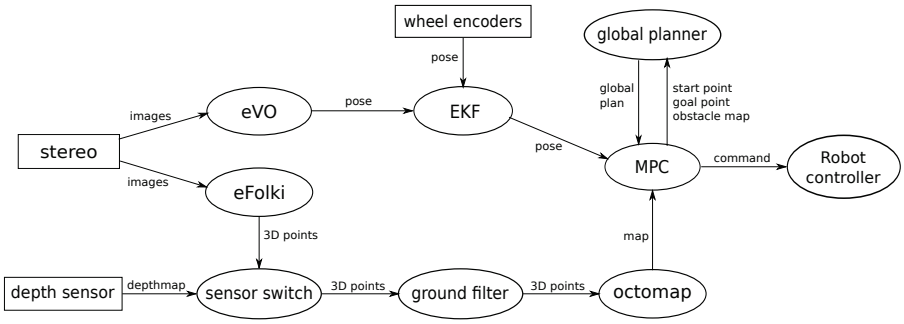


Fig. 1. Algorithmic architecture

written as

$$\begin{cases} x_k = x_{k-1} + t_e v_{k-1} \cos\theta_{k-1} \\ y_k = y_{k-1} + t_e v_{k-1} \sin\theta_{k-1} \\ \theta_k = \theta_{k-1} + t_e \omega_{k-1} \end{cases} \quad (2)$$

where t_e is the sampling period.

Fig. 2. Our mobile robotic platform equipped with two visual sensors (see text).

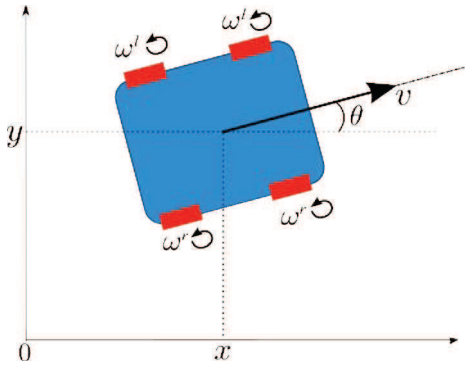


Fig. 3. Robot motion model.

III. Vision-based localization

A. Stereo Visual SLAM

Estimating the robot trajectory from its starting point relies on a stereo visual SLAM algorithm, called eVO [13]. Visual SLAM addresses the problem of ego-localization through the construction of a consistent map of the environment without prior information. Here we consider a map built from a limited number of 3D points located in a common frame defined as the robot frame at its starting point. Thanks to the calibrated stereo setup, the scale factor is known and the depth of landmarks in the sensor frame is measurable in a limited range (around 10 meters).

As in [5], the implemented algorithm is based on two tasks working in parallel: mapping and localization.

The mapping task consists in localizing new landmarks and discarding those which have gone out of sight. Except for the first stereo pair used to initialize the map, this operation is executed on-demand when the ratio of visible landmarks over the number of landmarks stored in the map falls below a threshold. The stereo pairs used to update the map are called *keyframes*. In practice, for each keyframe, a few hundred of Harris [14] or FAST [15] corners are extracted in the left image then matched by a coarse-to-fine exhaustive search along the epipolar lines. The relative-to-sensor localization of the novel landmarks is easily deduced from the disparity value. In order to be stored in the map, the landmarks are localized in the reference frame by applying the transformation corresponding to the current robot pose. In contrast to [1], [5], [16], the multiview refinement assuring a more precise landmark localization, inherent to SLAM techniques, were bypassed for computational reasons linked to the limited computational power of the original target robot. The map is then a collection of 3D points localized from a limited number of points of view. As shown on practical examples in [13], this scheme limits the estimation drift compared to standard dead-reckoning visual odometry techniques and delivers a pose at higher frequency. This operation takes approximately 20 ms on the embedded PC.

The localization task exploits unambiguous matchings between landmarks stored in the map and image features. These 2D-3D matchings are initialized during the mapping operation, then they are propagated along the visible landmark by tracking image features with KLT [14]. The full 6-degree-of-freedom pose (position and attitude) is deduced

from a two-steps procedure: the initial pose is estimated with the P3P algorithm using a RANSAC procedure [17], [18], which is then refined by minimizing the reprojection error of inlier matches selected by RANSAC. This task is carried out by considering only the left image. This operation is very fast (less than 10ms on the embedded PC).

This algorithm was benchmarked one year ago on the KITTI datasets [19], where it ranked first at the time of the submission. The measured drift on various sequences (KITTI or MAV) varies between 1 % and 2% of the trajectory length. In some intricate situation - the observed scene lacks of texture or when the robot navigates in highly dynamic environments-, the estimated trajectory can suddenly differ largely from the true trajectory. The proposed countermeasure consists in fusing by an EKF the pose parameters corresponding to a planar motion, denoted by $\mathbf{y}^{\text{evo}} = [x^{\text{evo}}, y^{\text{evo}}, \theta^{\text{evo}}]^{\text{T}}$ in what follows, with measurements coming from the wheel encoders.

B. EKF sensor fusion with outlier detection

As in [20], an extended Kalman filter (EKF) is used to cope with the nonlinear dynamical model (2). It provides an estimated state $\widehat{\mathbf{x}}$ and its associated covariance matrix $\mathbf{P}$ through the fusion of wheel encoders and stereovision measurements. It also allows outlier detection if the vision-based system faces momentarily an unstructured scene.

The estimated vehicle input vector, denoted by $\widehat{\mathbf{u}} = [\widehat{v}, \widehat{\omega}]^{\text{T}}$, can be computed from the wheel encoder measurements (averaged between the two wheels of each side) using (1). The prediction step of the filter based on this input information is thus equivalent to classical wheel odometry

$$\widehat{\mathbf{x}}_{k|k-1} = f\left(\widehat{\mathbf{x}}_{k-1|k-1}, \widehat{\mathbf{u}}_{k-1}\right) \tag{3}$$

The EKF framework also makes it possible to take into account the uncertainty related to wheel odometry (mostly due to wheel slip) in the input covariance noise matrix

$$\mathbf{Q}_k = \begin{bmatrix} \sigma_v^2(k) & 0 \\ 0 & \sigma_\omega^2(k) \end{bmatrix} \tag{4}$$

The noise variances σ_v^2 and σ_ω^2 can be chosen either constant or proportional to the squared angular speeds of the wheels, since a larger error is to be expected with higher rates. The propagation of the state covariance is achieved by

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^{\text{T}} + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^{\text{T}} \tag{5}$$

where

$$\mathbf{F}_{k-1} = \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\widehat{\mathbf{x}}_{k-1|k-1}, \widehat{\mathbf{u}}_{k-1}} \tag{6}$$

$$\mathbf{G}_{k-1} = \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\widehat{\mathbf{x}}_{k-1|k-1}, \widehat{\mathbf{u}}_{k-1}} \tag{7}$$

In the case of model (2), these matrices are equal to

$$\mathbf{F}_{k-1} = \begin{bmatrix} 1 & 0 & -t_e \widehat{v}_{k-1} \sin\widehat{\theta}_{k-1} \\ 0 & 1 & t_e \widehat{v}_{k-1} \cos\widehat{\theta}_{k-1} \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$\mathbf{G}_{k-1} = \begin{bmatrix} t_e \cos\widehat{\theta}_{k-1} & 0 \\ t_e \sin\widehat{\theta}_{k-1} & 0 \\ 0 & t_e \end{bmatrix} \tag{9}$$

The correction step of the EKF uses the visual odometry measurements $\mathbf{y}_k^{\text{evo}} = [x_k^{\text{evo}}, y_k^{\text{evo}}, \theta_k^{\text{evo}}]^{\text{T}}$, such that the innovation is equal to

$$\mathbf{r}_k = \mathbf{y}_k^{\text{evo}} - \widehat{\mathbf{x}}_{k|k-1} \tag{10}$$

and its covariance is

$$\mathbf{S}_k = \mathbf{P}_{k|k-1} + \mathbf{R}_k \tag{11}$$

where $\mathbf{R}_k = \text{diag}\left(\sigma_x^2, \sigma_y^2, \sigma_\theta^2\right)$, with noise variances related to the accuracy of eVO mentioned in Section III-A.

Finally, the updated state and covariance are

$$\widehat{\mathbf{x}}_{k|k} = \widehat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{k|k-1}\mathbf{S}_k^{-1}\mathbf{r}_k \tag{12}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I}_3 - \mathbf{P}_{k|k-1}\mathbf{S}_k^{-1}\right)\mathbf{P}_{k|k-1} \tag{13}$$

The vision-based system may generate large localization errors if the scene is not structured enough (e.g., a plain wall). If undetected, this may cause the output of the EKF to be erroneous and as a consequence endanger the robot mission and safety. Following sensor fault diagnosis techniques, a robust outlier detection scheme is considered by monitoring the innovation (10). If it exceeds some threshold $\boldsymbol{\epsilon}_{\mathbf{r}}$, which can be selected proportional to the innovation covariance (11), then only the prediction step is keeped as the state estimate (i.e., classical wheel odometry). The visual odometry system is then re-initialized at each timestep with the new predicted state, until the innovation falls below the threshold. The filter then proceeds back to the update step.

Algorithm 1 Robust EKF

At each timestep k

1) Compute $\widehat{\mathbf{u}}_{k-1}$ from wheel encoders
2) Predict $\widehat{\mathbf{x}}_{k|k-1}$ and $\mathbf{P}_{k|k-1}$ using (3) and (4)
3) Using stereovision measurements $\mathbf{y}_k^{\text{evo}}$, Compute the innovation using (10) and (11)
4) **if** $\mathbf{r}_k > \boldsymbol{\epsilon}_{\mathbf{r}}$
   Use $\widehat{\mathbf{x}}_{k|k-1}$ as the estimated state and reset eVO
   else
   Compute the update $\widehat{\mathbf{x}}_{k|k}$ and $\mathbf{P}_{k|k}$ from (12) and (13)
5) Go back to 1 with $k \leftarrow k+1$

The behavior of the EKF is illustrated in Figure 4 on real data. It can be seen that the filter successfully rejects visual odometry outliers due to the unstructured scene (see detection signal in Figure 5) and also corrects the drift of wheel odometry when visual information is available.

IV. Online environment mapping

Online environment mapping consists in aggregating 3D measurements, in particular depthmaps, in a global model thanks to the poses estimated by the EKF previously described. Two kinds of depthmap are considered here : those captured by the depth active camera end those calculated from stereo pair by the dense optical flow algorithm called eFolki [21].

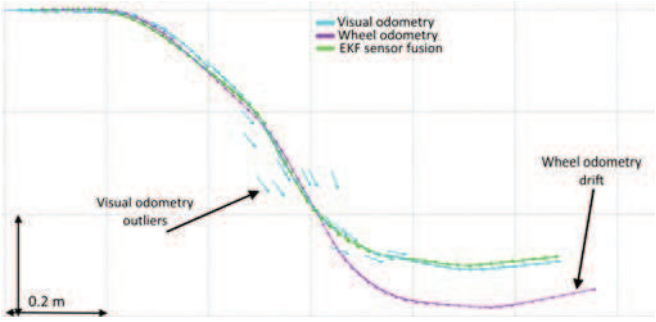


Fig. 4. Robust EKF sensor fusion (grid step size 1m)

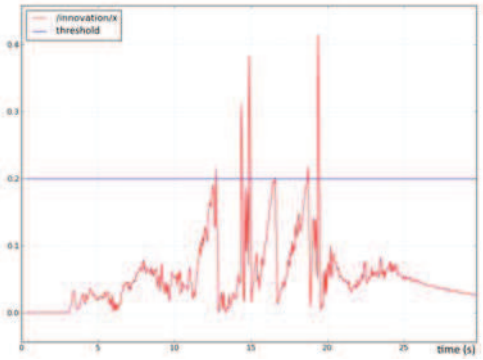


Fig. 5. Outlier detection via the innovation signal

A. Dense stereo-matching

eFolki is an evolution of the standard Lucas-Kanade [22] optical flow algorithm. Like for the dense LK algorithm, the basic problem is to register local windows centred around each image pixel $\mathbf{x}$ by minimizing a SSD (Sum of Squared Difference) criterion over a 2D motion vector $\mathbf{u}(\mathbf{x})$:

$$\sum_{\mathbf{x}'} w(\mathbf{x}'-\mathbf{x})\left(I_1(\mathbf{x}')-I_2(\mathbf{x}'+\mathbf{u}(\mathbf{x}))\right)^2, \qquad (14)$$

where w is a separable weighting function, uniform or Gaussian, of limited support $\mathcal{W}$, typically a square window parametrized by its radius r. The minimization of the criterion (14) is done by iterative Gauss-Newton coarse-to-fine pyramidal strategy as in a classical implementation of LK. However, using the first order expansion described in [23], an iteration can be completed with only one image interpolation per pixel, while the well-known PyramiLK algorithm [24] requires several image interpolations by pixel. The resulting code is remarkably fast on massively parallel architecture such as GPU.

The motion estimation greatly depends on the local image texture and fails in case of illumination changes. So, to achieve a high level of robustness in real-world environment, eFolki uses a Rank Transform [25] applied to the images before SSD minimization. In practice, the motion estimation in low texture areas is more noisy but the motion estimation stays convergent.

B. Multisensor occupancy grid

The default device is the depth active camera. However, in many situations especially outdoor, the depthmaps delivered by this sensor are incomplete or even empty. So, a mechanism based on the density of depthmaps acquired by active camera permits to switch automatically from active camera to passive stereorig.

The 3D point clouds output by this "sensor switch" (see figure 1) are then individually filtered for removing the ground plane thanks to a RANSAC-based search algorithm [26]. A prior is used to find the plane whose the perpendicular angle is near vertical.The result is then inserted in an Octomap model [27].

Octomap is a well-known implementation of a volumetric occupancy grid using an octree data structure. 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 uprise the free probability of intersected voxels while the end-point uprise the occupancy probabilty of its corresponding voxel. Using octree offers a very efficient memory structure but the update strategy is very computationally intensive. Only low framerate (near 1Hz) are available due to this limitation.

After the Octomap update, we project the map in a three state 2D map : {unexplored, free, occupied}, and use it for autonomous guidance.

V. Guidance for autonomous vision-based exploration

Model Predictive Control (MPC) is a usual method for the guidance of autonomous vehicles in complex environments, taking into account differential constraints [28]. Convergence results for this receding horizon strategy can be found in [29].

A. MPC principles

Considering the current robot position $\mathbf{x}_k$, 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 model (2).

$$\mathbf{U}_k = \{\mathbf{u}_k, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{k+H_c-1}\} \qquad (15)$$

$$\mathbf{X}_k = \{\mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \ldots, \mathbf{x}_{k+H_p}\} \qquad (16)$$

Finite control horizon H_c and prediction horizon H_p are considered for tractability. If $H_c < H_p$, control inputs at timesteps larger than H_c should be considered either constant (for linear speed) or null (for angular speed). Each control input vector $\mathbf{u}_k$ is bounded within the compact set $\mathcal{U}$ as

$$\begin{array}{c} -v_{\max} < v_k < v_{\max} \\ -\omega_{\max} < \omega_k < \omega_{\max} \end{array} \qquad (17)$$

and thus $\mathbf{U}_k \in \mathcal{U}^{H_c}$. A cost function $J(\mathbf{U}_k, \mathbf{X}_k)$ should be 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.

$$\begin{array}{c} \mathbf{U}_k^* = \arg\min_{\mathbf{U}_k \in \mathcal{U}^{H_c}} J(\mathbf{U}_k, \mathbf{X}_k) \\ \text{with } \mathbf{x}_i \text{ satisfying } (2), \\ \forall i \in [k+1; k+H_p] \end{array} \qquad (18)$$

The first component $\mathbf{u}_k^*$ of this sequence is then applied on the robot and the procedure is repeated at the next timestep using the new information gathered in the motion.

B. MPC costs

The main cost function is defined as

$$J = w_{\text{obs}} J_{\text{obs}} + w_u J_u + w_{\text{expl}} \left(b_e J_{\text{expl}} + \overline{b_e} J_{\text{nav}} \right) \quad (19)$$

where

- J_{obs} is the obstacle avoidance cost,
- J_u regulates the linear and angular speeds,
- J_{expl} is the exploration cost,
- J_{nav} the waypoint navigation cost.

J_u and J_{obs} are always active, while J_{expl} and J_{nav} correspond to separate mission phases and will never be active simultaneously: b_e is a boolean which is equal to 1 when exploration is active and to 0 when the robot should switch to waypoint navigation.

As detailed in what follows, the sub-costs $J_\bullet$ are all of unit norm. The weights $w_\bullet$ should be chosen to reflect their relative importance (see Section VI).

1) Obstacle avoidance: The obstacle avoidance cost penalizes the intersection of each predicted position in $\mathbf{X}_k$ with existing obstacles in the current occupancy grid. A morphological Euclidean distance transform is applied on the occupancy grid to obtain a distance map. This computation needs only to be performed on the area which can be reached by the vehicle on the prediction horizon, starting from its current position. The user-defined borders of the zone to be explored are also considered as obstacles.

Based on the map containing the distance of any vehicle position to the nearest obstacle, the following penalty function [30] is considered

$$f_o\left(\mathbf{x}_k\right) = \frac{1 - \tanh\left(\alpha\left(d_o\left(\mathbf{x}_k\right) - \beta\right)\right)}{2} \quad (20)$$

$$\alpha = \frac{6}{d_{\text{des}} - d_{\text{sec}}} \quad (21)$$

$$\beta = \frac{1}{2}\left(d_{\text{des}} + d_{\text{sec}}\right) \quad (22)$$

where

- $d_o\left(\mathbf{x}_k\right)$ is the distance between the vehicle position at time k and the nearest obstacle.
- d_{des} is a desired distance to obstacles, beyond which they are ignored.
- d_{sec} is a safety distance that must not be reached by the vehicle, leading to full penalty.

The continuous function f_o is designed to be equal to 1 when $d_o < d_{\text{sec}}$ and to zero for $d_o > d_{\text{des}}$ (Figure 6). The obstacle avoidance cost is computed as

$$J_{\text{obs}} = \frac{1}{H_p} \sum_{i=k+1}^{k+H_p} f_o\left(\mathbf{x}_i\right). \quad (23)$$

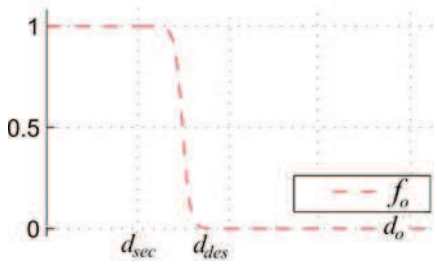


Fig. 6. Penalty function for obstacle avoidance

2) Control cost: J_u encompasses the regulation of the sequence of linear speeds to a nominal value v_0 (possibly negative) and penalization of large angular speeds, on the control horizon.

$$J_u = \frac{1}{2H_c} \sum_{i=k}^{H_c-1} \left(\frac{\omega_i^2}{\omega_{\max}^2} + \frac{w_v \left(v_i - v_0\right)^2}{\left(\|v_0\| + \|v_{\max}\|\right)^2} \right) \quad (24)$$

3) Vision-based exploration: The objective of the exploration mission is to maximize the area seen during the mission, within user-defined borders, while avoiding already explored locations to reduce the duration of the mission [12]. The definition of an exploration trajectory is tightly coupled with the characteristics of the embedded sensor: here, the vision sensor has a triangular field of view with the same heading angle as the vehicle (see Figure 7).

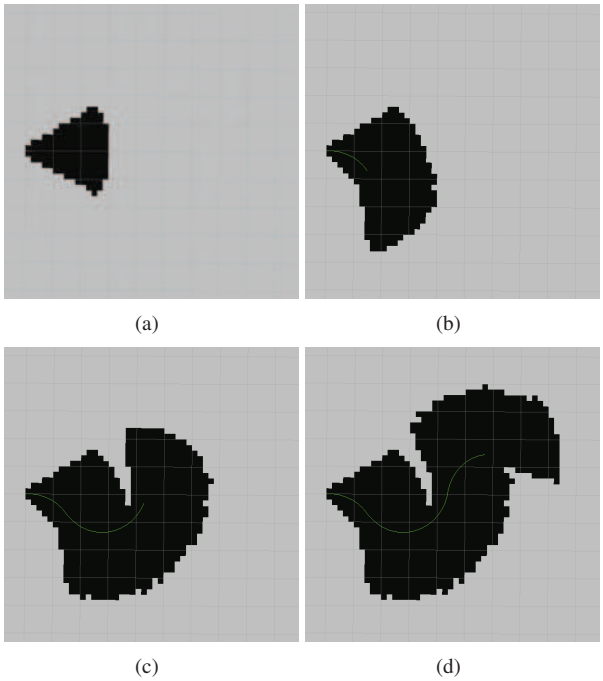


Fig. 7. Area explored (in black) for a sample trajectory (in green)

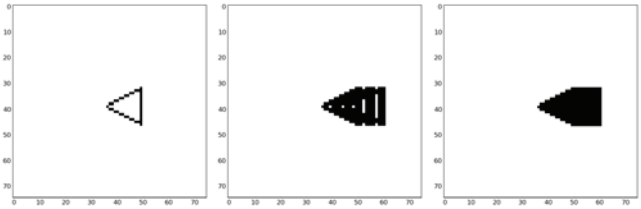
A copy of the occupancy grid is considered for recording exploration progress, obstacles being considered as explored boxes. The explored grid at timestep k is denoted by $\mathbf{G}(k)$, discretized with a uniform spatial step d_{grid} and of dimensions $l \times h$ (which could be limited to the area that can be

reached by the vehicle on the prediction horizon). Each of its components $g_{i,j}(k)$ takes the value 1 if the corresponding location has been explored and 0 otherwise, i and j being the grid coordinates.

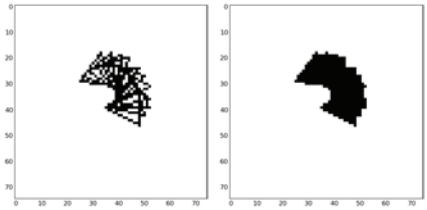
For any predicted trajectory corresponding to a control input sequence $\mathbf{U}_k$, the best exploration cost should be the one that favors the highest number of unexplored locations. To update the exploration grid, the intersection between its cells and the sensor field of view at each predicted position is computed by applying a contour detection algorithm (Bresenham line drawing) followed by a morphological closing operation on the predicted sequence (see Figure 8). The grid situation updated with a given predicted trajectory until time $k + H_p$ is denoted by $\mathbf{G}(k + H_p)$. The exploration cost to be minimized (and thus negative) is then

$$J_{\text{expl}} = \frac{d_{\text{grid}}^2}{H_p \mathcal{A}_{\text{fov}}} \sum_{i=1}^{l} \sum_{j=1}^{h} [g_{i,j}(k) - g_{i,j}(k + H_p)] \qquad (25)$$

where $\mathcal{A}_{\text{fov}}$ is the area of the sensor field of view.



(a) Two prediction steps



(b) Area covered on the prediction horizon

Fig. 8. Morphological operations for prediction of explored area, taking into account the sensor field of view

4) Navigation: The following cost is built to reach a waypoint $\mathbf{p}_w$, given the predicted positions of the robot $\mathbf{p}_i = [x_i, y_i]^{\text{T}}$,

$$J_{\text{nav}} = \frac{1}{H_p v_{\max} t_e} \sum_{i=k+1}^{k+H_p} \|\mathbf{p}_w - \mathbf{p}_i\|^2 \qquad (26)$$

If the waypoint is unreachable by the vehicle on the prediction horizon with the obstacle-free trajectory of minimal cost, i.e., $\|\mathbf{p}_w - \mathbf{p}_{k+H_p}\| > \epsilon_{\text{d}}$ with ϵ_{d} a small distance threshold, then a grid-based A* path is computed and the waypoint $\mathbf{p}_w$ is re-assigned to the farthest reachable point on this roughly planned trajectory. Another solution could consist in increasing the size of the prediction horizon to compensate for the observed distance to the waypoint.

C. Mission supervision

The MPC strategy is able to govern the exploration mission and obstacle avoidance within the reach of the prediction horizon. For large maps where the distance covered during the prediction horizon is much smaller than the size of the user-defined map to explore, a higher layer supervision process is defined.

Detection of mission ending is achieved by assessing whether the MPC cost is similar for all predicted trajectories and the optimization process results in null control inputs on successive timesteps. In this case, if there remains unexplored areas in the exploration grid, the robot switches to waypoint navigation to reach the nearest such location. Once it is reached, exploration resumes. If the entire map has been explored, then the starting point is designated as a waypoint and the mission ends when it is reached.

VI. Experiments

A. MPC implementation

As in [30], the same control input is applied on all the control horizon for computational tractability, thus the optimization problem (18) has only two variables to find, v^* and ω^*, and $\forall i \in [k; k + H_c - 1],\ \mathbf{u}_i^* = [v^*, \omega^*]^{\text{T}}$. For steps between H_c and H_p, the linear speed remains equal to v^* while the angular speed is set to zero. Tuning parameters of the EKF and MPC algorithms are indicated in Table I.

TABLE I
MPC AND EKF PARAMETERS

$w_{\text{obs}} = 40$	$t_e = 0.25$s (MPC)	$v_{0/\max} = 0.6$m/s	$\sigma_v^2 = 10^{-2}$
$w_u = 1$	$t_e = 0.05$s (EKF)	$\omega_{\max} = 0.6$rad/s	$\sigma_\omega^2 = 10^{-2}$
$w_v = 5$	$H_c = 10$	$d_{\text{grid}} = 0.2$m	$\sigma_{x/y}^2 = 10^{-2}$
$w_{\text{expl}} = 15$	$H_p = 20$	$\mathcal{A}_{\text{fov}} = 4.5\text{m}^2$	$\sigma_\theta^2 = 10^{-3}$

The deterministic global optimization algorithm DIRECT [31] was used for solving (18), using the *nlopt* package. This strategy was always able to find a result in less than 0.1s on the embedded computer, which compares favorably with the duration of the MPC timestep.

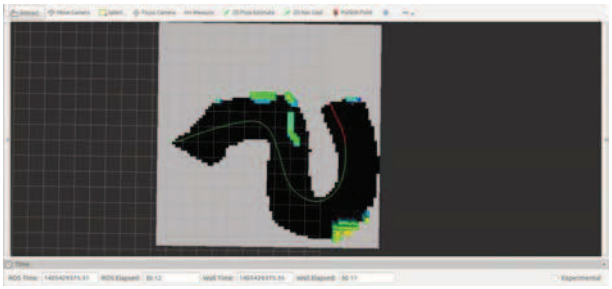
B. Preliminary Experimental results

We have experimented the proposed system in the parking of our research center. The mission consists in exploring an squared area of $15m \times 15m$.

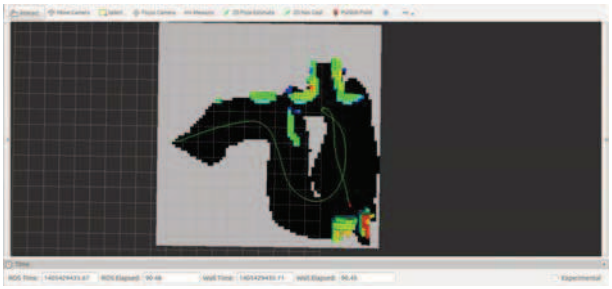
In a first step, we have validated the control functions by using only he wheel encoders. The figure 9 shows the mission progress at 4 different moments when :

- the robot goes behind an obstacle (subfigures (a) and (b)).
- the robot is blocked by an environment element (subfigure (c)). ;
- the robot selects a waypoint and switches to waypoint navigation. The waypoint is depicted by the pink point (subfigure (d)).
- the robot stops the mission (subfigure (e)).

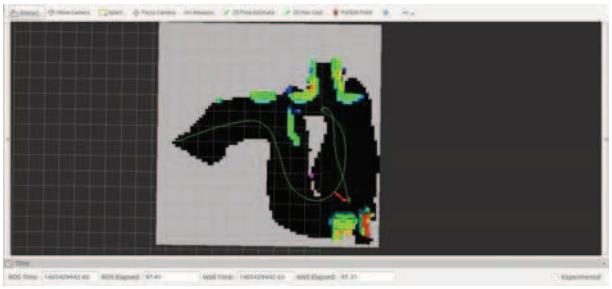
(a) Exploration behind the obstacle



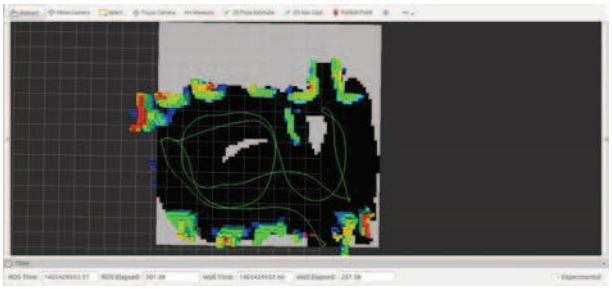
(b) Exploration behind the obstacle



(c) Robot is blocked



(d) Switch to waypoint navigation



(e) End of the mission

Fig. 9. Progress of the parking exploration at 4 different moments (Screenshot from Ros Rviz). Here, the robot localization uses only the wheel encoders.

In a second time, the robot trajectory is estimated by Visual-SLAM. The preliminary results are shown in figure 10 are equivalent to the ones obtained by wheel-odometer. In a second time, the robot trajectory is estimated by Visual-SLAM. The preliminary results are shown in figure 3 are equivalent to the ones obtained by wheel-odometer. The comparison of the second screenshot with the third one highlights a limited drift. This can be explained by a too small angular error between vision-based estimation and gyrometer-based prediction for being recovered by EFK.

VII. Conclusions and perspectives

A vision-based algorithmic architecture to tackle fully autonomous exploration missions with a mobile robot has been presented. It relies only on measurements coming from visual sensors and wheel encoders, which are fused in an EKF. The resulting state estimate is combined with visual information to build a map of the environment, which is exploited by a MPC scheme to define trajectories favouring unexplored locations without obstacle collision.

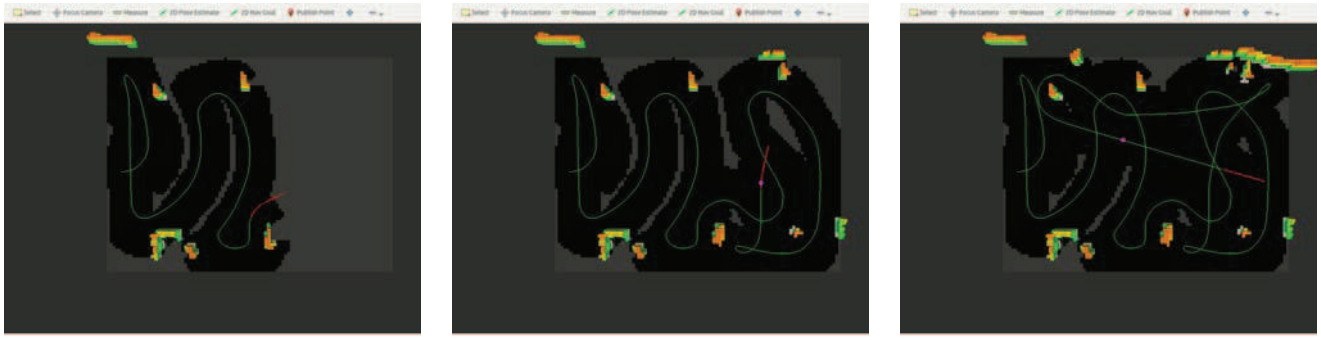
Preliminary experimental results have highlighted the interest of the approach and its fully embedded capabilities. There is room for improvement in the vision, mapping and control algorithms, which will receive further attention in the future.

In particular, the precision of the stereo visual SLAM could be improved by fusing with the IMU sensor and by updating the landmarks localization in a multi-view refinement strategy. In parallel, we are working on the environment model to increase the update rate of the 3D model.

The interaction between the visual sensors and trajectory definition could be further enhanced by selecting control inputs that lead to areas where vision-based localization would not be endangered, while here the sensor field of view was only taken into account for exploration purpose.

References

[1] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, “Vision-based autonomous mapping and



(a) (b) (c)

Fig. 10. Progress of the parking exploration at 3 different moments (Screenshot from Ros Rviz) with robot trajectory estimated by vision.

exploration using a quadrotor MAV," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal*, 2012, pp. 4557–4564.

[2] A. Mallet, S. Lacroix, and L. Gallo, "Position estimation in outdoor environments using pixel tracking and stereovision," in *IEEE ICRA*, vol. 4, San Francisco, USA, April 24-28 2000, pp. 3519–3524.

[3] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 3946–3952.

[4] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 29, no. 6, pp. 1052–1067, 2007.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the International Symposium on Mixed and Augmented Reality, Nara, Japan*, 2007, pp. 225–234.

[6] D. Scaramuzza and F. Fraundorfer, "Visual odometry: Part i - the first 30 years and fundamentals," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, December 2011.

[7] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part ii - matching, robustness, and applications," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 78–90, June 2012.

[8] R. Findeisen, L. Imsland, F. Allgower, and B. A. Foss, "State and output feedback nonlinear model predictive control: An overview," *European Journal of Control*, vol. 9, no. 2-3, pp. 190–206, 2003.

[9] C. Leung, S. Huang, and G. Dissanayake, "Active SLAM using model predictive control and attractor based exploration," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China*, 2006, pp. 5026–5031.

[10] T. M. Howard, C. J. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *Field and Service Robotics*, ser. Springer Tracts in Advanced Robotics, A. Howard, K. Iagnemma, and A. Kelly, Eds. Springer Berlin Heidelberg, 2010, vol. 62, pp. 69–78.

[11] L. Bascetta, D. Cucci, G. Magnani, M. Matteucci, D. Osmankovic, and A. Tahirovic, "Towards the implementation of a MPC-based planner on an autonomous all-terrain vehicle," in *Workshop on Robot Motion Planning: Online, Reactive, and in Real-time, The IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal*, 2012, pp. 7–12.

[12] T. Gorecki, H. Piet-Lahanier, J. Marzat, and M. Balesdent, "Cooperative guidance of UAVs for area exploration with final target allocation," in *Proceedings of the 19th IFAC Symposium on Automatic Control in Aerospace, Würzburg, Germany*, 2013, pp. 260–265.

[13] M. Sanfourche, V. Vittori, and G. L. Besnerais, "eVO: A realtime embedded stereo odometry for mav applications," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan*, 2013, pp. 2107–2114.

[14] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Jerusalem, Israel*, 1994, pp. 593–600.

[15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of the European Conference on Computer Vision, Graz, Austria*, vol. 1, 2006, pp. 430–443.

[16] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo," *International Journal of Computer Vision*, vol. 94, no. 2, pp. 198–214, 2011.

[17] 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*, vol. 24, no. 6, pp. 381–395, 1981.

[18] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[19] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA*, 2012, pp. 3354–3361.

[20] L. Teslić, I. Škrjanc, and G. Klančar, "EKF-based localization of a wheeled mobile robot in structured environments," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 2, pp. 187–203, 2011.

[21] A. Plyer, G. L. Besnerais, and F. Champagnat, "Real-time Lucas-Kanade optical flow estimation for real-world applications," *Journal of Real Time Image Processing*, 2014.

[22] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, Canada*, 1981, pp. 674–679.

[23] G. Le Besnerais and F. Champagnat, "Dense optical flow by iterative local window registration," in *IEEE International Conference on Image Processing 2005*. IEEE, 2005, pp. I–137.

[24] J. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," Technical report. Intel Corporation, Tech. Rep., 2001.

[25] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," *European Conference on Computer Vision*, pp. 151–158, 1994.

[26] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011, pp. 1–4.

[27] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[28] 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*, vol. 3, 2001, pp. 2301–2308.

[29] A. Jadbabaie, J. Yu, and J. Hauser, "Unconstrained receding-horizon control of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 5, pp. 776–783, 2001.

[30] Y. Rochefort, H. Piet-Lahanier, S. Bertrand, D. Beauvois, and D. Dumur, "Model predictive control of cooperative vehicles using systematic search approach," *Control Engineering Practice*, 2014, in press.

[31] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.