M. Sanfourche, A. Plyer,
A. Bernard-Brunel, G. Le Besnerais
(Onera)

E-mail: martial.sanfourche@onera.fr

DOI : 10.12762/2014.AL08-09

3DSCAN: Online Ego-Localization and Environment Mapping for Micro Aerial Vehicles

We present 3DSCAN (3D Scene Characterization for Autonomous Navigation), a software application for state estimation and environment modeling using low-cost 3D sensors, such as a stereorig and RGBD cameras. For state estimation, we describe an original keyframe-based stereoscopic visual odometry technique, which can run at more than 20Hz on a lightweight computer. This so-called 'efficient Visual Odometry' (eVO) has been evaluated on several datasets and provides accurate results and limited drift, even for indoor/outdoor trajectories. Environment modeling aggregates instantaneous depthmaps in a volumetric Octomap [15] representation. Stereoscopic depthmaps are computed by a very fast dense matching algorithm derived from eFolki, an optical flow code implemented on GPU. These developments are combined in the 3DSCAN software, which is successfully demonstrated on our MAV (Micro Aerial Vehicle) system, following indoor, outdoor or mixed trajectories.

Introduction

From an automation point of view, navigation consists in computing a safe and achievable trajectory and in controlling its execution. It requires a precise knowledge of both the dynamic state of the vehicle (position, attitude, speed) and of the 3D structure of the environment.

These prerequisites are very difficult to meet for a Micro Aerial Vehicles (MAV) flying low through a cluttered environment. The first difficulty arises from the partial knowledge of the environment. Despite the spread of available georeferenced information (images and maps), they still cannot be considered to be of sufficient accuracy and density for a mission in an unprepared environment such as flying around a building, avoiding unmapped obstacles, such as trees or parked cars, and entering through an open window. In this context, the vehicle should be able to build online a representation of its environment using embedded exteroceptive sensors.

The second difficulty is, of course, the very limited payload of MAV. Since 2005, demonstrations of on-line mapping have been done using heavy UAV (helicopter with weight $>$ 25 kg) [1, 42] equipped with 3D vision sensors, such as lidar. However, a high resolution (HR) inertial measurement unit (IMU) and high grade differential GPS receiver were used to localize the vehicle. The very limited payload and power of MAV precludes the embedding of lidar and HR IMU. As a result, most current demonstrations of MAV use external resources, in particular external systems, for the estimation of the MAV state, such as multi-camera localization systems.

In this paper, we present a solution for state estimation and environment modeling based on low-cost 3D sensors, compatible with indoor and outdoor environments. There are now several solutions for these 3D sensors, which allow fast mapping of obstacles and lead to a well-behaved ego-localization problem, compared to solutions based on monocular 2D sensors subject to scale ambiguity and its drift. We first describe ego-localization using a stereoscopic visual odometer and then on-line modeling of the environment, the combination of the two functions being called 3DSCAN 3D Scene Characterization for Autonomous Navigation. This 3DSCAN system is demonstrated on publicly available data Kitti [12] and through several experiments performed at Onera using our MAV. In the latter demos, only the first function of 3DSCAN (ego-localization) runs on-board, but the proposed modeling solution is already compatible with recent embedded architectures.

Related Works and Contributions

Localization and mapping are active research fields since more than twenty years and an enormous amount of literature exists. In the following, we review some relevant references on these two topics and discuss our contribution.

Ego-localization

Vision-based ego-localization has reached a high level of maturity over the last decade. For example, NASA's two Mars Exploration Rovers (Spirit and Opportunity) have been successfully using stereo visual odometry since 2003. 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). These approaches are briefly summarized below. For a much more detailed review, we advise the reading of the recent two-part tutorial by D. Scaramuzza and F. Fraundorfer [40, 11].

- 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 regard to 3D reference data, for instance a cloud of 3D points, recorded in the camera frame at tk. VO can be distinguished based on 2D-3D associations [33, 14, 29], 3D-3D associations [20], 2D associations through 3 views using the trifocal constraint [18], or through 4 views using the quadrifocal constraint [6]. From relative motion information, the full trajectory can be estimated by simple dead-reckoning [14, 29] (Dead-Reckoning Visual Odometry, DRVO), or by fusion with inertial measurements; see [22, 3, 38] in the aerial context.

- V-SLAM addresses the problem of self-localization by constructing a globally-consistent map of the environment. This map is usually a sparse representation made of a limited number of landmarks, often 3D points. State variables (i.e., ego-localization and speed) and positions of 3D landmarks are estimated according to a joint criterion based on a 2D projection error. This can be done by filtering techniques like Extended Kalman Filter (EKF) or by multi-view optimization methods like Bundle Adjustment (BA). It is interesting to note that these approaches differ in the frequency of map updating operations. Filtering approaches [7, 2] update the map at each new frame, so as to maintain the consistency of the linearization, which requires the map to be limited to a few hundred landmarks. In contrast, optimization methods [28, 19, 39, 17, 32] wait until the baseline is sufficiently large to allow good localization of 3D landmarks. The latter approach leads to the notion of "keyframes", i.e., the frames used to update the map. Though the selection of keyframes is generally done by considering statistics over the feature tracking, [47] proposes a criterion based on image projection error.

The proposed method, denoted by eVO (efficient Visual Odometer) combines characteristics of both approaches, while being oriented towards low computational cost rather than optimality [31]. Therefore, as in VO, the 3D points of the reference map are estimated at one time instant and are not refined using others views. The notion of keyframe is also used, as in optimization approaches of V-SLAM: the 3D map is computed only at keyframe instants. Cameras are localized by pose estimation with regard to the current map, using a 2D-3D association strategy. We show that this keyframe strategy not only reduces the cost of the algorithm, but also significantly reduces the drift of the estimation error with respect to DRVO. As a result, the proposed eVO is able to run at video frame rates (15 to 25 Hz) on the limited computer available on our MAV.

Note that Nister et al. have proposed in ref. [33] a VO approach with keyframes (they called them reference frames) for ground vehicle applications. More recently, the Pixhawk team [10] have reported scene modeling using a MAV with a stereo sensor and on-board VO; however, there are very few details regarding the odometry in their paper, while we present here a parametric study of the performance of the algorithm on various datasets. In particular, eVO has been successfully tested on the online Kitti benchmark and ranks at the 6th position, and 4th among methods based on stereo data [12][1]

Environment modeling

Let us first discuss 3D sensors. In the proposed solution, we use two concurrent 3D sensors: a stereorig and an active Asus Xtion RGBD camera, a type of sensor made popular by Microsoft's Kinect device. They have complementary characteristics, the RGBD camera being very efficient indoors, but blind in outdoor situations, where the greater amount of natural textures allow a good stereovision performance. For stereo matching, we use a very fast dense algorithm eFolki, a dense Lucas-Kanade (LK) algorithm published in [4] and recently implemented on GPU (Graphic Processing Unit) architectures [34]. It allows dense and reliable 3D maps to be obtained at video rate on a lightweight laptop with a GT650M GPU.

Environment modeling amounts to aggregating noisy 3D point clouds into a consistent and compact 3D model. Note that the model should include not only the occupied areas, but also the free space and the unvisited areas. This data is required for trajectory planning and replanning. Proposed in the seminal work of Elfes [8], occupancy grids have become the standard for 2D and 3D environment models in mobile robotics [30, 1, 15]. In case of 3D modeling, a standard approach consists in subdividing the workspace into cubic volume elements of equal size called voxels [30].

This simplistic representation presents two disadvantages: (1) the maximal extension of the explored area must be known in advance; (2) the memory occupation becomes rapidly intractable for large scale environments. In [1], the authors propose a two-layer model, combining a local representation using standard voxel grid and a global rough polygonal model organized by height slices. This model is successfully used for automatic navigation in urban canyons, but appears not easily scalable. In [16], the model consists in a 2D regular grid, where each cell stores a list of parallelepipedic volumes corresponding to occupied or free areas. Memory efficient, this solution avoids sampling artifacts in the vertical direction but requires the knowledge of the horizontal workspace extension. In [15], occupied and free areas are also explicitly represented, but using a multi-resolution occupancy grid organized as an octree structure (hence its name: 'Octomap'). Such a structure offers a useful flexibility for modeling unknown areas. By adding levels to the tree, the spatial resolution can be adapted to the local 3D structure of the scene, or the global size of the workspace can be expanded easily. For these reasons, we use Octomap, thanks to the freely distributed C++ library[2].

[1]Please refer to the Kitti benchmark at http://www.cvlibs.net/datasets/kitti/eval_odometry.php.

[2]Please refer to the website http://octomap.github.io/

Organization of the paper

This paper is organized as follows. The overall architecture of 3DSCAN is presented in § " System overview". The description and performance evaluation of the ego-localization module eVO are presented in § " Efficient visual odometry (eVO)". § " Environment modeling" presents the 3D modeling module (stereo processing and aggregation). Reconstructions from Kitti data or from data obtained during test flights with our MAV are presented in § " 3DSCAN results". Finally, we conclude and propose some directions for future work.

System overview

Software architecture

Given image and depth data provided by sensors or read from files, the 3DSCAN processing chain builds a 3D environment model as an Octomap grid. Three processing modules, working as an individual thread at different frequencies, are combined: (1) the eVO module computes the camera pose at video rate, (2) the eFolki module computes the depthmap from a rectified and equalized stereo pair provided by eVO and (3) the Octomap module uses the estimated pose and depthmap to aggregate relative 3D data into a global model. These components communicate using ROS (Robot Operating System, www.ros.org). ROS also provides interface modules to obtain images from sensors, or from files and visualization modules.

Figure 1 depicts the implemented software architecture and data exchanges through the module network. In this organization, eVO, the stereo odometry module, plays a central role. In addition to calculating the camera position, the module geometrically and radiometrically rectifies a stereo pair and achieves a temporal sub-sampling of the sequence by automatically selecting keyframes. Note that the camera poses are saved by the Transform-Frame server (ROS/TF server) for further usage. When eVO selects a stereo pair as a new keyframe, the rectified stereo pair is processed by eFolki in order to compute a depthmap, which is converted into a relative-to-sensor 3D point cloud. In parallel to the stereo process, the RGBD sensor node emits depthmaps at 3Hz. The '3D data source selector' selects the most appropriate sensor (stereo or RGBD camera) depending on the density of the RGBD depthmap and transmits a point cloud to the Octomap server for aggregation in the environment model. This involves searching in the pose database stored in the module TF server for the sensor pose at the date indicated in the point cloud message.

Since ROS performs an abstraction of the hardware layer, the 3DSCAN chain has been deployed on various hardware units: PC workstation, laptop, MAV + ground station. The implementation on a MAV and its ground station (e.g., a MAV system) is described in the following section.

Implementation on a MAV system

We have deployed 3DSCAN on a real MAV system composed of a mid-range laptop used as a ground station and the Ascending Technologies Pelican[3] quadrotor depicted in figure 2.

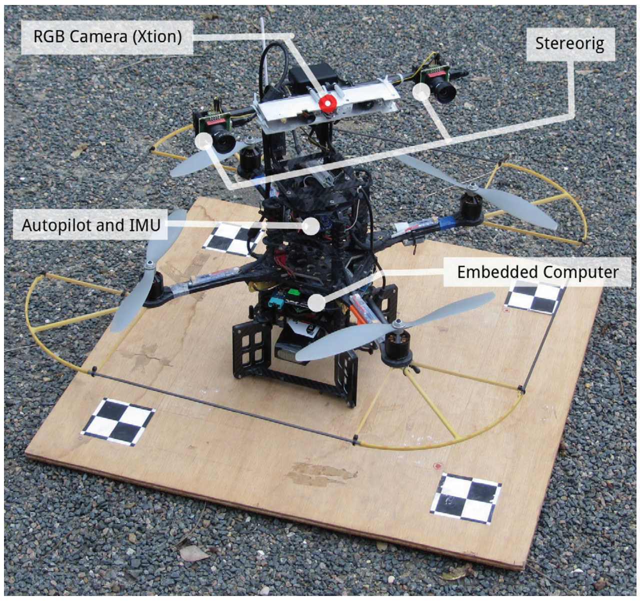


Fig. 2 - Our AscTec Pelican MAV on its landing pad. The visual sensors - stereorig and RGBD camera - are located at the top of the vehicle. The vehicle has a total take-off weight of 2 kg (including the LiPo Battery).

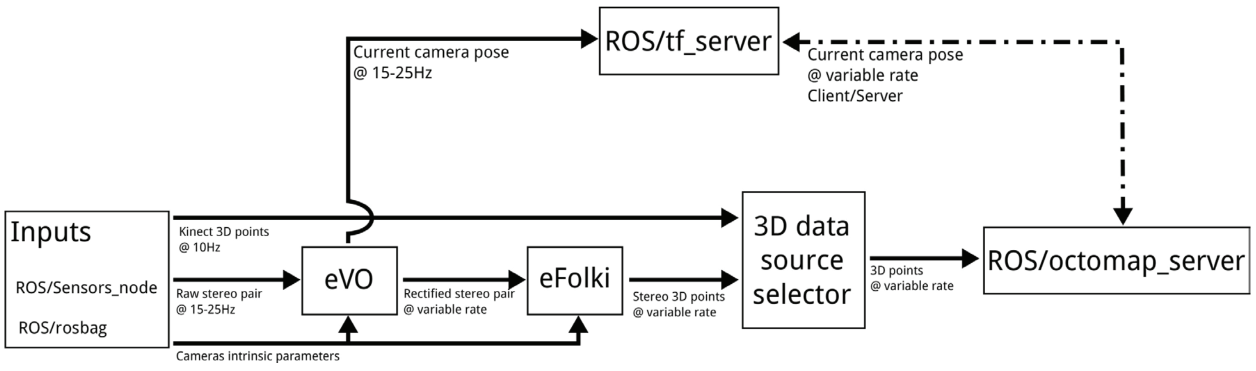

ROS/tf_server
Current camera pose @ 15-25Hz
Current camera pose @ variable rate Client/Server
Inputs
ROS/Sensors_node
ROS/rosbag
Kinect 3D points @ 10Hz
Raw stereo pair @ 15-25Hz
eVO
Rectified stereo pair @ variable rate
eFolki
Stereo 3D points @ variable rate
3D data source selector
3D points @ variable rate
ROS/octomap_server
Cameras intrinsic parameters

Fig. 1 - The 3DSCAN software architecture. Boxes correspond to ROS nodes. Boxes with a name beginning by 'ROS/' indicate a module provided in the standard ROS library. Solid unidirectional arrows indicate data exchanges in streaming mode, while dot dashed bidirectional arrows indicate exchanges in client/server mode.

[3]Please refer to the manufacturer website for details: http://www.asctec.de/uav-applications/research/products.

The MAV equipment includes a MEMS-based IMU, a low grade GPS receiver and a 3D magnetometer. These sensors are connected to the autopilot providing a standard stabilization mode and waypoint-based navigation. We have added an Asus Xtion RGBD camera and a stereorig composed of two electronically synchronized USB cameras separated by a 28 cm long baseline equipped with a 5.5 mm S-mount lens. This configuration provided a usable range of 10 meters for environment modelling[4]. The cameras are two IDS-Imaging UI-1241LE based on a 1.3 MegaPixel global-shuttered CMOS from e2V. Since the native resolution is too large for onboard processing, the binning mode is activated to capture VGA frames without field of view reduction. These sensors are connected to an embedded PC animated by an Intel dual-core Core2Duo 1.86 GHz.

The limited computational performance of the on-board PC and the requirement of a Cuda Compliant GPU for dense stereo matching by eFolki have led to the full processing chain being dispatched on two computers. The ego-localization by eVO runs onboard, while the entire environment modeling task is done on the ground station: a light macbook laptop equipped with a mid-range Nvidia GT 650M GPU. The datalink between the two computers is provided by Wifi-N. Since the transmission of all video streams is impracticable through the datalink, temporal subsampling is performed. For the RGBD camera, the ROS module permits constant subsampling and we have set the output frequency to 3Hz. For the stereo pairs, the subsampling is done by the mechanism selecting keyframe in eVO. This aspect will be discussed in the following section.

Efficient visual odometry (eVO)

Algorithm overview

As already mentioned in the introduction, eVO builds a map updated in a keyframe scheme as in ref. [28, 19]. In the monocular SLAM case, the keyframe structure is mainly motivated by the need for a minimal baseline to localize new 3D landmarks. With our stereo setting, landmarks are instantaneously localized in 3D. Improving the accuracy of a landmark localization requires the stereorig to get substantially closer to the landmark or to displace the sensor lengthways more than the baseline. Hence, in the case of a smooth motion of the stereorig (with respect to the rate of odometry), updating the map at each frame is useless and the keyframe scheme is a better choice.

In contrast to other keyframe-based SLAM, our system differs by the way in which the map is updated. In standard approaches, the positions of visible landmarks are refined at each keyframe by minimizing a multi-view re-projection criterion with bundle adjustment methods. Here, we skip this step because of the limited computational capacity of the embedded PC. In practice, landmarks are then localized once - the first time they are seen - in the global frame using the current estimated pose.

Direct combination of noisy measurements - camera pose and landmark position - brings eVO closer to DRVO, i.e. dead-reckoning methods. However, using the keyframe approach, this update is done at a lower rate in eVO than in DRVO, with the advantage of a reduced drift. A comparison between these two approaches on real datasets is presented in § "Tuning and advantage of the Keyframe Scheme".

Finally, the other advantage of this structure concerns the computational cost. Indeed, 3D localization by a stereorig is not computationally free. Combining a keyframe scheme with a pose computation algorithm using 2D-3D associations avoids computing the 3D structure at each new stereo frame. More interesting still, this approach allows the global process to be divided into one monocular task, the Tracking and Pose computation, executed for each left image acquired, and one stereo task, (Mapping) executed on demand.

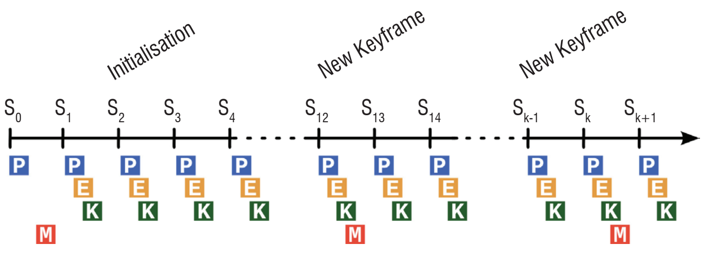


Fig. 3 - Temporal sequencing of the eVO module. The P-box, E-box, K-box and M-box stand respectively for the Image Pre-processing module, the Egomotion module, the Keyframe selector module and the Mapping module.

The eVO process can be described in four modules working sequentially, as depicted in figure 3:

- Image pre-processing: preliminary image warping and equalization;
- Egomotion: estimation of the position and the attitude of the stereorig in the reference frame;
- Keyframe selection: deciding whether a new keyframe is necessary;
- Mapping: stereo pair processing so as to update the landmark map.

Each module is described in the following section; here we briefly describe the eVO process. The algorithm starts by calling the Mapping module, which initializes 3D landmarks. The next available stereo pair is processed by the Egomotion module, which yields the current pose and indicates how many landmarks are still visible. This indicator is used by the keyframe selection module to decide that the current stereo pair is a new keyframe. In figure 3, this loop is repeated until the 13th stereo pair (denoted S12), which is selected as a keyframe. At this point, the mapping module is called to update the map: it adds new landmarks and prunes older ones.

Description of the eVO components

Here we give a detailed description of the eVO components, in the order in which they appear in figure 3.

Image pre-processing Module

The two images are stereo-rectified using the knowledge of the intrinsic parameters. In order to deal with indoor to outdoor (or vice versa) transitions, which lead to locally large illumination changes, two adjustments were necessary. The first one concerns the hardware: the cameras are set to automatically adapt their exposure time, in order to reach a specified intensity average under the constraint that the exposure time cannot exceed a maximal value. The second adjustment consists in equalizing image histograms to avoid dark images.

[4]Under the assumption of a mean disparity error of 0.5 pixels; the usable range is defined as the maximal distance before the precision of 3D localization exceeds the half-width of a voxel (20 to 30 centimeters).

Mapping Module

This module is called when the current stereo-pair is declared as a new keyframe. It uses a stereorectified pair of images to generate an initial map and to update it if necessary by extracting and matching new interest points. The synoptic diagram of this module is shown in figure 4.

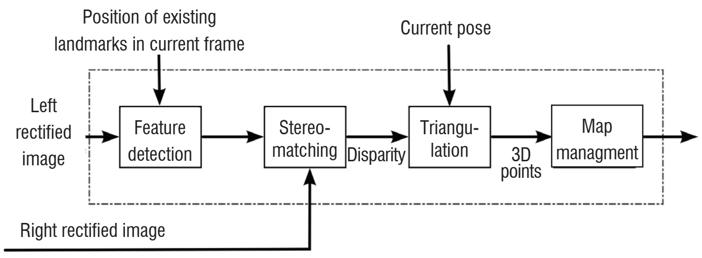


Fig. 4 - Mapping Module structure

The first operation consists in extracting Nf (between 250 and 350 for VGA images) interest points in the left image. This process is done under two geometrical constraints: (i) a minimal separation distance between two features; (ii) a maximal dispersion of the features over the image plane. The former constraint is generally included in feature extractors (like those in OpenCV), while the latter is enforced by a classical bucketing strategy. The image support is subdivided into Nr non-overlapping regions (8X6 regions for VGA images) and the Nf / Nr more relevant features within each region are kept. In order to deal with regions that do not have enough texture, a relaxation technique is used to increment the tolerated amount of features by region. Note that the extraction accounts for mapped landmarks successfully tracked from the previous keyframe, so as to detect only the correct number of new features and maintain Nf features per keyframe.

Two feature detectors have been evaluated: the Harris detector (Shi-Tomasi [43]) and the FAST detector [36]. As expected, the FAST detector is faster than the Harris detector and allows a keyframe (VGA format) to be processed in 55 milliseconds on average (see table 2). On the tested sequence, the choice of the detector has a very limited impact on the estimated trajectories.

In the second step, the features newly detected in the left images are matched in the right image. Based on dense stereovision algorithms, feature stereo matching is done by means of exhaustive searches along the epipolar lines. In practice, the Zero-mean Normalized Cross-Correlation (ZNCC) is used as the image similarity criterion and we test a range of disparities corresponding to 3D points located at least 70 centimeters away from the stereorig. In order to reduce the processing time, we adopt a coarse-to-fine multi-scale approach, with a two-level image pyramid. At the lowest resolution, the image is reduced by a factor of 4 in each direction and the size of the ZNCC window is set to 3X3 pixels. The match is then propagated to the full resolution level and refined by a local search within a region with a radius of 6 pixels, using a 9X9 ZNCC window. In practice, the number of tested disparity hypotheses is largely reduced. In our configuration (focal distance = 5.5 mm and depth greater than 70 cm), this approach allows the number of tested hypotheses to be reduced from 220 to less than 70. Finally, the ZNCC scores are thresholded to prune ambiguous associations

At this point, feature positions, disparities, stereorig parameters and the current pose estimation are used to localize the new landmarks in the reference frame by triangulation. Finally, new landmarks and their image signature are inserted into the map, while the untracked landmarks are removed.

Egomotion Module

As soon as some landmarks have been localized in 3D, the egomotion module estimates the position and the attitude of the left camera frame, by tracking the landmarks in the successive images acquired by the left camera. Figure 5 shows the internal mechanism and the module inputs/outputs.

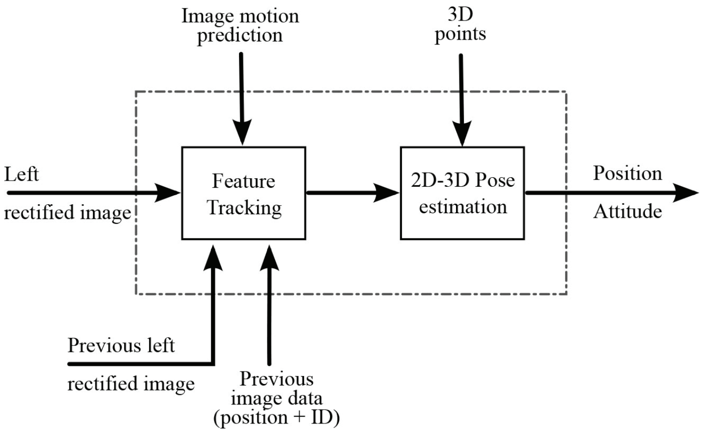


Fig. 5 - Egomotion Module. This module uses the left image only.

As in [23], the features are tracked through the video sequence acquired by the left camera using KLT [43]. In order to pre-emptively prune wrong temporal matchings, the fundamental matrix is robustly estimated using a Least Median of Squares scheme (LMedS) [37]; this operation is henceforth referred to as 'Fcheck'. Since this estimation can be unstable in the case of small relative motion, it is automatically disabled when the motion of features is less than a threshold.

We have also evaluated an active search process, where the search for temporal matches is guided by a prediction of the motion. Without inertial data, as for instance in the KITTI datasets, we use a simplistic motion prediction model based on constant linear and angular speeds. The motion estimated between the two previous frames is then used for motion prediction. If inertial data is available (as for instance in the MAV experiments), we only compensate for a global rotation of the image. Both methods help to reduce the search area for temporal matching.

Features	SHI-TOMASI [43]		FAST [36]	
Frame type	Keyframe	Standard	Keyframe	Standard
Average (ms)	74.2	12.4	56.1	12.4
Std (ms)	4.6	3.6	5.6	3.6
Min (ms)	62.3	5.8	40.8	5.6
Max (ms)	99.9	32.8	72.7	31.5

Table 1 - eVO processing time for one 672_480 stereo pair on a Core2Duo 1.86GHz. Measurements obtained by averaging over 10 Monte-Carlo runs.

From the temporal matchings provided by KLT, associations are established between 3D landmarks stored in the map and current image features. Given these 2D-3D matches, the camera pose (position and attitude) is robustly estimated within a RANSAC procedure [9]. In practice, we have implemented our own RANSAC framework with an online adaptation of the number of iterations, as proposed by Peter Kovesi [21]. For each random sample, the pose is estimated with the Perspective-3-Point (P3P) algorithm [9, 45]. A bucketing strategy is used to enforce a minimal separation distance between the image features selected in the triplet given to the P3P algorithm. The P3P method often produces multiple solutions (up to 4): in such a case, all of the solutions are considered as random samples in the RANSAC voting process.

The RANSAC procedure returns an initial pose solution and a set of inliers. The pose is refined by minimizing the re-projection error of inlier matches. This non-linear least-squares optimization is solved using the motion-only optimization functions provided in the Lourakis SBA code [24].

Keyframe selection Module

As proposed in [28], a new keyframe is initialized as soon as the ratio between the number of successfully tracked features and the number of 3D points visible on the last keyframe drop under a threshold, denoted by t and set by default to t = 0:8. We discuss the algorithm sensitivity to parameter t in § "Tuning and advantage of the Keyframe Scheme".

Implementation and processing time

The implementation of eVO uses two well-known open-source tools: OpenCV and ROS (Robot Operating System, www.ros.org). Most of the low level image processing — image warping, tracking, feature extraction and template matching — is based on the OpenCV library. This library is optimized for the SIMD instruction set of the on-board CPU (Intel SSE). At the system scale, eVO works on ROS to deal with the physical sensors and share the trajectory estimation with the environment modeling part of 3DSCAN.

In table 1 we present the processing times measured on the embedded computer of our MAV: Ascending Technologies Mastermind Intel Core 2 Duo 1.86 GHz working on Ubuntu 12.04 32bits. Figure 6 shows how the computational time is distributed over the various components of the processing chain. These results demonstrate the great difference between keyframe and standard frame processing time, due to the fact that the 3D landmark generation is bypassed for the latter.

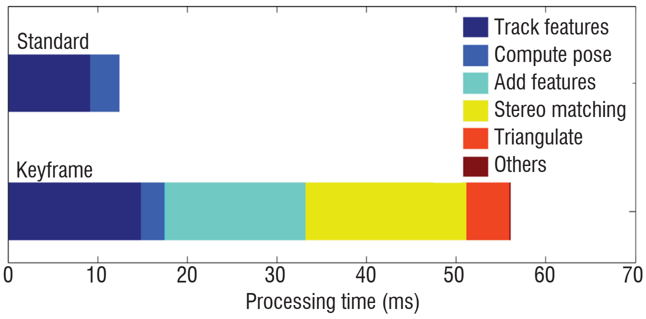


Fig. 6 - Relative computing time of eVO components. Measurements made by averaging over 10 Monte-Carlo runs, using a FAST feature detector [36].

As a consequence, the overall computational performance of eVO depends on the ratio between standard frame and keyframe numbers. In our implementation, this ratio is not fixed but varies with the success rate of the tracking, which itself depends on the vehicle dynamics. However, as discussed later in § "Tuning and advantage of the Keyframe Scheme", the best tuning of the keyframe selector leads to an average keyframe ratio of less than 30%. This means that the average computing time is less than 25 ms/frame. We can also note that the monocular egolocalization process (i.e., processing of a standard frame) could be run at a very high frame rate (up to 80hz) on one core of the embedded PC if the bandwidth of the USB-bus allowed it.

Evaluation

Datasets and performance measurements

Our system has been evaluated on multiple and varied data. Some of it was acquired using our own stereorig, either hand-held or carried by the MAV. No ground-truth state is available for this data, but we have followed loop trajectories in order to use the drift between the first and last frames as a performance indicator. An example of an outdoor experiment with a 60m-long loop is presented in figure 7, showing a drift of approximately 1% of the trajectory length.

We have also used the KITTI odometry dataset [12] composed of 22 video sequences acquired by a car equipped with several sensors (Velodyne R lidar, high resolution IMU and GPS-RTK, stereorig). The video collection covers a large range of environments (highway, suburban or town center) and trajectory profiles (loops, road sections) from one hundred meters to a few kilometers. The first half of the collection is supplied with ground-truth in order to adjust the algorithm parameters. The second half of the collection is used to benchmark algorithms.

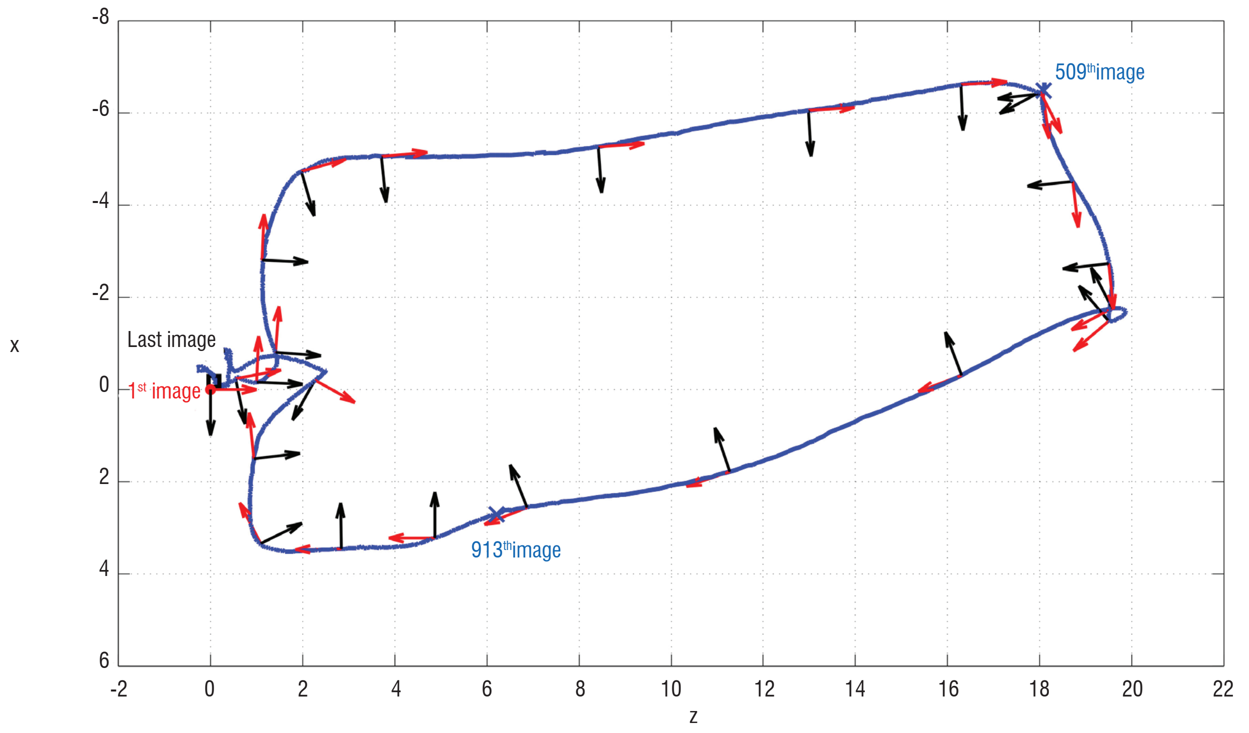
The KITTI Team also provides some performance metrics, together with a tool to compute them on the estimated trajectories. These metrics are: a translational drift expressed as a percentage of the total traveled distance and a rotational drift expressed in degrees by traveled meter. Scores are averaged over all possible sub-sequences of variable lengths, from 100 m to 800 m.

Since our system includes a random sampling scheme (RANSAC), we have performed Monte-Carlo simulations and measured statistical indicators (average performance, standard deviation, median, min-max values).

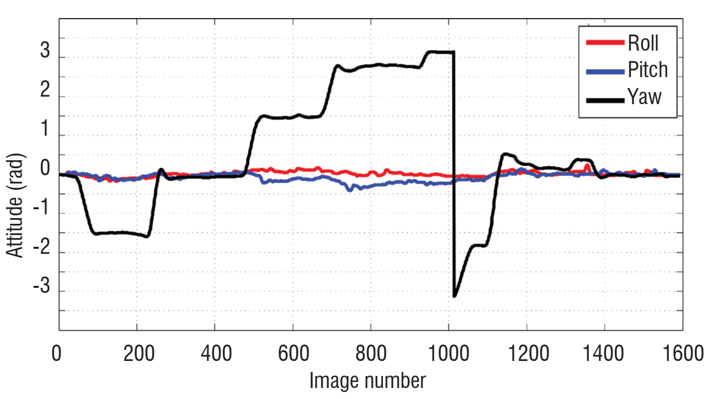
Figure 8 presents the estimated trajectories obtained after 25 Monte-Carlo runs on Sequence 08 of the KITTI odometry dataset. This trajectory in a suburban environment is 2 kilometers long and comprises many moving objects (vehicles, pedestrians and cyclists). On average, the estimated trajectory in the horizontal plane (XZ) is well estimated with a drift of only 4 meters. As usual in odometry, large angular errors occur at each important turn change. The estimation along the third dimension shows a bias at the beginning, which is probably due to an error in the ground truth and a significant variance at the end. We could constrain the eVO estimator to maintain a constant height above the ground, but we choose not to do so, since we intend to use the same algorithm for MAV data.

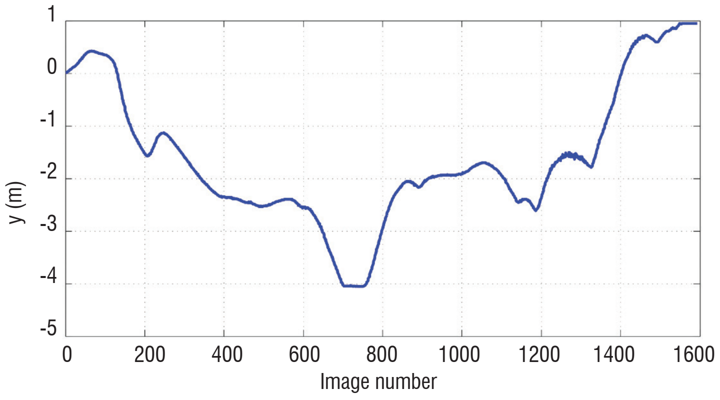
In the following section, we study the advantage of the keyframe scheme and discuss the tuning of the parameters of the Keyframe selection module, before presenting the global evaluation of eVO on the KITTI benchmark.

(a)



(b)



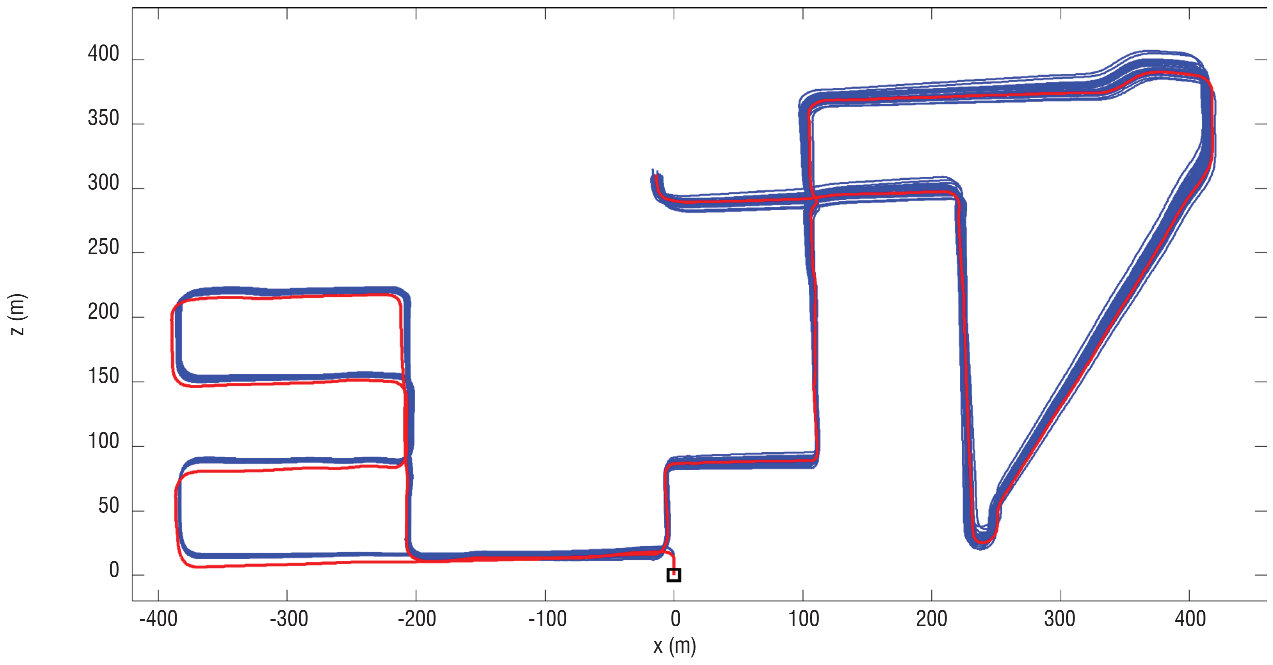


(c)

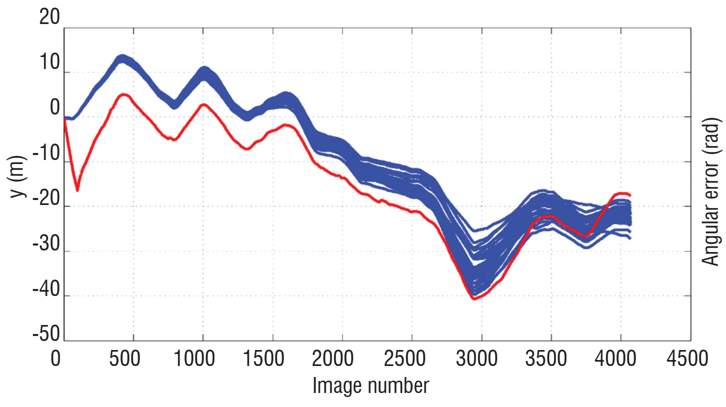
(d)

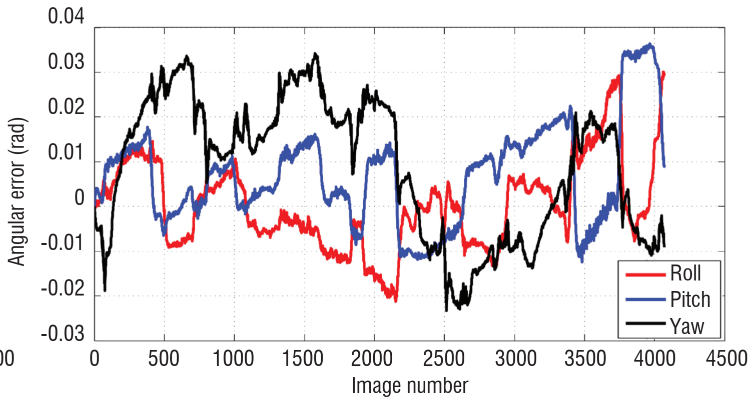
Fig. 7 - Trajectory estimated by eVO from the sequence 20120727.3 acquired during an outdoor flight of the MAV.
(a) 4 frames of the video sequence (the 1st, 509th, 913th and last image).
(b) Estimated trajectory. The red and black arrows indicate the attitude of our MAV (red: the front of the MAV, black: its right).
(c) Estimated attitude. The measurements provided by the embedded AHRS are not precise enough to serve as ground truth.
(d) Estimated height profile (the Y axis points downward). Note that the actual starting point is approximately 80 cm above the landing pad; hence, the total drift is less than 50 centimeters.

(a)



(b)





(c)

(d)

Fig. 8 - Result of eVO on the“08” sequence of the KITTI odometry dataset.
(a) Four images of the sequence.
(b) Trajectories on the XZ plane (red: ground truth, blue: estimated). Shown in red: the ground truth. Shown in blue: 25 trajectories obtained after as many Monte-Carlo runs.
(c) Average angular errors (in radians).
(d) Trajectories in the 3rd dimension.

Tuning and advantage of the Keyframe Scheme

Here, we discuss the advantages provided by the keyframe scheme in regard to the ego-localization performance, beyond its computational efficiency discussed previously. First, we compare our algorithm with a classical dead-reckoning visual odometer (DRVO) built with the same software components. Then, we investigate the influence of two parameters controlling the keyframe generation, the threshold t defined earlier and the activation of the Fcheck module. All of the results obtained on MAV-representative sequences are summarized in table 2, while results on the KITTI dataset are shown in table 3.

eVO vs. DRVO. As expected, the keyframe scheme allows the localization drift of eVO to be reduced compared to DRVO, even for settings that favor the generation of new keyframes. This is the case when choosing t = 1:0, which means generating a new keyframe as soon as one landmark is lost by the tracking process. The gain is particularly important with MAV data, as shown by comparing the total localization error presented in the two first rows of tables in table 2. On the KITTI dataset, the advantage is less important but significant, with a 10%-reduction of the drift, see table 3. This can be explained by the fact that, due to the car's speed, KITTI sequences exhibit larger inter-frame motion, reducing the interest of the keyframe scheme.

Parameters controlling the key-frame selection. We first study how the drift varies with respect to the ratio t, while the Fcheck module (which checks for the consistency of matches with the epipolar constraint) is activated. On the MAV sequences (table 2) the lower the parameter t, the lower the average localization error, but the higher the dispersion of the results. On the Kitti dataset (table 3) we observe that the choice t = 0.6 leads to larger errors. This is due to a lower frame rate and a higher vehicle speed, which means that the odometry uses tracked features that are farther from the camera and are badly localized. Finally, we choose t = 0.8 as a good trade-off.

The Fcheck procedure also has a significant influence on the number of keyframes. If this validation step is bypassed, the number of keyframes is reduced by half in all processed sequences (for the same ratio t). In the majority of our tests, this entails an error growth, particularly on the KITTI dataset, where the translational drift increases from 1.46 to 1.63. In practice, we choose to enable Fcheck by default.

Result on the KITTI Odometry Benchmark

Table 4 presents the average scores of eVO on the KITTI evaluation dataset, compared to other published methods. eVO obtains a very good performance, with an average translation drift of 1.76% and an angular error of 0.0036°/m. As on the date of its first submission to the IROS conference (March 2013), this performance allowed eVO to rank first. One year later, it is still 4th among methods that use only stereo data - note that methods using lidar data have been recently introduced in the KITTI table and have taken the two first positions.

Environment modeling

In the previous section, we have described how the stereo data is processed in order to estimate the pose of the system during its displacement. These estimated poses are used to fuse 'instantaneous' 3D data into a 3D model of the visited environment. 3D data can be depth measurements provided by an active RGBD sensor or stereo depthmaps. The latter are provided here by a fast and dense stereo-matching code on GPU, which is described in § "Dense stereo-matching". The chosen environment modeling framework is presented in § "Dense stereo-matching".

Dense stereo-matching

Classically, dense stereo-matching algorithms are based on systematic exploration in the disparity space, to evaluate radiometric similarities between pixels of the two images. Here, dense disparity maps are computed using a dense Lucas-Kanade (LK) algorithm [26] derived from an original optical flow algorithm eFolki, described in [35]. The resulting code is remarkably fast on a massively parallel architecture such as GPU. In the following sections, we recall the equations of the algorithm, first published in [4], discuss its implementation on GPU, describe some adaptations made to increase the robustness of the estimated disparity on real stereo images, and finally present a local indicator of the consistency.

All evaluations are performed on data provided in the KITTI benchmark [12].

Efficient dense matching by the LK algorithm

The basic problem of the dense LK algorithm is to register local windows centered around each image pixel $\mathbf{x}$ by minimizing a SSD (Sum of Squared Difference) criterion over a 2D motion vector $\mathbf{u}(\mathbf{x})$:

$$\sum_{\mathbf{x}'} w(\mathbf{x}'-\mathbf{x})\left(I_1(\mathbf{x}')-I_2(\mathbf{x}'+\mathbf{u}(\mathbf{x}))\right)^2 \qquad (1)$$

where w is a separable weighting function, uniform or Gaussian, of limited support $\mathcal{W}$, typically a square window parameterized by its radius r. Since we consider here dense matching of rectified stereo data, where epipolar lines are aligned with the horizontal axis of the images, the motion vector is reduced to a scalar disparity: $\mathbf{u}(\mathbf{x}) = [d(\mathbf{x});0]$.

The minimization of criterion 1 is done by an iterative Gauss-Newton coarse-to-fine pyramidal strategy, as in classical implementations of LK. However, using the first order expansion described in [25], an iteration can be completed with only one image interpolation per pixel, while the well-known PyramLK algorithm [5] requires several image interpolations per pixel. An iteration of this convergent dense matching strategy, denoted eFolki, consists in:

$$\begin{aligned}
I_2^{(k)}(\mathbf{x}) &= I_2\left(\mathbf{x}+\begin{bmatrix} d^{(k)}(\mathbf{x}) \\ 0 \end{bmatrix}\right), \forall \mathbf{x} (\text{interpolation}) \\
\delta I^{(k)} &= I_2^{(k)} - I_1 - \nabla_x I_1 \otimes d^{(k)} \\
c^{(k)} &= w * \left(\nabla_x I_1 \otimes \delta I^{(k)}\right) \\
d^{(k+1)} &= c^{(k)} \varnothing \left(\nabla_x I_1^2\right)
\end{aligned}$$

where $d^{(k)}$ is the previous disparity guess, ∇_x is the image gradient operator in the x-direction and operator $\otimes$(respectively $\varnothing$) is the component-wise multiplication (respectively division). One can readily observe that eFolki is ideally suited for GPU implementation, because each iteration requires only very regular operations on the images: separable convolutions, pixelwise operations and image interpo-

Sequence	Method	Error X (m)	Error Y (m)	Error Z (m)	Keyframe ratio
Name: **2010727.2** Image Number: 2039 Trajectory length: 150 m hand-held	DRVO	-2.4 ±0.04	-0.5 ±0.05	2.0 ±0.04	100%
	EVO τ=1.0	-1.28 ±0.08	-0.17 ±0.11	1.20 ±0.08	96%
	EVO τ=0.8	-0.95 ±0.18	-0.30 ±0.11	1.01 ±0.14	36%
	EVO τ=0.6	-0.73 ±0.21	-0.06 ±0.25	0.83 ±0.18	19%
Name: **20120724.3** Image Number: 1675 Trajectory length: 70 m Acquired by MAV	DRVO	-6.8 ±0.4	2.05 ±0.2	4.7 ±0.4	100%
	EVO τ=1.0	-0.8 ±0.2	0.23 ±0.24	0.45 ±0.12	91%
	EVO τ=0.8	-0.61 ±0.34	0.23 ±0.35	0.4 ±0.17	27%
	EVO τ=0.6	-0.33 ±0.60	0.35 ±0.47	0.21 ±0.3	13%

Table 2 - Localization error at the end of two closed trajectories acquired with the stereorig of our MAV and ratio of keyframes for different algorithms or algorithm settings.

Sequence	Method	Translational drift (%)	Rotational drift (deg/m)	Keyframe ratio
Name: **Kitti Benchmark** Training SDataset Acquired by a car	DRVO	1.56 ±0.007	0.00166 ±0.00008	100%
	EVO τ=1.0	1.45 ±0.015	0.00145 ±0.0001	99.8%
	EVO τ=0.8	1.46 ±0.014	0.00144 ±0.0002	79.6%
	EVO τ=0.6	1.53 ±0.017	0.00151 ±0.0002	37.8%

Table 3 - Angular and translation drift indicators measured on the KITTI Odometry dataset for various algorithms or algorithm settings.

Rank	Method	Setting	Translation	Rotation	Runtime	Environment
3	MFI	st	1.30%	0.0030 [deg/m]	0.1 s	4 cores @ 2.5 Ghz (C/C++)
4	VoBa	st	1.46%	0.0030 [deg/m]	0.1 s	1 core @ 2.0 Ghz (C/C++)
5	SSLAM	st	1.57%	0.0044 [deg/m]	0.5 s	8 cores @ 3.5 Ghz (C/C++)
6	**eVO**	**st**	**1.76%**	**0.0036 [deg/m]**	**0.05 s**	**2 cores @ 2.0 Ghz (C/C++)**
7	SOVI	st	1.80%	0.0079 [deg/m]	0.1 s	4 cores @ 2.5 Ghz (Matlab)
8	D6DVO	st	2.04%	0.0051 [deg/m]	0.03 s	1 core @ 2.5 Ghz (C/C++)
9	MICP_VO	st	2.13%	0.0065 [deg/m]	0.01 s	1 core @ 2.5 Ghz (C++)
10	SSLAM-HR	st	2.14%	0.0059 [deg/m]	0.5 s	8 cores @ 3.5 Ghz (C/C++)
11	VIS02-S	st	2.44%	0.0114 [deg/m]	0.05 s	1 core @ 2.5 Ghz (C/C++)
12	GT_VO3pt	st	2.54%	0.0078 [deg/m]	1.26 s	1 core @ 2.5 Ghz (C/C++)

Table 4 - Kitti Odometry benchmark chart at 2014-02-03. Please note that only stereo-based algorithms are presented; however, the ranks are those of the published Kitti, where lidar-based methods occupy the two first places. The eVO result is obtained by tracking at the most 500 Shi-Tomasi interest points extracted from 20 x 8 regions. The tracking is initialized with the previous motion. The ransac threshold is set to 1.0, while the parameter τ is equal to 0.8.

Method	out noc	out all	avg noc	avg all	density	GT650M
SSD	32.9%	34.2%	7.2 pix	7.9 pix	100%	27 ms
SSS+Rank+WRA	13.7%	15.4%	2.9 pix	3.3 pix	100%	108 ms

Table 5 - Evaluations of the different variants of the eFolki dense matching technique on Kitti stereo training databases.
'Rank' denotes Rank-n pre-filtering and 'WRA' means Window Radius Adaptation; see text.
Columns 4 and 5 give the percentage of pixels with an error greater than 3 pixels.
The average computing time for a mid-range GPU is shown in the last column.

lations. In 2009, we demonstrated a CUDA implementation of this algorithm able to compute a dense OF estimation on a full HD video (1920 x 1080) in less than 20 ms on a 285 GTX board [34].

Increasing robustness

As is well known, SSD is not a robust criterion and using the previous algorithm on a real-world image leads to inhomogeneous results, as illustrated in the second line of figure 9. However, following the work of Sun et al. for Horn-Schunk OF methods [41], we have found that simple modifications of the algorithm, essentially pre-filtering and adaptation of the coarse-to-fine strategy, can greatly improve the result.

The first problem is that the motion estimation greatly depends on the local image texture and fails in the event of illumination changes. To correct this, we apply a Rank $-n$ transform [46] to the images before SSD minimization. Each pixel $\mathbf{x}$ is replaced by the number of neighboring pixels with an intensity lower than $I_{(\mathbf{x})}$. This transform is fast and has only one parameter: the radius n of the neighborhood. Transformed images have a compressed intensity range, which increases the robustness and homogeneity of the eFolki result.

The second issue is related to convergence: ensuring the convergence of the LK iteration often requires large windows to be chosen, at the cost of a lower resolution of the estimated flow. The solution proposed here is to vary the radius of the window during the iterations: we denote this strategy 'WRA' for Window Radius Adaptation. In practice, our solution consists in adding a loop at each pyramid level and progressively reducing the window size.

The effects of these modifications are illustrated in figure 9 on an image of the Kitti Stereo Dataset. The modifications lead to an estimate (third line of the figure), which appears significantly more accurate and reliable than the previous one. Quantitative comparative measures are given in table 5. The proposed modification leads to a reduction by a factor of 2 in the number of erroneous pixels and the average disparity errors. According to the current Kitti stereo benchmark, our algorithm ranks only at around the 40th position; however, it is among the fastest methods. In addition, its limited accuracy appears sufficient for our 3D modeling task.

A local indicator of reliability

An important issue when using dense stereo-matching for environment modeling and autonomous navigation is to be able to assess locally the reliability (and the accuracy) of the estimated disparity. In particular, it is important to detect regions where disparity estimation has failed, so as to avoid dangerous movements toward undetected obstacles or to plan a revisit to fill up the map.

We propose to compare depth values estimated respectively from the forward disparity $d_{1\leftarrow 2}$ computed using criterion (1) and the backward disparity $d_{1\leftarrow 1}$ computed by exchanging I_1 and I_2 in (1). More precisely, for each pixel $\mathbf{x}$ in I_1, we compute the error ε_Z defined as:

$$\varepsilon_Z(\mathbf{x}) = \frac{fb\left(d_{1\leftarrow 2}(\mathbf{x}) - d_{2\leftarrow 1}\left(\mathbf{x} + d_{1\leftarrow 2}(\mathbf{x})\right)\right)}{d_{1\leftarrow 2}(\mathbf{x})\, d_{2\leftarrow 1}\left(\mathbf{x} + d_{1\leftarrow 2}(\mathbf{x})\right)}$$

where f is the focal distance in pixels and b is the stereo baseline in meters. The threshold (in meters) is typically chosen equal to the voxel resolution of the 3D model.

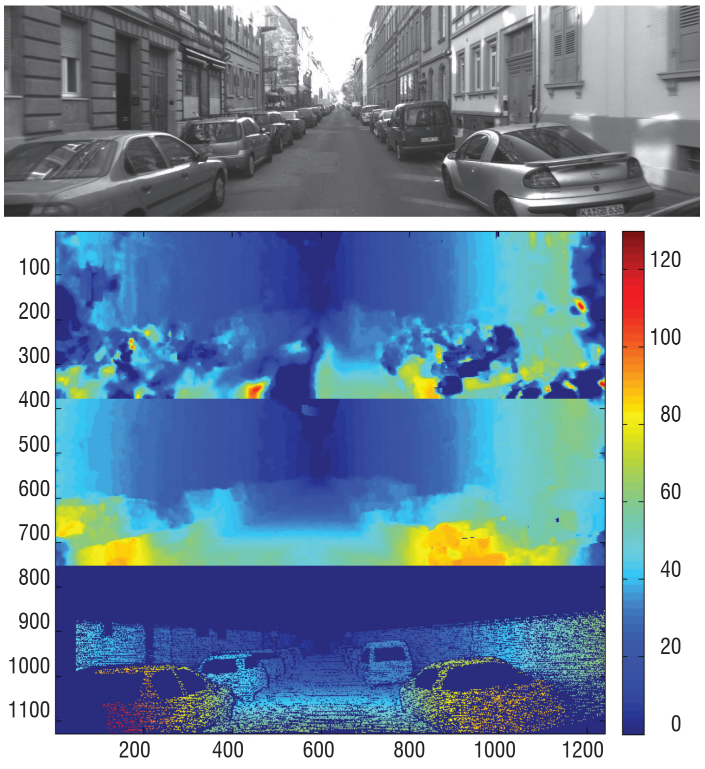


Fig. 9 - Stereovision results on the 172th image of the KITTI Stereo benchmark. From top to bottom: the left image of the stereo pair; disparity map estimated with SSD minimization; disparity map estimated using the modifications (rank transform and window radius adaptation); Ground Truth acquired by a Velodyne sensor.

Building the 3D model of the environment

Octomap model

As discussed in the introduction, a volumetric representation of the 3D environment can be obtained by subdividing the visited space with a regular 3D grid. Each elementary part is called a voxel and stores, for instance, the occupancy probability, as proposed in [27]. Occupancy probabilities are updated by ray-tracing techniques. For a sensor (stereo or active RGBD) delivering a depthmap in some known image geometry, each pixel of the depthmap defines a ray and a 3D point located on this ray approximately at the depth stored in the pixel. All of the voxels that belong to the segment linking the sensor pixel and the 3D point are processed, i.e., their probability of occupancy is updated according to some model of the 3D sensor accuracy.

In the probabilistic 3D mapping framework Octomap of [15] a multi-resolution grid based on an octree data structure replaces the standard regular 3D grid. This solution permits an automatic adaptation of the map resolution to the local 3D geometry, with the advantage of smaller memory requirement and faster data access. Moreover, the octree representation can be defined without a precise prior knowledge of the size of the visited environment. Indeed, when room is needed for new areas, the octree is expanded by a new level. In practice, Octomap is limited to 16 levels, hence to 215 voxels. Note that the Octomap framework provides labels to denote voxels that are in free space and also voxels that have not been explored yet, see figure 10.

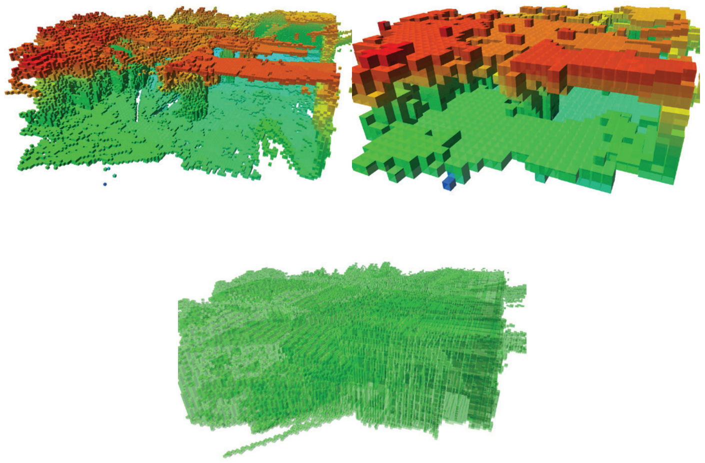


Fig. 10 - Octomap model of a parking area in the basement at Onera.
Top left: 3D occupancy model at the finer scale. Voxels with a probability higher than 80% are colored with a colormap related to their height above the reference plane, which is the horizontal plane at the starting position of the MAV.
Top right: rough 3D model, which can be readily obtained from the octree representation.
Bottom line: freespace voxels (fine scale) colored in transparent green.

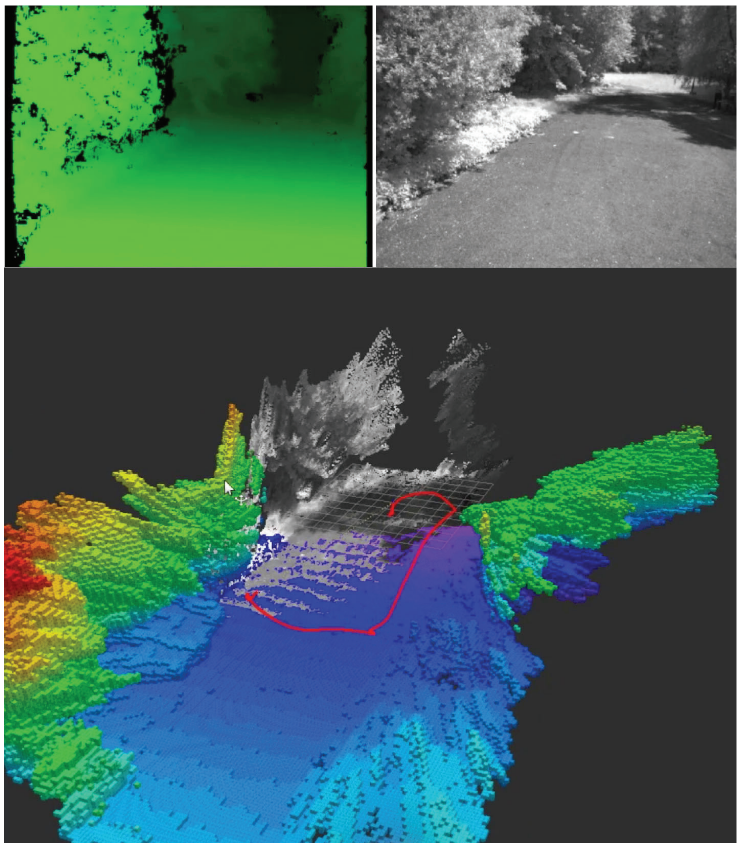


Fig. 11 - Illustration of online 3D scene modeling, outdoor flight.
Voxel resolution: 20 cm.
Top: estimated stereo depthmap and left image recorded by the stereorig.
Bottom: current estimated trajectory of the MAV (red curve), re-projected 3D map (graylevels) and current 3D model (voxel in colors)

3DSCAN results

Outdoor MAV flight

Figure 11 presents a 3D reconstruction obtained on-line from stereo data during an outdoor flight of the MAV (sequence 20120727.3). The estimated trajectory, shown in red in the 3D representation of figure 11, is presented in more detail in figure 7: it is a loop approximately 60 m long. We present, not the final reconstructed model, but images extracted from a screenshot of the ground station during the flight. The current frame taken by the left camera and the corresponding stereo depthmap are presented in the top part of the figure. The forward/backward consistency check described previously has been used to eliminate areas near the edges of the trees that cannot be seen in the two images. The instantaneous 3D map is re-projected in the 3D model with graylevel texture from the current left image. The occupancy model represents obstacles previously detected during the flight. The voxel size is 20 x 20 x 20 cm and the color is related to the height above the initial horizontal plane. Since this reference plane was not aligned with the ground, the color level of the reconstructed ground is variable. Note that the shapes of the scene 3D objects are elongated along the view axis of the onboard stereorig, because of the limited accuracy of 3D triangulation. However, this model provides a good localization of the obstacles that are closest to the MAV during its flight, which is the main objective for this exploration mission. A refined model could be built by getting around the 3D structures, as illustrated in Fig. 13 below.

Indoor MAV flight

Figure 12 presents a 3D reconstruction obtained on-line from stereo data during an indoor flight of the MAV in a parking area located in the basement of a building at Onera. The complete model of the visited part of the parking area was presented in figure10. The estimated trajectory, shown in red in the 3D representation of figure 12 is again a loop approximately 30 m long. As before, we present the left image, the associated depthmap and the current 3D model during the final part of the flight. Details such as the obstacle on the ground and the pipes on the left wall are clearly visible in the reconstructed model. Figure 13 shows how the post in the middle of the parking area is refined as the MAV flies around it: it is at first reconstructed with a large elongation in the viewing direction (left image) then, as the MAV gets around, its shape is refined and fits its actual support more precisely (right image).

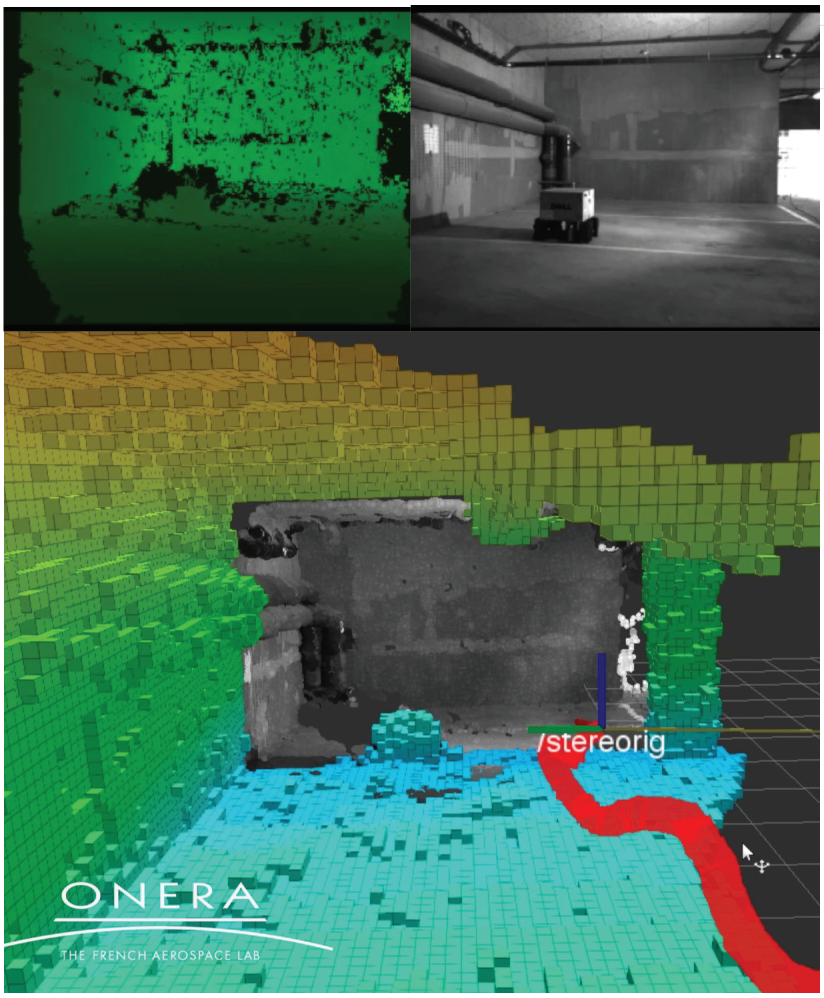


Fig. 12 - Illustration of online 3D scene model, indoor flight. Voxel resolution: 20 cm.
Top: estimated stereo depthmap and left image recorded by the stereorig.
Bottom: current estimated trajectory of the MAV (red curve), re-projected 3D map (graylevels) and current 3D model (voxel in colors)

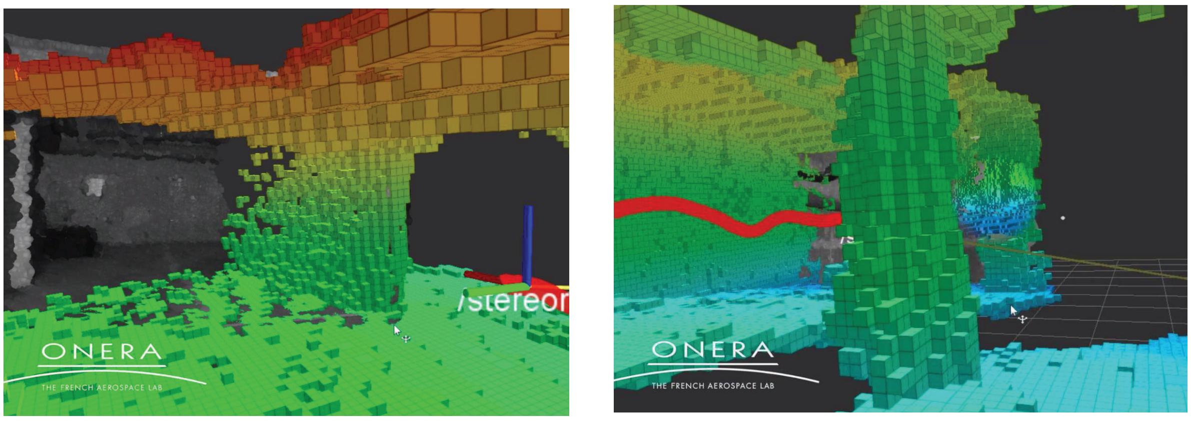


Fig. 13 - Refining the shape of 3D objects by flying around them, see text.

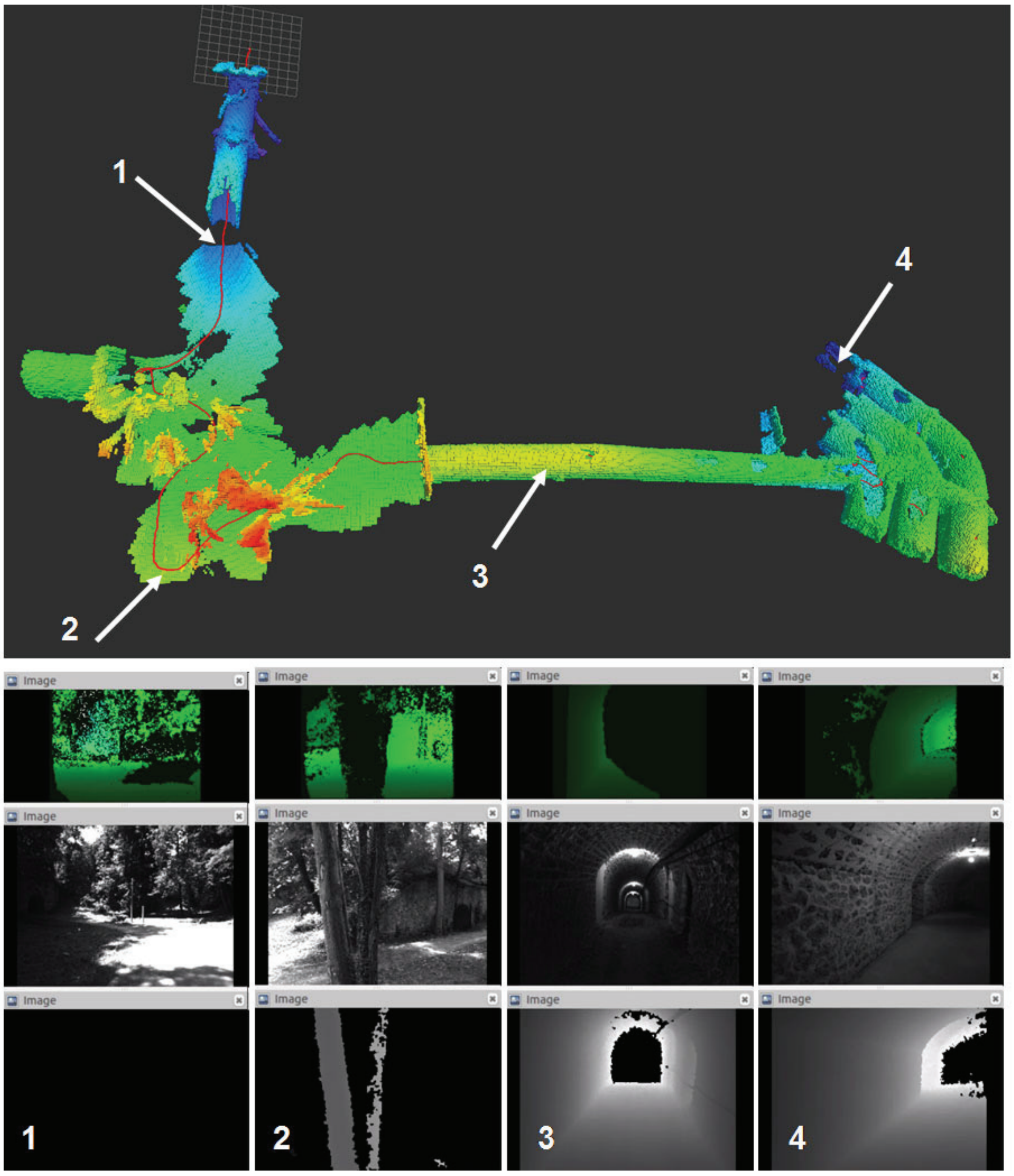


Fig. 14 - 3D Model of the “Caponière”, a historical underground location at Onera.
The thick red line denotes the MAV trajectory. The screenshots on the bottom row present the available data during the experiment, at 4 instants indicated by numbers on the trajectory. For each instant, we present the stereo depthmap (top image in green levels), the left image (B/W image in the middle) and the Xtion depthmap (bottom image in graylevels).

Indoor/outdoor trajectory

Figure 14 shows a large and complex reconstructed model of the "Caponière" area at Onera's Center in Palaiseau (France). The MAV, which is hand-held in this experiment for security reasons, travels along a 200m trajectory passing through a tunnel and a wooded area before going inside a long gallery leading to a centenary underground gunpowder warehouse. The trajectory is computed online and on-board at a 20Hz rate by eVO using stereo data. Note that the estimated trajectory is smooth, despite the transitions between indoor and outdoor areas. The model aggregates stereo or 3D data from the embedded Asus Xtion RGBD camera. The default device is the Xtion, which is, when available, usually more accurate than stereo depths. However, in many situations, especially outdoors, the depth-map delivered by the Xtion is incomplete, or even empty. When less than 80% of the pixels are measured by the Xtion, we use the stereo depthmap. Examples of data delivered by stereo and Xtion, and the switch between them, are presented in the lower part of figure 14. Essentially, stereo is used outside and Xtion inside the tunnels. Note however that, in some situations (see times 2 and 4 in figure 14), both sensors deliver useful information. Designing better fusion rules for both sensors during the modeling and odometry processes is the subject of future studies.

Conclusion

In this paper, 3DSCAN, an efficient framework for egolocalization and 3D modeling of the environment from stereo and RGBD data, has been presented. First, we have demonstrated state estimation from stereo data at 20Hz using one core of the Core2Duo 1.86 GHz on-board the MAV. Higher rates, typically 50Hz, could easily be obtained using multi-threading and with a more recent computer. This visual odometer, denoted eVO, has been evaluated on publicly available stereo data with very good results. Second, a non-supervised 3D modeling software application has been developed using the Octomap framework. It uses stereo data, processed by our fast dense matching code eFolki on GPU and 3D data obtained from an Xtion active RGBD sensor. On our ground station, a light laptop with a mid-range GT 650M GPU, the depthmap computation (limited to the 1-8 m range) and integration into the 3D model runs in 1 to 2 s, which is sufficient for the dynamics of our quadrotor. In the Kitti setup, the 3D data range is greater (up to 30 m), the vehicle is much faster and explores larger areas; hence, the 3D modeling requires a powerful workstation to run with the same rates.

Our current work is aimed at using 3DSCAN for autonomous navigation of MAV in unknown environments, with control and planning issues. Some improvements and adaptations are necessary to improve its robustness and to embed the system on the MAV (using a novel embedded CPU board). We intend to add a multi-view refinement step in eVO for the fusion of eVO with other sensors available onboard (IMU and GPS) to improve the quality, rate and reliability of state estimation. In terms of perception, in the absence of GPU onboard, eFolki will be replaced by an efficient dense stereo-matching algorithm, such as SGBM [13]. We are also working on long-term modeling, including loop closure detection and the associated correction of the 3D model. Finally, we also intend to make use of recent advances in computational photography to obtain 3D data with more compact and lightweight co-designed sensors, such as the 3D chromatic depth-from-defocus camera presented in [44] ■

Acronyms

CPU	(Central Processing Unit)
eVO	(Efficient Visual Odometer)
GPS	(Global Positioning System)
GPS-RTK	(GPS Real-Time Kinematic)
GPU	(Graphics Processing Unit)
IMU	(Inertial Measurement Unit)
KITTI	(Karlsruhe Institute of Technology and Toyota Technological Institute)
KLT	(Kanade-Lucas-Tomasi (feature tracker))
MAV	(Miniature Aerial Vehicle)
RANSAC	(Random Sampling Consensus)
RGBD	(Red, Green, Blue + Depth (4-channel cameras))
ROS	(Robotic Operating System)
SGBM	(Semi-Global Block Matching)
SIMD	(Simple Instruction Multiple Data)
SSD	(Sum of Squared Differences)
UAV	(Unmanned Aerial Vehicle)
ZNCC	(Zero-mean Normalized Cross-Correlation)

References

[1] F. ANDERT, F. ADOLF - *Online World Modeling and Path Planning for an Unmanned Helicopter.* Autonomous Robots, 27:147–164, 2009.

[2] P.F. ALCANTARILLA, L.M. BERGASA, F. DELLAERT - *Visual Odometry Priors for Robust Ekf-slam*. IEEE International Conference on Robotics and Automation (ICRA), pages 3501–3506, 2010.

[3] M. ACHTELIK, A. BACHRACH, R. HE, S. PRENTICE, N. ROY - *Stereo Vision and Laser Odometry for Autonomous Helicopters in gps-denied Indoor Environments*. Proceedings of the SPIE Unmanned Systems Technology XI, volume 7332, Orlando, Florida, 2009.

[4] G. LE BESNERAIS, F. CHAMPAGNAT - *Dense Optical Flow Estimation by Iterative Local Window Registration.* IEEE International Conference on Image Processing (ICIP), pages 137–140, Genova, Italy, September 2005.

[5] J.Y. BOUGUET - *Pyramidal Implementation of the Affine Lucas Kanade Feature Tracker - Description of the Algorithm*. Technical report, Technical report. Intel Corporation, 2001.

[6] A. I COMPORT, E. MALIS, P. RIVES - *Accurate Quadrifocal Tracking for Robust 3d Visual Odometry*. IEEE International Conference on Robotics and Automation (ICRA), pages 40–45, Roma, Italy, April 2007. IEEE.

[7] A.J. DAVISON, I.D. REID, N.D. MOLTON, O. STASSE - *Monoslam: Real-time Single Camera Slam*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 29(6):1052–1067, 2007.

[8] A. ELFES - *Using Occupancy Grids for Mobile Robot Perception and Navigation*. Computer, 22(6):46–57, June 1989.

[9] M.A. FISCHLER, 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.

[10] F. FRAUNDORFER, L. HENG, D. HONEGGER, G. HEE LEE, L. MEIER, P. TANSKANEN, M. POLLEFEYS - *Vision-based Autonomous Mapping and Exploration Using a Quadrotor Mav*. IEEE/RSJ International Conference on Intelligent robots and systems (IROS), pages 4557–4564, Algarve, Portugal, October 2012.

[11] F. FRAUNDORFER, D. SCARAMUZZA - *Visual Odometry: Part ii - Matching, Robustness, and Applications*. IEEE Robotics and Automation Magazine, 19(2):78–90, June 2012.

[12] A. GEIGER, P. LENZ, R. URTASUN - *Are we Ready for Autonomous Driving? the Kitti Vision Benchmark Suite*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3354 – 3361, Providence, RI (USA), June 2012.

[13] H. HIRSCHMÜLLER - *Stereo Processing by Semiglobal Matching and Mutual Information.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2):328–341, 2008.

[14] A. HOWARD - *Real-time Stereo Visual Odometry for Autonomous Ground Vehicles*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3946– 3952, 2008.

[15] A. HORNUNG, K. M.WURM, M. BENNEWITZ, C. STACHNISS, W. BURGARD - *OctoMap: an Efficient Probabilistic 3D Mapping Framework Based on Octrees*. Autonomous Robots, 34(3), 2013.

[16] W. MORRIS, I. DRYANOVSKI, X. JIZHONG - *Multi-volume Occupancy Grids: an Efficient Probabilistic 3d Mapping Model for Micro Aerial Vehicles.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1553–1559, Taipei (Taiwan), October 2010.

[17] K. KONOLIGE, M. AGRAWAL, J. SOLA - *Large-scale Visual Odometry for Rough Terrain.* 13th International Symposium of Robotics Research, Hiroshima, Japan, November 2007.

[18] B. KITT, A. GEIGER, H. LATEGAHN - *Visual Odometry Based on Stereo Image Sequences with Ransac-based Outlier Rejection Scheme.* IEEE Intelligent Vehicles Symposium (IV), pages 486–492, San Diego, CA (USA), June 2010. IEEE.

[19] G. KLEIN, D. MURRAY - *Parallel Tracking and Mapping for Small ar Workspaces*. International Symposium on Mixed and Augmented Reality, Nara, Japan, November 2007.

[20] M. KAESS, K. NI, F. DELLAERT - *Flow Separation for Fast and Robust Stereo Odometry.* IEEE International Conference on Robotics and Automation (ICRA), pages 3539–3544, Kobe, Japan, May 2009.

[21] P. D. KOVESI - *MATLAB and Octave Functions for Computer Vision and Image Processing*. Centre for Exploration Targeting, School of Earth and Environment, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/_pk/research/matlabfns/>.

[22] J. KELLY, G. S. SUKHATME - *An Experimental Study of Aerial Stereo Visual Odometry.* IFAC Symposium on Intelligent autonomous vehicles, 2007.

[23] J. KELLY, S. SARIPALLI, G. SUKHATME - *Combined Visual and Inertial Navigation for an Unmanned Aerial Vehicle*. Christian Laugier and Roland Siegwart, editors, Field and Service Robotics, volume 42 of Springer Tracts in Advanced Robotics, pages 255–264. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-75404-6 24.

[24] M.I. A. LOURAKIS, A.A. ARGYROS - *SBA: a Software Package for Generic Sparse Bundle Adjustment*. ACM Trans. Math. Software, 36(1):1–30, 2009.

[25] G. LE BESNERAIS, F. CHAMPAGNAT - *Dense Optical Flow by Iterative Local Window Registration*. IEEE International Conference on Image Processing 2005, pages I–137. IEEE, 2005.

[26] B. D. LUCAS, TAKEO KANADE - *An Iterative Image Registration Technique with an Application to Stereo Vision*. IJCAI, volume 81, pages 674–679, 1981.

[27] H.P. MORAVEC, A. ELFES - *High Resolution Maps from Wide Angle Sonar*. International Conference on Robotics and Automation (ICRA), pages 116–121, St Louis, MI (USA), March 1985.

[28] E. MOURAGNON, M. LHUILLIER, M. DHOME, F. DEKEYSER, P. SAYD - *Real Time Localization and 3d Reconstruction*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, pages 363–370, 2006.

[29] A. MALLET, S. LACROIX, L. GALLO - *Position Estimation in Outdoor Environments Using Pixel Tracking and Stereovision*. IEEE ICRA, 2000.

[30] H.P. MORAVEC - *Robot Spatial Perception by Stereoscopic Vision and 3d Evidence Grids*. Technical Report CMU-RI-TR-96-34, Carnegie Mellon University, September 1996.

[31] V. VITTORI, M. SANFOURCHE, G. LE BESNERAIS - *evo: a Realtime Embedded Stereo Odometry for Mav Applications*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2107–2114, Tokyo, Japan, November 2013.

[32] C. MEI, G. SIBLEY, M. CUMMINS, P. NEWMAN, I. REID - *Rslam: a System for Large-scale Mapping in Constant-time Using Stereo*. International Journal of Computer Vision, pages 1–17, 2010. Special issue of BMVC.

[33] D. NISTER, O. NARODITSKY, J. BERGEN - *Visual Odometry for Ground Vehicles Applications*. Journal of Field Robotics, 23(1):3–20, 2006.

[34] A. PLYER, G. LE BESNERAIS, F. CHAMPAGNAT - *Folki-gpu: a Powerful and Versatile Cuda Code for Real-time Optical Flow Computation.* GPU Technology Conference, San Jose, CA (USA), October 2009.

[35] A. PLYER, G. LE BESNERAIS, F. CHAMPAGNAT - *Real-time Lucas-kanade Optical Flow Estimation for Real-world Applications*. 2014.

[36] E. ROSTEN, T. DRUMMOND - *Machine Learning for High-speed Corner Detection*. European Conference on Computer Vision, volume 1, pages 430–443, May 2006.

[37] P. J ROUSSEEUW - *Least Median of Squares Regression*. Journal of the American Statistical Association, 79(388):871–880, 1984.

[38] C. HUERZELER S. WEISS L. KNEIP R. VOIGT, J. NIKOLIC, R. SIEGWART - *Robust Embedded Egomotion Estimation*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2694–2699, San Francisco, Ca (USA), September 2011.

[39] D. DUBE S. A. SCHERER, A. ZELL - *Using Depth in Visual Simultaneous Localisation and Mapping.* IEEE International Conference on Robotics and Automation, St. Paul, Minnesota, USA, May 2012.

[40] D. SCARAMUZZA, F. FRAUNDORFER. *Visual Odometry: Part i - the First 30 Years and Fundamentals*. IEEE Robotics and Automation Magazine, 18(4):80–92, December 2011.

[41] D. SUN, S. ROTH, M. J. BLACK - *Secrets of Optical Flow Estimation and their Principles*. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, pages 2432–2439, 2010.

[42] S. SCHERER, S. SINGH, L. J. CHAMBERLAIN, M. ELGERSMA - *Flying Fast and Low Among Obstacles: Methodology and Experiments*. The International Journal of Robotics Research, 27(5):549–574, May 2008.

[43] J. SHI, C. TOMASI - *Good Features to Track*. 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 593 – 600, 1994.

[44] P. TROUVÉ, F. CHAMPAGNAT, G. LE BESNERAIS, J. SABATER, T. AVIGNON, J. IDIER - *Passive Depth Estimation Using Chromatic Aberration and a Depth from Defocus Approach*. Applied optics, 52(29):7152–7164, 2013.

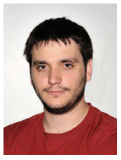
[45] S. UMEYAMA - *Least-Squares Estimation of Transformation Parameters Between Two Point Patterns*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 13(4):376–380, April 1991.

[46] R. ZABIH, J. WOODFILL - *Non-Parametric Local Transforms for Computing Visual Correspondence*. Computer Vision—ECCV'94, pages 151–158, 1994.

[47] M. MEILLAND, A.I. COMPORT, P. RIVES - *Dense Visual Mapping of Large Scale Environments for Real-Time Localisation*. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francsiso, 2011.

AUTHORS

Martial Sanfourche graduated from Universite de Cergy-Pontoise in computer Sciences (2001) then received the Ph.D. degree in image and signal processing from the Universite de Cergy-Pontoise in 2005. After a postdoctoral position at CNRS-LAAS, he joined Onera/DTIM in 2007 where is now a research engineer in computer vision. His current research interest include online and offline visual localization and mapping for robotic systems.



Aurelien Plyer graduated from Universite Pierre et Marie Curie (Paris 6) in 2008 and received the Ph.D degree in Image Processing from the Universite de Paris 13, in 2013. His research deals with low level video processing and 3D environement perception for robotics, he uses GPU programming in order to implement real-time processing.

Anthelme Bernard-Brunel holds a technological university level diploma in Electrical and Computer Engineering issued by the IUT de Ville d'Avray (2012). Since, Anthelme follows an apprenticeship for being graduated in electrical engineering and computer science. It alternates between courses at UPMC Polytech 'Paris and his job at the Onera/ DTIM.

Guy Le Besnerais graduated from the Ecole Nationale Superieure de Techniques Avancees in 1989 and received the Ph.D. degree in physics from the Universite de Paris-Sud, Orsay, France, in 1993. He joined the Onera in 1994, where he is now a senior scientist in the Information Processing and Modelization Department. His work concerns inversion problems in imagery and computer vision.