HAL open science

Visual localization and servoing for drone use in indoor remote laboratory environment

Fawzi Khattar, Franck Luthon, Benoît Larroque, Fadi Dornaika

To cite this version:

Fawzi Khattar, Franck Luthon, Benoît Larroque, Fadi Dornaika. Visual localization and servoing for drone use in indoor remote laboratory environment. Machine Vision and Applications, 2021, 32 (1), pp.32. 10.1007/s00138-020-01161-7 . hal-03126470

HAL Id: hal-03126470

https://cnrs.hal.science/hal-03126470v1

Submitted on 31 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Noname manuscript No.
(will be inserted by the editor)

Visual Localization and Servoing for Drone Use in Indoor Remote Lab Environment.

Fawzi Khattar · Franck Luthon · Benoit Larroque · Fadi Dornaika.

Received: date / Accepted: date

Abstract In this paper, we present a localization system for the use of drone in a remote lab. The objective is to allow a drone to inspect remote electronic instruments autonomously, as well as to return to its base and land on a platform for the recharge of its batteries. In addition, the drone should be able to detect the presence of a teacher in the lab, and to center the human face in the image in order to enable remote student-teacher communication. To achieve the first objective, the localization approach is composed of a monocular SLAM (Simultaneous Localization and Mapping) algorithm PTAM (Parallel Tracking and Mapping) and an estimation based on the homography transform. For the face-drone servoing, the approach is based on the 3D Candide model. Both approaches work in real-time. Quantitative and qualitative experiments are presented that show the robustness of both methods.

Keywords 3D Control command, · Face tracking · Homography · Remote Laboratory · Monocular · Planar object · Pose estimation · Quadcopter · SLAM · Template matching.

Fawzi Khattar
LIUPPA Computer Science laboratory at the University of Pau & Pays Adour / E2S UPPA, Anglet 64600, France.
E-mail: fawzi.khattar@gmail.com

Franck Luthon
LIUPPA Computer Science laboratory at the University of Pau & Pays Adour / E2S UPPA, Anglet 64600, France.
E-mail: franckluthon@iutbayonne.univ-pau.fr

Benoit Larroque
SIAME Engineering Science laboratory at the University of Pau & Pays Adour / E2S UPPA, Anglet 64600, France.
E-mail: benoit.larroque@univ-pau.fr

Fadi Dornaika
University of the Basque Country UPV/EHU, San Sebastian, Spain.
IKERBASQUE, Basque Foundation for Science, Bilbao, Spain. E-mail: fadi.dornaika@ehu.eus

1 Introduction

This paper presents an implementation of the idea on how computer vision and drones can enhance remote lab experience for students in STEM education (Science, Technology, Engineering, and Mathematics). As technology is evolving at a fast pace, learning methods must change and adapt in order to be effective. Nowadays students have access to the latest technologies (smart phones, touch-pads, drones, virtual reality games ...), but most of them are being taught in the same old fashioned learning style: students attend courses and do their traditional hands-on lab work, followed later on by exams. However this learning style is not the best to ensure good learning outcomes today. In particular for practical work, students are tied to certain schedules and dedicated rooms in order to do their hands-on training. Remote labs can solve this problem by giving students the freedom to do their lab work from anywhere and at anytime. Furthermore, most of the students do not find motivation in studying certain subjects because they do not see their use in their every day life and because they find the material boring. A great way to overcome this problem is by incorporating new technologies and game like approaches in the learning process. In this paper, we use the remote lab platform called LaboREM (for remote laboratory), that comes with motivating elements such as a robotic arm and a mobile camera in the learning process. For more detailed information about LaboREM features and its use, the reader is referred to [1]. The objective of this lab is to enable students to perform electronic experi-

ments remotely by allowing them to control a robotic arm and a drone. The robotic arm replaces the student's hand while the drone equipped with a camera replaces his eyes by sending visual feedback of the remote experiment, in particular the result of an experiment displayed on the front panel of electronic measurement instruments. Another use for the drone is to allow remote student-teacher interaction in case there was a teacher available in the remote lab facility at the time of the experiment. For a more detailed explanation of the use of the drone in LaboREM the reader is refered to [2].

In this paper, we present the implementation done in order to allow the drone to achieve the two objectives. The paper is organized as follows: section 2 presents related work and approaches for the visual localization of a robot. In section 3, we present the control system with the feedback loop for different scenarios. Section 4 explains the localization approach used for the object inspection scenario. In section 5, we present our approach for the localization of a drone with respect to a face. In section 6, we present quantitative and qualitative experiments of both approaches.

2 Related Work

As previously stated, the objective is to allow a drone to autonomously navigate a 3D indoor environment to search for front panels of instruments or for a human face. After the mission is done, the drone must be able to return back home and land on a platform for the automatic recharge of its batteries. In order to accomplish this task, a continuous estimate of the 3D pose of the drone must be available as well as a control system that calculates the appropriate commands to be sent to the drone to make it fly from an initial 3D position to a desired 3D position.

Many sensors can be used for the pose estimation of a quadcopter. For outdoor environments, GPS (Global Positioning System) sensors are the best solution to get a coarse localization of a quadcopter. In [3] authors propose a visual SLAM (Simultaneous Localization And Mapping) system that fuses GPS data with visual estimates during initialization in order to estimate the pose of the drone. It was shown that, relying on the GPS after the initialization step actually increases the error, so the authors in [3] proposed to use only a visual estimate in addition to other inertial measurements after this step.

For indoor environments or GPS denied environments, a localization system must rely on other sources of information such as cameras (monocular, stereo or depth), IMU (Inertial Measurement Unit), LIDAR (Light Detection and Ranging) or others. Sensors can be embedded on the drone or can be placed in the environment. The latter case is more suitable for constrained environments where the drone will be always in the same 3D environment, as a change in the environment implies re-installation of the sensors in the new environment. Furthermore, artificial markers can be placed in the scene to facilitate the task of localization [4]. These markers can be reflective and detected by an external localization system deployed in the environment that gives accurate position estimate. In [5] authors estimate the 3D pose of a quadcopter in an outside environment by using bundle adjustment [6] across several fixed cameras subject to dynamic constraints of a quadcopter. The detection of the quadcopter is done using background subtraction and cross correlation. Although deploying sensors and artificial markers in the environment can be an easy solution to the problem of localization, this is not always possible (for example in railway inspection applications [7]). Deploying sensors on the drone offers a more flexible approach, as it allows for easy plug and play scenarios. This implies also that the localization algorithms must be able to handle different types of environments. On-board stereo rig cameras [8] and RGBD cameras [9] have been investigated. They have the advantage that they allow for absolute pose estimate. However this comes with an additional weight and power consumption that is critical in drone applications.

Monocular cameras offer a good trade-off between weight, consumption and the information it can provide. Although monocular SLAM algorithms are not sufficient for absolute localization due to the scale ambiguity problem arising from the projection of the 3D world into a 2D image, adding more information from other sensors or knowledge about objects available in the environment can compensate for this.

The first approach to solve the SLAM problem was using EKF (extended Kalman filter) with a state vector composed of the robot pose and all the landmark positions in the map. Since the state vector is of dimension $6+3\times n$ where n is the number of landmarks in the map and 6 is the number of DOF (degrees of freedom) defining the pose of the robot in 3D space (if it is a flying robot), sensor updates require computation time that is quadratic in the number of landmarks. This complexity stems from the fact that the covariance matrix maintained by the Kalman filter has $O(n^2)$ elements, all of which must be updated even if just a single landmark is observed [10]. Because of its computational cost, EKF SLAM can only be used with a small number of landmarks.

After EKF-SLAM, FastSLAM [11] was proposed to solve the above mentioned problem. It is based on the observation that the problem of estimating landmark positions can be decoupled into n independent estimation problems given that the robot path is known. Thus the problem is decomposed into $n + 1$ problems: the problem of estimating the robot pose and n estimation problems for the n landmark positions. FastSLAM uses a particle filter framework where each particle has its own estimate of the robot path, and n independent Kalman filters to estimate the position of the landmarks. This framework is of complexity $O(M \log(n))$ where M is the number of particles. Afterwards, a new approach for SLAM was introduced, that makes use of the idea that consecutive frames in a video contain highly redundant information. In other words, not all the frames should be used for bundle adjustment. Only some frames, called key frames, are used for map refinement which highly reduces the amount of computation needed for bundle adjustment and allow for real time applications. The first algorithm to follow this approach is PTAM (Parallel Tracking And Mapping) [12]. It separates the tracking (pose estimation of the camera) and the mapping (building 3D map of the environment) into two separate tasks running on parallel threads which allow real time SLAM even on mobile devices. The keypoint-based tracking algorithm extracts FAST corners [13] and tracks them through the video using SSD (sum of squared differences) performed on an image pyramid of the current frame constrained by the current estimation on the camera pose and a motion model.

All previous approaches decompose the SLAM problem into two steps: (i) feature extraction, and then (ii) pose estimation, map building and refinement. While this simplifies the problem, lots of information available in the image is not used. For example, features lying on edges are not extracted as they are not considered as good points to track. To make use of this information, other types of methods use direct image alignment to estimate the pose and build the map [14]. In [15], authors propose a framework called LSD-SLAM for solving SLAM problems based on direct image alignment on pixels with high gradients, leading to a semi dense map. Their denser reconstructions, as compared to the sparse point map of feature-based SLAM systems, could be more useful for other tasks than just camera localization. Their system is able to run on a CPU and to build large scale maps.

In [16], authors present a feature-based SLAM system using ORB features (Oriented FAST and Rotated BRIEF) for their robustness and real-time capabilities [17]. Their main contribution is to extend the versatility of PTAM to environments that are intractable for that system. In their paper, they compare ORB-SLAM to PTAM and LSD-SLAM in terms of accuracy and of ability to relocalize after tracking failure. In terms of accuracy, they state that ORB-SLAM and PTAM are similar in open trajectories and outperform LSD-SLAM. As for relocalization, they find that PTAM is only able to relocalize frames which are near to the keyframes due to the little invariance of its relocalization method, while ORB-SLAM accurately relocalizes more than the double of frames than PTAM could.

In this paper, we build on the system presented in [18], for localization and control of a low cost drone (AR Drone 2.0) in unknown environment. Their system is based on the fusion of PTAM pose estimates coupled with sensor measurements in a Kalman filter for localization. For control, they use traditional 4 feedback loops with PID controllers. Since PTAM is a monocular SLAM algorithm, and in order to allow absolute navigation in the environment, they propose a maximum likelihood algorithm that uses ultrasound measurement in addition to PTAM to recover the scale of the map during an initialization step.

However as discussed before, PTAM has limitations in the types of scenes it can handle, in particular scenes with not enough features. In our scenario of remote lab, the drone must be able to navigate different types of environments: the whole remote lab facility to get a broad view of the lab, as well as the space near electronic devices in order to center them in the image and let the student see the result of the practical experiment. The latter environment can be difficult for PTAM to track and, as we will show later, PTAM will loose tracking in this scenario. In order to deal with this, and to extend this approach, we propose here to use the object of reference itself as a landmark for pose estimation when the drone is navigating this environment. Furthermore, to be able to control the drone to center the face of a teacher in the image, the position of the drone with respect to the face of the teacher must be estimated. For this purpose, we use the Candide deformable 3D model of the face and fit it to the image using facial landmarks extracted from the face using [19]. These are the two main contributions of the current paper [20].

3 Visual control of the quadcopter

The drone used in this work is the low-cost AR Drone 2.0 shown in Figure 1.

In order to control the 3D position and orientation of the quadcopter, a closed-loop control is used, taking as a feedback the video stream and sensors data as shown in Figure 2.

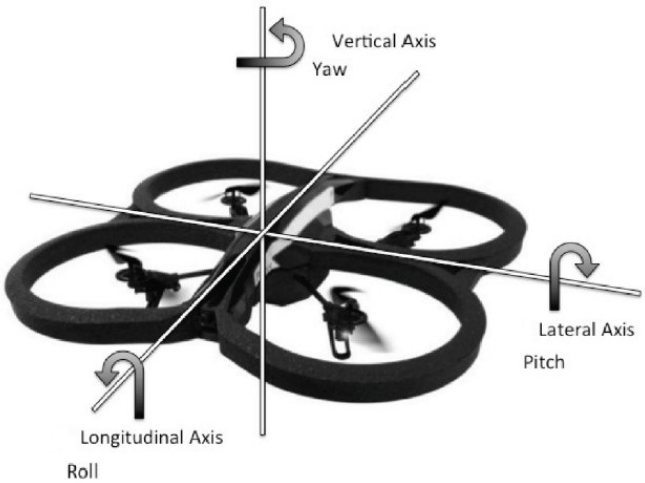


Fig. 1: AR Drone 2.0 with attached referential system and angle conventions (roll, yaw and pitch) [21].

If the objective is object inspection, the pose of the drone is estimated using one of two blocks (block (a) or (b) in Figure 2) depending on the distance between the camera and the object of interest. PTAM is used in normal case when the drone is exploring the environment and is far from the object. If the drone is near the object, a localization system based on the homography transform as explained in the next section is used instead.

If the objective is to control the drone to maintain a relative position from a human face, the algorithm explained in section 5 is used as a feedback sensor for the control loop (shown in dashed line in Figure 2). The controlled degrees of freedom associated with the quadcopter are the 3D translation vector and the yaw angle. Each degree is controlled by a closed loop control system with a traditional PID controller.

4 Object-Camera pose estimation

Planar objects are a well defined type of objects that are widely available in human made environments. Incorporating the information that the object of interest is planar is of great benefit for object-camera pose estimation. The homography matrix is a matrix that relates 3D points lying on a plane to their 2D projections. Given this transform, one can directly calculate the rotation and translation matrix as done in [22]. In order to estimate the homography that maps any plane into another plane by means of perspective projections, several methods can be used. These methods are usually classified into local (feature-based) and global (featureless) methods.

Given a template image of the planar object, local methods extract local key-points and attribute a descriptor to each of them both in the template image and in the current image. After this step, key-points (at least four key-points) in both images are matched according to a similarity metrics performed on the descriptors. Given the point correspondences, the homography matrix is estimated using robust methods like RANSAC (Random Sample Consensus) in order to deal with the presence of outlier correspondences. Local methods can work well with no prior information on the homography parameters. However in some cases, the robust computation may be computationally expensive and not work in real time. A survey about key-point detectors and descriptors can be found in [23].

On the other hand, global methods use all the information in the image and attempt to find the homography matrix that best aligns the template patch to the test image. This process however gives rise to non-linear minimization (NLM) problems, that can be solved using iterative algorithms like gradient descent or LM (Levenberg-Marquardt) [24]. Thus, a good initialization is necessary to guarantee the convergence of those algorithms. Different similarity functions exist to measure the degree to which two patches are aligned, the most used ones being the SSD and the enhanced correlation coefficient (ECC) [25]. In practice, the first one uses a brightness model in order to cope with variation of additive and multiplicative change of illumination [26], whereas the latter is by definition insensitive to those illumination changes. These methods have the advantage that they can run in real time and give good results if a coarse estimate of the homography parameters is known. Thus the two families of methods are complementary. The first one is robust with no priors needed but computationally expensive, while the second is fast and works well if a prior is available.

4.1 Proposed approach

Here, both approaches are used in order to estimate the 3D pose of the quadcopter with respect to the object of interest as shown in Figure 3. The first approach is used for detecting the object of interest as well as for recovering from a tracking loss. The second one is used in the tracking process. The approach is divided into two steps: detection and tracking [27].

In the detection step, template matching on a pyramid of the image is used to search for the desired object. If the normalized correlation coefficient is greater than a predefined threshold α, the object is declared detected. Once the detection is done, a homography

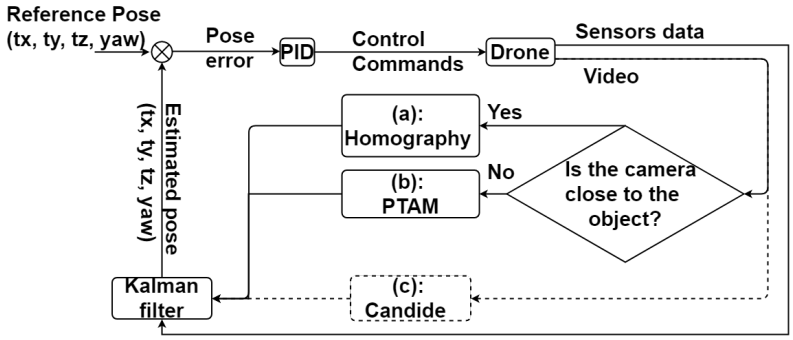


Fig. 2: Feedback control loop.

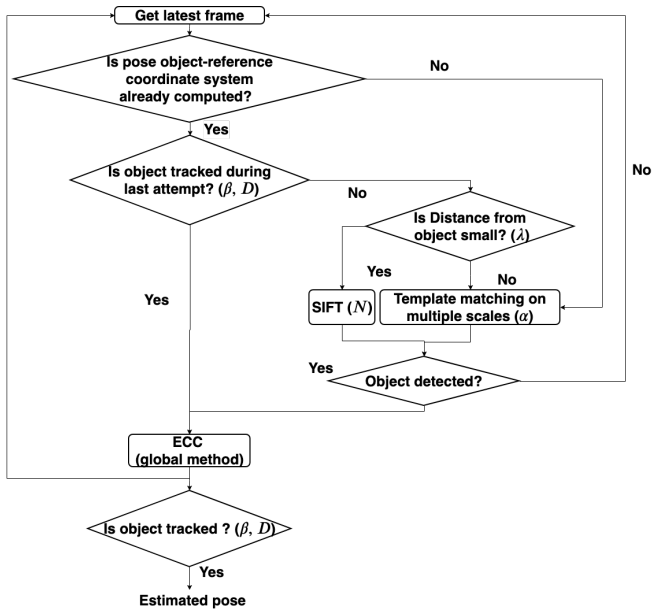


Fig. 3: Homography estimation diagram.

transformation is computed by using the refined bounding box of the detected object to determine the object-camera relative pose. A command is then sent to the drone in order to move it closer to the object. Template matching is used to allow successful detection of the object despite its long distance from the camera and its small size, as key-points detector fails to detect and put in correspondences key-points if the object of interest doesn't occupy a certain amount of image pixels. However once the distance between the camera of the drone and the instrument is less than a threshold λ, the SIFT (Scale Invariant Feature Transform) descriptor [28] is used to allow more robustness to orientation changes.

The object is declared detected if the number of matched key-points is greater than a predefined value N. Once the object is detected, the tracking stage begins. As a rough estimation of the homography matrix is available from the detection stage, it is used as an initial solution for the next frame, and the ECC algorithm is applied to estimate the homography in this frame. The homography estimation is propagated in this way from one frame to the next one, and used as a prior for the ECC algorithm. However, sometimes the ECC algorithm will fail to converge due to several reasons. For example, communication problems between the quadcopter and the computer makes the latest estimated homography not close enough to the real solution of the current frame, which prevents the algorithm convergence. Besides, the image quality can be degraded by motion blur or H264 encoding/decoding problems. In this work, tracking loss is declared if the ECC algorithm is unable to converge or if it converges to an unrealistic estimation. At each frame, we compute the 3D pose of the quadcopter with respect to the planar object. By monitoring the estimated traveled distance between two consecutive frames and comparing it to

a threshold D, we can detect a loss of tracking. Another threshold β is also imposed on the difference of each angle of orientation (yaw, roll, pitch). If the tracking fails, we resort back to the local method (SIFT) if the quadcopter-object distance is relatively small, or to the template matching method in the other case, to reinitialize the ECC tracker as shown in Figure 3. This pose estimate is fused with inertial measurements sent by the drone in a Kalman filter framework in order to smooth this estimate, and to provide robustness when the visual tracker fails. The Kalman filter is used also to compensate for time delays as done in [18].

4.2 Parameter setting

The proposed approach contains parameters that need to be chosen. In our experiments, we choose the following values for the various parameters. As regrads the ECC, a correlation threshold α of 0.8 gave good results for different types of objects. For the N parameter (required for SIFT) depending on the type and the size of the object, different values were chosen; however we always set it to a number between 10 and 30. β and D were set to 15 degrees and 30 cm respectively, in order to test if the object tracking is converging or not. For λ, different values can be used depending on the size of the object. In our case, it was set to a number between 1 and 3 meters.

5 Face-Camera pose estimation

Human-drone interaction is a new way of controlling drones. In [29] authors use face pose and hand gestures in order to allow human-drone interaction. Their face pose estimation process is based on the Viola & Jones face detector [30]. They compute a face score vector by applying frontal and side face detector on the flipped and the original image. Using this face score and a machine learning technique, they estimate the *yaw* angle of the face pose. The distance from the face is estimated by the size of the face bounding box. Hand gestures are used to give orders to the drone to move to an orientation while maintaining the distance from the face. In [31], authors also use hand gestures and face localization for drone-human interaction. Their approach is unique for the fact that it allows the drone to approach a human that is 20 meters away, by detecting periodic hand gestures. The drone then approaches the target by tracking its appearance. Once at a short distance, the drone centers the face of the subject and detects hand gestures in order to take a picture. However, the orientation of the face is not estimated and the user has to be facing the camera in order to take a frontal photo.

In the current work, we adopt a 3D approach that models the human face in 3D and subsequently uses full perspective projection in order to recover the 3D face pose parameters. By using this modeling, and matching it with image specific data related to the face, all the 6 pose parameters are inferred. The 3D modeling is based on the Candide deformable 3D face model.

5.1 CANDIDE 3D model

CANDIDE is a parameterized 3D face model specifically developed for model-based coding of human faces. CANDIDE is controlled by 3 sets of parameters: global, shape and animation parameters. The global parameters correspond to the pose of the face with respect to the camera. There exist 6 global parameters: 3 Euler angles for the rotation and 3 for the translation (t_x, t_y, t_z). The shape parameters adjust facial features position in order to fit to the morphology of different subjects (eye width, distance between the eyes, face height etc). The animation parameters adjust facial features position in order to display facial expressions and dynamic animations (smile, lowering of eyebrows). The 3D generic model is given by the 3D coordinates of its vertices $P_i, i = 1, n$. where n is the number of vertices. This way, the shape, up to a global scale, can be fully described by a $3n$ vector g, the concatenation of the 3D coordinates of all vertices:

$$g = G + S\tau_s + A\tau_a \tag{1}$$

G is the standard shape of the model, the columns of S and A are the shape and animation units, $\tau_s \in \mathbb{R}^m$ and $\tau_a \in \mathbb{R}^k$, are the shape and animation control vectors, respectively.

5.2 Inferring pose parameters

In order to determine the pose from the 3D model, we have to fit this model to the face data available in the image. Fitting the model means determining its different parameters: pose, shape and animation parameters. In this work, only pose and shape parameters are of interest for us, however recovering the animation parameters could be a relevant way to allow human-drone interaction based on facial expressions. Different approaches attempt to adapt the model in different ways. However, the majority of them follow a step by step approach, starting by estimating the shape parameters τ_s in order to adapt the 3D model to different face anatomy

and then estimating the pose and animation parameters. From the face image, many face related data can be used to fit the 3D model. In [32], the authors use the gray scale appearance of the image to adapt the 3D model after estimating its shape parameters off-line. In our work, we make use of the advancement in facial landmark detection and use these landmarks to adapt the model and recover the 3D face pose from a set of 3D-to-2D correspondences. The shape parameters are estimated using a frontal picture of the subject following the method described in [33]. We use the facial point detector in [19], that can detect 68 2D landmarks on a face in one millisecond by a pre-trained ERT (Ensemble of Regression Trees), given that a face image patch is available. However, since the algorithm needs a region of interest that contains a face, the total time for its execution from the detection of the face to the detection of the landmarks is more than one millisecond due to the computationally expensive face detection step. One way to reduce this time and make the process work at more than 30 fps (frames per second) is to perform a search for the face around the latest detected bounding box of the face, instead of searching for the face in the whole image. We make use of only 46 points from the 68 points given by the landmark detector. The points are chosen to be semantic and mostly rigid, thus eliminating points along face contour. Once the 2D landmarks are detected in the image, we use state of the art pose estimation algorithms that are based on 3D-2D point correspondences to recover the pose. This problem is known in the literature as PnP (Perspective n Point). Many algorithms attempt to solve this problem. The P3P algorithm [34] (perspective 3 point) can estimate the pose using only 3 point correspondences. Other algorithms like EPNP [35] (Efficient Perspective N Point) can handle any number of points. Another approach is to use non-linear minimization techniques to recover the pose that best minimizes the distance between the projected 3D points and the 2D points. However, this method requires an initial guess of the pose parameters in order to converge to the global minimum. This initial guess can be made available using the estimated pose from the previous frame or using any closed-form solution like EPNP, P3P, etc. in case it is not available. We propose to use the Levenberg-Marquardt technique as it gives good results and fast execution time. The face-camera pose estimation process is shown in Figure 4. The pose used to control the drone is computed by fusing the visual pose from the 3D model with inertial and ultrasound measurements in a Kalman filter as done in [18].

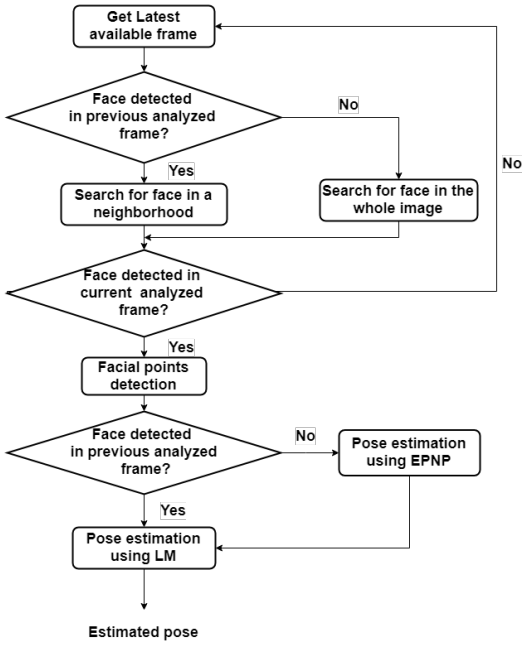


Fig. 4: Face-camera pose estimation.

6 Experiments

Before presenting the different scenarios for evaluation, we begin with an evaluation of the performance of the face-camera pose estimation. We compare the accuracy of different techniques and their execution time on a database for pose estimation (UPNA head pose database) [36]. The UPNA database contains 120 videos corresponding to 10 different subjects, 12 videos each, in which the subject changes its head pose by following guided and free movement. The ground-truth relative 3D face motion is known for all frames in all videos. From ours tests, we conclude that all the techniques converge to the same optimal solution, when followed by a non-linear minimization (NLM), namely Levenberg-Marquardt optimization (LM). As shown in Table 1, all methods give comparable results for t_x, t_y, t_z and *roll*. However the first two methods give bad estimation for the *yaw* and *pitch* degrees of freedom. High error or noise in these parameters will result in amplifying noise in the feedback control loops for the control of the drone. Thus the Levenberg- Marquardt method, that tracks the 3D face from a frame to the next one based on an initial estimate, is chosen for pose estimation and servoing. If the tracking fails at a certain time, we use EPNP for initialization.

In order to evaluate the proposed implementation of 3D pose estimation and 3D pose-based servoing for both objectives, we design three different scenarios. These scenarios are the following: behavior of the system in

Table 1: Average pose errors and computation time for different face pose estimation methods. t_x, t_y, t_z are in millimeters, *roll*, *yaw*, *pitch* in degrees, time in milliseconds.

Method	$\mathbf{t_x}$	$\mathbf{t_y}$	$\mathbf{t_z}$	**roll**	**yaw**	**pitch**	**time**
EPNP	11,84	7,11	12,67	0,55	3,74	2,39	0.117
Ransac P3P	12,50	7,79	18,34	1,56	6,52	6,11	0.898
EPNP + LM	11,51	7,23	13,38	0,56	2,28	1,45	0.363
Ransac P3P + LM	11,51	7,23	13,38	0,56	2,28	1,45	1.138
LM	11,51	7,23	13,38	0,56	2,28	1,45	0.234

response to perturbations when asked to inspect an object, autonomous visual inspection of planar object, and drone-face visual servoing.

6.1 Response of the system when faced with perturbations using the object as a localization landmark

In this set of experiments, the drone is subject to several impulse along each of its DOF separately. The localization system is the one based on the homography transform. First we give impulses to the quadcopter to push it left and right along its x axis. Figure 5 shows the pose estimation of the drone.

One can see how the impulse given to the drone can cause the tracking of the object to fail (pose estimate drawn in black) because of their amplitude and because the object is out of the field of view of the quadcopter in some cases. However despite this fact and since the estimated pose is not only based on the visual modality but also on fusing sensor measurements and a prediction model in a Kalman filtering framework, the pose estimation is accurate enough to bring the quadcopter to a position where the visual tracking will resume, and the object can be centered again in the image very fast (pose estimate drawn in green in Figure 5). Specifically the roll angle measured by the inertial sensor and the speed estimate given by the bottom camera of the drone, in addition to the prediction model of the Kalman filter, are responsible to push the drone back to its position and make the tracking recover.

The same observation can be made for the y axis in Figure 6, where the pitch estimate from the inertial sensor, in addition to speed estimate from the bottom camera and the prediction model of the Kalman filter, will help the recovery process.

For the yaw DOF, the inertial measurement of it will help the tracking recover as shown in the experiment of Figure 7. However for perturbation along the z axis, the approach was not found robust as shown in Figure 8. And this can be explained as follows: in fact, the only information other than the visual pose estimate for the elevation of the drone is the ultrasound sensor that measures its elevation from the ground. In the Kalman filtering framework, this has been incorporated by taking the difference of successive readings of this sensor as an observation of its vertical speed. However, since this sensor is highly affected by uneven ground surface, where its value can increase or drop instantaneously when the drone is not hovering above a flat area, only differences that are lower than a certain threshold γ (typ. default value $\gamma = 10cm$) are taken into account as observations in order to filter out false measurement. In the case where the drone is pushed up or down instantly the difference of the elevation measurement gets above the threshold, and thus these observations are not taken into account by the Kalman filtering framework. Thus the drone can stay up or down in extreme perturbation on the z axis as shown in Figure 8 where the drone was not able to fly back and recover after the last impulse. To solve this issue one could increase the threshold γ, however the performance of the drone in normal situation and the control of its elevation will be affected. A video of such an experiment can be found in [37].

6.2 Scenario realization: take off, object localization, examination and return to base

In remote lab context, a typical scenario is the following: the remote student will send a command to the quadcopter to go and inspect an electrical device. After receiving this command, the server tells the quadcopter to carry out the different tasks in order to accomplish the mission. The several steps are shown in Figure 9. A video of the experiment can be found in [38].

6.2.1 Stage 1 - System initialization

In this stage, the drone takes off, initializes the SLAM algorithm and estimates the scale of the 3D map. As it is shown in the plot of the z position against time in Figure 9, the drone is asked to perform an upward and downward displacement after SLAM initialization. The objective is to change its height in order to gather data needed for scale estimation of the 3D map (difference of vertical displacement between ultrasound sensor and SLAM algorithm). For this purpose the drone

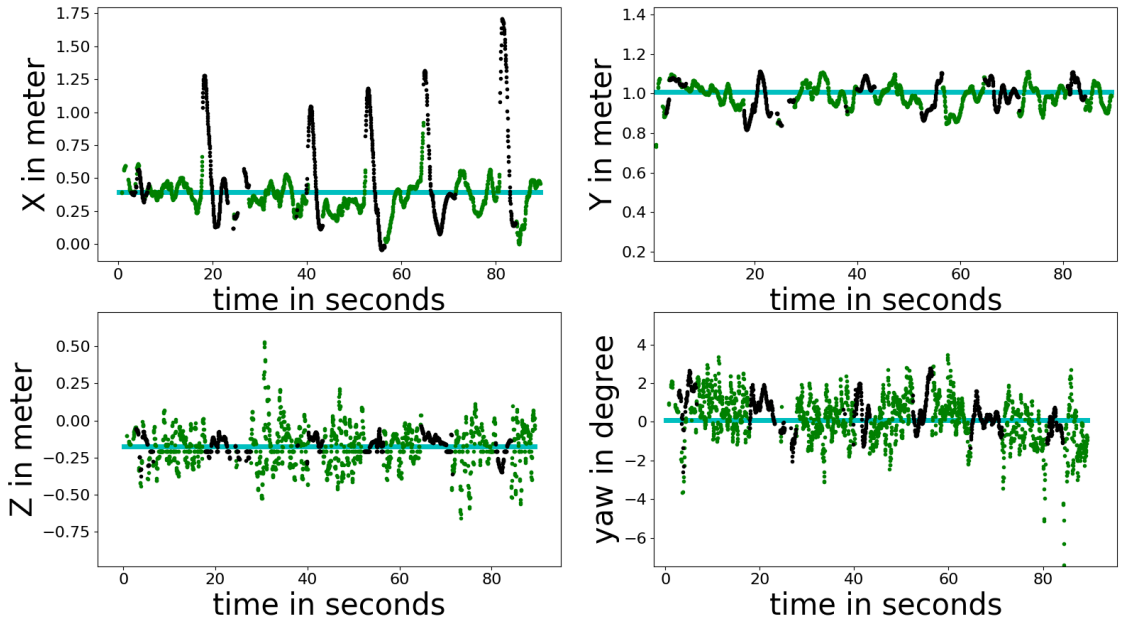


Fig. 5: Estimated pose of the quadcopter when faced with perturbations along the x axis. In green: the estimated pose when the visual tracking of the object is working; in black: the estimation is based solely on navigation data and the Kalman filter prediction model when the visual tracking fails. Cyan horizontal lines depict the desired 3D pose.

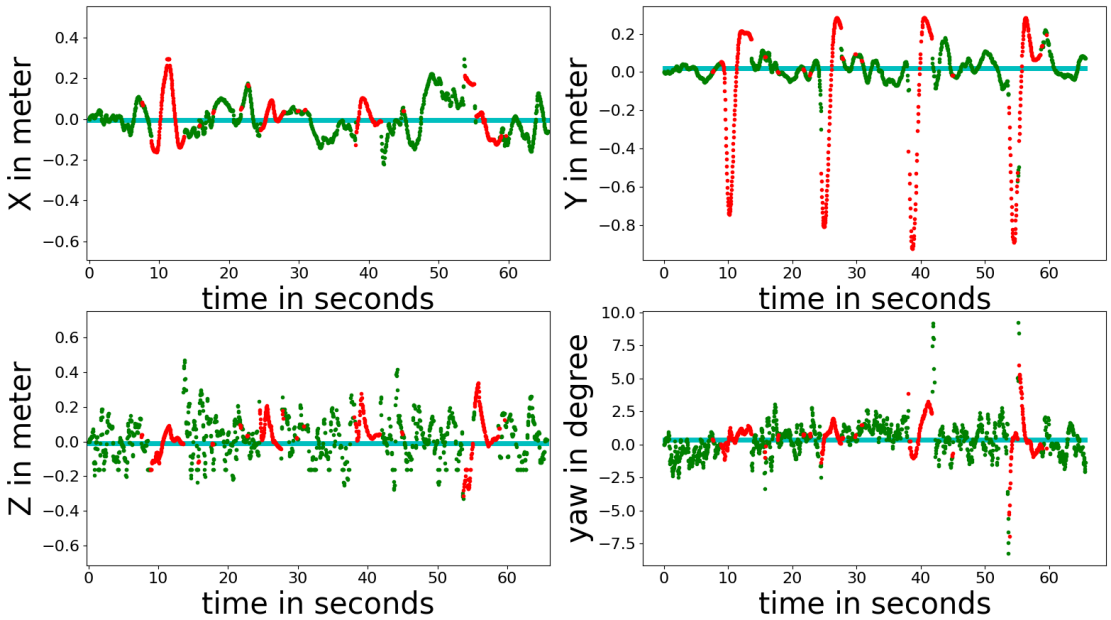


Fig. 6: Estimated pose of the quadcopter when faced with perturbations along the y axis. In green: the estimated pose when the visual tracking is working; in red: the estimation is based solely on navigation data and the Kalman filter prediction model when the visual tracking fails. Cyan horizontal lines depict the desired 3D pose.

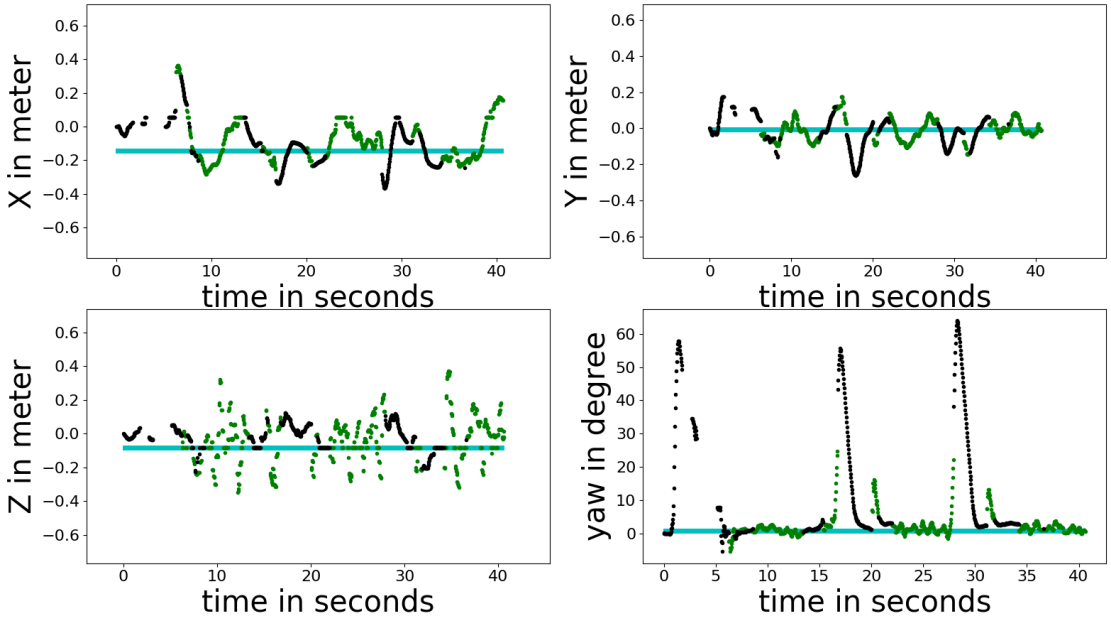


Fig. 7: Estimated pose of the quadcopter when faced with perturbations along the yaw DOF. In green: the estimated pose when the visual tracking is working; in black: its the estimation based solely on navigation data and the Kalman filter prediction model when the visual tracking fails. Cyan horizontal lines depict the desired 3D pose.

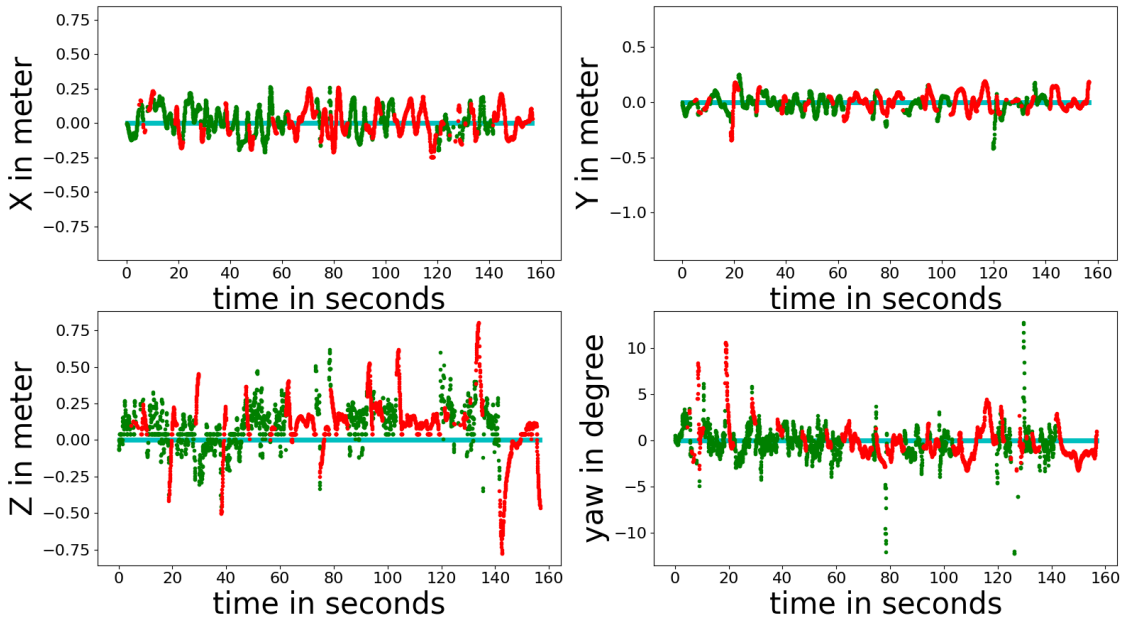


Fig. 8: Estimated pose of the drone when faced with perturbations along the z axis. In green: the estimated pose when the visual tracking is working; in red: the estimation is based solely on navigation data and the Kalman filter prediction model when the visual tracking fails. Cyan horizontal lines depict the desired 3D pose.

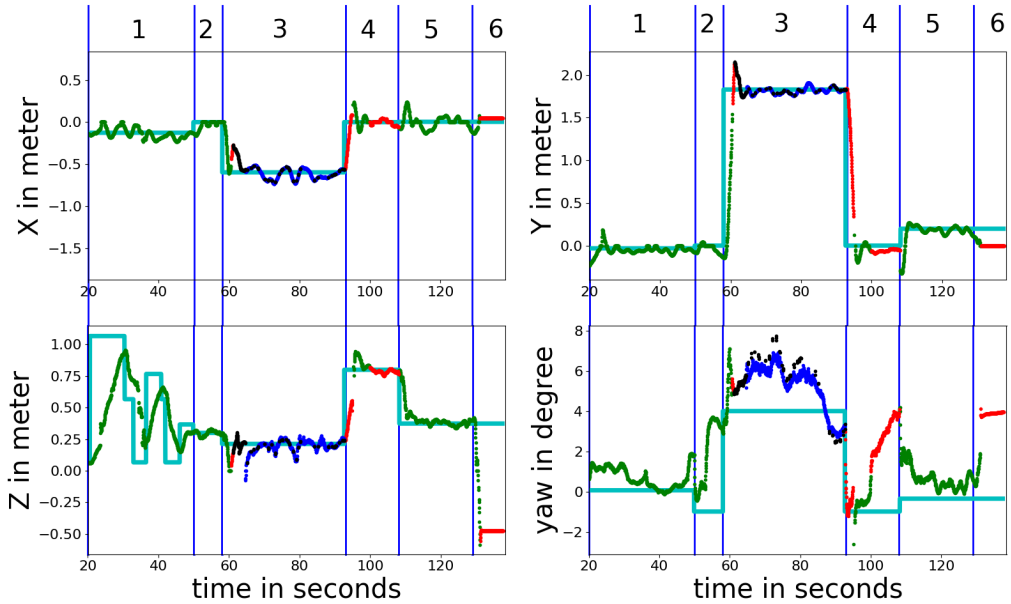


Fig. 9: Pose estimate of the object inspection scenario. Numbers 1 to 6 correspond to the stages explained in the text. In green: the estimated pose when the visual tracking of SLAM is working; in red: the estimation based solely on navigation data when SLAM loses tracking; in blue: the estimated pose from the homography transform while inspecting the object; in black: when the object tracking fails. Cyan curves depict the desired 3D pose.

is asked to perform a slow and steady ascending, descending movement for some seconds.

6.2.2 Stage 2 - Object detection and localization

After the scale is estimated, the instrument is searched in the image by using the approach described previously (template matching) and its position in the 3D map is estimated as explained in section 4. In a general case, to be able to search for the instrument, a path planning and search algorithm must be used, especially if the object of interest does not lie in the field of view of the camera. However, here this is simplified by considering that the electrical instrument is already in the camera field of view and hence only detection and servoing are required. Once the object is detected in the image, the homography from the 3D world plane of the electrical instrument to the image plane is estimated. Based on the estimated homography, the 3D pose is estimated. We have now a 3D rigid transformation between coordinate systems of the instrument and the camera. Since the 3D pose of the quadcopter with respect to the visual SLAM coordinate system is known, the 3D pose of the planar object in that coordinate system can be calculated by cascading multiple rigid transforms between coordinate systems as shown in Figure 10.

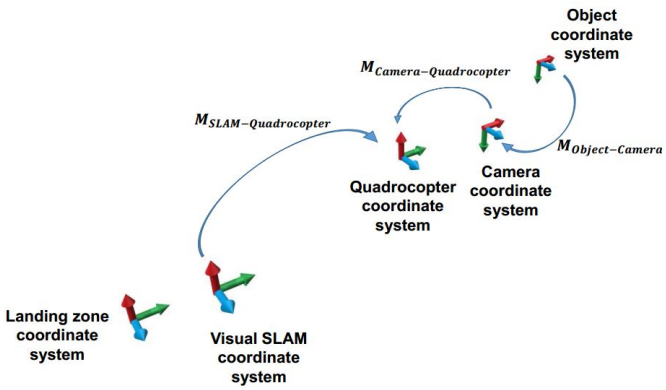


Fig. 10: Different coordinate systems and rigid transformations used in the application.

Thus, two localization sources are now available for visual feedback control. Based on the pose that is provided, it is possible to control the quadcopter through visual servoing in order to have a rigid link between the quadcopter and the instrument. To this end, one needs at any time the current localization information for the feedback control loops of the quadcopter. We emphasize the fact that the two sources of localization cannot be both available for all configurations and for all actual poses of the quadcopter. Indeed, the visual SLAM works well when there are enough key-points in the im-

age to detect and to put in correspondence with 3D map points. However, as the quadcopter approaches the planar instrument, most of the key-points will disappear and the visual SLAM algorithm might lose tracking. In the latter case, we use the 3D pose estimate given by the homography algorithm. In this way the quadcopter is able to fly and inspect the front panel of an electrical instrument.

6.2.3 Stage 3 - Object inspection

In stage 3, the drone is asked to move towards the object and center it in the image for a desired amount of time. The pose used here is the pose from the homography transform using the object of interest as a landmark.

6.2.4 Stage 4 - Returning to the base

After the mission is over, the drone will return to its starting position. It flies at a certain high altitude to get a broader view of the ground in order to localize the landing platform. The pose estimate used in this stage is the one from the SLAM system as the drone is moving away from the object.

6.2.5 Stage 5 - Localizing the landing platform and preparing for landing

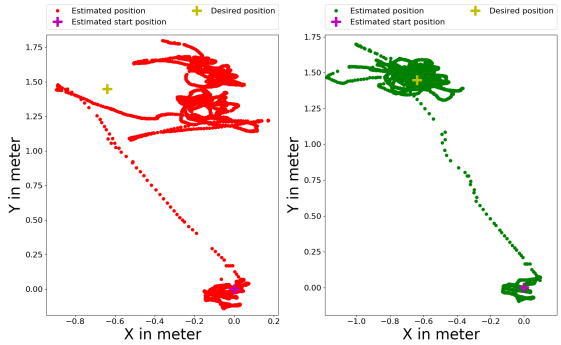
Similarly to the object detection and localization, the landing platform is detected and its position is localized in the 3D map using the approach described before in section 4 applied on the downward camera of the drone. The drone is asked to hover above the platform center at a certain altitude preparing for landing.

6.2.6 Stage 6 - Landing

A landing command is sent to the drone. The drone lands on the platform and the mission is over.

6.3 Comparison between SLAM and the proposed approach

In order to compare the proposed approach described here with the approach used in [18], we perform the following experiment. The same experiment as in the previous section is repeated until the stage 3 of object inspection. All data used for pose estimation are recorded including navigation data and the video frame from the drone as well as the estimated position. We perform an offline test of the SLAM approach by giving it the same data that were recorded on the actual flight (control commands, navigation data, video frames) and we record the estimation of the pose from this approach. Figure 11 shows a 2D plot of the pose estimate from both approaches, where the SLAM approach drifts and gives bad estimate when the drone is close to the object (*i.e.* at a distance of about 80cm). This can be also seen in Figure 12, the SLAM system loses tracking as soon as the drone approaches the object, and pose estimate is imprecise and begins to drift.



a) b)

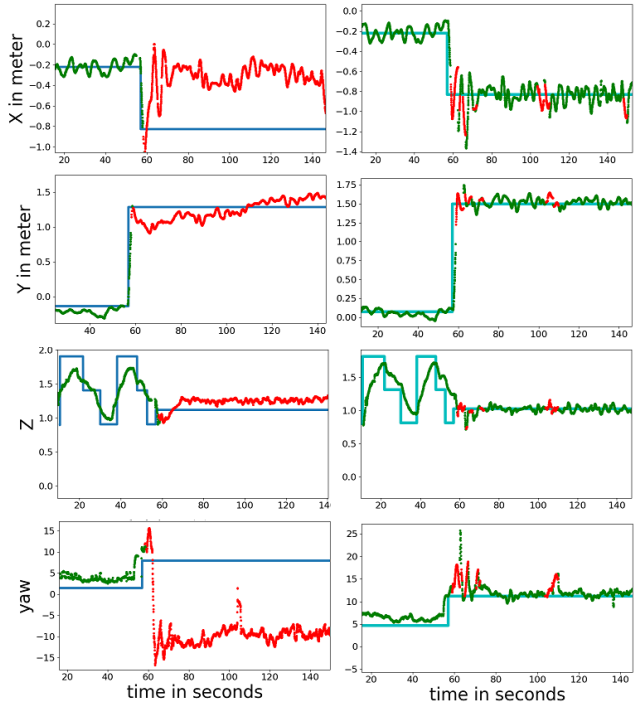
Fig. 11: Estimated pose in the XY plane: (a) with [18], (b) with the proposed approach. The yellow cross represents the desired position of the drone (at 80 cm from the object).

6.4 Face-Drone servoing

In this last scenario, we test the performance of the face-camera visual servoing system explained in section 5. The drone has to fly, detect a face, align its line of sight with that of the face, and center the face in the image while maintaining a fixed distance with it. In this experiment, the teacher is in motion in order to induce perturbation to the control system. The drone has to correct for the user displacement and the out-of-plane orientation of his face. Figure 13 shows the experiment seen from the camera of the drone and from an external camera. A video of the experiment is available in [39].

7 Conclusion

In this paper, we propose a system for using a low cost quadcopter in remote labs. Our contribution is a localization system that is able to deal with all situations encountered in this kind of missions. The localization system can handle indoor configurations when the drone is exploring the 3D environment, when it is examining an instrument and when it is going back to land on



a) b)

Fig. 12: Pose estimate using: (a) the PTAM approach only [18], and (b) the proposed approach. In red: pose estimates when the visual tracking fails (SLAM or homography); in green: when visual tracking is good. Cyan curves depict the reference pose.

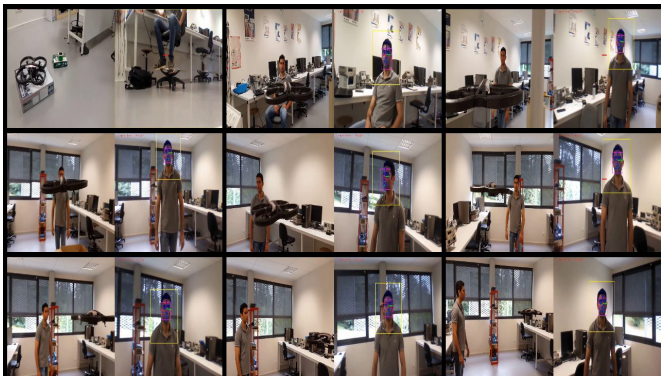


Fig. 13: Third scenario experiment: face tracking. 9 images corresponding to different time instants during the experiment. On the right, images taken from the on-board drone camera, on the left images recorded by a cellphone at the same time instant.

the automatic recharge platform. Furthermore, we allow the drone to search for a teacher in order to allow remote student-teacher communication (in case there is a teacher in the lab). This is done by proposing an approach for head pose estimation that uses 3D modelling of human face (Candide model) in addition to a state of the art facial landmark detection. Experiments were extensively done and prove the robustness of the approach and its efficiency despite the low-cost drone used. The combination of the two localization systems was found best suitable for this task due its simplicity and performance. The three videos available online illustrate the results [37], [38], [39]. The system works in real-time on a CPU (frame rate of 30 images per second).

References

1. Franck Luthon and Benoit Larroque. LaboREM a remote laboratory for game-like training in electronics. *IEEE Transactions on Learning Technologies*, 8(3):311–321, 2015
2. Franck Luthon, Benoit Larroque, Fawzi Khattar, and Fadi Dornaika. Use of gaming and computer vision to drive student motivation in remote learning lab activities. 2017
3. Rodrigo Munguía, Sarquis Urzua, Yolanda Bolea, and Antoni Grau. Vision-based SLAM system for unmanned aerial vehicles. *Sensors*, 16(3):372, 2016
4. Daniel Eberli, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Vision based position control for MAVs using one single circular landmark. *Journal of Intelligent & Robotic Systems*, 61(1-4):495–512, 2011
5. Artem Rozantsev, Sudipta N Sinha, Debadeepta Dey, and Pascal Fua. Flight dynamics-based recovery of a UAV trajectory using ground cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6030–6039, 2017
6. Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment - a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999
7. Koppány Máthé and Lucian Buşoniu. Vision and control for UAVs: A survey of general methods and of inexpensive platforms for infrastructure inspection. *Sensors*, 15(7):14887–14916, 2015
8. Konstantin Schauwecker and Andreas Zell. On-board dual-stereo-vision for the navigation of an autonomous MAV. *Journal of Intelligent & Robotic Systems*, 74(1-2):1–16, 2014
9. Gerardo Flores, Shuting Zhou, Rogelio Lozano, and Pedro Castillo. A vision and GPS-based real-time trajectory planning for a MAV in unknown and low-sunlight environments. *Journal of Intelligent & Robotic Systems*, 74(1-2):59–67, 2014
10. José A Castellanos, José Neira, and Juan D Tardós. Limits to the consistency of EKF-based SLAM. *IFAC Proceedings Volumes*, 37(8):716–721, 2004
11. Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598, 2002
12. Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, pages 225–234, 2007
13. Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443. Springer, 2006
14. Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual SLAM for RGB-D cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106, 2013
15. Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014
16. Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015
17. Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011
18. Jakob Engel, Jurgen Sturm, and Daniel Cremers. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems*, 62(11):1646 – 1656, 2014. Special Issue on Visual Control of Mobile Robots
19. Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014
20. Fawzi Khattar. *Enriching Remote Labs with Computer Vision and Drones*. PhD thesis, University of Pau & Pays Adour, Anglet, France, 2018
21. John Paulin Hansen, Alexandre Alapetite, I Scott MacKenzie, and Emilie Møllenbach. The use of gaze to control drones. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 27–34. ACM, 2014
22. Gerard Medioni and Sing Bing Kang. *Emerging topics in computer vision*. Prentice Hall PTR, 2004
23. Scott Krig. *Interest point detector and feature descriptor survey*. Springer, 2014
24. Sam Roweis. Levenberg-Marquardt optimization. *Notes, University of Toronto*, 1996
25. Georgios D Evangelidis and Emmanouil Z Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, 2008
26. Fadi Dornaika. Registering conventional images with low resolution panoramic images. *The 5th International Conference on Computer Vision Systems*, 2007
27. Fawzi Khattar, Fadi Dornaika, Benoit Larroque, and Franck Luthon. 3D object-camera and 3D face-camera pose estimation for quadcopter control: Application to remote labs. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 99–111. Springer, 2018
28. David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999
29. Jawad Nagi, Alessandro Giusti, Gianni A Di Caro, and Luca M Gambardella. Human control of UAVs using face pose estimates and hand gestures. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pages 252–253. ACM, 2014
30. Paul Viola and Michael J Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004
31. Mani Monajjemi, Sepehr Mohaimenianpour, and Richard Vaughan. UAV, come to me: End-to-end, multi-scale situated HRI with an uninstrumented human and a distant UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4410–4417, 2016
32. Fadi Dornaika and Franck Davoine. On appearance based face and facial action tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(9):1107–1124, 2006
33. Luis Unzueta, Waldir Pimenta, Jon Goenetxea, Luís Paulo Santos, and Fadi Dornaika. *Efficient deformable 3D face model fitting to monocular images*, chapter 8, pages 154–180. Advances in Face Image Analysis: Theory and Applications. Bentham Science Publishers, 2016

34. Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003
35. Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPNP: An accurate o(n) solution to the PNP problem. *International Journal of Computer Vision*, 81(2):155, 2009
36. Mikel Ariz, José J Bengoechea, Arantxa Villanueva, and Rafael Cabeza. A novel 2D/3D database with automatic face annotation for head tracking and pose estimation. *Computer Vision and Image Understanding*, 148:201–210, 2016
37. Perturbations. `https://youtu.be/Kr6TnjoByZ0`
38. Instrument inspection. `https://youtu.be/PTMVeJizjF8`
39. Face tracking. `https://youtu.be/XytlzOUdaDk`