An Assisted Bilateral Control Strategy for 3D Pose Estimation of Visual Features

Nicola Battilani, Riccardo Spica, Paolo Robuffo Giordano, Cristian Secchi

***Abstract*— Teleoperating a quadrotor equipped with a monocular camera for exploring a wide area in search of something has become a common practice in many application scenarios (e.g. search and rescue). In order to efficiently plan operations, estimating the 3D pose of a point of interest is as important as detecting it. In this paper we propose a novel bilateral teleoperation architecture where an estimation scheme is exploited for recovering the position of a set of visual features while an operator steers the motion of the quadrotor UAV. The operator acts on a force-feedback master device that produces force cues meant to suggest where to drive the quadrotor for improving the convergence rate of the estimation process. The effectiveness of the proposed teleoperation strategy is validated by means of hardware in the loop simulations.**

I. INTRODUCTION

Even the most modern computer softwares are still far from being able to fully reproduce the sophisticated cognitive capabilities of a human brain. As a result, in many applications involving complex decision making, it is still not possible, and sometimes not desirable, to rely on fully autonomous robots.

On the other hand, computers (and thus robots), can very effectively process enormous amounts of data and solve lower level tasks with an efficiency that is order of magnitudes higher compared to a human brain. In addition to this, robots can survive in very hostile environments, such as underwater, or in deep space, or in areas highly contaminated by chemical and nuclear pollutants.

These considerations have motivated a vaste literature exploring shared control architectures for teleoperation [1] which try to get the best out these two worlds by synergistically combining the capabilities of the robot with the high-level cognitive capabilities of the user.

The typical scenario considered in these works, represented in Fig. 1, entails the presence of (at least) one robot, equipped with onboard sensors, and communicating with a remote user. The user specifies some *high level commands* through an interface. If possible, the robot executes the commands while also taking into account additional motion and sensing *constraints* (e.g. avoiding obstacles, maintaining a desired formation with other robots, and so on). In addition to this, the robot also provides an *informative feedback* to the user about its performance. Haptic force cues, among others, have demonstrated to be particularly effective in this context [2].

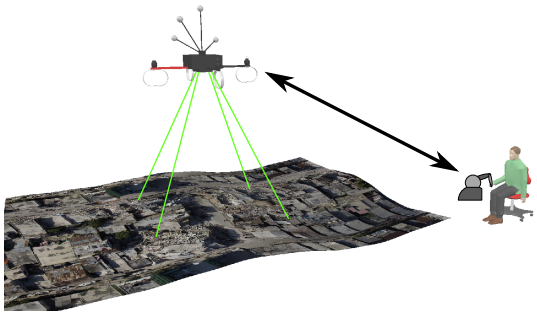

Fig. 1: An example of the considered scenario.

One application that has been subject to particular interest in this field is that of human guided navigation for search and rescue or exploration tasks [3]. A robotic platform that has proven to be particularly effective in this context is the quadrotor. Indeed, this platform can move in 3-D, and provides a good combination of agility and structural simplicity, coupled with the possibility of stationary flight and vertical takeoff and landing. However, a drawback of quadrotors (and, in general, of small-size flying vehicles) is their limited payload and energy autonomy that imposes very strict constraints on the amount of sensing and computational hardware that can be carried on board.

Because of these limitations, cameras are still among the most common sensors for quadrotor UAVs. While maintaining limited requirements in terms of both payload and energy consumption, cameras can, indeed, provide a large amount of information about the environment geometry and semantic content (see e.g. [4]). For the applications considered in this paper, monocular cameras are particularly convenient sensors. Indeed the distance between the robot and the environment is typically much larger than the size of the robot and, consequently, than the interocular distance of any stereo-camera that the robot could realistically carry. In these conditions, accuracy of stereo vision reconstruction can be compromised [5].

A very large number of works has considered the problem of reconstructing the geometry of an environment using monocular cameras. These algorithms can be broadly classified into visual odometry, visual SLAM and structure from motion approaches. For a recent literature review in this field we refer the reader to [6], [7].

Comparably, less attention has been devoted to designing control architectures that would ensure fast convergence of

N. Battilani and C. Secchi are with the University of Modena and Reggio Emilia, Reggio Emilia 42122, Italy {nicola.battilani,cristian.secchi}@unimore.it

R. Spica is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA rspica@stanford.edu

P. Robuffo Giordano is with CNRS at Irisa and Inria Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes Cedex, France prg@irisa.fr

the vision based reconstruction process. As well known, indeed, the accuracy of monocular vision based reconstruction is highly affected by the trajectory followed by the robot/camera during the estimation process (see e.g. [8] for an overview of the existing work).

Even less works consider how these active estimation strategies can be integrated in a shared control telemanipulation framework. Visual stream is often one of the feedback provided to the operator in a bilateral teleoperation system and vision has been exploited for building virtual fixtures (see e.g. [9] for a recent example) but in these cases the information collected by the sensor is exploited for building a constraint.

In exploration tasks (e.g. search and rescue) it is necessary to cover a wide area and to detect and reconstruct the position of point of interests (e.g. survivors). This can be efficiently done by teleoperating a quadrotor with a monocular camera providing a feedback to the operator. While from the analysis of the visual stream coming from the camera it is quite simple to detect a point of interest, estimating the position of a point of interest is much more involved. Since the monocular camera cannot directly provide the position of a detected feature, it is necessary to collect more images and to exploit the camera motion within some estimation scheme in order to build a reliable estimation of the position of the feature. The performance of any visual-based estimation scheme is, however, also affected by the camera motion itself, so that one needs to (actively) control the camera motion/trajectory in order to increase the convergence/accuracy of the employed estimator.

The goal of this paper is to build a bilateral control architecture that allows to teleoperate a quadrotor equipped with a simple monocular camera over a non flat environment where the 3D position of some features needs to be reconstructed. The active estimation strategy recently proposed in [8] is here exploited for estimating the position of the detected features and for generating a force feedback informing the operator about the best direction to steer the quadrotor for maximizing the estimation convergence speed. Furthermore, a computationally cheap methodology for keeping track of the reconstructed features and of their positions will be illustrated. Finally, some hardware in the loop simulations will be presented in order to validate the effectiveness of the proposed teleoperation architecture. A sketch of the system developed in this paper is reported in Fig. 1.

The main contribution of this paper is the development of a novel teleoperation architecture that allows to implement an assisted exploration and estimation task. The teleoperation system is formally proven to be safe for the user and characterized by a stable behavior.

The rest of the paper is organized as follows. In Sec. II some background on the estimation strategy proposed in [8] is provided. The problem to be addressed is stated in Sec. III and the bilateral control architecture with an active estimation force feedback is illustrated and analyzed in Sec. IV. A computationally cheap strategy for maintaining estimated features during the exploration is shown in Sec. V. In Sec. VI we report the results of the experimental validation the proposed architecture and in Sec. VII we draw some conclusions and address future work.

II. BACKGROUND

In this section we will provide some background on the active estimation technique that will be exploited for generating the assistive force in the bilateral teleoperation architecture. More details can be found in [8].

Consider a calibrated pinhole camera moving through space with a velocity $(v, \omega) \in \mathbb{R}^6$, where $v = (v_x, v_y, v_z) \in \mathbb{R}^3$ and $\omega = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ represent the camera linear and angular velocity in the camera frame, respectively. Let $P = (X, Y, Z) \in \mathbb{R}^3$ be a 3D point in the scene and let $s = (x, y, z) = \begin{pmatrix} X & Y & Z \end{pmatrix} / \|P\|$ be the bearing vector, i.e. the unit-norm vector pointing from the camera center to the 3D point. Setting $\chi = \frac{1}{\|P\|}$, it can be shown that the quantities s and χ obey the following dynamics:

$$\begin{cases} \dot{s} = f_m(s, v, \omega) + \Omega^T(s, v)\chi \\ \dot{\chi} = f_u(s, \chi, v, \omega) \end{cases} \tag{1}$$

where:

$$\begin{aligned} f_m(s, v, \omega) &= s \times \omega \\ \Omega(s, v) &= -v^T(I_3 - ss^T) \\ f_u(s, \chi, v, \omega) &= \chi^2 s^T v \end{aligned} \tag{2}$$

While s can be directly measured from the camera by segmenting the location of the observed point on the image plane, χ (the scale factor) cannot be directly observed. Using an Extended Kalman Filter (EKF) on (1), it is possible to build an asymptotic estimation of the position of the feature with respect to the camera frame ${}^C\hat{p} = (\hat{s}, \hat{\chi})$ and its corresponding covariance ${}^C\Sigma$ by exploiting the measured s and the (known) camera velocity $(v,\, \omega)$. From the estimation of the feature in spherical coordinates, it is trivial to build a cartesian estimation by computing $P = \frac{\hat{s}}{\hat{\chi}}$.

As discussed in [8], the convergence speed of the estimation depends on the quantity $\|\Omega\| = \Omega\Omega^T$ which is given by

$$\|\Omega\| = v^T v - (s^T v)^2. \tag{3}$$

Since $\|\Omega\|$ depends on the measurement s and on the linear velocity v, the convergence speed can be maximized by choosing a v such that, for the current s, (3) is maximized. As proposed in [8] this can be done instantaneously by implementing a gradient ascent technique that can be embedded, in a hierarchical fashion, with other velocity control techniques. In particular, in [8] the null space based approach [10] has been exploited and the following velocity control law has been proposed:

$$\dot{v} = \frac{v}{\|v\|^2} k_1 \left(\kappa_{des} - \kappa\right) + k_2 \left(I_3 - \frac{vv^T}{\|v\|}\right) J_v^T \tag{4}$$

where $k_1 > 0$ and $k_2 \geq 0$ are control gains, $\kappa_{des} = \frac{1}{2} v_0^T v_0$ and $\kappa = \frac{1}{2} v^T v$. J_v is the gradient of $\|\Omega\|$ with respect to the linear velocity, which has the following expression

$$J_v = 2v^T \left(I_3 - ss^T\right). \tag{5}$$

The first term in (4) has a higher priority and its role is to enforce the constraint $\|v\| = \|v_0\|$. The role of the second term, implemented in the null space of the first constraint, is to select the best direction for maximizing $\|\Omega\|$ by imposing to follow the gradient direction.

If one needs to estimate the 3D position of $N > 1$ features, the gradient ascent can be performed on the average gradient

$$J_v = \sum_{i=1}^{N} W_i J_{v,i} \tag{6}$$

where $J_{v,i}$ is the gradient corresponding to each feature and W_i are scalar weights that can be used for driving the camera towards some preferred directions. For example, by setting $W_i =^C \Sigma_i$, the camera would tend to move towards the directions that maximizes the convergence of the estimation of the more uncertain features.

III. PROBLEM STATEMENT

We consider a UAV locally controlled to maintain a certain altitude, far from obstacles (e.g. trees, rocks), flying over a non flat environment. The UAV is endowed with a monocular camera pointing downwards and we assume that the direction of the field of view is constant. This can be achieved by either limiting the velocity of the UAV or by mounting the camera on a controlled gimbal, or by reprojecting the image onto a virtual plane [11].

A set of features of interest is disseminated in the environment in unknown locations and a user has to teleoperate the UAV in order to find them and identify their position using the data coming from the camera. A force feedback for assisting the user in speeding up the estimation process has to be designed.

The overall teleoperation system needs to be safe for the user and guarantee stable dynamics. Furthermore, since the whole environment cannot be framed by a single image, a strategy for keeping track of the identified features needs to be designed.

In the rest of the paper, we consider that the communication delay between the master and the slave is negligible (e.g. the user is in the vicinity of the UAV). In case of bigger delays the proposed master side dynamics can be passified exploiting a tank based approach as, e.g., in [12], [13].

IV. SHARED TELEOPERATION ARCHITECTURE

The master is a fully actuated system that can be described, in the workspace, by the following Euler-Lagrangian model

$$M(x_m)\ddot{x}_m + C(x_m, \dot{x}_m)\dot{x}_m + D\dot{x}_m = F_m + F_h \tag{7}$$

where $x_m \in \mathbb{R}^6$ and $\dot{x}_m \in \mathbb{R}^6$ are the pose and the twist of the end-effector respectively. $M(x_m) = M^T(x_m) > 0$ is the inertia matrix, $C(x_m, \dot{x}_m)$ is the term encoding the centrifugal and Coriolis forces and $D = D^T > 0$ is the matrix representing the viscous friction in the robot. $F_h, F_m \in \mathbb{R}^6$ are the wrenches due to the interaction with the human and with the slave side respectively. We assume that gravity is locally compensated. As well known [14], such a mechanical system is passive with respect to the pair $(F_h + F_m, \dot{x}_m)$.

The slave robot is the UAV. We assume that the slave is controlled to maintain a fixed altitude and to behave as a kinematic system in the other directions (using e.g. [15]). Thus, the controlled UAV can be modeled as:

$$\dot{x}_s = v_s \tag{8}$$

where $x_s \in \mathbb{R}^2$ is the cartesian position of the quadrotor on the horizontal plane in which the robot is allowed to move and $v_s \in \mathbb{R}^2$ is the corresponding linear velocity. It is straightforward to show that the slave is a passive system with respect to the pair (v_s, x_s) using the positive storage function $H_s = \frac{1}{2} x_s^T x_s$. In fact:

$$\dot{H}_s = \frac{\partial^T H_s}{\partial x_s}\dot{x}_s = x_s^T v_s \tag{9}$$

In order to deal with the difference between the workspace of the master, which is finite, and the workspace of the slave, which is infinite, we exploit the position of the master for driving the velocity of the slave. Unfortunately, (7) is not passive with respect to the pair $(F_h + F_m, x_m)$ [14]. In order to preserve the passivity of the master, we pre-compensate the master using the technique proposed in [16] in order to make the master passive with respect to the pair $(F_h + F_m, r_m)$ using the storage function $H_m = \frac{1}{2}\rho r_m^T M(x_m) r_m$, with

$$r_m = \rho \dot{x}_m + \lambda \rho x_m \tag{10}$$

where $\rho, \lambda > 0$ are design parameters. Thus, by properly choosing λ and ρ we can achieve $r_m \approx K x_m$, where $K > 0$ is the desired gain.

Since the UAV can move on a fixed plane, only two DOFs of the master are exploited to drive the slave. Thus, in order to allow the user to control the velocity of the UAV by the (scaled) position of the master we set:

$$v_s = Q r_m \tag{11}$$

where $Q \in \mathbb{R}^{2\times 6}$ is defined as $Q = \begin{pmatrix} I_2 & 0 \end{pmatrix}$ where I_2 is the identity matrix of order 2 and 0 is a null matrix of proper dimensions. Q has the role to associate only two DOFs to the motion of the UAV[1].

On the slave robot, the feature estimation algorithm illustrated in Sec. II is run. In order to suggest the user where to to move the robot so as to speed up the feature estimation process, we design a force feedback inspired by (4).

The main idea is that the velocity commanded to the slave has the higher priority and that the force feedback suggests a correction in the velocity direction to be sent to the UAV. More formally, given a set of N features detected by the onboard camera, we propose to implement the following assistive force feedback on the master:

$$F_a = k\left(I_2 - \frac{v_s v_s^T}{v_s^T v_s}\right)\bar{J}_v^T = k\left(I_2 - \frac{Q r_m r_m^T Q^T}{r_m^T Q^T Q r_m}\right)\bar{J}_v^T \tag{12}$$

[1]In the definition of Q given in this paper we consider to associate the first two DOFs of the master to the slave. All the results of the paper keep on holding if we associate any two other DOFs to the slave

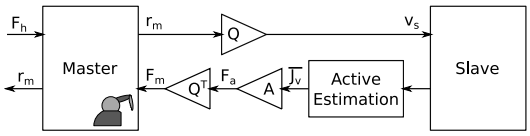

Fig. 2: The teleoperation architecture. The active estimation block computes the direction to track $\bar{J}_v$ and $A = k\left(I_2 - \frac{v_s v_s^T}{v_s^T v_s}\right)$ projects it in the null space of the velocity of the slave.

where the Jacobian corresponding to each feature is obtained using the strategy proposed in Sec. II and J_v is obtained as in (6). $\bar{J}_v = PJ_v$, where P is the matrix projecting J_v on the plane the UAV is constrained to move. This allows to obtain a force directed in the best direction for maximizing the estimation convergence on the plane the UAV is moving. The projection is useful for providing the user with a force that can inform her/him about a direction the UAV can be actually moved towards. Finally, the gain $k > 0$ can be used to scale the assistive force.

By construction $F_a^T Q r_m = 0$, meaning that the assistive force is orthogonal to the velocity commanded to the slave. Thus, the assistive force tends to deviate the master towards a position corresponding to the best velocity the slave should track for optimizing the estimator convergence speed.

The proposed teleoperation system is represented in Fig. 2.

Thus, we can close the loop by transmitting to the user the assistive force. In other words, we can join master and slave through the following interconnection:

$$\begin{cases} F_m(t) = Q^T F_a(t) \\ \\ v_s(t) = Q r_m(t) \end{cases} \tag{13}$$

In general, (13) is not power preserving, i.e. $F_m(t)^T r_m \neq v_s^T x_s$ and, therefore, we cannot state that the overall teleoperation system is passive. Nevertheless, it is possible to prove that the teleoperation system is safe for the user and that no energy is produced at the master side because of the coupling in (13)

Proposition 1: The system the user is interacting with is passive with respect to the pair (r_m, F_h).

Proof: The user interacts with the master. Thanks to the precompensation proposed in [16] we have that

$$(F_h + F_m)^T r_m = F_h^T r_m + F_m^T r_m \geq \dot{H}_m \tag{14}$$

From (13) we have that $F_m = F_a$ and since $F_a^T r_m = 0$ it follows that:

$$F_h^T r_m \geq \dot{H}_m \tag{15}$$

which proves that the system the user is interacting with is passive with respect to (r_m, F_h). ■

In passivity based bilateral teleoperation systems it is necessary to prove that the overall teleoperation system is passive in order to guarantee a passive and, therefore, safe behavior of the system. This is because, usually, the coupling between master and slave is passive and makes the overall teleoperation system a unique, virtual, mechanical system. In our case, the coupling (13) is not passive but the design of F_a energetically decouples master and slave and, therefore, the coupling between the robots does not produce energy, potentially harmful for the user on the master. Thus, because of this energetic decoupling, the user enjoys an informative force feedback while interacting with a passive and safe dynamical system.

The next result proves that the teleoperation system behaves well even if the user does not interact with the master. This means that there is no internal destabilizing effects on the teleoperation system. Let $x = (Q^T r_m^T, x_s^T)^T \in \mathbb{R}^4$ be the state of the overall teleoperation system and let $\Lambda = \{x \in \mathbb{R}^4 \text{ s.t. } x = (0^T x_s^T)\}$.

Proposition 2: If $F_h = 0$ the x asymptotically converges to Λ

Proof: The precompensated master is detectable and, therefore, because of the passivity proven in (15), we have that the configuration $r_m = 0$ is asymptotically stable. As shown in (13), the velocity input of the slave is r_m which asymptotically vanishes. Thus, asymptotically, $r_m \to 0$ and consequently $x_s \to x^*$, which implies that $x \to \Lambda$. ■

This simply means that, as expected, when the user leaves the master she/he is commanding, then the master moves to the zero position and, consequently, the UAV stops in an equilibrium position. Because of the energetic decoupling due to F_a, the force feedback does not introduce sustained unwanted dynamics.

V. STORING AND MAINTAINING THE FEATURES

Only the features in the camera field of view are estimated and used for building the force feedback in (12). Of course during the exploration of an area, it is necessary to store the identified features and to recover them when they enter in the field of view again.

Using the strategy proposed in Sec. II and Sec. IV, the 3D position of each feature is precisely estimated with respect to the camera frame. In order to keep track of the estimated features that, because of the motion of the UAV, move out of the field of view it is necessary to express their estimated position with respect to the world frame. To this aim, we assume the quadrotor is endowed with a GPS sensor. Notice that the presence of the GPS, due to the inaccuracy of the sensor, would not be sufficient for identifying the 3D position of each feature which is instead possible thanks to the strategy proposed in this paper.

This section presents a computationally cheap algorithm, suitable for implementation on the low cost hardware, for storing the 3D features to produce the force feedback.

A. Main Algorithm

At each time step, Alg. 1 is executed for storing the estimated positions of the features and for producing the force feedback to be sent to the master side.

The algorithm requires the current list L where the 3D positions of the estimated features are stored, the velocity v of the UAV, the position of the camera with respect to a world reference frame C_{GPS} obtained by the GPS and the

Algorithm 1: Overall algorithm.

Data: Require:L, v, C_{GPS}, r_m
Data: Provide: L,F_a
1 $img \leftarrow$ getImage()
2 $S \leftarrow$ detect(img)
3 $L \leftarrow$ updateList(S, img, C_{GPS},L)
4 $F_a \leftarrow$computeForce(S, v, r_m)

variable r_m. The list L is stored in the computational unit of the UAV, v and C_{GPS} are collected by the sensors on the UAV and, finally, r_m is received by the master. After the execution, an updated list L of the estimated features and the force F_a are computed and the latter is communicated to the master side.

At each step the camera acquires an image of the environment the UAV is exploring (Line 1) and then, using a feature detector, a list S of the bearings of the detected feature is built (Line 2). This information is exploited for updating the estimation of the features using the observer described in Sec. II (Line 3) and for updating the list of features L. Finally, the force F_a for suggesting to the user the best direction to take in order to maximize the convergence speed of the estimator is computed as indicated in (12) (Line 4).

In the following, we will detail a computationally efficient strategy for building and updating the list L of estimated features.

The strategy illustrated in Sec. II allows to build an estimate of each feature in the camera frame. Nevertheless, in order to keep track of all the estimated features even when they move out from the camera field of view, it is necessary to refer them to the world frame even if the price to pay is an increase of uncertainty due to the GPS. During the exploration it can happen that for estimating some features, some other features exits from the field of view while their position is only partially estimated. Thus, when a feature enters in the field of view again, it is necessary reinitialize its estimation. This can be done by exploiting the current GPS information for mapping back the estimation with respect to the world frame stored in L to the current camera frame. Nevertheless, this would lead to an estimation with a very high covariance and, therefore, to a bad reinitialization. A better reinitialization can be done by using vision, instead of GPS, for computing the relative motion of the camera. Since the camera is moving on a plane, only translations are possible. If one feature is visible in two different images and its 3D position with respect to the camera frame is available, it is possible to build the relative translation of the camera between the two images without resorting to any global information. Since only an estimate of the 3D positions of the features is available, we can build a good estimate of the relative translation by averaging the information we can get from more features present in both images. Suppose to keep a list of neighbors, i.e. a list of features belonging to the same image, for each feature. If, when a partially estimated feature re-enters in the field of view of the camera, at least a certain number N of its neighbors are in the current image, then it is possible to build an estimate of the relative translation of the camera and to map the estimate of the feature into the current camera frame and use it to re-initialize the estimator. N is a design parameter. The larger is N, the better is the estimation of the relative translation we can get. Nevertheless, if N is too large, it may happen that the reinitialization using neighbors is never activated.

For the generic i^{th} feature, the estimation strategy illustrated in Sec. II allows to build an estimate ${}^C\hat{p}_i = (\hat{s}_i, \hat{\chi}_i)$ of the 3D position and its corresponding covariance matrix ${}^C\Sigma_i$. Exploiting the GPS information it is possible to easily map the estimate in the camera frame into an estimate $({}^Wp_i, {}^W\Sigma_i)$ in the world frame.

For each feature, the following information is stored in L

- the estimated position with respect to the camera frame, ${}^C\hat{p}_i$, and its corresponding covariance, ${}^C\Sigma_i$, that can be built using the strategy illustrated in Sec. II.
- the estimated position with respect to the world frame, ${}^W\hat{p}_i$, and its corresponding covariance, ${}^W\Sigma_i$, that can be easily obtained exploiting the GPS information
- A list $\mathcal{N}_i$ of neighbors, i.e. features that appear in the same image
- A status

The status of a feature can be: tracked, if the feature was visible in the previous image, lost, if the feature was not visible in the previous image. The list update procedure is outlined in Alg. 2.

B. List Update Algorithm

The algorithm requires the current image img, the list of stored features L, the set of bearing measurements S and the GPS measurement and it provides and updated list of features L. As a first step all the elements in L are scanned and if a feature is not in the current image its status is set to **lost** (Line 3). Then, each feature in the image is considered for updating the list L. If the feature is not in L, it means that it is a new feature and therefore it is added to the list (Line 8). The estimate of the feature with respect to the camera frame is initialized by taking the corresponding bearing from S, by setting some initial value for χ and for the covariance matrix (Line 9). The function $wnsUpdate()$ sets the rest of the information related to the feature (Line 10). In particular, the estimate of the feature with respect to the world frame is obtained by the data :

$$f.world = ({}^WT_C^C\hat{p}, {}^C\Sigma_{c_{\hat{p}}} + N_{GPS})$$

where ${}^WT_C$ is the position of the camera with respect to the world frame, obtained by the GPS, and N_{GPS} is the covariance of the GPS sensor. Furthermore, the function sets:

$$f.\mathcal{N} = update(f, img)$$

$$f.status = \mathbf{tracked}$$

where the update function populates $\mathcal{N}$ with all the other features in the image. Finally, the status of the feature is

Algorithm 2: UpdateList

Data: Require: img, L, S,C_{GPS},
Data: Provide: L
1 **for** $f \in L$ **do**
2 **if** $f \notin img$ **then**
3 $f.status =$ **lost**
4 **end**
5 **end**
6 **for** $f \in img$ **do**
7 **if** $f \notin L$ **then**
8 add(f,L)
9 $f.camera = (s, \chi_0, {}^C\Sigma_0)$
10 wnsUpdate()
11 **else**
12 **if** $f.status ==$ **tracked then**
13 $f.camera = updateEstimate()$
14 wnsUpdate()
15 **else**
16 **if** $f.\mathcal{N} \cap img \geq N$ **then**
17 $f.camera =$ $updateNeighbors(\mathcal{N}, img)$
18 wnsUpdate()
19 **else**
20 $f.camera =$ $updateGPS(f.camera, C_{GPS})$
21 wnsUpdate()
22 **end**
23 **end**
24 **end**
25 **end**

set to **tracked**. If the feature is in a **tracked** status its estimation is updated using the estimator illustrated in Sec. II (Line 13) while the rest of the information is updated using the function $wnsUpdate()$ (Line 14). If the feature is in a **lost** state, two cases need to be distinguished. If there are at least N common features between the current image and the neighbors of the lost feature, then the relative translation between the position of the camera corresponding to the estimate of the feature and the current one is evaluated as:

$$
\begin{aligned}
{}^{new}T_{old} &= \frac{1}{n}\left(\sum_{i=1}^{n}\left(\frac{\hat{s}_i}{\hat{\chi}_i} - \frac{\hat{s}_i^{\mathcal{N}}}{\hat{\chi}_i^{\mathcal{N}}}\right)\right. \\
{}^{new}\Sigma_{old} &= \frac{1}{n^2}\sum_{i=1}^{n}\Delta\Sigma_i
\end{aligned}
\tag{16}
$$

where $n = \#(\mathcal{N} \cap img)$ is the cardinality of the set of common features. For ease of notation, we indicated with the superscript $\mathcal{N}$ the corresponding feature in the neighbors list. $\Delta\Sigma_i = {}^C\Sigma_i + {}^C\Sigma_i^{\mathcal{N}}$ is the sum of the covariances of the current feature estimation and of the estimation stored in the neighbors list. Once the relative translation is available, the old estimation of the feature can be mapped to the new frame and the price to pay is the covariance ${}^{new}\Sigma_{old}$ which is much lower than the covariance of the GPS. If there are not enough common features, then the (partial) estimation of the feature needs to be mapped back in the camera frame exploiting the GPS information (Line 20).

VI. EXPERIMENTS

In this section we evaluate the effectiveness of the proposed teleoperation architecture by means of experiments and simulations.

A pre-compensated quadrotor characterized by the dynamics (8) and moving on a plane at height $z = 32\ m$ has been simulated in the V-REP simulation environment[2]. The robot is endowed with a GPS (affected by a zero mean gaussian noise with covariance matrix $N_{GPS} = 7.5I_3$). The velocity measurement is also affected by a zero mean Gaussian noise with covariance matrix $N_v = 0.5I_3$. Since the estimate of the bearing is the measurement itself, in the following we will show the results related to the estimation of χ. The master is a phantom omni by Geomagic[3] and it has been pre-compensated as indicated in Sec. IV with $\rho = \frac{1}{\lambda} = 0.001$.

First of all, in order to show the benefits of moving in the direction of (12) when the camera is constrained to a plane. To this aim, we consider a scenario with a single feature that always stays in the field of view of the robot. We set in a non zero position the master and do not apply any force to the master ($F_h = 0$). In this way the quadrotor starts moving. The force feedback moves the master, and consequently drives the quadrotor, along the direction of F_a. We have implemented two different cases. First we set $k = 1$ in (12), i.e. the quadrotor is moved along the direction that maximizes the estimation convergence speed, and the we set $k = -1$, i.e. the quadrotor is moved in the opposite direction. In Fig. 3 the estimation error on χ is plotted. It can be clearly seen that moving along the suggested direction can bring a great advantage in terms of velocity of estimation with respect to other directions along which an operator may drive the UAV.

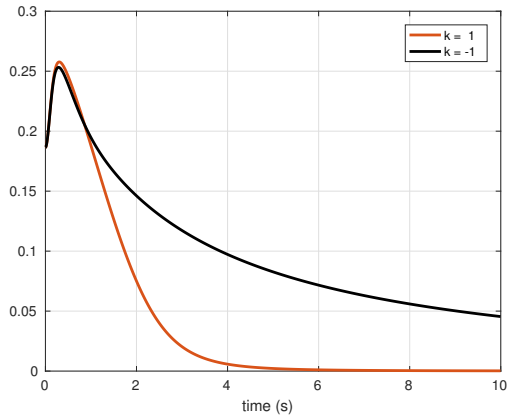

Fig. 3: The evolution of the estimation error of χ with $k = 1$ and $k = -1$.

In the second experiment an operator is asked explore an environment with 25 features randomly placed at an height

[2]http://www.coppeliarobotics.com
[3]http://www.geomagic.com/

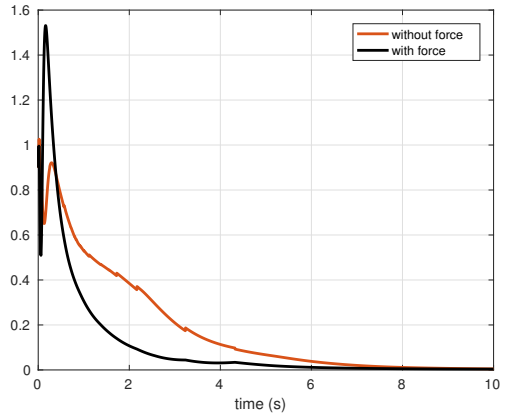

Fig. 4: Evolution of χ_{tot} with and without force feedback.

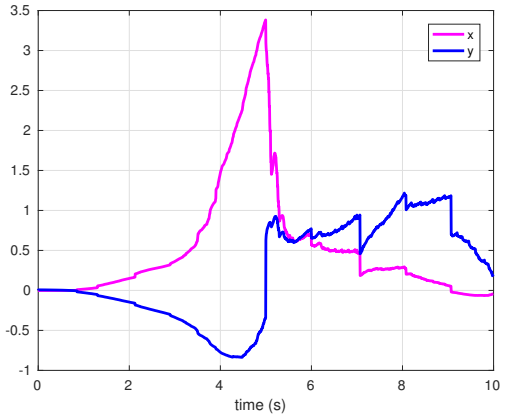

Fig. 5: Force fed back to the operator.

between 0 m. and 15.5 m. Also in this case, we assume that all the features always stay in the field of view of the camera. For the computation of (12) we have computed $\bar{J}_v$ considering the covariance of the estimation of the features as weights in (6).

In Fig. 4 the sum of the squares of the estimation errors $\chi_{tot}(t) = \sum_{i=i}^{25}(\chi_i(t) - \hat{\chi}_i(t))^2$ with and without force feedback is plotted. The initial peak in the case of the force feedback is due to the reaction of the user to the initially applied force. What is more important is that the force feedback helps the user to speed up the estimation process. The force fed back to the user is shown in Fig. 5. Of course the advantage is smaller with respect to the one in Fig. 3 since, during teleoperation, the user does not always move exactly against the direction suggested by (12).

In the attached video, a teleoperation experiments where the user has to explore an environment with 225 features is shown. Only a fraction of the feature can stay in the field of view of the camera. The estimation of the feature and their re-initialization is managed by the strategy proposed in Sec. V.

VII. CONCLUSIONS AND FUTURE WORKS

In this work we have proposed a novel bilateral control architecture that allows to teleoperate a quadrotor with a monocular camera and to estimate the pose of a set of features using the approach proposed in [8]. A force feedback informing the user about the best direction to drive the quadrotor along for maximizing the estimation convergence speed has been implemented. Hardware in the loop simulations have shown that the proposed teleoperation system allows to speed up the feature pose estimation significantly.

Future work aims at validating the proposed strategy with a real quadrotor. Furthermore, we will consider the teleoperation of a multi-quadrotor system for the exploration and estimation of the features.

REFERENCES

[1] C. Niemeyer, G.and Preusche, S. Stramigioli, and D. Lee, "Telerobotics," in *Springer Handbook of Robotics*, 2nd ed., B. Siciliano and O. Khatib, Eds. Springer, 2016, ch. 43, pp. 1085–1105.

[2] A. Franchi, C. Secchi, M. Ryll, H. H. Bulthoff, and P. R. Giordano, "Shared control : Balancing autonomy and human assistance with a group of quadrotor uavs," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 57–68, Sept 2012.

[3] Y. Liu and G. Nejat, "Robotic urban search and rescue: a suervey from the control perspective," *Journal of Intelligent and Robotic Systems*, vol. 72, pp. 147–165, 2013.

[4] B. Liu, X. He, and S. Gould, "Joint semantic and geometric segmentation of videos with a stage model," in *IEEE Winter Conference on Applications of Computer Vision*, March 2014, pp. 737–744.

[5] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer Science & Business Media, 2012, vol. 26.

[6] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

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

[8] R. Spica, P. R. Giordano, and F. Chaumette, "Active structure from motion: Application to point, sphere and cylinder," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1499–1513, 2014.

[9] M. Selvaggio, G. Notomista, F. Chen, B. Gao, F. Trapani, and D. Caldwell, "Enhancing bilateral teleoperation using camera-based online virtual fixtures generation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 1483–1488.

[10] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The null-space-based behavioral control for autonomous robotic systems," *Intelligent Service Robotics*, vol. 1, no. 1, pp. 27–39, 2008.

[11] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2113–2118.

[12] C. Secchi, A. Franchi, H. H. Blthoff, and P. R. Giordano, "Bilateral teleoperation of a group of uavs with communication delays and switching topology," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 4307–4314.

[13] F. Ferraguti, N. Preda, A. Manurung, M. Bonfe, O. Lambercy, R. Gassert, R. Muradore, P. Fiorini, and C. Secchi, "An energy tank-based interactive control architecture for autonomous and teleoperated robotic surgery," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1073–1088, 2015.

[14] C. Secchi, S. Stramigioli, and C. Fantuzzi, *Control of Interactive Robotic Interfaces: a port-Hamiltonian Approach*, ser. Springer Tracts in Advanced Robotics. Springer, 2007.

[15] D. Lee, A. Franchi, H. I. Son, C. Ha, H. H. Blthoff, and P. R. Giordano, "Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 4, pp. 1334–1345, Aug 2013.

[16] A. Franchi, C. Secchi, H. Son, H. Bulthoff, and P. Robuffo Giordano, "Bilateral teleoperation of groups of mobile robots with time-varying topology," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1019–1033, Oct 2012.