POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Kinect-based natural interface for quadrotor control

Original
A Kinect-based natural interface for quadrotor control / Sanna, Andrea; Lamberti, Fabrizio; Paravati, Gianluca; Manuri, Federico. - In: ENTERTAINMENT COMPUTING. - ISSN 1875-9521. - STAMPA. - 4:3(2013), pp. 179-186. [10.1016/j.entcom.2013.01.001]

Availability:
This version is available at: 11583/2505606 since: 2020-07-13T15:32:19Z

Publisher:
Elsevier

Published
DOI:10.1016/j.entcom.2013.01.001

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Elsevier postprint/Author's Accepted Manuscript

© 2013. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/.The final authenticated version is available online at: http://dx.doi.org/10.1016/j.entcom.2013.01.001

(Article begins on next page)

12 October 2024

Manuscript source file
Click here to view linked References

A Kinect-based Natural Interface for Quadrotor Control

Abstract

This paper presents a new and challenging approach to the control of mobile platforms. Natural User Interfaces (NUIs) and visual computing techniques are used to control the navigation of a quadrotor in GPS-denied indoor environments. A visual odometry algorithm allows the platform to autonomously navigate the environment, whereas the user can control complex manoeuvres by gestures and body postures. This approach makes the Human-Computer Interaction (HCI) more intuitive, usable, and receptive to the user's needs: in other words, more user-friendly and, why not, fun. The NUI presented in this paper is based on the Microsoft Kinect and users can customize the association among gestures/postures and platform commands, thus choosing the more intuitive and effective interface.

Keywords:

natural user interface, Kinect, quadrotor control, interactive systems, visual odometry.

1. Introduction

The control of mobile platforms plays a key role in several application fields ranging from surveillance to entertainment. A framework to control a quadrotor is presented in this paper; the proposed solution supports both autonomous flight and manual control by user's body postures. The

autonomous flight system has been designed for GPS-denied indoor environments, whereas the human-computer interaction (HCI) has been based on gestures/postures, thus implementing a so called Natural User Interface (NUI). NUIs have been investigated since early eighty's (voice and gestures are used to control a GUI in [5]). Among NUIs, gesture-based interfaces always played a crucial role in human-machine communication, as they constitute a direct expression of mental concepts [25]. For example, nowadays mobile platforms can be remotely piloted by using multi-touch devices [22] [23] that also act as display devices through the use of interactive streaming functionalities [18]. The analysis of images coming from on-board cameras allows mobile platforms to perform target tracking and following tasks [19] [20] [21]. The variety of hand and body gestures, compared with traditional interaction paradigms, can offer unique opportunities also for new and attractive forms of HCI [24]. Thus, new gesture-based solutions have been progressively introduced in various interaction scenarios (encompassing, for instance, navigation of virtual worlds, browsing of multimedia contents, management of immersive applications, etc. [28][39]) and the design of gesture-based systems will play an important role in the future trends of the HCI.

Human-Robot Interaction (HRI) is a subset of HCI and can be considered as one of the most important domains of the Computer Vision. Although a lot of works based on gesture recognition in the domain of HRI are known in the literature (Section 2 briefly reviews the most appropriate ones) recent technological advances have opened new and challenging research horizons. In particular, controllers and sensors used for home entertainment can be exploited also as affordable devices supporting the design and implementation

of new kinds of HRI.

The aim of this work is to create a human-robot interaction framework to allow a quadrotor both to perform autonomous navigation tasks (by completing path-following missions constituted by a sequence of pre-specified way-points) and to be controlled by user's body postures (for instance, when complex actions/movements have to be performed). The main requisites needed to implement a system capable of controlling the aerial vehicle by means of user's posture are: 1. extracting spatial information from specific parts of the body 2. recognizing postures from this information 3. associating recognized postures to specific commands to be sent the quadrotor. In this work, the Microsoft Kinect [15] is used as gesture tracking device; recognized postures are then used to control an Ar.Drone quadrotor platform [2] (in the following of the paper the terms: Ar.Drone, quadrotor, and platform will be used interchangeably). The user is the "controller", and a new form of HRI can therefore be experienced. Interaction with quadrotors via Microsoft Kinect is not new. In particular, the ETH Zurich group proposes a way to dynamically interact with quadrotors based on the position of arms [10]. The local 3D coordinates of the user's arms are mapped to the local 3D coordinates of the quadrotor; in this way, a direct mapping between arms coordinates and quadrotor's position can be established. However, the approach adopted in this paper is different from the one adopted in [10]. Indeed, in this paper gesture recognition is used to trigger discrete control commands without using an external tracker system to obtain the quadrotor position. The two methods can be considered as complementary, since the approach presented in this paper is useful for the navigation over

a path which length is not known a priori, whereas the ETH Zurich group approach [10] is useful for performing local navigation tasks where a higher degree of precision is needed. Tests proved that the platform can be easily controlled by a customizable set of body movements, allowing for an exciting, fun, and safe experience even for non-skilled users. In order to allow the platform to autonomously fly indoor environments, a pose estimation system exploiting two different techniques is able to process images received from an on-board camera to support the navigation. In this work, the on-board camera is looking downward. Position and flight altitude are continuously measured by a feature-based pose estimation algorithm that analyzes the image features of the camera view, thus allowing the system to estimate the location and the orientation of the platform in the environment. Moreover, a second technique (namely tag-based pose estimation) exploits some visual markers (tags) placed on the floor, at well known positions, in order to reset localization drifts cumulated by the feature-based pose estimation system [7]. The combination of these two techniques results into an efficient and robust visual odometry algorithm. The overall framework introduces a number of challenges to be addressed, from the control of the quadrotor through body postures to the analysis of the images coming from the on-board camera to infer the flight attributes of the quadrotor in a GPS-denied environment. The main contribution of this paper is the proposal of a new integrated framework able to efficiently and intuitively support both autonomous and piloted flight in indoor environments.

The paper is organized as follows: Section 2 reviews the main HRI solutions and briefly introduces the Ar.Drone. Section 3 describes the NUI and

the mapping between gestures and commands. Section 4 presents the quadrotor pose estimation algorithm and its performance. Finally, conclusions are drawn in Section 5.

2. Background

This Section is split in two parts: the first part presents the state-of-the-art of NUIs with a particular focus on HRI, whereas the second part describes the quadrotor used for tests.

2.1. Natural user interfaces

In HRI-based systems, especially in safe critical applications such as the search-and-rescue and military ones, it is increasingly necessary for humans to be able to communicate and control robots in a natural and efficient way. In the past, robots were controlled by human operators using hand-controllers such as sensor gloves and electromechanical devices [32]. With these devices, the speed and simplicity of the interaction were significantly constrained. To overcome the limitations of such electro-mechanical devices, gesture and body posture recognition techniques have been introduced. In particular, body postures can be recognized by using sensors which need to be worn as well as vision based techniques.

For example, the approach of controlling mobile platforms by body postures (e.g. trunk tilt) is presented in [31] and [34]. A belt interface, encompassing a set of sensors to recognize user bendings, allows the user to control the robot motion and to receive tactile, visual and auditory feedback from the remote mobile robot.

On the other hand, vision based techniques [25] do not require to wear any contact devices, but use a set of sensors and algorithms for recognizing gestures. Therefore, the type of communication based on gestures can provide an expressive, natural and intuitive way for humans to control robotic systems. One benefit of such systems is that they propose natural ways to send geometrical information to the robot, such as: up, down, etc. As seen in [4], through the recognition of gestures, a natural language for HRI can be created, relying on non invasive systems such as a camera to identify user gestures for comparison with a predefined gesture database. Gestures may represent a single command, a sequence of commands, a single word, or a phrase and may be static or dynamic. Such a system should be accurate enough to provide the correct classification of gestures in a reasonable time.

The ability to recognize gestures is important for an interface developed to understand user's intentions. Interfaces for robot control that use gesture recognition techniques have been studied in depth, as using gestures represents a formidable challenge. In fact, several issues arise from environments with complex backgrounds, from dynamic lighting conditions, from shapes to be recognized (in general, hands and the other parts of the human body can be considered as deformable objects), from real-time execution constraints, and so on.

A lot of work has been focused on hand gesture recognition for human robot interaction. For instance, a gesture-based architecture for hand control of mobile robots was proposed in [30]. Gestures were captured by a data glove and gesture recognition was performed by Hidden Markov Model statistical classifiers. Then, the interpreted gestures were translated into commands to

control the robot. The use of a data glove was then replaced by markers in [12]. Two cameras provided the information to triangulate the position of the hand markers, allowing gesture recognition to take place and control a 6-DOF (Six Degrees Of Freedom) robot with a high precision. An alternative identification of the hand posture was proposed in [8]. The hand posture was identified from the segmented temporal sequence obtained by the Hausdorff distance method. A real-time vision-based gesture recognition system for robot control was implemented in [4]. Gestures were recognized using a rule-based approach by comparing the skin like regions in a particular image frame with the predefined templates in the system memory. Another hand gesture recognition system for robot control, which uses Fuzzy-C-Means algorithm as gesture classifier to recognize static gestures, was proposed in [35] and [36]. Static and dynamic gestures were recognized by a Fuzzy-C-Means clustering algorithm in [26].

2.2. Quadrotors and the Ar.Drone platform

Quadrotors are used in a large spectrum of applications ranging from surveillance to environmental mapping. Quadrotors are used singularly as well as in swarm; in the latter case, the task of coordination is always a critical issue. Quadrotors can be used both outdoor and indoor; outdoor platforms use, in general, autopilots for autonomous navigation, whereas several localization techniques (mainly based on computer vision) are exploited to determine position and orientation of indoor platforms, where GPS data are unavailable (for instance, [1], [6], [27], [29], [37]).

The human interface plays a key role when a quadrotor and, in general, any flying platform has to be directly controlled by the user. RC-transmitters

and joysticks are the two most common input devices used to control quadrotors. Innovative solutions use multitouch devices (e.g., Apple iPhone [2] and Microsoft Surface [33]) and game controllers (e.g., Nintendo Wiimote [38]). Initial attempts of Microsoft Kinect usage to control the Ar.Drone have been proposed in [42] and [43]. In both cases, hand gestures are translated into commands for the platform.

The Parrot AR.Drone [2] is a quadrotor helicopter with Wi-Fi link and two on-board cameras: a wide angle front camera and a high speed vertical camera. Software clients to control the platform are available: Microsoft Windows/Linux PC clients and an application for the iPhone can be used to control the Ar.Drone by keyboard, joystick or a multitouch device. The Parrot AR.Drone provides automatic "procedures" for takeoff, landing, and hovering. A public SDK is available to implement custom applications for the quadrotor control; the Windows client has been used as the starting point to develop the proposed solution (see Section 3). The SDK can be used to connect the AR.Drone to ad-hoc Wi-Fi networks, send commands (takeoff, land, up/down, rotate, and so on), receive, decode and display live video streams from the two cameras, receive and interpret navigation data and battery status. Although the Ar.Drone is sold in Europe at a price of about 300 euros as *the flying video game*, an impressive number of customers use this platform for technical and research purposes.

3. The natural user interface

A high-level description of the NUI is provided in Fig. 1. The user's body is tracked by the Microsoft Kinect [15], which is connected to a PC

Figure 1: A high-level description of the NUI.

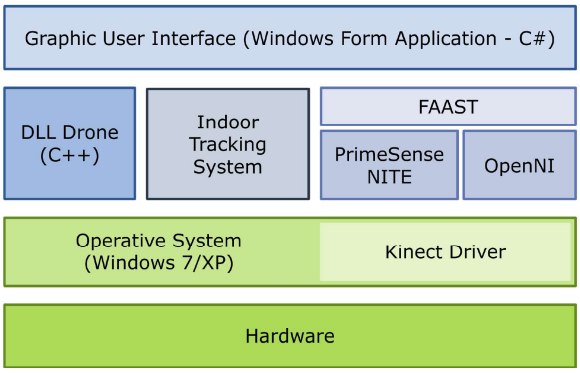

Figure 2: Software layers of the NUI.

(the control station) via USB; gestures (body postures) are translated into commands to be sent to the platform via a Wi-Fi connection. The goal is to let the user completely control the quadrotor movements by using the body as a kind of natural controller; moreover, an ad-hoc developed GUI (Graphics User Interface) allows the user to remotely oversee the platform as flight altitude, navigation data (telemetry), and video streams from the on-board cameras are displayed on the control station screen.

From the software point of view, the architecture of the NUI is shown

in Fig. 2. The stack composed by FAAST (Flexible Action and Articulated Skeleton Toolkit [11]), OpenNI - PrimeSense Nite, and the Kinect drivers is used to capture and decode body postures. FAAST is a middleware to facilitate integration of full-body control with games and VR applications using OpenNI-compliant depth sensors (e.g., the Microsoft Kinect). The toolkit incorporates a custom VRPN (Virtual-Reality Peripheral Network [40]) server to stream the user's skeleton over a network, allowing VR applications to read the skeletal joints as trackers using any VRPN client. FAAST can also emulate keyboard inputs triggered by body posture and specific gestures.

On the other hand, the OpenNI Framework [17] provides the interface for physical devices and for middleware components. APIs enable modules to be registered in the OpenNI framework and to be used to produce sensory data. OpenNI covers communication with both low-level devices (e.g., Microsoft Kinect), as well as high-level middleware solutions (e.g., FAAST). OpenNI can interact with the Microsoft Kinect by the OpenKinect library [16]. Body postures detected by FAAST are used by the GUI to trigger a modified version of the keyboard-based Ar.Drone client (the DLL Drone module in Fig. 2), thus implementing an effective and robust command chain to control the platform. Moreover, the GUI has been designed to receive information about position and orientation of the platform from an "external" system (e.g, an optical tracking system or a visual pose estimation system), thus supporting the implementation of AI (Artificial Intelligence) mechanisms to replace the user's control.

Fig. 3 shows the data exchanged among the NUI components. The Ar.Drone sends the GUI navigation data and the video stream and receives

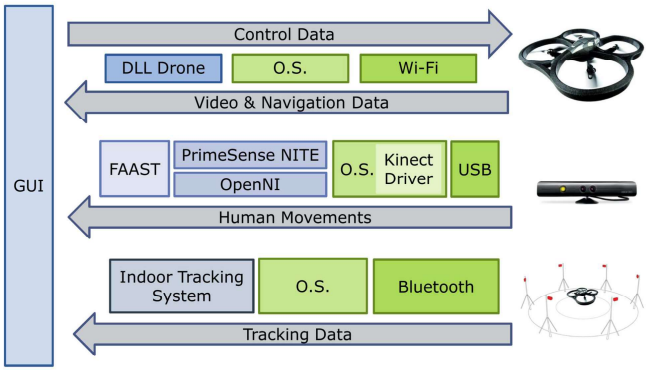

Figure 3: Data exchanged among the NUI components.

navigation commands. Each command is the *translation* of a body posture according to Table 3. This table is used by FAAST to trigger a set of keyboard events related to platform commands. The correspondence between body postures and commands for the quadrotor described in the table has been conceived basing on a series of tests performed using the real platform and the common sense of the designers of the overall framework. This translation table has been used for the user tests hereinafter described, whom have been properly instructed. However, it is worth noting that the correspondences can be easily adapted to the users need (e.g. for left-handed people). Each posture (also called action) is associated with a threshold; for instance the syntax `lean_forward 15` sets a lean forward of at least 15 degrees to activate the corresponding action. The threshold defines the *sensibility* in recognizing a given body posture and it can be thought as the joystick "deadzone", i.e., the region of movement that is not recognized by the device.

The user can customize the association among actions and platform com-

mands, thus choosing the body postures more intuitive and effective. Threshold values of 20-25 degrees have been experienced as a good tradeoff between robustness (i.e., the system detects the right posture) and sensibility (i.e., the size of the "deadzone"). A screenshot of a flying session is shown in Fig. 4: the GUI of the control station is visible on the left, whereas the platform is shown on the right. A video showing an example of Ar.Drone control by body movements can be found in [41]. The video allows to appreciate both the intuitiveness of the HRI and the graphics output the user can exploit to control the platform.

To evaluate the efficiency of the proposed NUI, a comparative analysis involving different human machine interfaces has been carried out. The methodology adopted for the evaluation has been based on a set of tests to be performed by a group of users. Tests consisted in repeating one or more navigation tasks by using a joystick, a multitouch device (iPhone), and the proposed solution. Users were asked for performing complete flight sessions (also called missions) from takeoff to landing. The results have been gathered by measuring the time needed to complete the sessions and the precision in landing has been considered to mark each mission has completed (C), uncompleted (U) or semi-completed (S) basing on the distance between the expected landing position and the real one. The expected landing position was a target box of size 56 cm $\times$ 56 cm. In particular, a mission is considered completed if the user was able to land on the target box, semi-completed if the landing was partially outside the target box and uncompleted if the landing was outside the expected landing position. Tables 1 and 2 shows the collection of these preliminary results. In particular, it is worth observ-

Figure 4: Screen of the control station (left) and flying platform (right).

ing that the proposed solution reached a score of 100% completed missions. These tests emerged some useful information for the usability of the system. Despite the fact that the use of Microsoft Kinect reached the most accurate result, this comes at the expense of speed of completion of the mission. On the other hand, the use of the iPhone device was critical for those who have never used a multi-touch device. Concerning the performance of the recognition technique, during the tests all the user postures were correctly recognized.

4. Visual estimation of the quadrotor position

Two defined body postures (see Table 3) allow the user to switch from manual to autonomous flight and viceversa. When the quadrotor is not controlled by user's postures, it autonomously moves in the environment by following a pre-specified path composed by a set of points named way-points. In order to localize the platform in the environment with respect to a well-defined World Coordinate System (WCS), a visual odometry algorithm has been implemented. The basic idea behind the algorithm is to use the features present in the environment, in particular on the floor, to estimate the position

Table 1: User study - Experimental results using three different input devices

Device	User	Time 1 [s] (Test Result 1)	Time 2 [s] (Test Result 2)
Joystick	user01	45 (S)	46 (C)
	user02	22 (U)	21 (C)
	user03	26 (C)	20 (S)
	user04	23 (C)	13 (S)
iPhone	user01	61 (C)	20 (S)
	user02	47 (U)	43 (C)
	user03	42 (U)	67 (U)
	user04	37 (U)	45 (U)
Kinect	user01	82 (C)	70 (C)
	user02	48 (C)	56 (C)
	user03	63 (C)	54 (C)
	user04	58 (C)	61 (C)

Table 2: User study - Statistics

Device	Average Time [s]	(U)ncompleted %	(S)emi-completed %	(C)ompleted %
Joystick	27	12,5 %	37,5 %	50 %
iPhone	45,25	62,5 %	12,5 %	25 %
Kinect	61,5	0 %	0 %	100 %

Table 3: Correspondence between body postures detected by FAAST [11] and commands for the quadrotor

Body posture	Command	Body posture	Command
Right arm up	Takeoff	Right arm down	Landing
Lean forward	Go forward	Lean backward	Go backward
Lean right	Go right	Lean left	Go left
Left arm up	Go up	Left arm down	Go down
Left arm out	Turn left	Right arm out	Turn right
Right foot up	Aut. flight on	Left foot up	Aut. flight off
Rest position	Hovering		

of the quadrotor vertical camera. The pose estimation system estimates the 3D rotation and translation of the quadrotor from 2D images coming from an on-board camera. The SiftGPU implementation of Lowe's SIFT [13] has been used to accomplish this task. The accuracy of this kind of measure is strongly affected by several parameters: the number of extracted features, the quality of the camera, the illumination of the environment, and so on. Therefore, significative errors can be cumulated over the time (drifts). In the proposed framework, drifts are bounded by placing a certain number of tags on the floor; each tag is placed at a well defined position with respect to the WCS. When a tag enters in the camera field of view, the pose estimation system switches from the feature-based to the tag-based working mode, thus resetting the cumulated drifts. The ArToolKit library [3] has been used to implement the tag-based pose estimation system.

The visual odometry algorithm, which is the base brick to estimate the

position of the quadrotor, is shown in Fig. 5. A thread is in charge to gather the frame coming from the vertical camera and to cope with the synchronization of the two pose estimation systems (feature-based and tag-based). Then, two threads run concurrently. The first thread searches for a visual marker in the frame, whereas the second thread calculates the correlations (matches) between features extracted from the current frame and features computed over a reference image. When a tag is identified, the absolute position of the camera is computed and values are forwarded to the feature-based pose estimation system to reset the drifts. Also the altitude (measured by the ultra sound sensors of the quadrotor) is sent to the feature-based pose estimation block in order to minimize errors; in particular, it is assumed that the platform is flying over a planar surface. Then, a switch block selects the most appropriate pose estimation algorithm for the current frame and sends position values to a filter for noise minimization. The algorithm provides position and flight altitude of the quadrotor, thus allowing to map it in the environment. Algorithm steps are repeated for each frame when the platform flies autonomously.

The tag-based pose estimation system analyzes a given frame to identify a visual marker. The first step of the algorithm is marker detection, during which the system extracts information about any marker placed into the current frame through thresholding and corner detection techniques. Whenever a marker is detected, it should be identified; during this phase, a unique marker identifier is extracted depending on its pattern. Since each marker is placed in known locations, the identification of a marker returns early geo referential data that will be refined in the next steps. Basing on the position

of marker's edges and corners, a pose estimation block extracts the position of the marker with respect to the camera reference system. In particular, the transformation matrix of the marker is computed. A change of the reference system through an inversion of the transformation matrix is necessary to find the position of the camera with respect to the framed marker. Finally, the new transformation matrix is used in combination with the early geo referential data of the specific marker to find the absolute position of the camera (and therefore of the quadrotor) in the environment.

Whenever markers cannot be detected, the feature-based algorithm is triggered. The current frame coming from the vertical camera is analyzed to compute a list of keypoints and their descriptors, which represent the salient features of the image. This process is performed through the SiftGPU implementation of Lowe's SIFT [13]. During the keypoints matching phase, the current list of keypoints is compared against a previous list of keypoints, computed at a reference frame, to find any correspondence among them. The reference image is not continuously updated at each analyzed frame, rather it is updated when the number of matches is lower than a given threshold. The next step is the pose estimation computed on the common features of the two considered images (the current and the reference frame). This process is performed by the Orthogonal Iteration (OI) algorithm (described in the details in [14]) that returns the new position and orientation in the camera reference system. Therefore, a change of the reference system is needed to express the position and orientation of the quadrotor with respect to an absolute reference system, which origin is placed where the mission started.

The feature-based pose estimation algorithm has the clear benefit of being

able to estimate the position and rotation of the quadrotor without knowing in advance any characteristics of the surrounding environment. Position and rotations are computed incrementally with respect to the take-off location and attitudes. On the other hand, the main drawback is that it is heavily influenced by drifts. The tag-based pose estimation algorithm has the main advantage of being more precise than the feature-based one. This behavior is clearly visible in the study of the position error shown in Figure 7, where the position error due to the use of the tag-based pose estimation is significantly lower. On the other hand, the main drawback of the tag-based pose estimation algorithm is due to the fact that for the correct functioning the environment should contain reference points.

4.1. Performance

Several tests have been performed to characterize the two pose estimation systems in order to evaluate the error during the localization process. Tests were aimed at measuring the error of the two separate systems and, afterwards, the error of the two systems working together as shown in Fig. 5. Moreover, characterization tests were aimed at identifying the best setup of the environment; in particular, the size of the tags, the optimal flight altitude and the minimum number of floor features have been determined. All these parameters are strongly dependent on the quality of the vertical camera that, unfortunately, in the current setup provides very low resolution frames, thus limiting the maximum flight altitude. A satisfactory trade-off can be found using 20×20 cm tags, flying at an altitude of about 70 cm and extracting about 250 matches at each correlation step. Assuming the floor as the $X - Y$ plane, Fig. 6 shows an example of error characterization when,

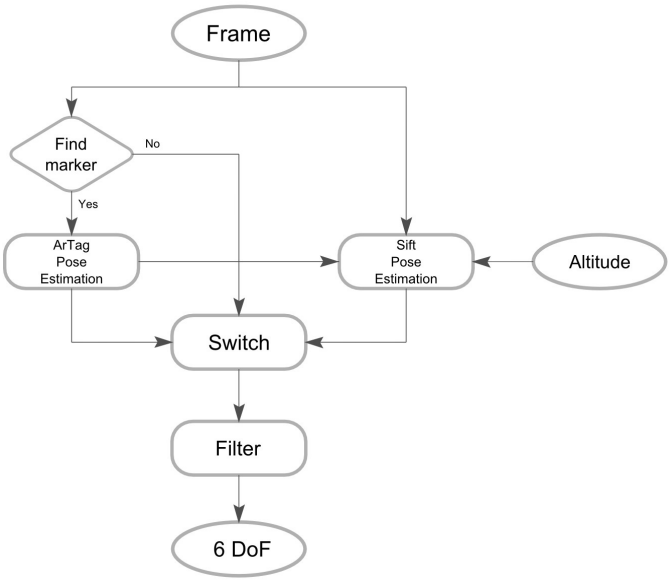

Figure 5: The flow chart of the pose estimation algorithm.

with the above setup, both pose estimation systems are used. Fig. 6 deals with the error along the Y coordinate, but similar results have been obtained for the X coordinate. The altitude is measured by the ultra sound sensors of the platform, whereas orientation errors are on the average of 2 degrees around a rotation axis (the most important angle is the heading, that is the rotation around the Z axis).

The measures obtained by the algorithm presented in Section 4 have been compared with values computed by an infrared tracking system [9]: the blue curve represents the value of Y measured by the tracking system, whereas the red curve shows values obtained by the proposed solution. It is worth observing that drifts grow with the number of samples (i.e., the time) and they are almost reset when a tag is found in the frame (the red curve is

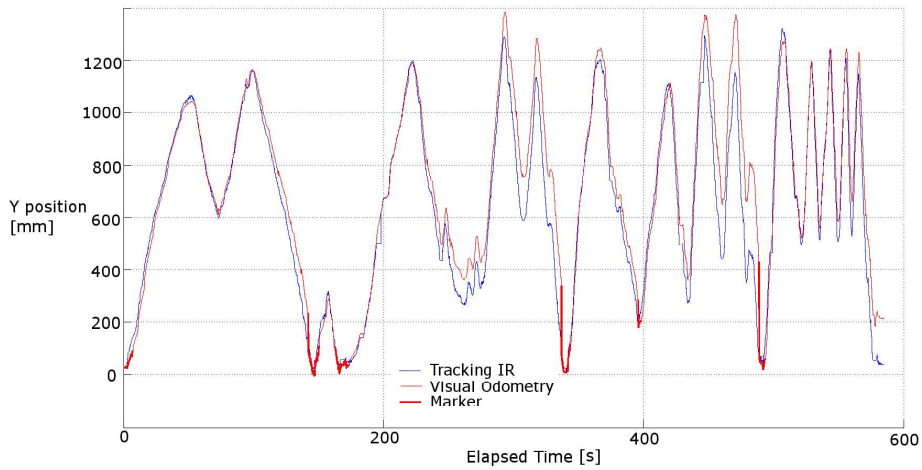

Figure 6: Measure of the position along the Y coordinate.

thicker). This allows the system to keep the error bounded, thus providing a robust localization solution.

5. Conclusion

This paper presents a framework for controlling the navigation of a quadrotor in GPS-denied environments. A NUI based on body gestures/postures allows the user to control the platform, whereas a hybrid visual odometry algorithm (based on both tag detection and features extraction) supports autonomous navigation.

The latency of the system has been also measured: the term latency denotes the delay between a change in user's posture and the execution of the corresponding command. The measure has been performed by analyzing the video sequence in [41] and counting the number of frames elapsed between user and Ar.Drone movements. An average latency of 0.3 seconds has been

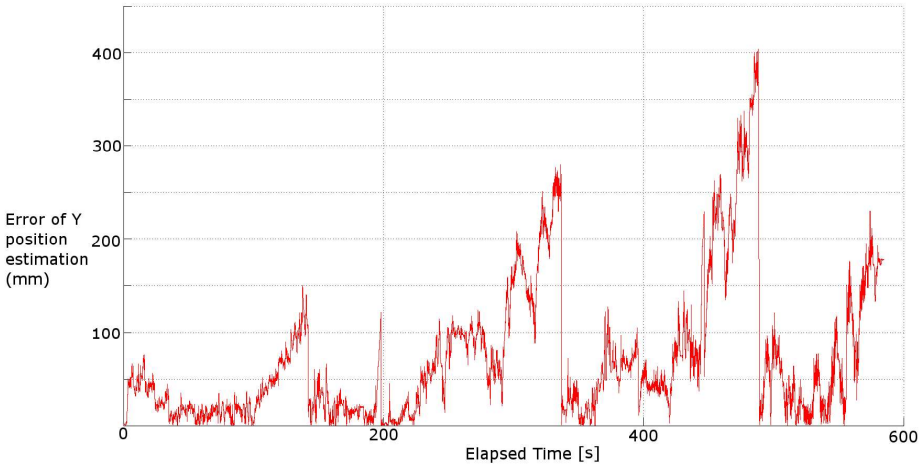

Figure 7: Measure of the error along the Y coordinate.

experienced. Thus, about three commands can be executed in a second, which is fully consistent both with the platform's dynamic and the "user's dynamic". On the other hand, the term latency assumes a different meaning when the autonomous flight is enabled. In this case, the latency is the delay between the acquisition of a frame by the vertical camera and its processing on the control station. The experienced latency is about 2 seconds and this limits the speed of the platform.

Results presented in this paper showed how affordable devices such as the Microsoft Kinect are opening new scenarios allowing to create innovative forms of HRI unthinkable until a few months ago. The evolution of devices designed to implement novel user-centric forms of entertainment will provide researchers with alternative tools to re-design more intuitive, robust and fun HRI paradigms.

References

[1] M. Achtelik, A. Bachrach, R. He, S. Prentice, N. Roy, Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments, in: Int. Aerial Robotics Competition, 1st Symposium on Indoor Flight Isssues, 2009.

[2] Ar.Drone web site, http://ardrone.parrot.com

[3] Artoolkit documentation, http://www.hitl.washington.edu/artoolkit

[4] S.J. Hong, N.A. Setiawan, C.W. Lee, Real-time Vision Based Gesture Recognition for Human-Robot Interaction, in: Proc. of the 11th International COnference KES 2007 and XVII Italian workshop on neural networks conference on Knowledge-based intelligent information and engineering systems: Part I, 493-500, 2007.

[5] R.A. Bolt, Put-that-there: Voice and gesture at the graphics interface, in: Proc. Siggraph, ACM NY, 262-270, 1980.

[6] F. Caballero, L. Merino, J. Ferruz, A. Ollero, Vision-Based Odometry and SLAM for Medium and High Altitude Flying UAVs, Journal of Intelligent and Robotic Systems, 54(2009), 137-161.

[7] C. Celozzi, G. Paravati, A. Sanna, F. Lamberti, A 6-DOF ARTag-Based Tracking System, IEEE Transactions on Consumer Electronics, 56(1), 203-210, 2010.

[8] N. Chao, M.Q. Meng, P. Xiaoping Liu, X. Wmg, Visual gesutre recognition for human-machine interface of robot teleoperation, in: the

IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, 1560-1565, 2003.

[9] S. De Amici, A. Sanna, F. Lamberti, B. Pralio, A Wii Remote-based infrared-optical tracking system. Entertainment Computing, 1(2010), 119-124.

[10] Interaction with a Quadrotor via the Kinect, ETH Zurich, http://www.youtube.com/watch?v=A52FqfOi0Ek

[11] FAAST web site, http://projects.ict.usc.edu/mxr/faast/

[12] J. Kofman, Xianghai Wu, T.J. Luu, S. Verma, S.: Teleoperation of a robot manipulator using a vision-based human-robot interface. IEEE Transactions on Industrial Electronics, 52(2005), 1206-1219.

[13] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, 60(2004), 91-110.

[14] C.-P. Lu, G. D. Hager, E. Mjolsness: Fast and Globally Convergent Pose Estimation from Video Images, IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 22, NO. 6, June 2000.

[15] Microsoft Kinect web site, http://www.xbox.com/en-US/kinect/

[16] OpenKinect web site, http://openkinect.org/wiki/Main_Page

[17] OpenNI web site, http://www.openni.org/

[18] G. Paravati, C. Celozzi, A. Sanna, F. Lamberti, A feedback-based control technique for interactive live streaming systems to mobile devices. IEEE Transactions on Consumer Electronics, 56(1), 190-197, 2010.

[19] A. Sanna, B. Pralio, F. Lamberti, G. Paravati, A novel ego-motion compensation strategy for automatic target tracking in FLIR video sequences taken from UAVs. IEEE Transactions on Aerospace and Electronic Systems, 45(2), 723-734, 2009.

[20] F. Lamberti, A. Sanna, G. Paravati, Improving robustness of infrared target tracking algorithms based on template matching. IEEE Transactions on Aerospace and Electronic Systems, 47(2), 1467-1480, 2011.

[21] G. Paravati, A. Sanna, B. Pralio, F. Lamberti, A genetic algorithm for target tracking in FLIR video sequences using intensity variation function. IEEE Transactions on Instrumentation and Measurement, 58(10), 3457-3467, 2009.

[22] G. Paravati, A. Sanna, F. Lamberti, C. Celozzi, A reconfigurable multi-touch framework for teleoperation tasks, in: 16th IEEE International Conference on Emerging Technologies and Factory Automation, 1-4, 2011.

[23] G. Paravati, B. Pralio, A. Sanna, F. Lamberti, A reconfigurable multi-touch remote control system for teleoperated robots, in: 29th IEEE International Conference on Consumer Electronics, 153-154, 2011.

[24] V.I. Pavlovic, R. Sharma, T.S. Huang, Gestural interface to a visual computing environment for molecular biologists, in: Proc. of the 2nd

Int. Conference on Automatic Face and Gesture Recognition, 52-73, 1996.

[25] V.I. Pavlovic, R. Sharma, T.S. Huang, Visual interpretation of hand gestures for human-computer interaction: a review, IEEE Transactions on Pattern Analysis and machine Intelligence, 19(1997), 677-695.

[26] V.S. Rao, C. Mahanta, Gesture Based Robot Control, in: Proc. of the 4th International Conference on Intelligent Sensing and Information Processing, 145-148, 2006.

[27] S. Se, P. Jasiobedzki, Stereo-Vision Based 3D Modeling and Localization for Unmanned Vehicles, International Journal, 13(2008), 46-57.

[28] T. Selker, Touching the future, Commun. ACM 51 (2008), 14-16.

[29] S.P. Soundararaj, A.K. Sujeeth, A. Saxena, Autonomous Indoor Helicopter Flight using a Single Onboard Camera, in: Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems, 5307-5314, 2009.

[30] C.J.P. Soshi Iba, J. Michael Vande Weghe, P.K. Khosla, An Architecture for Gesture-based Control of Mobile Robots, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 851-857, 1999.

[31] J. Sugiyama, D. Tsetserukou, and J. Miura, NAVIgoid: robot navigation with haptic vision, in: Proc. Int. Conf. on Computer Graphics and Interactive Technologies (ACM SIGGRAPH Asia 2011), Emerging Technologies, Article No. 9, Hong Kong, China, December 12-15, 2011.

[32] D.J. Sturman, D. Zetler, A survey of glove based input, IEEE Computer Graphics and Applications, 14(1994), 30–39.

[33] Controlling the AR Drone with Microsoft Surface, http://blogs.msdn.com/b/surface/archive/2011/01/27/controlling-the-ar-drone-with-surface.aspx

[34] D. Tsetserukou, J. Sugiyama and J. Miura, Belt tactile interface for communication with mobile robot allowing intelligent obstacle detection, in: Proc. IEEE World Haptics Conference (WHC 2011), Istanbul, Turkey, June 21-24, 2011, pp. 113-118.

[35] J.P. Wachs, H. Stern, Y. Eden, Parameter search for an image processing-Fuzzy c-Means hand gesture recognition system, in: Proc. of the IEEE Int. Conference on Image Processing, 341-344, 2003.

[36] J.P. Wachs, H. Stern, Y. Edan, Cluster Labeling and Parameter Estimation for the Automated setup of a Hand gesture Recognition System, IEEE Transactions Systems and Humans 35(2005), 932-944.

[37] Y. Wang, A. Camargo, R. Fevig, F. Martel, R.R. Schultz, Image Mosaicking from Uncooled Thermal IR Video Captured by a Small UAV, in: Proc. of the IEEE Southwest Symposium on Image Analysis and Interpretation, 161-164, 2008.

[38] Controlling the AR Drone with Nintendo Wiimote, http://www.youtube.com/watch?v=zJ50H-_431w&feature=player_embedded

[39] A. Wright, Making sense of sensors, Commun. ACM, 52(2009), 14-15.

[40] The Virtual-Reality Peripheral Network, http://www.cs.unc.edu/Research/vrpn/

[41] Controlling the AR Drone with Microsoft Kinect, http://www.youtube.com/watch?v=jDJpb4xXAJM

[42] Hand tracking to control the AR Drone with Microsoft Kinect, http://dronehacks.com/2010/12/21/controlling-the-ar-drone-with-a-kinect-controller/

[43] Hand tracking to control the AR Drone with Microsoft Kinect, http://www.youtube.com/watch?v=mREorv0hbY8