A Kinect-based Natural Interface for Quadrotor Control

Andrea Sanna, Fabrizio Lamberti, Gianluca Paravati, Eduardo Andres Henao Ramirez, and Federico Manuri

Politecnico di Torino, Dipartimento di Automatica e Informatica,
C.so Duca degli Abruzzi 24, I-10129 Torino, Italy
{andrea.sanna,fabrizio.lamberti,gianluca.paravati,eduardo.henaoramirez, federico.manuri}@polito.it

Abstract. The evolution of input device technologies led to identification of the natural user interface (NUI) as the clear evolution of the human-machine interaction, following the shift from command-line interfaces (CLI) to graphical user interfaces (GUI). The design of user interfaces requires a careful mapping of complex user "actions" in order to make 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. NUIs constitute a direct expression of mental concepts and the naturalness and variety of gestures, compared with traditional interaction paradigms, can offer unique opportunities also for new and attracting forms of human-machine interaction. In this paper, a kinect-based NUI is presented; in particular, the proposed NUI is used to control the Ar.Drone quadrotor.

Key words: Natural User Interface, Kinect, Quadrotor control, Interactive systems.

1 Introduction

Gestures are important factors in conversations between humans and researchers have designed and implemented several human-computer interaction (HCI) paradigms, thus creating the so called Natural User Interfaces (NUIs). NUIs have been investigated since early eighty's (voice and gestures are used to control a GUI in [3]). Among NUIs, gesture-based interfaces ever played a crucial role in human communication, as they constitute a direct expression of mental concepts [15]. The naturalness and variety of hand and body gestures, compared with traditional interaction paradigms, can offer unique opportunities also for new and attracting forms of HCI [14]. 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. [19][27]) 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 the HCI and can be considered as one of the most important Computer Vision domains. In HRI-based systems, especially in safe critical applications such as search-and-rescue and military, it is increasingly necessary that humans 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 [22]. These devices limit the speed and naturalness of interaction. To overcome the limitations of such electro-mechanical devices, vision based techniques [15] have been introduced. Vision based techniques do not require wearing of any contact devices, but use a set of sensors and computer vision techniques for recognizing gestures. Thus, 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 a system is that it is a natural way to send geometrical information to the robot, such as: up, down, etc. 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.

Although a lot of works of HRI by gestures are known in the literature (Section 2 briefly reviews the most appropriate) recent technological advances have opened new and challenging research horizons. In particular, controllers and sensors used for home entertainment can be affordable devices to design and implement new HRI forms.

Microsoft Kinect [11] is used as gesture tracking device in this paper; recognized gestures are then used to control the Ar.Drone quadrotor platform [1] (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 be experienced. Tests proved as the platform can be easily controlled by a customizable set of body movements, thus allowing an exiting, fun, and safe experience also to non skilled users.

The paper is organized as follows: Section 2 reviews the main HRI solutions and briefly introduces the Ar.Drone. Section 3 describes the system architecture and the mapping between gestures and commands. Finally, remarks about this experience and future investigation trends are presented in Section 4.

2 Background

The ability to recognize gestures is important for an interface developed to understand users intentions. Interfaces for robot control that use gesture recognition have deeply been studied as using gestures provides a formidable challenge. Several issues arise from the environment that contains a complex background and 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 works have have been focused on hand gesture recognition. For instance, an architecture of hand gesture-based control of mobile robots was proposed in [20]. The gestures were captured by a data glove and gesture recognition was done by Hidden Markov Model statistical classifiers. The interpreted gestures were translated into commands to control the robot. An alternative identification of the hand posture was also proposed in [5]. The hand posture is identified from the temporal sequence segmented obtained by the Hausdorff distance method. A real time vision based gesture recognition system for robot control was implemented in [2]. Gestures were recognized using rule based approach by comparing the skin like regions in a particular image frame with the predefined templates in the memory of the system. Another hand gesture recognition system for robot control, which uses Fuzzy-C-Means algorithm as gesture classifier to recognize static gestures, was proposed in [24] and [25]. Static and dynamic gestures are recognized by a Fuzzy-C-Means clustering algorithm in [18].

YCbCr segmentation to recognize hand gestures has been proposed in [21], whereas a real-time hand posture recognition using 3D range data analysis is presented in [9]. A background subtraction approach using video sequences is proposed in [17], whereas motion detection algorithms for gesture recognition are used in [10]. A trajectory-based hand gesture recognition, which uses kernel density estimation and the related mean shift algorithm, was presented in [16]. A method for detecting and segmenting foreground moving objects in complex scenes using clusters is used in [4]. Under the assumption that the target object occupies the entire image, the humans body proportions are considered and using (vertical and horizontal) histogram analysis the hand gesture is recognized by simply using a webcam in [8].

In this paper, a novel method of interaction and control of quadrotors by whole body movement recognition is described. Microsoft Kinect allows users to experience a new type of HRI able to provide an intuitive, robust and fun interaction form.

2.1 Quadrotors and the platform Ar.Drone

Quadrotors are used in a large spectrum of applications ranging from surveillance to environmental mapping. Quadrotors are used singularly as well as is swarm; in this last 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.

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 and the two most common input devices used to control quadrotors. Innovative solutions uses multitouch devices (e.g., the iPhone [1] and Microsoft Surface [23]) and game controllers (e.g., Nintendo Wiimote [26]). Initial attempts of Microsoft Kinect usage to control the Ar.Drone have been proposed in [30] and [31]. In both cases, hand gestures are translated in commands for the platforms.

The Parrot AR.Drone [1] is a quadrotor helicopter with Wi-fi link and two cameras: a wide angle front camera and a high speed vertical camera. Software clients to control the platform are available: Windows/Linux PC clients and an application for 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 to the AR.Drone ad-hoc Wi-fi network, send commands (take-off, land, up/down, rotate, and so on), receive, decode and display live video stream from the two cameras, receive and interpret navigation data and battery status. Although the Ar.Drone is sold in Europe to a price of about 300 euros as *the flying video game*, an impressive number of users use this platform for technical and research purposes.

3 System Architecture

A high-level description of the system is provided in Fig. 1. The user's body is tracked by the Microsoft Kinect [11], that is connect to a PC via USB; gestures (body poses) are translated in commands to be sent to the platform via Wi-Fi connection. The user will be able to completely control the quadrotor movements by using the body as a sort of natural controller; moreover, an ad-hoc developed GUI (Graphics User Interface) enables the user to remotely control the platform as attitudes flight, navigation data (telemetry), and the video stream from the onboard cameras are shown, thus releasing the user to directly see the quadrotor.

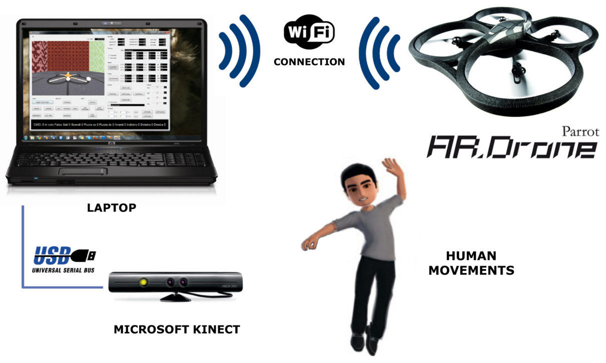

Fig. 1. A high-level description of the system.

From the software point of view, the architecture is shown in Fig. 2. The stack composed by FAAST (Flexible Action and Articulated Skeleton Toolkit [7]), OpenNI - PrimeSense Nite, and the Kinect drivers is used to capture and decode body poses. FAAST is a middleware to facilitate integration of full-body control with games and VR applications using OpenNI-compliant depth sensors (e.g., Microsoft Kinect). The toolkit incorporates a custom VRPN (Virtual-Reality Peripheral Network [28]) 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 input triggered by body posture and specific gestures.

On the other hand, the OpenNI Framework [13] provides the interface for physical devices and for middleware components. APIs enable modules to be registered in the OpenNI framework and 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 [12].

Body poses 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 optical tracking system. Information coming from the optical tracker (the affordable system proposed in [6] has been used for tests) allow to implement mechanisms of AI (Artificial Intelligence) to control the quadrotor, thus replacing the user.

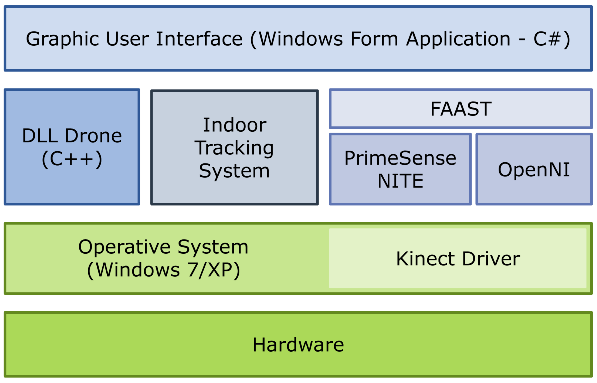
Graphic User Interface (Windows Form Application - C#)
DLL Drone (C++)
Indoor Tracking System
FAAST
PrimeSense NITE
OpenNI
Operative System (Windows 7/XP)
Kinect Driver
Hardware

Fig. 2. Layers of the software architecture.

Fig. 3 shows the exchanged data among system components. The Ar.Drone sends the GUI navigation data and the video stream, whereas it receives navigation commands. Each command is the *translation* of a body pose according to Table 1. This table is used by FAAST to trigger a set of keyboard events related to platform commands. Moreover, each pose (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 thresholds define the *sensibility* in recognizing body poses and they can be thought as the joystick deadzone, that is the region of movements which are not recognized by the device.

The user can customize the association between action and platform commands, thus choosing the body poses more intuitive and effective. Threshold

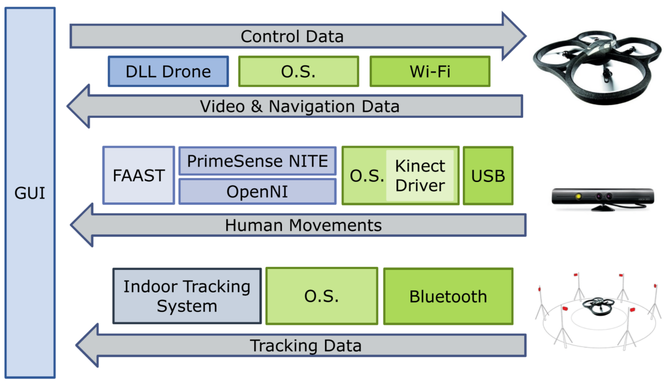

Fig. 3. Exchanged data among system components.

Table 1. Correspondence between body poses and commands for the quadrotor

Body pose	Ar.Drone command
Right arm up	Take-off
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
Rest position	Hovering

values of 20-25 have been experienced as a good tradeoff between robustness (i.e., the system really detects the right pose) and sensibility (i.e., the size of the "deadzone"). A video showing an example of Ar.Drone control by body movements can be found in [29]. The video allows to appreciate both intuitiveness of the HRI and the graphics output the user can use to control the platform.

4 Conclusion and remarks

This paper presents an example of NUI based on body gestures/movements to control a quadrotor. Although this work shows a challenging and exciting scenario, a more accurate and rigorous study is necessary to evaluate the efficiency of this kind of solution. A comparative analysis involving interfaces based on: keyboard, joystick, a multitouch device, and the proposed solution is scheduled. A set of users will be asked to control the Ar.Drone to perform a well defined mission; results in terms of completion times, number of interactions with the system (i.e, number of commands necessary to complete the mission), and sub-

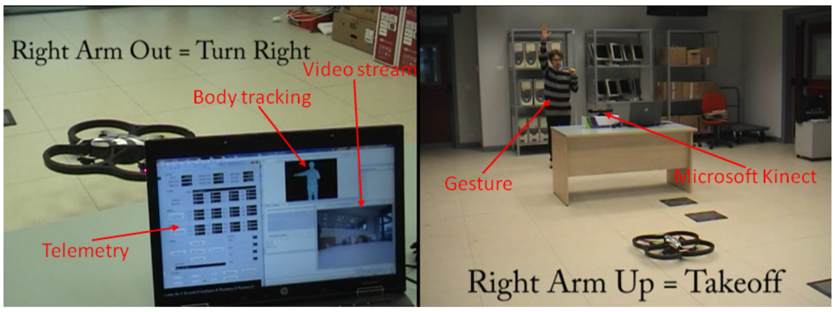

Fig. 4. Two pictures of the experimental setup. On the left the laptop console is shown, whereas the Microsoft Kinect is visible on the right.

jective user evaluations will be the input of a statistical analysis necessary to assess performance, robustness, and usability of the proposed interface.

At this moment, the whole latency of the system has been measured: the term latency denotes, in this case, the delay between a user's movement and the execution of the corresponding command. The measure has been performed by analyzing the video sequence in [29] and counting the number of frames elapse between user and Ar.Drone movements. An average latency of 0.3 seconds has been experienced. Thus, about three commands can be executed in a second, that is fully consistent both with the platform's dynamic and the "user's dynamic".

Affordable devices such as 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 provides researchers alternative tools to re-design more intuitive, robust, and fun HRI paradigms.

References

1. Ar.Drone web site, http://ardrone.parrot.com
2. Bhuiyan, M.A., Ampornaramveth, V., Muto, S., Ueno, H.: Realtime vision based Gesture Recognition for Human Robot Interaction. In: the IEEE International Conference on Robotics and Biometrics, pp. 413–418 (2004)
3. Bolt, R.A.: Put-that-there: Voice and gesture at the graphics interface. ACM Comput. Graphics, 14:3, 262-270 (1980)
4. Bugeau, A., Pérez, P.: Detection and segmentation of moving objects in complex scenes. Computer Vision and Image Understanding, 113:4, 459–476 (2009)
5. Chao, N., Meng, M.Q., Xiaoping Liu, P., Wmg, X.: Visual gesutre recognition for human-machine interface of robot teleoperation. In: the 2003 IEEEIRSJ Intl. Conference on Intelligent Robots and Systems, pp. 1560–1565 (2003)
6. De Amici, S., Sanna, A., Lamberti, F., Pralio B.: A Wii Remote-based infrared-optical tracking system. Entertainment Computing, 1:3-4, 119–124 (2010)
7. FAAST web site, http://projects.ict.usc.edu/mxr/faast/

8. Koceski, S., Koceska, N.: Vision-based gesture recognition for human-computer interaction and mobile robot's freight ramp control. In: the 32nd International Conference on Information Technology Interfaces, pp. 289–294 (2010)
9. Malassiotis, S., Aifanti, N., Strintzis, M.G.: A gesture recognition system using 3D data. In: the 1st International Symposium on 3D Data Processing Visualization and Transmission, pp. 190–193 (2002)
10. Mariappan, R.: Video Gesture Recognition System. In: the International Conference on Conference on Computational Intelligence and Multimedia Applications, pp. 519–521 (2007)
11. Microsoft Kinect web site, http://www.xbox.com/en-US/kinect/
12. OpenKinect web site, http://openkinect.org/wiki/Main_Page
13. OpenNI web site, http://www.openni.org/
14. Pavlovic, V.I., Sharma, R., Huang, T.S.: Gestural interface to a visual computing environment for molecular biologists. In: 2nd intern. conf. on automatic face and gesture recognition, pp. 52–73. IEEE Computer Society, Los Alamitos (1996)
15. Pavlovic, V.I., Sharma, R., Huang, T.S.: Visual interpretation of hand gestures for human-computer interaction: a review. IEEE Transactions on Pattern Analysis and machine Intelligence, 19, 677–695 (1997).
16. Popa, D., Simion, G., Gui, V., Otesteanu, M.: Real time trajectory based hand gesture recognition. WSEAS Transactions on Information Science and Applications, 5:4, 532–546 (2008)
17. Ribeiro, H.L., Gonzaga, A.: Hand Image Segmentation in Video Sequence by GMM: a comparative analysis. In: the 19th Brazilian Symposium on Computer Graphics and Image Processing, pp. 357–364 (2006)
18. Rao, V.S., Mahanta, C.: Gesture Based Robot Control. In: the 4th International Conference on Intelligent Sensing and Information Processing, pp. 145–148 (2006)
19. Selker, T.: Touching the future. Commun. ACM 51, 14–16 (2008)
20. Soshi Iba, C.J.P., Michael Vande Weghe, J., Khosla, P.K.: An Architecture for Gesture-based Control of Mobile Robots. In: the IEEEIRSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 851–857, (1999)
21. Stergiopoulou, E., Papamarkos, N.: A New Technique for Hand Gesture Recognition. In: the IEEE International Conference on Image Processing, pp. 2657–2660 (2006)
22. Sturman, D.J., Zetler, D.: A survey of glove based input. IEEE Computer Graphics and Applications, 14, 30–39 (1994)
23. Controlling the AR Drone with Microsoft Surface, http://blogs.msdn.com/b/surface/archive/2011/01/27/controlling-the-ar-drone-with-surface.aspx
24. Wachs, J.P., Stern, H., Eden, Y.: Parameter search for an image processing-Fuzzy c-Means hand gesture recognition system. In: the IEEE Int. Conf. on Image Processing, pp. 341–344 (2003)
25. Wachs, J.P., Stern, H., Edan, Y.: Cluster Labeling and Parameter Estimation for the Automated setup of a Hand gesture Recognition System. IEEE Transactions Systems and Humans 35:6, 932–944 (2005)
26. Controlling the AR Drone with Nintendo Wiimote, http://www.youtube.com/watch?v=zJ50H-_431w&feature=player_embedded
27. Wright, A.: Making sense of sensors. Commun. ACM 52, 14–15 (2009)
28. The Virtual-Reality Peripheral Network, http://www.cs.unc.edu/Research/vrpn/
29. Controlling the AR Drone with Microsoft Kinect, http://www.youtube.com/watch?v=jDJpb4xXAJM

30. Hand tracking to control the AR Drone with Microsoft Kinect, http://dronehacks.com/2010/12/21/controlling-the-ar-drone-with-a-kinect-controller/
31. Hand tracking to control the AR Drone with Microsoft Kinect, http://www.youtube.com/watch?v=mREorv0hbY8