Teleoperation Using Google Glass and AR.Drone

João Marcelo Teixeira
Departamento de Estatística e Informática
UFRPE
Recife, Brazil
jmxnt@deinfo.ufrpe.br

Ronaldo Ferreira, Matheus Santos, Veronica Teichrieb
Voxar Labs
Informatics Center – UFPE
Recife, Brazil
{rfaf, mfcs, vt}@cin.ufpe.br

Abstract— **This paper proposes using a wearable device for visualization and control in association with an Unmanned Aerial Vehicle applied to structural inspection of buildings. More specifically, an AR.Drone is controlled through head positions and gestures performed by an operator wearing a Google Glass, and images captured by the drone are visualized on Glass's screen. We discuss problems that arise when such a solution is developed, along with limitations that come from today's available technology and how to overcome them.**

Keywords— teleoperation; UAV; Google Glass; AR.Drone

I. INTRODUCTION

Structural inspection intends to evaluate the condition of the evident foundation performance, roof, and structure of the building in order to provide information related to their condition and an opinion as to whether they are in need of repair [1][2][3]. Data obtained will provide insight into the overall condition of the property and information that will assist in maintaining it in the best possible condition during future years. The scope of this inspection includes visual observations of those portions of the foundation, roof and structural components readily visible without moving or removing items causing visual obstruction.

Inspections to some areas of the structure may be limited by safety or site conditions, with most common limitations related to direct access at roof or entry to crawlspace. In order to alleviate such inherent difficulties, teleoperation-based solutions could be applied. According to [4], teleoperator devices enable human operators to remotely perform mechanical actions that usually are performed by human arm and hand. In the context of this work, it replaces human vision capability and mobility. Thus, teleoperators, or the act of teleoperation, extends human capabilities to remote, physically hostile, or dangerous environments. Compared to telecommunication, teleoperation conquers space barriers in performing manipulative mechanical actions at remote sites, while the former conquers space barriers in transmitting information to distant places.

This work proposes using a wearable device for visualization and control in association with an Unmanned Aerial Vehicle (UAV) [5] [6] applied to structural inspection of buildings. We discuss problems that arise when such a solution is developed, along with limitations that come from today's available technology and how to overcome most of them.

Vision, hearing and touch senses are relatively easy to transmit but smell and taste are more complicated. Fortunately, these two senses are less used machine teleoperation. The proposed teleoperation application presented in this work focuses on the first sense listed, which is user vision.

The main advantage of using such a wearable device to teleoperate an UAV is the naturalness of interaction between human and device. For example, to directly use head movements to control UAV direction seems to be more intuitive for most users. Another hypothesis is the flexibility that it provides to the user regarding his/her mobility, as it is possible to walk along the construction area while seeing UAV's view. Such assumptions will be detailed on next sections.

The rest of this paper is organized as follows. Section 2 presents related works regarding teleoperation in different scenarios, problems inherent to each of them and how they have been solved. Section 3 provides details regarding hardware infrastructure used in this work, the reason for choosing specific devices and how to use them. The project development is explained in section 4. This section includes aspects taken into consideration while designing the structural inspection activity by a teleoperator device and how interaction was mapped between operator/wearable device and UAV. Section 5 lists achieved results and proposes adaptations about how to extend the current solution in order to enable a broader set of scenarios. Finally, section 6 draws some conclusions and proposes future applications for methods and ideas developed herein.

II. RELATED WORK

Some researchers consider the genesis of teleoperation as being the creation of Polygraph, famously used by Thomas Jefferson, in 1805 [7]. Such device was capable of producing a copy of a piece of writing simultaneously with the creation of the original, using pens and ink, as shown in Fig. 1. By typically using a pantograph mechanism, a four-bar linkage with parallel bars enabled motion at one point to be reproduced at another point.

At the year of 1948, Ray Goertz, from the US Atomic Energy Commission, created the first Master-Slave manipulator [12]. His goal was to protect workers from radiation, while enabling precise manipulation of materials. The terms "slave" and "master" were used for the first time,

representing a device responsive to another and the controlling device, respectively. Since then, teleoperator devices evolved from mechanical linkages and cables, to electrical and hydraulic servomechanisms with force/haptic feedback and wireless remote control.

Nowadays, many areas benefit from teleoperation: space exploration [13], military/defensive applications [14], surveillance [15], underwater vehicles [17], telerobotics in forestry and mining applications [19][20], telesurgery and telepresence robots [21].

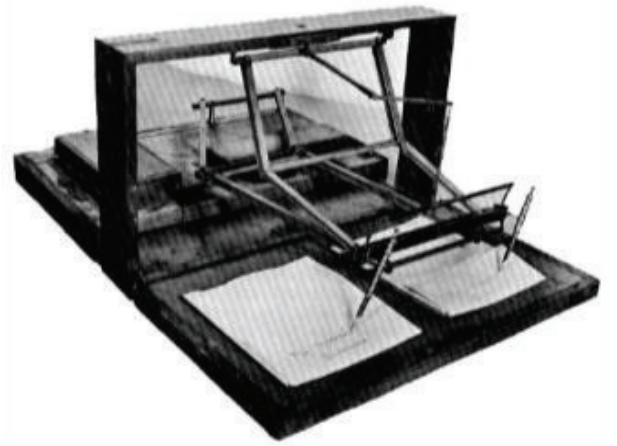

Fig. 1. Genesis of teleoperation: Polygraph [7].

Space is a very appropriate environment for teleoperation applications. Physical presence of a human to operate a vehicle in space requires many resources (Moon/Mars exploration) or is totally impossible (Sun exploration). Therefore, it is more efficient to use teleoperated vehicles [8]. Fig. 2 illustrates Robonaut [13], which is an additional control method, beyond ground control, that involves Robonaut mimicking crewmember motions via gloves, a vest and a 3D visor.

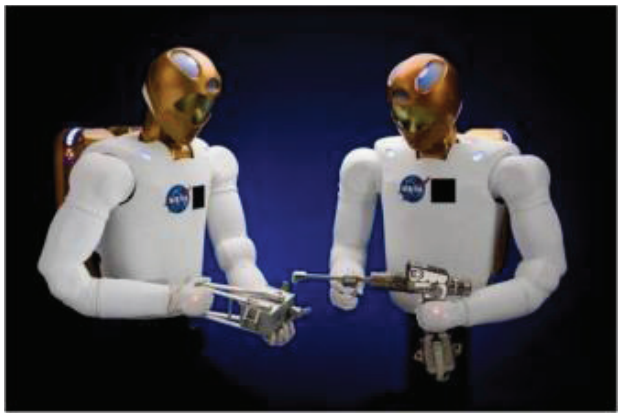

Fig. 2. Robonaut Tele-operations System [13].

As happens with space exploration, military applications include recon missions. Gathering information about an enemy or unknown landscape is a very common task for a military operation. In this scenario, individuals with higher spatial ability usually perform better while teleoperating robots than those with lower spatial ability. Fig. 3 illustrates Predator [14], an UAV originally developed by US Air Force for reconnaissance and forward observation roles, carrying cameras and other sensors. Later, it was modified and upgraded to carry and fire two AGM-114 Hellfire missiles or other munitions.

Fig. 3. Predator MQ-1 (top) and its teleoperator (bottom) [14].

Similar to military operations, security field presents some teleoperation systems. Due to increased number of terrorist actions, most police departments have created bomb squads for deactivating bombs. An example of teleoperation related to surveillance is the Rotundus robot [15]. It is a spherical robot without external feet or wheels which moves through the balance of a weight inside the sphere. It has two surveillance cameras behind shielded glass areas and can reach speeds of up to 70km/h. Another example, the Secom [16], a robot produced for Tokyo police department, can be used for surveillance and for stopping a person by spraying a cloud of gas. Both robots are illustrated in Fig. 4.

Underwater operations were one of the first mobile applications where teleoperation techniques were adopted. This class of ROVs (Remotely Operated Vehicles) represents the largest commercial market for mobile vehicle teleoperation, especially due to ultra deep water exploration for gas industry. Fig. 5 illustrates a robot for exploring the sea and another one dedicated to repairing underwater gas extraction machinery, capable of reaching a maximum depth of 2,000 meters.

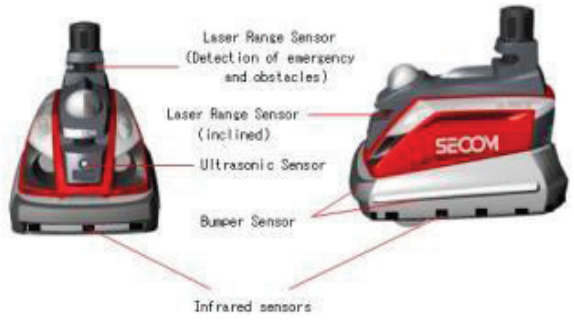

Fig. 4. Surveillance robots: Rotundus [15] (top) and Secom [16] (bottom).

Fig. 5. Underwater teleoperated robots: Victor 6000 [17] (top) and H2000 [18] (bottom).

Using heavy equipment in forestry and mining and hazards of falling trees, rough terrain and caving in mine galleries have imposed the use of teleoperated robots. In this field of application, teleoperated robots can be classified into three different categories: excavating machines, exploration robots and rescue robots. Fig. 6 illustrates some robots used in forestry/mining context.

Fig. 6. Forestry/mining robots: Centauroid Robot Work Partner [19] (top) and Groundhogbot [20] (bottom).

Another well-known example of teleoperation is in medical field. Technology has revolutionized surgery practices with the

creation of robotic devices and complex imaging. This enabled much less invasive operations, but still requires surgeons to control machines. Precision that comes from teleoperated robots for surgery makes more homogeneous results, since both experienced and less-experienced surgeons tend to deal with the controller, and not directly with the patient. Fig. 7 illustrates a teleoperation system used for telesurgery.

Fig. 7. Da Vinci Surgical System [21].

The concept of telepresence means that operator feels like he/she is present at teleoperator site. Already the simple camera-monitor combination creates some level of presence, but usually a more sophisticated system is used in order to call it telepresence. Typical ways to create telepresence are cameras that follow operator's head movements, stereovision, sound feedback, force feedback and tactile sensing. To provide perfect telepresence, all human senses should be transmitted from teleoperator to operator site [8].

In [9], authors highlight benefits of controlling teleoperated devices, such as UAVs, inspired by types of interaction humans have with birds, specifically falconeering. Such approach make interacion and control far more natural. Flying Head [10] is an UAV control mechanism which syncronizes human head and robot motions. Similar to what we propose in our work, they map user's head movements into robot actions. In order to visualize the drone's view, a Sony HMD is used.

A. Data Processing

Not in all scenarios is possible to interact in a straightforward manner using telepresence. Some environments are naturally hard to perceive, like ones with low or non existing light (caves, ocean bottom, outdoor at night), bad weather conditions (raining, snowing, cloudy, misty), among others.

To solve the perception problem, some telepresence systems are built using specific sensors based on their purpose. Predator MQ-1, for example, is also equipped with an infrared camera (used in low light/night conditions) (Fig. 8).

Although costs of more sophisticated equipment are getting more accessible nowadays, they are still prohibitive for development of low cost solutions, and even using high quality sensors, there are some issues that teleoperators have to face, like points-of-interest identification, for example.

In order to overcome these difficulties, some researchers are using data processing techniques, like Computer Vision and Augmented Reality, for solving most varied problems in their teleoperation solutions.

Fig. 8. Predator's Night vision camera.

Rapid Imaging Software Inc., with collaboration of US Air force, developed SmartCam3D (SCS) [32], an Augmented Reality system that assists teleoperator to identify points-of-interest (such as specific landmarks, vehicles present in UAV area, land points, among others) (Fig. 9).

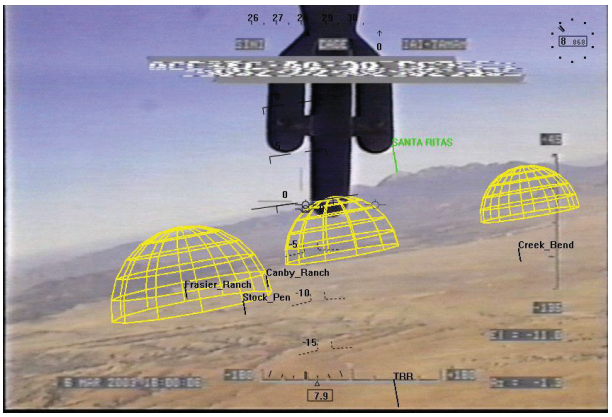

Fig. 9. SCS system identifying three points-of-interest.

Other areas than military can benefit from using data processing techniques. FlyAR [33] is an AR interface for live flight supervision and creating flight paths of a MAV (Micro Aerial Veheicles) (Fig. 10 and Fig. 11). Developed for assisting teleoperator in tasks like aerial reconstructions of buildings, FlyAR is useful to avoid possible obstacles present in environment and help to achieve desired positions for best image capture.

Fig. 10. FlyAR planned flight path.

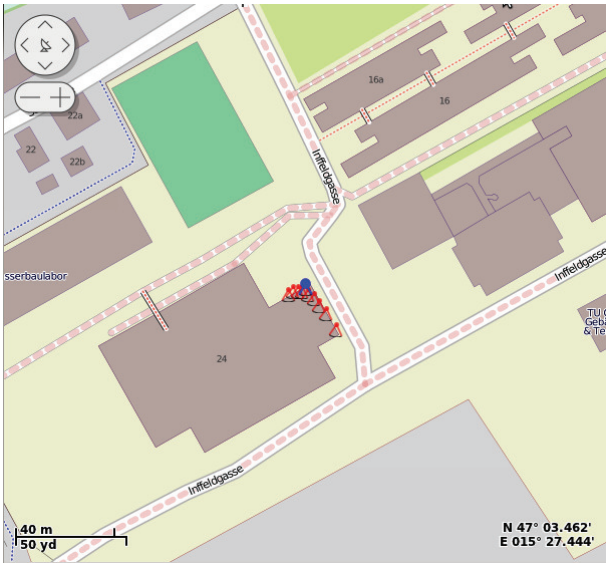

Fig. 11. FlyAR flight path up visualization.

With a similar purpose, the work described by [34] defines a teleoperation AR system using an UAV and a set of sensors, such as thermal and color cameras, temperature, humidity, dust and CO sensors, for helping teleoperator in internal and external building inspection (Fig. 12).

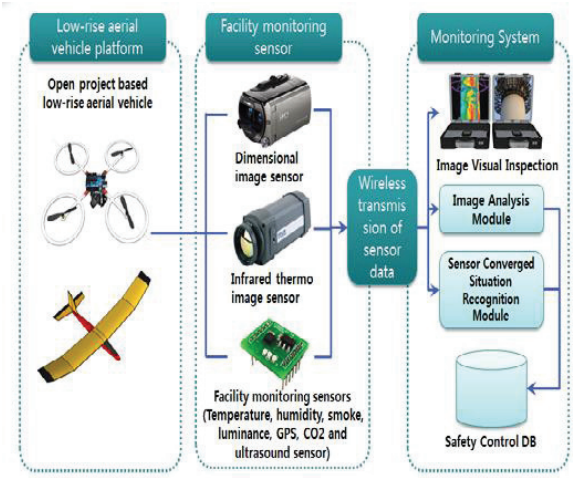

Fig. 12. Teleoperation system presented in [34].

[35] focus in using teleoperation systems in civil engineer scenarios as well. This work presents an ongoing study using an UAV equipped with a high resolution camera with stabilization control, a laser range finder and a set of radio frequency transmitters (Fig. 13). The interface used by teleoperator is heterogeneous: computers, tablets, smartphones or pilot controlling can be used (Fig. 14).

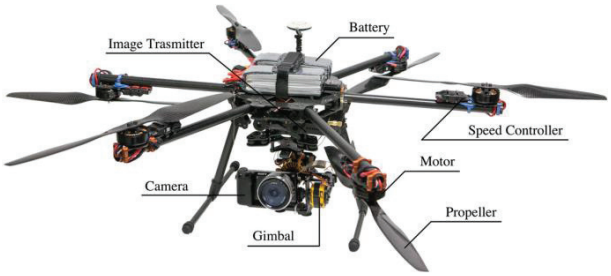

Fig. 13. UAV used by [35].

Fig. 14. Teleoperator interface proposed by [35].

The use of data processing techniques can be useful in locomotion assist in indoor scenarios, a difficult task if there are a significant number of objects obstructing ROV's path. In [36] it is presented an augmented free-viewpoint teleoperation

system interface. The approach proposed uses a robot equipped with an omnidirectional color camera and four depth sensors arranged around the robot. The teleoperador controls the robot using an HMD with head tracking and a joystick. Images received from robot and commands sent to it are intermediated by a server workstation. Augmented information is processed by server and is sent to teleoperator alongside images (Fig. 15 and Fig. 16).

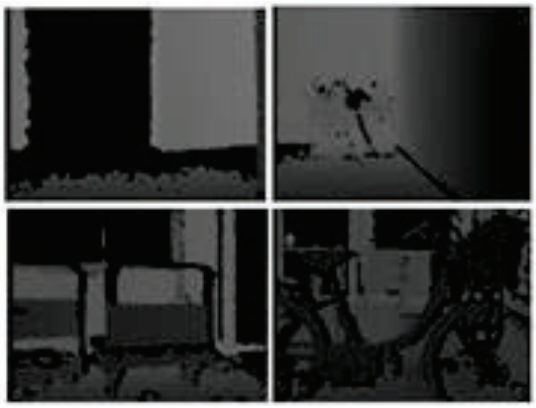

Fig. 15. Depth images (left) captured using the system proposed by [36].

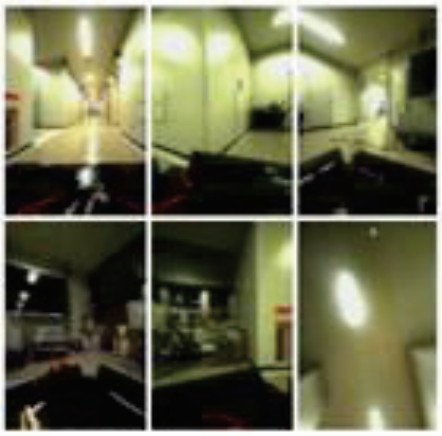

Fig. 16. Omnidirectional images captured using the system proposed by [36].

B. *Aerial and other Remotely Controlled UAVs*

[39] proposed a solution for three-dimensional localization, mapping and characterization of a tunnel environment. Aspects such as mobility, perception and localization are deeply explored. [40] proposed a hierarchical passive teleoperation control architecture that enables achieving better precision and overal task performance while controlling aerial robots. [41] proposed an intuitive multimodal haptic interface for teleoperation of aerial robots. This enables interconnecting multiple input devices, using standard interconnection rules in bond graphs. [42] proposed a system aimed at providing users with improved perception of the robot's remote environment.

III. HARDWARE INFRASTRUCTURE

Most drone applications in civil engineering focus on land surveying. In other words, an UAV flies over construction site and gives general information about current state of entire construction. From high-resolution images captured, it is also possible to construct digital elevation models to improve decision making by engineers.

In order to construct a similar system to be used for structural inspection, two main devices, common to all teleoperation applications, are needed: an operator and a teleoperator. The operator device, which is human-controlled, in this work was chosen to be the Google Glass [22], due to its compactness and naturalness of interation. As teleoperator device, Parrot's AR.Drone [23] was chosen, since it has been on the market for a while and there are a lot of different applications already developed for this device.

Troche [11] maps head movements captured using Glass to remotely operate a drone. To the best of authors knowledge, there is no work that maps Glass gestures to drone commands and also access drone camera images directly on Google Glass, enabling telepresence through vision. Our work combines different technologies to make it possible.

Data exchange between glass and drone requires wireless network communication. Since AR.Drone creates a local wireless network to which controller devices can connect, we decided to use it as communication channel. In fact, when constructing a network architecture for such teleoperation application, two possibilities come to mind, as shown in Fig. 8: directly connecting both glass and drone, or using a PC as bridge between these devices. For simplicity reasons, we adopted the second approach, because we were not able to find any working SDK capable of directly providing control and image information exchange between glass and drone.

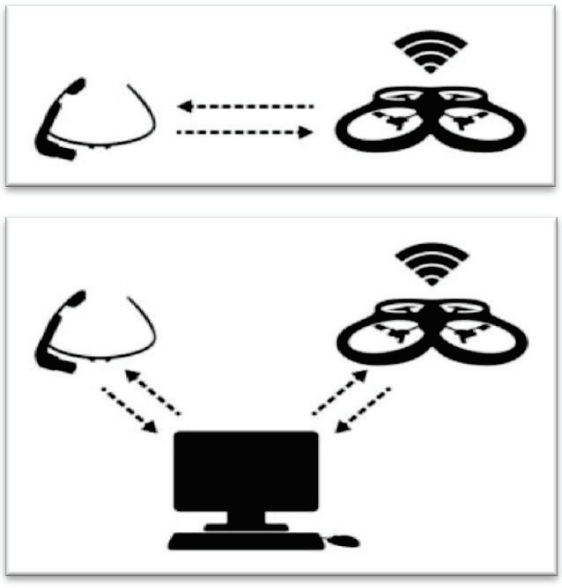

Fig. 17. Two alternatives for connecting Google Glass and AR.Drone: direct-connection (top) and through an intermediate PC (bottom).

The PC (laptop) used along with glass and drone in our tests had the following configuration: Intel Core i7-3720QM 2.6GHz CPU, 16GB of RAM, 512GB SSD HD, NVIDIA GeForce GTX 680M, running Windows 8.1 Professional.

Both AR.Drone and Google Glass will be explained in detail, and which specific features of them where used in the proposed application.

A. AR.Drone

An UAV is defined as an uninhabited motorized vehicle that can be controlled by a remote control, semi-autonomous, autonomous, or a combination of the above capabilities. It can be used for military purposes, to solve specific tasks like traffic surveillance, meteorological surveillance, carrying specific payload, solving tasks where presence of a human pilot can be unhealthy or dangerous or for general purposes such as taking a picture from a certain height from the earth or observing a target from another point of view. As compared to a manned aerial vehicle, this one has some advantages like: reduced cost, longer endurance and less risk to the crew [6]. The UAV used in this work is the Parrot AR.Drone 2.0, and its components are illustrated in Fig. 9.

Fig. 18. AR.Drone 2.0 components (http://ardrone2.parrot.com/).

AR.Drone 2.0 includes two new hardware sensors compared to its previous 1.0 version: a 3 axes magnetometer, and a pressure sensor. This magnetometer is mandatory for Absolute Control feature, which follows the direction in which the controller is tilted. The pressure sensor allows AR.Drone 2.0 to know its height regardless of the ultrasound performance (after 6 meters, ultrasound cannot measure height). It also has other sensors, such as navigation boards.

AR.Drone 2.0 uses an HD (720p - 30fps) front facing camera. This camera can be configured to stream both 360p (640x360) or 720p (1280x720) images. AR.Drone 2.0 bottom facing camera is a QVGA (320x240 - 60fps) camera. This camera pictures will be upscaled to 360p or 720p for video streaming. AR.Drone 2.0 has a master USB port, with a standard USB-A connector. This USB port is currently used for USB key video recording.

In order to develop an application capable of remotely controlling AR.Drone, one of the available SDKs must be used. There are different options, which mainly vary on features, programming language and platform. Table 1 compares some of the drone SDKs found in literature.

TABLE I. COMPARISON BETWEEN SOME AR.DRONE SDKS.

SDK	Language	Features
CVDrone [24]	C/C++	• Integrates drone operation and OpenCV • Allows full control and image access from both cameras
Javadrone [25]	Java	• Allows full control • Runs on multiple platforms • Has version compatibility problems
ARDrone.Net [26]	C#	• Allows full control and image access from both cameras
YADrone [27]	Java	• Allows full control • Image access does not work on Android platforms
EasyDrone [28]	Java	• Provides easier movement commands and complex behaviours such as face detection/face following and tag recognition

CVDrone was chosen in this project because of its integration with OpenCV [29]. This way, once an image was received from drone, it could be encoded and transmitted to Glass.

B. Google Glass

Google's most recent gadget, Glass [38], works as a 50g Android-based wearable computer. As shown in Fig. 10, it is similar to an eyewear, but with advantages of having embedded processing power and also input and output peripherals, all in same package.

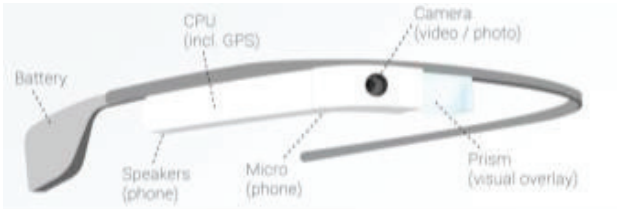

Fig. 19. Google Glass components (http://www.brille-kaufen.org). Touchpad is located outside CPU's enclosure.

For the proposed teleoperation application, the following components are used:

- *Prism (output):* to display content captured by drone;
- *Gyroscope (input):* for controlling drone rotation in a natural way;
- *Touchpad (input):* for controlling other drone commands;
- *Wireless connection (WiFi):* for data exchange between Glass and Drone.

Tests performed by authors showed that Glass battery has a maximum lifetime of one hour, when used in full load situations (recording a 720p resolution video or running computer vision algoritms). Therefore, Google Glass was not meant to be an active device. It was originally conceived to give punctual information (notifications) to user. Working that way, its battery can last a day or more.

The touchpad (located on the external part of the CPU case, on the lateral side of device) supports multitouch interaction. It is possible to identify single or double tap gestures, and also when user swipes its fingers in a specific direction.

All content on Glass can be displayed on its 640x360 display, located on the top right view field of user as a 64cm screen and 2.4m away. Since the prism is not placed right in front of user's view, it does not interfere significatively with his/her vision. Due to the fact that the prism is located slightly above user's eye of sight, creation of see-through augmented reality applications is compromised. In order to implement them, developers must first solve alignment problem between virtual screen (projected by the prism) and real world content, which means that a small portion of image captured by Glass camera matches virtual display area. This mapping may vary according to distance between Glass and real object being viewed.

Glass platform was designed so that existing Android SDK just works fine. This enables developers to code in a familiar environment, but for a uniquely novel device [30]. In addition, all of existing Android development tools can be used, and the developed software for Glass (called Glassware) is delivered as a standard Android package (APK).

Google also provides Glass Development Kit (GDK), which works as an add-on to Android SDK that lets developers build Glassware that runs directly on Glass. In order to make sure that a project is compatible with Glass platform, developer must simply set Android target to version 4.0.3 (which corresponds to API version 15). Another detail that must be taken into consideration is that application input must be mapped to Glass input peripherals, because of the fact that interaction is performed through touchpad and other sensors (camera, gyroscope, accelerometer, GPS, etc.).

IV. Experiment

This section describes how the project was designed, tools used and some decisions regarding user interaction for controlling drone through Glass.

A. Architecture

The architecture designed for the application is basically composed by two major modules: an Android client built specifically for Google Glass (implemented using Java), responsible for managing image visualization and remote control, and a PC server (implemented using C++), responsible for intermediating communication between drone and glass (Fig. 8 bottom).

Client module has three sub-modules: UI, Image Receiver and Command Sender, as shown in Fig. 11. UI module is responsible for displaying images received from Image Receiver module and for capturing interaction events (gestures and head motion), used for sending commands to Command Sender module.

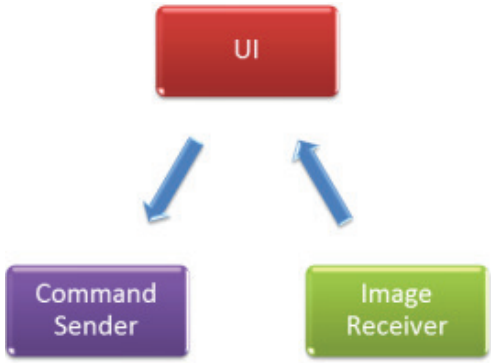

Fig. 20. Client module architecture.

Image Receiver module is responsible for receiving images captured by drone from network and routing them to UI module. Command Sender module is responsible for sending commands over network for controlling the drone, which has eight possible options: take off, land, rotate right and left, go front, up and down, and stand (used to stop current command being executed).

Server module has, analogously to Client module, three sub-modules: Drone Manager, Image Server and Command Server, as shown in Fig. 12. Drone Manager takes care of communication between PC and drone, acquiring images captured from its camera and sending to it commands for performing desired actions.

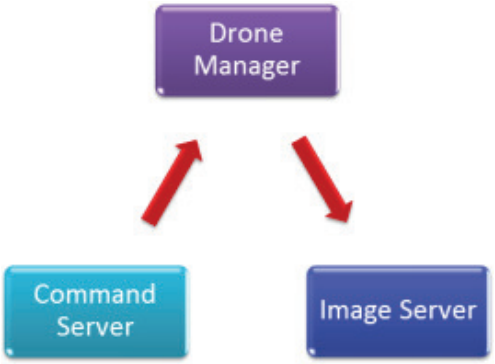

Fig. 21. Server module architecture.

Image Server is the module responsible for PC/Glass image streaming, retrieving images captured from Drone Manager, compressing them into JPEG format, and then sending data through network. Command Server is responsible for receiving commands sent from Glass and routing them to AR.Drone manager module.

B. Implementation

The two modules were built based on different technologies, communicating with each other using TCP/IP network protocol. This communication uses two distinct

approaches, one for syncing images and other for syncing commands.

Image sync approach consists in encoding the image into JPEG format, using 30% of compression quality. Server side sends compressed data and client side must decode it properly before rendering it.

Command sync approach consists in sending a single byte that represents one of nine possible actions that drone can perform. Once connection has started, client side must send commands using the same convention adopted by server. For example, client must send the byte representation of number zero (0) for server to route to drone the "take off" action.

Client module, which runs in Google Glass, was developed using Android SDK 15 (version 4.0.3). Client's UI module was implemented using an ImageView (responsible for displaying images received), a SensorManager (responsible for capturing angulation data from Glass internal compass), a GestureDetector (responsible for capturing gestures performed on Glass touchable surface) and an Activity (which holds both ImageView and SensorManager) class.

Before establishing connection to PC, Glass must know a priori its IP address, which is hardcoded on application and obtained from connection to wireless network provided by the drone. Client's Image Receiver module was implemented using a Socket that connects to server for receiving JPEG encoded images, a BitmapFactory, used for decoding data received from Socket, and a Thread that performs a continuous reading of Socket's input and uses the decoding result for updating UI's ImageView.

Client's Command Sender module was implemented using (similar to Image Receiver) a Socket that connects to server for sending UI's captured inputs (from both GestureDetector and SensorManager). Mapping between user actions and drone controls is listed in Table 2.

TABLE II. ACTION MAP BETWEEN GLASS AND DRONE.

Operator Gesture	Teleoperator Action
Head position to front	Drone stands still
Rotate head to left	Rotate drone to left
Rotate head to right	Rotate drone to right
Swipe right	Drone goes forward
Swipe left	Drone stands still
Swipe up	Drone goes up
Swipe up with two fingers	Drone goes down
Two-tap	Drone takes-off or lands

PC Server module was developed using C++ language, Asio Library [31], OpenCV [29] and CVDrone API [24]. Drone Manager was implemented using CVDrone's Ardrone object, responsible for capturing and sending data to drone, and a Thread, used for continuous drone data update.

Image Server was developed using OpenCV's *imencode* function, for encoding drone's images, a Socket that connects to client for sending encoded image, and a Thread that performs a continuous writing in Socket's output.

Finally, Command Server was developed using a Socket that connects to client for receiving commands, and a Thread, used for continuous reading and interpreting of Socket's input data, and routing them to drone via DroneManager.

Source code for the entire system (both client and server sides) is available for download and distributed as open-source in the following address: http://goo.gl/YoWA5q .

V. RESULTS

After implementation was concluded, we performed two classes of tests, related to indoor and outdoor environments. In order to capture images displayed on Google Glass, we used Droid@Screen application (http://droid-at-screen.ribomation.com/), which receives from a dedicated USB stream every image that is displayed on Glass. It is important to say that such tool is used for debugging purposes, so that image refresh rate is low (approximately 1 image every 200ms).

All tests were performed by a team of 4 people, 3 of them being project developers. Since tasks to be performed were not complex, none of the 4 testers took more than 5 minutes per test to completely execute it a single time. All four people showed similar results, which are described in sequence.

Fig. 13 shows first test realized. In this test the operator basically had to perform a two-tap (tap with two fingers) gesture on Glass touchpad and client application would send a "take off" or "land" action to the drone, according to its current state.

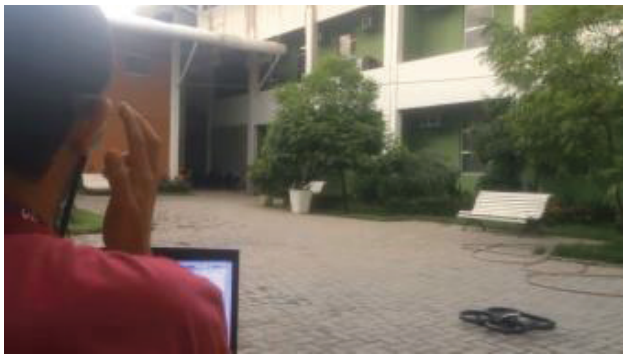

Fig. 22. Operator interacting with Google Glass (two-tap gesture).

Fig. 14 illustrates the image operator sees on Glass while controlling the drone in an indoor scenario, looking for some structural damage on a wall. Even with a JPEG compression quality of 30%, it is still possible to extract relevant information from it, such as the structural damage on wall and close to ceiling.

Fig. 23. Operator's vision while performing a structural inspection task using the AR.Drone (image captured from Glass screen using Droid@Screen application).

Fig. 15 shows another test performed, now regarding an outdoor scenario. The small image on the top right shows drone's view transmitted to Google Glass, while the rest of the image shows an outside view of the drone.

Fig. 24. Open field navigation test. Screen on top right shows image viewed on Glass and captured by AR.Drone's camera.

From tests, it was possible to validate two important aspects of any teleoperation system:

1. *tracking*: slave's ability to follow master;
2. *transparency*: whether operator feelings match real environment perception.

The first one was validated by testing all mapped commands given by head positions and gestures to control the drone. Since PC simply routes commands to drone, it was capable to execute all the given commands accordingly. Also, due to the ammount of data representing a command (a single byte), delay from command to drone action is almost imperceptible, being the same as when it is controlled from a mobile device (cellphone or tablet).

Naturalness of interaction regarding rotation was responsible for validating transparency aspect of the system. Mapping head rotation movements like a joystick not just enables operator to perform minimum head gestures (decreasing effort) but also to view different content in front of him/her as drone rotates following user's head. One insteresting detail is that sometimes operators tended to perform other head gestures, as if they were also controlling the drone. For example, when drone was tested in strong wind conditions, its stability was a little bit compromised. Sometimes operators moved their head up or down hoping that this would make drone flight more stable. We intend to evaluate these new ways of interaction in a near future and possibly change how user interacts with Glass in some situations. One possibility that is being analyzed is voice control, which is natively supported by Google Glass.

A video comprising realized tests can be found in following address: http://goo.gl/3t9OCt .

VI. Application Possibilities

One of the advantagens of having a computer serving as communication bridge between drone and glass is the fact that it can add intelligence to entire process, instead of only passing images from one side to another. This is particularly interesting since neither drone nor glass has enough processing power to support complex real time computer vision algorithms. By isolating system's "intelligence" on the bridge computer, it is possible to replace the machine with a more powerful version, without changing drone and glass parts. This section lists three possible application scenarios. First two of them regard outdoor operations, while last one focuses on a indoor task.

A. Corrosion Inspection

Engineering domain demands working with structures that must overcome bad or even awful weather conditions. But these structures are planned for a specific purpose: some for heritage, like Egyptian pyramids; other for real world problems, like cruising an entire riverbed. Independently of purpose and weather surrounding the subject, all of them are exposed to agents that promote damages, fractures and fatigue. Effect of these engineering structures corroder agents can be attenuated but not stopped and if their action is relegated, structures' reliability will be severely compromised.

Fig. 25. Corrosion inspection application scenario.

A corrosion inspection scenario consists in an interdisciplinary system in which user can navigate among several stages of dated stress on a given material or structure. Corrosion inspection is very useful, for example, for predicting building collapse due to fatigue of beams and pillars. The proposed system applies AR techniques to allow users to freely move around subject and adjust visual aspect of the model accordingly to desired date of stress, as exemplified in Fig. 25. It leads to speed up learning curve of beginners, and for experts it decreases time spent on determining how corroded a material is, since it is possible to predict its lifetime by just visually

approximating aspect of real and synthetic objects. Using AR instead of VR is due to the higher degree of users' immersion and familiarity reported in literature.

B. Civil Construction Support

Using markers for tracking a construction site presents some problems. In order to be possible to estimate pose at a given frame, a marker has to be visible on camera field of view. Since environment has a wide range, several markers have to be spread along construction site. In addition, each marker has to be calibrated with the building pose. Marker tracking can fail due to partial occlusion by any elements present on site, such as structures, tools and vehicles. It can also fail when markers appear too small on camera frame, requiring that camera cannot be too far from the tracked target or that markers with increased size have to be used, which are limiting factors. Beyond, important parts of building may be hidden by markers. They also cannot be positioned at places where there is a risk of being damaged by constant handling of construction materials. All of these cited issues suggest using a markerless tracking approach.

In civil engineering, MAR can be used to give useful information for guiding the work. Fig. 26 a building under construction augmented with graphics that indicate its future aspect in subsequent project phases, along with relevant data about how it is going to evolve. Project plan, stored in CAD plants, is exploited for both tracking and visualization purposes. It provides a 3D model of the building, which can be used by the system to perform camera pose estimation. Pose estimation makes possible for engineers to visit the building and visualize augmented information in real-time. CAD model is also source for information to be displayed over construction image.

Fig. 26. Civil construction application scenario.

C. Indoor Navigation Support

In 2011, Google Maps for Android began introducing floor plans of shopping malls, airports, and other large commercial areas. Unlike GPS, there is not a standard way of building an indoor positioning system (IPS). Google's approach tracks device via WiFi — it knows where the WiFi hotspots are in a given building, and through signal strength triangulation it can roughly work out where devices are.

Depending on location accuracy, it is possible to use it for autonomous drone navigation, besides navigation support to user that controls drone via Glass. We have evaluated an IPS SDK named Ubee [37]. An example application of its functionality is illustrated in Fig. 27. It works by triangulating signals from pre-registered hotspots and gives corresponding 2D position on map. Conclusion from tests performed shows that it has not enough precision to be used on an autonomous navigation scenario, being only suitable for approximate location on a minimap.

Fig. 27. Ubee´s store locator based on indoor positioning.

VII. Conclusion

This work proposed a teleoperation application capable of remotely controlling an AR.Drone for structural inspection through Google Glass wearable device. The advantage of using Google Glass as visualization/control device is naturalness of movements mapped to drone. Drone rotation is directly mapped to user's head relative rotation, while other commands are gesture-based using Glass touchpad. Feasibility of such an application was demonstrated along with its applicability on civil engineering area.

As explained in Experiment section, there is still possibility for improvement. For simplicity reasons, Glass/PC/AR.Drone network scheme was chosen. This decision increases network traffic, since data must be transfered twice to reach their destination (drone to PC, PC to Glass and vice versa). Because AR.Drone network is used for connecting both Glass and PC, it sometimes gets overloaded and loses image frames and also drone commands. We believe that a direct connection between Google Glass and AR.Drone should improve network communication quality.

In order to have this second version of the system up and running, we will have to build or modify an existing drone SDK for directly controlling drone on Android device (in our

case, Google Glass). As stated before, none SDKs tested provided both control and imaging for Glass platform.

REFERENCES

[1] D. Fischetti, *Structural Investigation of Historic Buildings: A Case Study Guide to Preservation Technology for Buildings, Bridges, Towers and Mills*. Wiley, 2009. [Online]. Available: http://books.google.com.br/books?id=u9Fl7Pjj9sUC

[2] E. Fisk and R. Rapp, *Introduction to Engineering Construction Inspection*. Wiley, 2004. [Online]. Available: http://books.google.sn/books?id=Q0V-QgAACAAJ

[3] C. Farrar and K. Worden, *Structural Health Monitoring: A Machine Learning Perspective*. Wiley, 2012. [Online]. Available: http://books.google.com.br/books?id=MIbcyIaMDMYC

[4] T. Sheridan, "Teleoperation, telerobotics and telepresence: A progress report," *Control Engineering Practice*, vol. 3, no. 2, pp. 205–214, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/096706619400078U

[5] J. Gundlach, *Designing Unmanned Aircraft Systems: A Comprehensive Approach*, ser. AIAA Education Series. American Institute of Aeronautics and Astronautics, 2012. [Online]. Available: http://books.google.com.br/books?id=LF LygAACAAJ

[6] P. Fahlstrom and T. Gleason, *Introduction to UAV Systems*, ser. Aerospace Series. Wiley, 2012. [Online]. Available: http://books.google.com.br/books?id=00BSVExrgRsC

[7] F. M. Brodie, *Thomas Jefferson: an intimate history*. WW Norton & Company, 1974.

[8] S. Lichiardopol, "A survey on teleoperation," Dept. Mech. Eng., Dynamics Control Group, Technische Universiteit Eindhoven, Eindhoven, Dept., Mech. Eng., Dyn. Control Group, The Netherlands, Tech. Rep. DCT2007, vol. 155, 2007.

[9] W. S. Ng and E. Sharlin, "Collocated interaction with flying robots," in *RO-MAN,* 2011 IEEE. IEEE, 2011, pp. 143–149.

[10] K. Higuchi and J. Rekimoto, "Flying head: a head motion synchronization mechanism for unmanned aerial vehicle control," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2013, pp. 2029–2038.

[11] J. Troche. (2014, Feb.) Glass quadcopter commander. [Online]. Available: https://github.com/josetroche/GlassARDroneCommanderPy

[12] R. C. Goertz, "Manipulator for slave robot," Apr. 4 1961, US Patent 2,978,118.

[13] W. Bluethmann, R. Ambrose, M. Diftler, S. Askew, E. Huber, M. Goza, F. Rehnmark, C. Lovchik, and D. Magruder, "Robonaut: A robot designed to work with humans in space," *Autonomous Robots*, vol. 14, no. 2-3, pp. 179–197, 2003.

[14] J. A. Tirpak, "The robotic air force," *Air Force Magazine*, vol. 80, no. 9, pp. 70–74, 1997.

[15] Q. Zhan, Y. Cai, and C. Yan, "Design, analysis and experiments of an omni-directional spherical robot," in *IEEE International Conference on Robotics and Automation*, pp. 4921–4926, 2011.

[16] Y. Kusuda, "How japan sees the robotics for the future: observation at the world expo 2005," *Industrial Robot: An International Journal*, vol. 33, no. 1, pp. 11–18, 2006.

[17] I. Fleet. (2014, Feb.) Victor 6000: A new robot to explore the deep ocean floors. [Online]. Available: http://flotte.ifremer.fr/fleet/Presentation-ofthe-fleet/Underwater-systems/VICTOR-6000

[18] E. Robotics. (2014, Feb.) H2000 - 2000 m work class rov. [Online]. Available: http://www.ecarobotics.com/en/robotic-vehicle/robotics-naval-rov-h2000-2000-m-work-class-rov/596.htm

[19] J. Suomela and A. Halme, "Cognitive human machine interface for a centauroid robot," in *ServiceRob Conference*, 2001.

[20] A. Morris, D. Silver, D. Ferguson, and S. Thayer, "Towards topological exploration of abandoned mines," in *IEEE International Conference on Robotics and Automation*, pp. 2117–2123, 2005.

[21] G. T. Sung and I. S. Gill, "Robotic laparoscopic surgery: a comparison of the da vinci and zeus systems," *Urology*, vol. 58, no. 6, pp. 893–898, 2001.

[22] E. Ackerman, "Google gets in your face [2013 tech to watch]," *Spectrum*, IEEE, vol. 50, no. 1, pp. 26–29, 2013.

[23] T. Krajn´ık, V. Von´asek, D. Fi˘ser, and J. Faigl, "Ar-drone as a platform for robotic research and education," in *Research and Education in Robotics*, pp. 172–186, 2011.

[24] puku0x. (2014, Feb.) Cv drone (= opencv + ar.drone). [Online]. Available: https://github.com/puku0x/cvdrone.

[25] Codeminders. (2014, Feb.) Javadrone. [Online]. Available: https://code.google.com/p/javadrone.

[26] T. Endres. (2014, Feb.) Ardrone.net. [Online]. Available: https://github.com/RobQuistNL/ARDrone-Control-.NET.

[27] D. Bade. (2014, Feb.) Yadrone: Yet another open framework for controlling the ar.drone 2.0. [Online]. Available: http://vsis-www.informatik.unihamburg. de/oldServer/teaching//projects/yadrone/

[28] P. Kenai. (2014, Feb.) Easydrone - the javadrone composer. [Online]. Available: https://kenai.com/projects/easydrone

[29] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[30] Google. (2014, Feb.) Glass development kit. [Online]. Available: https://developers.google.com/glass/develop/gdk/

[31] Boost. (2014, Feb.) Asio c++ library. [Online]. Available: http://think-async.com/

[32] Gloria L. Calhoun ; Mark H. Draper ; Michael F. Abernathy ; Michael Patzek ; Francisco Delgado. Synthetic vision system for improving unmanned aerial vehicle operator situation awareness, 2005. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=864224.

[33] Zollmann, S. ; Hoppe, C. ; Langlotz, T. ; Reitmayr, G. "FlyAR: Augmented Reality Supported Micro Aerial Vehicle Navigation" , in IEEE Transactions on Visualization and Computer Graphics, 2014, pp 560 – 568.

[34] Sung-suk Choi ; Dept. of Comput. Sci., Sunchon Nat. Univ., Sunchon, South Korea ; Eung-kon Kim. Design of Construction Stability Test System using Small Unmanned Aerial Vehicle based on Image Processing. 16th International Conference on Advanced Communication Technology (ICACT), 2014

[35] Wen, M. and Kang, S. Augmented Reality and Unmanned Aerial Vehicle Assist in Construction Management. Computing in Civil and Building Engineering (2014): pp. 1570-1577.

[36] Okura, F. Grad. Sch. of Inf. Sci., Nara Inst. of Sci. & Technol. (NAIST), Nara, Japan Ueda, Y. ; Sato, T. ; Yokoya, N. Teleoperation of Mobile Robots by Generating Augmented Free-viewpoint Images. Published in: International Conference on Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ, pp 665 – 671.

[37] Ubee (2014, Sep) Ubee. [Online] Available: https://github.com/ubee

[38] Google Glass (2014, Dec). [Online] Available: https://support.google.com/glass/

[39] J. Larson, B. Okorn, T. Pastore, D. Hooper, J. Edwards. Counter tunnel exploration, mapping, and localization with an unmanned ground vehicle. Proc. SPIE 9084, Unmanned Systems Technology XVI, June 3, 2014.

[40] A.Y. Mersha, S. Stramigioli, R. Carloni. "On Bilateral Teleoperation of Aerial Robots," Robotics, IEEE Transactions on , vol.30, no.1, pp.258,274, Feb. 2014.

[41] X. Hou, and R. Mahony. "An intuitive multimodal haptic interface for teleoperation of aerial robots." Robotics and Automation (ICRA), 2014 IEEE International Conference on. IEEE, 2014.

[42] T. J. M. Sanguino, J. M. A. Márquez, T. Carlson, J. D. R. Millán. "Improving Skills and Perception in Robot Navigation by an Augmented Virtuality Assistance System." Journal of Intelligent & Robotic Systems (2014): 1-12.