Object Detection and Spatial Coordinates Extraction Using a Monocular Camera for a Wheelchair Mounted Robotic Arm

Alessandro Palla(✉), Alessandro Frigerio, Gabriele Meoni, and Luca Fanucci

University of Pisa, Pisa, Italy
alessandro.palla@for.unipi.it, a.frigerio@studenti.unipi.it,
gabriele.meoni@ing.unipi.it, luca.fanucci@unipi.it

Abstract. In the last decades, smart power wheelchairs have being used by people with motor skill impairment in order to improve their autonomy, independence and quality of life. The most recent power wheelchairs feature many technological devices, such as laser scanners to provide automatic obstacle detection or robotic arms to perform simple operations like pick and place. However, if a motor skill impaired user was able to control a very complex robotic arm, paradoxically he would not need it. For that reason, in this paper we present an autonomous control system based on Computer Vision algorithms which allows the user to interact with buttons or elevator panels via a robotic arm in a simple and easy way. Scale-Invariant Feature Transform (SIFT) algorithm has been used to detect and track buttons. Objects detected by SIFT are mapped in a tridimensional reference system collected with Parallel and Tracking Mapping (PTAM) algorithm. Real word coordinates are obtained using a Maximum-Likelihood estimator, fusing the PTAM coordinates with distance information provided by a proximity sensor. The visual servoing algorithm has been developed in Robotic Operative System (ROS) Environment, in which the previous algorithms are implemented as different nodes. Performances have been analyzed in a test scenario, obtaining good results on the real position of the selected objects.

Keywords: Robotic arm · Power wheelchair · Visual Servoing · PBVS · Eye-in-hand · Computer Vision · SIFT · Features extraction · PTAM · ROS · Human machine interface · Assistive technology · Open-source

1 Introduction and State of the Art Presentation

Nowadays many people are affected by motor skill impairments and their number is constantly increasing. The cardinal causes are often various kinds of diseases associated with age, neurologic and muscular disorders like SMA (Spinal Muscular Atrophy), Muscular Dystrophy, Multiple Sclerosis and Duchenne Dystrophy or Cerebral Palsy, but there are many other reasons that can cause mobility impairments, for example injuries derived by a car/motorbike/work accident.

O. Gaggi et al. (Eds.): GOODTECHS 2016, LNICST 195, pp. 224–232, 2017.
DOI: 10.1007/978-3-319-61949-1_24

The possibility of moving in an autonomous way gives individuals a remarkable physical and psychological sense of well-being. Electric-powered wheelchairs are often a suitable option for those who are unable to self-propel a manual wheelchair. These wheelchairs are available on the market in a wide variety of seat widths, depths and heights, and feature a number of armrest, leg rest and seating options.

Electric-powered wheelchairs are typically controlled by a joystick, but higher-end models offer a multitude of alternative control options such as proximity switches, sip-n-puff, head arrays, infrared switches and magnetic angle sensors just to name a few. This way they can be also used by people with severe motor skill impairments. Driving a power wheelchair is still a quite difficult task for people with low vision, visual field reduction, spasticity, tremors, or cognitive deficits. In order to give also to these people a higher degree of autonomy, and to lighten the duties of those who assist them, a large number of solutions have been studied by researchers since the 1980s, by using technologies originally developed for mobile robots to create the so called smart wheelchairs.

A smart wheelchair typically is a standard powered wheelchair with the addition of a set of sensors to collect environmental data and a computer unit to process them and to find obstacles and hazards. One of the first examples of autonomous wheelchairs is proposed in [1], where a wheelchair is equipped with sonars and a vision system to identify landmarks and correct the trajectory in a hallway.

Recently, more sophisticated systems also implement robotic arms for gesture emulation, such as interacting with objects like bottles, glasses, buttons etc. On the market there are just a few examples of Wheelchair Mounted Robotic Arm (WMRA) systems, such as:

- The Manus WMRA (Fig. 1a), manufactured by Exact Dynamics. This system, developed since the mid of 80s, entered in production at the beginning of the 90s and consists in a 6 DOF arm that can be programmed in a manner comparable to industrial robotic manipulators [2].
- The Raptor WMRA (Fig. 1b), manufactured by Applied Resources. This manipulator is much simpler in respect of the Manus one. Indeed it has 4 DOF robotic arm that can be directly controlled with either a joystick or a 10-button controller [3]. Typically, the joystick that controls the manipulator arm is located on the armrest opposite to the input device that controls the steering of the power wheelchair.

Systems like the one presented in [3] require the user to manually control the arm position and move it to the desired place. Performing tasks that need high accuracy and precise gestures, such as button pressing, could be very hard for people with severe motor skill disability using this kind of control. For that reason, more autonoums solutions are usually preferred.

For example, in the work presented in [4] authors follow a very promising Eye-in-Hand approach designing a WMRA system. It consists in a 7-DoF wheelchair mounted robotic arm (WMRA) with a camera placed in the end-effector.

Fig. 1. Manus & Raptor robotic arm

The robotic arm control system uses an Image Based Visual Servoing (IBVS) approach described with a Speeded Up Robust local Features detection (SURF) algorithm in order to detect the features from the camera picture. In an IBVS system, the arm is controlled through information about the distance of the object from a desired position in the image plane, without the necessity of a pose estimation of the target.

Another example is presented in [5], in which a robotic arm is used to press a recognized buttons. In this experiment an IBVS system is implemented in a Raspberry Pi board, using a Raspberry Pi Camera module and a Touch Screen. From an user point of view, the control of the arm is very simple: the video captured by the camera is processed and showed on the screen and the user just have to select which button he wants to press with the robotic arm, then it will move autonomously to the target. Using a Linux based device and a monocular camera - implemented in mobile phones and tablets - increases the portability of the system, also leading to a low-cost, light and small solution. A simple setup like this will have although very reduced computational capability due to the performance of the involved board, which leads to limited computer vision performance.

In our work, we present a system to be used in a Position Based Visual Servoing (PBVS). The system setup is similar to [5], but a parallelization of the processing is possible thanks to the use of Robotic Operative System (ROS) environment, that allows to map different tasks into different "ROS nodes", increasing the computer vision performance and the quality of the control feedback. Differently from the IBVS approach, the PBVS controls the arm via real world coordinates, estimating the pose of the target, giving the arm control node accurate information about the position of the tracked objects.

2 System Description

Our project aims to help people with motor skill impairments to interact with the surrounding environment by performing simple gestures like pressing buttons, knocking on doors, etc. Among these tasks the most complex and computationally intensive is certainly an elevator panel interaction. In this situation the control system shall detect the buttons, track the selected one and control the arm in order to reach it. That is why in this project we focused mainly on this scenario.

The utilized hardware consists of:

- A 5 Degrees of Freedom (DOF) robotic arm
- A Raspberry Pi 3 Model B
- A Raspberry Camera Module V1.3
- A Linux Laptop
- A HCSR-04 Ultrasonic Proximity Sensor

The Raspberry Camera Module is a light camera capable of capturing video at up to 1080p at 30 fps. Thanks to its small dimensions it can be easily placed on the end effector of a robotic arm without interacting with the motor movements. The HCSR-04 proximity sensor is also connected to the Raspberry Pi 3 Board, using the GPIO port. This sensor provides a wide range of measurements, from 2 cm to 400 cm with an accuracy of 3 mm. In Fig. 2 we can see the complete setup of the Raspberry Pi module, which is connected via Wi-Fi to a Linux-based computer.

As previously described, the software architecture is developed in a ROS environment, both on the Raspberry Pi and on the Linux Workstation. The

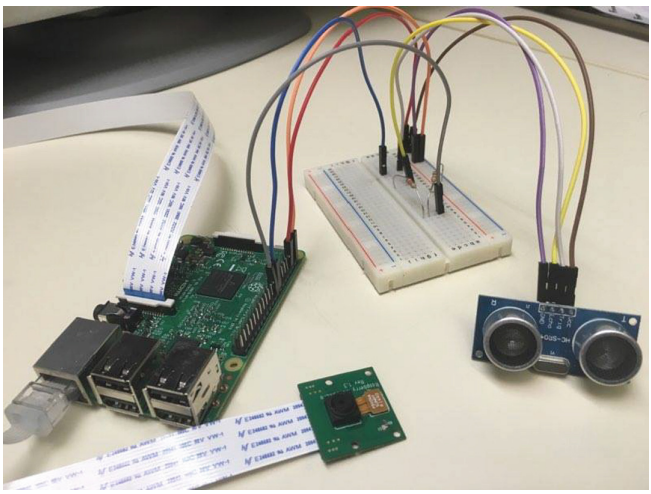

Fig. 2. The Raspberry Pi Setup, including Camera Module and HCSR-04 sensor

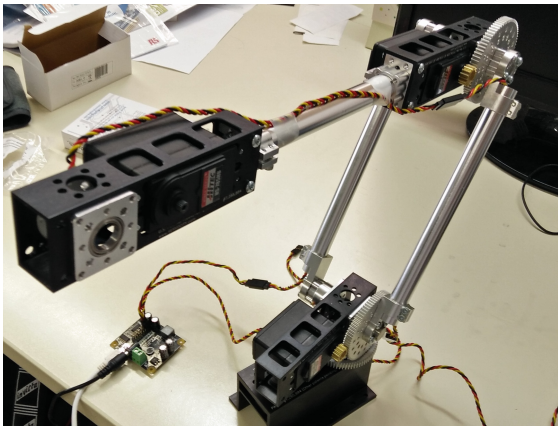

Fig. 3. The robotic arm

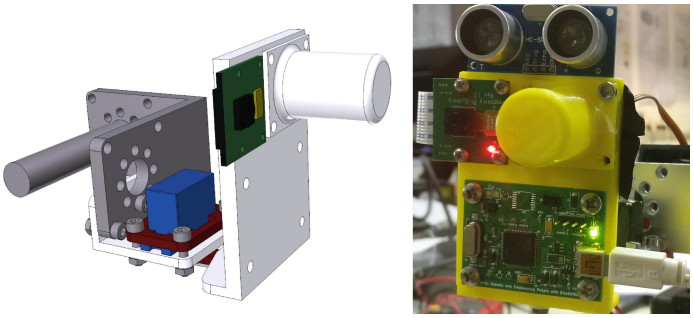

Fig. 4. End effector and camera module

Raspberry board holds two ROS nodes, one controlling the camera and publishing video frames with a resolution of 640×480, the other polling the proximity sensor, calculating and publishing the object distance. The Workstation runs the ROS master node, the Computer Vision node, the Parallel Tracking and Mapping node and the Scale Extimation node. The information obtained by this system will be sent to the ROS robotic arm controller nodes, which will handle the Inverse Kinematics and the calculation of the arm's joints positions. Figure 3 shows the robotic arm used in this project, while Fig. 4 shows the end effector on which the camera module is mounted.

3 Computer Vision

The Computer Vision task consists in elaborating frames in order to detect the buttons' position in the robot reference frame. After the user selects one of them, the system shall track the selected button, calculating its position in real world

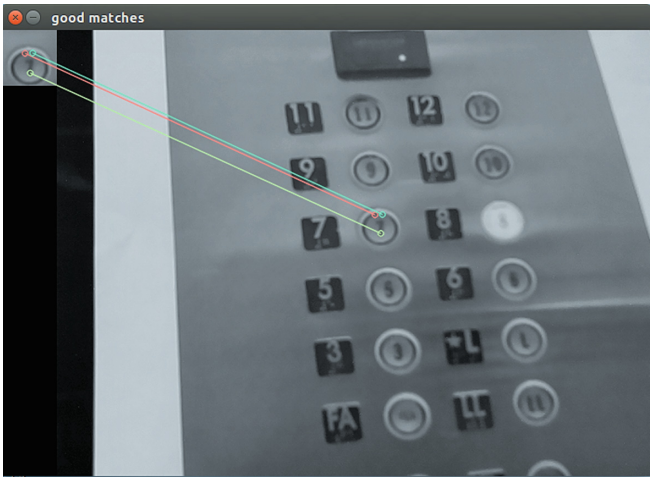

Fig. 5. SIFT extracted features matched to the ROI keypoints

coordinates, in order to provide them to the robotic arm controller in a Position Based Visual Servoing (PBVS) approach. In order to do this, we can separate the computer vision architecture into four different parts:

- Shape detection, in which buttons are detected through their shape;
- Features extraction, using the Scale-Invariant Feature Transform (SIFT) algorithm;
- Parallel Tracking and Mapping, to extract the 3D information out of a monocular camera video;
- Scale estimation and 3D reprojection of the tracked button.

The shape detection is processed via a prebuilt OpenCV function that provides circles centers and radiuses. SIFT algorithm is applied on a small Region of Interest (ROI) around the selected button. In the following frames, the Computer Vision node will extract SIFT features and match them to the ROI ones using Fast Library Approximate Nearest Neighbor (FLANN) [6], realizing the visual tracking of the button. Figure 5 shows extracted keypoints matched to the original button ones.

Parallel Tracking and Mapping [7] algorithm is used to generate a pointcloud of the environment seen by the camera, as shown in Fig. 6. However the information coming from PTAM is not sufficient to obtain real world dimensions since the pointcloud is scaled by an unknown λ scale factor. A Maximum-Likelihood Estimation method [8] is used to estimate λ, getting information from the PTAM pointcloud and the distance provided by an ultrasound proximity sensor.

In order to obtain the button 3D position we need to relate the detected button in the image plane to the pointcloud points. For that reason, it is necessary to project the pointcloud into the image plane, as shown in Fig. 7.

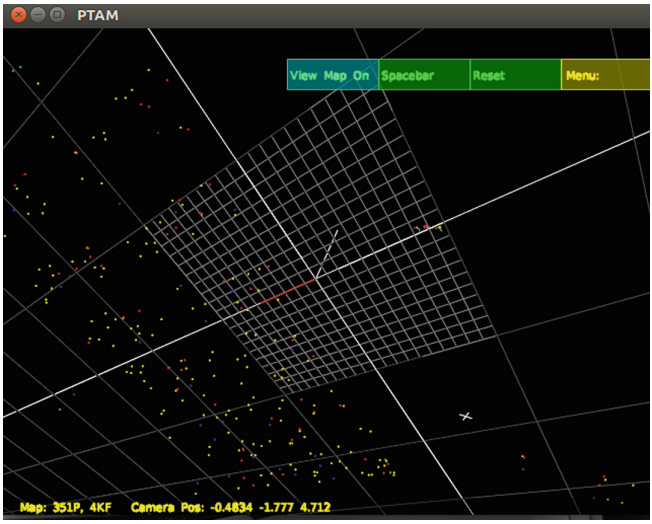

Fig. 6. Generated pointcloud by PTAM

In Eq. 1 is expressed the relationship between the two coordinate systems. The parameters f_x and f_y are the focal lengths of the camera, c_x and c_y are the optical centers. Those parameters are extracted via the OpenCV camera calibration routine.

$$\begin{cases} x_i = f_x * \dfrac{X_c}{Z_c} + c_x \\ y_i = f_y * \dfrac{Y_c}{Z_c} + c_y \end{cases} \tag{1}$$

In order to find the real world coordinates X_c, Y_c, Z_c, another equation is needed.

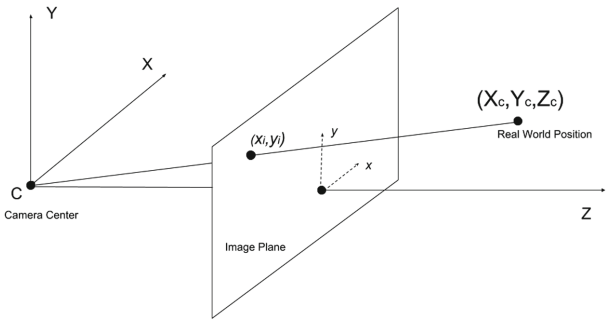

Fig. 7. Real world coordinates and image plane coordinates

The 3D points are projected to the camera plane and the points close to the target are selected. Then a plane is fitted between those points in the real world coordinate system with a Least Squares method.

The plane parameters a, b, c, d obtained from the previous step are used in Eq. 2

$$a * X_c + b * Y_c + c * Z_c + d = 0 \tag{2}$$

The real world button position is finally calculated in Eq. 3 from Eqs. 1 and 2 using the obtained plane parameters and the camera optical center and focal lengths.

$$\begin{cases} X_c = \dfrac{(x_i - c_x) * Z_c}{f_x} \\ Y_c = \dfrac{(y_i - c_y) * Z_c}{f_y} \\ Z_c = -\dfrac{d}{a * \dfrac{x_i - c_x}{f_x} + b * \dfrac{y_i - c_y}{f_y} + c} \end{cases} \tag{3}$$

4 Results

The system was tested in a controlled environment: the camera is positioned at a distance range of [18–50] cm. Figure 8 shows measurements taken on the principal camera axis. Table 1 shows the performance of the system in this situation.

Table 1. System performance on principal axis measurement

Min error	Max error	Std deviation	Mean absolute error	R^2
0.05 cm	0.95 cm	0.36 cm	0.31 cm	0.99%

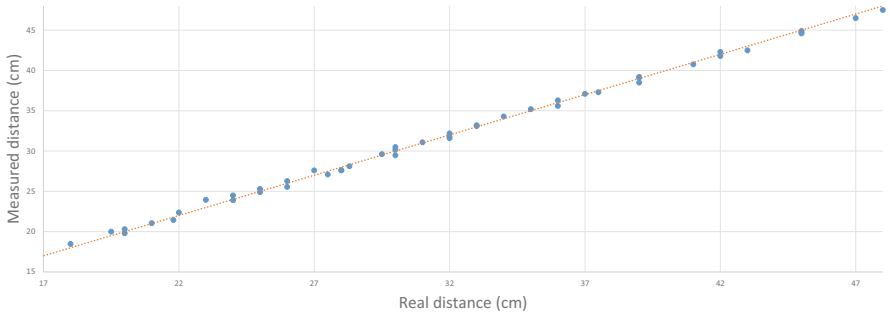

Fig. 8. Real vs. measured distance between end effector and object

5 Conclusion

In this paper we presented a system that calculates the real world coordinates of tracked objects in order to control a robotic arm with a PBVS approach.

The advantage of using a PBVS approach is to have accurate information of the 3D surroundings, allowing a simple implementation of the arm control algorithm to generate the arm movement trajectory. The use of the ROS environment leads to a modularity of the software architecture, making the implementation of new features possible without modifying the other ROS nodes. In addition, executing tasks in different nodes allows to split the calculations into different machines. Finally ROS and all the other softwares and libraries used are open source, and in combination with the Raspberry Pi and Camera Module, realize a low-cost and portable system, simplifying the algorithms implementation. In the next future the computer vision algorithm will be embedded into the robotic arm framework.

Acknowledgment. This research was supported by Fondazione Cassa di Risparmio di Lucca in the framework of the project "RIMEDIO: Il braccio Robotico Intelligente per Migliorare l'autonomia delle pErsone con DIsabilità mOtoria".

References

1. Madarasz, R.L., Heiny, L.C., Cromp, R.F., Mazur, N.M.: The design of an autonomous vehicle for the disabled. IEEE J. Robot. Autom. **2**(3), 117–126 (1986)
2. Eftring, H., Boschian, K.: Technical results from manus user trials. In: International Conference on Rehabilitation Robotics, ICORR 1999 (1999)
3. Hillman, M., Gammie, A.: The bath institute of medical engineering assistive robot. In: Proceedings of ICORR, vol. 94, pp. 211–212 (1994)
4. Elarbi-Boudihir, M., Al-Shalfan, K.A.: Eye-in-hand, eye-to-hand configuration for a WMRA control based on visual servoing. In: 2013 IEEE 11th International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), pp. 1–6. IEEE (2013)
5. Palla, A., Frigerio, A., Sarti, L., Fanucci, L.: Embedded implementation of an eye-in-hand visual servoing control for a wheelchair mounted robotic arm. In: IEEE ICTS4eHealth (2016)
6. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **36** (2014)
7. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proceedings of Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007), Nara, Japan, November 2007
8. Engel, J., Sturm, J., Cremers, D.: Scale-aware navigation of a low-cost quadrocopter with a monocular camera. Robot. Autonom. Syst. **62** (2014)