UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Guilherme Martins Pereira

Laser Pointer Driving Assistant for Robotic Wheelchairs' Navigation

Assistente de Navegação com Apontador Laser para Conduzir Cadeiras de Rodas Robotizadas

Campinas

2018

Guilherme Martins Pereira

Laser Pointer Driving Assistant for Robotic Wheelchairs' Navigation

Assistente de Navegação com Apontador Laser para Conduzir Cadeiras de Rodas Robotizadas

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia da Computação.

Orientador: Prof. Dr. Eric Rohmer

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Guilherme Martins Pereira, e orientada pelo Prof. Dr. Eric Rohmer

---

Campinas

2018

Agência(s) de fomento e nº(s) de processo(s): CAPES
ORCID: 0000-0003-4604-1927

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

P414L

Pereira, Guilherme Martins, 1989-
Laser pointer driving assistant for robotic wheelchairs' navigation / Guilherme Martins Pereira. – Campinas, SP : [s.n.], 2018.

Orientador: Eric Rohmer.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Cadeira de rodas. 2. Robôs - Sistema de controle. 3. Visão de robô. 4. Navegação de robôs móveis. 5. Visão por computadaor. I. Rohmer, Eric, 1974-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Assistente de navegação com apontador laser para conduzir cadeiras de rodas robotizadas
Palavras-chave em inglês:
Wheelchair
Robots - Control system
Robot vision
Navigation of mobile robots
Computer vision
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Eric Rohmer [Orientador]
Paula Dornhofer Paro Costa
Leonardo Rocha Olivi
Data de defesa: 07-12-2018
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Guilherme Martins Pereira RA: 163853
Data da Defesa: 07 de dezembro de 2018
Título da Tese: Laser Pointer Driving Assistant for Robotic Wheelchairs' Navigation

Prof. Dr. Eric Rohmer (Presidente, FEEC/UNICAMP)
Profa. Dra. Paula Dornhofer Paro Costa (FEEC/UNICAMP)
Prof. Dr. Leonardo Rocha Olivi (DENE/UFJF)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Abstract

Assistive robotics solutions help people to recover their lost mobility and autonomy in their daily life. This document presents a low-cost navigation assistant designed for people paralyzed from down the neck to drive a robotized wheelchair using the combination of the head's posture and facial expressions (smile and eyebrows up) to send commands to the chair. The assistant provides two navigation modes: manual and semi-autonomous. In the manual navigation, a regular webcam with the OpenFace algorithm detects the user's head orientation and facial expressions (smile, eyebrows up) to compose commands and actuate directly on the wheelchair movements (stop, go front, turn right, turn left). In the semi-autonomous, the user controls a pan-tilt laser with his/her head to point the desired destination on the ground and validates with eyebrows up command which makes the robotized wheelchair performs a rotation followed by a linear displacement to the chosen target. Although the assistant need improvements, results have shown that this solution may be a promising technology for people paralyzed from down the neck to control a robotized wheelchair.

Keywords: Computer Vision; Robotized Wheelchair; Assistive Technology; Human Machine Interface.

Resumo

As soluções de robótica assistida ajudam as pessoas a recuperar sua mobilidade e autonomia perdidas em suas vidas diárias. Este documento apresenta um assistente de navegação de baixo custo projetado para pessoas tetraplégicas para dirigir uma cadeira de rodas robotizada usando a combinação da orientação da cabeça e expressões faciais (sorriso e sobrancelhas para cima) para enviar comandos para a cadeira. O assistente fornece dois modos de navegação: manual e semi-autônomo. Na navegação manual, uma webcam normal com o algoritmo OpenFace detecta a orientação da cabeça do usuário e expressões faciais (sorriso, sobrancelhas para cima) para compor comandos e atuar diretamente nos movimentos da cadeira de rodas (parar, ir à frente, virar à direita, virar à esquerda). No modo semi-autônomo, o usuário controla um laser pan-tilt com a cabeça para apontar o destino desejado no solo e valida com o comando sobrancelhas para cima que faz com que a cadeira de rodas robotizada realize uma rotação seguida de um deslocamento linear para o alvo escolhido. Embora o assistente precise de melhorias, os resultados mostraram que essa solução pode ser uma tecnologia promissora para pessoas paralisadas do pescoço para controlar uma cadeira de rodas robotizada.

Palavras-chaves: Visão Computacional; Cadeira de Rodas Robotizada; Tecnologia Assistiva; Interface Humano Computador.

Contents

1 Introduction

According to (World Health Organization, 2011), more than one billion people in the world suffer from some form of disability, being about 200 million with considerable functional difficulties. Thus, people with severe disabilities claim for technologies to help them recover their lost mobility and autonomy in their daily lives. More specifically, people paralyzed from down the neck depend on a caregiver almost all the time in the manner that any helpful technology that could make them feel more independent would be very welcome. For this reason, this project aims at developing a novel driving assistant system for controlling a robotized wheelchair, providing a safe and hands-free solution as the users can not make use of their hands.

There are several works which present different Human Machine Interfaces (HMI) for impaired people who have suffered severe injuries to control robotized wheelchairs. The Brain Computer Interfaces (BCIs) capture electrical signals from the user's brain which may be transformed into commands for the chair. These devices are noisy and not easy to work. Also, they are not comfortable since the user must attach sensors to the head.

The eye track interfaces are equipped with a camera to capture images from the eye and estimate the region where the user is looking. The signals may be used to describe the intention of the user and actuate as commands for the wheelchair. This solution has the problem called "Midas Touch" which the interface may capture signals from non-intentional gestures of the user. Also, in some cases, the camera must be attached to a structure on the user's head which is not comfortable.

The Electromyography (EMG) based HMIs are also another uncomfortable solution for people paralyzed from down the neck to control a robotized wheelchair. The electrodes must be fixed on the user's head to capture electrical activities triggered by muscular contractions and transformed to commands for the chair. Beyond the comfort aspect, the signal is noisy, and the user may sweat and have the sensors unglued from the face skin.

The image-based HMIs may be a promising solution to compose the navigation assistant for impaired people. Some works discuss algorithms to detect facial expressions estimate the user's head posture based on images captures from regular web-cams and depth cameras. Therefore, the user may control devices and interact with other systems such as a robotized wheelchair to send commands and navigate without using his/her hands.

Thus, we present the development of the image-based HMI called *OpenFace* which

consists of a regular webcam and an image processing algorithm to estimate the user's head angles and the detection of facial expressions which can be combined to compose commands for the navigation assistant.

Also, we present the development of a pan-tilt laser subsystem installed on the robotized wheelchair to compose the semi-autonomous navigation mode. The servos are controlled by the user's head orientation, which points the laser dot on the ground and with a facial expression he/she validates the intention to reach the destination, and the wheelchair performs the dislocation until the chosen target.

2 Literature Review

Nowadays more than 700 million persons around the world have some disability or handicap, and during the last years, assistive robotics has been proposing several solutions to allow a more independent life to these people (FARIA *et al.*, 2014). Power wheelchairs operated by a joystick and a set of buttons have improved the mobility for several users. However, those with the most severe disabilities (e.g. people who suffered high cervical spinal cord injuries (SCI)) can not benefit from this technology. In this context, robotized wheelchairs have been developed mainly to provide navigation capabilities and appropriate interfaces according to the users' necessities (COWAN *et al.*, 2012).

Several works investigate a variety of hands-free devices to interact with wheelchairs, but especially Brain Computer Interfaces (BCI) which have the capacity of monitor the users' brain activity and transform their intentions into commands without activating any muscle or peripheral nerve (MILLÁN *et al.*, 2010). In (ITURRATE *et al.*, 2009), they have implemented a P300 neurophysiological protocol and automated navigation which the user concentrates on the location he wants to go by visualising a screen that displays a real-time construction of the scenario. The visual stimulation with the reachable spots elicits the neurological phenomenon, and the electroencephalogram (EEG) signal processing detects the target location. The prototype was validated with five healthy subjects, and the results have shown that all participants were able to complete the driving sessions in scenarios with obstacles successfully.

In (LONG *et al.*, 2012), the authors report a hybrid BCI that combines the detection of two patterns of electrical activities involving neurones (mu/beta rhythm resulting from motor imagery and the P300 potential) to control a simulated and a real wheelchair. The user can perform a left or right-hand motor imagery to actuate in the direction of the wheelchair (left or right turn). The speed of the wheelchair is controlled when the user performs foot imagery (to decelerate) or focus on a flashing button of the graphical user interface (GUI) to accelerate. They have conducted experiments with five subjects controlling both real and simulated wheelchairs. According to the authors, the results have demonstrated the effectiveness of their system and method, and the combination of multiple signal modalities (motor imagery and P300 potentials) is useful not only to improve accuracy but also to provide multiple independent control signals.

Eye-based interfaces are another kind of solution investigated in the context of assistive technologies to support people with severe motor impairments. In (TOMARI *et al.*, 2013),

they propose a wheelchair framework which provides for the user a multi-input hands-free interface (HFI) to send commands to a robotized wheelchair. The system uses a multimodal control strategy combining both the user and the computer actuation. The HFI consists of a regular webcam that gets the natural gaze to instruct the directions of the wheelchair movements (imitating a joystick control) and a simple like-bite switch button to stop the chair or change the navigation mode (semi-autonomous/manual). When the user assumes manual control, the computer combined with information from the environment (laser and Kinect for obstacle detection) assists this by overriding any command that may cause a collision.

In (GAUTAM *et al.*, 2014) they developed an optical-type eye tracking system to control a power wheelchair and perform a manual navigation. The system consists of a wearable glasses with a fixed small camera pointed to one of the user's eye which captures a sequence of iris images and uses the gaze directions as commands to actuate on the wheelchair. The software uses Daugman's algorithm to segment the pupil and deducts the direction which the user is looking at. When he/she moves his eyes balls up the wheelchair moves forward, left moves it left, right moves it right, and in all other cases, the wheelchair stops.

Another type of assistive HMI is the Tongue Drive System (TDS). In (KIM *et al.*, 2013) they investigate the efficacy of a TDS versus a keypad and TDS versus a sip-and-puff device (SnP), for controlling a power wheelchair and a computer. The tests were applied to a group of volunteers with spinal cord injury (SCI) at C6 or above, and two groups of healthy subjects. The results have shown that TDS demonstrated similar accuracy with the traditional assistive technologies, but the TDS performance measures were up to three times better, even for the subject with SCI (who were familiar with SnP device). The characteristics of the human tongue associated with TDS enabled individuals who suffered severe injuries to access computers and drive wheelchairs at speeds faster than traditional assistive technologies.

(HUO; GHOVANLOO, 2010) also presents a wireless tongue–computer interface (TCI), unobtrusive and minimally invasive to control a powered wheelchair. They tested the prototype with 13 naive subjects with high-level spinal cord injuries (C2–C5), and the results have shown that the solution can potentially provide its end users effective control over both computers and wheelchairs. Furthermore, the authors claim that the TDS in its current form is twice as fast as some EEG-based BCIs that are tested on trained humans (WOLPAW *et al.*, 2002).

In (ESCOBEDO *et al.*, 2013) they describe a method to perform semi-autonomous navigation on a wheelchair by estimating the user's intention using a face pose recognition system (Kinect camera to interact with the user). The estimator algorithm uses Bayesian

network approach, and the experimental wheelchair is equipped with a second Kinect camera to sense the environment and a Laser Range Finder (LRF) Lidar LMS-200 to provide a safe navigation with path planing and obstacle avoidance while the user is just concerned with "looking where he wants to go".

In (ROHMER *et al.*, 2015b), the authors implemented a framework to allow an operator with disability to interact with a smart environment using hands-free devices (small movements of the face or limbs through Electromyograph (EMG), or Electroencephalograph (EEG), among others). The work also details the integration and testing of four control strategies (manual control, shared control, point to go, and fully autonomous) for assistive robotic vehicles.

In this context, this project aims at implementing the proof of concept investigated in (ROHMER *et al.*, 2015a) which proposes a laser-based driving assistance system for robotic wheelchairs. The system consists of an Inertial Measurement Unit (IMU) placed above the user's head to captures its posture, and pan-tilt laser fixed on the chair which points on the ground and represents the destination he/she wants to reach autonomously. The wheelchair is also equipped with a low-cost depth-camera (Kinect sensor) that models a traversability map to define if the pointed destination is reachable or not. If reachable, the red laser dot turns green, and the operator can validate the target with an Electromyogram (EMG) device attached to his face, which records facial muscular contraction. After confirming the desired destination, the algorithm uses the traversability map to calculate the path considering the obstacles and the wheelchair performs the trajectory autonomously. They have implemented and tested the whole solution in V-REP (ROHMER *et al.*, 2013), and this technique may be a possible mean of navigation for people paralysed from down the neck.

3 Investigation of Hands-free HMIs

This chapter discusses some hands-free HMIs which are able to detect signals fro the head such as its orientation and/or facial gestures which could be used to compose the navigation assistant presented in the previous chapter. We discuss EMG based HMIs, the comercial solution Emotiv EPOC and image based interfaces JoyFace and RealSense.

3.1 Electromyography Based HMIs

Electromyography (EMG) is a bio-signal which is applied in various fields of study such as motor control, neuromuscular physiology, movement disorders, postural control, human-machine/robot interaction and so on. Although the main works propose the use of EMG sensors for prosthetic hand control applications (KYRANOU *et al.*, 2016), recent works present the recognition of different facial gestures using facial neuromuscular activities for HMI applications.

According to (HAMEDI *et al.*, 2013), facial electromyograms analysis is a complicated field in biomedical signal processing where accuracy and low computational cost are significant concerns. They proposed a facial gesture recognition-based interface by recording EMG signals and applying a very fast versatile elliptic basis function neural network (VEBFNN) to classify different facial gestures. The EMGs were recorded through three channels via three pairs of surface rounded pre-gelled Ag/AgCl electrodes.

In (HAMEDI *et al.*, 2012), the authors compare the classification performances by using different well-known feature extraction methods on facial EMGs. Six time-domain feature extraction methods have been evaluated, and Root Mean Square (RMS) gives the best performance in estimating the value of facial EMGs.

Although some works discuss the viability of using EMG sensors to compose HMI applications in assistive technologies, we can point some disadvantages of this kind of solution. According to (REAZ *et al.*, 2006), EMGs are known to be one of the most contaminated signals with a low signal to noise ratio. Thus, to achieve clear EMGs, its use requires some precautions such as clean the user's skin before fixing the electrodes and position them correctly on the face.

EMG based HMIs are wired solutions which require in most of the times dedicated and specific hardware to read and process the signals. Moreover, in most cases, the detection of different patterns requires several common processing stages which are not simple:

noise reduction, conditioning, smoothing, data windowing, segmentation, feature extraction, dimension reduction, and classification.

Finally, EMG based HMIs are not comfortable neither practicable to compose our navigation assistant. The pre-gelled electrodes probably will not be fixed for a long time since the user sweats and will not be able to send commands using facial expressions.

3.2 Emotiv - Brain Computer Interface

The Emotiv EPOC is an Electroencephalography (EEG) device released in December 2009 intended for playing computer games for a price of US$299. Although the EPOC model is not classified as a medical device, it provides useful features which could potentially help people with severe disabilities access assistive technology (AT). The device is composed of 14 electrodes which monitor the electrical activities of the brain to detect thoughts, facial expressions, and emotions. Also, the device counts with a gyroscope sensor to capture the user's head orientation.

In (LIEVESLEY *et al.*, 2011), two experiments show that the Emotiv EPOC neuroheadset can be used as an interface for non-impaired users to transfer information to a computer, which could, in turn, be used to control assistive technologies. In the first, 12 non-impaired subjects used the neuroheadset to control a computer with their facial expressions. In the second experiment, three non-impaired subjects were trained to use the neuroheadset to control a computer with their thoughts.

In (RECHY-RAMIREZ *et al.*, 2012), they present a Human Machine Interface for hands-free control of an electric powered wheelchair using the Emotiv EPOC device. Although the device provides facial expressions detection, the two navigation modes consider only the gyroscope signals to estimate the user's head orientation and actuate on the chair. In mode 1 a screen displays for one second each of the four possible wheelchair movements (go forward, turn right, turn left, stop) and the system uses only one head movement (head up or head down) to select the command. It has been designed mainly for patients with limited head movements. In mode 2 the user performs a specific head movement, e.g. Up, Right Left and Down, to issue one of the four control commands.

The Emotiv EPOC may be an option for our hands-free HMI to control our robotized wheelchair. The device is not subject to light variations as its sensors rely mainly on EEG and gyroscope signals which is an advantage compared to image-based technologies. However, this solution also has some disadvantages which may compromise the user experience in daily usage. Firstly, it requires the user to fix the neuroheadset on his/her head which may be

Figure 1 – Emotiv EPOC device with sensors to estimate the head orientation and detect facial expressions, thoughts and emotions. Adapted figure.[1]

uncomfortable. Secondly, although there are not sticky gels (as in EMG based solutions), the saline-based electrodes must be wet to provide a high-quality signal for the three available suites ('coginitiv', 'expressiv' and 'affectiv') of the development kit.

3.3 Image-based HMIs

Image-based HMIs may be promising solutions for assistive technologies. Although they depend on controlled light conditions, one of the main motivations is the comfort aspect since it does not require the user to attach any sensor on his/her head. Thus, this subsection presents the investigation of JoyFace and RealSense HMIs.

3.3.1 Intel RealSense Technology

The Intel RealSense Technology [2] provides different depth cameras and development kits which can be used for several applications using depth data. Although, this model is discontinued, the Intel RealSense F200 camera depicted in Figure 2 consists of a color sensor with full 1080p RGB resolution and an Intel-developed depth camera to capture the reflected rays generated by the IR laser projector. The manufacturer provides a Software Development Kit (SDK) with several algorithms of hand/finger tracking, facial analysis, speech recognition, augmented reality, background segmentation and more others possible to integrate apps. Although the SDK was developed in C++, the wrapper layer exposes its interface in a variety of programming languages i.e. C#, Processing, Java, Unity etc.

One of the samples of the SDK is the Face Tracking algorithm, which can be used to detected facial expression and integrate other applications. The algorithm locates the face in a rectangle and further identifies the feature points (eyes, mouth, etc.) with which the algorithm calculates the scores for a few supported facial expressions such as eye closed and

[1] https://www.emotiv.com/epoc/
[2] https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html

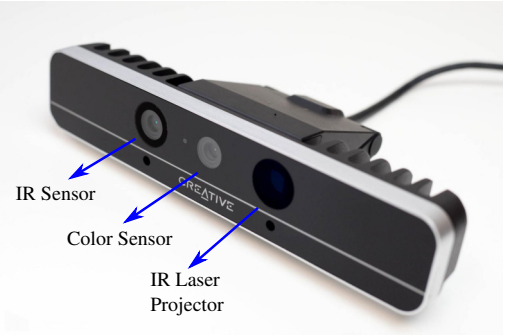


Figure 2 – *Intel RealSense F200* camera.

eyebrow turning up. Figure 3 illustrates some screenshots of the application detecting four facial expressions which could be used as commands to our navigation assistant: *kiss*, *mouth open*, *smile* or *eyebrows up*.

Figure 3 – *Intel RealSense F200* camera running with SDK sample to get the facial expressions.

3.3.2 JoyFace

We have investigated a second image-based HMI called JoyFace (MOTA, 2017) illustrated in Figure 4. It consists of a regular webcam and a computer vision algorithm to track the face with a squared landmark, and by positioning the nose out of its centroid reference we can send commands *up*, *right*, *left* and *down* to integrate other systems. Moreover, it can detect a smile as an additional command. Furthermore, the algorithm does not depend on a specific sensor (such as in Intel RealSense Solutions) which makes it cheaper and easier for research purposes.

3.3.3 JoyFace and RealSense Comparison for Wheelchair Navigation

In (PEREIRA *et al.*, 2017), the authors compare the RealSense solution and JoyFace used as HMIs to control a robotized wheelchair using facial expressions and head displacement. Figure 5 depicts the system overview which consists of a robotized wheelchair and one of the HMIs (RealSense or JoyFace).

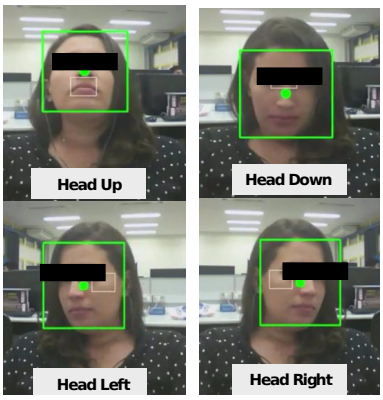


Figure 4 – *JoyFace* HMI using a regular webcam to get head movements (MOTA, 2017).

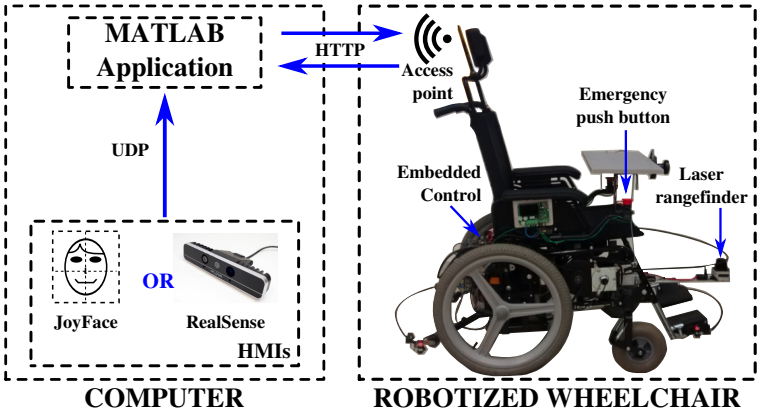


Figure 5 – System overview. The operator controls the wheelchair using one of the HMIs: JoyFace or RealSense. The commands are sent via UDP to a Matlab application which filters the messages and actuates on the wheelchair by sending HTTP requisitions using the REST paradigm (PEREIRA *et al.*, 2017).

The volunteers were asked to test one HMI at a time which captures either head movements (JoyFace) or facial expressions (RealSense), each one with four possible commands to actuate on the wheelchair. The interface application sends the captured commands via UDP socket to a Matlab application which filters the received messages and actuates on the chair using a Representational State Transfer (RESTful) architecture (SOUZA *et al.*, 2013).

The Matlab Application implements a fast prototype of the high-level navigation algorithm which handles two threads: one to listen to the received messages on the UDP socket and another to filter the valid commands and actuate on the chair accordingly. Table 1 shows how the fours commands of JoyFace and RealSense are associated with the high-level commands of the wheelchair. Go front applies a fixed linear velocity of 150 mm/s; Turn left increments the rotational velocity (counterclockwise is positive) and Turn right decrements it, being that its absolute value is limited to 10 deg/s. Although the operator can use commands head down, and smile to stop the wheelchair, for security reasons the system is equipped with emergency stop buttons that can be pressed by the operator as well as back and front limit

switches fixed on the chair (activated with contact) to avoid collisions.

Wheelchair movements	***JoyFace***	***RealSense***
Go front	*head up*	*kiss*
Turn right	*head right*	*eyebrows up*
Turn left	*head left*	*mouth open*
Stop	*head down*	*smile*

Table 1 – *JoyFace* and *RealSense* commands associated with high-level commands to control the wheelchair.

Both JoyFace and RealSense HMIs were tested with ten healthy volunteers, being 9 men and 1 woman who have experimented the interfaces to control the robotized wheelchair on the same day and perform a navigation in a predefined route with obstacles illustrated in Figure 6.

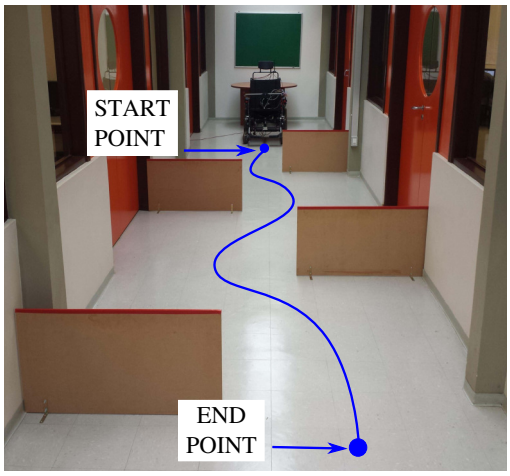


Figure 6 – Corridor with obstacles.

At the end of the experiment, the volunteers were asked to answer a questionnaire which was composed of the following questions regarding their experience with JoyFace and RealSense:

- Which HMI has demanded more mental efforts?
- Which HMI has demanded more physical efforts?
- Which HMI has offered more security?
- Which HMI was easier to use?

The questionary was inspired by the NASA Task Load Index (NASA-TLX) methodology (NASA, 2011) to evaluate the HMIs effectiveness and performance.

Table 2 shows the subjects' impressions comparing JoyFace and RealSense regarding the fours aspects of the questionnaire. Most of them pointed JoyFace to be the safest, easiest to use and the one which required less mental demand. In contrast, the majority of the subjects have answered that RealSense requires less physical demand.

	JoyFace	***RealSense***
Mental demand	20%	80%
Physical demand	60%	40%
Security	60%	40%
Usability	70%	30%

Table 2 – Comparison between *JoyFace* and *RealSense* HMIs.

The results in Table 3 compares JoyFace and RealSense, considering the lap times for each subject to perform the trajectory and how many times the wheelchair has stopped during the navigation. The best laps are marked in bold, and although we had 6/10 best lap times with JoyFace, we cannot conclude that this is indeed the fastest HMI for wheelchair navigation. The time averages for both are quite similar, but the higher number of stops with RealSense indicates that the operator had problems during the tests. The wheelchair has stopped mostly because RealSense has failed several times on recognizing facial expressions, so the subjects had to push the emergency button to avoid collisions. Other causes of the stops were panic or even lack of training to control the chair. The subjects have complained that RealSense was confusing mouth open with smile commands. Furthermore, the eyebrows up has failed more often with wearers of glasses, which is another problem of the algorithm.

Subject	***JoyFace***	**No. Stops**	***RealSense***	**No. Stops**
1	**01:37**	0	03:25	3
2	03:17	1	**03:06**	0
3	**01:44**	0	06:07	2
4	04:17	4	**03:58**	4
5	**01:34**	0	02:35	1
6	**02:07**	0	05:10	0
7	04:56	0	**02:50**	0
8	**02:03**	0	04:21	2
9	03:42	0	**02:47**	0
10	**01:59**	0	03:30	3
Avg.	**02:43**	**0,5**	**03:46**	**1,5**

Table 3 – Lap times (in minutes) and number of emergency stops the subjects had to take due to collisions, imminence of collisions, or even panic during the navigation with *JoyFace* and *RealSense* HMIs.

Results have shown that both HMIs are comfortable as users does not need any sensors attatched to their face. The base RealSense SDK algorithm needs some improvements on the classification algorithm and a calibration feature should be developed to work properly with different subjects. Compared to JoyFace, the RealSense is not low-cost and depends on specific hardware to work.

3.4 Chapter Conclusions

In this chapter the discussed different possible hands-free HMIs to compose the wheelchair navigation assistant for disabled people. The first one was EMG which is not viable since it needs sensors attached to the user's which is uncomfortable and may cause several difficulties such as misclassification problems due to noisy signals or badly fixed sensors on the skin. The Emotiv EPOC has similar problems to the EMG based HMIs such as being uncomfortable and thus not viable for daily use.

Then, we presented JoyFace and RealSense image-based HMIs and an experiment to compare both solutions. Most of the subject succeeded on navigation in a corridor with obstacles by commanding the wheelchair using facial expressions (RealSense) or the head orientation (JoyFace). Since there are no sensors attached to the user's face image-based, HMIs are promising technologies to compose our navigation assistant.

4 OpenFace

4.1 Introduction

The *OpenFace* Framework [1] is an open source facial behavior analysis toolkit intended for computer vision and machine learning researchers, affective computing community and people interested in building interactive applications based on facial behavior analysis. *OpenFace* is the first toolkit capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation with available source code for both running and training the models. Furthermore, this tool is capable of real-time performance and can run from a simple webcam without any specialist hardware.

This chapter discusses the exploration of the *OpenFace* Framework and our contributions implement a low-cost HMI which detects facial expressions and head's posture to integrate the navigation assistant.

4.2 Methodology

This section discusses the steps from the initial investigation of the original *OpenFace* Framework until our implementation to create a viable HMI to control the wheelchair and integrate the navigation assistant. The *OpenFace* contributions have divided in some functionalities to compose reliable module to integrate our navigation system: detection of facial expressions; calibration mode to save/load parameters to run the application; UDP connection to send the detections and integrate with our navigation assistant.

4.2.1 FACS

Facial Action Coding System (FACS) is a system to taxonomize human facial movements by their appearance on the face. It was based on a system initially developed by a Swedish anatomist named Carl-Herman Hjortsjö and later adopted by Paul Ekman and Wallace V. Friesen (EKMAN; FRIESEN, 1978b). The system introduces the concept of Action Units (AUs) as the fundamental actions of individual muscles or groups of muscles which can be combined to produce facial expression. Thus, as the AUs are independent of any interpretation, they are useful for manual creation of facial expressions or even recognition of basic emotions.

[1] https://github.com/TadasBaltrusaitis/OpenFace

The FACS provides a complete index of facial expressions along with their respective AU codes and names. The *OpenFace* Framework is able to recognize a subset of AUs, specifically: *1, 2, 4, 5, 6, 7, 9, 10, 12, 14, 15, 17, 20, 23, 25, 26, 28*, and *45*. They are depicted in Figure 7 and described in Table 4 which points the muscular basis and description for each AU. The AU45 is not illustrated in Figure 7, but it corresponds to the *blink* of both eyes.

Figure 7 – Action units detected by the *OpenFace* Framework. Adapted from (EKMAN; FRIESEN, 1978a)

AU Code	Description	Muscular basis
01	Inner Brow Raiser	*Frontalis, pars medialis*
02	Outer Brow Raiser	*Frontalis, pars lateralis*
04	Brow Lowerer	*Corrugator supercilii, Depressor supercilii*
05	Upper Lid Raiser	*Levator palpebrae superioris*
06	Cheek Raiser	*Orbicularis oculi, pars orbitalis*
07	Lid Tightener	*Orbicularis oculi, pars palpebralis*
09	Nose Wrinkler	*Levator labii superioris alaquae nasi*
10	Upper Lip Raiser	*Levator labii superioris*
12	Lip Corner Puller	*Zygomaticus major*
14	Dimpler	*Buccinator*
15	Lip Corner Depressor	*Depressor anguli oris (a.k.a. Triangularis)*
17	Chin Raiser	*Mentalis*
20	Lip Stretcher	*Risorius w/ platysma*
23	Lip Tightener	*Orbicularis oris*
25	Lips part	*Depressor labii inferioris or relaxation of Mentalis*
26	Jaw Drop	*Masseter, relaxed Temporalis and internal Pterygoid*
28	Lip Suck	*Orbicularis oris*
45	Blink	*Relaxation of Levator palpebrae superioris*

Table 4 – Action units detected by the *OpenFace* Framework. The muscular basis corresponds to the main muscles of the face involved in each facial expression. Adapted from (EKMAN; FRIESEN, 1978a)

OpenFace can extract AUs from a single image, sequences of images or videos. The AUs are described in two ways: presence and intensity. The presence score assumes binary values: *0* if the AU is not present or *1* if present. The intensity score is represented on a 5

point scale and can assume *0* value (not present) or vary from *1* to *5* which are the presence at the minimum intensity and maximum intensity, respectively.

4.2.2 Signal Processing

Although *OpenFace* provides both intensity and presence signals for 18 AUs, we only needed few of them to compose our facial expression detector for the navigation assistant. Therefore, we firstly adapted the Framework to export all the AUs intensities and timestamps to an output file so we could plot the signals and make a visual analysis. Furthermore, we could apply transformations on the signals to enhance possible features and identify the most significant AUs for each kind of facial expression.

The *OpenFace* algorithm has been modified to calculate two additional signals for each AU: simple moving average (SMA) of the intensity (filtering step) and the derivative of the SMA. Equation 4.1 calculates the SMA (denoted by $\overline{AU}_{SMA}$) to get a more clean shape of the intensity signal:

$$\overline{AU}_{SMA} = \sum_{i=0}^{n-1} \frac{AU_{M-i}}{n} \tag{4.1}$$

Where: AU_M $(i = 0)$ is the actual intensity of the Action Unit; AU_{M-i} $(i \neq 0)$ represents a previous intensity value; and n is the number of intensity values considered on the SMA calculation. In our case we have assumed $n = 6$.

The derivative transformation provides a third signal with more notable representations for the rises and falls of the filtered intensities when a facial expressions is emitted. Equation 4.2 calculates the derivative of the SMA (denoted by AUd) which considers variation of $\overline{AU}_{SMA}$ over time.

$$AUd = \frac{\Delta \overline{AU}_{SMA}}{\Delta t} \tag{4.2}$$

Figure 8 depicts the AU14 intensity and both calculated signals, based on recorded data from a simple test which the user has emitted five sequential *smiles*. Although the filtered signal has a smaller module than the raw intensity, the filtered signal is smoother and provides better features to be identified. One should note that the calculation of the SMA signal introduces a delay of ≈ 300 ms with respect to the intensity signal. However, as the robotized wheelchair has a low response time, the delay does not affect the effectiveness of the navigation assistant.

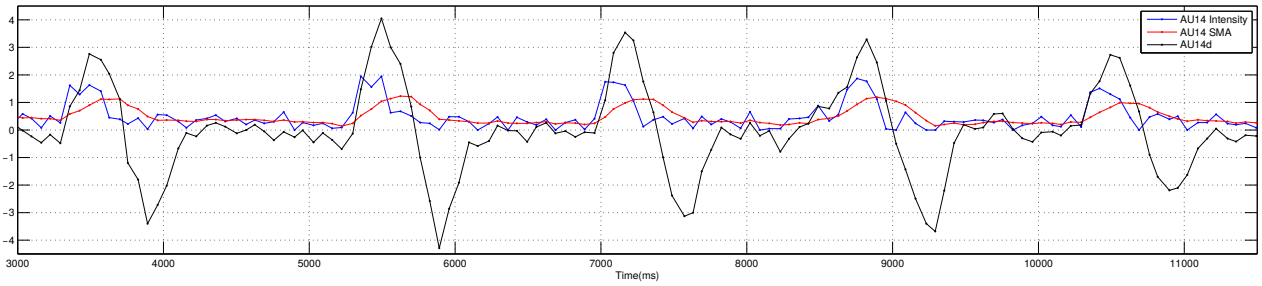


Figure 8 – AU06 Intensity and calculated SMA ans derivative signals for 5 *smiles*.

Based on the SMA and derivative signals, we have tested the *OpenFace* with *smile, eyebrows up, blink* and *kiss* (individually) to collect all AUs signals for each situation and evaluate the viability of implementing the detection of these facial expressions. For each facial expression, we have determined the most relevant AUs considering the following criteria:

- SMA and derivative signals visually well defined when the facial expression is emitted;
- Derivative signal exceeding *3* and *-3* points on scale for max and min values, respectively;
- Check if AU activation caused by the facial expression makes sense with FACS.

Figure 9 depicts the SMA and derivative signals for all the 18 AUs when a user makes 5 sequential *smiles*. One should note that AU06, AU10, AU12 and AU14 are the most relevant channels compared to the others. The signals are well shaped and the derivatives attend to the min and max acceptance criteria. Some values of the derivative peaks (max and min): AU06d(5.21, -3.80), AU10d(4.79, -4.23), AU12d(4.46, -3.77), AU14d(6.00, -5.90).

We repeated the same procedure for *eyebrows up, blink* and *kiss* and the most relevant AU signals are illustrated in Figures 10, 11 and 12. For the *eyebrows upd* the derivative peaks were: AU01d(6.27, -6.27) and AU02d(5.37, -5.38). For the *blink*, only AU45 has been activated properly with values: AU45d(3.17, -3.11). Finally, the *kiss* did not have any AU with the derivative signal attending the acceptance criteria. Although the AU23 depicted in Figure 12 had the best signals compared to the others AUs (very noisy), the SMA and derivative variations was not enough significative to compose our HMI.

The SMA of the AUs and their calculated derivatives have indicated that *eyebrows up, smile* and *blink* are the 3 possible facial expressions to compose the face image based HMI for our navigation assistant. The next subsection discusses the algorithm strategy to detect the facial expressions in our real-time application.

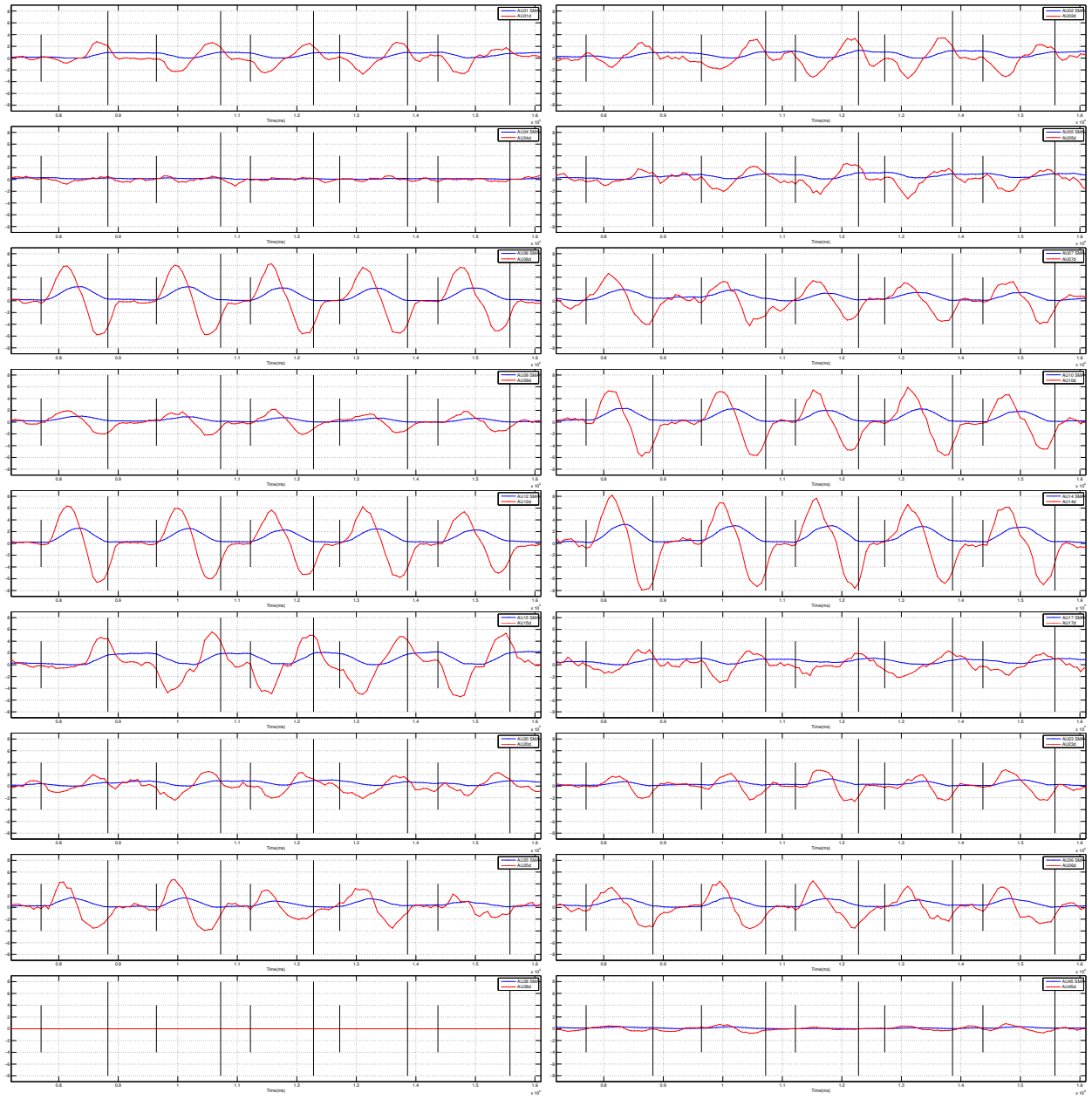


Figure 9 – Analysis of simple moving average (SMA) and derivative signals for all Action units in a test which the user has emitted five consecutive *smiles*. The vertical lines from -4 to 4 and from -8 to 8 indicates the beginning and the end of each emission, respectively. The start and stop marks (in *ms*) for each *smile* are: (7703, 8825), (9641, 10720), (11220, 12280), (12720, 13860), (14370, 15580). The derivative signals display the moments when the user has smiled and indicate the most relevant AUs (06, 10, 12, 14) associated with this facial expression.

4.2.3 Facial Expression Detection Algorithm

When the user emits a *smile, blink* or *eyebrow up* their respective AUs produce very similar derivative signals characterized by a positive increase and decrease followed by negative inverted variation. Also, the derivative signal reach similar maximum and minimum values, which means we could set upper and lower thresholds to detect this feature over time.

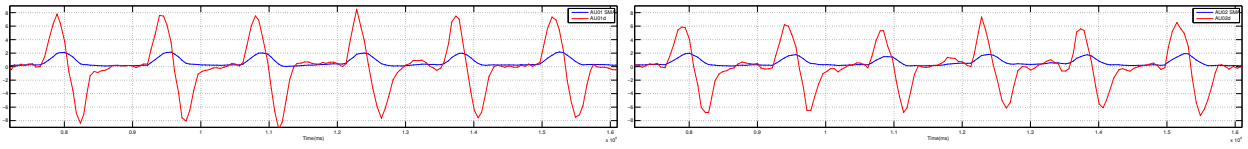


Figure 10 – The most significant AUs (AU01 and AU02) for *eyebrows up*.

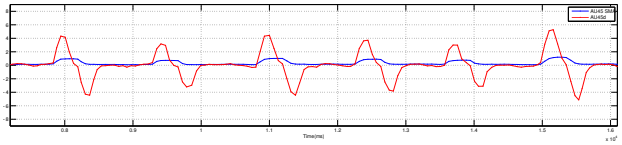


Figure 11 – The most unique AU (AU45) activated for *blink*.

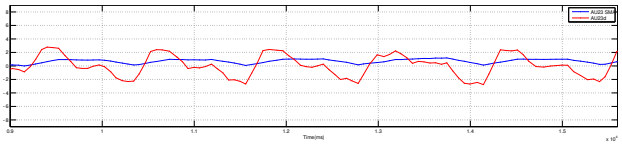


Figure 12 – The most similar AU (AU23) with bath quality signal activated for *kiss*.

However, this condition is not sufficient enough to distinguish between a facial expression intended to be a command for our navigation assistant and natural expressions emitted by the user during a conversation, for example. In order to minimize the "Midas Touch" problem and focusing on detecting the facial expressions as commands for the wheelchair, we have also considered the time duration between the first derivative value above the upper threshold and the first below the lower threshold.

Thus, we empirically developed our facial detection algorithm based on the signal analysis and the intention to identify *smile, eyebrows up* and *blink* as commands for the navigation assistant. The algorithm is divided in two parts: identify the activation of an Action Unit and classify the facial expression according to the relevant AUs discussed in the previous subsection. Equation 4.3 represents the Action Unit activation:

$$AU_a = \begin{cases} 1, & \text{if } AUd(t_U) \geq U_{ref} \text{ and } AUd(t_L) \leq L_{ref} \text{ and } t_L - t_U \leq \Delta T_{UL} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

Where: $AUd(t_U)$ is the first derivative value (detected in instant t_U) greater or equal some constant upper threshold U_{ref}, $AUd(t_L)$ is the first derivative value (detected in instant t_L) lower or equal some constant lower threshold L_{ref} and ΔT_{UL} is the time limit for the occurrence of $AUd(t_L)$ after $AUd(t_U)$.

Figure 13 depicts an example of the AU01 activation when a quick *eyebrows up* is emitted. The upper and lower thresholds have been set to 4.5 and -4.5 and the time limit

for activation (ΔT_{UL}), fixed in 600 ms. In this case, the first step of the algorithm outputs $AU01_a = 1$ (activated) for attending the upper and lower criteria and $t_L - t_U = 340$ ms $\leq \Delta T_{UL}$.

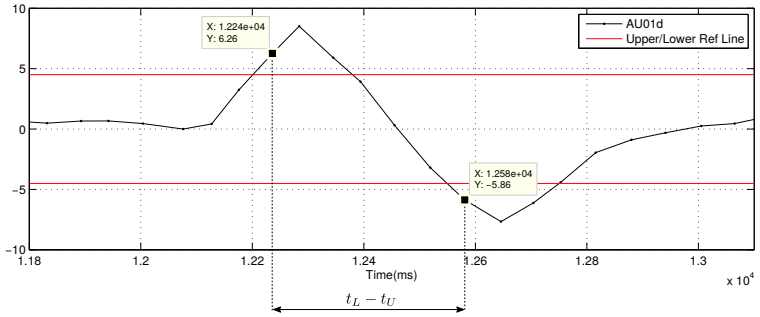


Figure 13 – AU01 activated when the user emits a quick *eyebrows up*.

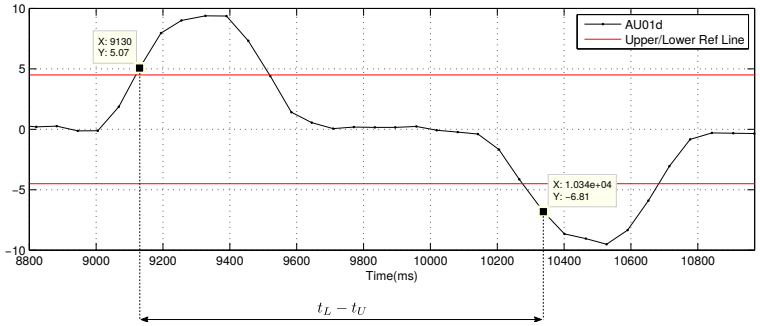


Figure 14 – AU01 not activated when the user emits a loong *eyebrows up*.

Conversely, Figure 14 illustrates the same AU when the user emits a longer *eyebrows up* and as expected, the first step of the algorithm does not detect the AU01 activation. Although the derivative values have reached both upper and lower criteria, the Action Unit is not activated ($AU01_a = 0$) because $t_L - t_U = 1210$ ms which exceeds the stablished $\Delta T_{UL} = 600$ ms.

The second step of the algorithm consists of checking the activated AUs and properly attribute the occurrence of the facial expression. As we discussed in the previous subsection, the *eyebrows up* can be identified by the activation of AU01 and AU02. However, we decided to monitor only AU01 as using both is redundant besides that simple experiments have shown that its derivative signal has reached greater positive and negative peaks than AU02.

The *smile* has stimulated mainly AU06, AU10, AU12, and AU14. In the first moment we have considered these four AUs in the algorithm, but we had a lot of false negatives when executing simple tests with the application. Therefore, we decided to ease the strict criteria and monitor only two Action Units: AU14 for having greater variations, and AU06 as a second condition to make the *smile* detection rigid enough to avoid false positives such as when the user talks or move his/her lips.

The *blink* detection is made only with AU45, which is the unique Action Unit provided by the base *OpenFace* Framework. Although a single *blink* is not a reliable facial expression to be used as a command for out navigation assistant, we have implemented its detection anyway.

Thus, Equations 4.4, 4.5 and 4.6 resume the algorithm for detection of *smile*, *eyebrows up* and *blink*, respectively:

$$smile = \min(AU06_a, AU14_a) \tag{4.4}$$

$$eyebrows\_up = AU01_a \tag{4.5}$$

$$blink = AU45_a \tag{4.6}$$

4.2.4 Calibration

The development of the facial expression detection algorithm has been led by the AUs signals analysis and empirical definition of the default constants U_{ref}, L_{ref} and ΔT_{UL} for each of the Action Units (AU01, AU06, AU14, AU45). However, the *OpenFace* application may have different classification behaviors depending on the light conditions or when used by different people which may have distinct signal variations when emitting facial expressions. For this reason, we have also incorporated a calibration feature to our modified version of the *OpenFace* application, so the software calculates the default constants (calibration parameters) to improve the activation (Equation 4.3) of the AUs and consequently the correct identification of the facial expressions as commands for the navigation assistant.

The calibration algorithm reproduces in an automated manner, the manual procedure (signal analysis) we have done to define the constants U_{ref}, L_{ref}, ΔT_{UL} (calibration parameters). When the user enters in calibration mode, he/she is asked to emit three facial expressions for each type. The software provides visual feedback in real-time indicating which one should be performed and when the user emits a *smile*, *eyebrows up* or *blink*, the software records the derivative signals and makes the automated signal analysis to extract the constants. After recording the AUs, the application calculates the calibrations parameters and export the values to a JSON (JavaScript Object Notation) file, which can be loaded depending on who is using *OpenFace* or the environment which may have different light conditions.

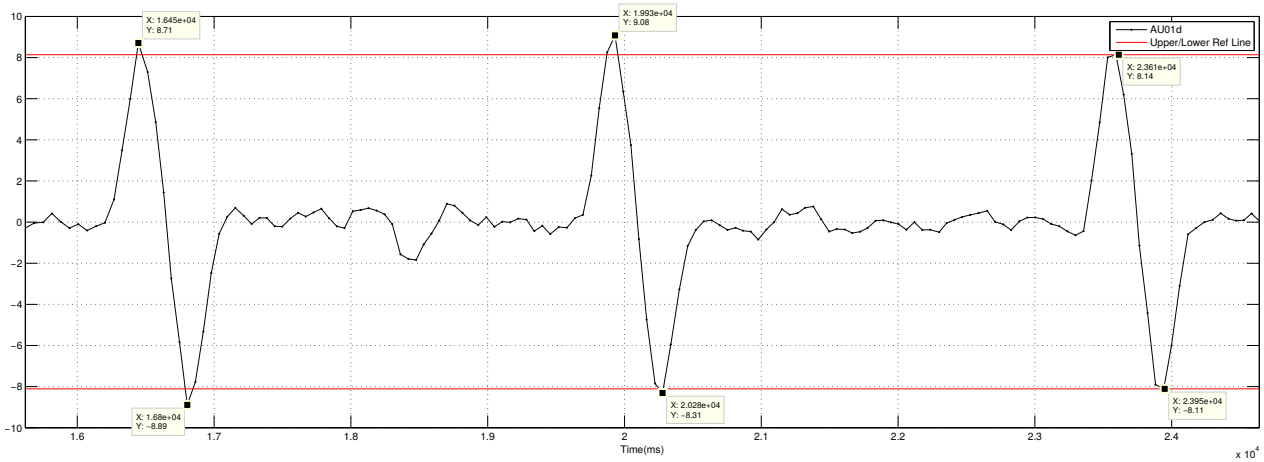


Figure 15 – Record of AU01 signal to extract the calibration parameters U_{ref} and L_{ref}.

Figure 15 depicts the AU01 derivative signal during the calibration of *eyebrows up*. The application records each emission (from 1600 ms to 17500 ms for the first one, e.g.) and select the max and min peaks (8.71 and -8.89). Finally, the algorithm calculates the $U_{ref} = 8.14$ which is the min of the max peaks (8.71, 9.08, 8.14), and the the $L_{ref} = -8.11$ which is the max of the min peaks (-8.89, -8.31, -8.11). Equations 4.7 abd 4.8 resumes the calculation of U_{ref} and L_{ref} for each AU, respectively:

$$U_{ref} = \min(\max()R1_{AU}), \max(R2_{AU}), \max(R3_{AU})) \tag{4.7}$$

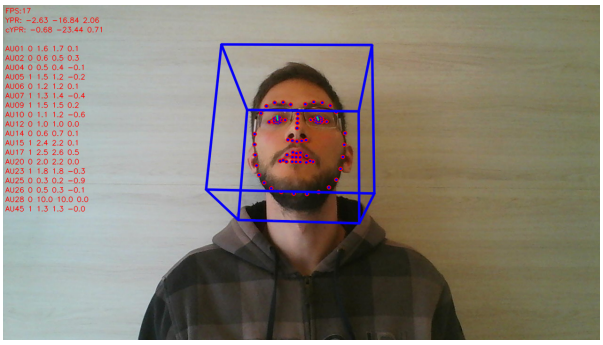
$$L_{ref} = \max(\min(R1_{AU}), \min(R2_{AU}), \min(R3_{AU})) \tag{4.8}$$

The signal analysis have demonstrated that the ΔT_{UL} does not show huge variation for different people and light conditions. Thus we have set the following values for AU01, AU06, AU14, AU45: 600, 600, 700 and 600 ms (respectively).

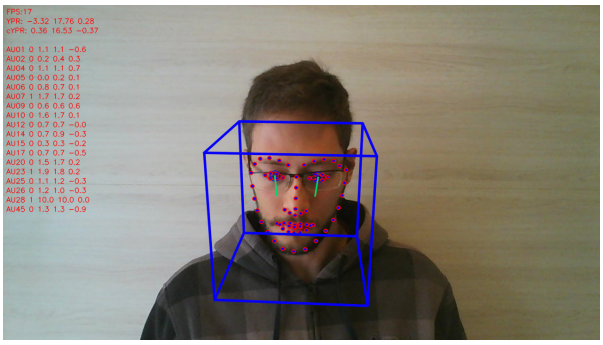
4.2.5 Head Orientation

Besides the AUs signals, *OpenFace* provides angles estimation of the head orientations which could be used as a hands-free HMI to control the *pan-tilt laser* pointer (to be discussed in Chapter 5) or combined with the facial expressions to compose commands for the navigation assistant. Thus, we have just transformed the *yaw*, *pitch* and *roll* angles from radian to degrees (human-readable) and implemented the functionality of setting the zero reference frame when the user emits a *smile*.

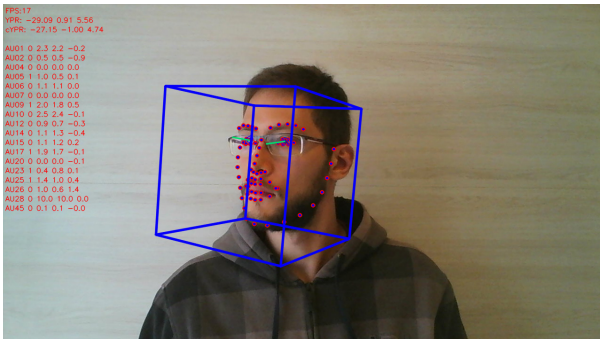
Figure 16 illustrates the head orientation angles adapted from the raw valus provided bu the *OpenFace* Framework. The upper left corner show the *YPR* (*yaw, pitch, roll*) angles in



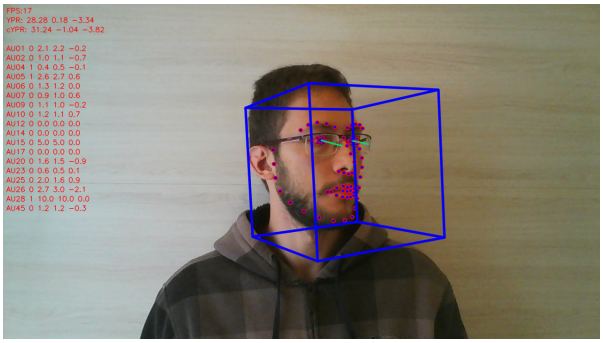
(a) Head up (−23.44°)



(b) Head down (16.53°)



(c) Head right (−27.15°)



(d) Head left (31.24°)

Figure 16 – Head orientation angles estimated by the *OpenFace* Framework. We have transformed the raw values from radian to degrees to provide a human-readable format.

degrees, and the $cYPR$ which are the corrected angles after the user has set the zero reference frame by positioning his head looking ahead and emitting a *smile* command.

4.3 Chapter Conclusions

In this chapter, we presented the development of the *OpenFace* application as a reliable hands-free module which estimated the head orientation and detects the following facial expressions: eyebrows up, smile and blink. Results have shown that *OpenFace* may be a viable solution to control the pan-tilt laser pointer with the head orientation (to be presented in Chapter 5) and trigger action on the navigation assistant using the available facial expressions.

5 Laser Pointer Controlled by Head Posture

5.1 Introduction

This chapter describes the mathematical modeling and development of the pan-tilt laser subsystem introduced by (ROHMER *et al.*, 2015a) in Chapter 2. The pan-tilt laser has been validated using a virtual environment, and the following sections describe the implementation details and solutions to make this module viable for practical experiments. In the next section,

5.2 Pan-tilt laser subsystem

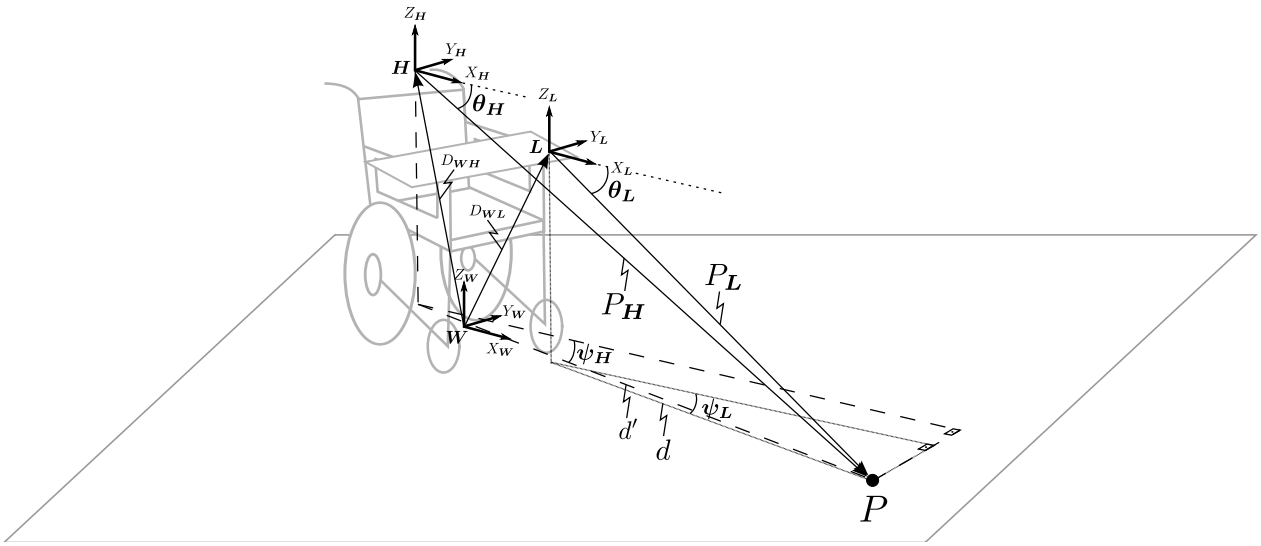


Figure 17 – Head's posture tracker calculations to estimate $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$ as functions of $\boldsymbol{\psi_H}$ and $\boldsymbol{\theta_H}$.

In (ROHMER *et al.*, 2015a), they present a head's posture tracker to control the pan-tilt laser. It consists of a 9 DOF Inertial Measurement Unit (IMU) placed on the user's head to estimate the posture in real-time and actuate on the laser by pointing it on the ground where the operator is looking. The user's head orientation can be calculated by fusing accelerometers, gyroscopes and magnetometers data using a Direction Cosine Matrix filtering (PREMERLANI; BIZARD, 2009) which provides a robust estimate of the head's posture to actuate on the pan-tilt mechanism. The system works under the assumption that the head aims at a particular point on the ground only by rotation of pan ($\boldsymbol{\psi_H}$) and tilt ($\boldsymbol{\theta_H}$) angles. Therefore, considering Figure 17 our goal is to find equations that express the

laser angles pan ($\boldsymbol{\psi_L}$) and tilt ($\boldsymbol{\theta_L}$) we want to control as functions of the head angles $\boldsymbol{\psi_H}$ and $\boldsymbol{\theta_H}$. One should note that there is no rotation between the reference coordinate systems $\boldsymbol{H}$ (head), $\boldsymbol{L}$ (laser) and $\boldsymbol{W}$ (wheelchair). Hence, firstly, we calculate the representation of target point P with respect to the user's head frame $\boldsymbol{H}$, called $P_{\boldsymbol{H}}$:

$$P_{\boldsymbol{H}} = \begin{bmatrix} P_{x_{\boldsymbol{H}}} \\ P_{y_{\boldsymbol{H}}} \\ P_{z_{\boldsymbol{H}}} \end{bmatrix} = \begin{bmatrix} d' \cos \boldsymbol{\psi_H} \\ d' \sin \boldsymbol{\psi_H} \\ -D_{\boldsymbol{WH}z} \end{bmatrix} = D_{\boldsymbol{WH}z} \begin{bmatrix} \frac{\cos \boldsymbol{\psi_H}}{\tan \boldsymbol{\theta_H}} \\ \frac{\sin \boldsymbol{\psi_H}}{\tan \boldsymbol{\theta_H}} \\ -1 \end{bmatrix} \tag{5.1}$$

where, $D_{\boldsymbol{WH}} = \begin{bmatrix} D_{\boldsymbol{WH}_x} & D_{\boldsymbol{WH}_y} & D_{\boldsymbol{WH}_z} \end{bmatrix}^\intercal$ is the translation of frame $\boldsymbol{H}$ with respect to the wheelchair frame $\boldsymbol{W}$ and $d' = \frac{D_{\boldsymbol{WH}z}}{\tan \boldsymbol{\theta_H}}$ is the distance between the projection of frame $\boldsymbol{H}$ on the ground and point P. Secondly, we calculate $P_{\boldsymbol{L}}$ which corresponds to the same point P with respect to the pan-tilt laser frame $\boldsymbol{L}$:

$$P_{\boldsymbol{L}} = \begin{bmatrix} P_{x_{\boldsymbol{L}}} \\ P_{y_{\boldsymbol{L}}} \\ P_{z_{\boldsymbol{L}}} \end{bmatrix} = \begin{bmatrix} d \cos \boldsymbol{\psi_L} \\ d \sin \boldsymbol{\psi_L} \\ -D_{\boldsymbol{WL}z} \end{bmatrix} = D_{\boldsymbol{WL}z} \begin{bmatrix} \frac{\cos \boldsymbol{\psi_L}}{\tan \boldsymbol{\theta_L}} \\ \frac{\sin \boldsymbol{\psi_L}}{\tan \boldsymbol{\theta_L}} \\ -1 \end{bmatrix} \tag{5.2}$$

.

Where, $D_{\boldsymbol{WL}} = \begin{bmatrix} D_{\boldsymbol{WL}_x} & D_{\boldsymbol{WL}_y} & D_{\boldsymbol{WL}_z} \end{bmatrix}^\intercal$ is the translation of frame $\boldsymbol{L}$ with respect to the wheelchair frame $\boldsymbol{W}$ and $d = \frac{D_{\boldsymbol{WL}z}}{\tan \boldsymbol{\theta_L}}$ is the distance between the projection of frame $\boldsymbol{L}$ on the ground and point P.

Now that we have a representation of the same point P with respect to the pan-tilt laser and user's head frames, we use equations 5.1 and 5.2 to express point P with respect to the wheelchair frame. Because there is no rotation between the coordinate systems, we can make the transformations considering only translations of $\boldsymbol{H}$ and $\boldsymbol{L}$ with respect to $\boldsymbol{W}$. If we express $P_{\boldsymbol{W}}$ considering the user's head frame we can write:

$$P_{\boldsymbol{W}} = D_{\boldsymbol{WH}} + P_{\boldsymbol{H}} = \begin{bmatrix} D_{\boldsymbol{WH}x} + D_{\boldsymbol{WH}z} \frac{\cos \boldsymbol{\psi_H}}{\tan \boldsymbol{\theta_H}} \\ D_{\boldsymbol{WH}y} + D_{\boldsymbol{WH}z} \frac{\sin \boldsymbol{\psi_H}}{\tan \boldsymbol{\theta_H}} \\ D_{\boldsymbol{WH}z} - D_{\boldsymbol{WH}z} \end{bmatrix} = \begin{bmatrix} D_{\boldsymbol{WH}x} + D_{\boldsymbol{WH}z} \frac{\cos \boldsymbol{\psi_H}}{\tan \boldsymbol{\theta_H}} \\ D_{\boldsymbol{WH}y} + D_{\boldsymbol{WH}z} \frac{\sin \boldsymbol{\psi_H}}{\tan \boldsymbol{\theta_H}} \\ 0 \end{bmatrix} \tag{5.3}$$

And expressing $P_{\boldsymbol{W}}$ considering the pan-tilt laser frame:

$$P_{\boldsymbol{W}} = D_{\boldsymbol{WL}} + P_{\boldsymbol{L}} = \begin{bmatrix} D_{\boldsymbol{WL}x} + D_{\boldsymbol{WL}z} \frac{\cos \boldsymbol{\psi_L}}{\tan \boldsymbol{\theta_L}} \\ D_{\boldsymbol{WL}y} + D_{\boldsymbol{WL}z} \frac{\sin \boldsymbol{\psi_L}}{\tan \boldsymbol{\theta_L}} \\ D_{\boldsymbol{WL}z} - D_{\boldsymbol{WL}z} \end{bmatrix} = \begin{bmatrix} D_{\boldsymbol{WL}x} + D_{\boldsymbol{WL}z} \frac{\cos \boldsymbol{\psi_L}}{\tan \boldsymbol{\theta_L}} \\ D_{\boldsymbol{WL}y} + D_{\boldsymbol{WL}z} \frac{\sin \boldsymbol{\psi_L}}{\tan \boldsymbol{\theta_L}} \\ 0 \end{bmatrix} \tag{5.4}$$

Finally, we equate 5.3 and 5.4, and solve the non-linear system to find $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$ as functions of $\boldsymbol{\psi_H}$ and $\boldsymbol{\theta_H}$, which results in equations 5.5 and 5.6:

$$\begin{cases} D_{\boldsymbol{WH}x} + D_{\boldsymbol{WH}z}\frac{\cos\boldsymbol{\psi_H}}{\tan\boldsymbol{\theta_H}} & = D_{\boldsymbol{WL}x} + D_{\boldsymbol{WL}z}\frac{\cos\boldsymbol{\psi_L}}{\tan\boldsymbol{\theta_L}} \\ D_{\boldsymbol{WH}y} + D_{\boldsymbol{WH}z}\frac{\sin\boldsymbol{\psi_H}}{\tan\boldsymbol{\theta_H}} & = D_{\boldsymbol{WL}y} + D_{\boldsymbol{WL}z}\frac{\sin\boldsymbol{\psi_L}}{\tan\boldsymbol{\theta_L}} \end{cases}$$

$$\tan\boldsymbol{\psi_L} = \frac{\sin\boldsymbol{\psi_H}\cot\boldsymbol{\theta_H}D_{\boldsymbol{WH}z}(D_{\boldsymbol{WH}y} - D_{\boldsymbol{WL}y})}{cos\boldsymbol{\psi_H}\cot\boldsymbol{\theta_H}D_{\boldsymbol{WH}z} + (D_{\boldsymbol{WH}x} - D_{\boldsymbol{WL}x})} \tag{5.5}$$

$$\tan\boldsymbol{\theta_L} = \frac{\cos\boldsymbol{\psi_L}D_{\boldsymbol{WL}z}}{cos\boldsymbol{\psi_H}\cot\boldsymbol{\theta_H}D_{\boldsymbol{WH}z} + (D_{\boldsymbol{WH}x} - D_{\boldsymbol{WL}x})} \tag{5.6}$$

Thus, we can use equations 5.5 and 5.6 to implement a position control of the laser angles ($\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$) based on the user's head orientation. The next section describes the mathematical model and implementation to estimate the target position on the ground pointer by the laser and the navigation algorithm to reach the destination.

5.3 Target position calculation and wheelchair navigation control

In the previous section, we presented the mathematical model to actuate on the pan-tilt laser based on the user's posture estimation. In this section, we describe the strategy to calculate the target position with respect to the laser frame considering that our pan-tilt system consists of low-cost micro-servos actuated by digital pulse-width modulation (PWM) signals. Additionally, we propose a simple navigation algorithm to reach the destination considering the target coordinates with respect to the wheelchair frame on the ground.

The prototype of the pan-tilt subsystem has been implemented using an *Arduino Mega 2560* with the control algorithm to actuate on both pan and tilt servos. According to equations 5.5 and 5.6, we can obtain the laser angles ($\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$) considering the estimated head angles. However, to actuate properly on the pan-tilt laser, firstly we must have a mathematical model which transforms the laser angles $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$ (in degrees) to the PWM signals pan ($\boldsymbol{S_\psi}$) and ($\boldsymbol{S_\theta}$) that control the servos positioning and may assume values from 0 to 180. Secondly, to minimize errors introduced by the mechanical structure of the pan-tilt in real experiments, we must have a calibration model which transforms $\boldsymbol{S_\psi}$ and $\boldsymbol{S_\theta}$ to $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$. Thus, the obtained laser angles can be used to calculate the target position with respect to the wheelchair frame, and finally, the navigation algorithm can take the chair to the chosen destination.

Figure 18 depicts the calibration method to transform $\boldsymbol{S_\psi}$ and $\boldsymbol{S_\theta}$ to $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$. The procedure relies on manually positioning the laser beam (by actuating on the servos) in some

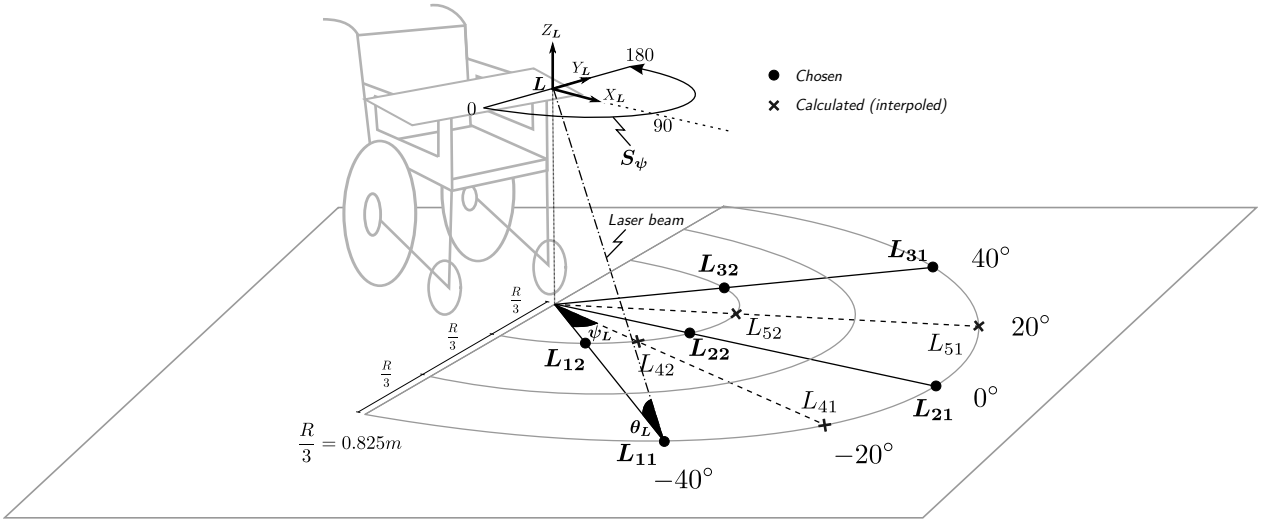


Figure 18 – Calibration method of servos to associate S_ψ and S_θ PWM signals to $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$ angles in degrees.

known landmarks and extract linear equations that express $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$ as functions of $\boldsymbol{S_\psi}$ and $\boldsymbol{S_\theta}$. Thus, the calibrations steps are:

1. Position the wheelchair in a manner that the $Z_{\boldsymbol{L}}$ axis of the pan-tilt laser frame gets aligned with the origin of the half-circumference, and the $X_{\boldsymbol{L}}$ axis aligned with the 0° orientation on the ground.

2. Point the laser beam in each of the six rounded landmarks (two for each of the directions: −40°, 0° and 40°) and save the correspondent values of $\boldsymbol{S_\psi}$ and $\boldsymbol{S_\theta}$. One should note that for simplification reasons, we assume the same $\boldsymbol{S_\psi}$ values for the two points of the same direction.

3. Run the calibration algorithm that performs the following calculations:

   a) Estimate $\boldsymbol{S_\theta}$ values for L_{41}, L_{42}, L_{51}, L_{52} using interpolation of the values for $\boldsymbol{L_{11}}$, $\boldsymbol{L_{12}}$, $\boldsymbol{L_{21}}$, $\boldsymbol{L_{22}}$, $\boldsymbol{L_{31}}$, $\boldsymbol{L_{32}}$.

   b) Calculate a linear regressions for each of the five directions (−40°, −20°, 0°, 20°, 40°) to find matrix α (equation 5.7) which consists of the equations parameters to transform a given $\boldsymbol{S_\theta}$ value to the correspondent $\boldsymbol{\theta_L}$ angle in degrees (equations 5.8).

$$\alpha = \begin{bmatrix} \alpha_{0_{(1)}} & \alpha_{0_{(4)}} & \alpha_{0_{(2)}} & \alpha_{0_{(5)}} & \alpha_{0_{(3)}} \\ \alpha_{1_{(1)}} & \alpha_{1_{(4)}} & \alpha_{1_{(2)}} & \alpha_{1_{(5)}} & \alpha_{1_{(3)}} \end{bmatrix} \tag{5.7}$$

$$\boldsymbol{\theta_L} = \begin{cases} \alpha_{1_{(1)}}\boldsymbol{S_\theta} + \alpha_{0_{(1)}}, & \text{if } \boldsymbol{S_\psi} < 65, \text{ closer to } -40^\circ \text{ (direction 1)} \\ \alpha_{1_{(4)}}\boldsymbol{S_\theta} + \alpha_{0_{(4)}}, & \text{if } 65 \leq \boldsymbol{S_\psi} < 90, \text{ closer to } -20^\circ \text{ (direction 4)} \\ \alpha_{1_{(2)}}\boldsymbol{S_\theta} + \alpha_{0_{(2)}}, & \text{if } 90 \leq \boldsymbol{S_\psi} < 105, \text{ closer to } 0^\circ \text{ (direction 2)} \\ \alpha_{1_{(5)}}\boldsymbol{S_\theta} + \alpha_{0_{(5)}}, & \text{if } 105 \leq \boldsymbol{S_\psi} < 141, \text{ closer to } 20^\circ \text{ (direction 5)} \\ \alpha_{1_{(3)}}\boldsymbol{S_\theta} + \alpha_{0_{(3)}}, & \text{if } \boldsymbol{S_\psi} > 141, \text{ closer to } 40^\circ \text{ (direction 3)} \end{cases} \tag{5.8}$$

The calibration of the system is done only once, and after that, we can estimate the coordinates of a target $P_{\boldsymbol{W}}$ with respect to the wheelchair frame on the ground by using one of the equations 5.8 to get the $\boldsymbol{\theta_L}$ and equation 5.4. One should note that we assume that the pan angle is related with the $\boldsymbol{S_\psi}$ values of the servo by equation 5.9, as we consider only the tilt rotation to calculate the α matrix (Figure 18 reinforces the idea).

$$\boldsymbol{\psi_L} = \boldsymbol{S_\psi} - 90^\circ \tag{5.9}$$

After computing the target position with respect to the wheelchair frame on the ground, we need to implement the navigation control to reach the chosen destination. The simplest algorithm consists of a rotation to align the $X_{\boldsymbol{W}}$ axis with the target followed by a linear dislocation to reach the chosen point. Although this solution is simple, it does not consider if the path is free of obstacles and possibly does not provide the most efficient trajectory. For this reason (also mentioned in (ROHMER *et al.*, 2015a)), it is necessary to integrate a depth sensor with an algorithm to provide two functionalities: traversability analysis and path planing. The first one consists of classifying the target as reachable or not by the chair based on a traversability map, before validating it. The second one relies on this same map to calculate the best route to the chosen point considering the obstacles in a horizontal scene. After that, we can implement a path following algorithm to reach the destination autonomously.

5.4 Chapter Conclusions

This chapter presented the mathematical model and implementation of the pan-tilt laser pointer controlled by the user's head posture. According to (ROHMER *et al.*, 2015a), this concept has been validated in V-REP robotic simulation framework (ROHMER *et al.*, 2013). However, we must validate it with real experiments to evaluate: the effectiveness of the pan-tilt control (actuation on the servos); target position calculation considering the

proposed calibration method; simple navigation algorithm to reach the chosen destination. Chapter 6 presents the experimental results of the pan-tilt laser pointer subsystem.

6 Laser Pointer Results

6.1 Introduction

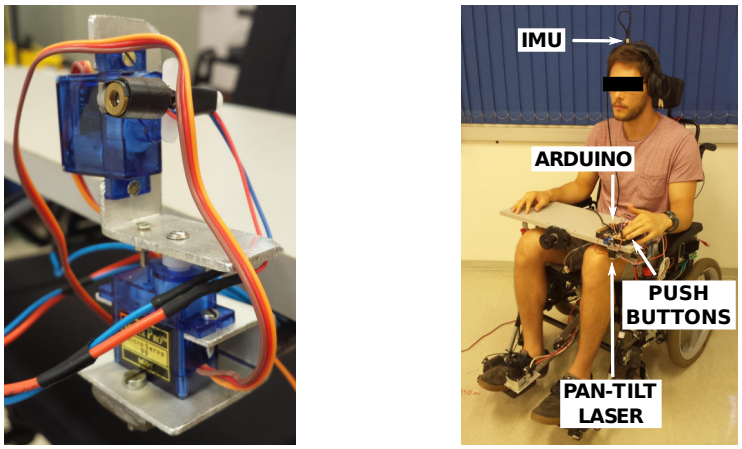
This chapter presents the implementation and experimental results to evaluate the viability of the proposed low-cost pan-tilt laser to compose our navigation assistant for robotized wheelchairs. Thus, this chapter focus on discussing the validation of the mathematical model involved in the pan-tilt system presented in Chapter 5. Instead of image-based HMI to control the laser pointer, we have used an IMU to acquire the user's head posture. Although the primary goal is to develop a hands-free assistant, the target validation has been implemented with push buttons for testing purposes. The promising results of the pan-tilt laser control and target calculation leaded the project to compose the final version of the navigation assistant presented in Chapter 7.

6.2 Methodology

Figure 19b depicts the system overview and its components to compose the experimental platform to validate the mathematical model of the pan-tilt laser presented in Chapter 5. Above the user's head, there is an Inertial Measurement Unit (IMU) *SparkFun 9 Degrees of Freedom* sensor stick to estimate the user's head posture. This device combines data acquired from accelerometer, gyroscope, and magnetometers to provide robust yaw, pitch and roll angles estimation.

The hand-made pan-tilt laser depicted in Figure 19a has been assembled with two *Micro servos 9g SG90 TowerPro*, aluminum sheets, and a diode laser and fixed on the wheelchair to point the target destination. Three push buttons were added to a breadboard to lock/unlock the pan-tilt laser as well as validate the chosen target. All electronic components are connected to an *Arduino Mega 2560* which implements the algorithm to control the pan-tilt servos and the target calculation considering the mathematical model of the calibration method discussed in Section 5.3.

The pan-tilt subsystem has been assembled on the robotized wheelchair developed by (JúNIOR, 2016), which provides an experimental platform to validate navigation algorithms. The wheelchair is equipped with an access point along with an API (Application Programming Interface) which accepts high-level commands via HTTP to actuate directly on the rear wheels and perform actions such as single rotations, heading movements, linear dislocations



(a) (b)

Figure 19 – (a) The low-cost pan-tilt laser used to point the destination. (b) System overview. The IMU sensor stick acquires the user's head posture, and the control algorithm implemented with the Arduino actuates on the pan-tilt servos. When the user presses the validation push button, the robotized wheelchair performs a heading movement followed by a linear displacement until the chosen target on the ground.

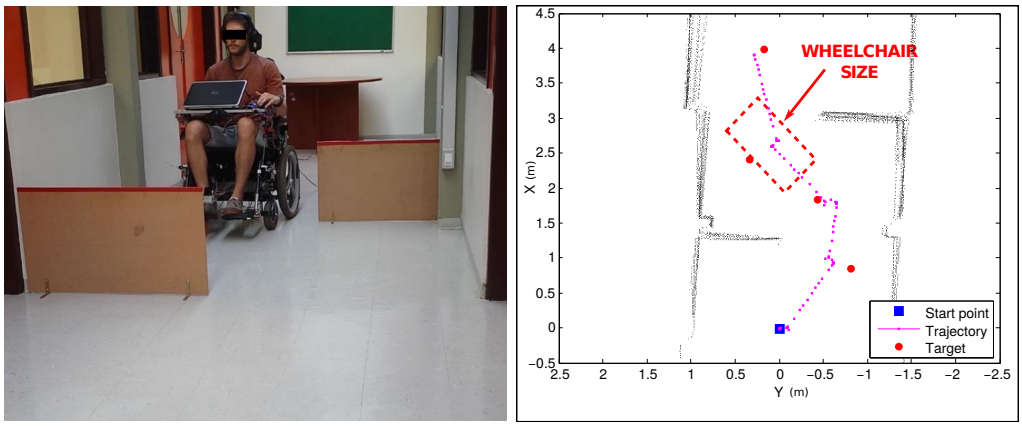
and stop command. Also, there are stop push buttons which can be used by the operator for emergency stops in case of imminent collisions.

The Arduino communicates is connected via USB (serial communication) with a notebook which runs a Matlab application to receive the target coordinates and implement the navigation algorithm to reach the chosen destination. When the user validates a target, the Matlab application transforms the received coordinates to the wheelchair frame on the ground and performs the proper heading movement followed by a linear displacement until the chosen destination.

6.3 Experimental Procedure and Results

Figures 20a illustrates an experiment which the user navigates in a corridor with obstacles using semi-autonomous navigation. The IMU positioned above the operator's head captures its posture to actuate on the pan-tilt laser, and by pressing a push button, the wheelchair performs a rotation of the angle with respect to the chosen target (heading movement) and then dislocates linearly.

The pan-tilt assembly, the wheelchair intrinsic parameters, and the head's posture estimation introduce errors on the targets calculations and wheelchair rotations. In Figure 20b, the distance error between the chosen destinations and the wheelchair stop marks are approximately *0.3 meters* which is acceptable considering the wheelchair size: *1.3* x *0.6 meters* (length x width). Also, even the performed trajectory was not long (around 4.75 *meters*), a



(a) (b)

Figure 20 – (a) The user operating the human-robot interface on a scenario with obstacles. (b) The performed trajectory based on the chosen targets. The dots of the corridor walls and obstacles were acquired with a laser rangefinder sensor fixed in front of the robotized wheelchair. The trajectory dots were collected during the navigation since the robotized wheelchair also provides odometry information. Finally, the targets were calculated by the pan-tilt laser and expressed with respect to the wheelchair frame. The axis X and Y represent the distance in meter with respect to the start point of navigation.

localization algorithm could improve our navigation assistant. It could be implemented on a local map even for a short period.

The obtained results have shown that the pan-tilt laser subsystem has calculated the targets correctly and the robotized wheelchair has reached the chosen destinations with acceptable distance errors. The simple navigation algorithm which combines a heading movement followed by a linear displacement for each chosen target may be a comfortable navigation mode to compose our system. However, the locomotion algorithm could be safer and more efficient with the integration of a depth camera with a scene descriptor to generate a safe trajectory considering the obstacles. Thus, the navigation algorithm could perform a path following until each chosen target.

6.4 Chapter Conclusions

In this chapter, we presented the implementation and validation of mathematical model involved on the pan-tilt laser subsystem. Although the target validation is triggered by a push-button which is not a hands-free component, results have shown that the implemented semi-autonomous navigation algorithm may be effective and comfortable for the user.

The IMU sensor to estimate the user's head orientation has been used for testing purposes in our experiment, and there are some aspects which makes this solution unpractical for our navigation assistant: not wireless, not comfortable, loses calibration easily, difficult

calibration method, suffers from magnetic influence on signal measurements. Thus, we could replace the IMU by the *OpenFace* HMI discussed in Chapter 4, since this solution is more comfortable and also provides a reliable estimation of the user's head.

7 Wheelchair Navigation Assistant with OpenFace and Laser Pointer

The *OpenFace* application described in Chapter 4 has shown to be a reliable and comfortable HMI to integrate the navigation assistant for our robotized wheelchair. The subsystem provides head's posture measurement that can be used to control the pan-tilt servos and position the laser beam on the desired spot. Also, *OpenFace* provides facial expressions detections which can be used to actuate directly on the wheelchair or send commands to validate the target position or lock/unlock the *pan-tilt laser* subsystem.

Therefore, this Chapter describes the hardware and software integration of the *pan-tilt laser* module discussed in Chapters 5 and 6, the *OpenFace* HMI and the robotized wheelchair with *manual* and *semi-autonomous* navigation algorithms for the assistant.

7.1 System Overview

Figure 21 depictures the integration of *OpenFace* HMI, *pan-tilt laser pointer* and *Matlab* application to compose the navigation assistant for robotized wheelchairs. Our experimental wheelchair provides an Access Point which we connected the *pan-tilt laser pointer* module and a notebook running both *OpenFace* and *Matlab* application.

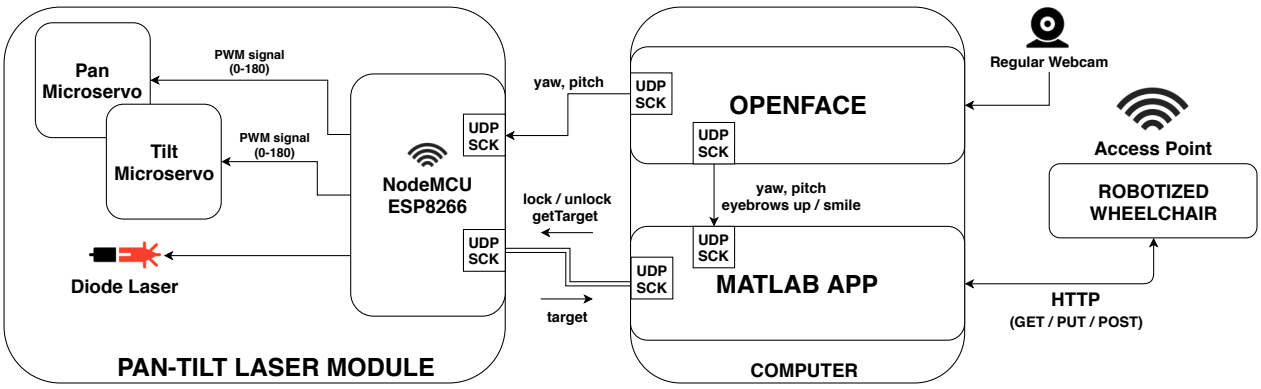


Figure 21 – Overview of subsystems integration to compose the navigation assistant for robotized wheelchairs.

The *OpenFace* application processes the captured images from the webcam and detects three facial expressions (*eyebrows up*, *smile* and *blink*) as well as the head's posture angles: *yaw* ($\boldsymbol{\psi_H}$) and *pitch* ($\boldsymbol{\theta_H}$) introduced in Section 5.2. This application contains two UDP sockets: one to send the head's posture angles to the *pan-tilt laser* module and actuate on

the servos and another to send both facial expressions and angles to compose the commands of the navigation assistant.

When enabled, the *pan-tilt laser pointer* reads the user's head posture from *OpenFace* through one UDP socket and actuates on the servos accordingly to point the laser on an approximate region on the ground where he/she is looking. When the target is validated, this the *pan-tilt laser* module receives a signal through a second UDP connection with *Matlab* application and returns the calculated destination coordinates with respect to the pan-tilt frame.

The *Matlab* application receives the user gestures captured from *OpenFace* (head's posture and facial expressions) and translate to commands to actuate on the *pan-tilt laser* module and on the wheelchair using HTTP requests. The navigation assistant provides two navigation modes: *manual* and *semi-autonomous*. In the first one, the user gestures actuated directly on the wheelchair movements (*front*, *rotate right* and *rotate left*) and in the second one, he/she chooses destination pointed by the laser pointer, and the wheelchair performs a heading movement followed by a linear displacement.

Table 5 shows the available combinations of gestures and their respective commands of the navigation assistant. If the user holds his/her *head up* and executes an *eyebrows up* the system puts a linear velocity of *150 mm/s* on both wheels. If the user makes a *smile* the *OpenFace* application sets the actual user's head posture as the zero reference and the *Matlab* application triggers two actions: stops the wheelchair and locks the pant-tilt laser pointer (if enabled). If the user holds his/her *head up right* or *up left* and makes an *eyebrows up*, the *Matlab* application applies rotational velocity of *10* or *-10 degrees/second* on the wheels (linear velocity is zero).

By positioning his/her head slightly down and confirming with an *eyebrows up* the user unlocks the *pan-tilt laser* and can navigate on *semi-autonomous* mode. After pointing the laser dot to the desired destination on the ground he/she confirms with a second *eyebrows up* and the *Matlab* application requests the target coordinates to the *pan-tilt laser* module. Based on that, the application calculates the necessary rotation and linear displacement and coordinates these movements. To return to *manual* mode the user can simply make a *smile* which will lock the *pan-tilt laser* and stop the wheelchair. Both navigation modes are independent and cannot be active at the same time.

Although *OpenFace* provides three facial expressions, we decided not to use the *blink* as this one is an involuntary movement and thus not safe for our navigation assistant purposes. In addition to the *OpenFace* algorithm which was designed to detect specific facial expressions and avoid the system to capture undesired commands, we decided to combine the user's head

Table 5 – Navigation assistant commands actuate on the wheelchair on manual and semi-autonomous modes.

Navigation Modes	**Commands**	**User Gestures**
Manual	Go front	*head up + eyebrows up*
	Rotate right	*head up right + eyebrows up*
	Rotate left	*head up left + eyebrows up*
	Stop wheelchair and lock pan-tilt laser	*smile*
Semi-autonomous	Unlock pan-tilt laser and switch to semi-autonomous mode	*head down + eyebrows up*
	Confirm target and go to destination	*eyebrows up*

posture with *eyebrows up* to compose not common gestures and provide good usability of the navigation assistant at the same time.

Figure 22 depictures a photo of the robotized wheelchair with some components of the navigation assisntat. We have fixed a regular webcam Logitech model C250 (1.3-megapixels) using a flexible arm which holds the camera in front of the user's head with approximately 30 cm of distance to provide images without cutting the face. The breadboard along with the *NodeMCU* development kit was fixed on the same side as the pan-tilt micro servos and powered with *5Vcc* from the notebook USB. The laptop is not present in this figure but it stands on the same adapted table as the *pan-tilt laser* module.

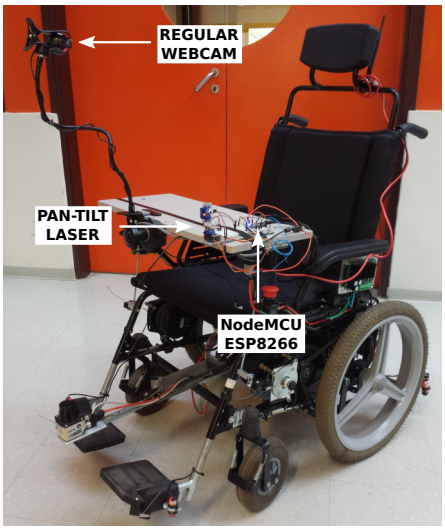


Figure 22 – Overview of subsystems integration to compose the navigation assistant for robotized wheelchairs.

In this section, we have discussed the system overview of the navigation assistant and how the modules communicate with each other. In Chapter 4 we have discussed the details of the *OpenFace* HMI and in the next two subsections (7.1.1 and 7.1.2) we will describe: both the hardware and software details of the *pan-tilt laser module*, followed by the *Matlab*

application which ties up all subsystems and possesses the main operation algorithm of the navigation assistant.

7.1.1 Pan-tilt Laser Subsystem

In Chapter 6 we have validated the pan-tilt laser pointer used to perform a semi-autonomous navigation in a corridor with obstacles. The obtained results have shown that the mathematical modeling presented in Sections 5.2 and 5.3 may compose a feasible solution to actuate on the micro servos as well as calculate the target position pointed by the laser with respect to the wheelchair. Thus, we have kept the base algorithm and improved the hardware and software to make the laser pointer a module to better integrate with OpenFace and Matlab application.

As shown in Figure 19a, the low-cost pan-tilt laser had been assembled with a diode laser and two *Micro servos 9g SG90 TowerPro* fixed on a hand-made aluminum structure. The electronic components were connected to an *Arduino Mega 2560* along with the IMU sensor to measure the user's head posture and control the micro servos. Also, the subsystem had push buttons to validate the target pointed by the laser dot as well as lock/unlock the actuation on the pan-tilt.

As we wanted to replace the old HMI (IMU + push buttons) by the hands-free solution OpenFace, we needed a board with wireless connectivity to integrate both subsystems. For this reason we have replaced the *Arduino Mega 2560* by the *Node MicroController Unit* (MCU) development kit (NODEMCU, 2014) containing a microchip *ESP8266* with built-in internet connectivity.

The *ESP8266* (Figure 23a) is a low-cost (about $ 2 USD) Wi-Fi System-on-a-Chip (SoC) with TCP/IP stack (Transmission Control Protocol/Internet Protocol) and a *32-bit microcontroller* produced by *Espressif Systems*. It comes with CPU of *80 MHz* (default) or *160 MHz*, *32 KiB* of instruction memory, *80 KiB* of user data memory and *16 GPIO* (General Purpose Input/Output) pin.

Although the *ESP8266* is an excellent choice to use as an embedded controller chip in mass-produced electronics, it is hard to access and to use in prototypes or experimental projects. Thus, instead of programming in low-level machine instructions and solder wires to accomplish simple experiments, the *NodeMCU* (Figure 23b) came to provide an open-source IoT (Internet of Things) platform built around the *ESP8266* for a cost of approximately $ 8 USD. The development kit board comes with a Lua based Firmware [1] (embedded Lua, also called eLua) built on top of the chip manufacturer's proprietary SDK (Software Development

[1] https://github.com/nodemcu/nodemcu-firmware

Kit) which provides a simple programming environment for prototyping. Also, the board has a built-in USB, a hardware reset button, Wi-Fi antenna, LED lights, and standard-sized GPIO pins to plug into a breadboard (IBM, 2018).

(a) (b)

Figure 23 – Hardware with built-in Wi-Fi used on the pan-tilt laser subsystem to facilitate integration with OpenFace and Matlab applications. (a) *ESP8266 12-E* Module, adapted from (AI-THINKER, 2017). (b) *NodeMCU* development kit built around *ESP8266* SoC, adapted from (NODEMCU, 2014).

At the end of 2015, the *ESP8266* Community released the called *Arduino core for ESP8266 WiFi chip* [2] which brings support to the Arduino environment by letting us flashing programs directly to the chip. This extension for Arduino IDE (Integrated Development Environment) comes with libraries to communicate over Wi-Fi using TCP and User Datagram Protocol (UDP) as well as many other features. Therefore, we have used this extension to adapted the algorithm used to obtain the results of Section **??** (written in Arduino programming language) and develop the new software architecture of the pan-tilt laser to run as a module and integrate with OpenFace and Matlab Application.

The Algorithm 1 describes the laser pointer operation to actuate on the servos and calculate the target position with respect to its frame. Firstly, we configure the *NodeMCU* to work as a station (abbreviated as STA in IEEE 802.11 (Wi-Fi) terminology) and connect to the wheelchair's *Access Point*. Then, we create two UDP socket servers to communicate with *OpenFace* and *Matlab* applications. The first one receives the user's head angles *yaw* and *pitch* (referred on Section 5.2 as $\boldsymbol{\psi_H}$ and $\boldsymbol{\theta_H}$, respectively) from the OpenFace application, and the second one communicates with *Matlab* application by receiving commands to *lock/unlock*

[2] https://github.com/esp8266/Arduino

the laser pointer and sending the calculated target coordinates pointed by the laser.

Algorithm 1: Pan-tilt laser pointer pseudo-code

```
1  connectToAccessPoint(ip, password);
2  startUdpSocketOpenFace(localPort123);
3  startUdpSocketMatlab(localPort456);
4  ptEnabled = false;
5  while true do
6      if ptEnabled then
7          yaw, pitch = readYPFromUdpSocketOpenFace();
8          if yaw and pitch then
9              pan, tilt = calcPTAngles(yaw, pitch) ;                    // Eq. 5.5 and 5.6
10             sPan, sTilt = fromDegreeToServoSignal(pan, tilt) ;        // Eq. 7.1 and 7.2
11             updatePTServos(sPan, sTilt);
12             target = calcTargetPos(sPan, sTilt);
13     matlabCommand = readCmdFromUdpSocketMatlab();
14     if matlabCommand = "lock" then
15         ptEnabled = false;
16     else if matlabCommand = "unlock" then
17         ptEnabled = true;
18     else if matlabCommand = "target" then
19         sendCmdToUdpSocketMatlab(target);
```

In line 9 we use Equations 5.5 and 5.6 to calculate $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$ based on the user's head angles $\boldsymbol{\psi_H}$ and $\boldsymbol{\theta_H}$ provided by *OpenFace*. Next, the function of line 10 uses Equations 7.1 and 7.2 to obtain the PWM servo signals $\boldsymbol{S_\psi}$ ($sPan$) and $\boldsymbol{S_\theta}$ ($sTilt$):

$$\boldsymbol{S_\psi} = k\boldsymbol{\psi_L} + shiftServoPan \tag{7.1}$$

$$\boldsymbol{S_\theta} = k\boldsymbol{\theta_L} + shiftServoTilt \tag{7.2}$$

The k is a sensitivity factor which can be adjusted to cause greater actuation on the servos signals as some users may have limited head's posture movements. We have used $k = 2$ to make it more sensible and allow the user to control the pan-tilt laser with smooth head movements. The $shiftServoPan$ and $shiftServoPan$ are defined on the calibration process and represent the zero reference signal values of the laser frame $\boldsymbol{L}$ (depicted in Figure 18). In our case, we have used $shiftServoPan = 86$ and $shiftServoTilt = 96$, considering the manner the servos have been fixed as well as an attempt to minimize the alignment errors introduced by the pan-tilt structure.

In line 12 the algorithm calculates the target position pointed by the laser using Equations 5.8 and 5.9 to obtain the estimated $\boldsymbol{\psi_L}$ and $\boldsymbol{\theta_L}$ angles (explained in Section 5.3)

and finally Equation 5.2 which gives $\boldsymbol{P_L}$.

The following lines of the pseudo-code (15, 17, 19) correspond to the interaction with *Matlab* application which sends commands to *lock/unlock* the laser pointer module or send the calculated target position to be reached by the wheelchair.The next subsection will discuss the *Matlab* application details and its algorithm.

7.1.2 Matlab Application Subsystem

The *Matlab* application communicates with all subsystems and possesses the algorithm to switch between the navigation modes (manual and semi-autonomous) as well as send high-level commands to actuate on the robotized wheelchair. We have used a Notebook PC with 8GB of RAM and CPU Intel Core i7 of 1.8GHz to run both *OpenFace* and *Matlab* application as they demand high computational cost. The Algorithm 2 shows the integrations with all subsystems and as well as the responsibilities of this module.

The first block of code makes the necessary configurations to communicate with the other subsystems and initializes some variables. In line 2 we set up the communication with the wheelchair by providing its host address and port, so the *Matlab* application can send high-level commands via HTTP requests to the *restThru* service embedded on the robotized wheelchair.

The *Matlab* applications reads the facial expressions and head's postures angles constantly (line 11) and depending on the combination of gestures shown in Table 5 we define the navigations mode by setting *navCmd* variable (line 12). When the user emits an *eyebrows up* the algorithm considers the *wcStatus* and triggers one of the three actions: requests the target position (line 20) to the *pan-tilt laser* or unlocks it (switch to *semi-autonomous mode* on line 23) or actuates manually on the wheelchair (line 25). If the user emits a *smile* command (line 26) the *Matlab* application sends one signal to stop the wheelchair (line 27) and a second one to lock the *pan-tilt laser* (line 30).

In *semi-autonomous mode*, when the *Matlab* receives the chosen target position $P_{\boldsymbol{L}}$ with respect to the *pan-tilt laser* frame, the algorithm applies Equation 7.3 to transform it to the wheelchair frame, followed by Equations 7.4 and 7.5 which gives us the necessary rotation angle $\boldsymbol{\psi_W}$ and linear displacement $\|P_{\boldsymbol{W}}\|_2$ (Euclidian distance) to reach the desired destination (line 34).

$$P_{\boldsymbol{W}} = D_{\boldsymbol{WL}} + P_{\boldsymbol{L}} = \begin{bmatrix} D_{\boldsymbol{WH}_x} + P_{\boldsymbol{L}_x} \\ D_{\boldsymbol{WH}_y} + P_{\boldsymbol{L}_y} \\ D_{\boldsymbol{WH}_z} + P_{\boldsymbol{L}_z} \end{bmatrix} \tag{7.3}$$

Algorithm 2: Matlab application pseudo-code

```
connectToAccessPoint(ip, password);
configureRestThruCommunication(wcIP, wcPORT);
startUdpSocketOpenFace(localPort789);
startUdpSocketPtLaser(localPort987);
ptEnabled = false;
targetRequested = false;
targetReceived = false;
reachedTarget = true;
wcStatus = 'STOPPED' ;                                                    // wheelchair status
while true do
    facialExp, yaw, pitch = readFacialExpAndYPFromUdpSocketOpenFace();
    navCmd = getNavCmd(yaw, pitch) ;          // SEMI-AUTONOMOUS or MANUAL(front, right, left)
    target = readTargetFromUdpSocketPtLaser();
    if target then
        targetReceived = true
    if facialExp = 'EYEBROWS_UP' then
        if wcStatus = 'STOPPED' and not targetRequested then
            if navCmd = 'SEMI-AUTONOMOUS' then
                if ptEnabled then
                    sendCmdToUdpSocketPtLaser('get_target') ;        // request target position
                    targetRequested = true;
                else
                    sendCmdToUdpSocketPtLaser('unlock') ;              // unlock pan-tilt laser
            else
                manualControlWheelchair(navCmd) ;            // go front, turn right, turn left
    if facialExp = 'SMILE' then
        http_post('stop') ;                                          // stop command to wheelchair
        wcStatus = 'STOPPED';
        reachedTarget = true;
        sendCmdToUdpSocketPtLaser('lock') ;                                 // lock pan-tilt laser
        targetRequested = false;
        targetReceived = false;
    if targetReceived then
        alpha, dislo = calcTarget(target); ;                          // Equations 7.3, 7.4 and 7.5
        reachedTarget = false;
        targetReceived = false;
    if not reachedTarget then
        if wcStatus = 'STOPPED' then
            http_post('heading', alpha) ;          // heading command to wheelchair (rotate alpha)
            wcStatus = 'HEADING'
        else if wcStatus = 'HEADING' then
            if http_get('heading') = 'finished' then
                http_post('dislocate', dislo) ;          // linear dislocation command to wheelchair
                wcStatus = 'LINEAR_DISLO';
        else if wcStatus = 'LINEAR_DISLO' then
            if http_get('dislo') = 'finished' then
                wcStatus = 'STOPPED';
                reachedTarget = true;
```

$$\boldsymbol{\psi_W} = \arctan \frac{P_{\boldsymbol{W}_x}}{P_{\boldsymbol{W}_y}} \tag{7.4}$$

$$\|P_{\boldsymbol{W}}\|_2 = \sqrt{P^2_{\boldsymbol{W}_x} + P^2_{\boldsymbol{W}_y} + P^2_{\boldsymbol{W}_z}} \tag{7.5}$$

After that, the last *if* block of code (starts in line 37) performs the *semi-autonomous* navigation by applying the calculated rotation and linear displacement. One should note that the algorithm must regularly check when these two steps finish, so it updates the wheelchair status and makes the navigation assistant available for the next moves.

7.2 Experimental Procedure

The modules of the navigation assistant have been developed separately and then appropriately integrated to provide a hands-free, safe and comfortable solution for wheelchair navigation. Thus, we have tested the assistant with the primary goal of evaluating the effectiveness of the solution when experimented by a healthy and well-trained user navigating in an indoor corridor with obstacles (same scene previously illustrated in Figure 6). Furthermore, the two secondary goals were: evaluate the advantages and disadvantages of each navigation mode separately; and demonstrate the gain in usability when *manual* and *semi-autonomous* modes can be chosen freely depending on the situation.

Thus, the experimental procedure has been divided in three navigation cases:

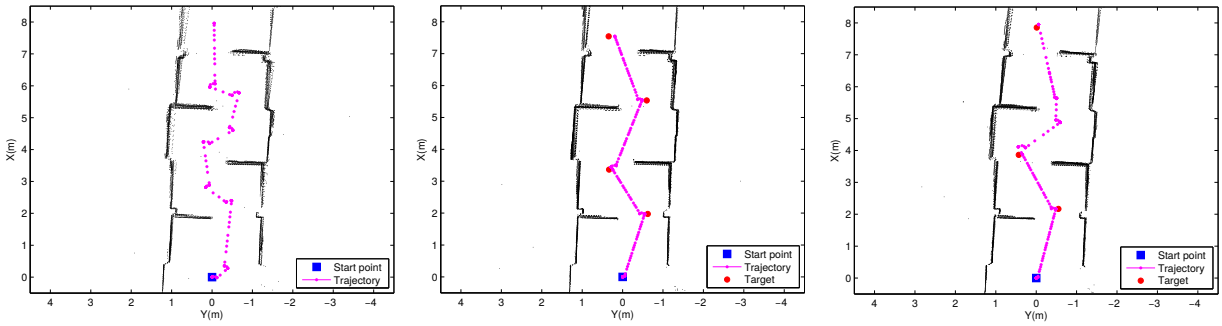
- Using only *manual* mode;
- Using only *semi-autonomous* mode;
- Combining *manual* and *semi-autonomous* mode according to the uses's desire.

For each one of cases, we have measured the lap time to navigate from start point to stop point as well as the number of emitted commands during the navigation. Furthermore, as we wanted to compare the best scenarios on each case, the user has performed several navigations to obtain a clean result without any facial expression misclassification or even wrong choice of targets which could lead the user to locking situations.

7.3 Results

Figure 24 depictures the trajectories obtained by navigating in a corridor with obstacles using manual, semi-autonomous and finally combining both modes. Our robotized

wheelchair is equipped with a laser rangefinder which measures the distances to objects (walls and obstacles) with respect to the *Start point* (initial reference). Also, the encoders installed on the rear wheels provide data for the robotized wheelchair to estimate the motion over time (odometry) as well as the position and orientation with respect to the initial reference.



(a) Manual (b) Semi-Autonomous (c) Semi-Auto. + Manual

Figure 24 – Comparison of navigation modes in a corridor with obstacles

Nav. Mode	No. Facial Exp.	Traj. Time (min)	Facial Exp. Period Avg. (sec)	Traj. Size (meters)
Manual	*32*	*02:45*	*5*	*10.13*
Semi-Auto + Manual	*17*	*03:28*	*12*	*9.10*
Semi-Autonomous	*12*	*03:37*	*18*	*8.38*

Table 6 – Comparison of three navigation scenarios with the assistant: using only manual mode; using only semi-autonomous mode; combining manual and semi-autonomous mode. For each navigation, we have measured: the number of emitted facial expressions, the time to perform the trajectory, the period average of facial expressions emission and the trajectory size.

Table 6 gathers the number of emitted facial expressions, the trajectory time, the period average of facial expression emission and the trajectory size considering each one of the three navigation scenarios.

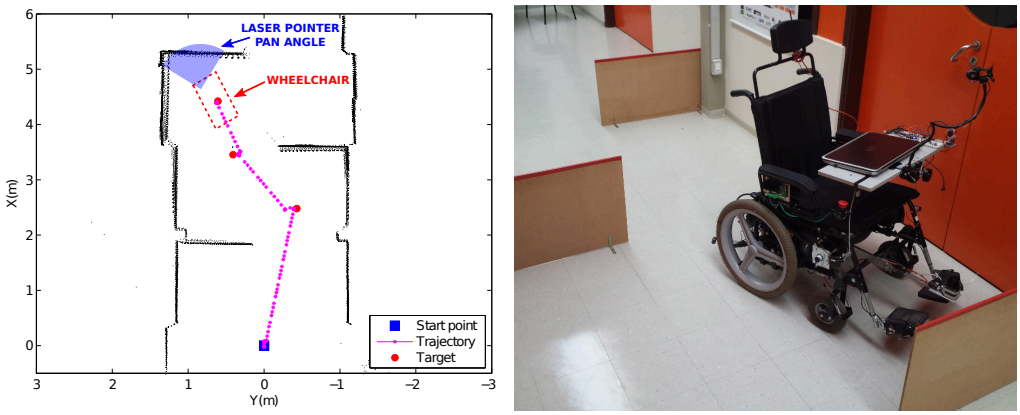
The navigation using only manual mode is illustrated in Figure 24a. The user has emitted *32 facial expressions* (*eyebrows up* and *smile*) to perform the trajectory in *02 minutes and 45 seconds* without any collision. The trajectory indicates a greater number of commands to pass through the obstacles, mainly pure rotations. Also, there is a delay of approximately *01 second* between the *smile* emission and the wheelchair stop which demands training to execute movements near obstacles.

Figure 24b depicts the navigation using only semi-autonomous mode. The user has emitted *12 facial expressions* (*eyebrows up* and *smile*) to perform the trajectory in *03 minutes and 37 seconds* without any collision. Although the trajectory indicates a more concise result with fewer commands than the manual mode, the user has spent more time choosing the

targets carefully on the ground to avoid blocking situations similar to the one illustrated in Figure 25. The semi-autonomous navigation requires more practice than the manual one as the user must develop a notion of space to avoid choosing insecure destinations or spots which lead to put the wheelchair in blocking situations.

Finally, Figure 24c illustrates the navigation combining manual and semi-autonomous modes. The user has emitted *17 facial expressions* (*eyebrows up* and *smile*) to perform the trajectory in *03 minutes and 28 seconds* without any collision. By letting the user choose the navigation mode according to his position on the corridor provides flexibility and avoid blocking situations as shown in Figure 25. The number of commands is greater than semi-autonomous and smaller than manual as well as the lap time, which indicates a well-balanced experience for the user. During the navigation, the operator has emitted around one facial expression every 12 seconds and may not feel as tired as the manual mode which demands more physical effort since the user needs to emit around one facial expression every 5 seconds.

In Figure 25 we have an example of a stressful and frustrating situation during navigation with the semi-autonomous mode. The trajectory map (Figure 25a) illustrates how the third chosen destination put the wheelchair in a region where the user found himself locked. He is unable to choose the next target without colliding to the obstacle due to the limited pan angle of the laser pointer. Even if the pan-tilt structure did not have this limitation, the user would have to over-rotate his head to choose the next destination, which is impracticable mainly for people with disabilities.



(a) (b)

Figure 25 – Locking situation using only the *semi-autonomous* navigation mode.

As future work, the implementation of a reverse camera with visual feedback could improve the user's experience in case of blocking situations. A third navigation mode could be integrated to the assistant by letting the operator send a new command to the wheelchair which makes it perform a reverse motion in order to take him/her out of the blocking region.

7.4 Chapter Conclusions

In this chapter, we presented the integration of the OpenFace HMI (Chapter 4) and the pan-tilt laser pointer (Chapter 5) to compose a low-cost navigation assistant for people paralyzed from down the neck to control a robotized wheelchair. The assistant system provides two navigation modes: manual and semi-autonomous. In the manual mode, the user must combine his/her head's posture with facial expressions (*eyebrows up* and *smile*) to actuate directly on the wheelchair movements (linear displacement, rotate right, rotate left and stop). In the semi-autonomous mode, the head posture controls the laser pointer fixed on the wheelchair and a *eyebrows up* triggers the dislocation until the desired location by combining a heading movement followed by a linear displacement.

The navigation modes have been tested individually in a corridor with obstacles, and results have shown that the combination of manual and semi-autonomous modes provide the best user experience as they have complementary particularities.

The semi-autonomous mode requires fewer commands than the manual mode and demands less effort for the user to reach the desired destination during navigation with obstacles. However, as our system does not provide obstacle avoidance, the semi-autonomous mode requires more practice and ability to navigate safely. Furthermore, the user can find itself locked when choosing targets close to obstacles, as the laser may not be able to point to a safe location. As mentioned in Chapter 6, a path planning and path following algorithm could improve the semi-autonomous mode considering the obstacles of the schene. The next chapter introduce a possible solution to integrate this project as a futire work.

The manual mode provides more flexibility and autonomy for the user to perform large linear displacements and pure rotations, which are handy for heading adjusts in specific situations: such as passing near to obstacles or even going through small doors. However, controlling the wheelchair manually all the time may be tiring as it demands more commands than the semi-autonomous mode. Moreover, heading orientation may require practice as the caster wheels introduce disturbance on simple rotation movements and there is a delay between the *smile* command and the wheelchair stop.

The combination of head postures with the facial expressions *eyebrows up* and *smile* may be a practical hands-free solution for the wheelchair navigation assistant. The commands are intuitive, and the transition between manual and semi-autonomous modes provides flexibility for the user to navigate on indoor environments. Furthermore, this combination tries to address the "Midas Touch" problem as the user must combine specific gestures to actuate on the wheelchair.

8 Future Works

Although we have developed a comfortable navigation assistant for robotized wheelchairs, the experiments with all subsystems integrated have indicated some improvements for our navigations assistant. One of them is the integration of a depth camera along with a terrain analysis algorithm to detect obstacles and calculate a safe trajectory based on the chosen destination. Also, a path following algorithm could improve the user experience when using the semi-autonomous navigation mode. Thus, this chapter discusses some initial results about this topic and points some related aspects as motivation for future works.

8.1 Traversability map

As mentioned in section 6 and confirmed in navigation experiments with all subsystems integrated (Section 7.3) an additional module composed by a depth camera and an analysis algorithm of the scene could improve our navigation assistant by considering obstacles during the semi-autonomous navigation. Thus, as future work we want to integrate the traversability map with the pan-tilt interface, making the laser beam continuous when pointing to a traversable location or blinking otherwise. This provides safety and additional information for the user to choose the reachable spots by the wheelchair. Furthermore, we want to use the traversability map to calculate a path planning and the control algorithm to perform an efficient trajectory.

Thus, to construct this map we need to integrate a depth camera to capture a 3D point cloud of the scene and use an algorithm such as the Unevenness Point Descriptor (UPD) available in (BELLONE *et al.*, 2014) to implement these two functionalities. The UPD is a visual algorithm for traversability assessment based on normal vectors analysis which provides a fast real-time implementation since it does not require any data processing or previously generated digital elevation map to classify the scene. The C++ UPD library calculates a traversability map from an RGB-D image, and the larger implications of this method reside in its applicability for path planning purposes in both indoor and outdoor environments. Hence, we want to integrate this algorithm with the pan-tilt laser to consult the traversability map in real-time while the user controls the laser dot, and after choosing the destination, we want to implement path planning and path following for the wheelchair navigation. Section 8.1.1 shows some partial results of the UPD library.

Additionally, the traversability map may provide information about the unevenness

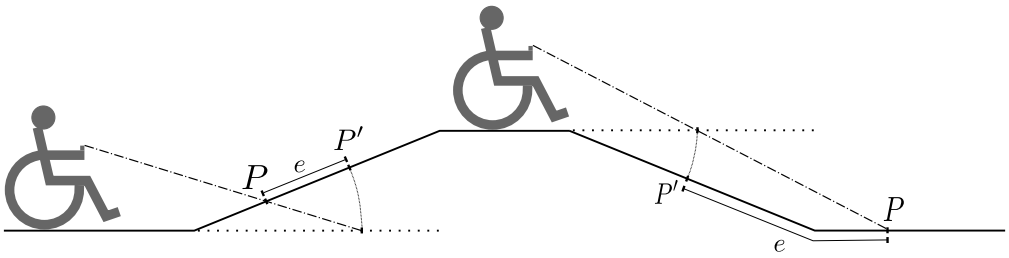


Figure 26 – Wheelchair navigation using the pan-tilt laser to point in a non-horizontal scene.

of the environment for navigation in non-horizontal terrains (e.g. scenes with ramps). The equations presented in sections 5.2 and 5.3 are valid only when the user points the laser in horizontal surfaces. Otherwise, as illustrated in Figure 26, the wheelchair may stop in a location P' far from the chosen target P. Moreover, the distance error e may be considerable depending on the inclination of the ramp and the surface geometry. Thus, as future work of this project, we intend to improve the navigation algorithm to operate correctly even when the user navigates in non-horizontal scenes.

8.1.1 Query point traversability assessment

Figure 27 illustrates the traversability analysis from an RGB-D image of the corridor with obstacles. Figure 27(a)(b) shows, respectively, a regular RGB image of the scene and the 3D point cloud obtained with Kinect and Point Cloud Library (PCL) (RUSU; COUSINS, 2011). Figure 27(c) illustrates the traversability map constructed from the same point cloud using UPD library. The algorithm labels the points with high values of ζ as traversable while others are considered not traversable. The ζ_k^q can be interpreted as a local inverse "unevenness index" since it assesses the degree of local roughness of each query point q in respect to its k neighbors (BELLONE *et al.*, 2014).

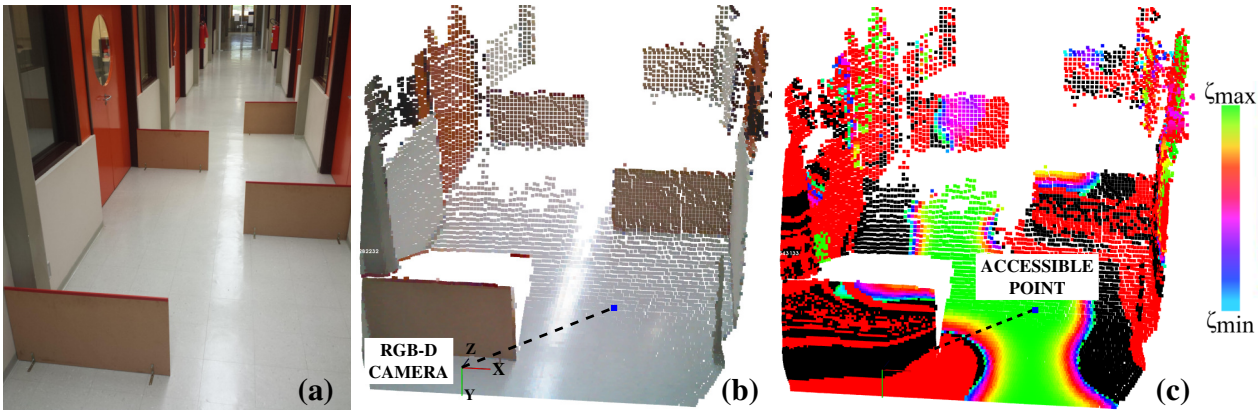


Figure 27 – Traversability analysis of a corridor with obstacles, which (a) is a regular RGB image of the scene, (b) is the 3D point cloud obtained from a RGB-D camera before the UPD calculation and (c) is the same point cloud after UPD processing.

As we want to integrate the traversability map with the pan-tilt interface in a real-time implementation, making the laser beamer continuous when pointing to a traversable location or blinking otherwise. The traversability map illustrated in Figure 27(c) has taken about 2 seconds to be calculated in a PC with 8GB of RAM and CPU Intel Core i7 of 1.8GHz. Thus, this process time makes the algorithm feasible to integrate the system, as we intend to run it as soon as the wheelchair stops. After this calculation, the time to assess the traversability of a query point in the 3D point cloud is insignificant, leading to a near real-time display of the laser dot's state (i.e., continuous red for accessible target or blinking one for unaccessible.)

8.2 Additional improvements

The obtained results with the navigation assistant have shown that both manual and semi-autonomous modes are effective when the wheelchair dislocates in a large corridor with obstacles. However, the user may need to cross doors or even dislocate in narrow corridors during navigation. Thus, a shared control algorithm could favor the user when navigating in small and difficult places. The integration of lateral distance sensors on the wheelchair along with obstacle a shared control algorithm could provide safety and convenience for the user when navigating on small environments, crossing opened doors or dislocating on narrow corridors.

The OpenFace application has been implemented to run in a notebook but could be adapted to run in a smartphone and enable integration with other subsystems.

The robotized wheelchair used in our experiments is equipped with emergency stop buttons. However, a sip and puff sensor could be integrated into the navigation assistant to provide an additional safety channel in case of collisions or unwanted movements of the chair.

9 Conclusions

Robotized wheelchair based mobility is of the trends in assistive robotics, mainly because not all persons possess the necessary cognitive and neuromuscular capacity needed to control regular power wheelchairs using a joystick and a set of buttons. Hence, this document presented a laser pointer driving assistant which provides an image-based HMI to capture the user's head posture and facial expressions to command a robotized wheelchair using manual and semi-autonomous navigation modes. Results have shown that the navigation assistant presented in Chapter 7 may be a possible solution to enhance the autonomy of people paralyzed from down the neck.

The *OpenFace* have shown to be an effective image-based HMI which provides a reliable estimation of the user's head orientation and three facial expressions (*smile*, *eyebrows up* and *blink*) to integrate the navigation assistant or be used for other applications. *OpenFace* is opensource and may integrate low-cost solutions since it works properly with regular webcams. The application works with different users and requires an initial calibration to get the necessary parameters to provide reliable facial expressions detection. Both head orientation angles and facial expressions are exposed to a UDP port which makes this HMI easy to use and to integrate.

We also developed the pan-tilt laser subsystem controlled by the user's head posture to point the desired location on the ground to be reached autonomously by the wheelchair. The initial results discussed in Chapter 6 presented the validation of the mathematical model for the pan-tilt servos control and the target calculation coordinates. The promising results motivated its integration to compose the navigation assistant and the results discussed in Section 7.3 demonstrated that our pan-tilt laser module can be controlled by the user's head orientation (estimated by *OpenFace*) and also provides a reliable estimation of the target's coordinates for the semi-autonomous navigation mode.

In Chapter 7 we discussed the integration of *OpenFace* module with the pan-tilt laser subsystem to compose our navigation assistant. Although the semi-autonomous navigation mode has shown to provide a comfortable experience for the user, we also developed the manual navigation mode which the user may control the wheelchair movements directly. Thus, the navigation assistant provides flexibility for the user to adjust his/her orientation and make specific movements with manual mode or even chose the target to be reached by the wheelchair (semi-autonomous mode) which is more comfortable and demands fewer commands than the manual mode. Also, manual mode may help the user to adjust his/her

orientation and avoid blocking situations for the semi-autonomous mode.

The navigation algorithm of the semi-autonomous mode implements a combination of heading movement with respect to the chosen target followed by a linear displacement. However, as suggested in Chapter 8 the integration of a scene descriptor with a path planning and path following algorithm would improve the navigation assistant and make the trajectories more efficient and safer.

Finally, we reinforce that both *OpenFace* and pan-tilt laser subsystem have been developed based on low-cost materials and open-source projects which makes it accessible for further investigations and improvements.

Bibliography

AI-THINKER. *ESP8266 12-E Module*. 2017. <https://www.ai-thinker.com/product/esp8266/>. Accessed 2018 Oct 13. Cited on page 47

BELLONE, M.; REINA, G.; Ivan Giannoccaro, N.; SPEDICATO, L. 3d traversability awareness for rough terrain mobile robots. *Sensor Review*, Emerald Group Publishing Limited, v. 34, n. 2, p. 220–232, 2014. Cited 2 times on pages 55 and 56

COWAN, R. E.; FREGLY, B. J.; BONINGER, M. L.; CHAN, L.; RODGERS, M. M.; REINKENSMEYER, D. J. Recent trends in assistive technology for mobility. *Journal of neuroengineering and rehabilitation*, BioMed Central, v. 9, n. 1, p. 20, 2012. Cited on page 12

EKMAN, P.; FRIESEN, W. *FACS: Facial Action Coding System*. 1978. <https://www.cs.cmu.edu/~face/facs.htm>. Accessed 2018 Nov 18. Cited on page 24

EKMAN, P.; FRIESEN, W. *Investigator's guide to the facial action coding system. Palo Alto*. [S.l.]: CA: Consulting Psychologists Press, 1978. Cited on page 23

ESCOBEDO, A.; SPALANZANI, A.; LAUGIER, C. Multimodal control of a robotic wheelchair: Using contextual information for usability improvement. In: IEEE. *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. [S.l.], 2013. p. 4262–4267. Cited on page 13

FARIA, B. M.; REIS, L. P.; LAU, N. A survey on intelligent wheelchair prototypes and simulators. In: *New Perspectives in Information Systems and Technologies, Volume 1*. [S.l.]: Springer, 2014. p. 545–557. Cited on page 12

GAUTAM, G.; SUMANTH, G.; KARTHIKEYAN, K.; SUNDAR, S.; VENKATARAMAN, D. Eye movement based electronic wheel chair for physically challenged persons. *Int. J. Sci. Technol. Res*, v. 3, n. 2, 2014. Cited on page 13

HAMEDI, M.; SALLEH, S.-H.; ASTARAKI, M.; NOOR, A. M. Emg-based facial gesture recognition through versatile elliptic basis function neural network. *Biomedical engineering online*, BioMed Central, v. 12, n. 1, p. 73, 2013. Cited on page 15

HAMEDI, M.; SALLEH, S.-H.; NOOR, A.; TAN, T.; KAMARUL, A. Comparison of different time-domain feature extraction methods on facial gestures' emgs. In: *Prog. Electromagn. Res. Symp. Proc*. [S.l.: s.n.], 2012. v. 12, p. 1897–1900. Cited on page 15

HUO, X.; GHOVANLOO, M. Evaluation of a wireless wearable tongue–computer interface by individuals with high-level spinal cord injuries. *Journal of neural engineering*, IOP Publishing, v. 7, n. 2, p. 026008, 2010. Cited on page 13

IBM. *Getting to know NodeMCU and its DEVKIT board.* 2018. <https://developer.ibm.com/tutorials/iot-nodemcu-open-why-use/>. Accessed 2018 Oct 13. Cited on page 47

ITURRATE, I.; ANTELIS, J. M.; KUBLER, A.; MINGUEZ, J. A noninvasive brain-actuated wheelchair based on a p300 neurophysiological protocol and automated navigation. *IEEE Transactions on Robotics*, IEEE, v. 25, n. 3, p. 614–627, 2009. Cited on page 12

JúNIOR, A. *Robotização de uma cadeira de rodas motorizada: arquitetura, modelos, controle e aplicações.* Dissertação (Mestrado) — School of Electrical and Computer Engineering, FEEC, UNICAMP., Campinas, Brazil, 2016. Cited on page 39

KIM, J.; PARK, H.; BRUCE, J.; SUTTON, E.; ROWLES, D.; PUCCI, D.; HOLBROOK, J.; MINOCHA, J.; NARDONE, B.; WEST, D. *et al.* The tongue enables computer and wheelchair control for people with spinal cord injury. *Science translational medicine*, American Association for the Advancement of Science, v. 5, n. 213, p. 213ra166–213ra166, 2013. Cited on page 13

KYRANOU, I.; KRASOULIS, A.; ERDEN, M. S.; NAZARPOUR, K.; VIJAYAKUMAR, S. Real-time classification of multi-modal sensory data for prosthetic hand control. In: IEEE. *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*. [S.l.], 2016. p. 536–541. Cited on page 15

LIEVESLEY, R.; WOZENCROFT, M.; EWINS, D. The emotiv epoc neuroheadset: an inexpensive method of controlling assistive technologies using facial expressions and thoughts? *Journal of Assistive Technologies*, Emerald Group Publishing Limited, v. 5, n. 2, p. 67–82, 2011. Cited on page 16

LONG, J.; LI, Y.; WANG, H.; YU, T.; PAN, J.; LI, F. A hybrid brain computer interface to control the direction and speed of a simulated or real wheelchair. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, IEEE, v. 20, n. 5, p. 720–729, 2012. Cited on page 12

MILLÁN, J. d. R.; RUPP, R.; MUELLER-PUTZ, G.; MURRAY-SMITH, R.; GIUGLIEMMA, C.; TANGERMANN, M.; VIDAURRE, C.; CINCOTTI, F.; KUBLER, A.; LEEB, R. *et al.* Combining brain–computer interfaces and assistive technologies: state-of-the-art and challenges. *Frontiers in neuroscience*, Frontiers, v. 4, p. 161, 2010. Cited on page 12

MOTA, S. Human-computer interface using facial expressions: a solution for people with motor disabilities. In: *4th BRAINN Congress*. Campinas, Brazil: [s.n.], 2017. Cited 2 times on pages 18 and 19

NASA. *NASA-TLX*. 2011. <http://www.nasatlx.com/>. Accessed 2017 Jun 10. Cited on page 20

NODEMCU. *An open-source firmware and development kit that helps you to prototype your IOT product within a few Lua script lines*. 2014. <http://nodemcu.com/index_en.html>. Accessed 2018 Oct 10. Cited 2 times on pages 46 and 47

PEREIRA, G.; MOTA, S.; ROHMER, E. Comparison of human machine interfaces to control a robotized wheelchair. In: UFRGS/PUCRS. *XIII Simposio Brasileiro de Automação Inteligente (SBAI)*. [S.l.], 2017. p. 2301–2306. Cited 2 times on pages 18 and 19

PREMERLANI, W.; BIZARD, P. Direction cosine matrix imu: Theory. *Diy Drone: Usa*, p. 13–15, 2009. Cited on page 33

REAZ, M. B. I.; HUSSAIN, M.; MOHD-YASIN, F. Techniques of emg signal analysis: detection, processing, classification and applications. *Biological procedures online*, v. 8, n. 1, p. 11, 2006. Cited on page 15

RECHY-RAMIREZ, E.-J.; HU, H.; MCDONALD-MAIER, K. Head movements based control of an intelligent wheelchair in an indoor environment. In: IEEE. *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*. [S.l.], 2012. p. 1464–1469. Cited on page 16

ROHMER, E.; PINHEIRO, P.; CARDOZO, E.; BELLONE, M.; REINA, G. Laser based driving assistance for smart robotic wheelchairs. In: IEEE. *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. [S.l.], 2015. p. 1–4. Cited 3 times on pages 14, 33, and 37

ROHMER, E.; PINHEIRO, P.; RAIZER, K.; OLIVI, L.; CARDOZO, E. A novel platform supporting multiple control strategies for assistive robots. In: IEEE. *Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on*. [S.l.], 2015. p. 763–769. Cited on page 14

ROHMER, E.; SINGH, S. P.; FREESE, M. V-rep: A versatile and scalable robot simulation framework. In: IEEE. *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. [S.l.], 2013. p. 1321–1326. Cited 2 times on pages 14 and 37

RUSU, R. B.; COUSINS, S. 3D is here: Point Cloud Library (PCL). In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: [s.n.], 2011. Cited on page 56

SOUZA, R.; PINHO, F.; OLIVI, L.; CARDOZO, E. A restful platform for networked robotics. In: IEEE. *Ubiquitous Robots and Ambient Intelligence (URAI), 2013 10th International Conference on*. [S.l.], 2013. p. 423–428. Cited on page 19

TOMARI, R.; KOBAYASHI, Y.; KUNO, Y. Enhancing wheelchair manoeuvrability for severe impairment users. *International Journal of Advanced Robotic Systems*, SAGE Publications, v. 10, n. 2, p. 92, 2013. Cited on page 12

WOLPAW, J. R.; BIRBAUMER, N.; MCFARLAND, D. J.; PFURTSCHELLER, G.; VAUGHAN, T. M. Brain–computer interfaces for communication and control. *Clinical neurophysiology*, Elsevier, v. 113, n. 6, p. 767–791, 2002. Cited on page 13