Feedbot: A Vision Augmented Feeding Robot

Alexandre Manuel Rodrigues Isidoro Candeias

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Doutor Manuel Ricardo De Almeida Rodrigues Marques
Prof. João Paulo Salgado Arriscado Costeira

Examination Committee

October 2018

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

A conclusão desta tese representa o terminar de uma das etapas mais desafiante e gratificante da minha vida. Ao longo destes 5 anos fiz do IST a minha casa, conheci inúmeras pessoas e aprendi mais do que alguma vez poderia esperar.

O primeiro agradecimento vai para os meus orientadores, o Dr. Manuel Marques e o Prof. João Paulo Costeira. Ao longo do último ano tive o prazer de trabalhar com estas duas pessoas fantásticas. O Manuel irá para sempre ficar marcado na minha vida como um exemplo a nível pessoal e académico, e para além de ser meu orientador irá para sempre ser um amigo. Ao Prof. JPC agradeço todo o entusiasmo e inspiração que sempre me transmitiu para os problemas de Visão, e por ser, sem dúvida, um exemplo de como dever ser um professor e um investigador. Aos 2, um obrigado é pouco e espero que o melhor do nosso trabalho esteja para vir!

Durante a realização desta tese tive o prazer de visitar a Carnegie Mellon University e trabalhar em três grupos de investigação diferentes (SIPG, HARP e CORAL). Esta experiência mudou sem dúvida a maneira como vejo a investigação, e quem sabe a minha vida. Estarei para sempre grato aos meus Orientadores, ao ISR-IST e ao IST por me terem proporcionado esta experiência!

Agradeço à Prof. Manuela Veloso, por fazer parte do comité desta tese e por me ter acolhido em Pittsburgh no seu CORAL lab. Irá ser para sempre uma fonte de inspiração e rigor e irei sempre sentir-me privilegiado por ter tido a oportunidade de trabalhar consigo. Obrigado!

I would also like to thank Prof. Henny Admoni and all people from HARP Lab and CORAL Lab, for providing their robots and making Pittsburgh feel like home. It was definitely a great experience and I hope that someday I can return.

A special thanks to my collaborator and friend Travers and his wife Taylor, who welcomed and treated me like family during my stay in the US. Without Travers's knowledge about robotics, this thesis would not be possible. Travers and Taylor, thank you!

Um agradecimento também à fundação EDP que financiou durante 2 anos parte dos meus estudos no IST através da bolsa EDP-Solidária e à Câmara Municipal de Almodôvar que tanto tem feito pelos estudantes da nossa terra.

À Prof. Raquel Forca, por insistir comigo para entrar no IST e por fomentar espírito científico em todos os seus alunos, por mais novos que sejam.

Gostaria também de agradecer a todos os meus amigos, em especial aos meus amigos de Almodôvar, aos meus amigos da RDP e aos meus amigos do curso de MEEC. Obrigado pelas brincadeiras, almoços, jantares, saídas, bons e maus momentos que passámos juntos nos últimos 5 anos.

Um obrigado à minha família, em especial aos meus avós, António e Júlia, às minhas tias Nélia e Isabel e ao meu primo Eduardo, por todo o apoio, amor e carinho que sempre me deram.

E finalmente, à minha mãe Maria da Piedade, a pessoa mais importante da minha vida, que sempre me amou incondicionalmente e me colocou à frente de tudo e todos. Embora por vezes eu não o retribua, a ti devo-te tudo o que sou hoje e este trabalho também é teu!

This work was funded by FCT grant [UID/EEA/50009/2013] and FCT project IF/00879/2012 (Projecto Exploratório) and we thank CMU-Portugal program for their partial support of this work through an ERI grant.

Abstract

People with disabilities are unable to execute several simple daily tasks, such as feeding or dressing, and often rely on other people to help them. Researchers have over time developed robotic feeding assistants to help at meals so that people with disabilities can live more autonomously. However, current commercial feeding assistant robots acquire food without feedback on acquisition success and deliver it in a preprogrammed location.

We propose Feedbot, an autonomous feeding robot arm that is augmented with vision to be capable of adapting his trajectories in real-time to the user and of having visual feedback on the food acquisition. Feedbot was tested in a real meal scenario and using two different robot arms.

We show how Discriminative Optimization (DO) can be applied to track the user's head pose so that the food can be effectively brought all the way to the user's mouth, rather than to a preprogrammed feeding location. Our results show the ability of DO to track the head pose, while achieving state of the art performance. Finally, we show that visual feedback can improve the effectiveness of food acquisition.

Keywords

Assistive Technologies, Manipulation Aids, Feeding Assistance, Computer Vision, Head Pose Tracking

Resumo

Pessoas que vivem com deficiências são impedidas de fazer as suas vidas normais, confiando em outras pessoas para realizar tarefas simples, como alimentar-se ou vestir-se. Investigadores desenvolveram ao longo do tempo assistentes de alimentação robótica para ajudar nas refeições, para que pessoas com deficiências possam ter uma vida mais autônoma. Os atuais robôs assistentes de alimentação comerciais adquirem alimentos sem feedback sobre o sucesso de aquisição e movem-se para um local pré-programado para fornecer os alimentos ao utilizador.

Neste trabalho, propomos o Feedbot, um braço robô de alimentação autônomo que é ampliado com visão para ser capaz de adaptar as suas trajetórias em tempo real ao utilizador e também melhorar a quantidade de alimento entregue ao utilizador. Feedbot foi testado num cenário de refeição real e com dois braços robô diferentes.

Mostramos como DO pode ser usado para fazer tracking da pose da cabeça do utilizador, de modo que a comida possa ser levada até à boca do utilizador, e não para um local de pré-programado. Os nossos resultados mostram que DO é capaz de alcançar um desempenho semelhante ao state of the art no rastreamento da pose da cabeça. Por fim, mostramos que o uso de feedback visual na captação de alimentos aumenta a eficiência da mesma.

Palavras Chave

Tecnologias Assistivas, Auxiliares de Manipulação, Assistência de Alimentação, Visão por Computador, Tracking de Pose da Cabeça

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ICP	Iterative Closest Point
DO	Discriminative Optimization
SUM	Sequence of Update Maps
SIFT	Scale Invariant Feature Transform
ROI	Region of Interest
MTCNN	Multi Task Cascaded Convolutional Neural Network
RANSAC	Random Sample Consensus
SVM	Support Vector Machine
DOF	Degrees of Freedom
URDF	Unified Robot Description Format
ADLs	Activities of Daily Living
EEG	Electroencephalography
GUI	Graphical User Interface
HMM	Hidden Markov Model
CNN	Convolutional Neural Network
ROS	Robot Operating System
MAE	Mean Absolute Error
GPU	Graphics Processing Unit
SIFT	Scale-Invariant Feature Transform
SVD	Singular Value Decomposition

1 Introduction

Contents

1.1 Motivation

The growth of the elderly population is one of the biggest problems that humanity has to face. Reports from the United Nations say that the elderly population in the world in 2017 was about 962M people and it is likely to increase more than double in 2050 to 2.1B [1]. Furthermore, if we compare the percentage of the population above 65 years in the total population since 1988, [2] we can see, as it is present in figure 1.1, that it is always increasing.

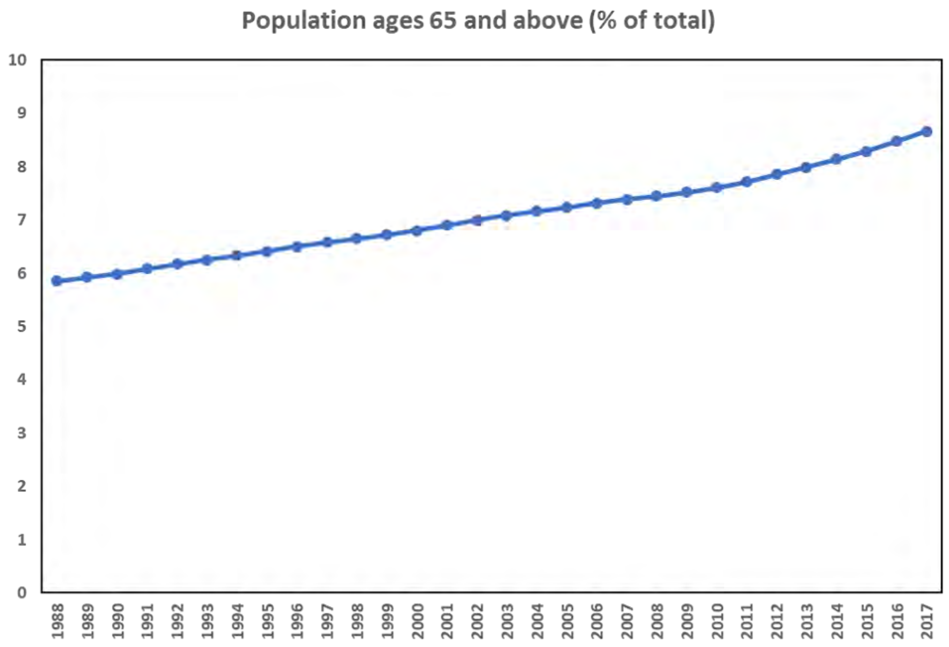


Figure 1.1: Percentage of People above 65 years in the total population since 1988. Data from [2].

Older people are more likely to need health care or caregiver services. If we add to these numbers the number of people that live with disabilities, like cerebral palsy or paralysis, we see that in the next years we will have a high demand in healthcare and caregiver services. This puts new challenges in our society since there are not enough people to take care of the incapable or the older ones.

Some people that live with strong disabilities are incapable of doing their Activities of Daily Living (ADLs). These activities are simple self-care tasks such as feeding, bathing, dressing or shaving. Since these people are not capable of doing these tasks by themselves, they need help from other people like caregivers or nurses. Helping people in their ADLs can be a difficult and tedious task to the caregiver. People living with disabilities are also less likely to live autonomously.

Robots can play a significant role in helping society to deal with these two problems. On one hand, they can give more autonomy to people who suffer from disabilities, on the other hand, robots will free caregivers of certain ”mechanical” tasks so that they can concentrate on the relational role.

Using a robot arm, people with upper-extremity disabilities can live more autonomously. For example in [3] it was studied how using a robotic arm, controlled through a joystick, can help people in their ADLs. Using a joystick-controlled robot arm was found extremely helpful since it gives back to the users some part of their autonomy. However, it is difficult to control a robot arm with a joystick and performing simple tasks like feeding. Using a joystick-controlled robot arm requires training and can take several minutes. Another disadvantage of using a joystick for controlling the robot arm is that some people, due to their disabilities, may be incapable of using it. To overcome these issues and empower disabled people with more autonomy, the robots must be autonomous and able to perceive the environment and adapt their behavior to each user.

In this work, we will focus on the task of autonomously feeding a user. Feeding is an essential ADLs and was reported by the users to be one of the most difficult tasks to execute controlling a joystick [3]. Feeding involves bringing the food from the plate location to the user's mouth. Even though this can be thought as being a simple task, it is not and it can be even more challenging for a robot, since it requires a perception of the environment and of the user's pose. The perception of the environment is required because the robot must be capable of detecting where the food is located and to scoop the food from the plate. At the same time, the robot must also be capable of detecting changes in the environment, for example, if there is food present on the plate. The perception of the user's pose is also important since the robot should move the end-effector to the mouth position and adapt its trajectories to different users that may have involuntary and diverse amplitudes of movements.

A first step in the direction of having a full autonomous feeding robot is done in our work. We propose the introduction of a vision system to achieve a full autonomous feeding robot that is capable of adapting his trajectories to the user's movements and of having feedback on the food acquisition.

1.2 Contributions

This thesis has major contributions. In particular:

Autonomous Feeding Robot (Feedbot). We built a fully autonomous robotic system that is capable of:

1. Acquiring food from a static plate using a demonstration of a scooping trajectory;
2. Detecting if the acquisition of the food was successful;
3. Real-time tracking of the user's mouth position;
4. Real-time adapting the trajectory of the robot end-effector from the plate to the user's mouth.

RGB-D head pose tracking system. We propose a tracking system with the ability of tracking the user's head at 28FPS and has the same accuracy as state of the art methods. This system makes it possible to track the user's mouth with a MAE below 1cm. Furthermore, our tracking system is very robust to extreme head pose variations being capable of detecting the correct head pose even in the case where the head pose is in profile.

RGB based food acquisition feedback system. Using a RGB camera we developed a food acquisition feedback system that is capable of detecting if there is food on the spoon or how much food is present. Real experiments show that this system provides significant improvements in terms of the food delivered to the user per distance traveled.

Feedbot allows to feed different users with different types of food. Moreover, our system is fully independent of the robot arm platform. We tested our system in two different robot arms, namely, Kinova MICO and Nyrio One [4].

1.3 Outline

The outline of this thesis is as follows. In chapter 2, we present an extensive study of the state of the art for assistive robotics evaluating the currently available feeding platforms. We also present a brief review of the computer vision techniques that can be employed on feeding robots. In chapter 3, we propose an autonomous feeding system (Feedbot). Chapter 4, describes the methods used to empower Feedbot with vision. In chapter 5, we present results of the performance for each part of Feedbot and results of a small real meal scenario experiment. Finally, in chapter 6, we present the conclusions of our work and we propose future directions to improve Feedbot.

2

State Of The Art

Contents

2.1 Assistive Feeding Robots

An assistive robot is a robot that provides aid or support to the human user [5]. This support can be through social interaction, for example, when using robots for cognitive stimulation exercises with elderly people [6]. Support in the form of physical interaction is also in the scope of assistive robotics, for example, robots that help the humans on ADLs. Researchers have made efforts in this type of assistive robots, making possible to use robots in some of these tasks like dressing [7] [8], shaving [9] and drinking [10].

One of the most important ADLs and the focus of this work is the task of feeding. The first effort to bring a fully capable robot arm to help people with upper arm disabilities was Handy1 [11]. Handy1 was specially built for people with cerebral palsy and it was tested with more than 50 users. With this robot, users could choose between 7 different food types using input buttons and the robot could move the food all the way from the plate to a preset mouth location. Although this robot was widely used and helpful, it could only move to a predefined location and it could not be used by people who cannot control the input buttons.

To overcome the problem of controlling the robot by this kind of users, in [12] the authors propose a new robot where the control is done by moving a pointing laser with the head. The authors report that users found their system to be easy and comfortable to use. Moreover, their system was reported to take an average time of 20 minutes per meal.

Following the developments in research, feeding assistant robots like Obi [13], MySpoon [14], MealBuddy [15], iEAT [16] and Bestic [17] became available for commercialization. All those platforms have some of the features present in Handy1 [11] and [12]. They have some type of user input based on hand or head controlled buttons and, after some user input or time-based event, they move the end-effector to a preset position. In short, they suffer from the same non-fulfillment as Handy1, since they cannot adapt if the user changes its pose during the meal or if the user is incapable of controlling the robot. However, Obi has some new feature that distinguishes it among other feeding robots. Obi can adapt the end-effector trajectory to each user but it requires a previous calibration step. Even if this feature is not real-time trajectory planning from the plate to the user's mouth, it allows some adaptation of the robot to each specific user requirements. To better illustrate how these commercial robots look like, figure 2.1 shows each one of these robots.

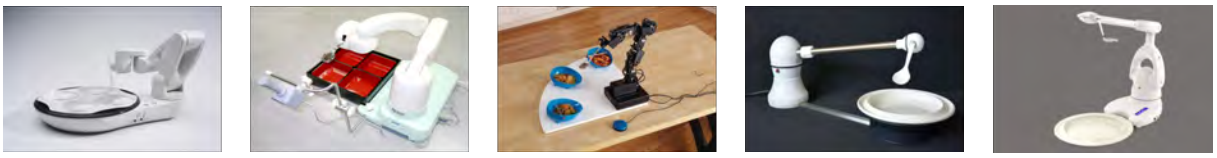


Figure 2.1: Commercial feeding robots. From left to right: Obi, MySpoon, MealBuddy, iEAT and Bestic

A recent review on robot feeding assistants research is described on [18]. Authors make an extensive analysis of robot feeding assistants, and present new robots in which the input from the user is based on eye tracking [19] or voice commands [20]. This work points out the same flaws as we described before. The current robotic feeding assistants have a lack of adaptation to the user and these robotic platforms cannot adapt to the user or the environment in real-time. Moreover, some of them are dependent on user input signals that can cause difficulties to some types of users. Authors conclude with some suggestions that can be used on new feeding robots. These suggestions are based in the inclusion of new user input signals such as Electroencephalography (EEG), they also point the importance of new robots to include visual perception to track the user's motion using, for example, a Kinect camera.

In [10] it is proposed a system that uses EEG signals and computer vision to help a user in the task of drinking using a robot arm. This system uses EEG signals to retrieve the user commands and a Kinect camera to track the user's mouth. EEG signals are used to give the start/stop command to the robot arm when performing each of the sub-tasks for drinking. The task of drinking was divided into various sub-tasks like picking up the cup or moving it to near the mouth. To achieve the mouth position, the robot plans a trajectory from the table to the current user's mouth point, this mouth point is given by a tracking system based on Viola-Jones face detector [21]. The mouth tracking system was evaluated with 6 subjects and the authors report an average error below 0.94cm in all the experiments. The whole system was also evaluated with one subject trained in using EEG signals. Using this system, the subject took between 2-3 minutes to take the cup from the table, drink a sip of water and put the cup back to the table.

In [22] it is proposed a feeding robotic system that uses EEG and a 4Degrees of Freedom (DOF) robot arm with a RGB camera attached to the end-effector. The user can choose between 3 different types of food using EEG signals. The robot arm is aligned with the user's mouth using the camera and doing visual servoing based on mouth detection, which uses Haar cascade classifiers. After the alignment, the vision system detects if the mouth is open and the robot arm moves from the resting position to the mouth. The vision system was tested on 6 subjects, the visual servoing was reported to take 10 seconds to align the mouth and the open/closed mouth classifier achieved 100% accuracy. The system was successfully tested in a real meal situation using 1 subject. However, the presented results are not enough to completely evaluate the vision servoing system, since with a RGB camera and the proposed methods it is very hard to detect the depth of the mouth and have the full 3D location of the mouth point.

The adoption of EEG signals as user input to control the robot can be very useful for users that cannot control a joystick/keyboard buttons, for example, users that cannot move their fingers. However, the training for using these signals is required and sometimes is difficult to learn how to use them. Furthermore, a device has to be attached to the user's head to measure these signals which can be uncomfortable.

General purpose mobile robots, like PR2 [23], were also used in the development of feeding platforms. In [24] is proposed a feeding platform using PR2. This robot was controlled by the user using a web-based Graphical User Interface (GUI). By using a GUI the user could control the start and stop command of feeding sub-tasks, like scooping or re-scooping or the command to start the movement from the plate to the mouth. A depth camera is used to track the user's head and have access to 3D position of the mouth. The tracking is done using an ARtag attached to the user's head. Results when feeding yogurt to 5 different participants showed that the robot is capable of feeding a user at a rate of 2 minutes per scoop. Using the same robot, research in multimodal anomaly detection was also done in [25]. This anomaly detection uses auditory, haptic, and visual signals gathered from different sensors present on PR2. These signals were used to train a Hidden Markov Model (HMM) and then new data is classified into anomalous or non-anomalous. They tested their framework for anomaly detection in feeding tasks to detect, for example, when the spoon missed the user in the delivery of the food or when the scooping failed.

Using general purpose mobile robots moving the feeding system to a different location would be very easy and the same robot could be used for different tasks beyond feeding. However, general purpose mobile robots are expensive and not yet available to common users at home. Another difficulty that arises when using this type of robots is using the feeding robot outside a home, for example in a restaurant.

Close research to the one that we present in this work was done in [26]. The authors propose to use DO [27] to track the user's head. However, the results do not report the accuracy of the tracking system or the time performance and do not test the whole platform in a real feeding environment.

Food Manipulation for feeding tasks was also investigated in [28], [29] and [30]. In [28] it is proposed a method for adapting non-optimal trajectories shown by the user to the optimal ones. This work can be used, for example, on adapting trajectories for Obi arm. An extensive dataset of people feeding different types of food to a static head model was gathered in [29]. This dataset provides information of the full 6DOF fork trajectories as well as the position of the head model.

Herlant's thesis [30] was focused on the strategies of food acquisition from the plate and on bite time prediction using social cues from group meal situations. For the food acquisition, Herlant used a strategy based on the prediction of the best pick up location using a Convolutional Neural Network (CNN) that takes as input a RGB-D image from the plate. Based on facial expressions features and audio acquired in real group meals, Herlant tried to predict the bite timing using HMM. Results showed that this bite time prediction had errors of less than 2 seconds in 90 % of the cases. The full system was tested on feeding different people on group meal situations, for that purpose a Kinova MICO arm was used and the food was moved from the plate to a pre-defined mouth position. When testing the whole system the

Robot	Path Planer	User MotionTracker	User Input Control	Type of Robot
Handy1 [11]	Predefined	No	Button	Research
Ishii et al [12]	Predefined	No	Head-Laser	Research
Obi [13]	User-Calibrated	No	Button	Commercial
MySpoon [14]	Predefined	No	Button	Commercial
MealBuddy [15]	Predefined	No	Button	Commercial
iEAT [16]	Predefined	No	Button	Commercial
Bestic [17]	Predefined	No	Button	Commercial
iCRAFT [19]	Predefined	No	Eye-Tracking	Research
Perera et al [22]	Real-Time	Yes	EEG	Research
Park et al [24]	Real-Time	Yes	Touch-GUI	Research
Herlant [30]	Predefined	Yes	Social Cues	Research

Table 2.1: Feeding Robots Summary

focus was on discovering if the bite timing prediction was good for the user in a social meal situation, which was reported to be better than an evenly-space bite timing.

We present in table 2.1 a summary of each one of the analyzed feeding robots. Each robot is classified in terms of the path planner, user motion tracker, user input control and the type of robot. The path planner can be: predefined, if the robot can only move to a predefined mouth position; user-calibrated, if the robot gives possibility to the user calibrate the mouth's position; or real-time adjusted, if the robot has some way of planing the motion from the plate to the mouth's position in real-time during the feeding. We also classify if the robot has some vision system for tracking the user's motion and what type of user input control it uses to give instructions to the robot. Finally, we classify if the robot is a commercial device or a research device.

2.2 Computer Vision Techniques for Feeding Robots

In our work, we are interested in building an autonomous robot capable of feeding a person. To make our Feedbot aware of the environment that is surrounding it, we rely on computer vision. More than the environment, in our work, we are interested in being able to know the head pose of the person that is being fed and the mouth's location. In this section, we will make a brief review of the computer vision methods applied to this problem.

One way of localizing the face of the user is using a RGB image and a face detector to localize the face in the image. Several methods have been proposed in this area of face detection. One of the first works used a neural network to compute the bounding box of the faces present in the image [31]. Following this work, Viola and Jones [21] proposed a cascade classifier to classify different patches in the image. Viola and Jones face detector is still one of the most used face detectors due to the good performance in terms of speed and accuracy.

More recently, deep learning face detectors are the state of the art in face detection on RGB image.

Works like MTCNN [32] or Face R-CNN [33] make use of CNN's to achieve great accuracy and real-time performance. However, this performance comes with a high computational cost and, sometimes, a Graphics Processing Unit (GPU) is needed.

The aforementioned methods are capable of detecting a bounding box in a RGB image that corresponds to a face. However, these methods are not robust to non-frontal views of the face and they do not solve the problem of obtaining the head pose of the user and the mouth's location. Even if we use these methods and know the 3D location of the pixels corresponding to the bounding box, for example, using a RGB-D depth sensor, is not clear which point is the corresponding mouth's point. Nonetheless, these methods can provide a good initialization to more complex and refinement methods that can solve our problem.

Another problem widely studied by the Computer Vision community, is the facial landmark detection and alignment problem [34] [35] [36] [37] [38] [39]. This problem is related to the alignment of pre-defined landmark points to the face that is present on the RGB image. Figure 2.2 illustrates the concept.

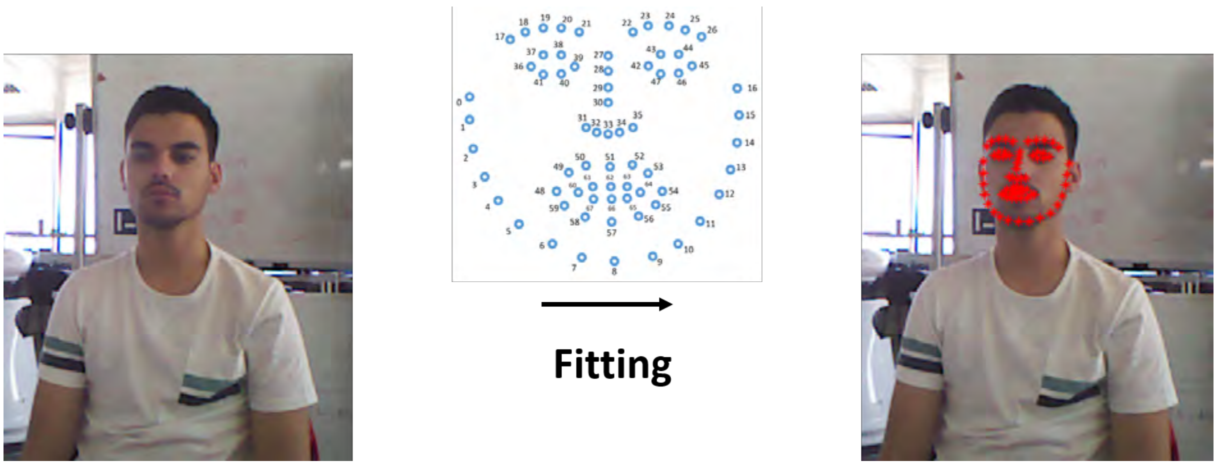


Figure 2.2: Example of facial landmark alignment. In the facial landmark alignment problem we are interested in fitting a landmark model(present in the middle) to the face present on the image. We show, marked on red in the right image, the result of the alignment of the landmark model, using OpenFace [34].

Recently, [34] [35](OpenFace) describes a face analysis toolbox that goes beyond facial landmark detection, since it provides tools for head pose estimation, expression analysis, and gaze estimation. All these tools are based on the facial landmark detection framework provided in [40]. OpenFace is of great value in facial expression analysis and it can be used in a wider field of applications like face analysis in healthcare systems or human-machine interaction interfaces. In our application, it can also be used to estimate the head pose of the user.

Cascade regression methods [41] have been applied to the task of face landmark detection and alignment [37] [36] showing great performance. These methods try to use a sequence of regressions to update the parameters of the model directly from features present in the image, for example, discovering the 2D pose (rotation and translation) of an object present in the image purely based in a sequence of regressors that accept as input features from the image. The regressors used can be, for example, a simple linear map between the features and an update in the pose parameters.

Supervised descent method [37] is an example of a cascade regression for face alignment, the authors motivate the regression as a supervised way of getting descent directions for solving non-linear least squares problems. In this work, the authors used Scale-Invariant Feature Transform (SIFT) features [42] and a linear regressor to update the landmark point positions. Using the same approach, Asthana et al. [38] propose to update the linear regressors in an online manner. The online update of the linear regressors provides a better performance in tracking scenarios since the updated regressors have the ability to adapt to the subject that is being tracked. Supervised descent method was also applied to dense 3D face alignment from 2D videos in [36], providing real-time performance in this task.

More recently, instead of using linear regressors, cascade CNN's were used [39] [43] providing great performance, even in large and difficult poses. However, using CNN's come with the cost of high computational power requirements and worst performance in terms of time consumption.

Methods based on RGB data, e.g facial landmark alignment, have known issues when dealing with large poses. This happens since in large poses it is very difficult to detect the face on the RGB image, thus causing to be very difficult to estimate the head pose using this kind of approach. The solution to solve this problem can be the use of richer data like RGB-D data. However, there is a lack of research on this field and methods for head pose tracking rely on 3D model registration using, for example, Iterative Closest Point (ICP) [44] or its variants [45]. An exception is a method proposed by Fanelli et al. [46] [47] [48], this method uses depth data combined with regression forests to directly estimate the head pose with good accuracy.

Our work tries to incorporate the use of cascade regression methods, that show great performance on the RGB face alignment problem, in the problem of 3D head pose estimation. We formulate the problem of head pose estimation as a 3D rigid model registration and make use of DO [27] [49] [50], that show great results in the task rigid model registration, to the head pose estimation problem. In particular, we applied this work to the tracking of the head pose. DO, when compared to methods like ICP, shows better robustness to outliers, a larger region of convergence and real-time performance in the task of 3D rigid model registration [27].

3

Feedbot: An Autonomous Feeding System

In this work, we are interested in building an autonomous feeding robot arm system (Feedbot) with the purpose of feeding people with upper arm disabilities. We discussed that there are many commercially available robotic feeding platforms. However, none of those is capable of adapting, in real-time, to the changes of the pose of the users or to have feedback of the environment that surrounds the robot.

To give autonomy to Feedbot we rely on computer vision. We are interested in solving two problems. First, we want to be capable of having a visual feedback of the user's pose, more specifically the location of the mouth. Second, we want to have visual feedback on the spoon.

Having a visual feedback of the user's pose is important since people that live with disabilities can be incapable of moving their heads to a preset location, and using this visual feedback we can move the arm's end-effector to the correct mouth's position.

Visual feedback on the spoon is also important because it will allow to estimate the amount of food that is present on the spoon. Using this information we can, for example, detect if no food was scooped and re-scoop, or see if the user has already ate the food from the spoon.

To solve these two problems we propose a hardware setup for Feedbot that consists in 3 parts: a robot arm, a RGB camera pointing to a spoon attached to the robot arm end-effector and a RGB-D camera which is pointing to the user. In figure 3.1, we show our system successfully delivering food to a user, as well as the hardware setup that we used in our experiments.

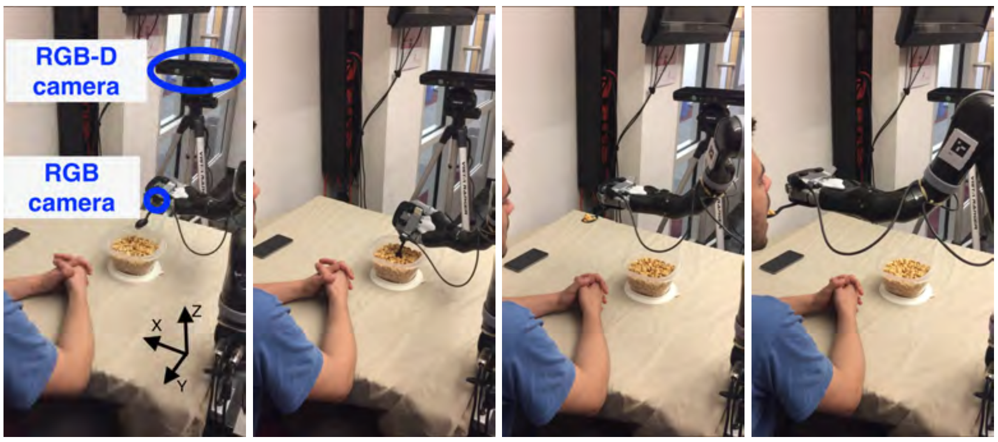


Figure 3.1: Our feeding system successfully delivering food to a user. The first image also labels the axis orientations of the robot coordinate system.

Feedbot is capable of integrating the visual information, given by the RGB and RGB-D cameras, with the capability of movement provided by a robot arm. We divided our software into two parts, the vision system and the robot control system as it is illustrated in figure 3.2.

The vision system is responsible for two main tasks: tracking the user's mouth and provide feedback for what is present on the spoon. In appendix A we present a mathematical model for the cameras used in the vision system. In the next chapter, we will introduce all the methods used to empower our system

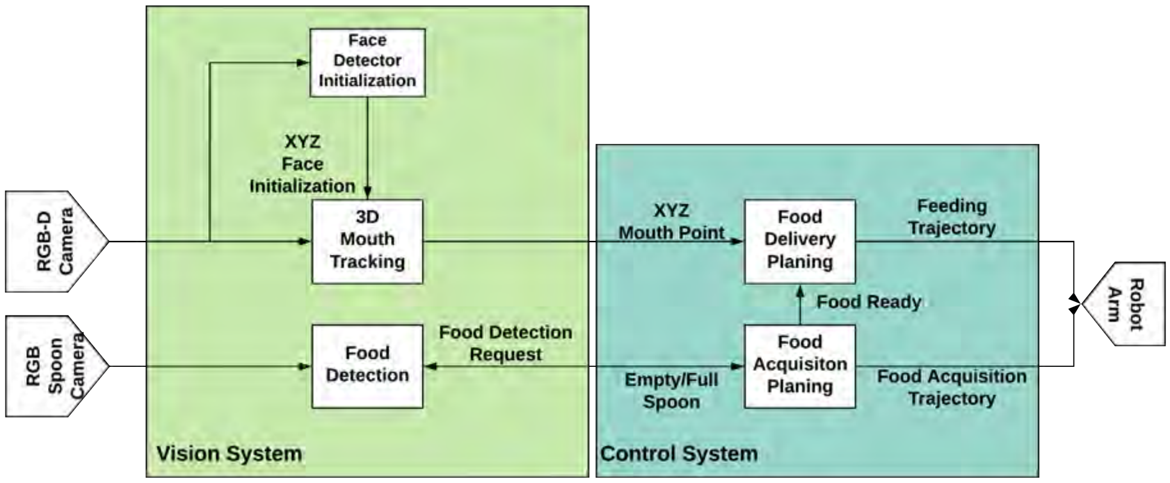


Figure 3.2: System software architecture.

with these capabilities.

The control system is responsible for moving the robot arm to the location provided by the tracking and for acquiring food from the plate.

Feedbot must be capable of switch between different states, these states are based on the action that the robot arm is performing. The transitions between states will be based on the user and environment feedback. We used the state machine that is present in figure 3.3.

Each state is described as follows:

1. **Waiting without food:** In this state, the robot is in a rest position near the plate and waiting without any food on the spoon. When the user shows the will to be fed the robot goes to state 2.

2. **Picking up food from the plate:** When reaching this state, the robot is picking up food from the plate using a predefined trajectory. In this state one of two things can happen, it is detected that enough food was acquired successfully the robot goes to state 3, or it is detected that not enough food was acquired and the robot keeps trying to acquire food from the plate, staying in state 2.

3. **Waiting with food:** After acquiring food from the plate, the robot arm stays quiet waiting for the user to show the intention to be fed. If the user does not show intention the robot keeps waiting with the food on the spoon.

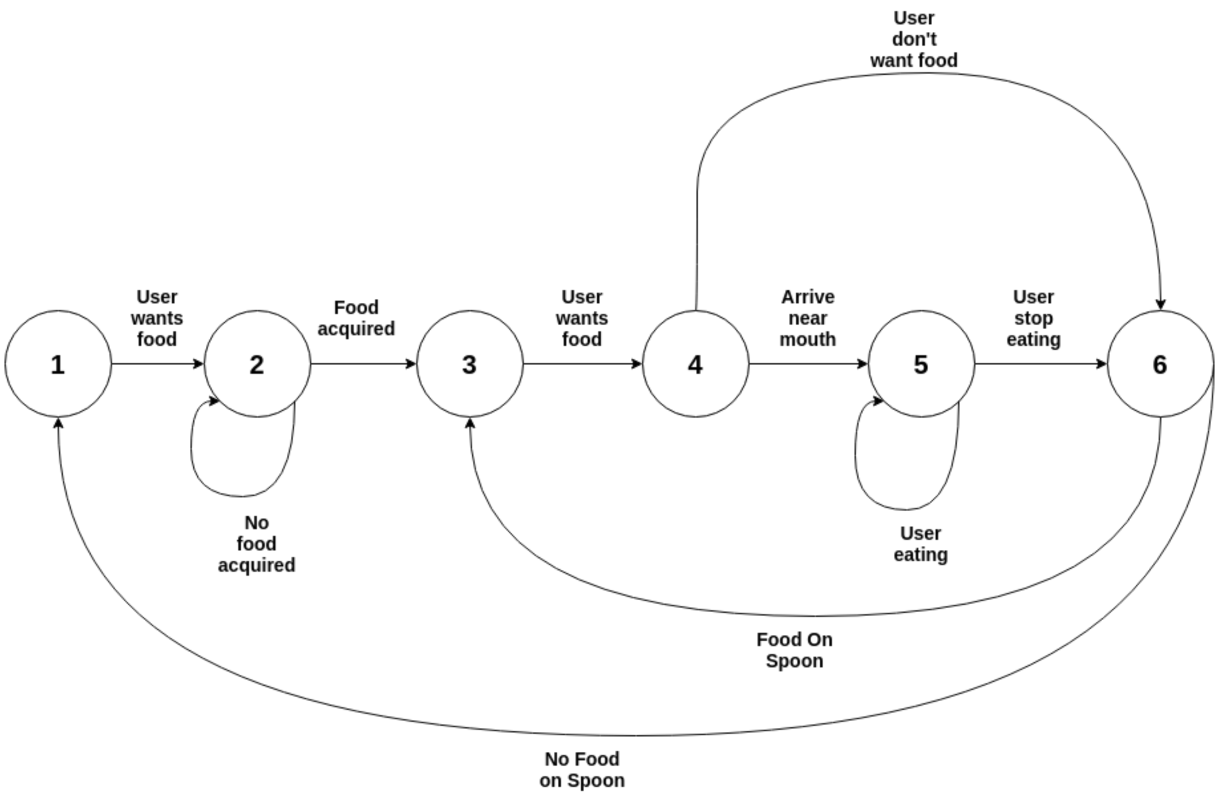
4. **Moving to near the mouth:** If the user shows the intention to be fed, the robot starts doing the movement from the plate to the user's mouth. This trajectory is planned and adapted in real-time based on the feedback given by the vision system. If the arm reaches within a certain distance from the user's mouth the robot goes to state 5. Otherwise, if we detect, while going to the user's mouth, that the user no longer wants to be fed, the robot goes to state 6.

5. **Waiting while food is on the spoon:** In this state, the robot stays with the spoon in front of the user until the user eats the food from the spoon. When the spoon is empty the robot goes to state 6, otherwise, the robot stays on state 5.

6. **Move back to plate** When the user stops eating, the robot starts moving back to the plate position. If it is detected that there is still food on the plate the robot goes to state 3. If no food is detected on the spoon the robot goes to the initial state.

As described before, the transitions between the states are emerged by actions of the user or by the environment. These actions are perceived by the vision system.

Our vision system is capable of perceiving if there is food on the spoon and of tracking the user's mouth. This corresponds to the actions of ”Food/No Food Acquired”, ”Food/No Food on Spoon”, and ”Arrive near mouth”. The perception of the intention of the user, like the action ”User wants/don't want food” and ”User eating/stop eating”, will be studied in future work. In this thesis, we assume that the user always wants food and that the user takes some seconds eating the food from the spoon.



State Names:

1 - Waiting without food
2 - Picking up food from plate
3 - Waiting with food
4 - Moving to near mouth
5 - Waiting while food is on spoon
6 - Move back to plate

Figure 3.3: Robot state machine.

4

Feedbot's Perception

Contents

4.1 Face Registration using 3D Information

In order to work, Feedbot needs to know where the mouth is located. Using the information of where the mouth is located, we can plan a trajectory from the plate to the user's mouth and deliver food to the user. We are, then, interested in the problem of tracking the person's face.

Tracking a person's face is a challenging task because faces are non-rigid objects and, in general, there is no good model to describe non-rigid objects. However, face movements are constrained by the head movements and the person's head is mostly a rigid body, so we will assume that the mouth is just a point in the rigid body shaped by the head. Using this assumption, our problem boils down to the head-pose estimation problem.

We will formulate the aforementioned tracking problem, in each time-step, as a 3D rigid model registration problem. We will assume that we have access to a point cloud that represents the front-view model of the person's head(P_M) and that we have access to a point cloud of a scene(P_S) with a new pose of the model.

Given 2 point clouds $P_S \in \mathbb{R}^{3\times N_S}$ and $P_M \times \mathbb{R}^{3\times N_M}$, the 3D registration problem is known as fitting a rotation matrix R and a translation t that aligns the point cloud P_M with its correspondences in P_S. The fitting can be formalized by the following least-squares problem:

$$\begin{aligned} &\underset{R,t,C}{\text{minimize}} \quad \sum_{i=1}^{N_S}\sum_{j=1}^{N_M} C_{ij} \left\| p_S^i - R\cdot p_M^j - t \right\|^2 \\ &\text{subject to} \quad det(R) = 1; \quad R^T R = I; \\ &\sum_{i=1}^{N_S}\sum_{j=1}^{N_M} C_{ij} \geq N; \quad \forall j \sum_{i=1}^{N_S} Cij \leq 1; \quad \forall i \sum_{j=1}^{N_M} Cij \leq 1; \quad C_{ij} \in \{0,1\}, \end{aligned} \tag{4.1}$$

where C is a matrix where each entry C_{ij} represents if the points p_S^i and p_M^j are correspondences. p_S^i is the point i of the point cloud P_S, and p_M^j is the point j of the point cloud P_M. Finally, N is the minimum number of correspondences that we want between the two point clouds.

Since we do not know the correspondences between the 2 point clouds, we have to include them in the optimization problem using the optimization variable C. The constraints, $\sum_{i=1}^{N_S}\sum_{j=1}^{N_M} C_{ij} \geq N$, $det(R) = 1$ and $R^T R = I$, prevent us from the trivial solution $C_{ij} = 0$. While the introduction of this optimization variable is necessary, it makes our problem a combinatorial non-convex optimization problem which is difficult to solve. Moreover, $\forall j \sum_{i=1}^{N_S} Cij \leq 1$ and $\forall i \sum_{j=1}^{N_M} Cij \leq 1$ ensure that we only have one correspondence for each point.

In figure 4.1 is illustrated an example of a registration problem between point clouds P_S and P_M as formulated in equation (4.1).

The registration problem with unknown correspondences is also formulated on [45] in a different way,

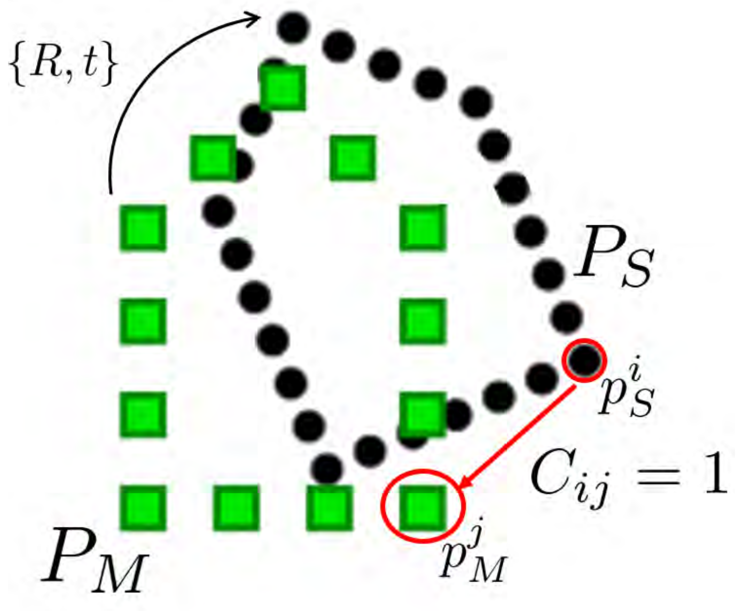


Figure 4.1: Registration problem example. In green points from point cloud P_M. In black, points from the point cloud P_S. Marked on red, correspondence between point p_S^i and p_M^j. Figure adapted from [27].

which can be seen as a relaxed version of problem 4.1. Instead of fixing the number of correspondences to N, the authors introduce a term in the cost function that reduces the cost based on the number of correspondences. They also relax the constraints, removing the constraint $C_{ij} \in \{0, 1\}$ and introducing the constraint $C_{ij} \geq 0$.

The optimization problem present in equation (4.1) is a difficult problem to solve, mainly due to the optimization variable C that represents the correspondences. If we assume that the correspondences are known we can re-formulate it leading to the known Procrustes problem:

$$\begin{aligned} \underset{R,t}{\text{minimize}} \quad & \sum_{i=1}^{N} \left\| p_S^i - R \cdot p_M^i - t \right\|^2 \\ \text{subject to} \quad & det(R) = 1; \quad R^T R = I. \end{aligned} \tag{4.2}$$

Despite a non-convex optimization problem, it has a closed form solution [51] given by:

$$
\begin{aligned}
&\bar{p_S} = \frac{1}{N}\sum_{i=1}^{N} p_S^i; \quad p_{Sc}^i = p_S^i - \bar{p_S};\\
&\bar{p_M} = \frac{1}{N}\sum_{i=1}^{N} p_M^i; \quad p_{Mc}^i = p_M^i - \bar{p_M};\\
&H = \sum_{i=1}^{N} p_{Mc}^i \cdot {p_{Sc}^i}^T;\\
&R^* = U \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(UV^T) \end{pmatrix} V^T;\\
&t^* = \bar{p_S} - R^*\bar{p_M};
\end{aligned}
\tag{4.3}
$$

where matrices U and V are provided by the Singular Value Decomposition (SVD) of the matrix H, which is given by $H = U\Lambda V^T$.

When the correspondences between the two point clouds are not known a common procedure to solve the optimization problem (4.1) is using ICP [44]. ICP is an iterative method that given an initial rotation R^0 and translation t^0 iterates over the two point clouds. First, it solves problem (4.1) for the correspondences, picking as correspondence the nearest point from the other point cloud. With these correspondences, we can apply the solution present in (4.3) to discover a new rotation and translation. After applying this new rotation and translation ICP keeps iterating between these two steps until the distance between correspondences is below a certain threshold. ICP does not guarantee the optimal solution to problem (4.1) and it is prone to local minimum.

4.2 Discriminative Optimization (DO)

DO [49] is a general framework to solve optimization problems when there is training data available. It can be seen as an iterative method that learns descent directions directly from training data. Instead of using ICP to solve the aforementioned 3D registration problem we will use DO. In this section, we will present each step in detail and give the basic formulation to use this general framework.

Usually, we have a set of N data points given by $S = \{(x_1, y_1); ...(x_N, y_N)\}$ and we are interested in fitting a model $f(x, \theta)$ that minimizes the error $e_i = y_i - f(x_i, \theta)$ through all the data points. To do that, we define some error criterion $\psi(e_i)$ and the problem can be formulated as the optimization problem:

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{N} \psi(e_i), \tag{4.4}$$

where ψ is a penalty function of the error and θ the model parameters. A common choice to use as penalty function is $\psi(e_i) = \|e_i\|^2$. Generally, we do not know what type of noise is affecting our data points so we do not know which is the correct penalty function to use.

Assuming that we know the correct penalty function, to solve the optimization problem (4.4), we generally use gradient descent iterations:

$$\theta_{k+1} = \theta_k - \eta \, \nabla_\theta (\sum_{i=1}^{N} \psi(e_i)) \Big|_{\theta_k} , \tag{4.5}$$

where k is the iteration index and η is a hyperparameter that defines the step size for each iteration. To use gradient descent we have to use a differentiable penalty function, but some types of noise can require a non-differentiable penalty function.

There exist mainly two issues when using this way of formulating optimization problems: (1) we do not know what penalty function is the best to use for the data that we have available; (2) to solve it using gradient descent we must use a differentiable penalty function. Using a different penalty function from the optimal one can lead us to a solution that is not the one that we want. Basically, we design a wrong optimization problem.

To overcome this problem, DO tries to learn the best penalty function to use in a data-driven approach. Instead of using gradient descent, DO proposes an update rule for the parameter θ given as:

$$\theta_{k+1} = \theta_k + D_{k+1} h(\theta_k), \tag{4.6}$$

where the matrices D_{k+1} form a Sequence of Update Maps (SUM) and $h(\theta)$ is a function that applies θ to the data and extracts features from that data. For example, if we are dealing with facial landmark registration in images, $h(\theta)$ can be a function that extract SIFT [42] features from a patch near the image landmark pixels.

If we compare DO update rule (4.6) with gradient descent update rule (4.5) we see that they are very similar. In practice, we want that the product $D_{k+1}h(\theta_k)$ has the same rule as the product $-\eta \, \nabla_\theta(\sum_{i=1}^{N} \psi(e_i))\Big|_{\theta_k}$ but, without specifying the penalty function $\psi(e_i)$. The matrix D_{k+1} is a linear mapping from the feature space to the descent directions space.

Assuming that we have a dataset with N problem instances given by the tuples $\{\theta_0^{(i)}, \theta_*^{(i)}, h^{(i)})\}_{i=1}^{N}$, for each problem we know the initial transformation parameter $\theta_0^{(i)}$, the ground truth parameter $\theta_*^{(i)}$ and also the feature extraction function $h^{(i)}$ that is indexed with i to refer that it extracts features from data in the i^{th} problem instance. To be more clear, if we are interested in face landmark alignment these problem instances will be a set of N images and tuples where $\theta_0^{(i)}$ will be the initial position of the landmarks in each of the images and $\theta_*^{(i)}$ the optimal position of the landmarks to align the landmark model with the face present in image i, $h^{(i)}$ will be a function that extracts features in patches near the current landmark positions for the image i. This dataset can be used to compute the optimal matrix D that ensures minimal error in the update rule (4.6) by solving a ridge regression:

$$D_{k+1} = \underset{D}{arg\,min} \frac{1}{N} \sum_{i=1}^{N} \left\| \theta_*^{(i)} - \theta_k^{(i)} + Dh^{(i)}(\theta_k^{(i)}) \right\|_2^2 + \lambda \|D\|_F^2 . \tag{4.7}$$

Since DO needs a sequence of matrices the authors proposed algorithm 4.1 to learn a sequence of K matrices that form the SUM.

Algorithm 4.1: TRAIN *DO*

```
input  : ${(\theta_0^{(i)}, \theta_*^{(i)}, h^{(i)})}_{i=1}^N$, $K$, $\lambda$
output: ${D_k}_{k=1}^K$
for $k := 0$ to $K-1$ do
    Compute $D_{k+1}$ with (4.7)
    for $i = 1$ to $N$ do
        Update $\theta_{k+1}^{(i)} := \theta_k^i - D_{k+1}h^{(i)}(\theta_k^i)$
    end
end
```

Once learned the SUM, we can apply the update rule (4.6) to new problem instances as present in algorithm 4.2.

Algorithm 4.2: INFERENCE *DO*

```
input  : $\theta_0, h, \{D_k\}_{k=1}^K, maxIter, \epsilon$
output: $\theta$
Set $\theta := \theta_0$
for $k := 0$ to $K$ do
    Update $\theta := \theta - D_k h(\theta)$
end
Set $iter := K + 1$
while $\|D_K h(\theta)\|_2 \geq \epsilon$ and $iter \leq maxIter$ do
    Update $\theta := \theta - D_k h(\theta)$
    Update $iter := iter + 1$
end
```

Because we only have a finite number of maps and a finite number of iterations may not be enough to achieve a stationary point. After iteration K, we keep iterating with the final map of the SUM.

Until here, we did not mention how to define a h function. In [49] the authors proposed a general way to define a h function for solving general optimization problems using DO. They also applied it to computer vision problems like camera pose estimation or image denoising. Although this general way of defining a h function is proved to give good results, in our work we are interested in the specific problem of 3D Rigid Registration. As we will show in the next section, a specific h function for this problem was already defined in [27].

4.3 3D Face Tracking Using DO

In this work, we are interested in tracking the user's face. By tracking the user's face, we can compute the 3D location of the user's mouth and plan a trajectory from the plate to that point and feed the user.

Given a point cloud P_S^t gathered by the RGB-D sensor at instant t and a 3D point cloud model P_M of the user's face, the tracking problem can be thought as discovering the 3D rigid body transformation (R^t, t^t) that align the model point cloud P_M to the point cloud P_S^t. In practice, we are solving at each time-step the problem (4.1). In our case, we use, at time-step t, the previous estimations of iterations $t-1$.

For simplicity, in this section, we will represent the point clouds in homogeneous coordinates:

$$\tilde{R}^t = \begin{bmatrix} R^t & t^t \\ 0_3 & 1 \end{bmatrix}; \tilde{P}_S^t = \begin{bmatrix} P_S^t \\ \mathbf{1}^T \end{bmatrix}; \quad \tilde{P}_M = \begin{bmatrix} P_M \\ \mathbf{1}^T \end{bmatrix}. \tag{4.8}$$

This will allow us to write the 3D rigid transform (R^t, t^t) between P_M and P_S^t as $\tilde{P}_S = \tilde{R}\tilde{P}_M$, it is important to notice that this equation will only hold for points that have a correspondence.

4.3.1 DO applied to Rigid Tracking

Parameterization and Feature Function

Since DO's framework does not admit constraints in the optimization problem, to solve problem (4.1) we have to use another parameterization of the rigid body transformation known as the Lie algebra [27].

Lie algebra simply maps the rigid transformation tuple (R, t) to $\mathbb{R}^6$, each element in $\mathbb{R}^6$ will be associated with a rigid transformation via the exponential map. We will denote our parameter vector $\theta \in \mathbb{R}^6$, and the mapping between that parameter and the corresponding rigid transform (R, t) will be given as:

$$\begin{aligned}
&\theta = (u \quad w)^T, \quad u, w \in \mathbb{R}^3;\\
&\phi = \sqrt{w^T w}, \quad A = \frac{\sin(\phi)}{\phi};\\
&B = \frac{1-\cos(\phi)}{\phi^2}, \quad C = \frac{1-A}{\phi^2};\\
&R = I + Aw_\times + Bw_\times^2;\\
&V = I + Bw_\times + Cw_\times^2;\\
&t = Vu;\\
&exp(\theta) = \begin{bmatrix} R & Vu \\ 0_3 & 1 \end{bmatrix}.
\end{aligned} \tag{4.9}$$

Using this parameterization for rigid transformations we can use DO without having to explicitly use constraints to ensure that the solution is a rotation matrix.

In [27] the authors propose a $h : \mathbb{R}^6 \times \mathbb{R}^{3\times N_s} \to \mathbb{R}^{2N_M}$ function to use DO with this parameterization, the proposed function is present in equation (4.10).

$$\begin{aligned}
[h(\theta, P_S)]_a &= \frac{1}{z} \sum_{p_S \in S_a^+} exp(-\frac{1}{\sigma^2} \|\tau(p_S;\theta) - p_M^a\|^2)\\
[h(\theta, P_S)]_{a+N_M} &= \frac{1}{z} \sum_{p_S \in S_a^-} exp(-\frac{1}{\sigma^2} \|\tau(p_S;\theta) - p_M^a\|^2)
\end{aligned} \tag{4.10}$$

The image of this function is a vector in n $\mathbb{R}^{2N_m}$. To better illustrate how the entries of this vector are computed, we illustrate that in figure 4.2.

The computation of each entry for each of the model points is as follows:

1. We compute the normal(n_a) for that model point(p_M^a) based in a neighborhood of that point.
2. Using the normal, we define an hyperplane that separates the points of the scene point cloud (p_S) in two sets, the points that are in the ”front” of the model point (S_a^+) and the points that are in the ”back” of the model (S_a^-).
3. The distances of the points that are in the ”front” are all weighted by a radial basis function and summed and this value is placed in the entry corresponding to the model point, lets assume that p_M^a corresponds to the point indexed by 1, this value is then placed in the entry 1 of the h vector.
4. The same thing is done for the points in the ”back”, but the value is placed in the entry $N_m + 1$ of the h vector.
5. Finally, the entries are all normalized so that they sum to 1, this normalization corresponds to the value of z.

This function can be seen as a shape correlation [52] between the scene point cloud and the model point cloud.

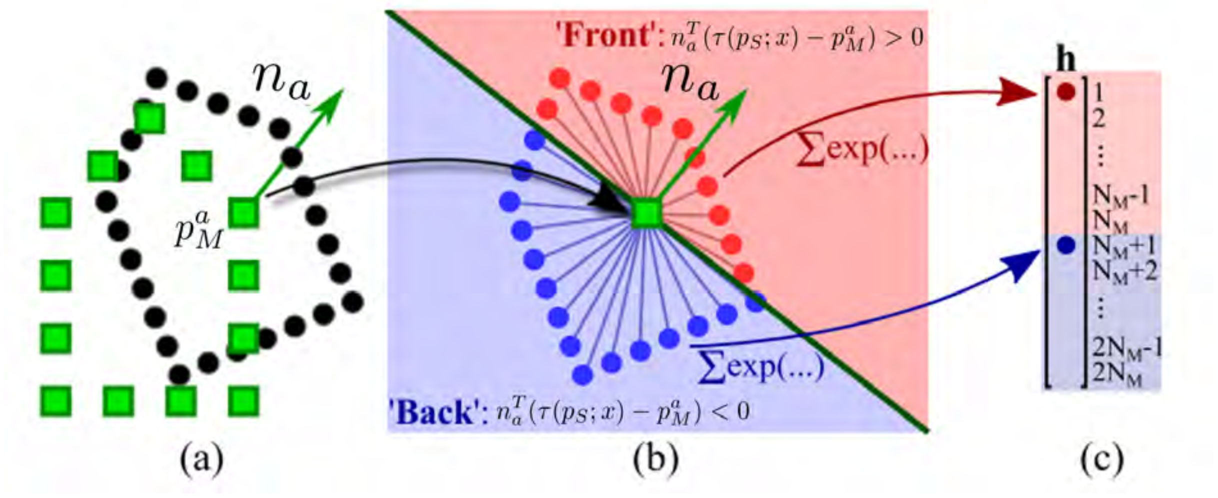


Figure 4.2: Feature extraction by function h. (a) In green, points from the model point cloud P_M, in black, points from the scene point cloud P_S. (b) In red, points in the "front" of the model's point represented in green, in blue, points in the "back". (c) h function representation with the assignment of the weights to each of the vector indices. Image adapted from [27]

.

It is important to notice that the rigid transformation that we compute using DO with this feature function is the transformation that aligns the scene point cloud with the model given by: $p_M = \tau(s_b, \theta) = R^* p_S + t^*$, which is not the transformation that we want: $p_S = Rp_M + t$. However, the relation between these two transformations is known as: $(R, t) = (R^{*T}, -R^{*T}t)$.

Training Normalization

Once defined the h function and assuming that we have the 3D model of the user's face(P_M), we can train the SUM. We normalize the model point cloud to fit in the [-1, 1] box using the following normalization:

$$\begin{aligned} P_M^1 &= P_M - \min(P_M), \\ P_M^2 &= P_M^1 - \max(P_M^1) \cdot \frac{1}{2}, \\ P_{M_n} &= P_M^2 \cdot \frac{1}{maxabs(P_M^2)}, \end{aligned} \tag{4.11}$$

P_{M_n} will represent the normalized model point cloud. min and max correspond to the operation of computing the bigger and smaller element along each axis of the 3D points present in the point cloud P_M. $maxabs$ correspond to the operation of taking the absolute value of each element of the point cloud and then choose the biggest element.

If we use homogeneous coordinates this normalization operation can be written in a matrix $S \in \mathbb{R}^{4\times4}$ given by:

$$S = \begin{bmatrix} I_3 \cdot \frac{1}{maxabs(P_M^2)} & a \\ 0_3 & 1 \end{bmatrix}, \tag{4.12}$$

where $a = [P_M^2(:,1) - P_M(:,1)] \cdot \frac{1}{maxabs(P_M^2)}$ and $P_M^2(:,i)$ will denote the i-th column of the matrix P_M^2.

After normalization, we downsampled the point cloud to have a model with around 300-400 points, this downsampling is important to accelerate the computation time of DO.

Finally, we generated a synthetic dataset containing problem instances by applying random rotations and translations to the normalized model. With this dataset we can use the algorithm present in 4.1 to train a SUM.

Tracking Scheme

With the trained SUM and the 3D user's face model, we have everything we need to apply DO and track the user's face. In figure 4.3 we present a scheme that illustrates the steps of our tracking algorithm based on DO.

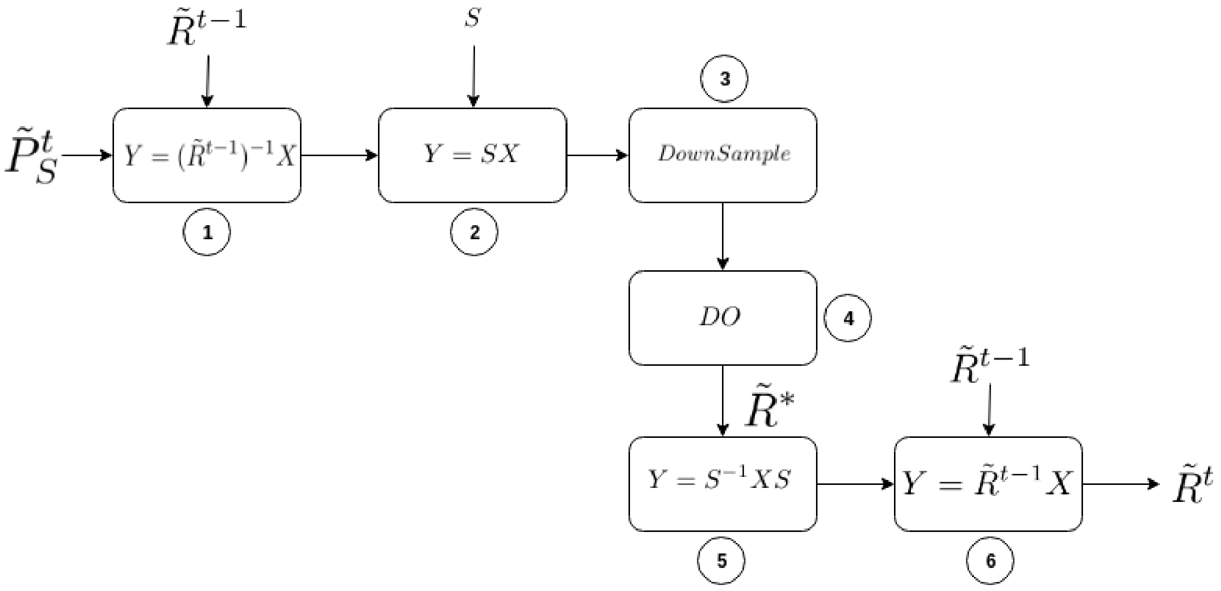


Figure 4.3: DO tracking scheme.

Given a point cloud of the scene, received by the RGB-D sensor in each time-step $\tilde{P}_S^t$, we aim to compute the transformation that ensures $\tilde{P}_S^t \approx \tilde{R}^t \tilde{P}_M$. Our tracking method works as follows:

1. The inverse of the previously computed rigid transformation $(\tilde{R}^{t-1})^{-1}$ is applied to the point cloud scene. This is done in order to put the point cloud $\tilde{P}_S^t$ close enough to the model point cloud $\tilde{P}_M$. Usually, the rigid transform between two consecutive instants is small and this step is enough to put the 2 point clouds near each other.

2. The same normalization that was used in training is applied to the new point cloud.

3. In order to accelerate the computations of DO, in this step, the point cloud is downsampled.

4. DO is used to estimate the transformation between the previous point cloud and the normalized model point cloud. The output of DO will not be the transformation between the model and the scene point cloud, this happens because we applied the previous transformation to the scene point cloud and also a normalization.

5. In order to compute the right transformation, we first de-normalize the transformation discovered by DO.

6. Finally, to compute the final transformation that aligns the model point cloud $\tilde{P}_M$ with the scene point cloud $\tilde{P}_S^t$ the inverse of the rigid transformation applied in step 1 is applied to the previously computed rigid transformation.

4.3.2 Initialization

3D registration algorithms, like ICP and DO, require an initialization step that places the model of the face close to the face in the scene. We use a face detector to estimate an initial guess of where the face is in the RGB-D image. The difference between ICP and DO is that this initialization can be worst when using DO since it has a much larger convergence region.

A popular face detector is the Viola Jones face detector [21]. Though this face detector is fast, it is not invariant to different face poses. MTCNN [32] is more robust to different face poses and also provides landmarks of the mouth, nose, and eyes as it is present in figure 4.4. Despite MTCNN is based on deep learning, it can still run on a CPU at a lower speed. Since initialization does not happen frequently, the speed of the initialization step is not an important concern, and we remove the need for a GPU in our system by running initialization on a CPU.

We initialize our face tracker using the average 3D location of the facial landmarks given by MTCNN, these landmarks are present in figure 4.4. The 3D locations of these landmark points can be computed because we are using an RGB-D camera. With these 3D points, we approximate the initial translation t^0 of the face model in the scene as:

$$t^0 = \frac{1}{L}\sum_{i=1}^{L} p_S^i - \frac{1}{N_M}\sum_{j=1}^{N} p_M^j, \tag{4.13}$$

where L is the number of landmarks and N_M is the number of points in the model. For the initial rotation of the model, we assume $R^0 = I$, where I is the 3×3 identity. MTCNN landmarks can also be used to compute the mouth's 3D location in the model's point cloud which will be used as the 3D point to where the robot arm must move.

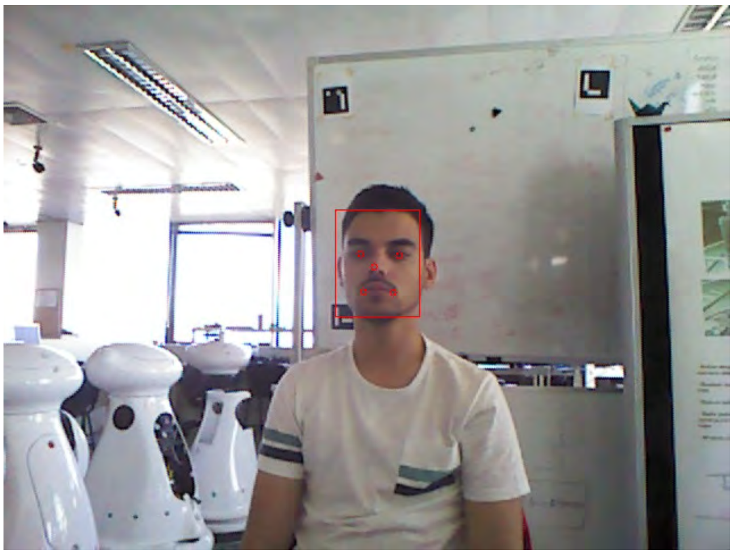


Figure 4.4: In red output provided by MTCNN.

4.3.3 Acquiring a 3D face model

Our system must be capable of gathering, in an automatic way, the user's 3D face model. This is important because we want our system to deal with different users. To gather this 3D face model we used the approach described in figure 4.5.

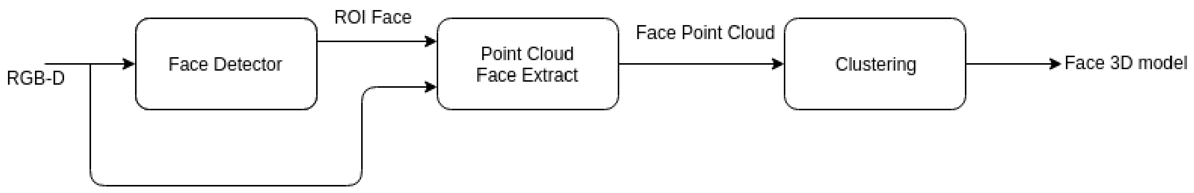


Figure 4.5: Extraction of the 3D face model.

Given an image from the RGB-D sensor, we use a face detector such as Viola-Jones [21] or MTCNN [32]. The face detector provides a ROI containing the face of the user, as it is present in figure 4.6.

We select the 3D points corresponding to the ROI, leading to the point cloud present in figure 4.7.

Marked with circles, in figure 4.7, we can see points that do not correspond to the user's face. These points belong to the background. To remove these points we used a clustering algorithm [53] and select the points corresponding to the cluster with the highest number of points. Using this method, we compute a point cloud corresponding to the user's face, which is presented in figure 4.8.

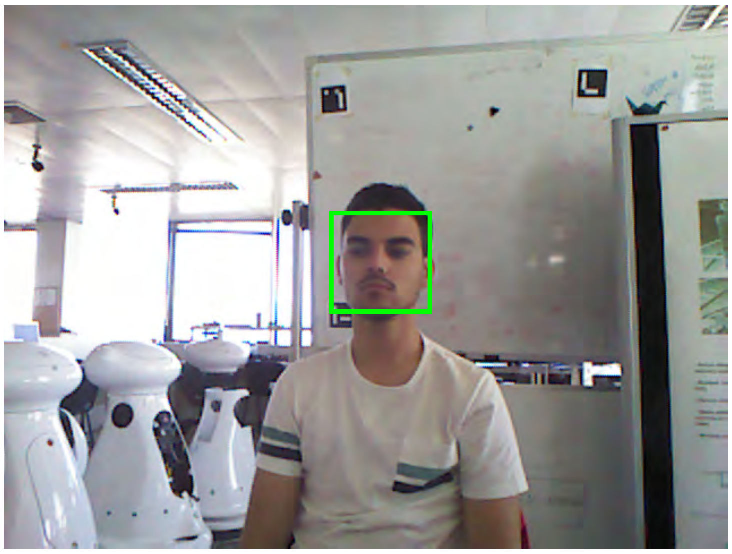


Figure 4.6: In green ROI given by the face detector.

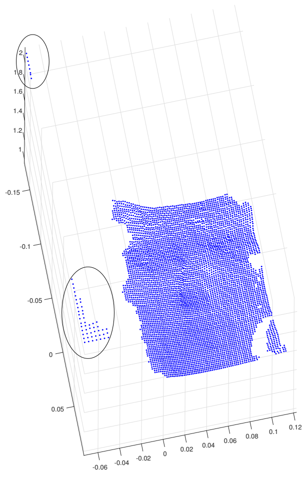


Figure 4.7: Point Cloud corresponding to the ROI present in figure 4.6.

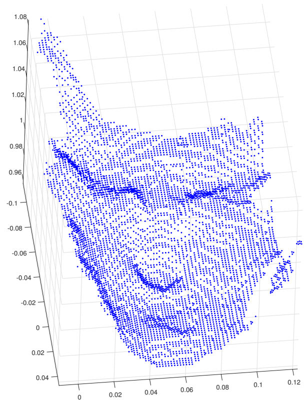


Figure 4.8: Final user's face point cloud gathered with our method.

4.4 Camera-Arm Calibration

In order to the robot use the information from the RGB-D camera in a meaningful way, we have to calibrate the robot base frame with the RGB-D camera frame. We want to do this calibration in an automatic way, so that we can use Feedbot quickly after changing the position of the camera or the robot arm. We want to compute the rigid transformation (R, t) between the camera reference frame and the robot reference frame as it is illustrated in figure 4.9.

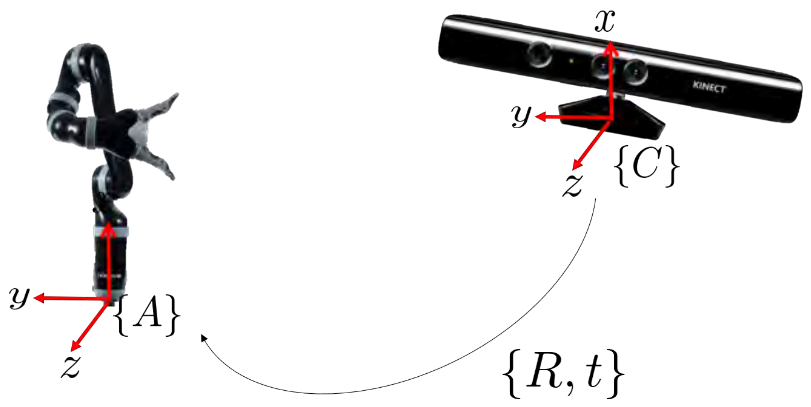


Figure 4.9: Calibration between Robot Arm reference frame ($\{A\}$) and Camera reference frame ($\{C\}$).

If we detect corresponding points between the two reference frames, we end up in the same framework as the 3D registration problem with known correspondences (4.2). To establish the correspondences, we used a known object attached to the robot end effector.

An orange ball was attached to the robot arm end effector and assuming that the robot arm kinematics are accurate, this means that we will know, in the robot reference frame, where the ball is located. The last step to compute the correspondences is to locate the center of the ball in the camera reference frame.

To compute the ball location in the camera reference frame we took leverage of the properties of the ball. We choose an orange ball as the object to calibrate because it is easy to detect in the RGB image using color segmentation, and we know a parametric model to the 3D points of the object, which will be good to compute a precise location of the ball center.

To detect the 3D points that correspond to the ball, we start by doing RGB color segmentation based on the HSV color space. After the segmentation, we use algorithm [54] to detect the contour of the ball, this contour is used to estimate a minimum area rectangle that contains the ball. This rectangle will be used as ROI to get the 3D corresponding points of the ball. An example of the output ROI is present in figure 4.10.

To compute the center of the ball we fit a sphere to the 3D points gathered previously. The sphere model is given by: $(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2$. Since we know the radius of the ball, fitting the

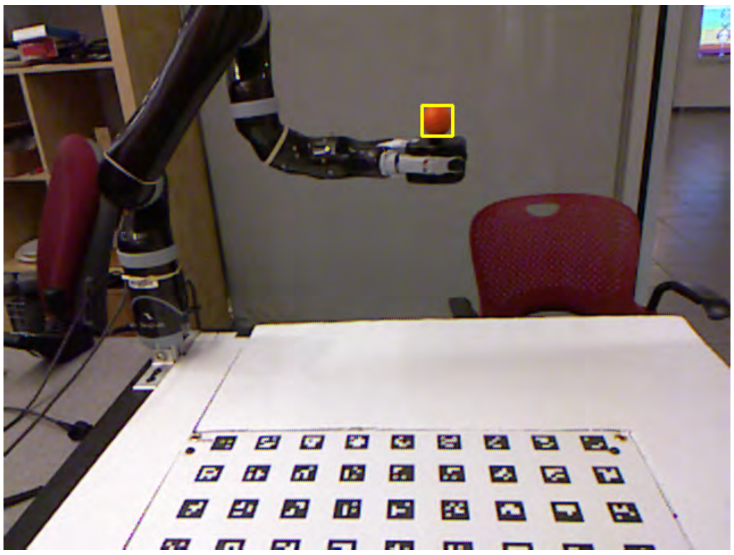


Figure 4.10: In yellow, ROI provided after segmentation and refinement on RGB image.

center can be written as the non-linear least squares problem:

$$\underset{x_c, y_c, z_c}{\text{minimize}} \quad \sum_{i=1}^{N} \Big((x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2 - r^2 \Big)^2, \tag{4.14}$$

non-linear least squares problems are in general difficult to solve. However, they can be solved using the Levenberg-Marquardt method. Instead of using this method we will try to reformulate the problem as a linear least squares.

We can re-write the sphere model equation as:

$$\begin{bmatrix} 1 & -2x & -2y & -2z \end{bmatrix} \begin{bmatrix} {x_c}^2 + {y_c}^2 + {z_c}^2 \\ x_c \\ y_c \\ z_c \end{bmatrix} = r^2 - x^2 - y^2 - z^2, \tag{4.15}$$

with this new equation we can reformulate the problem of sphere fitting 4.14 as:

$$\underset{a,b,c,d}{\text{minimize}} \quad \sum_{i=1}^{N} (\begin{bmatrix} 1 & -2x_i & -2y_i & -2z_i \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} - r^2 + {x_i}^2 + {y_i}^2 + {z_i}^2)^2, \tag{4.16}$$

this problem is a linear least squares problem in the variables a, b, c and d and can be solved easily. We can solve it and then assume that $b = x_c$, $c = y_c$ and $d = z_c$.

Solving (4.16) is not the same as solving the non-linear version (4.14), the two problems are equivalent only if we put the $a = b^2 + c^2 + d^2$ constraint in 4.16. We can think of solving (4.16) as solving a relaxed version of (4.14).

To prevent bad sphere fitting due to outliers we use Random Sample Consensus (RANSAC) in the model fitting. These outliers are present in the data points due to the same reason that was presented in the 3D user's face model gathering in section 4.3.3.

Finally, in order to estimate the transformation between the robot base frame and the camera base frame, the robot arm, with the ball attached, moves to different positions. In each of these positions, we take a RGB-D picture and we use the previously discussed method to get the 3D position of the center of the ball. The center of the ball positions in the camera reference frame and the same positions in the robot base frame are saved. After getting a sufficient number of correspondences (around 20-50) we solve the registration problem 4.2 and we get the transformation between the two frames. This procedure takes around 2-5 minutes and can be done in an automatic way.

4.5 Food Detection

To see what is on the spoon, we place a tiny RGB camera on the end-effector of the robot arm as shown in figure 4.11. We use the images provided by this camera for two purposes, to detect if there is enough food to serve to the user and to detect if the user ate the food from the spoon. To detect if there is enough food on the spoon we use a detection algorithm that we can tune to specify the required amount of food, to do this we need to have a correspondence between the food image and the weight of food. To detect if the user has eaten the food, we define a classifier with two classes: ”empty” and ”non-empty” spoon, which receives as input the images from the spoon's camera.

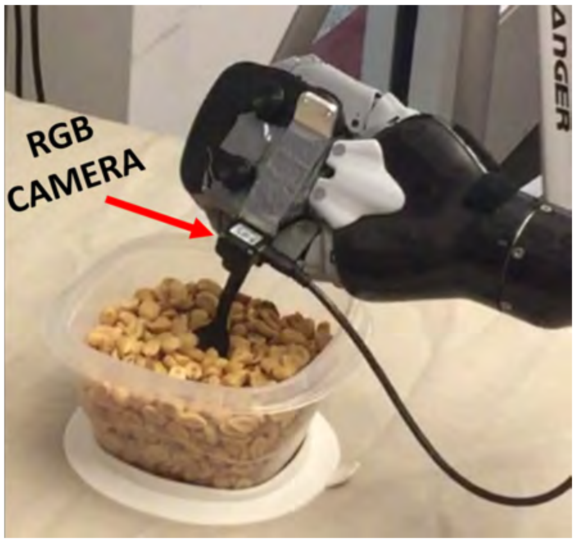


Figure 4.11: RGB camera attached to the robot end effector that is pointing to the spoon.

The first problem to solve is that the image provided by the camera does not contain only the spoon. To solve this problem, we developed a calibration step that computes a mask of the spoon's location. Calibration is performed by acquiring an image with a white sheet of paper under the spoon. Otsu's method [55] is used to select a threshold and segment the image which provides a spoon's mask. This method works well because the colored spoon contrasts sharply with the white sheet of paper. Using this calibration step, our camera will provide images like the one present in figure 4.12.

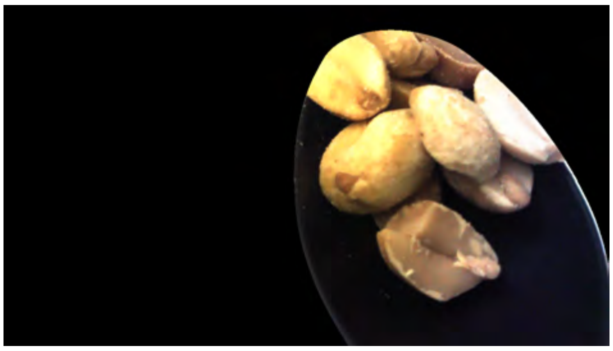


Figure 4.12: Example masked image of the spoon after the calibration step.

4.5.1 Food Weight Estimation

We need to have a measure of how much food is present on the spoon in order to be capable of deciding when is enough food to deliver to the user or not.

In the calibration step, we also record a histogram, H_s, of the H channel of the HSV image of the empty spoon after applying the calibration mask. The recorded histogram, H_s, is compared with the histograms of subsequent images of the spoon, H_a. To compare the histograms, we used the normalized correlation between the two histograms,

$$d(H_s, H_a) = \frac{\sum_{i=1}^{N}(H_s(i) - \bar{H}_s)(H_a(i) - \bar{H}_a)}{\sqrt{\sum_{i=1}^{N}(H_s(i) - \bar{H}_s)^2 \sum_{i=1}^{N}(H_a(i) - \bar{H}_a)^2}}, \tag{4.17}$$

where $\bar{H} = \frac{1}{N}\sum_{i=1}^{N} H(i)$ is the histogram mean. If the value of the correlation is below a certain threshold, then the algorithm reports that there is enough food on the spoon.

We train a linear regression to discover the relationship between the histogram correlation value and the weight of the food on the spoon. This regression was trained using a dataset containing examples of spoons with food and the correspondent weight. This regression can be used to inform the choice of the threshold for “enough food” on the spoon.

4.5.2 Empty/Non Empty Spoon Classification

Another task that Feedbot must be capable of is to distinguish between an empty spoon and a non-empty spoon. This is important since we do not want to present empty spoons to the user. It is also important because we can know when the user has finished eating the food present on the spoon.

We want to classify the masked images gathered by the camera in two classes, to classify these images we tried two different classifiers: a linear SVM and a logistic regression.

SVM Classifier

Assuming that we have a dataset composed by tuples of data points ,$\{(x_1, y_1), ..., (x_N, y_N)\}$, where $x_i \in \mathbb{R}^d$ denotes the input features, and $y_i \in \{-1, 1\}$ denotes the class to which the data point belongs, namely, empty spoon and non-empty spoon.

A linear SVM is a classifier that tries to discover the hyperplane, $x^T\beta + \beta_0 = 0$, that better separates the data based in the maximum margin criterion. If the data is separable by a hyperplane there are many hyperplanes that can be the solution. The maximum margin criterion ensures the solution that guarantees a hyperplane that has the largest possible distance between the two classes. The distance between the classes and the hyperplane (Margin) will be denoted as M. In figure 4.13 we have an example of a linear SVM in $\mathbb{R}^2$.

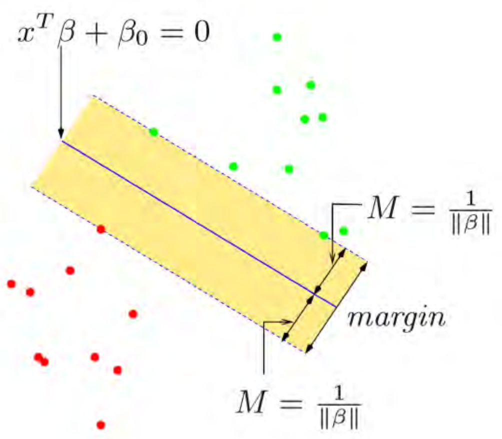


Figure 4.13: Example of a linear SVM in $\mathbb{R}^2$. Points from different classes in green and red. Figure from [56].

We can formalize the problem of computing the parameters of the linear SVM as the optimization problem:

$$\begin{aligned} \underset{\beta_0,\beta}{\text{maximize}} \quad & M \\ \text{subject to} \quad & y_i({x_i}^T\beta + \beta_0) \geq M; \quad i = 1, ..., N; \\ & \|\beta\| = 1. \end{aligned} \tag{4.18}$$

We can remove the constraint $\|\beta\| = 1$ doing $y_i({x_i}^T\beta + \beta_0) \geq M \|\beta\|$. Since this inequality still hold if we multiply both sides by a scalar, we can choose $M = \frac{1}{\|\beta\|}$, which leads to the final optimization problem:

$$\begin{aligned} \underset{\beta_0,\beta}{\text{minimize}} \quad & \|\beta\|^2 \\ \text{subject to} \quad & y_i({x_i}^T\beta + \beta_0) \geq 1, \quad i = 1, ..., N \end{aligned} \tag{4.19}$$

This is a convex optimization problem, in particular, a quadratic program. Quadratic programs are easy to solve using toolboxes like CVX [57]. There is a big limitation when solving this optimization problem, it only has a solution if the data is linearly separable, which means that for the binary case that we can separate the data in two sets using a hyperplane. In general, data is not linearly separable, due to noise or its nature.

To adapt the optimization problem (4.19), we can introduce slack variables ϵ_i, one for each of the data points, that allows having some data points that fall in the wrong region of the hyperplane. Introducing these slack variables leads to the following optimization problem (see [56] for a better explanation):

$$\begin{aligned} \underset{\beta_0,\beta,\epsilon_i}{\text{minimize}} \quad & \|\beta\|^2 + \lambda \sum_{i=1}^{N} \epsilon_i \\ \text{subject to} \quad & y_i({x_i}^T\beta + \beta_0) \geq 1 - \epsilon_i, \quad i = 1, ..., N; \\ & \epsilon_i \geq 0 \quad i = 1, ..., N, \end{aligned} \tag{4.20}$$

where λ is a hyperparameter that represents how much misclassifications we allow, larger λ corresponds to allowing fewer misclassifications since we increase the cost of having points in the wrong side of the hyperplane.

The introduction of these slack variables do not destroy the convexity of our problem, so we can still solve it using convex optimization tools.

Logistic Regression Classifier

Another way to classify if there is food or not in the spoon is using logistic regression. Logistic regression is widely used in classification problems, we will assume that we only have 2 classes denoted by 0 and 1. It assumes a conditional probability distribution for the class that the data belongs given by:

$$\begin{aligned} Pr(Y=0|X=x) &= \frac{exp(x^T\beta+\beta_0)}{1+exp(x^T\beta+\beta_0)}; \\ Pr(Y=1|X=x) &= \frac{1}{1+exp(x^T\beta+\beta_0)} = 1 - Pr(Y=0|X=x) \end{aligned} \tag{4.21}$$

Given a dataset $\{(x_1, y_1), ..., (x_N, y_N)\}$, we can estimate the parameters of the distribution, β and β_0, using maximum likelihood estimation.

Maximum likelihood estimation is based on maximizing the likelihood function, or equivalently the log-likelihood function [56]. In our case, the log-likelihood function is given by:

$$\begin{aligned} l(\beta,\beta_0) &= \sum_{i=1}^{N} y_i log(p(x_i;\beta,\beta_0)) + (1-y_i) log(1-p(x_i;\beta,\beta_0)); \\ &= \sum_{i=1}^{N} y_i(x_i^T\beta+\beta_0) - log(1+exp(x_i^T\beta+\beta_0)), \end{aligned} \tag{4.22}$$

where $p(x_i;\beta,\beta_0) = \frac{exp(x_i^T\beta+\beta_0)}{1+exp(x_i^T\beta+\beta_0)}$. Maximizing this function is equivalent to minimizing the function $-l(\beta,\beta_0)$. We can formalize the parameter estimation as the optimization problem:

$$\underset{\beta_0,\beta}{\text{minimize}} \quad \sum_{i=1}^{N} -y_i(x_i^T\beta+\beta_0) + log(1+exp(x_i^T\beta+\beta_0)), \tag{4.23}$$

this optimization problem is convex.

The input feature, x, for the logistic regression is the value of the histogram correlation (4.17) between an empty spoon and each of the examples in the dataset.

4.6 Summary

In this section, we presented methods to meet the requirements of the vision system proposed in chapter 3. We remember the reader that our vision system has two modules, one capable of tracking the user's mouth location and other capable of giving visual feedback from the spoon attached to the end-effector.

We started by proposing a new head-pose tracking system based on DO that assumes the mouth's point is rigidly attached to the head. It can be used to track the user's mouth position in real-time.

We also proposed a method to detect how much food is on the spoon using a linear regressor and a method to classify if there is food or not on the spoon using a SVM classifier and a Logistic Regression.

These methods can be used to have visual feedback on the spoon.

The control system of Feedbot is not the focus of this thesis. To give to Feedbot the capability of real-time adapting the position of the end-effector to the output provided by the vision system we use the approach described in appendix B. Another task that the control system is responsible is the acquisition of food from the plate, to acquire food we use a recorded trajectory from the dataset [29].

An essential step to integrate the control and vision system is to calibrate the cameras to the robot arm reference frame. To do that, we proposed an automatic calibration method that is independent of the robot arm that is used. This method is based on the detection and registration of a known object.

Using the methods proposed in this chapter and the needed hardware setup we built Feedbot, that was proposed in chapter 3.

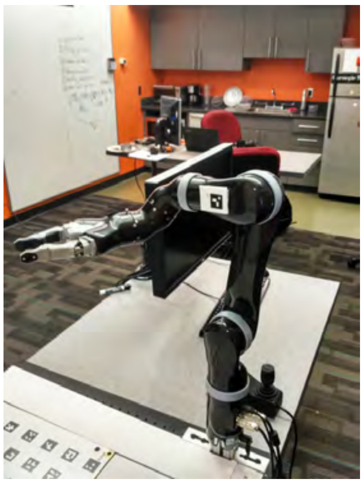
5

Feedbot's Experimental Evaluation

Contents

5.1 Feedbot Evaluation

We tested Feedbot using two different robot arms, namely Kinova MICO and Niryo One [4], showing that Feedbot system can use different robot arms. In figure 5.1, we show these two different robot arms.


(a) Kinova MICO robot arm.

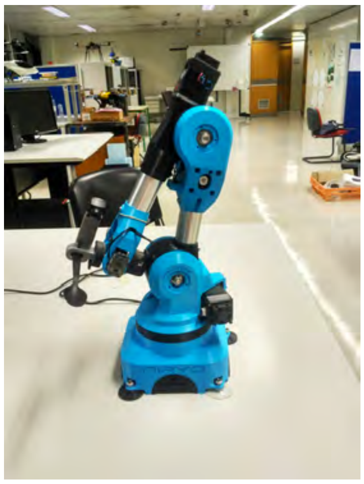

(b) Niryo One robot arm.

Figure 5.1: We tested the proposed system using Kinova MICO robot arm and Niryo One robot arm.

Kinova MICO is a 6DOF commercially available robot arm that is highly used by wheelchairs users. Niryo One is a 6DOF low cost and easy to use 3D printed robot arm, the whole system is open source and fully compatible with Robot Operating System (ROS). It is commercially available, but can also be constructed since the hardware is open source.

Kinova MICO Setup

To test our Feedbot and validate the whole system components in a real meal scenario, we did a small experiment with two different persons [1]. In this experiment, we used as robot arm a Kinova MICO, the setup used in the experiment can be seen in figure 3.1. To be capable of using our tracking method, for each of the two subjects we acquired a 3D face model as described in section 4.3.3.

In this study we used three different types of food, namely, peanuts, M&M's and mac'n cheese. We used different types of food in order to test our system in different aspects. Peanuts were used since they are difficult to feed without falling off the spoon if the robot end-effector is too shaky, in figure 5.2 we present an example of Feedbot feeding peanuts to one of the persons in the experiment.

To test the capability of Feedbot to deal with foods of different colors we used M&M's. M&M's are good to test this aspect since each of them has a different color and together they form a very unusual

[1] In `https://www.youtube.com/watch?v=X7McqWk1AK8` it is available a video showing Feedbot performing different tasks on this experiment

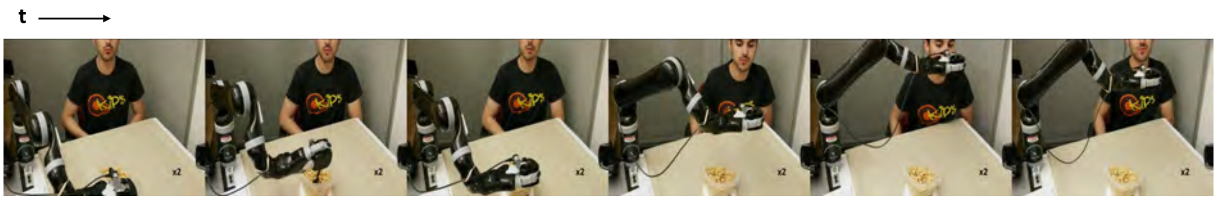


Figure 5.2: Feedbot feeding peanuts.

color pattern, this pattern was good to test if the RGB camera that looks to the spoon, and detects if there is enough food on the spoon, can still work with different color patterns. In figure 5.3, we present an example of Feedbot feeding M&M's to the second subject.

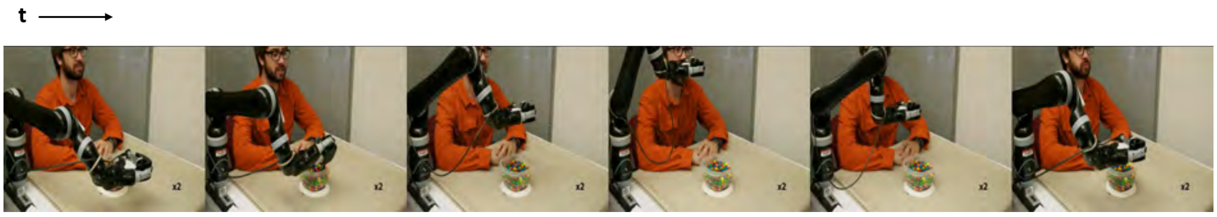


Figure 5.3: Feedbot feeding M&M's.

During this experiment, we confirmed that our procedure for getting the feedback of how much food is present on the spoon has some drawbacks since it is based on the comparison of the color histogram. In cases where we got many M&M's with the same color as the spoon, we got some false empty spoons detections. This problem can be solved using a different color for the spoon, in all our experiments we used a black spoon, using, for example, a white spoon could solve this problem. Another way of solving this problem would be using a depth camera instead of a RGB camera. However, the camera is at a short distance from the spoon and needs to be very small (see figure 4.11), these reasons make it difficult to have a depth camera that meets our requirements.

Using mac'n cheese as testing food, we were capable of test the Feedbot's acquisition feedback, since mac'n cheese is difficult to scoop and sometimes gets stick to the plate. As we can see in figure 5.4 when the scoop movement fails to acquire the food from the plate, our feedback system is capable of reporting that the spoon is empty and Feedbot tries to scoop again.

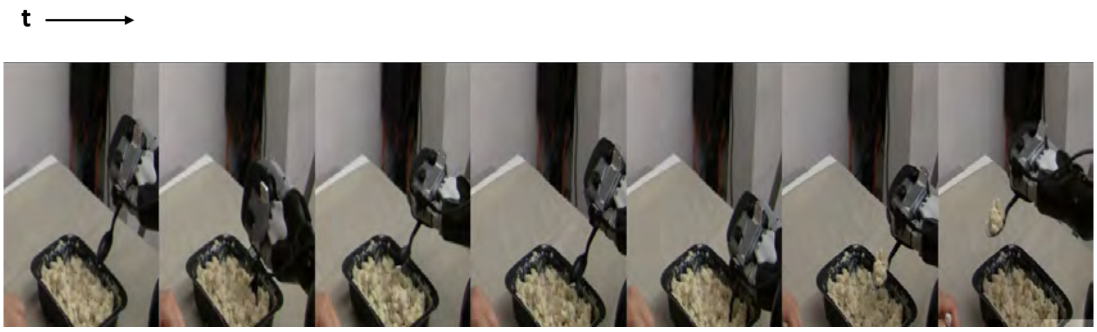


Figure 5.4: Feedbot feeding mac'n cheese.

Niryo One Setup

To validate that Feedbot system works with different robot arms, we did an experiment using a Nyrio One robot arm [2]. In this experiment, we tested Feedbot with one user that suffer from upper-extremity disabilities, in this test we were more focused on testing the capability of Feedbot to track and adapt its trajectories to a user that is moving in real-time.

In figure 5.5, it is shown that Feedbot is capable of adapting its trajectory to the user that is moving.

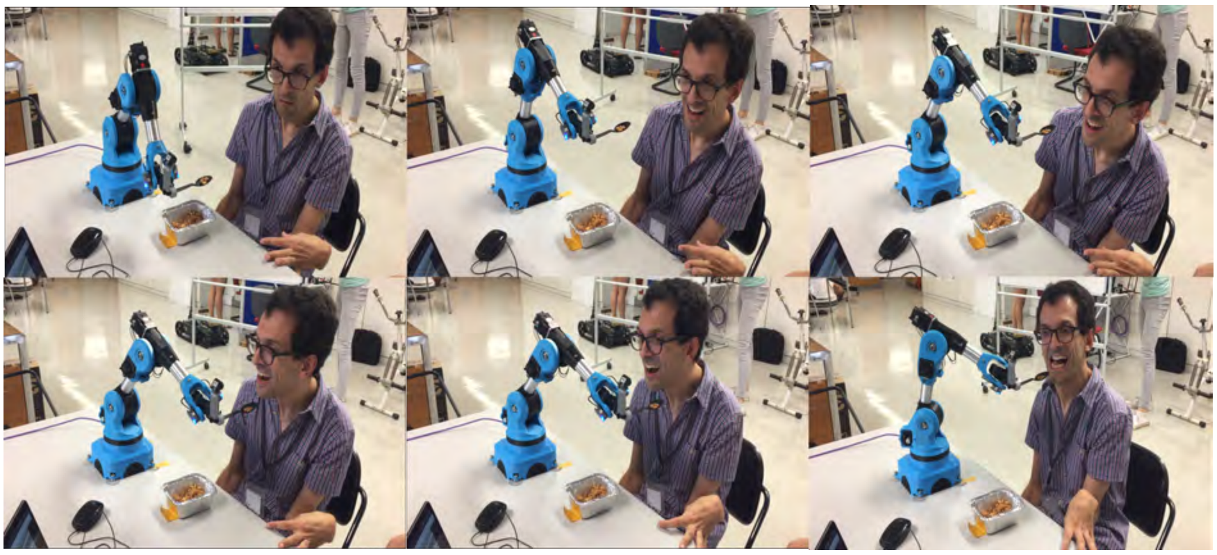


Figure 5.5: Feedbot feeding moving subject.

The aforementioned tests were not thought to exhaustively test Feedbot, the purpose of these tests was to do a preliminary test to see the practicability of Feedbot in a real meal scenario, more tests to the robustness of Feedbot will be done in future work. However, these preliminary tests confirm that Feedbot is capable of handle simple food types and can be used by different persons. Furthermore,

[2]In `https://drive.google.com/open?id=1ERdaiKPNRJwpXLIKbw1lN5gVGjOQ8NDw` it is available a video showing Feedbot performing real-time trajectory adaptation in this experiment

it is also confirmed that the different parts of Feedbot can work together and that Feedbot works with different robot arms.

5.2 3D Head Pose Tracking

5.2.1 Offline Validation

To validate the performance of Feedbot's tracking system, we used two public available RGB-D datasets for head pose estimation, namely, Biwi [46] and ICT3DHP [58]. The evaluation on these datasets was done offline, this means that we give to our algorithm as much time as it needs to process each frame and that no frame is missed.

We tested our algorithm against two state of the art methods for 3D head pose estimation, namely, OpenFace [34] and Fanelli et al. method [46]. In our tests, we used the implementation provided by the authors with the default parameters. It is important to mention that the method proposed by Fanelli et al. [46] is not for head pose tracking, it is a frame by frame head pose estimation method. In our evaluation, when this method fails to compute the head pose in a given frame, we assume the last successful head pose estimation as result.

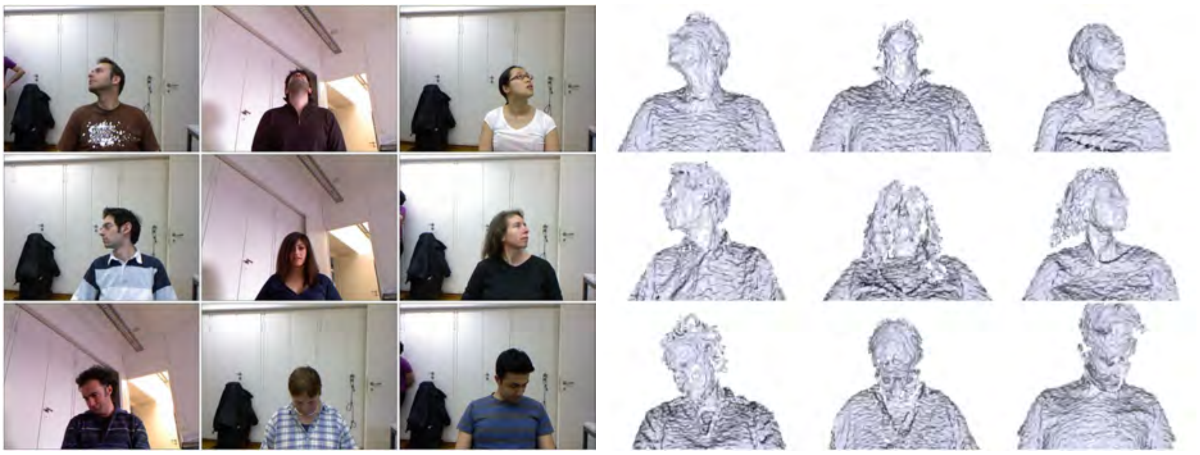
In all our experiments, we acquired a 3D head model using the method described in section 4.3.3. This model was acquired using the first frame on each of the dataset sequences. Using the model, we trained 30 SUM as described on algorithm 4.1, for training we used 30000 synthetically generated random rotations and translations of the model and including model occlusions, noise, and outliers.

5.2.1.A Biwi Dataset

Biwi dataset is composed by 24 RGB-D sequences of 20 different people (4 people were recorded twice), in figure 5.6 we present some examples of RGB and depth images of this dataset. Biwi contains a wide range of head poses and the ground truth is given in the form of the 3D head position and rotation in the camera frame.

This dataset was acquired with frame by frame estimation in mind and not for tracking, which means that some frames are missing. However, this dataset is widely used on head pose tracking benchmarks, mainly to test algorithms in difficult scenes [35], since there is a higher than normal temporal gap between some frames.

Due to the difficult nature of this dataset, DO fails to converge in 4 of the 24 sequences. For examples of these failures, we refer the reader to appendix C. We removed these sequences from the dataset in the comparison against OpenFace and Fanelli et al. OpenFace [34] has a failure detection step, thus it can recover from failure cases which is not possible with DO since we do not have a failure detection



(a) RGB images. **(b)** Depth images.

Figure 5.6: Examples of images present in Biwi dataset.

step. Fanelli's method [46] is based on frame by frame estimation, therefore, when it fails it simply starts again in the next frame and does not accumulate the error as DO.

The results for the 20 valid sequences, in terms of the MAE for the Euler angles of the head rotation are presented in table 5.1 and for the head position in table 5.2.

Method	Rx	Ry	Rz	Mean
DO(ours)	**2.83**	6.60	**4.70**	**4.71**
OpenFace [34]	7.55	6.12	10.02	7.90
Fanelli et al. [46]	3.91	**4.88**	10.69	6.49

Table 5.1: MAE Euler angles in degrees for the Biwi dataset.

Method	Tx	Ty	Tz	Mean
DO(ours)	0.84	0.75	1.03	0.87
OpenFace [34]	1.07	0.96	3.21	1.74
Fanelli et al. [46]	**0.32**	**0.28**	**0.27**	**0.29**

Table 5.2: MAE head position in cm for the Biwi dataset.

In this dataset, we can conclude that our method outperforms state of the art methods in terms of the MAE for the head rotation Euler angles. The analysis of the results show that our method has higher MAE for the Ry Euler angle. It was expected since is in this direction that people have a wider amplitude of movements with their heads. Rotations in the Ry direction correspond to movements like the ones present in figure 5.6 for the top left and top right subjects. Moreover, we conclude that OpenFace has more than twice the error that our method has in the Rx direction, this can be explained due to the occlusion of the face when people rotate their heads in this direction, movements of this type are present in the middle column images of figure 5.6. Face occlusions create problems in methods that depend on facial landmarks detection like OpenFace. For the head position, we conclude that our method can still be competitive relative to other state of the art methods and provides a good accuracy with a MAE less than 1 cm.

5.2.1.B ICT3DHP Dataset

ICT3DHP [58] is composed by 10 RGB-D sequences of 10 different people. In figure 5.7 we present an example of a sequence present in this dataset. It was acquired with the purpose of benchmarking head pose tracking methods, however, the ground truth is only precise for the rotation of the head. The head annotated position is referenced by the authors as not precise, due to this we only present results for the estimation of the head rotation.

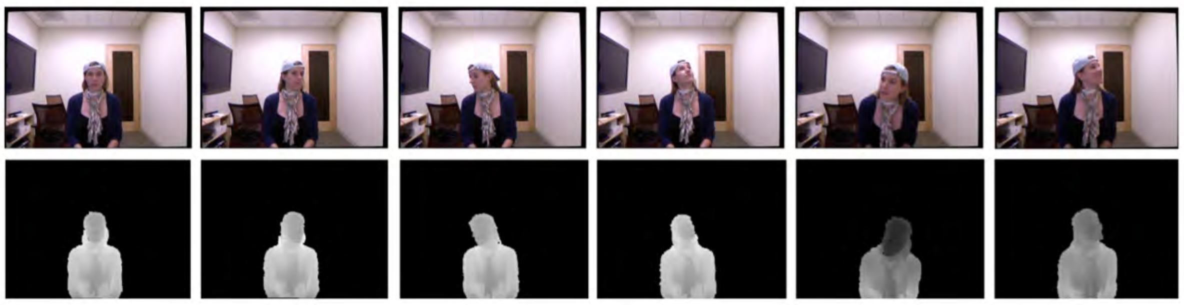


Figure 5.7: Example of a RGB(top) and Depth(down) sequence present in the ICT3DHP dataset, figure from [59].

We present in table 5.3 the MAE for the rotation Euler angles in the ICT3DHP dataset.

Method	Rx	Ry	Rz	Mean
DO(ours)	**3.06**	6.88	2.89	4.28
OpenFace [34]	3.23	**3.44**	**2.88**	**3.19**
Fanelli et al. [46]	7.76	8.27	7.69	7.91

Table 5.3: MAE Euler angles in degrees for the ICT3DHP dataset.

Our results in ICT3DHP dataset confirm the consistency and competitive performance of our method across different datasets. We can conclude that the performance of Fanelli et al. is worst than in the Biwi dataset, this can be explained due to the use of the default parameters set by the authors, these parameters were tuned for the Biwi dataset. Although OpenFace has the better MAE results in this dataset, we can see that in the Rx direction DO still has the better results, thus confirming our, previously discussed, idea that OpenFace fails when dealing with face occlusions.

The evaluation of DO across these two different datasets reveals that it is capable of achieving better, or equal performance when compared with other state of the art head pose estimation algorithms. Moreover, DO reveals a better generalization to different datasets, is not sensitive to parameter tuning and is capable of dealing with extreme face occlusions.

5.2.2 Real-Time Validation

To test the performance of our tracking system in real-time, we acquired a RGB-D video of a person moving his head in front of the camera, using ROS bag system. Using ROS, we ensure that we can simulate the real-time application. Using the recorded video if the tracking takes more computation time than the camera acquisition time, some frames are skipped. In the first part of the video, we ask the user to look in different directions but to keep his torso still. In the second part, we ask the user to move his head to different places in addition to looking in different directions.

The error of our tracking system was measured, in terms of absolute error and MAE, by annotating the mouth point in our dataset at a frame-rate of 2 FPS. The error in each frame is presented in Figure 5.8. The experiment was performed on a single thread on a Intel Core i7-6500U CPU 2.50GHz with 8GB of memory. Our tracking achieved speeds of 28FPS.

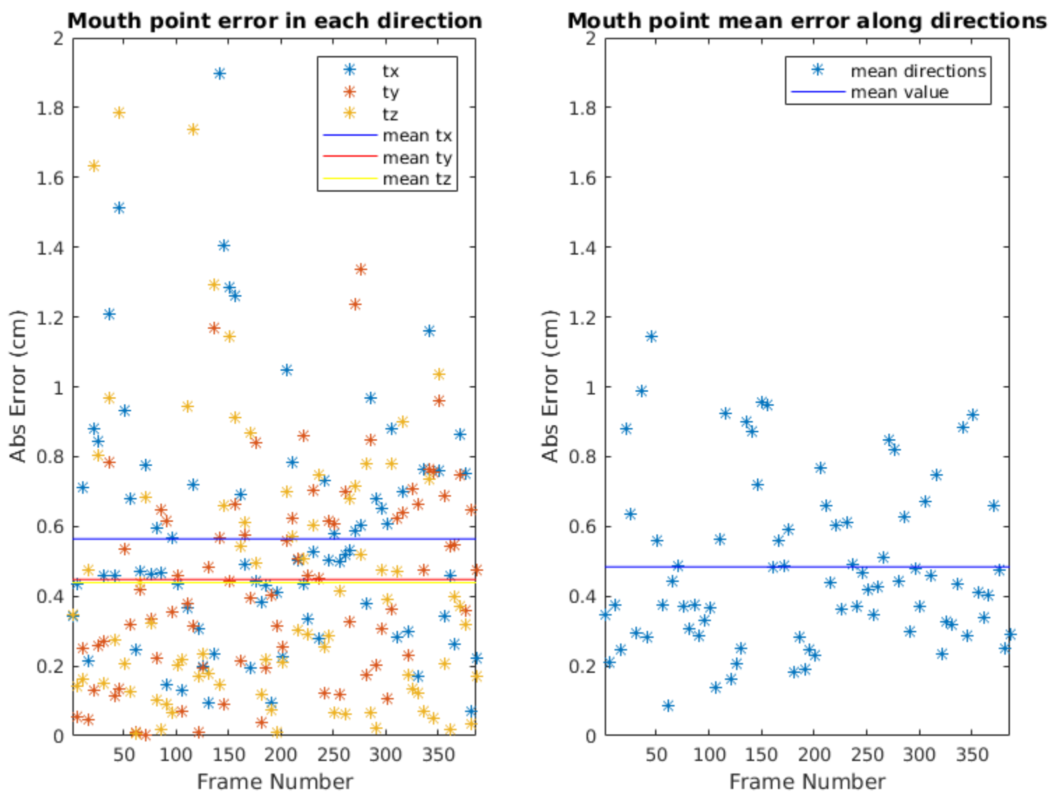


Figure 5.8: Tracking error of DO in each frame. On the left, the asterisks represent the absolute value of the error in each frame for each of the directions, the lines represent the MAE along the sequence for each of the directions. On the right, the asterisks represent the MAE for each of the frames and the line represent the mean of these values along all frames of the sequence.

Figure 5.8 shows that our tracking method can track the person's mouth with an absolute error below 2cm through the entire video. This is on the order of magnitude of the mouth's size, so the output of our system will not be far from the real location of the mouth. These results are of the same order of magnitude as the ones present on table 5.2, thus demonstrating the consistency of our results.

To have a better visual insight about the output of our tracking method, we show in red on Figure 5.9 the output mouth point of our algorithm.

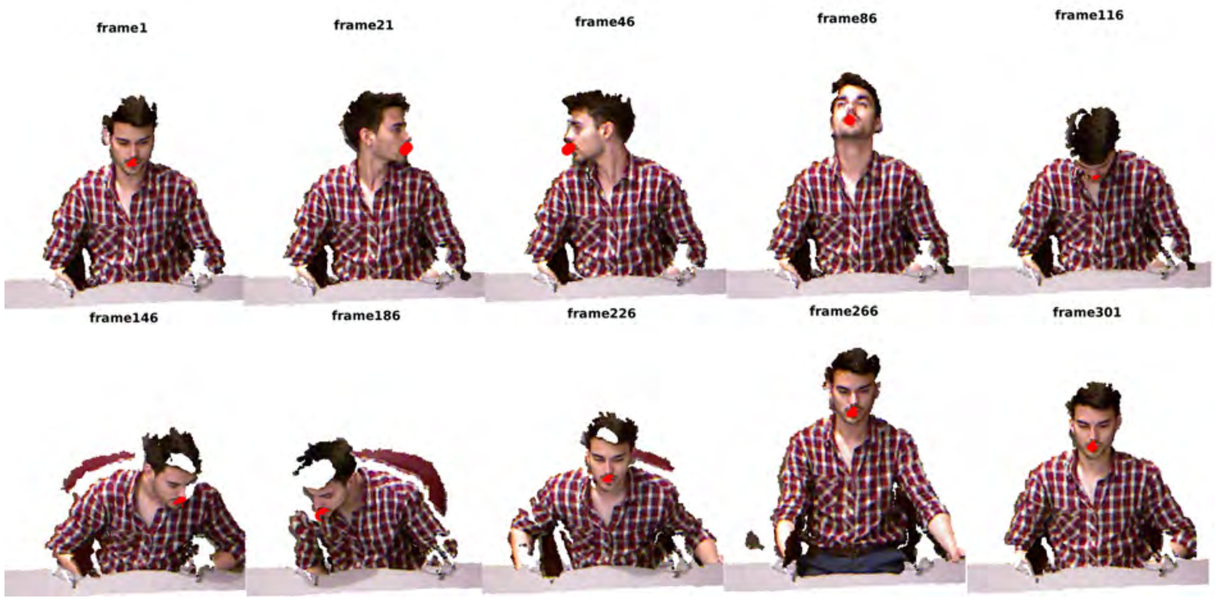


Figure 5.9: Tracking output for the user's mouth.

5.3 Food Detection

In this section, we evaluate the performance of our two food detection methods. To train the classifiers and the linear regression we used scikit-learn [60]. We split our datasets into 70% train data and 30% test data. To evaluate and train our algorithms for food detection, we gathered two small datasets, one containing 387 images labeled with the weight of food and another one with 119 images of empty spoons and 118 images of non-empty spoons. We present in figure 5.10 examples of the pictures gathered for these datasets.

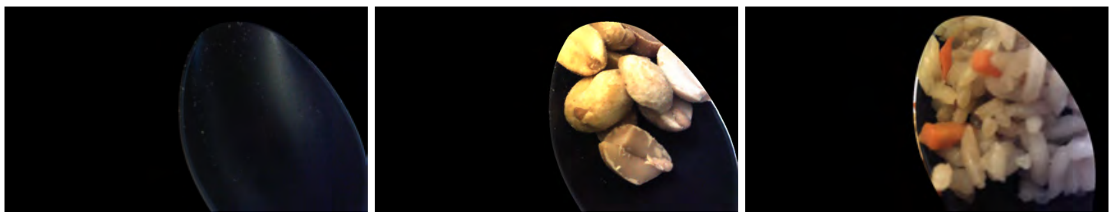


Figure 5.10: Example masked images of the plastic spoon, showing the spoon empty, with nuts, and with rice.

5.3.1 Food Weight Estimation

Figure 5.11 shows the linear regression fit modeling the weight of food in the spoon with the correlation of histograms between the current image of the spoon and the image of the empty spoon. The correlation measure is computed, as shown in Equation 4.17, by taking the histogram of the image of the spoon with food on it and correlating that with the histogram of the baseline image of the spoon without food on it.

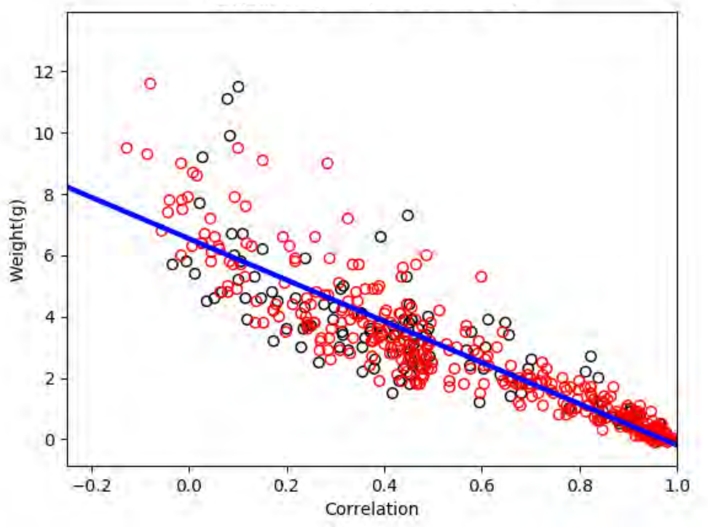


Figure 5.11: Relationship between the correlation measure and the weight of food on the spoon. The data points used for training are shown in red, the data used for test is shown in black, and the linear regression is shown in blue.

The regression has a mean squared error of 1.37g and a correlation coefficient (r^2) of 0.75 on the test set. There is a strong negative correlation between the weight of the food that is on the spoon and the correlation between the histograms. Even if the model is not perfect in measuring the amount of food that is on the spoon, we can use this model to calibrate our correlation threshold to require more or less food per bite.

5.3.2 Empty and Non-empty Spoon Classification

In the classification task, we achieved an accuracy of 95.8% with the logistic regression and 98.6% with the linear SVM. The SVM was trained using the whole H channel histogram and the logistic regression using only the correlation between each sample of the train data and an empty spoon histogram example. We present, in Figures 5.12 and 5.13, the confusion matrix results for the two classification methods.

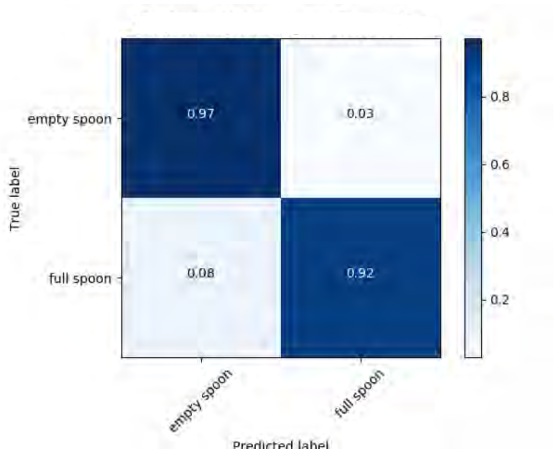


Figure 5.12: Confusion Matrix for Logistic Regression

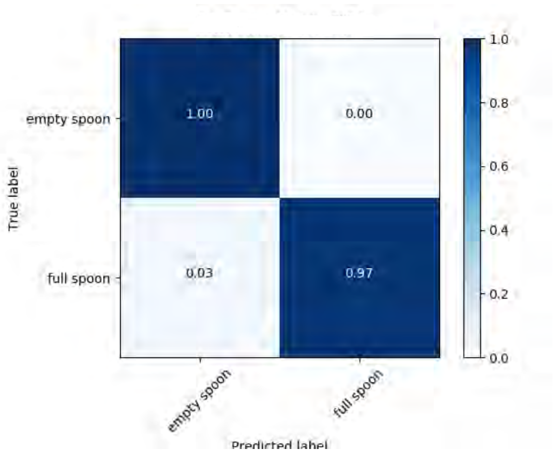


Figure 5.13: Confusion Matrix for SVM

These results validate our classification approach to discover if there is still food on the spoon. More interestingly, looking at the confusion matrix for the linear SVM (figure 5.13), we can see that it will not classify an empty spoon as a full spoon. This is very important since it means that using this approach Feedbot will not give empty spoons to the user.

5.4 Ablation Study Food Acquisition

Setup

To analyze the importance of various components of Feedbot, we perform an ablation study in the feeding task where we measure the efficiency of Feedbot when it does and does not use certain food acquisition strategies. We define the efficiency to be the mass of food that is delivered to the feeding location as a function of the total distance that the tip of the spoon has traveled. We report the mass as a function of distance traveled rather than the time taken because we want to negate any effect that setting the robot to a faster or slower speed would have on the results.

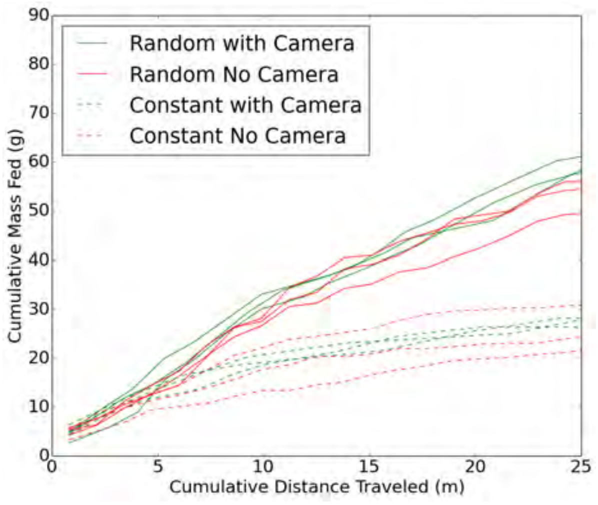
The Feedbot components that we alter in our ablation study are 1) whether or not we re-scoop if the spoon-facing camera detects that not enough food was acquired and 2) whether Feedbot always scoops from the same position on the serving plate or whether it scoops from a uniformly random position within a 6cm x 3cm rectangle on the plate. We perform the ablation experiment on two different kinds of food: peanuts and fried rice. In this way, we can determine if the importance of system components depends on the type of food.

For consistency in results, we use the same random seed for all trials that randomize the scooping location on the plate. To speed up data collection, the robot dumps the food directly onto the scale after food acquisition. Then, in our data analysis, we add in the distance to the user's mouth and back.

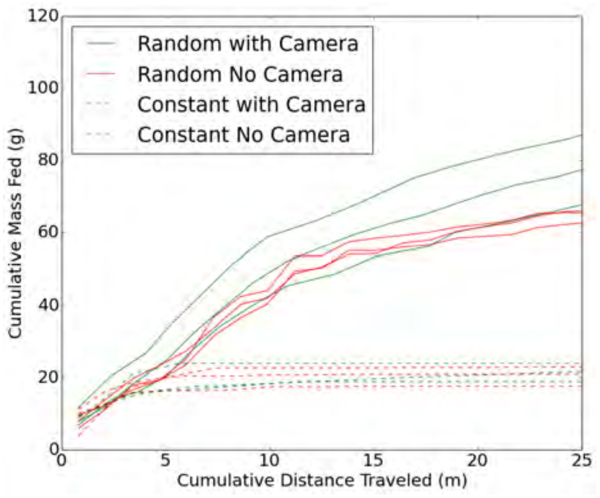
Cutting out the travel time between the plate and the user's mouth cuts the data acquisition time by a factor of three On average, the distance traveled by the spoon tip in a single scooping motion is 32cm and the average distance to and from the user's mouth is 49cm in each direction. We use these average distance values when representing the amount of food served as a function of distance.

Results

In our ablation study for food acquisition, we find that vision feedback significantly increases the amount of food brought to the user in each bite. For the spoon-facing vision feedback system, we use a histogram correlation cutoff of 0.5 for both rice and nuts. For rice, when randomizing the scooping location, the average amount of rice served per bite in the first 20 bites across three trials is 4.8g with camera feedback and 3.2g without camera feedback. Likewise, for nuts it is 3.8g with camera feedback and 2.7g without. There is an added distance that the end-effector travels in re-scooping when using camera feedback. Despite this added cost in distance, Figure 5.14 shows that even when looking at the amount of food served as a function of the total distance that the spoon tip travels (including the added distance required to re-scoop), after 25m random scooping with camera feedback consistently outperforms random scooping without camera feedback.



(a) Feeding Nuts



(b) Feeding Rice

Figure 5.14: Charts showing the amount of food fed (in grams) as a function of the distance the tip of the spoon has traveled, including the distance traveled in re-scooping. The chart includes results for scooping location randomization (solid lines) compared to a constant scooping location (dotted lines), and it includes results with (green lines) and without (red lines) camera feedback on whether food was successfully acquired. We ran three trials for each test type.

For both rice and nuts we find that randomization of the scooping location is a useful technique in food acquisition. Figure 5.14 also shows that randomization coupled with visual feedback on the success of a scoop is the most effective. Most interestingly, we find that the importance of our system components depends on the type of food used. We find that randomizing the location of scoops is more important when serving rice instead of nuts. This could be related to the fact that nuts were observed to “settle” after each scoop, whereas fried rice will tend not to fill in the hole left after scooping. Thus, scooping the same place for fried rice will quickly result in very little (almost no) mass acquired by the spoon in subsequent scoops.

6

Conclusions and Future Work

In this work, we explored how vision can be introduced in a robotic feeding platform. By introducing a vision-based approach in Feedbot, we made it more aware of the environment and more autonomous, making it capable of adapting its trajectories to the user in real-time. Feedbot is fully capable of feeding different types of food to a user using a spoon. We tested Feedbot using two different robot arms, Kinova MICO and Niryo One, proving that it can be used in different types of robot arms. Using a robot arm like Niryo can help us to highly deploy our Feedbot in institutions, due to the low cost of this arm.

We applied DO to the task of head pose tracking showing that this method is capable of achieving the same, or even better, performance than state of the art methods. DO was evaluated in terms of the speed performance achieving real-time performance, thus showing its potential to be used for real-time applications like robotics. However, in our tests, we also understood the limitations of using DO for tracking without using a validation step. Future work will be done to improve the robustness of our head pose tracking method.

Food detection methods, based on visual information, were introduced in this work. We found that using very simple models, based on visual information, can achieve good performance. This is very important, because these methods, due to their simple nature, can be applied in real-time since they do not have high computational cost.

Using an ablation study, we measured the impact of visual feedback in the food acquisition step. We show that the use of visual feedback significantly impacts the performance of robotic feeding platforms in terms of the food delivered to the user. This is an important finding, on one hand, it shows the need to have food detection methods in feeding robots, on the other hand, it shows that current commercial available feeding robots are not efficient since they do not have any type of feedback in the food acquisition step.

Future work to fill the gaps in the Feedbot's state machine (figure 3.3) needs to be done. We remember that some of the transitions, that are related to the detection of the user's intention, are still time-based. Modeling the user's intention is a very difficult problem since, to the best of our knowledge, there are no good mathematical models for that. A possible way of contributing for solving this problem could be based on the user facial expression, pose, and some type of supervised signals from caregivers (e.g trajectories of the spoon during a meal), try to detect patterns that reveal the user's intention to be feed. Another topic that still needs research is the problem of controlling and adapting the trajectory of the robot as the plate gets empty. This can be done, for example, using visual feedback on the plate and on the spoon.

Finally, we are planning to test and evaluate Feedbot in meal scenarios with more people that suffer from upper-extremity disabilities. We plan to quantitatively and qualitatively analyze the efficacy of Feedbot. We will also compare Feedbot to the type of baseline in current commercially-available robots.

Bibliography

[1] "World Population Prospects: The 2017 Revision — Multimedia Library - United Nations Department of Economic and Social Affairs." [Online]. Available: https://www.un.org/development/desa/publications/world-population-prospects-the-2017-revision.html

[2] "Population ages 65 and above (% of total) — Data." [Online]. Available: https://data.worldbank.org/indicator/SP.POP.65UP.TO.ZS?end=2017&name_desc=false&start=1988&view=chart

[3] R. K. Al-Halimi and M. Moussa, "Performing Complex Tasks by Users with Upper-Extremity Disabilities Using a 6-DOF Robotic Arm: A Study," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 6, pp. 686–693, 6 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7553526/

[4] "Niryo Smart Collaborative Robot For You." [Online]. Available: https://niryo.com/

[5] D. Feil-Seifer and M. J. Matarić, "Defining socially assistive robotics," in *Proceedings of the 2005 IEEE 9th International Conference on Rehabilitation Robotics*, vol. 2005. IEEE, 2005, pp. 465–468. [Online]. Available: http://ieeexplore.ieee.org/document/1501143/

[6] "Augmented Human Assistance." [Online]. Available: http://aha.isr.tecnico.ulisboa.pt/

[7] S. D. Klee, B. Q. Ferreira, R. Silva, J. P. Costeira, F. S. Melo, and M. Veloso, "Personalized assistance for dressing users," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9388 LNCS, pp. 359–369, 2015.

[8] Z. Erickson, M. Collier, A. Kapusta, and C. C. Kemp, "Tracking Human Pose During Robot-Assisted Dressing using Single-Axis Capacitive Proximity Sensing," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2245–2252, 9 2017. [Online]. Available: http://arxiv.org/abs/1709.07957

[9] K. P. Hawkins, P. M. Grice, T. L. Chen, C. H. King, and C. C. Kemp, "Assistive mobile manipulation for self-care tasks around the head," in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIR2AT 2014: 2014 IEEE Symposium on Computational Intelligence in Robotic Rehabilitation and Assistive Technologies, Proceedings*, 2014, pp. 16–25.

[10] S. Schröer, I. Killmann, B. Frank, M. Völker, L. Fiederer, T. Ball, and W. Burgard, “An autonomous robotic assistant for drinking,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June. IEEE, 5 2015, pp. 6482–6487. [Online]. Available: http://ieeexplore.ieee.org/document/7140110/

[11] M. Topping, “Early experience in the use of the ‘Handy 1’ robotic aid to eating,” *Robotica*, vol. 11, no. 06, p. 525, 11 1993. [Online]. Available: http://www.journals.cambridge.org/abstract_S0263574700019366

[12] S. Ishii, S. Tanaka, and F. Hiramatsu, “Meal assistance robot for severely handicapped people,” in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1995, pp. 1308–1313. [Online]. Available: http://ieeexplore.ieee.org/document/525461/

[13] “Obi — Robotic feeding device designed for home care.” [Online]. Available: https://meetobi.com/

[14] “SECOM - Meal-assistance Robot.” [Online]. Available: https://www.secom.co.jp/english/myspoon/

[15] “Meal Buddy,” p. 3. [Online]. Available: https://www.performancehealth.com/meal-buddy-systems

[16] “Assistive Innovations - iEAT Feeding Robot.” [Online]. Available: https://www.assistive-innovations.com/en/eatingdevices/ieat-feeding-robot

[17] “Comanio Care: Bestic.” [Online]. Available: https://www.camanio.com/us/products/bestic/

[18] I. Naotunna, C. J. Perera, C. Sandaruwan, R. Gopura, and T. D. Lalitharatne, “Meal assistance robots: A review on current status, challenges and future directions,” in *2015 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 12 2015, pp. 211–216. [Online]. Available: http://ieeexplore.ieee.org/document/7404980/

[19] P. Lopes, R. Lavoie, R. Faldu, N. Aquino, J. Barron, M. Kante, B. Magfory, and W. Meleis, “Eye - Controlled Robotic Feeding Arm Technology (iCRAFT),” pp. 1–24, 2011. [Online]. Available: http://www.ece.neu.edu/personal/meleis/icraft.pdf

[20] B. House, J. Malkin, and J. Bilmes, “The VoiceBot: A Voice Controlled Robot Arm,” Tech. Rep., 2009. [Online]. Available: http://www.lynxmotion.com/Category.aspx?

[21] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–511–I–518, 2001. [Online]. Available: http://ieeexplore.ieee.org/document/990517/

[22] C. J. Perera, T. D. Lalitharatne, and K. Kiguchi, "EEG-controlled meal assistance robot with camera-based automatic mouth position tracking and mouth open detection," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 5 2017, pp. 1760–1765. [Online]. Available: http://ieeexplore.ieee.org/document/7989208/

[23] "Overview — Willow Garage." [Online]. Available: http://www.willowgarage.com/pages/pr2/overview

[24] D. Park, Y. K. Kim, Z. M. Erickson, and C. C. Kemp, "Towards Assistive Feeding with a General-Purpose Mobile Manipulator," Tech. Rep., 5 2016. [Online]. Available: http://arxiv.org/abs/1605.07996

[25] D. Park, H. Kim, and C. C. Kemp, "Multimodal anomaly detection for assistive robots," *Autonomous Robots*, pp. 1–19, 2018. [Online]. Available: https://doi.org/10.1007/s10514-018-9733-6

[26] C. Silva, J. Vongkulbhisal, M. Marques, J. P. Costeira, and M. Veloso, "Feedbot - A robotic arm for autonomous assisted feeding," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10423 LNAI, pp. 486–497, 2017. [Online]. Available: http://link.springer.com/10.1007/978-3-319-65340-2_40

[27] J. Vongkulbhisal, F. De La Torre, and J. P. Costeira, "Discriminative optimization: Theory and applications to point cloud registration," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 3975–3983, 2017. [Online]. Available: http://arxiv.org/abs/1707.04318

[28] T. Rhodes and M. Veloso, "Robot-driven Trajectory Improvement for Feeding Tasks," in *IEEE Internation Conference on Intelligent Robots Robots*, 2018. [Online]. Available: http://www.cs.cmu.edu/~mmv/papers/18iros-RhodesVeloso.pdf

[29] T. Bhattacharjee, H. Song, G. Lee, and S. S. Srinivasa, "Food manipulation: A cadence of haptic signals," 2018. [Online]. Available: http://arxiv.org/abs/1804.08768

[30] L. V. Herlant, S. S. Srinivasa, C. G. Atkeson, C. Ri, J. Forlizzi, C. Hcii, L. A. Takayama, and S. Cruz, "Algorithms, Implementation, and Studies on Eating with a Shared Control Robot Arm," Ph.D. dissertation, 2018. [Online]. Available: http://www.cs.cmu.edu/afs/cs/user/lcv/www/herlant-thesis.pdf

[31] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 1, pp. 23–38, 1998. [Online]. Available: http://www-prima.imag.fr/jlc/Courses/2016/PRML/rowley-ieee-PAMI.pdf

[32] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Processing Letters*, vol. 23,

no. 10, pp. 1499–1503, 4 2016. [Online]. Available: http://arxiv.org/abs/1604.02878http://dx.doi.org/10.1109/LSP.2016.2603342

[33] H. Wang, Z. Li, X. Ji, and Y. Wang, "Face R-CNN," Tech. Rep., 2017. [Online]. Available: https://arxiv.org/pdf/1706.01061.pdfhttp://arxiv.org/abs/1706.01061

[34] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L.-P. Morency, "OpenFace 2.0: Facial Behavior Analysis Toolkit," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 5 2018, pp. 59–66. [Online]. Available: https://ieeexplore.ieee.org/document/8373812/

[35] T. Baltrusaitis, P. Robinson, and L. P. Morency, "OpenFace: An open source facial behavior analysis toolkit," in *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*. IEEE, 3 2016, pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/document/7477553/

[36] L. A. Jeni, J. F. Cohn, and T. Kanade, "Dense 3D face alignment from 2D videos in real-time," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, FG 2015*, no. MAY, 2015. [Online]. Available: http://lightside.hu/pub/articles/Jeni15FG_ZFace.pdf

[37] X. Xiong and F. De La Torre, "Supervised descent method and its applications to face alignment," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 6 2013, pp. 532–539. [Online]. Available: http://ieeexplore.ieee.org/document/6618919/

[38] A. Asthana, S. Afeiriou, S. Cheng, and M. Pantic, "Incremental Face Alignment in the Wild," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2014. [Online]. Available: https://ibug.doc.ic.ac.uk/media/uploads/aasthanacvpr2014.pdf

[39] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face Alignment Across Large Poses: A 3D Solution," in *CVPR IEEE Conference*, 2016. [Online]. Available: http://www.cbsr.ia.ac.cn/users/

[40] T. Baltrušaitis, P. Robinson, and L. P. Morency, "Constrained local neural fields for robust facial landmark detection in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 354–361. [Online]. Available: https://www.cl.cam.ac.uk/~tb346/pub/papers/iccv2013.pdf

[41] P. Dollár, P. Welinder, and P. Perona, "Cascaded pose regression," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1078–1085. [Online]. Available: https://pdollar.github.io/files/papers/DollarCVPR10pose.pdf

[42] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. [Online]. Available: https://link.springer.com/content/pdf/10.1023%2FB%3AVISI.0000029664.99615.94.pdf

[43] X. Zhu, X. Liu, Z. Lei, and S. Z. Li, "Face Alignment In Full Pose Range: A 3D Total Solution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8122025/

[44] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[45] S. Gold, C. P. Lui, and A. Rangarajan, "New Algorithms for 2D and 3D Point Matching : Pose Estimation and Correspondence," *Pattern Recognition*, vol. 31, no. 8, 1999. [Online]. Available: https://papers.nips.cc/paper/977-new-algorithms-for-2d-and-3d-point-matching-pose-estimation-and-correspondence.pdf

[46] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. van Gool, "Random Forests for Real Time 3D Face Analysis," *International Journal of Computer Vision (IJCV)*, vol. 101, no. 3, pp. 437–458, 2013. [Online]. Available: https://data.vision.ee.ethz.ch/cvl/gfanelli/pubs/ijcv.pdf

[47] G. Fanelli, T. Weise, J. Gall, and L. Van Gool, "Real Time Head Pose Estimation from Consumer Depth Cameras," in *Proceedings of the DAGM Symposium on Pattern Recognition*, 2011, pp. 101–110. [Online]. Available: https://data.vision.ee.ethz.ch/cvl/gfanelli/pubs/dagm11.pdfhttp://link.springer.com/10.1007/978-3-642-23123-0_11

[48] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *Computer Vision and Pattern Recognition*, 2011, pp. 617–624. [Online]. Available: https://data.vision.ee.ethz.ch/cvl/gfanelli/pubs/cvpr11.pdfhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5995458

[49] J. Vongkulbhisal, F. De La Torre, and J. P. Costeira, "Discriminative Optimization: Theory and Applications to Computer Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8338139/

[50] J. Vongkulbhisal, "Discriminative Optimization: Theory and Applications to Computer Vision Thesis," Ph.D. dissertation, CMU, 2018. [Online]. Available: http://users.isr.ist.utl.pt/~jpc/ftp/thesisjayakorn.pdf

[51] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-D rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, pp. 272–290,

1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.2196&rep=rep1&type=pdf

[52] Y. Tsin and T. Kanade, “A Correlation-Based Approach to Robust Point Set Registration,” in *ECCV 2004. Lecture Notes in Computer Science, vol 3023*, 2004, pp. 558–569. [Online]. Available: http://www.cs.cmu.edu/~ytsin/KCReg/KCReg.pdfhttp://link.springer.com/10.1007/978-3-540-24672-5_44

[53] M. Daszykowski and B. Walczak, “Density-Based Clustering Methods,” *Comprehensive Chemometrics*, vol. 2, pp. 635–654, 2010. [Online]. Available: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

[54] S. Suzuki and K. be, “Topological structural analysis of digitized binary images by border following,” *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 4 1985. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0734189X85900167

[55] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. [Online]. Available: http://ieeexplore.ieee.org/document/4310076/

[56] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[57] M. Grant and S. Boyd, “CVX: Matlab Software for Disciplined Convex Programming, version 2.1,” http://cvxr.com/cvx, 3 2014.

[58] T. Baltrusaitis, P. Robinson, and L. P. Morency, “3D Constrained Local Model for rigid and non-rigid facial tracking,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2610–2617. [Online]. Available: http://www.cl.cam.ac.uk/research/rainbow/emotions/

[59] T. Baltrušaitis, “Automatic facial expression analysis,” Ph.D. dissertation, University of Cambridge, 2014. [Online]. Available: https://www.cl.cam.ac.uk/~tb346/pub/thesis/phd_thesis.pdf

[60] P. A. C. D. B. M. P. M. D. E. Vanderplas, J., “Scikit-learn: Machine learning in Python.” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[61] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

[62] I. Sucan and S. Chitta, “MoveIt.” [Online]. Available: http://moveit.ros.org/

[63] R. Diankov, “Automated Construction of Robotic Manipulation Programs,” Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, 8 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

[64] C. Welman, “Inverse kinematics and geometric constraints for articulated figure manipulation,” *Science*, vol. C, no. April, p. 77, 1993. [Online]. Available: http://run.usc.edu/cs520-s15/ik/welman.pdfhttp://lib-ir.lib.sfu.ca/bitstream/1892/7119/1/b15233406.pdf

A

Appendix A: Cameras

The simplest mathematical model for a camera is the pinhole model where each 3D point((x, y, z)), represented in the camera coordinate system, is transformed to a position in the image plane((u, v)). The image plane coordinates are given by the perspective projection:

$$\begin{aligned} u &= -f\frac{x}{z} \\ v &= -f\frac{y}{z}. \end{aligned} \tag{A.1}$$

Since most of the available cameras are digital, we have to transform these coordinates to the pixel sensor spacing [61]. A more realistic model is given by:

$$\begin{aligned} u &= \frac{f_x x}{z} + c_x \\ v &= \frac{f_y y}{z} + c_y, \end{aligned} \tag{A.2}$$

where f_x and f_y denotes independent focal lengths in each of the directions, and c_x and c_y denotes the optical center of the camera expressed in pixel coordinates on the image plane.

We can express this model in homogeneous coordinates in a matrix form:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{A.3}$$

The pinhole model is not very accurate due to the distortion introduced by the camera lens. We could also include distortion models in the camera model such as radial distortion [61]. Introducing these distortion models introduces also complexity in the model, being impossible of writing the camera model as a linear transformation.

There are many types of cameras available on the market. The most common cameras are the RGB cameras. These cameras in each pixel store a value related to the intensity of the radiation color of the objects present in the image. Although RGB cameras are good for perceiving the world, due to the projective nature of the camera it is very difficult to recover the 3D coordinates for each pixel of the image.

Stereo cameras are composed of 2 RGB cameras and can recover the 3D coordinates for each pixel based on the matching of pixels that correspond to the same 3D location. If we know the matching pixels we can solve equation (A.3) since we have 3 unknowns and 4 equations. However, to get the matching between the pixels in two different RGB images is a hard computer vision problem.

Active stereo cameras instead of using two cameras use a projector and a camera, the projector projects a light pattern to the 3D objects making it easy to match the points between the virtual projector camera and the true camera. An example of a camera of this type is the Kinect V1.

Depth cameras, like stereo or active stereo cameras, provide a measure of the depth from the camera to the object. In these cameras, each pixel stores the value of the distance from the camera to the objects along the z-axis. With the value of the depth and the camera parameters we can simply compute the other coordinates doing:

$$\begin{aligned} x &= \frac{Z}{f_x} u - c_x \\ y &= \frac{Z}{f_y} v - c_y. \end{aligned} \tag{A.4}$$

B

Appendix B: Robot Arms

In terms of a mathematical model for a robot arm, we can simply think that a robot arm is a non-linear function, $\mathcal{F}(\Theta) : \mathbb{R}^{DOF} \rightarrow \mathbb{R}^6$, that maps the angle in each joint's motor to the end-effector position and orientation. We assume that the output of this function is a $\mathbb{R}^6$ vector where the first 3 entries are the position and the other entries are the orientation of the end-effector in an axis-angle representation, the output coordinates are related with the base frame of the robot arm. This function is usually called Forward Kinematics of the robot arm. It can be automatically computed using an Unified Robot Description Format (URDF) file description of the arm and MoveIt! [62] or OpenRave [63].

Usually, we are interested in computing the joint angles, Θ, that can make the end-effector move to a goal position $x \in \mathbb{R}^6$. This is known as the Inverse Kinematics problem which is given by:

$$\Theta = \mathcal{F}^-1(x). \tag{B.1}$$

Since $\mathcal{F}$ is a non-linear function, solving (B.1) is a difficult problem which involves solving a system of non-linear equations.

A naive approach to solve (B.1) is linearize the problem [64],

$$\dot{x} = \frac{\partial \mathcal{F}(\Theta)}{\partial \Theta}\dot{\Theta}, \tag{B.2}$$

and solve for small increments around Θ_0, this results in a linear system of equations given by:

$$\begin{aligned} \Delta x &\approx J\Delta\Theta \\ \equiv \Delta\Theta &\approx J^{-1}\Delta x, \end{aligned} \tag{B.3}$$

where $J = \frac{\partial \mathcal{F}(\Theta)}{\partial \Theta}|_{\Theta_0}$. We assume that J has an inverse J^{-1} which is not always true, for example in robot arms with 5DOF the matrix J is not square. This approach also fails when the robot arm is near singular configurations. To solve for positions far away from the initial configuration of the robot Θ_0, we can still use this approach doing it in an incremental way.

C

Appendix C: Biwi Dataset Supplementary Results

In this appendix we present supplementary results on the Biwi dataset.

As we discussed, in section 5.2.1.A, DO has problems in some sequences of the Biwi dataset. In figure C.1 we present the results for the MAE of the Euler angles in each of the sequences. We can see that in sequences 2,3,6 and 15 the MAE error is high when compared with the other sequences. These sequences were removed from the evaluation of section 5.2.1.A.

To better illustrate how is the error across frames for a sequence were DO failed to converge, we show in figure C.2 the MAE for each of the frames for sequence 6.

We can see clearly that DO after 150 frames diverge and never converges again. Since all the methods have a large error in this region of the sequence we can conclude that in that region is very difficult to track the head-pose. For the other methods, even if the error is very large near frame 150, they can recover in the next frames since OpenFace has a validation step and Fanelli is based on frame by frame estimation.

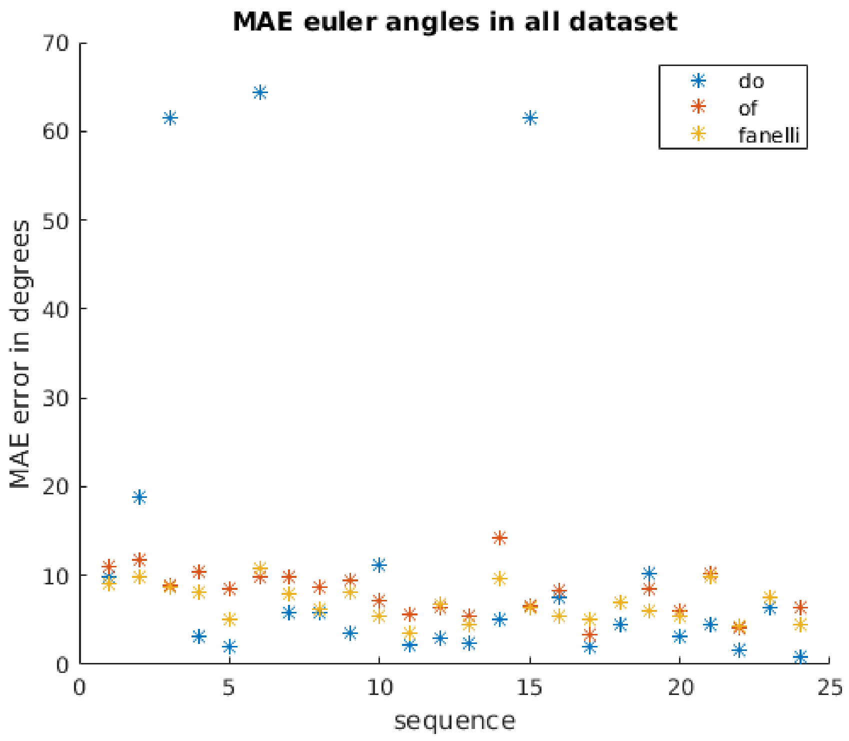


Figure C.1: MAE in all sequences of Biwi dataset.

In figure C.3 we show how is the MAE across frames for sequence 17, where DO shows the best results. We can see that the behavior of the tested methods is very similar but DO shows lower MAE when other methods fail.

We can also see by figure C.3 and C.2 that there is evidence that when OpenFace starts to show a bigger error the same occurs with Fanelli.

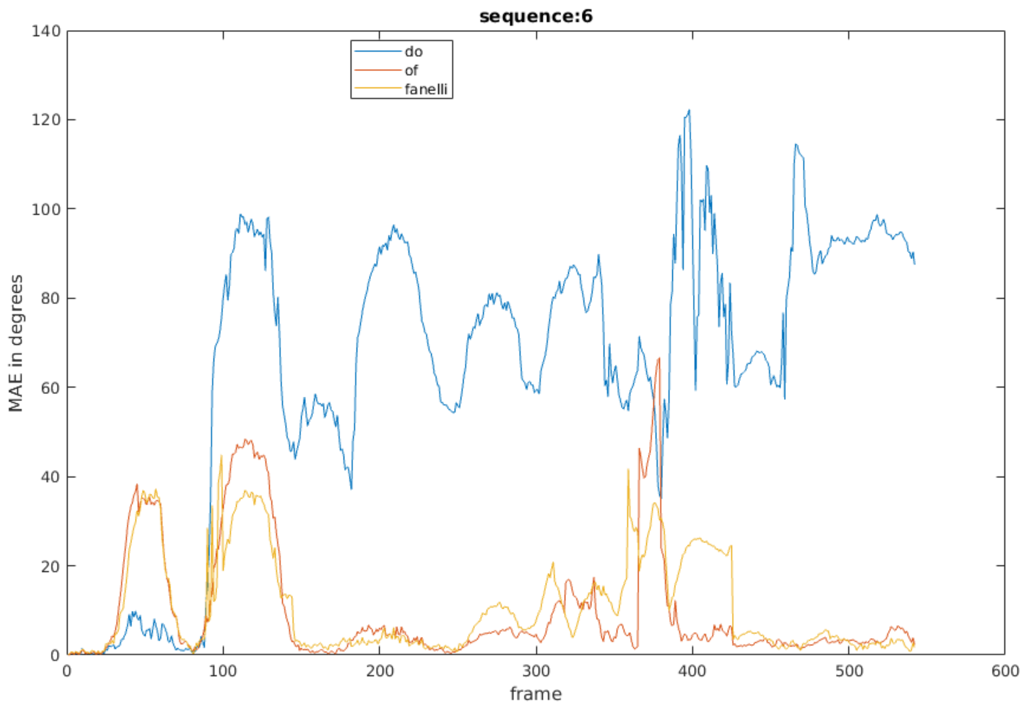


Figure C.2: MAE of sequence 6 of Biwi dataset.

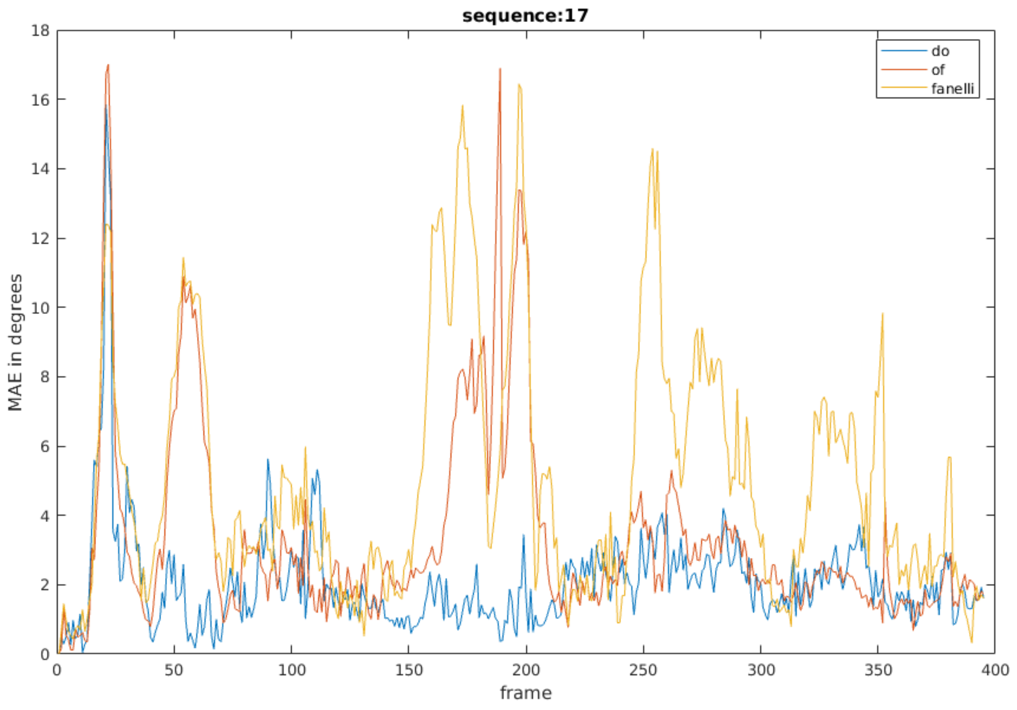


Figure C.3: MAE of sequence 17 of Biwi dataset.