

Titre: Assistive Robotic arm control using point of gaze Estimation
Title:

Auteur: Cédric Leblond-Ménard
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Leblond-Ménard, C. (2023). Assistive Robotic arm control using point of gaze Estimation [Thèse de doctorat, Polytechnique Montréal]. PolyPublie.
Citation: <https://publications.polymtl.ca/53362/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/53362/>
PolyPublie URL:

**Directeurs de
recherche:** Sofiane Achiche
Advisors:

Programme: Génie mécanique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Assistive Robotic Arm Control Using Point of Gaze Estimation

CÉDRIC LEBLOND-MÉNARD

Département de génie mécanique

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*

Génie mécanique

Avril 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Cette thèse intitulée :

Assistive Robotic Arm Control Using Point of Gaze Estimation

présentée par **Cédric LEBLOND-MÉNARD**

en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

David SAUSSIÉ, président

Sofiane ACHICHE, membre et directeur de recherche

Abolfazl MOHEBBI, membre

Luciano OLIVEIRA, membre externe

DEDICATION

*To my fiancée Marika,
without whom I would have never
had the drive to complete this thesis.*

"I know that I know nothing"

- Socrates

or the more informal version:

"I'm smart enough to know that I'm dumb."

- Richard P. Feynman

*Science is a journey of which we forgot the starting point and for the end of which we are
oblivious to.*

ACKNOWLEDGEMENTS

I would first like to thank my research director, Professor Sofiane Achiche, for accepting me as his student and for his support, advices and friendliness throughout my time at his laboratory. It has been both a pleasure and an honor to work under his supervision and to bring to life this project, starting back from my Bachelor's degree 3rd year individual project.

I would also like to thank all of previous and current students of Pr. Achiche's laboratory, with whom we shared many laughs and great discussions on all matters, both relevant and completely irrelevant to our work. Philosophy was definitely at the center of several of our discussions. A special thanks to Yann-Seing Law-Kam Cio and Gabriel Picard-Krashevski, who both supported me in this project as co-supervisor of my internship at the start of this project and intern respectively, but most importantly friends and co-workers.

I would also like to extend my thanks to Pr. Reza Hoseinnezhad at RMIT University for his supervision during my visiting research fellowship at RMIT University.

I am also grateful to my fiancée, Marika Girard, and our families, who supported me through the best and worst times of this project. Their advice and open ears were invaluable in reaching the end of this journey.

Another special thanks to Jean-François Gamache at Kinova Inc. for both his help and interesting discussions, but also for providing us with a JACO robotic arm.

Finally, I would like to thank the National Sciences and Engineering Research Council of Canada (NSERC), the Fonds de recherche - Nature et technologies (FRQNT), the Trans-MedTech Institute and the Mitacs organization for their financial support through various scholarships.

RÉSUMÉ

Les avancées récentes de la robotique permettent aux personnes atteintes de troubles musculo-squelettique de s'équiper d'un bras robotique d'assistance pouvant être placé sur leur fauteuil motorisé. Il a été démontré que ces bras robotiques améliorent l'autonomie et l'habileté de ces personnes, notamment pour accomplir les tâches importantes du quotidien, telles que se nourrir seul.

Cependant, certains problèmes perdurent, en particulier le fait que le temps pour accomplir les manœuvres à l'aide d'un bras robotique reste encore élevé, en particulier pour les personnes atteintes de troubles musculo-squelettiques. Une des manœuvres les plus longues est l'approche et la saisie d'un objet, car elle demande de contrôler 6 degrés de liberté à l'aide d'une manette commandant seulement 3 axes à la fois. Aussi, le contrôle à l'aide de manettes demande un haut niveau de dextérité, souvent difficile à atteindre par les utilisateurs ciblés.

Pour résoudre ces problématiques, l'objectif de ce projet est de développer une solution de contrôle du bras robotique d'assistance à l'aide d'un système de suivi du regard de l'utilisateur. Pour ce faire, il est nécessaire de suivre les yeux de l'utilisateur et d'en déduire la direction du regard, de reconstruire la scène en face de l'utilisateur par stéréovision, d'y faire la correspondance entre la direction du regard et les objets présents dans la scène pour finalement planifier la trajectoire du bras robotique pour atteindre l'objet et le prendre.

La solution finale détaillée dans cette thèse permet donc de faire l'approche et la prise d'objet automatique en estimant la direction du regard de l'utilisateur et en détectant les objets en face de ce dernier. Ce système n'est aucunement intrusif car il ne nécessite pas que le bénéficiaire porte un objet comme des lunettes. Il fonctionne en présence d'illumination du soleil, ce qui n'est pas le cas d'autres types de système. Il permet aussi de prendre des objets sans contrainte sur leur type et ne nécessite aucune calibration par utilisateur.

Utilisant un modèle original d'estimation de la direction du regard et mettant en place un système de détection et sélection d'objets basé sur la détection de surface plane telle une table et de l'allure de l'objet par apprentissage profond en parallèle, le système permet de planifier une trajectoire automatiquement en évitant les collisions avec la scène.

À la suite d'essais exécutés dans le contexte de cette thèse, il est possible de constater une réduction du temps pour accomplir une tâche de préhension d'un objet d'environ 72.0% pour un taux de succès de 90% , tout en obtenant un score de satisfaction élevé d'en moyenne 4.6 sur 5 selon un questionnaire complété par les participants.

ABSTRACT

In recent years, advances in robotics allow people suffering from upper limb disabilities to equip their motorized wheelchair with an assistive robotic arm. These are generally controlled through a specialized joystick device using filters and calibrated specifically to the user's force and disability. These robotic arms have been proven to improve the autonomy of their users, especially to accomplish tasks of daily living such as eating.

However, some issues still persist, such as that manipulating the robotic arm is still time intensive for the users, especially in the case of approaching and grasping an object, since it requires moving the robotic arm in 6 degrees of freedom using a joystick that typically provides only 3 axes of motion. Also, joystick controls are complicated and require a high degree of dexterity, which is often lacking in potential users. Furthermore, the learning curve for using this technology is steep and the adaptation time is often too long, which leads to low adoption rates for potential users.

To mitigate these issues, this PhD project aims to develop a control system for assistive robotic arms using user point of gaze estimation and gazed-at object detection using computer vision. This implies that the developed system must estimate the gaze direction and origin of the user, reconstruct the scene in 3D in front of the user, detect objects in the scene and match the user's gaze to the best fitting object. The developed system must also be able to generate a valid and collision-free path to the object, leading to a grasping motion, thus being able to automatically approach and grasp an object that is gazed-at by the user.

The system described in this thesis is non-intrusive since it does not require the user to wear any devices, instead relying on mounting points on the user's motorized wheelchair. It is also less sensitive to lighting conditions, working with direct sunlight which is not the case for most of the available gaze control systems in the literature. Furthermore, it allows the grasping of objects independent of their type, as long as it is found to fit inside the robot's gripper. The developed system uses a novel gaze estimation deep learning model and an object detection subsystem that uses both the detection of a flat surface such as a table and a deep learning segmentation model based on object classes in parallel, allowing for robust object detection.

From the data collected through device trials performed in the context of this PhD thesis, the system has demonstrated a 72.0% reduction in the time to grasp varied objects over to the same tasks performed with a joystick, while keeping a high success rate of 90% and a high satisfaction score of 4.6 out of 5.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF SYMBOLS AND ACRONYMS	xiv
LIST OF APPENDICES	xv
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 LITERATURE REVIEW	4
2.1 Assistive robotic manipulators	4
2.2 Human-Computer Interfaces for Assistive Robotics	5
2.3 Gaze Estimation	8
2.3.1 Pupil Center Corneal Reflection Method	8
2.3.2 2D Regression	10
2.3.3 3D Model Family	11
2.3.4 Cross-Ratio Family	12
2.3.5 Appearance-Based Family	13
2.3.6 Shape-Based Family	15
2.3.7 Overview of Families	15
2.4 Object Detection	15
2.4.1 Face Detection, Alignment and Tracking	15
2.4.2 Object Detection and Background Segmentation	20
2.5 3D Reconstruction	21
2.5.1 Theory of Stereoscopic Systems	21

2.5.2	Block Matching, Structured Lights and Texture Projection	23
2.6	Motion Planning	23
2.6.1	Path Planning	24
2.6.2	Trajectory Planning	25
2.6.3	Conclusion on Motion Planning	26
CHAPTER 3	RATIONALE AND GENERAL OVERVIEW	27
3.1	Problematic and Frontier of Knowledge	27
3.2	Objectives	28
3.2.1	General Objectives	28
3.2.2	Specific Objectives	28
3.2.3	Contributions	29
3.2.4	Scope	29
3.3	General Overview and Articles Description	29
3.3.1	First Article	30
3.3.2	Second Article	30
3.3.3	Third Article	31
3.3.4	Technology Choices and Justifications	31
CHAPTER 4	ARTICLE 1: NON-INTRUSIVE REAL TIME EYE TRACKING USING FACIAL ALIGNMENT FOR ASSISTIVE TECHNOLOGIES	33
4.1	Abstract	33
4.2	Introduction	33
4.2.1	Eye Tracking for Assistive Robotic Arms Control	34
4.2.2	Objectives	35
4.2.3	Contributions	35
4.3	Related Work	35
4.4	Methodology	37
4.4.1	Workflow and implementation details	37
4.4.2	Training and validation	44
4.4.3	Performance comparison methodology	45
4.5	Results	45
4.6	Discussion	46
4.6.1	Interpretation	46
4.6.2	Limitations and further work	46
4.7	Conclusion	48

CHAPTER 5	ARTICLE 2: SEMI-SYNTHETIC DATASET AUGMENTATION FOR APPLICATION-SPECIFIC GAZE ESTIMATION	50
5.1	Abstract	50
5.2	Introduction	50
5.3	Methodology	53
5.3.1	Facial Alignment	53
5.3.2	Pose Under Perspective Projection	53
5.3.3	Textured Face Mesh Generation	55
5.3.4	Inverse Pose Transform Correction to Gaze Direction	56
5.3.5	Head Pose Distribution	56
5.3.6	Sample Pose Transform	57
5.3.7	Mesh Projection and Normalization	57
5.3.8	Gaze Estimation Model Training	58
5.4	Results and Discussion	59
5.5	Conclusion	60
CHAPTER 6	ARTICLE 3: UNCONSTRAINED, CALIBRATION-FREE, FREE-VIEWING AND NONINTRUSIVE GAZE ESTIMATION-BASED CONTROL SYSTEM FOR ASSISTIVE ROBOTIC ARMS	61
6.1	Abstract	61
6.2	Introduction	61
6.2.1	Objective	63
6.2.2	Contributions	64
6.3	Related Work	64
6.4	Methodology	65
6.4.1	Gaze Estimation	68
6.4.2	Scene Reconstruction	72
6.4.3	Object Detection	73
6.4.4	Detected Object Selection	77
6.4.5	Path Planning	79
6.4.6	Additional control modes	84
6.4.7	Control Flow	86
6.5	Results	89
6.5.1	Population	89
6.5.2	Tasks	89
6.5.3	Data Collection	90

6.6	Discussion	93
6.6.1	Interpretation	93
6.6.2	Limitations and future work	94
6.7	Conclusion	95
CHAPTER 7 GENERAL DISCUSSION		98
7.1	Improvements and Comparison to Previous Work	98
7.2	Return on the Objectives	99
7.2.1	Specific Objective 1	99
7.2.2	Specific Objective 2	99
7.2.3	Specific Objective 3	100
7.2.4	Specific Objective 4	100
CHAPTER 8 CONCLUSION		101
8.1	Summary of Works	101
8.2	Limitations	102
8.3	Future Research	103
REFERENCES		105
APPENDICES		123

LIST OF TABLES

Table 2.1	Overview and comparison of eye tracking methods families	16
Table 4.1	Accuracy for the MPIIGaze Dataset	47
Table 4.2	Accuracy for the GazeCapture Dataset	47
Table 4.3	Accuracy for the UTMultiview Dataset	47
Table 4.4	Performance Timings (Full Detection and Gaze Estimation)	47
Table 5.1	Comparative results in gaze errors when trained on the original and augmented datasets	59
Table 6.1	Gaze estimation error decrease when using person-specific calibration compared to no calibration	71
Table 6.2	Participant data and distribution	90
Table 6.3	Overall Time to Accomplish a Task	91
Table 6.4	Success Rates (%) Across All Trials	92
Table 6.5	Collision Rates (%) Across All Trials	92
Table 6.6	Responses about the perceived attributes of the gaze control system .	93

LIST OF FIGURES

Figure 1.1	Assistive robotic arm mounted on a motorized wheelchair.	1
Figure 1.2	Example of a joystick for controlling assistive robotic arms	2
Figure 2.1	Example of a chin-actuated joystick.	6
Figure 2.2	Examples of head-free gaze control systems for assistive robotic arms.	7
Figure 2.3	Corneal reflection and bright pupil	9
Figure 2.4	Bright pupil and dark pupil	9
Figure 2.5	Relative position of pupil center and glint	10
Figure 2.6	Schematic of a typical gaze tracking system, where P is the pupil and G is the glint	11
Figure 2.7	An example of a 3D model of a human eye	12
Figure 2.8	Representation of the virtual tangent plane	13
Figure 2.9	Projective relations existing in a cross-ratio setup	13
Figure 2.10	Example of a template matched to an eye image	15
Figure 2.11	The four base features (A, B, C, D) of random size within the detection window	18
Figure 2.12	A workflow diagram of a Viola-Jones detector over a single sub-window	18
Figure 2.13	Example of a HOG trained HOG pattern being used for face detection	19
Figure 2.14	Geometric relations behind a stereoscopic system	22
Figure 4.1	Our workflow is mostly sequential, with the head pose being used to normalize the eye patch images and correct the estimated gaze. . . .	38
Figure 4.2	A simplified overview of our model.	42
Figure 4.3	The Irisblock is a variation on the BlazeFace blocks.	43
Figure 4.4	Our gaze estimation and iris landmarks recognition model.	43
Figure 5.1	Distribution of the gaze direction and head pose pitch and yaw with respect to the camera's center axis for the MPIIFaceGaze dataset [1].	52
Figure 5.2	Sample application of gaze estimation for object selection for automatic grasping in assistive robotics.	52
Figure 5.3	Simplified visual workflow of the augmentation method with real ex- ample images.	54
Figure 5.4	Geometric construction example of the virtual camera and the aug- mented eye position.	58
Figure 5.5	Distribution of the gaze direction and head pose pitch and yaw with respect to the camera's center axis for the UTMultiview dataset [2]. .	60

Figure 6.1	Photo of the complete system setup with annotated parts.	67
Figure 6.2	Approximate depiction of the reference frames of the different sensors and the robot.	68
Figure 6.3	Added variance output dimension to the gaze estimation model. . . .	71
Figure 6.4	Example of object segmentation including the object and its lid as separate detections.	76
Figure 6.5	Example of a concave object's detection bounding box containing another object without any overlap in their masks.	77
Figure 6.6	Reduction of the 3D object selection problem to 2D by projecting onto the horizontal plane.	78
Figure 6.7	Depiction of the offset position and direction of the linear grasping motion.	82
Figure 6.8	Possible grasping solution around the object.	83
Figure 6.9	Depiction of the motion steps of the gaze vector control mode.	85
Figure 6.10	Remote controller used for controlling the robot with the labels for each button.	86
Figure 6.11	Progression of app states and robot motions given the button inputs and state results.	87
Figure 6.12	Average execution time per task and standard deviation for the joystick and gaze control system.	90
Figure 6.13	Average execution time per task for each age group.	91
Figure 6.14	Approximate position of the objects for each task.	96
Figure 6.15	Process diagram of the complete gaze estimation, object detection and path planning workflow.	97

LIST OF SYMBOLS AND ACRONYMS

AAM	Active Appearance Model
ARA	Assistive Robotic Arm
ARM	Assistive Robotic Manipulator
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DoF	Degrees of Freedom
GPU	Graphics Processing Unit
HCI	Human-Computer Interface
HMI	Human-Machine Interface
HOG	Histogram of Oriented Gradients
IK	Inverse Kinematics
NIR	Near Infra-Red
PCCR	Pupil Center Corneal Reflection
PRM	Probabilistic Road Map
RANSAC	Random Sample Consensus
RRT	Rapidly Exploring Random Tree

LIST OF APPENDICES

Appendix A	Research Protocol (French)	123
------------	----------------------------	-----

CHAPTER 1 INTRODUCTION

Assistive technologies are products, systems, software programs or equipment that aim to improve or maintain the autonomy or functional capabilities of people with disabilities. Such technologies are diverse and can range from simple solutions such as a magnetic board and stickers used to communicate to complex exoskeletons used for physical rehabilitation [3].

With the fast-paced advances in modern robotics and mechatronics, assistive technologies have seen an increase in complexity and today it is possible for people with physical limitations, such as upper limb disabilities, to acquire an Assistive Robotic Arm (ARA), also known as Assistive Robotic Manipulator (ARM), typically installed on a motorized wheelchair, as seen in Figure 1.1.



Figure 1.1 Assistive robotic arm mounted on a motorized wheelchair. *Taken from [4].*

Such a system is generally controlled via a joystick-based interface as seen in Figure 1.2, albeit a lot of research work is still ongoing in designing a user-friendly and effective Human-Computer Interface (HCI), also sometimes called Human-Machine Interface (HMI), for controlling such robots. Indeed, the problem of designing an HCI that is easy to use while being fast and affordable is a complex challenge. This PhD thesis aims to design and develop such a system.

Indeed, considering that upper limb disabilities are varied and thus the physical abilities of



Figure 1.2 Example of a joystick for controlling assistive robotic arms

the people suffering from those can range from mild lack of dexterity to complete paralysis, few assumptions for ease of use and effectiveness apply to all potential users. For example, while a voice controlled HCI might be a good option in some cases, not all users have the necessary vocal functions to use these HCIs. Since the vision of people with most upper limb disabilities, severe muscular dystrophies or tetraplegia remains functional [5, 6], a gaze estimation-based control method is thus the basis of the research work of this PhD thesis.

As will be detailed in Chapter 2, several gaze estimation-based control system are proposed as HCIs in the literature [7–10], these are often not adaptable to most scenarios of daily living, such as when direct sunlight illumination occurs either outdoor or close to a window. Most of them are also costly, using gaze estimation systems priced in the thousands of dollars and some of them require either the user to wear bulky glasses or go through long person-specific calibration procedures. Also, the gaze estimation-based control systems described in the literature are limited to specific, known objects and do not adapt well to new objects and scenarios, which limits their application for real world use.

Therefore, the work presented in this PhD thesis builds upon the proof-of-concept presented in [8], which relied on a low-cost commercial gaze estimation device that was ill-suited to 3D gaze estimation since it was designed for screen-based gaze estimation. It was also limited in application to indoor and with almost no sunlight illumination scenarios due to its use of the infrared light spectrum. It required extensive calibration and the user could not move their head without requiring recalibration. It could also not segment objects to grasp, which limited its application to just a few select objects. The underlying goal is thus to solve the remaining issues with gaze estimation-based HCIs for ARAs as to create a prototype as close as possible to real world use.

Chapter 2 will review the literature on the state of the art in gaze estimation-based HMIs for ARAs and will present the current limitations of such systems. It will also review the background knowledge required for this project. Chapter 3 will synthesize the problematic and explain the rationale of this thesis. Chapter 4 contains the first paper describing the development, training and validation of the gaze estimation model. Chapter 5 contains the second paper describing the data augmentation technique used for training the gaze estimation model to improve the accuracy of the model in application-specific scenarios. Chapter 6 contains the third paper describing the development of the HMI and the experiments conducted to validate the system. Chapter 7 discusses the developed system by comparing it to the previous proof of concept and also returns on the objectives of the project. Finally, Chapter 8 will present the conclusions of this PhD thesis and the future work that can be done to improve upon the results presented herein.

CHAPTER 2 LITERATURE REVIEW

This literature review is divided into 6 parts. First, a review and description of currently available ARAs (also known as ARMs) will be presented. Then, a review of the HCIs used to control ARAs will be presented. After that, the different gaze estimation methods will be reviewed. Then, methods of object detection, both for general objects and faces, will be detailed. Following, the 3D reconstruction techniques will be reviewed. Finally, the methods used for motion planning of serial manipulators will be presented.

2.1 Assistive robotic manipulators

Assistive robotic manipulators are generally a smaller and lighter version of robotic arms typically used in industrial or scientific applications but made to operate safely in the presence of humans. Their goal is to leverage such a technology to enable greater autonomy for users with musculoskeletal disorders and upper-limb disabilities by accomplishing actions that the user cannot do alone. Such actions are typically grasping and moving objects, drinking, eating, opening doors, flipping switches, etc.

Several assistive robotic manipulators exist such as Manus/iARM series by Exact Dynamics [11], the WAM arm by Barrett Technology [12], the DLR LWR-III arm by DLR Robotics and Mechatronics Center [13] and the JACO/MICO series of ARAs by Kinova Robotics [14]. Most of the available robots are thus serial robots, with a varying number of Degrees of Freedoms (DoFs) typically ranging from 5 to 7 without counting the DoF related to the opening or closing of the end effector's "fingers".

Such robots have been known to be beneficial to people with upper-limb disabilities [15–19]. It has been reported that ARAs lead to a tangible improvement in various tasks and activities in a home and community setting, while it also improved the self-perceived quality of life of the users too [19]. Additionally, ARAs improve autonomy and reduce the cost of caregiving [18]. In 2013 in the United States alone, an estimated 150,000 people with disabilities could benefit from using an ARA, including people with muscular dystrophy, spinal cord injury, spinal muscular atrophy, amyotrophic lateral sclerosis, multiple sclerosis, cerebral palsy, rheumatoid arthritis, post-polio syndrome, locked-in syndrome and other severe motor paralysis [16].

Although the ARAs have many benefits, several shortcomings are still highlighted in studies regarding their usage. Such issues are generally related to dexterity and how tedious it is to learn how to control a 5 DoF+ robotic arm, especially for a user with upper-limb disabilities.

In [16], it is said that a vision-based interface with autonomous path planning would lead to a tremendous improvement in user loading in grasping and pick-and-place tasks. Furthermore, these tasks are said to be high priority due to their high interaction requirements. [17]

2.2 Human-Computer Interfaces for Assistive Robotics

As discussed previously, HCIs are an essential part of the use of ARAs. Special considerations have to be made when designing HCIs for ARAs, as typically users can be:

- Unable to make precise movements
- Have involuntary twitches and spastic movements
- Unable to move some/all upper extremities

These peculiarities are variable and can change from one user to another due to the broad spectrum of disabilities addressed by ARAs, which adds another layer of complexity.

The most common HCI solution is the joystick-based one, which is generally what commercial ARAs are equipped with. Typically, a 3 axes (front and back, side-to-side and twist) joystick is used to control the robot in Cartesian coordinates by splitting the position (3D) and orientation (3D) into two separate modes that use all 3 joystick axis. [15] As such, there is some level of intelligence built into this type of systems, as the transformation between joint-space velocities (i.e. in the robot's own joint-angle coordinate system) and Cartesian-space velocities has to be computed using the Jacobian matrices of the corresponding robot. Most of these systems also include pre-set trajectories or movements, such as "return to home".

Due to the aforementioned considerations, such a HCI still has great room for improvement, as controlling a joystick while lacking dexterity and having involuntary twitches can be a very tedious task. Indeed, joystick-based solutions require a significant amount of time to learn and to accomplish simple tasks even for healthy users [20]. Filters have been implemented to reduce the involuntary robot movements induced by jerks from the user [15], but such systems are far from ideal as described in [16,17]. This issue is even more exacerbated by the fact that potential users might have specific disabilities which make the use of a joystick via their hands impossible. For example, Ristic-Durrant [21] developed a chin-actuated joystick, which without further assistance from an intelligent system, would make usage of ARAs even more difficult. Another example of a chin-actuated joystick can be seen in Figure 2.1. As pointed by Argall [22], the limits of the control interfaces currently available for commercial

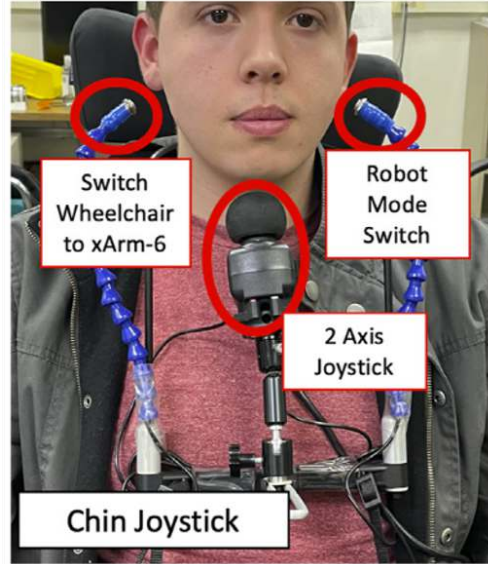


Figure 2.1 Example of a chin-actuated joystick. *Taken from [23].*

ARAs creates a situation where the people that could benefit the most of ARAs are simply unable to control them due to their severe motor disability.

As such, recent works have suggested several ways of controlling ARAs. The ARoMA system [24] uses a fixed depth camera (using a pseudo-structured light camera called Senz3D [25] based on Intel Realsense technology [26]) to feed 3D scene information to an ARA, for which the Inverse Kinematics (IK) is computed using OpenRAVE's IKFast [27] to make the ARA grasp an object. Another system by Bousquet-Jetté et al [28] uses a depth camera, scene segmentation and an object recognition algorithm to grasp a known set of objects.

While most HCI systems for ARAs use 2D or 3D cameras to extract information from the scene presented in front of the user and plan an action given that information, the action itself is generally taken via another sensor or interface. For example, the ARoMA-v2 system [29] builds on top of the original system and adds voice recognition via the Microsoft Speech API to control both low-level movements (left, right, up, down, forward, backward and rotations) and higher-level ones (push it, tap it, close hand and open hand). Likewise, [30] uses speech recognition, but adds another 3D camera, a Kinect by Microsoft [31], for gesture recognition, which is then used to accomplish different pre-set actions. The work presented in [32] uses a similar approach but adding face detection and tracking to position objects after being grasped (such as eating/drinking). The works presented in [33–37] use a touchscreen or click interface to select objects and actions assisted by computer vision, using the scene-facing 2D or 3D camera.

One recurring method for interfacing and selecting objects is using eye tracking techniques,

due to the fact that users naturally gaze at objects and actions they are trying to accomplish and in most cases, users even with tetraplegia retain complete control over their eyes [38–45]. Such techniques usually revolve around two ways of acting on the robot: estimating the gaze direction and head position of the user to then estimate the gaze point in 3D space which is then used to select an object to grasp in a scene or move the robotic arm [38, 39, 41, 42, 44], recognizing specific eye movements (e.g. up and down) as actions or compute the gaze point on a 2D screen to accomplish a list of actions [40, 43, 45–49]. To the best of our knowledge, no nonintrusive eye tracking-based techniques exist for dictating a robot’s grasping point and accomplishing the grasping motion both indoors and outdoors.

Indeed, gaze control methods that rely on a screen [40, 43, 45–49] forces the user to look away from the scene and therefore can be arduous for some of them [10]. To this effect, Pr. Sofiane Achiche’s laboratory explored the eye tracking-based ARA control methods [7, 50], which form the basis and preliminary work of this PhD thesis. The proof of concept system described in [8] proves the usefulness of using eye tracking for grasping actions due to the lower time and effort required, even with obstacles present in the scene. However, this system is not robust to head movements and outdoor illumination due to the use of the IR spectrum. Furthermore, it uses a hard-coded approach for grasping objects and does not detect objects per se, but rather aims at grasping a single point which is part of the object which makes the system less flexible to different object sizes and shapes. Finally, the calibration procedure requires a long time and includes multiple manipulations. It can also be invalidated by the user simply moving their head.

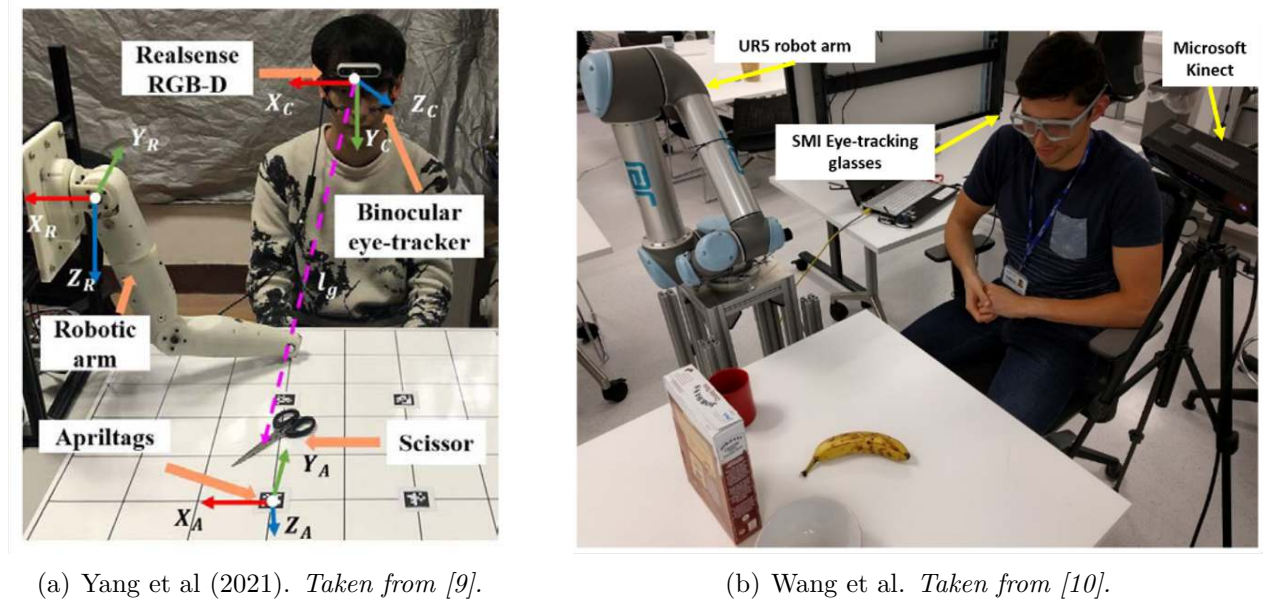


Figure 2.2 Examples of head-free gaze control systems for assistive robotic arms.

2.3 Gaze Estimation

Eye tracking is a task which generally aims at obtaining the user's eye(s) position and orientation. Gaze estimation, also called gaze tracking, generally estimates the gaze point of the user, i.e., what the user is looking at. As in this thesis we are mostly interested in computing the gaze position (fixations) and not the other type of eye movements (saccades, pursuits, scan path, etc.), the focus of this literature review is on the different methods of gaze estimation.

Literature review papers presented in [51–54] describe the most common and advanced eye tracking methods and devices available. This research proposal will not dive into the intrusive eye tracking systems, such as electrooculography and electromagnetic contact lenses [54], as these are not adapted for day-to-day use. We will neither dive into expensive and complex setups, such as dual Purkinje images [53] eye trackers, as they require specialized lenses and servo-controlled mirrors [55].

As such, most eye tracking techniques are based on video images of the eyes and can be classified into five different families 2D regression, 3D model, cross-ratio, appearance-based and shaped-based [52]. Before detailing these categories, it is important to first describe the underlying detection and tracking technology shared by many research works: the Pupil Center Corneal Reflection (PCCR) method.

2.3.1 Pupil Center Corneal Reflection Method

This method of tracking is based on using an Infra Red (IR) LED and an IR camera to produce an easily trackable glint (corneal reflection) on the cornea of the user's eyes, such as shown in Figure 2.3, because most of the indoor ambient light is not in the Near Infra-Red (NIR) spectrum and because IR light is nonintrusive to the user. The input of the gaze tracking algorithm will therefore be the relative position between the glint and the pupil center. This difference, for a fixed head position, is directly related to the gaze angle of the user.

It is important to note that illuminating the eyes can cause a phenomenon known as bright/dark pupil, as light can be reflected by the retina. For this reason, if the illuminator is on the same optical path as the camera or recording device, the pupil will appear brighter than the iris. If it is not the case, the pupil will appear darker than the iris. This is the same phenomenon as the one that causes "red eyes" in photos and is illustrated in Figure 2.4. A bright pupil is easier to track as it is a very bright and circular spot, which is easy to detect using basic thresholding techniques or even well-known circular detectors such as Hough

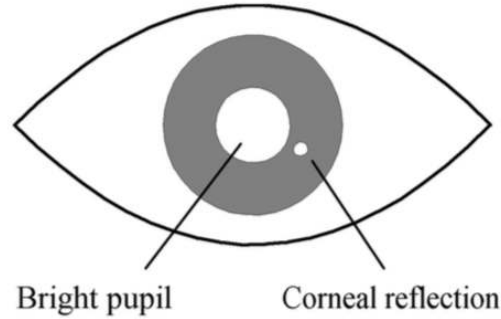


Figure 2.3 Corneal reflection and bright pupil. *Taken from [54].*

Transform-based ones [56], but generally does not function outdoors due to the much more present ambient NIR light.

A dark pupil is harder to track (and distinguish from e.g. eyelashes), but necessitates a less constrained setup, the camera and illuminator do not have to be on the same optical line. Although it is easier to work with a dark pupil setup outdoors, it is still not ideal as the corneal glint from the illuminator might not be the only glint on the eye, as the sun, street lamps and other sources of unaccounted for illuminations will produce NIR reflections.

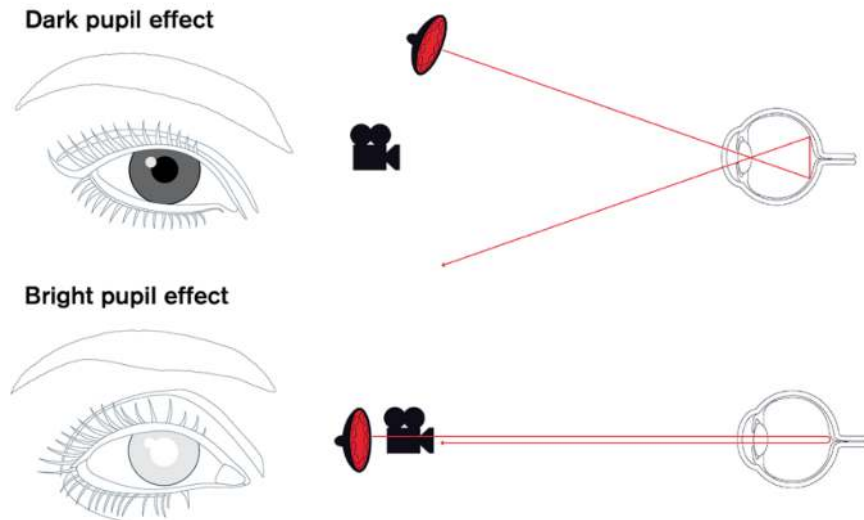


Figure 2.4 Bright pupil and dark pupil. *Taken from [57].*

Some methods, such as the ones presented in [58] and [56] take advantage of both bright pupil and dark pupil by using a pair of images one after the other; the first image with a simple NIR illuminator off the optical axis and the second image with an added NIR illuminator on the optical axis switching on for the duration of the picture. This simplifies the pupil detection problem to a simple binary image subtraction, as the first image will lead to a

"black" blob at the location of the pupil, while the second image will generate a white blob. It is assumed that the bright pupil is the only difference between the images.

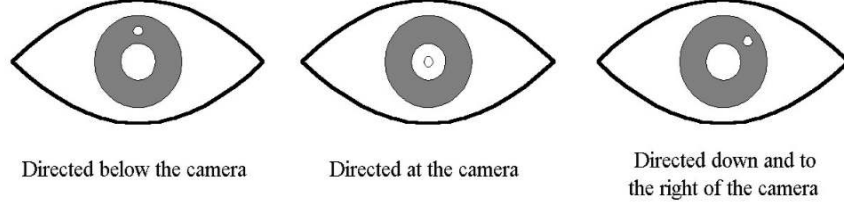


Figure 2.5 Relative position of pupil center and glint. *Taken from [54].*

2.3.2 2D Regression

As discussed previously, from the relative position (expressed as a vector) of the glint (G) and pupil center (P), it is possible to compute a value that corresponds to a gaze angle if the user's head position is fixed in place, as the only variable in the remaining system is the user gaze direction (the NIR LEDs, the camera and the head are all fixed in space). Hence, there exists a function that maps the relative position of P and G to a screen target location (see Figure 2.6):

$$f : (x_G - x_P, y_G - y_P) \rightarrow (x_s, y_s) \quad (2.1)$$

where (x_G, y_G) is the glint position, (x_P, y_P) is the pupil position and (x_s, y_s) is the position of the gaze point on the screen (e.g. in pixels).

This function f is generally obtained through a polynomial regression between calibration points where the user is asked to gaze at certain specific screen coordinates and the corresponding distance between the glint and pupil is recorded. This regressed relation is then used to estimate the user's gaze for other points on the screen after the calibration. An example of a commonly used regression polynomial in state-of-the-art methods [56] is:

$$x_s = a_0 + a_1(x_G - x_P) + a_2(y_G - y_P) + a_3(x_G - x_P)(y_G - y_P) + a_4(x_G - x_P)^2 + a_5(y_G - y_P)^2 \quad (2.2)$$

$$y_s = b_0 + b_1(x_G - x_P) + b_2(y_G - y_P) + b_3(x_G - x_P)(y_G - y_P) + b_4(x_G - x_P)^2 + b_5(y_G - y_P)^2 \quad (2.3)$$

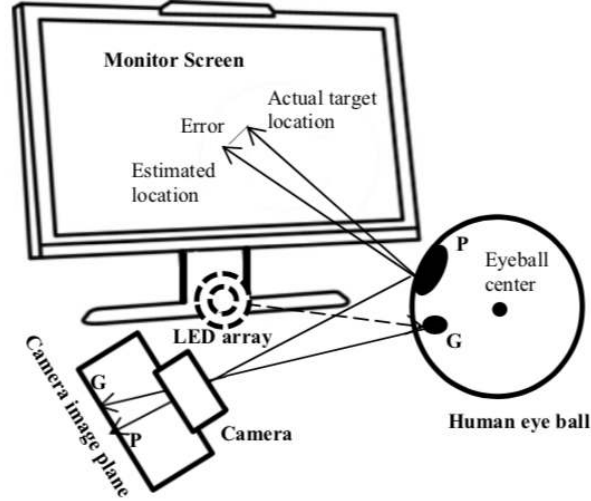


Figure 2.6 Schematic of a typical gaze tracking system, where P is the pupil and G is the glint. Taken from [52].

Parameters a_i and b_i are the regression parameters.

Several mapping functions and calibration procedures have been studied [59]. Such a process is generally not robust to head motion when using regression, but several alternatives to this basic regression method have been proposed [60,61] to alleviate the errors induced by head movements, generally requiring a second NIR LED. This is due to the fact that regression generally makes use of only the difference in position between the center of the pupil and the glint, which does not directly give head pose information. Work presented in [58] claims that adaptive exposure control combined with variable intensity LEDs can lead to suitability for outdoor use, but without validating the hypothesis.

2.3.3 3D Model Family

By opposition to the 2D regression method, the 3D model technique uses a 3D mathematical model of the eyes to obtain the mapping function from the glint-pupil vector to the screen coordinates. Such a model includes several parameters, some of which arise from the dimensions of the user's eyes, others from the geometrical arrangement of the setup, e.g. the location of the camera(s), illuminator(s) and the screen.

Most systems using 3D models still use the PCCR as a way of tracking the eye movements [62–64], while using either several illuminators or multiple cameras to achieve robustness against head movements, even with a single point calibration [65]. 3D measurements based methods using new depth cameras such as Kinects [31] generally simplify the head motion compensation due to the availability of direct depth measurement, and thus position in 3D

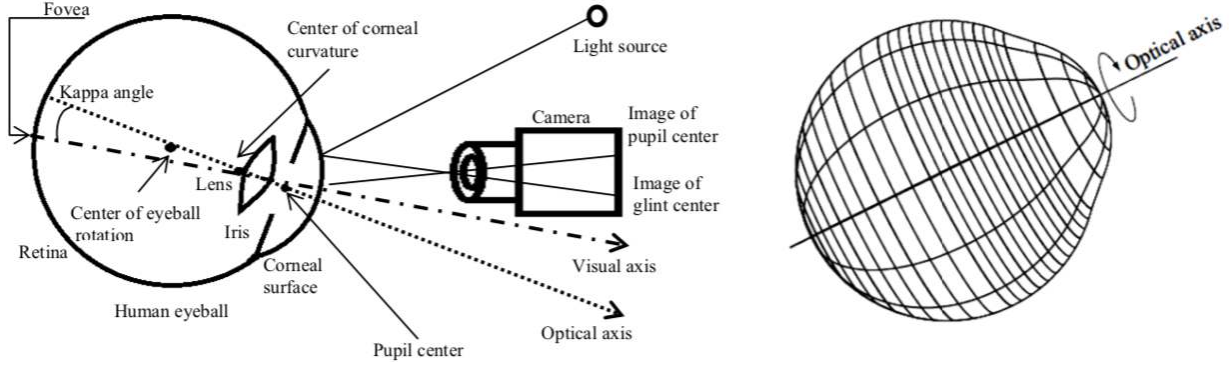


Figure 2.7 An example of a 3D model of a human eye. The Kappa angle is the deviation between the visual axis and the optical axis, which varies from person to person. *Taken from [52].*

space. A method proposed in [66] relies on geometric constraints, e.g. all iris center points are distributed on a sphere originated from the eyeball center, to estimate the center of the eyeball and its radius and thus compute the optical axis. According to the authors, it is then possible to obtain an accuracy of less than 4 degrees without relying on glint tracking.

2.3.4 Cross-Ratio Family

Cross-ratio based methods use properties of geometric projections to deduce the position of the gaze point on a screen. This requires a regular generally rectangular pattern of NIR LEDs for which there exists a projective relation to from the camera image plane to a virtual tangent plane obtained from the reflections of the LEDs on the eye [67]. The relation from the reflection on the eye to the virtual plane is challenging to compute for general cases, but by limiting the range of head movements (user in front of the screen, 30-50 cm away and moving less than 30 cm from the center), it is possible to approximate this relation, as explained in [67]. See Figure 2.8 and 2.9 for visual depictions of the cross-ratio based methods.

Once the projection properties are known, the invariant value of the resulting projective space can be used to compute the inverse projection for another set of points, which results in the position of the gaze point.

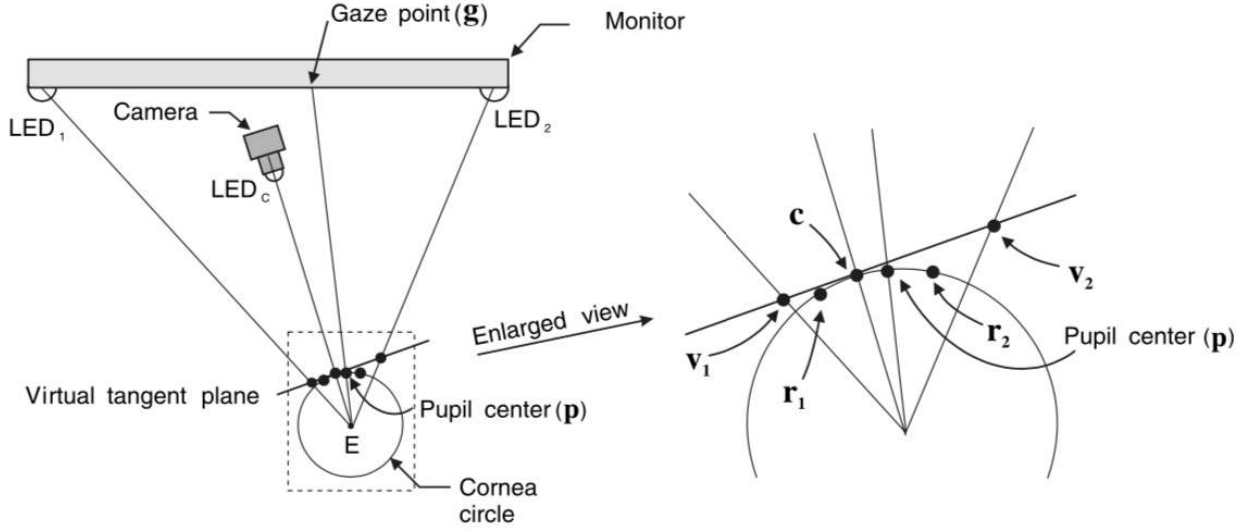


Figure 2.8 Representation of the virtual tangent plane. The points r_1 , r_2 and c are the glints, while v_1 and v_2 are the virtual projection points. Taken from [67].

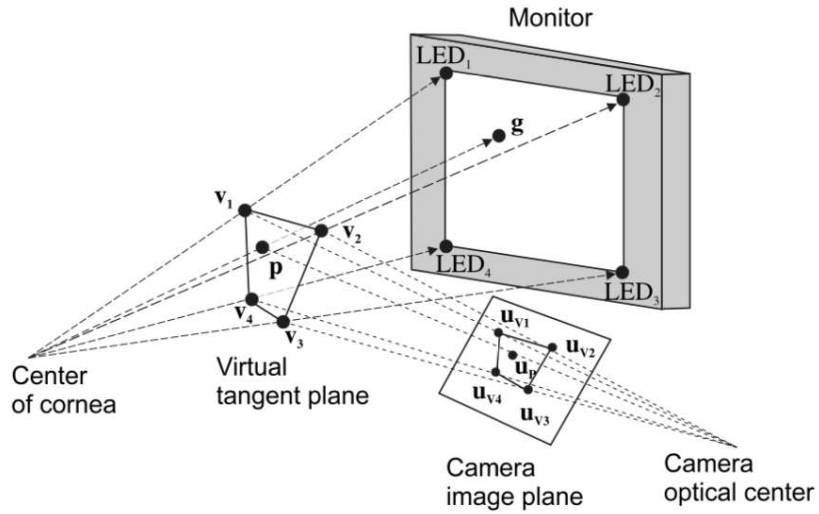


Figure 2.9 Projective relations existing in a cross-ratio setup. The v_i points are the reflection of the LEDs on the computed virtual plane, while u_{v_i} are the corresponding projected points on the camera image plane. The point p is the pupil center and g is the gaze point. Taken from [67].

2.3.5 Appearance-Based Family

Appearance-based techniques use a classifier/regressor trained from extracted features of the image of the eye to compute the gaze direction of the user. Generally, the workflow is as follows:

1. Locate the user's eyes in the image.

2. Extract relevant image features based on appearance.
3. Classify/regress these features into a gaze angle or point on a screen.

Such methods [68] rely on a statistical model of the eye image, both using shape and texture features, trained on a set of images annotated with landmarks and selects the most relevant combination of features using a Principal Component Analysis (PCA) for both shape and texture images [52]. Then, an Active Appearance Model algorithm is used to fit the model onto a new eye image [69] (step 1). A mix of Gaussian Mixture Models for large eye movements and Histogram of Oriented Gradients models for smaller movements are trained to predict the gaze angles or gaze point on a screen for a set of training images (step 2). Lastly, the trained models are then used on live images to obtain the gaze points (step 3).

Several other methods exist, including Local Binary Pattern Histogram coupled with several classification models such as support vector machine (SVM), neural network and k-Nearest Neighbors, Haar features and cascades [52]. Lately, with the advent of deep learning and Convolutional Neural Network (CNN), several models have been trained to accomplish either all the steps of the workflow, or the last two steps, relying on state-of-the-art techniques for face tracking. Such methods have been implemented in real-time (10-15 frames per second) on a mobile device, although only outputting a gaze point on a phone or tablet screen at a somewhat low accuracy (about 2 cm on the screen) [70]. Other CNN based methods have good accuracy, an improvement of 22% over previous state-of-the-art, from training on the UT Multiview dataset with 64,000 eye images, but without any information on execution speed in real life settings [71]. More recently, CNN appearance-based methods have significantly improved the accuracy while claiming real-time performance on modern computers [72, 73], but without any performance figures or indications in required computing power.

Another notable method uses the webcam on a laptop computer combined with reinforcement learning obtained via user interaction with their web browser [74]. It is thus using multiple inputs, both from the pupil tracker and the user interactions, on which a linear regressor is trained in real-time. The described method uses different types of face detection and alignment algorithms to obtain the eyes location (step 1) but suggests a fast pupil detection algorithm based on finding the region in the eye image with the highest contrast to its surrounding area using an integral image approach. As tested in this PhD project, the accuracy of this method lowers when bright reflections are close to the pupil or on the pupil due to the assumption of a clear contrast between the pupil and the sclera.

2.3.6 Shape-Based Family

As described in [52], shape-based models use a deformable templates of the eye region that are then adjusted to an eye image by using normalized cross-correlation for example, as can be seen on Figure 2.10. Those are generally not robust to varying eye shapes and do not compensate for some head movements [52].

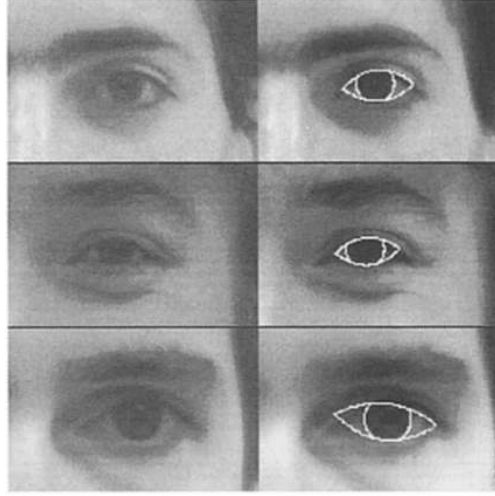


Figure 2.10 Example of a template matched to an eye image. *Adapted from [52, 75].*

2.3.7 Overview of Families

Table 2.1 shows a general overview of the strengths and weaknesses of each method family and whether or not it requires or generally makes use of PCCR. It should be noted that accuracy is not discussed thoroughly due to the low accuracy required by the proposed system, which will be discussed in Section 3.

2.4 Object Detection

In the context of this thesis, this literature review will focus on two general problems linked to object detection and tracking; namely the methods and algorithms for face detection, tracking and alignment and the general techniques for object detection and foreground/background segmentation.

2.4.1 Face Detection, Alignment and Tracking

Face detection aims at identifying the position of faces in images, whereas face alignment is about finding landmarks in a face image, e.g., eyes, mouth, nose, jawline, etc. It is important

Table 2.1 Overview and comparison of eye tracking methods families

Family	Strengths	Weaknesses	PCCR
2D Regression	Simple mathematically and low hardware requirements (a NIR camera and LEDs).	Very sensitive to head movements.	Yes, required.
3D Model	Generally robust or tolerant to head movements, generally inexpensive computationally	Requires a more extensive hardware setup (multiple cameras and/or multiple LEDs, depth camera, etc.)	Yes, in most cases.
Cross-ratio	No need for an eye model or hardware calibration, tolerant to some head motion.	Increased error with distance, limited head motion and requires a fixed LED setup in front of the user.	Yes, required.
Appearance-based	Low hardware requirements; only a simple low-resolution camera is needed.	Sensitive to light changes, generally more expensive computationally, generally requires a large dataset for robustness, generally less precise than other methods.	No.
Shape-based	Low hardware requirements, good accuracy.	Sensitive to head pose, occlusions and variable eye shapes, expensive computationally.	No.

to note that face detection is not face recognition, as it does not necessarily produce biometric features which can then be compared between faces for recognition purposes. Face alignment can be done in 2D [76], where the depth, or distance from the camera, of such landmarks are not considered, or 3D, where the actual 3D position of the landmarks is computed, thus giving a 3D surface corresponding to the face [77].

Face Detection

After a review of the most widely used face detection methods, we can conclude that face detection is generally carried out using three general steps:

1. Region proposal or sub-window sampling.
2. Feature extraction.
3. Classification using extracted features.

Since detection needs to scan what is effectively the whole image, this process can be quite computationally expensive. It should be noted that face detection generally makes use of machine learning, since the classification techniques used are generally obtained via training on an annotated dataset. This is because facial landmarks are not trivially described, and require a large set of features to produce accurate results, as discussed in [78].

As an extension of face detection, real-time face detection is a problem that has first been solved by Viola and Jones [78] in 2001 using a boosted cascade of classifiers made from simple feature filters. These simple features are Haar-like and consist of an ensemble of 4 base features varying in size. Such features can be computed extremely quickly by using an integral image representation.

These features are then built into a degenerate decision tree, also called cascade, where branches lead to either sample rejection or to another decision node. One of the largest contributions from Viola and Jones [78] work was a framework of selecting these features in increasing order of "difficulty" (i.e. the detection rate and number of false positives decrease), where the first features have high accuracy (most often 100%) but high false positive rates (up to 40%), and limiting the full cascade to a comparatively small number of the most important features. It uses an adapted version of AdaBoost [79], which limits the weak learners to a single feature, therefore acting as a feature selection process (see Figures 2.11 and 2.12).

It was the first time a real-time face detector could produce accurate detection (90+%) but at a cost of a high false positive rate. Other contenders have been proposed since, as the one using Histogram of Oriented Gradients (HOG) features and a classifier, generally a linear SVM [80]. Work on deformable parts models with parts based on local HOG features has been presented with good results for pedestrians [81] and face detection. Open-source libraries are available with implementations of these algorithms, such as OpenCV implementing the Viola-Jones framework [82] or Dlib [83] implementing a HOG feature extractors as described in [81] and a structural SVM described in [84]. Figure 2.13 illustrates the features expressed by HOG.

Most of the ongoing research work in face detection focuses on using neural networks [86] with a sliding window or using region proposal. Some of the best work that has been shown in the field is DeepFace from Facebook AI Research [87]. While these algorithms have

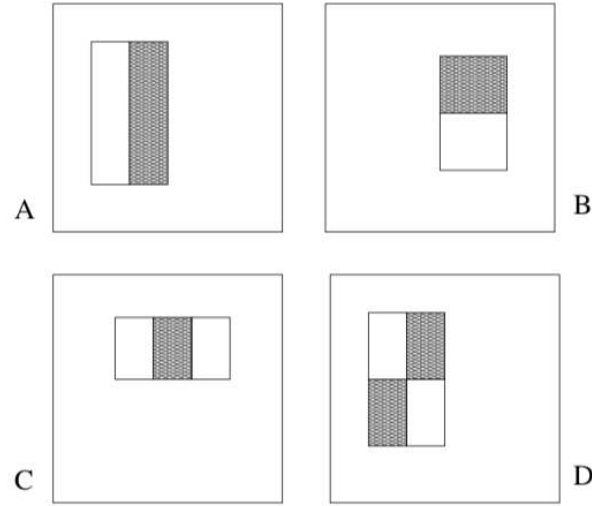


Figure 2.11 The four base features (A, B, C, D) of random size within the detection window. The white regions are pixels added together, while the black regions are subtracted. *Taken from [78].*

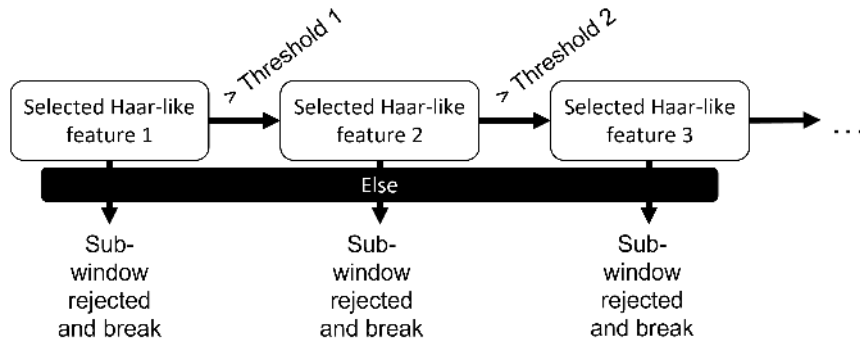


Figure 2.12 A workflow diagram of a Viola-Jones detector over a single sub-window.

great accuracy (up to 97.35% in the case of DeepFace), they are generally computationally expensive [88]. However, recent work on small and fast neural networks have shown promising results, producing comparable performance while improving accuracy [89]. These new CNNs use an anchors-based approach instead of a sliding windows or region proposal, with the network predicting object class probabilities for a set of anchor boxes tiled across the input image, with a refinement step at the end to find adjusted bounding boxes from the fixed anchors. This allows for a single shot multiple face detection over multiple scales, reducing the computational complexity. Even more recently, these advances in CNN have led to realtime face detection even on mobile Graphics Processing Unit (GPU) by using depthwise convolutions [90].

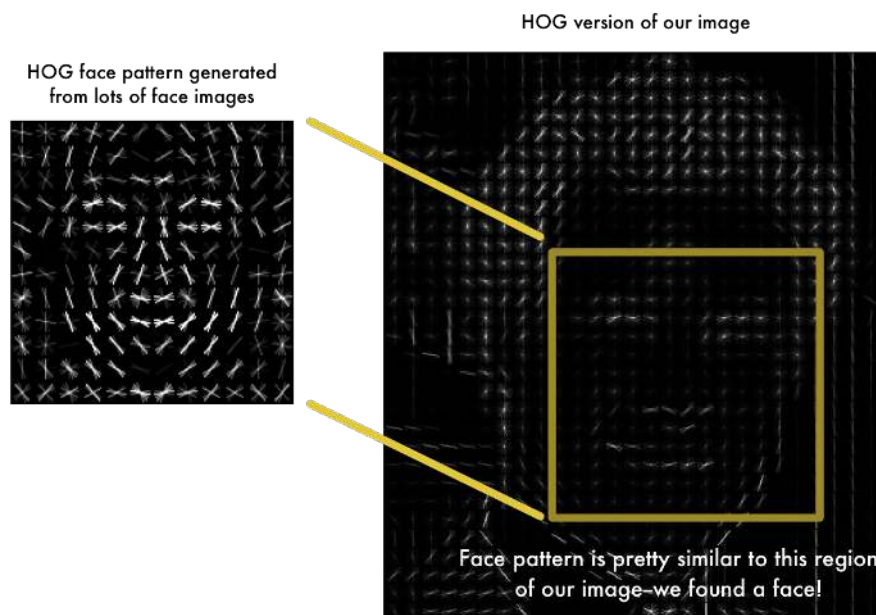


Figure 2.13 Example of a HOG trained HOG pattern being used for face detection. *Taken from [85].*

Face Alignment

The goal of face alignment is to, given an input image that we know is a face, find the location of specific landmarks which are shared across all faces, e.g. a mouth, a nose, a jawline, eyes, eyebrows, etc. In 2D, those landmarks are assumed to be on the same plane, whereas in 3D, we are interested in the 3D position of these landmarks, where depth is important. As face alignment typically operates on smaller images, as a face has already been detected in the original image, for which the scale of the shape to align does not vary much because the input image is assumed to be filled with a face. The smaller images generally result in faster computation speed over face detection [76].

Some of the most used face alignment algorithms include a method proposed by [76] based on a cascade of regressors learned via gradient boosting [91], a variant on boosting algorithms for which each regressor in the cascade "corrects" the last by minimizing the error via a cost function (in this specific case, the squared error). This alignment algorithm starts by placing an initial "mean shape" learnt during training, and then sampling the image at points relative to the "mean shape" decided during training by using a similarity transform to approximate a perspective transformation. Using a learned regression cascade, it then computes an update vector, a vector of values to "bend" or move the landmarks of the previous shape to better fit the actual image. This is done several times until convergence.

Another popular method for face alignment is described in [92], where a similar approach

is used, but using local binary features and a two-stage regression method to compute the shape update vector. First, we compute the local binary features for a set of points around each landmark, each of which are then fed onto separate regression trees. The output of those trees is then concatenated, and a linear regression model is then globally applied to compute a global shape adjustment vector. The author claims 3% better accuracy over previous state-of-the-art accuracy (2014) while 2 to 10 times faster, in the thousands of frames per second, due to the sparse nature of binary features. The update vector can therefore be computed efficiently using lookup tables and vector additions instead of a global matrix multiplication.

More recently and leveraging improvements in computational performance of CNNs designed for mobile devices, 3D face alignment networks based on depthwise convolutions like in [90] have been released [93], with pretrained weights made available by Google’s Mediapipe [94]. These achieve an even higher accuracy than reported in [92], while retaining mobile real-time performance.

2.4.2 Object Detection and Background Segmentation

In the context of this PhD project, the goal of object detection is to segment the graspable objects in the scene, so as to be able to compute dimensional values (such as height, width, depth and geometric center) for accurate grasping. Other types of information could be useful though, as knowing whether the object is a liquid container (e.g., a bottle) or a doorknob for example could be used to accomplish specific actions (opening a door, drinking) or to obtain meta-dimensional values (such as generic shapes) for 3D reconstruction.

Again, as object detection is an extremely broad subject, the most popular and easily accessible real-time object detection framework will be discussed. It should be noted that general object detection is a problem where deep learning and CNN based methods are the state-of-the-art, with several models available through the most popular deep learning frameworks. TensorFlow [95] proposes an object detection API with several models [96], some especially made for mobile platforms. Other examples of fast CNN based object detectors are the YOLO framework [97] or the Faster R-CNN framework [98].

Although the fastest object detections models output simple bounding boxes, the detected object’s real boundaries are most often not a box, but rather much more complex. As such, other object detection models also output a segmentation mask on the color image, as is the case for the popular Mask R-CNN [99] model. Having a precise mask of the detected object can then be used in the 3D reconstruction of the object when applied to the depth image.

To train object detection models, several public datasets are available [100–104], although

the most relevant are those which contains objects of daily living activities, such as the very popular COCO dataset [101]. Of particular interest is the LVIS dataset [104], which not only contains mostly objects of daily living activities because it is based on the COCO dataset, but also includes extensive added annotations and more than 1000 object categories.

Finally, when a point cloud is available, it is possible to segment objects based on the table (a plane) on which they sit. Such a segmentation technique was used with success in [28], using a virtual probing approach. Other methods that do not rely on the assumption that a table must be aligned with an axis use a Random Sample Consensus (RANSAC) approach to plane segmentation [105] instead.

2.5 3D Reconstruction

In the context of this PhD project, 3D reconstruction is used to estimate the 3D position of objects of interest, such as the face and eyes of the user and most importantly the scene facing the user. By having a 3D map of the scene, it is then possible to compute the correspondence between the users' gaze direction and a point or object in the scene facing them. The map is also used to compute a path to the object of interest for the robot to grasp it.

The principle behind accurate 3D reconstruction based on 2D images is to estimate a depth using either: a prior knowledge of the objects being looked at (e.g. their sizes), a scanning depth sensor such as laser time-of-flight or through the use of multiple points of view [106]. For this project, only the latter is of interest, as we assume no prior knowledge of the objects and scanning depth sensors are usually expensive. As such, there are two ways of reconstructing a 3D scene from multiple points of view, either via a stereo or multiple cameras setup or using motion, commonly known as simultaneous localization and mapping (SLAM). Since the latter requires the camera to move, it is thus restrictive. The focus of this literature review is therefore on stereo cameras and other multi-camera setups.

2.5.1 Theory of Stereoscopic Systems

The first assumption behind a stereoscopic system is that the cameras are a perfect pinhole model [106]. Although this assumption seems restrictive, it is generally a good approximation of calibrated cameras (either via pre-calibrated cameras or by going through a calibration procedure). Another assumption behind the basic theory is that both cameras are coplanar, i.e., they are both situated on an axis, both pointing in the same direction, with a distance between them commonly called "baseline". As shown in Figure 2.14, for a point object P at coordinates (x_P, y_P, z_P) , the projections on the cameras' image plane (both with focal length

f) of the object, named P_1 and P_2 , are at image coordinates (U_1, V_1) and (U_2, V_2) respectively. The relation between those two projections is thus:

$$V_1 = V_2 = f \frac{y_p}{z_p} \quad (2.4a)$$

$$U_1 = f \frac{x_p}{z_p} \quad (2.4b)$$

$$U_2 = f \frac{x_p - b}{z_p} \quad (2.4c)$$

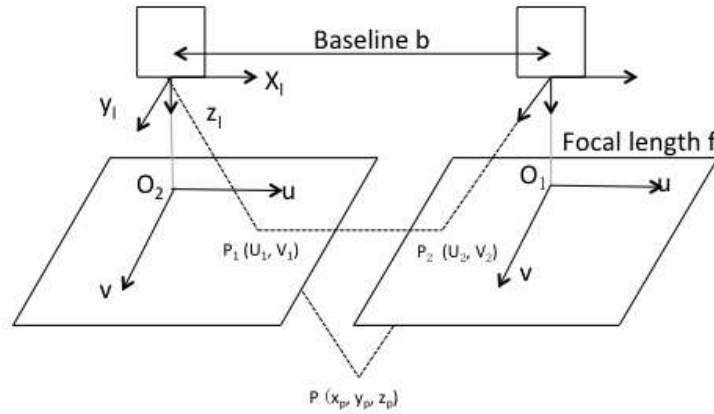


Figure 2.14 Relations behind a stereoscopic system. *Taken from [107].*

From these relations, the distance between P_1 and P_2 is purely on the u axis and is given the name "disparity" or d :

$$d = U_1 - U_2 = f \frac{b}{z_p} \quad (2.5)$$

It is thus possible to obtain the depth z_p by computing the disparity value. This then allows for the inverse matrix of the projection to be computed. Obtaining the disparity map of a stereo image is therefore the main objective when using a stereo camera setup [106].

2.5.2 Block Matching, Structured Lights and Texture Projection

The main challenge with obtaining the disparity is finding which point P_2 on the second image corresponds to the point P_1 on the first image. A commonly used solution is via block matching, in its basic form or in its more accurate but slower semi-global form [108] (both implemented in OpenCV).

Since those algorithms are matching blocks between the two images, issues arise when the images contain regions of low texture, as it is much harder to distinguish which pixels or blocks match between the two images. In general, those regions are either smoothed by matching their disparities to those around, or simply left with a null value [106]. This leads to reconstruction artifacts which can prevent from accurately reconstructing the 3D scene. Examples of regions of low textures include flat surfaces like tabletops, walls, rounded objects with little to no texture, etc. 3D reconstruction of those surfaces therefore generally fails, creating either holes or wrongly shaped objects.

Therefore, solutions using structured light have been proposed, with one widely used system being the Kinect by Microsoft [31] due to its low cost and availability. These systems solve the texturing problem by projecting a known pattern of light, generally in the NIR spectrum, on the scene while capturing the reflected light via a camera. As the projection pattern is known, the projector essentially works as an "inverse" camera in a stereoscopic system. The main issue with such systems is that they require specific lighting conditions, i.e., low ambient NIR illumination, and thus only works indoors.

Lately, hybrid systems which combine a stereo camera setup with a separate NIR dot projector are gaining popularity, with Intel marketing their Realsense integrated solutions at a low cost [26]. Such a system works using normal block matching algorithms, but with the added texture information due to the NIR dot projector. Those systems can therefore work indoors and outdoors, albeit without the added accuracy from the projected texture in the latter case, as the projected light is generally not visible due to ambient NIR illumination. They also come with an application-specific integrated circuit which takes care of the processing and block matching, freeing up some computational performance on the host machine.

2.6 Motion Planning

Motion planning is comprised of both path and trajectory planning. Path planning defines a geometric path for which the robot must follow without defining dynamical attributes, such as velocity and acceleration. On the other hand, trajectory planning defines velocity, acceleration and/or jerk along the path [109–112]. Therefore, motion planning, in the context

of assistive robotics, is the task of obtaining a trajectory for the robot that ensures that the robot arm:

- Does not hit any obstacles (such as a table, walls or any objects).
- Is in a position to accomplish the wanted action at the end of the trajectory.
- Does not hit itself or be stuck in a singularity.
- Is slow enough and constantly checking for new obstacles to increase safety.

Briefly, motion planning finds a series of configurations (or a continuous function expressing these configurations) parametrized by the time from an initial configuration of the robot to a goal configuration [112].

A configuration is a specific point in the configuration space of the robot, which itself represents the set of all possible positions of all the points comprised in the robot's geometry depending on the position of the joints of the robot [111,112]. As such, knowing the position of the joints of the robot, direct kinematics is used to obtain the position in the Cartesian workspace of all the points belonging to the robot structure. On the other hand, knowing a position of a point, such as an obstacle, in the workspace, the inverse kinematics can be used to obtain the set of joint positions, or configurations, that intersects it. This is generally non-trivial for serial manipulators as the equations are non-linear and an analytic solution might not exist [112], which is often the case for serial manipulators without a spherical wrist.

2.6.1 Path Planning

Path planning is often seen as a search problem [113] on which several review of the state-of-the-art are available, such as [113–119]. While those works survey a wide variety of path planning algorithms, they do not necessarily apply to the case of serial manipulators (which have a high dimensionality), as is the case of ARAs. Indeed, [113] states that sampling-based path planning algorithms are usually preferred in the case of serial manipulators. This is because these techniques avoid building a complete representation of all the obstacles from the workspace to the configuration space. As discussed previously, this transformation is costly and difficult, and because of that, the collision-checking performed by sampling-based algorithms is better adapted as it uses forward kinematics only [112].

The most widely used path planning algorithms are, according to [113,118,119]:

- Rapidly Exploring Random Tree (RRT) and its modified versions

- Probabilistic Road Map (PRM) and its modified versions

PRM methods build a set of pre-computed collision-free *roads* from randomly sampled nodes in the search space and then use that reduced search space to generate a complete road from an initial configuration to a goal. This simplifies the problem to a 1D search which is generally done through a search algorithm such as A* [111, 112, 119, 120]. Because of the computational burden of first computing the road map, it is generally used in cases where multiple queries are performed on the same environment [111].

Also, the RRT algorithm is used to explore the search space randomly and, when using its modified RRT* version, leads to asymptotic optimality [119]. This algorithm is generally preferred for single queries, as it generally converges slowly [118].

While node-based algorithms such as A* are used for path planning [113], those generally identify a path in the workspace of the robot and not the configuration space due to its high dimensionality, which leads to suboptimal paths and longer movement times as is the case in [28] and [8, 50]. As stated previously, the high dimensionality of the configuration space and the difficulty of representing the obstacles in that space due to inverse kinematics makes sampling-based algorithms better adapted [112, 113].

2.6.2 Trajectory Planning

Trajectory planning is often seen as an optimization problem that, given an initial path, should produce a trajectory (a set of points to be interpolated from or continuous functions) minimizing for specific values, such as energy, jerk and length [110]. It is often formulated with constraints too, such as:

- Maximum torque at each joint.
- Joint position limits.
- Collision avoidance.

Furthermore, according to [121], trajectory optimization can generally be used in one of two ways:

1. Starting from a collision-free path obtained from a path planning algorithm, a trajectory planning algorithm is used to generate an optimized trajectory that is still collision-free.

2. Starting from a trivial (straight line) or a random path, a trajectory planning algorithm is used to both find a collision-free path and generate a trajectory.

As such, it might be tempting to use a trajectory planning algorithm to accomplish both the path and trajectory generation, but in general, trajectory optimization algorithms are prone to stay stuck in local minima [121–125]. Moreover, they generally lead to non-smooth trajectories because the optimization prefers collision avoidance over trajectory smoothness [126].

Some of the most widely used and readily available trajectory planning algorithms are available in MoveIt [127] and OMPL [128], including CHOMP [125], STOMP [123] and TrajOpt [121]. While CHOMP is generally faster than STOMP, STOMP is less prone to local minima and hence has a higher success rate [123]. The newly implemented TrajOpt algorithm leads to generally much faster computation, an order of magnitude faster, and has a good success rate, although it is still being prone to local minima [121].

2.6.3 Conclusion on Motion Planning

In conclusion, while complete motion planning is possible using a trajectory planning algorithm such as TrajOpt or CHOMP, care should be taken as it might lead to non-smooth trajectories or it might simply be unable to find a collision-free path. To ensure a solution, a path planning algorithm should be used to first find a feasible path, then a trajectory optimization can be done to obtain a smooth, collision-free path.

CHAPTER 3 RATIONALE AND GENERAL OVERVIEW

3.1 Problematic and Frontier of Knowledge

The previous literature review goes on to show that, although ARAs are beneficial for their users, there is still a very present need for better HCIs for ARAs, as most commercial robotic arms are used via a joystick interface which requires good dexterity and experience to use efficiently. Given that the people who could benefit from ARAs have a wide spectrum of upper limb disabilities with varying degrees of dexterity, joysticks are generally not well adapted to all cases, requiring difficult to use adaptations such as a tongue-controlled joystick [39]. Furthermore, they generally require a consequent amount of time and workload to accomplish simple actions such as grasping an object.

Problem 1: The human-computer interfaces provided with commercially available assistive robotic arms are generally not adapted to users needs living with low dexterity and upper limbs disabilities.

Problem 2: To the author’s best knowledge, no solution presented in the literature is robust and versatile enough to be broadly adopted by people that could potentially benefit from these assistive robotic arms.

One promising solution is the use of gaze tracking to control such robots. Although proofs of concept exist, as the ones that were developed by Pr Achiche’s research team [7, 8, 28] using either a stereo vision, a Kinect Camera or Facelab5, several problems still stand in the way of broad adoption:

- No robust, low cost and nonintrusive gaze tracking solution exists that compensates for head motion and works both indoors and outdoors.
- The existing proof of concept requires a full desktop computer for processing, making it impractical to embed on a motorized wheelchair due to size, weight and power requirements.
- The proof of concept’s interface is not adapted to users with disabilities due to development constraints and its scope.
- The proof of concept’s processing and robot movements still require a consequential time to accomplish a grasping task (over 16 seconds).

- The proof of concept’s grasping is currently hard-coded and cannot adapt to all objects, since it does not perform any object segmentation.
- The proof of concept does not segment the 3D scene into separate objects, leading to missed grasps and knock overs.
- The proof of concept uses extensive calibration procedures which has to be repeated periodically and whenever a new user uses the system.

A novel gaze tracking technique should therefore be developed, combined with a scene segmentation algorithm and better path planning and grasping, all the while being able to run on an embedded platform and in a robust to illumination conditions manner.

3.2 Objectives

As discussed previously, there is a need for a better way to interact with assistive robotic arms which could work for almost all cases of upper limb disabilities and thus lead to broad adoption. For such a system to succeed, robustness to outdoor illumination and ease of use should be some of the top priorities. It should also be developed on an embedded computer and thus have low computational requirements as to achieve real-time performance. This PhD thesis therefore proposes a solution to the previously described problems.

3.2.1 General Objectives

The general objective of this PhD project is as follows:

Develop an embedded vision-based solution for facilitating the usage of assistive robotic arms for people with upper limb disabilities.

It is worth noting that the developed solution needs to function in unconstrained environments which includes both indoor and outdoor settings.

3.2.2 Specific Objectives

This objective can be further divided into four specific objectives:

1. Develop a calibration-free, low cost, non-intrusive and embedded gaze estimation system.
2. Develop a real-time 3D scene reconstruction system using computer vision with object detection and segmentation.

3. Develop a real-time path/trajectory planning algorithm taking the full robot geometry into account and leading to fast grasping times.
4. Validate the developed system with device trials with typically developed participants.

3.2.3 Contributions

The original and significant contributions of this thesis are the methods and processes developed to achieve the previously described objectives. Indeed, the developed system is the first of its kind to reach a high level of maturity and functionality, as it is the first to be able to perform real-time gaze tracking, 3D scene reconstruction, object detection and segmentation, path planning and grasping on an embedded platform. Furthermore, it is the first to be able to perform all of these tasks robustly in unconstrained environments, including with sunlight illumination.

Therefore, the methods described can be used as both a basis and a framework on which to build further gaze-assisted control modes for robotic arms, as some are described in Chapter 6.4.6.

3.2.4 Scope

The scope of the project is limited to developing a robust to illumination conditions gaze control system for ARAs, and as such should not go out of its way to develop solutions to broad and general problems such as: general gaze estimation, object detection and path planning.

The developed system can therefore sacrifice on generality to accomplish the previously discussed objectives. One example of such a sacrifice is on the gaze estimation system: we can safely assume that a mounting point for a camera exists on the motorized wheelchair or robot base and as such, no body-mounted solution for the gaze tracking system is required.

Another example is on the accuracy and reach of such a system. In fact, wheelchair mounted ARAs generally have a reach of up to 1 meter. Therefore, the system should not be designed for accurate grasping or scene segmentation for distances past the robot's reach.

3.3 General Overview and Articles Description

This thesis contains three (3) articles which describes parts of the gaze control system or its development. These are titled as follows:

1. Non-Intrusive Real Time Eye Tracking Using Facial Alignment for Assistive Technologies (Chapter 4)
2. Semi-synthetic Dataset Augmentation for Application-Specific Gaze Estimation (Chapter 5)
3. On The Improvement of Gaze Control Methods for Assistive Robotic Arm (Chapter 6)

3.3.1 First Article

The first article describes the novel gaze estimation deep learning model developed for this system. It details its implementation and training, with the main contributions as follows:

- It describes an open-source gaze estimation workflow that can be trained on publicly available datasets which performs at a comparable level to current state-of-the-art in terms of accuracy while being much faster to execute, especially on Central Processing Units (CPUs).
- The developed code does not require person-specific calibration, has low compute performance requirements, is non-intrusive and works reliably in indoors and outdoors light conditions since it does not use PCCR.

3.3.2 Second Article

The second article describes a novel data augmentation technique which allows for a shift in the headpose and gaze direction bias of existing gaze estimation datasets. Indeed, most publicly available gaze estimation datasets were obtained through the participants looking at target points on a computer, laptop, tablet or phone screen with a camera placed above the screen taking pictures of the participant.

Therefore, the distribution of head poses and gaze angles of these datasets are not representative of the application for controlling ARAs since the camera is placed much further below the user.

The second article therefore describes a solution to this issue by using a facial mesh reconstruction technique and synthetic rendering from real images, which leads to vastly improved accuracy in this application.

3.3.3 Third Article

The third and last article in this thesis describes the entire control system and the implementation details for all parts of the system except the gaze estimation model. It also details the changes made to improve the gaze estimation model in the context of an HCI for ARAs.

It also contains the device trials results, discussion and conclusions. It demonstrates that the use of a gaze estimation control system for ARA translates to drastic improvements in time to grasp objects and learning curve while having a high satisfaction score.

Considering this article synthesises the different parts and workflows of the final complete system, the discussion contained in this article is essentially the general discussion of the gaze control system. Chapter 7 will instead discuss the implications of the results obtained from all articles in the context of the literature review of Chapter 2.

3.3.4 Technology Choices and Justifications

As there are some limitations due to cost, time and availability constraints, this section describes the technology choices made during this PhD thesis.

1. To ensure that the control system can be mounted on a motorized wheelchair with limited battery life, an Nvidia Jetson AGX Xavier [129] embedded computer was selected with its integrated CUDA-capable GPU and maximum 30 Watts of power consumption. The developed system runs on this embedded computer.
2. While the previous proof of concept and the first prototypes of this thesis were programmed mostly in the C++ programming language, it was changed to Python with some C++ when necessary for easier and faster prototyping.
3. The 3D camera used in this system is the Intel Realsense D415 [26], which contains an application-specific integrated circuit which computes the disparity between its two imagers, freeing up computational resources. The library to interface with it is available on most platforms and in multiple programming languages, including Python.
4. The ARAs used in this project are the MICO and JACO arms as sold by Kinova [130] as they were available in the laboratory of Pr. Sofiane Achiche.
5. Because this project does not use ROS [131] both due to its limited availability in Python and due to real-time processing constraints, the MoveIt [127] motion planning library could not be used for this project.

6. Because of the unavailability of Kinova’s drivers on embedded ARM architecture-based computers, no trajectory planning could be done. Indeed, an open-source driver for communicating with the robot named *libkindrv* [132] was used, but with unsuccessful attempts to use the velocity or torque control modes. Therefore, only joint position points were sent to the robot. While this might be a large constraint, it also implies that such a system can work even with very basic access to the robot’s control methods.

CHAPTER 4 ARTICLE 1: NON-INTRUSIVE REAL TIME EYE TRACKING USING FACIAL ALIGNMENT FOR ASSISTIVE TECHNOLOGIES

Cedric Leblond-Menard, Sofiane Achiche

Accepted by IEEE Transaction on Neural Systems and Rehabilitation Engineering journal on January 17th, 2023. TNSRE-2022-00336.R2

4.1 Abstract

Most affordable eye tracking systems use either intrusive setup such as head-mounted cameras or use fixed cameras with infrared corneal reflections via illuminators. In the case of assistive technologies, using intrusive eye tracking systems can be a burden to wear for extended periods of time and infrared based solutions generally do not work in all environments, especially outside or inside if the sunlight reaches the space. Therefore, we propose an eye-tracking solution using state-of-the-art convolutional neural network face alignment algorithms that is both accurate and lightweight for assistive tasks such as selecting an object for use with assistive robotics arms. This solution uses a simple webcam for gaze and face position and pose estimation. We achieve a much faster computation time than the current state-of-the-art while maintaining comparable accuracy. This paves the way for accurate appearance-based gaze estimation even on mobile devices, giving an average error of around 4.5° on the MPI-IGaze dataset [71] and state-of-the-art average errors of 3.9° and 3.3° on the UTMultiview [2] and GazeCapture [72, 133] datasets respectively, while achieving a decrease in computation time of up to 91%.

4.2 Introduction

Assistive robotic arms (ARA) are known to be greatly beneficial to people with upper-limb disabilities. [15–19] Therefore, great efforts are put into the human-computer interfaces (HCI) to control these robots. Indeed, most commercially available ARAs come with joystick-based control devices, but those require significant amount of time to learn how to be used and to accomplish simple tasks, even for typically developed users [20]. Even though control filters have been implemented to help reduce movements caused by involuntary jerks from the user [15], the joystick based HCIs are generally much more frustrating to use than more automated HCI, depending on the user’s impairment. [16] In some cases, potential users

might have specific disabilities which make the use of a joystick via their hands impossible. As pointed by [22], the limits of the control interfaces currently available for commercial ARAs creates a situation where the people that should benefit the most of the ARAs are unable to control them due to their severe motor disability. Thus, the research work presented in [16] reports that a vision-based interface with autonomous path planning would lead to a significant improvement in user workload in grasping and pick-and-place tasks. Indeed, the user gaze in grasping tasks has been proven to be a large part of predicting intent and point of contact [134].

As such, current trends include using a camera, either grayscale/color or combined with a depth sensor, to detect objects on the scene presented in front of the user and a HCI to make decisions on what and how to manipulate these objects [8, 28–30, 32, 34]. Recent work has promoted the use of eye tracking as a HCI as demonstrated by [7, 8, 38, 39, 41–45, 134, 135], albeit with significant limitations that will be further discussed in the next section.

4.2.1 Eye Tracking for Assistive Robotic Arms Control

Several extensive literature reviews are available that describe some of the current and previous state-of-the-art methods of eye tracking and gaze estimation, with a recent example being [52]. An even newer literature review has recently been published with comparative results in terms of angular gaze accuracy for deep learning based methods. [136] An overview of all the available methods is out of the scope of this paper.

One of the most popular and commercially available eye tracking techniques relies on using infrared (IR) corneal reflection aptly named *Pupil Center Corneal Reflection* method (PCCR). This method relies on an IR illuminator producing a distinctive reflection on the user eye’s cornea for which the position can be compared with the more easily identifiable iris center (because it usually appears as a deep black circle). While this method produces accurate results [52], its reliance on IR reflection makes it particularly prone to error when exposed to sunlight, as [8] experienced. Otherwise, other accurate methods rely on the user wearing a head-mounted eye tracker or other intrusive systems when used in controlling ARAs.

Other methods have been recently proposed that make use of a single monocular camera to estimate the gaze direction [72, 73, 137–139], but those are, in our experience, either imprecise or ill-suited for embedded systems due to their computing performance requirements, especially in the case where no person-specific calibration frames are used.

Therefore, in this paper a novel non-intrusive real-time gaze estimation technique for use with ARAs that work reliably both indoors and outdoors while requiring as low a power as

necessary when running on embedded systems without requiring person-specific calibration. The performance of the developed method is then compared to openly available state-of-the-art algorithms.

4.2.2 Objectives

Given the context of using ARAs, we therefore have the following research objective:

Develop an open-source real-time gaze estimation model which can run on mobile and embedded devices in both indoor and outdoor environment using a single camera and without person-specific calibration.

Indeed, if this objective is completed, the resulting model could be used in systems such as [8, 28] to provide a more versatile solution at a low cost.

4.2.3 Contributions

The contributions of this paper are as follows:

- We demonstrate that using state-of-the-art real-time face detection and iris alignment convolutional neural network (CNN) based algorithms, a gaze estimation system can provide an increased accuracy in estimating the gaze angles comparable to the current state-of-the-art while requiring no person-specific calibration and having a low performance requirement such as running in real-time on mobile devices.
- We bring forth a new framework of eye tracking tailored for robotics controlled that works reliably both indoors and outdoors while providing real-time performance, over 20 frames per second even on a single CPU core, and without being intrusive to the user. This model also outputs facial landmarks useful for assistive tasks as well.

4.3 Related Work

As introduced earlier, several methods that make use of a single camera aimed at the user’s face have been recently described in the literature. As a comparison basis, we take particular interest in gaze estimation datasets that include full images of the user’s face, as this allows for a more realistic scenario in the context of assistive devices as the face position must be located with respect to the camera to give an accurate estimation of the gaze angles.

One of the most widely used datasets for gaze estimation proposing full facial images is MPIIFaceGaze [1]. MPIIFaceGaze is a modified subset of the widely used MPIIGaze dataset [140] that includes complete faces instead of the eye region only found in the original dataset. It consists of 37,667 images and their corresponding gaze data taken across 15 participants. A recent paper described a new dataset called RT-GENE [138], but this dataset corresponds to scenarios where the user is not the main object of the picture and thus the faces are generally far from the camera. As this does not represent a typical scenario for gaze estimation control of ARAs, this dataset was not used in the context of this paper.

On the MPIIGaze dataset, the currently most accurate method available in the literature is called FAZE [72] and uses a combination of deep learning models to estimate the gaze direction and head pose of the user. This method implements a metalearned model to generate the weights of a gaze estimation network from only a few calibration frames (32 and less). As reported by the authors [72], the accuracy when using no calibration frames is 5.23° . The authors note that a real-time demo is available while providing no further description of computational performance.

Furthermore, the authors of the RT-GENE dataset proposed a gaze estimation model based on first extracting the facial landmarks using Multi-Task Cascaded Convolutional Networks [141] then correcting for the face image for the pose perspective against averaged face pose landmarks. The corrected eye patches are then extracted from the image and fed to VGG-16 networks, in an ensemble or not [142], one for each eye patch, to estimate the gaze direction. The reported gaze estimation error is 4.3° on MPIIGaze using an ensemble of 4 models.

Most other recent deep learning-based methods have achieved an angular accuracy ranging from 4.1° to 7.3° , as described in [136].

In 2019, a gaze estimation dataset named GazeCapture generated from phone and tablet gaze estimation trials was used [133] for angular gaze estimation [72]. While the main use of the dataset is to train gaze estimation models for estimating a gaze point on a mobile device’s screen, thus the accuracy values generally given are in pixels, it is possible to use the dataset for gaze angles estimation by converting the pixel points to 3D points and then gaze angles, as is done in [72].

Moreover, another dataset that is widely used for gaze estimation tasks is the UTMultiview dataset. [2] This dataset offers a larger variety of head poses and gaze directions by using synthesized images from reconstructed faces using an array of cameras.

4.4 Methodology

4.4.1 Workflow and implementation details

The overall workflow proposed here can be separated in four distinct sections, which will be described here. These sections are:

- 1) Face detection
- 2) Head pose correction
- 3) Eye patch extraction
- 4) Iris detection and gaze estimation

As such, the input of the workflow is an image containing the face of the user. This image should be a color image and contains enough details to distinguish the iris from the sclera and eye contour. The output is composed of the gaze angles, pitch for the up-down eye motion and yaw for the left-right motion, and the gaze origin is the center point between the eyes. To ensure the accessibility of this gaze estimation method, we ensure that no step requires person-specific calibration, rather relying on existing large dataset on which to train on once prior to using this model.

Initially, a face detection algorithm is run on the image to detect the most prominent face in it. From that detection, we then must extract the location of the eyes, which is generally done through facial landmark extraction (generally named facial alignment). Given the position of the eyes, we can then extract a pair of image patches of each eye. These image patches are then used as the input of the gaze estimation algorithm, which outputs the gaze angles (direction). Since the position of the user's head should be known to infer the gaze point of gaze from the gaze direction, the facial landmarks can be used to compute the origin of the gaze vector.

Moreover, to reduce the influence of the head pose on the variability of the input images, we removed the roll component of the head pose and used the roll-removed head pose yaw and pitch angles as input to our model. Indeed, the world coordinates gaze direction $\vec{g}$ is a function of the head pose matrix H and the eye gaze direction with respect to the face $\vec{g}_e$:

$$\vec{g} = H \vec{g}_e$$

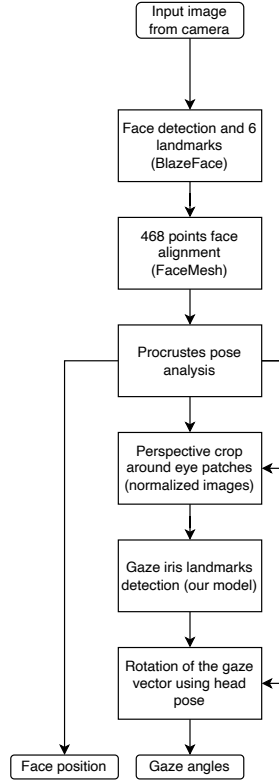


Figure 4.1 Our workflow is mostly sequential, with the head pose being used to normalize the eye patch images and correct the estimated gaze.

Note that the vectors and matrices given here are in homogeneous coordinates. Since $\vec{g}$ and $\vec{g}_e$ are directions, the last component of the homogeneous vectors is 0. The method of obtaining the head pose H and applying the perspective transform will be discussed in Section 4.4.1.

Face detection

In general, the approximate face location must be known beforehand to crop the image around the face center and perform the facial feature alignment and iris detection. Once a general location of the face is known, it is possible to skip the face detection step and use the general region of the previously detected face to perform the facial alignment step, especially if a confidence score is available from the alignment step which is true in our case. This thus saves time in processing a frame in a continuous video stream. A simple static threshold can be used such that if the confidence score falls below a fixed value, the face detection step is run again. For every face alignment step, we compute the center position of the face and check how far it moved from the last frame, and then re-centering the face-cropped image by moving the center by how far the face alignment center moved.

It should be noted that in the comparative results of this paper, this strategy is not used,

since the datasets used for comparison are not videos, but rather distinct images and as such this strategy cannot work.

One of the first successful real-time face tracking algorithm was described by Viola and Jones using Haar-like cascades [78] and is still used. More precise and stable real-time algorithms have been proposed afterward, including histogram of oriented gradients (HOG) with support vector machines (SVM) and linear binary pattern (LBP) cascades, with HOG being generally the most accurate of these methods [143]. Recently, breakthroughs in small optimized CNNs led to very accurate face detection yet real-time performance on embedded computers and portable devices. One such state-of-the-art model, named BlazeFace, was developed and trained by Google Research with very high precision and fast inference time on mobile devices [90]. Based on MobileNetV1/V2 [144, 145], the architecture implements further optimization including increasing the receptive field size by using 5×5 kernels over 3×3 which are cheaper than adding layers with more computationally expensive pointwise convolutions.

Whereas our initial work was based on a modified HOG algorithm [81] parallelized and ported to CUDA, the recently release BlazeFace algorithm performs about as fast in our experience while providing increased accuracy comparable to state-of-the-art real-time models according to [90]. This is thus the chosen algorithm for our workflow. Furthermore, this algorithm outputs the location of 6 facial landmarks (the two eye centers, the two ear centers, the nose center, and the mouth center) which we can use to compute and correct for the head roll by assuming the left and right landmarks should be on a horizontal line.

Head pose correction

As stated previously, we can correct for the head pose by using the detected facial landmarks and applying a perspective transform to the eye image patches as a form of normalization, as was suggested in [71]. This allows the network to train on estimating the gaze direction without having to directly account for the head pose variation in the image. The pose normalization technique we used is based on the one described in [138].

As such, this requires several facial landmark points to be known on which to find the optimal series of rotation to align the actual head pose with a reference (sometimes called “canonical”) head pose.

An effective real-time method of facial alignment recently published is called FaceMesh [93] and is part of the same augmented reality framework as BlazeFace developed by Google. It uses a straightforward residual neural network that outputs a set of heatmaps for each facial

feature (in this case 468 points) on which subpixel maximum estimation is done to find the input image location of the landmarks. This aligner is trained to not only output the 2D pixel location of each landmark, but also a depth value associated with each landmark that corresponds to the difference with the average depth of the face, while keeping the same scale (or aspect ratio) as the horizontal coordinates.

The face alignment model is learned from deforming a reference model with dimensions given in centimeters but projected to the image according to the camera model i.e., the camera matrix and distortion coefficients. Therefore, we can iteratively find the projected 3D transformations that best fit the transformation from the reference model and the computed landmarks points through a Procrustes analysis, also known as an orthogonal Procrustes problem, as used by the team at [146] and described in [147].

Using this method, we thus compute the head pose with respect to the camera, the previously described H . We can then find the gaze vector origin which is assumed to be the eye landmarks center point by using the canonical model’s eye center $\vec{e}_c$ and head pose matrix H :

$$\vec{e} = H\vec{e}_c$$

Here, $\vec{e}$ is the eye center position with respect to the camera. Since the vectors here are positions, their last component is 1.

Given the direction to $\vec{e}$ and head pose H , we can find the perspective transformation matrix W that derotates the image as to make the head appear upright and that aligns the camera view axis with the eye position e while reprojecting the image as if the eye was at a new distance $\|e\|^*$ and a new camera focal length l^* . This method was first proposed by [2] and then revisited by [148], the latter being the normalization technique used in this paper. By choosing the ratio between the new eye distance $\|e\|^*$, the new camera focal length l^* and the width and height of the normalized image, the resulting normalized image will have the de-rolled eye centered in the image with a specific and constant scale. In our case, the values used for $\|e\|^*$ and l^* are 600 millimeters and 650 pixels respectively. The distance $\|e\|^*$ of 600 millimeters is generally suggested by the literature [2] whereas the focal length l^* is generally found by trial and error until a value is found that has the right scaling ratio as to make the eye appear the chosen size for the model and input image resolution.

This removes parts of the perspective variance due to the head pose roll and camera parameters from the image and thus reduces the complexity of the problem to be learned by the gaze estimation model [71, 72, 138].

Eye patch extraction

From the facial landmarks, eye image patches must be extracted for the iris detection and gaze estimation part of our workflow as seen on Figure 2. To do so, we use the average position in 3D of the eye from the previous step perform the perspective transformation normalization described in [138]. This method corrects for the head pose by warping the image as to align the eye patch normal vector to the vector between the camera’s position and the eye position in 3D and re-projecting the image at a constant distance. This constant distance is set as to allow for a 25% margin on each side of the eye patch image, as is required by the iris detection and gaze estimation model that will be discussed in the following section.

Iris detection and gaze estimation

It is intuitive to think that the eye region landmarks and the iris center contain enough information about the user’s gaze to estimate its direction. Indeed, the literature seems to point to a clear consensus on the subject [137–139, 149].

As such, we start from a model made to obtain the position of the iris given a cropped image of the eye on the right side from the user’s perspective. For this model, we used as basis the very lightweight (few learned parameters) method recently described in [150] to align the eye region and find the iris center and contour landmarks.

While our initial work was based on an explicit relation in position between the eye region features (namely the lower sclera contour) and the iris center, where the vector component in the difference in position was used as the input of a basic polynomial regression or SVR regression to keep the computational burden to a minimum. This unfortunately in our experience lead to poor accuracy when compared to the state of the art, even when corrected for the head pose.

Considering the architecture of the aforementioned CNN model, we can infer that there is a lot of information encoding the position of the eye region landmarks and iris in the latent space between the backbone and the landmark heads (see Figure 4.2).

Therefore, we ended up modifying the iris detection model by getting rid of the iris and eye landmarks splits of the original model and adding a new split to estimate the gaze direction. By using the initial model’s weight and biases of the backbone, we can ensure that the model has a headstart in learning eye-specific encodings. We then concatenate the computed features of the head with the eye poses (face pose at each eye) as an input to a final, fully connected set of layers. The final output of this set of layers, a multi-layer perceptron (MLP), is thus the estimated gaze angle.

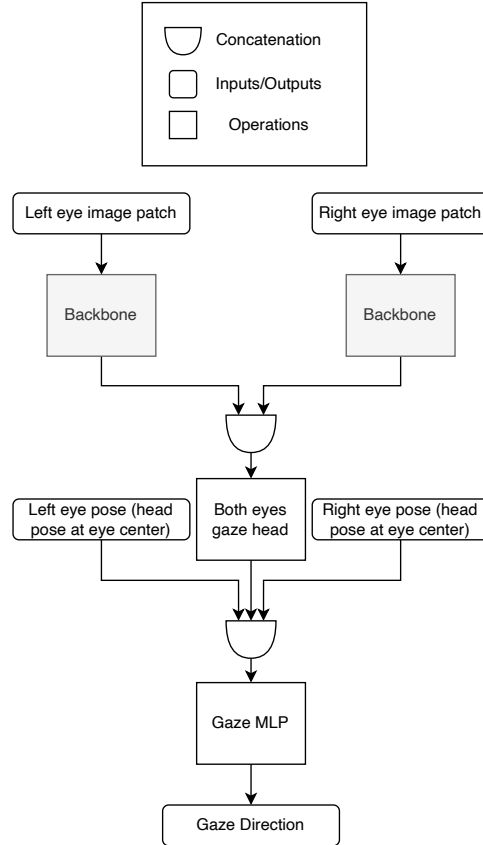


Figure 4.2 A simplified overview of our model. Blocks with a darker fill represent parts of the model reused from the trained iris landmarks model from Mediapipe.

The backbone and split are made of Irisblocks [150] which are themselves based on Blazeblocks [90] which are themselves a modification of Residual blocks [151] where a convolution and activation layer is followed by a depthwise convolution followed by a normal convolution while the residual feedforward is an identity function or a max pool layer when the stride of the convolution layers lead to a reduction in the width and height dimensions. The sum of the convolutions and the residual are then passed through an activation function, PReLU in this case. In the specific case where the number of channels of the output of the block is higher than the number of input channels, the residual is padded to ensure the same number of channels when it is added to the output of the convolutions (see Figure 4.3)

Furthermore, we can double the model to make use of images from both eyes using a horizontally flipped image of the left eye and concatenate the feature vectors of each model split as the input for the gaze estimation head to improve the accuracy at the cost of increased computational complexity, as seen in Figure 4.2 and 4.4.

By reusing the same backbone weights but horizontally flipping the left eye image, we found that our model often achieves better generalization, i.e., lower evaluation error.

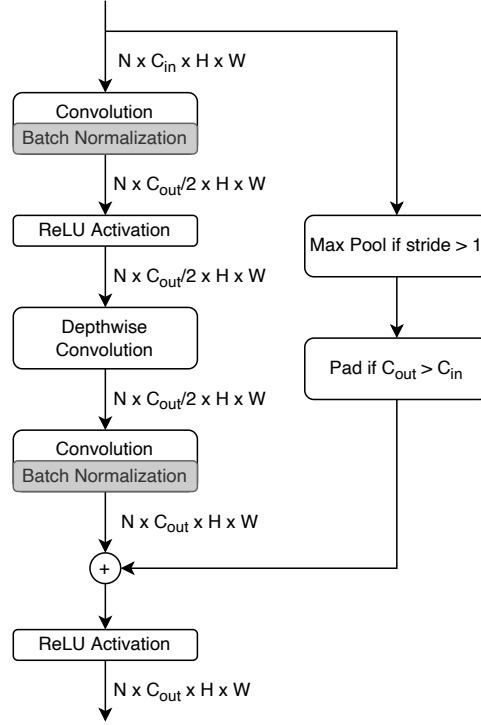


Figure 4.3 The Irisblock is a variation on the BlazeFace blocks. They are a form of Residual blocks, but using a depthwise convolution to reduce the number of operations. The batch normalization layers used during training are in darker shade.

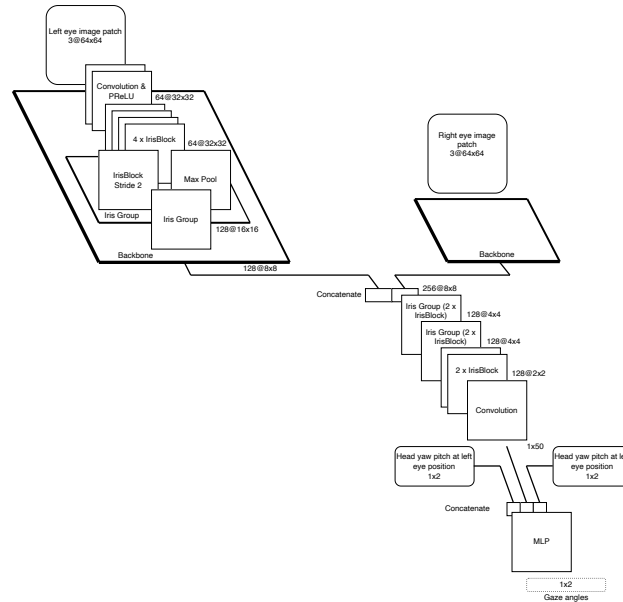


Figure 4.4 Our gaze estimation and iris landmarks recognition model. The tensor dimensions are given between each operation. For the sake of simplicity, some operations implementations are not duplicated but are shown as white boxes.

Also, as suggested in [138], the head pose angles obtained from the Procrustes analysis and used for image normalization can be added as input to the model to account for some change in appearance of the eye due to the pose differences between frames. As such, we directly concatenate the head pose angles to the other features before the final fully connected layers.

Other implementation details

All the used deep learning models were implemented in PyTorch [152] and PyTorch Lightning [153], allowing them to run on both the CPU and GPU interchangeably. As such, our method is written in Python and PyTorch and makes the use of OpenCV [154] just for some basic image processing functions.

Although we train our modified version of the iris landmarks localization model proposed in [150], our workflow uses both the BlazeFace [90] model and the FaceMesh [93] model as trained by and available in Mediapipe [146], albeit with the models converted to PyTorch while keeping the same parameter values. The original model was implemented for TensorFlow-Lite [155] inference.

4.4.2 Training and validation

Given our gaze estimation method is based on the iris landmarks localization model from [150], we start by re-using the model’s parameters that correspond to the unmodified model backbone as the backbone of our model. This allows us to re-use the learned feature extraction for the eye and iris landmarks to help convergence of the optimization.

Training is done on the MPIIGaze [71] dataset by following the splits given in [72] and [138]. As such, for the MPIIGaze dataset, we use a leave-one-out approach for the test set.

We trained our model using the Ranger21 [156] optimizer, which implements recent progress on Rectified Adam [157] and other methods that rely less on fine-tuning hyperparameters. The batch size is 32, the learning rate is set to the default 0.001 and the β_1 and β_2 parameters are set to 0.8 and 0.7 respectively to slow down the training speed. We also use a weight decay penalty of 0.001. The loss function is the mean square errors between the output gaze angles and the ground truth gaze angles as given by the direction between the eye center points (between the eyes) and the gaze targets of the dataset.

4.4.3 Performance comparison methodology

In order to allow for a common-ground comparison between the different algorithms tested, the PyTorch version of the tested methods was preferred if other implementations were available. Furthermore, we did not make use of optimized runtimes such as TensorRT that would provide an increase in inference speed for the same reason. All algorithms are run from Python and the inference time is taken from the start of the computation, just after the image frame is loaded in memory, to the end of the gaze estimation after the gaze angles are obtained, without any user interface processing or drawing functions being run.

The error metric e used is the arc cosine of the cosine similarity measure, given by computing the angle between the real gaze direction $\vec{g}$ given by the annotations of the dataset and the estimated gaze direction $\vec{g}_e$ given by the gaze angles, defined as:

$$e = \cos^{-1} \frac{|\vec{g} \cdot \vec{g}_e|}{\|\vec{g}\| \|\vec{g}_e\|}$$

This metric has interesting properties for comparison, as it is always positive and thus the average corresponds to the angular error between the directions, as opposed to the mean angular error in the pitch and yaw angles separately.

4.5 Results

The results are presented in three distinct tables. Table 4.1 presents the results as either given by the original authors or in our case using the dataset supplied head pose information for the MPIIGaze [71] dataset.

Table 4.2 presents the results for the GazeCapture dataset [133] using the annotations and evaluation split defined by [72].

Table 4.3 on the other hand contains the results for the UTMultiview dataset [2] using the 3-fold evaluation split as used in [2, 158]. In our case, since our model uses both eyes images as input, we rendered the same head poses for each sample (-36 to 36°), but rendering both eyes instead of one at a time. This needs to be done as the left eye and right eye images of the pre-rendered images given with the dataset are not matched to one another.

Table 4.4 contains the results we obtained running the implementation of the compared algorithms as distributed by the original authors but modified to remove all drawing and interface-related parts of the original code to allow for a better comparison in performance. All implementations are available on the authors Github repositories [158, 159]. The trained weights used are those made available by the original authors and we use a 7,500 images subset of the MPIIFaceGaze dataset [1] to evaluate on.

We decided not to include accuracy figures of this evaluation since it is not clear on what dataset the provided weights were trained on for the FAZE model [72].

In Table 4.4, the time performance metric (inference time) is given as an average in three scenarios: using GPU acceleration, without GPU acceleration but using all CPU cores and without GPU acceleration and using a single CPU core. This gives a better idea of the available performance in power constrained scenarios such as in mobile applications and embedded systems.

4.6 Discussion

4.6.1 Interpretation

As the results show, our model reached accuracy levels comparable to the current state-of-the-art while being significantly faster, especially when running only on the CPU. Given the original goal of using this gaze estimation model for assistive technologies such as controlling an assistive robotic arm directly mounted on a motorized wheelchair, this performance improvement is significant. We thus achieve a reduction of computing time of 39% against the fastest method in the worst-case scenario and up to 91% against the slowest in the best-case scenario.

Moreover, as we can see from the frame times, our GazeIrisLandmarks model requires lower computation time due to the low number of parameters. In fact, as can be seen from the multi-core and single core frame times, the overhead of splitting the task on multiple cores makes the overall computation almost as slow as running it on a single core. This is confirmed by the model size used, with ours being 3.8 megabytes, whereas the RT-GENE 4 models ensemble is 1312 megabytes in total and the FAZE model is 27.5 megabytes.

As for the accuracy of our model, Tables 4.1, 4.2 and 4.3 show that our model can reach accuracy figures comparable to the current state-of-the-art, being better in certain datasets and slightly worse in others.

4.6.2 Limitations and further work

Throughout our research, we have found that performance and computational requirements are rarely discussed extensively, let alone analyzed quantitatively, especially when comparing gaze estimation techniques. There is currently no benchmark or standardized way of comparing computational requirements of gaze estimation methods. This thus requires making the original model code available online for comparisons to be made and thus limits the number

Table 4.1 Accuracy for the MPIIGaze Dataset

Method	Accuracy
FAZE (no calibration) [72]	5.2°*
RT-GENE (4-ensemble) [138]	4.3 ±0.9°
GazeIrisLandmarks (ours)	4.5 ±1.2°

*The standard deviation is not provided in the article.

Table 4.2 Accuracy for the GazeCapture Dataset

Method	Accuracy
FAZE (no calibration) [72]	3.5°
GazeIrisLandmarks (ours)	3.3°

As this dataset has a single evaluation set and not a leave-one-person-out evaluation, no standard deviation figures are provided. No standard deviation figures were provided in the FAZE article as well [72].

Table 4.3 Accuracy for the UTMultiview Dataset

Method	Accuracy
RT-GENE (4-ensemble) [138]	5.1 ±0.2 °
GazeIrisLandmarks (ours)	3.9 ±0.5 °

Table 4.4 Performance Timings (Full Detection and Gaze Estimation)

Method	Frame Times (milliseconds)		
	GPU	CPU	SC
FAZE (no calibration) [72]	228	254	431
RT-GENE (4-ensemble) [138]	36.4	255	323
GazeIrisLandmarks (ours)	22.2	38.9	38.3

Frame times are given from the moment after the image frame is available to the moment after which the gaze vector is available. GPU corresponds to a run with GPU acceleration, CPU using cores but without GPU acceleration and SC is using a single CPU core. Results are given without using the annotations but instead relying on the full workflow as made accessible by the authors to compute the estimated gaze direction and origin.

of models that can be compared.

Indeed, because performance numbers such as inference time depend on the hardware on which the numbers are produced, comparison must often be relative, where performance numbers for all compared methods must be run on the same computer as is the case for this paper.

It should also be noted that advances in technologies and new dedicated hardware, such as inference hardware now found in some smartphones could favor some algorithms over others,

which is a limitation of the comparison method used in this paper.

Moreover, as demonstrated by [72], deep learning models can be adapted to leverage meta-learning frameworks to improve accuracy of models by using calibration with very few samples. Given the context of assistive technologies, very long calibration procedures might be off-putting or just not possible but acquiring just a few calibration samples might be a good way to improve person specific accuracy and is therefore considered as a possible path of improvement. The drawback of such an approach is the added hyperparameter tuning required on top of the very high training computational requirements since higher order gradients are required [72].

Finally, the work presented in this paper focuses on appearance-based gaze estimation methods using singular cameras. Other methods exist, such as infrared reflections-based PCCR methods [52] which are often found in commercial eye trackers or calibration-based methods [139].

Some examples of the limitations of appearance-based gaze estimation methods without calibration includes lower accuracy when compared to PCCR methods and training dataset biases, where for example the ethnicity of the subjects in the training dataset limits the attainable accuracy with some users [160].

Furthermore, the domain of gaze and head pose angles for which an appearance-based model achieves accurate results is in our experience limited by the domain on which it is trained. As such, use cases for appearance-based models should take into consideration the training dataset’s gaze and head pose angles domain and the typical use case angles to ensure they remain within the dataset’s domain.

4.7 Conclusion

We have shown that leveraging existing very small pre-trained models for eye region landmarks recognition and modifying the structure to access the latent information within the model can lead to accuracy comparable to the current state-of-the-art while significantly reducing the computational requirements. While we achieve a 4.5° average error which is similar to the current state-of-the-art for appearance based gaze angle estimation for the MPIIGaze dataset [136], we see a decrease in computation time ranging from 39% up to 91% against the current appearance-based gaze estimation methods publicly available. We also achieve a state-of-the-art 3.9° average error on the UTMultiview dataset [2] and 3.3° error on the GazeCapture dataset [72, 133].

This also allows the model to produce face and eye landmarks that can be used for other

vision-based assistive tasks as well without needing further computation because these features are computed as part of the gaze estimation workflow. Furthermore, we show that a holistic approach describing a complete workflow such as proposed in this paper leads to improved accuracy when translated to real-world scenarios due to the high dependency of certain models to accurate and manually obtained annotations.

An implementation of our method, including a trained model, training description and real-time demo is available at: [*https://github.com/cedriclmenard/fastgaze*](https://github.com/cedriclmenard/fastgaze)

CHAPTER 5 ARTICLE 2: SEMI-SYNTHETIC DATASET AUGMENTATION FOR APPLICATION-SPECIFIC GAZE ESTIMATION

Cedric Leblond-Menard, Gabriel Picard-Krashevski, Sofiane Achiche

*Submitted to Computer Vision and Image Understanding journal on December 16th, 2022.
CVIU-22-1040*

5.1 Abstract

Although the number of gaze estimation datasets is growing, the application of appearance-based gaze estimation methods is mostly limited to estimating the point of gaze on a screen. This is in part because most datasets are generated in a similar fashion, where the gaze target is on a screen close to camera’s origin. In other applications such as assistive robotics or marketing research, the 3D point of gaze might not be close to the camera’s origin, meaning models trained on current datasets do not generalize well to these tasks. We therefore suggest generating a textured tridimensional mesh of the face and rendering the training images from a virtual camera at a specific position and orientation related to the application as a mean of augmenting the existing datasets. In our tests, this lead to an average 47% decrease in gaze estimation angular error.

5.2 Introduction

In recent years the number of gaze estimation datasets is growing [1,2,71,133,138]. However, the application of appearance-based gaze estimation methods remains challenging due to the range of the gaze angles and head poses within those. While some datasets have a very large distribution of gaze angles and head poses, gaze estimation models trained on those still tend to be much less accurate when compared to datasets with a much limited distribution [136].

To begin with, the gaze angles are the pitch and yaw angles of the gaze direction vector, which is the normalized vector between the gaze origin, generally defined as either an eye or the middle point between the eyes, and the point of gaze of a user. As for the head pose, it is generally defined as a 3D rotation matrix which describes the rotation of the head of the user with respect to the camera’s reference frame.

As such, as the range of gaze angles and head poses of the dataset’s samples increases, the problem complexity also increases and a learning-based model for gaze estimation will have

to learn a more complex distribution, which generally leads to lower accuracy.

In certain applications, a large gaze angle and head pose range might not be relevant, such as is the case for assistive robotics. Indeed, in these cases, the gaze distribution might be limited to a specific forward facing region with respect to the user. For example, the allowed gaze direction for object selection in an assistive robotics task such as automated grasping might be limited to a box region in front of the user which corresponds to the robot reach [8]. Although the gaze distribution might have a similar range width or variance in this application as most available gaze estimation datasets, the center or mean of the region of gaze might be very different.

In fact, most datasets currently available were created from images taken from a webcam-type camera, generally fixed on a screen, laptop, tablet or phone, while the user is looking at a target shown on the screen or device [1, 71, 133]. As can be seen from the distribution presented in Fig. 5.1, this limits the distribution of gaze angles to regions close to or below the camera. While this is an accurate scenario for gaze estimation models where the goal is to estimate the point of gaze on a screen, it does not translate well to other applications where the user might not be looking directly toward the camera in 3D space.

On the other hand, other datasets [2, 138, 161, 162] have tried to expand the limited distribution of the gaze angles in datasets generated from point on the screen by placing multiple cameras at several positions around the user [2, 161] or by placing the user at random positions [138, 162]. While this might solve the issue of the limited distribution, it remains challenging to train on and can lead to poor accuracy due to the limited number of samples spread over a large range of gaze angles and head poses [136].

To mitigate this issue, we suggest generating a mesh of the detected faces of the users in the dataset and then training a gaze estimation model by using pose transformed version of the face meshes and corresponding gaze directions. This is done through methods similar to what has already been done to augment the span of the gaze angles range [2, 163], but these methods are adapted to instead improve the application-specific gaze estimation accuracy, e.g., assistive robotics tasks such as automated grasping. We also provide a simple way of generating the face mesh without having to use multiple viewpoints of the same sample through using multiple cameras as done in [2].

As later demonstrated in this paper, this allows to reach a much higher application-specific gaze estimation accuracy. We will use an assistive robotic arm controlled via gaze estimation as an example application in this paper. See [8] for more details about the application and refer to Fig 5.2 for a visualization and explanation.

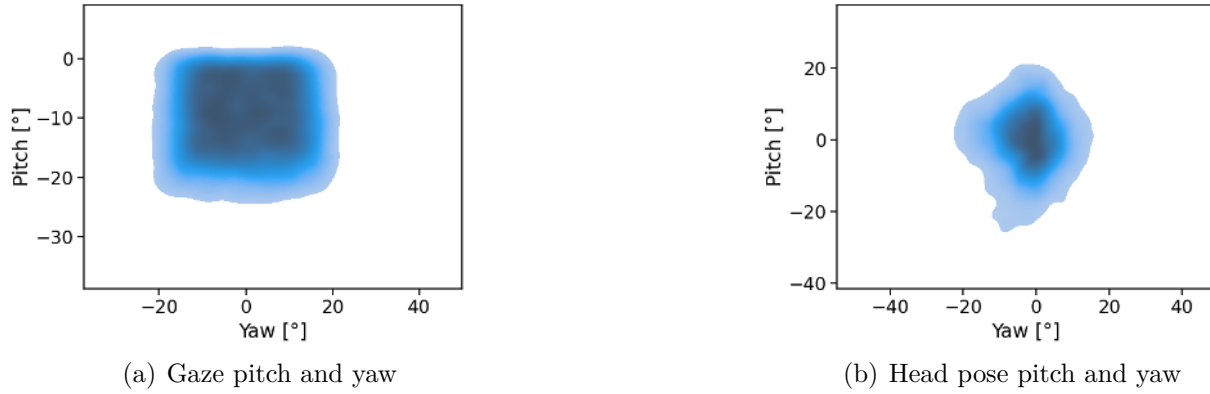


Figure 5.1 Distribution of the gaze direction and head pose pitch and yaw with respect to the camera’s center axis for the MPIIFaceGaze dataset [1].

Note that the gaze distribution is completely skewed toward looking under the camera. While this is representative of a scenario where we try to find the gaze point on a screen with a webcam above it, our sample application would lead to the complete opposite, where the user is looking almost exclusively above the camera.

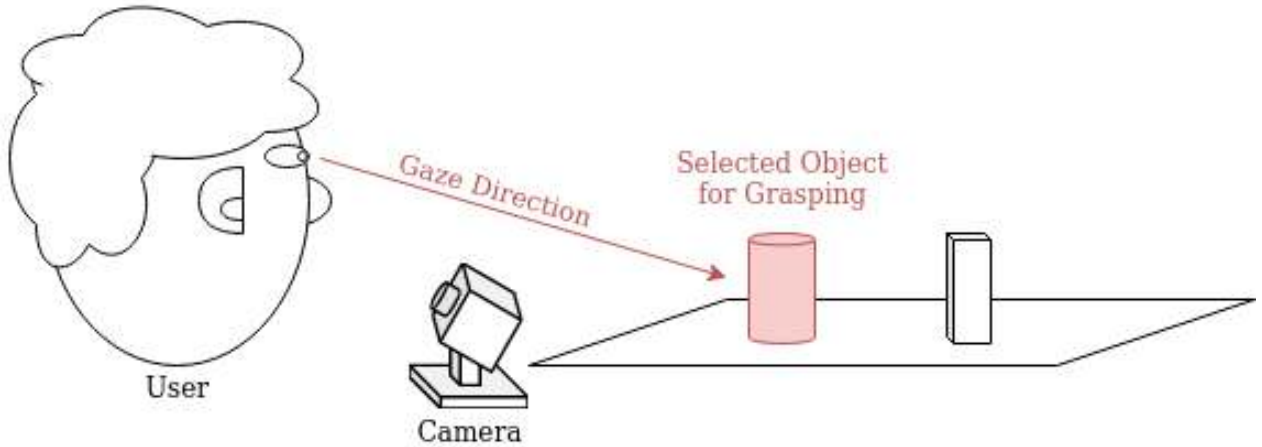


Figure 5.2 Sample application of gaze estimation for object selection for automatic grasping in assistive robotics.

As a sample application, we use the task of object selection to be grasped by an assistive robotic arm. In this scenario, a camera is used to estimate the gaze direction of the user to select an object. Once an object is selected, an assistive robotic arm (not shown) mounted close to the user, e.g. on their motorized wheelchair, moves in to grasp the object. In this application, the camera is mounted in front of the user typically on their motorized wheelchair, but behind the scene ahead and its objects. It is angled upward to look at the user without blocking the user’s view.

5.3 Methodology

The methodology used in this paper follows 8 steps:

1. Align the face of each dataset sample to a face model.
2. Solve for the pose of the face under a perspective projection using the obtained aligned facial landmarks.
3. Generate the textured mesh of the face.
4. Apply the inverse pose transform to the gaze direction to make it correspond with the generated centered face mesh.
5. Compute the sampling distribution given the expected pose of the head and its variance for the application.
6. Apply pose transforms of the head to the samples by sampling from the head pose distribution.
7. Project the face mesh to virtual normalized camera plane for each eye.
8. Use the transformed samples as input for training a gaze estimation model.

Refer to Figure 5.3 for a visualization of the workflow with respect to the modification made to the original image. The following sections will delve into the details of the previously described steps.

5.3.1 Facial Alignment

Using any facial alignment method [164] for which the base face model includes 3D coordinates, align each face of the dataset from the corresponding sample image. In our case, we used the model developed by [93], but any accurate model can do.

After running the face alignment model, a set of 2D points in pixels is obtained. These points correspond to facial features or areas as learned by the facial alignment model. We will call the set of obtained 2D points in screen (pixels) coordinates $\mathbf{X}_s$.

5.3.2 Pose Under Perspective Projection

Because the previously obtained face points $\mathbf{X}_s$ are in 2D screen coordinates, they cannot be directly used to find the pose of the head between the base pose given by the face model

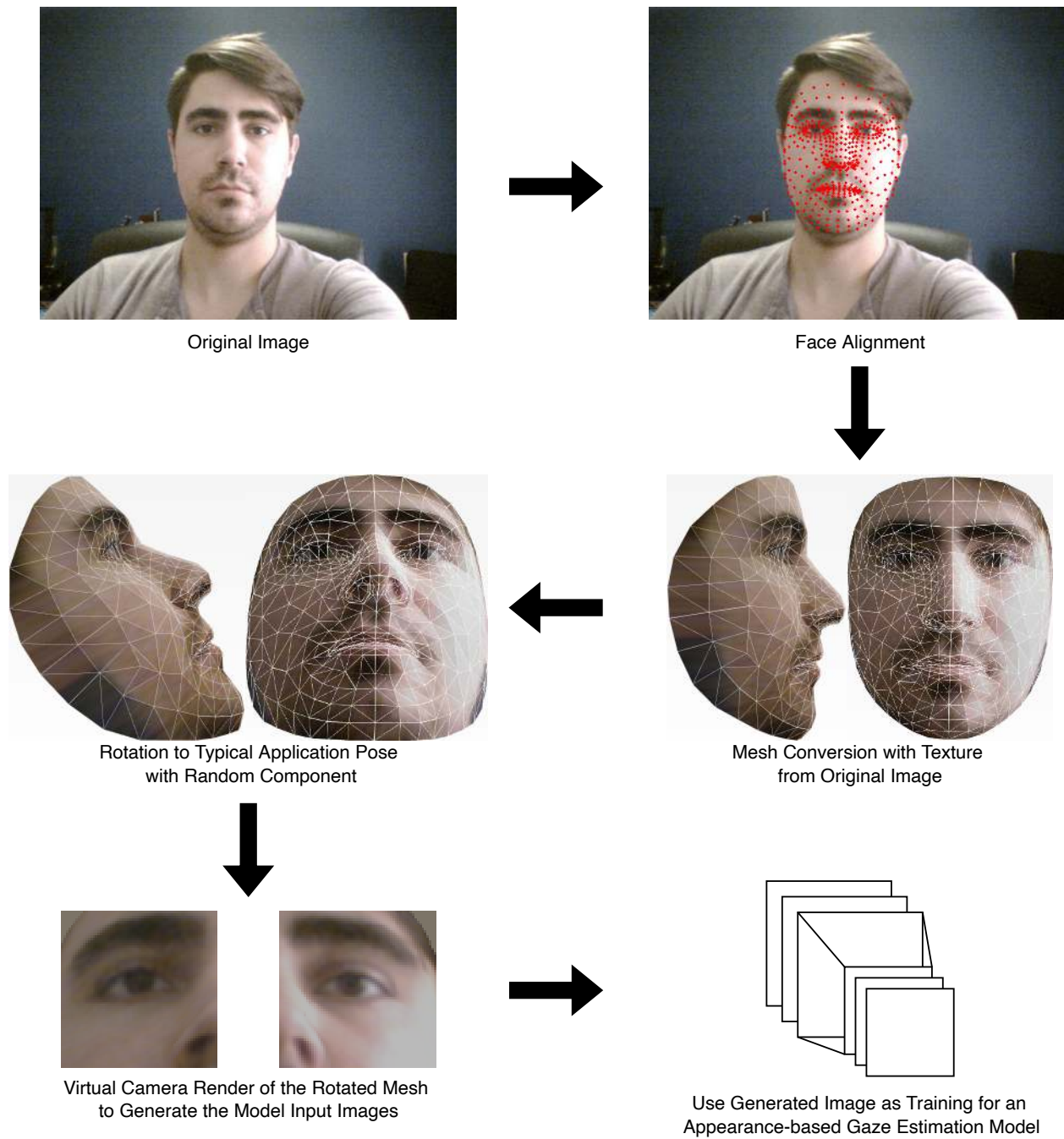


Figure 5.3 Simplified visual workflow of the augmentation method with real example images.

The red points and wireframe on the meshes were added for visualization purposes and are not part of the actual workflow. It should also be noted that the rotation demonstrated has no random component, but instead is a rotation of exactly 30 degrees in pitch and 0 degrees in yaw, which is the mean of the random rotation used in this paper, as presented in Eq. 5.3. The person visible in the image has given their consent to appear in this paper.

and the actual pose in the computed sample. Indeed, algorithms developed to solve the orthogonal Procrustes problem cannot be used as is to solve the problem under projective transformation. For such a case, several methods have been proposed and the most common ones have been implemented under OpenCV's `solvePnP` function [165]. This function finds the best SE(3) transformation (simultaneous rotation and translation in 3D) which, when projected according to the sample's camera matrix and distortion coefficient, minimizes the error between the input face 2D points and the computed face model 3D points under the pose and projection transforms. Simply put, we find the transformation ${}^b\mathbf{T}_a$ which maps the base face model 3D points to a new set of 3D points corresponding to the actual pose of the face in the sample:

$${}^b\mathbf{T}_a = \underset{\mathbf{T} \in \text{SE}(3)}{\operatorname{argmin}} ||\hat{\mathbf{X}}_s - \mathbf{X}_s||^2 = \underset{\mathbf{T} \in \text{SE}(3)}{\operatorname{argmin}} ||\mathbf{C} \mathbf{T} \mathbf{X}_m - \mathbf{X}_s||^2 \quad (5.1)$$

$\hat{\mathbf{X}}_s$ corresponds to the face points obtained by applying the estimated transformation $\mathbf{T}$ to the base face model points $\mathbf{X}_m$ and projecting those using the camera matrix $\mathbf{C}$. It is assumed that the screen-space coordinates $\mathbf{X}_s$ and $\hat{\mathbf{X}}_s$ and the face model 3D coordinates $\mathbf{X}_m$ are expressed in homogenous coordinates, namely $[u, v, 1]$ and $[X, Y, Z, 1]$, and as such have been normalized as to have their last component normalized to 1.

5.3.3 Textured Face Mesh Generation

To generate the textured face mesh, we need:

- A set of 3D vertices.
- A set of mesh faces defined by 3 or more of the previously defined vertices.
- A texture image.
- A (u, v) coordinate on the texture image for each 3D vertex.

It is worth noting that the mesh faces is the name given to the polygonal faces of a mesh, not a human face. These mesh faces are generally triangles.

The set of 3D vertices can be defined as the model face points $\mathbf{X}_m$. The texture image can simply be the input face image used in the alignment step (step 1). When creating the mesh's texture using the face image, the (u, v) texture coordinates to associate to each vertex are $\mathbf{X}_s$, the facial landmark positions in screen space as obtained during the alignment.

The main difficulty here is to generate the faces from the set of 3D vertices if the base face model does not contain any mesh face definitions, which is not the case for the method we used [93]. If the mesh faces definitions are not included, we suggest using surface reconstruction methods such as the Ball-Pivoting Algorithm [166] to generate the set of mesh faces associated with the vertices, which are themselves the facial landmarks.

5.3.4 Inverse Pose Transform Correction to Gaze Direction

As the gaze annotations from the datasets are given with respect to the actual pose of the face and not the centered, base face model pose, we need to correct the gaze direction vector using the previously computed pose. Given the actual gaze direction as per the dataset annotation $\vec{g}_a$, the centered base gaze direction $\vec{g}_b$ can be defined as:

$$\vec{g}_b = {}^b\mathbf{T}_a^{-1} \vec{g}_a \quad (5.2)$$

5.3.5 Head Pose Distribution

Given the application that the gaze estimation model will be used for, we need to define the expected pose of the user's head. In the case of our example case, we know the camera is mounted in front of and below the user, aiming 30 degrees up. We should thus expect the user's head to be centered on the image, but with a head pose pitch of 30 degrees up. In other words, the user is looking forward while the camera, positioned in front of the user but below them, looks up at the user's face with an angle of 30 degrees.

As this is the perfect case and not actually reflective of the reality, we will assume a variance of 10 degrees in both the head pitch and yaw. We thus define a bivariate normal distribution giving the head pitch p and yaw y as:

$$\begin{pmatrix} y \\ p \end{pmatrix} \sim N \left(\begin{pmatrix} \mu_y = 0 \\ \mu_p = 30 \end{pmatrix}, \begin{pmatrix} \sigma_y^2 = 10 & 0 \\ 0 & \sigma_p^2 = 10 \end{pmatrix} \right) \quad (5.3)$$

The μ parameters are the means whereas the σ^2 parameters are the variances. This distribution will be sampled when generating the augmented dataset to define the augmented head pose for each sample.

It should be noted in this case that the roll angle is not computed, as the normalization procedure that is generally used in gaze estimation models removes the roll component [2,148]. As such, there is no need to compute a roll just to remove it when normalizing the images.

5.3.6 Sample Pose Transform

By sampling the yaw and pitch from the previous distribution for each sample, we can calculate a rotation matrix to apply to the face mesh and gaze direction annotation. This computed rotation matrix will be the new augmented head pose, and thus we rotate the base face model and gaze direction using this rotation matrix.

We define the rotation $\mathbf{R}(y, p)$ around the pitch and yaw as:

$$\mathbf{R}(y, p) = \mathbf{R}_{\text{pitch}}(p) \mathbf{R}_{\text{yaw}}(y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(p) & -\sin(p) \\ 0 & \sin(p) & \cos(p) \end{bmatrix} \begin{bmatrix} \cos(y) & 0 & \sin(y) \\ 0 & 1 & 0 \\ -\sin(y) & 0 & \cos(y) \end{bmatrix} \quad (5.4)$$

We then apply this rotation to the face mesh points $\mathbf{X}_m$ and the gaze direction $\vec{g}$ to generate our augmented sample from the dataset.

$$\mathbf{X}_m' = \mathbf{R}(y, p) \mathbf{X}_m \quad (5.5)$$

$$\vec{g}' = \mathbf{R}(y, p) \vec{g} \quad (5.6)$$

5.3.7 Mesh Projection and Normalization

Using any rendering method, we can render the face mesh as an image, but to do so requires knowing the camera position and parameters, i.e., the field of view or focal length and the image pixel size. To simplify the workflow, we merge the normalization procedure often done as a preprocess step in gaze estimation [2, 148] with the rendering of the face mesh.

Indeed, because we want our eye patch images to be normalized in distance, focal length and image pixel size, we use the normalized parameters to create a virtual camera viewpoint.

For a given eye (left or right) and assuming a normalized distance d_n and a normalized focal length f_n , we create a virtual camera situated at a z distance of d_n from the eye position on the transformed face mesh $e' = [e'_x, e'_y, e'_z]$ with focal length f_n (see Figure 5.4 for an example geometric construction). As such, the pose of the camera would point toward the position z axis and its origin o_c would be situated at:

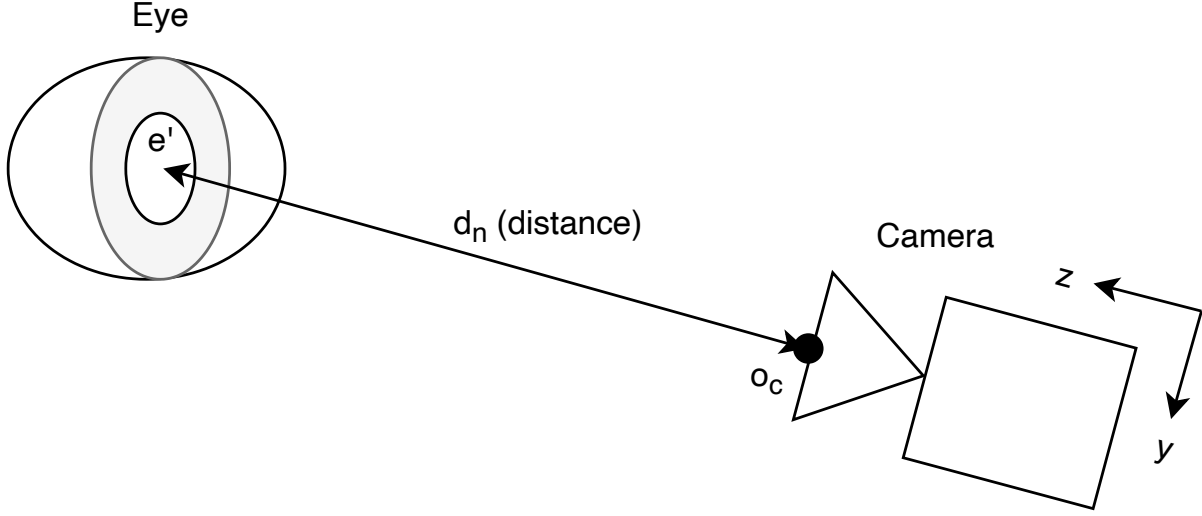


Figure 5.4 Geometric construction example of the virtual camera and the augmented eye position.

$$o_c = [e'_x, e'_y, e'_z - d_n] \quad (5.7)$$

Since the rendering scene is now composed of the transformed face mesh $\mathbf{X}'_{\mathbf{m}}$ and the virtual camera with focal length f_n , origin o_c and aimed toward the positive z axis, we can generate an eye patch image with the normalized image size required by our gaze estimation model. This can be done through any rendering tool, but we used Pyrender [167] to do so.

This rendering part should be done on the left eye and right eye if both are required as input for the gaze estimation model.

5.3.8 Gaze Estimation Model Training

Once every sample in the dataset has been augmented using the previous steps, we can use the samples as input to a gaze estimation model. In our case, we use the eye image patches (left and right) and the head pose $\mathbf{R}(y, p)$ as input to our model and we use the transform gaze direction $\vec{g}'$ to compute the loss.

When evaluating the model or using it for inference, we simply use the normalization method described in [148] to generate the preprocessed input data and we denormalize the output gaze using the method described in the same paper. This has to be done because we do not generate a face mesh at evaluation time, but we use the actual image from the camera. Given that the training was done in a normalized space with the camera at a distance d_n (in our case 600 mm) and a normalized focal length f_n (in our case 650 px/mm), we must normalize

the images to that space before using them as input to our trained model.

5.4 Results and Discussion

To analyse the performance of our dataset augmentation method, we trained a gaze estimation model based on a MobileNet architecture [168] on UTMultiview [2], MPIIFaceGaze [1] and GazeCapture [133] with and without our augmentation method. Training for all datasets is done with the same hyperparameters that were tuned for the best accuracy *before augmentation*. This should ensure that the improvement comes from the augmentation and not the choice of hyperparameters.

After training our model, we then evaluate it on a series of 69 samples from 3 different users taken on the setup as described by the example application (see Fig. 5.2). The results are detailed in Table 5.1. The error values e are given by computing the angle between the true gaze direction $\vec{g}$ and the estimated gaze direction $\vec{g}_e$:

$$e = \cos^{-1} \frac{|\vec{g} \cdot \vec{g}_e|}{\|\vec{g}\| \|\vec{g}_e\|} \quad (5.8)$$

Table 5.1 Comparative results in gaze errors when trained on the original and augmented datasets

Training Dataset	Original	Augmented	Reduction (%)
UTMultiview*	9.2°	6.9°	25%
MPIIFaceGaze	9.5°	3.0°	68%
GazeCapture	6.0°	3.0°	49%

*The UTMultiview dataset does not include full face images and thus it is not possible to use our own mesh generation technique and we therefore rely on the existing meshes contained in the dataset.

As can be seen from Table 5.1, the angular error decreases by 47% on average. This is a significant increase in accuracy and leads to more consistent results in our sample application. This increase in accuracy does not significantly increase training time since it can be preprocessed once and trained on multiple times afterward without having to go through the augmentation steps again. It does not increase the development time significantly either since it has a very small amount of parameters to tune for, namely the bivariate normal distribution’s means and variance, and the fact that they can be easily inferred from the application rather than a costly hyperparameter search.

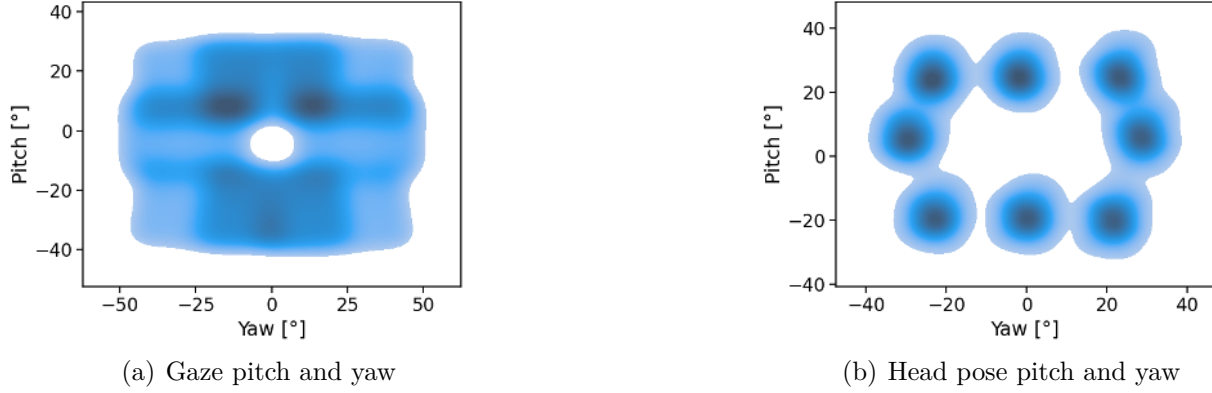


Figure 5.5 Distribution of the gaze direction and head pose pitch and yaw with respect to the camera's center axis for the UTMultiview dataset [2].

Note that the gaze distribution in this case also includes sample where the user is looking above the camera, which is closer to our example application's average gaze direction.

The case of the UTMultiview [2] dataset should also be noted, since the error reduction is lower for this dataset. Indeed, because the dataset does not include the full face in the images, it is thus not possible to generate the face mesh from our facial alignment method, which is also used in the evaluation step. Therefore, the base face pose used as the "zero" pose is not the same for both models. This leads to a discrepancy between the training data's annotations distribution and the evaluation's distribution.

Also, as can be seen from Fig 5.5, the distribution of the gaze angles for the dataset also includes the user looking above the camera, which is not the case for MPIIFaceGaze [1] and GazeCapture [133]. While the mean of the distribution might not coincide with the average gaze direction of the evaluation using our example application, it still includes more samples relevant to our application and thus it is to be expected that the UTMultiview-trained model will perform similarly with and without augmentation.

5.5 Conclusion

In conclusion, the developed augmentation method can be done quickly and leads to a significant (47%) decrease in gaze estimation error in gaze estimation for assistive robotics tests, where three different users were asked to look at an object randomly placed in front of them over 69 samples. It does not require a costly hyperparameter search and can be used on existing datasets to adapt them to application-specific usages.

CHAPTER 6 ARTICLE 3: UNCONSTRAINED, CALIBRATION-FREE, FREE-VIEWING AND NONINTRUSIVE GAZE ESTIMATION-BASED CONTROL SYSTEM FOR ASSISTIVE ROBOTIC ARMS

Cedric Leblond-Menard, Sofiane Achiche

Submitted to Robotics and Autonomous Systems journal on March 2nd, 2023. ROBOT-D-23-00163

6.1 Abstract

Assistive robotic arms can significantly improve the quality of life of their users by increasing their autonomy. However, it is often discussed in the literature that the human-machine interfaces used are often a bottleneck to the usage of such assistive devices. Indeed, most commercially available assistive robotic arms are shipped with joystick-like interfaces to be controlled by the users hand, feet, mouth, breath etc. Given that the vision remains functional in even the most severe cases of muscular dystrophy, gaze-based control methods are often suggested in the literature. In this paper, we suggest a novel system that overcomes most of the limitations of such systems described in the literature while being affordable and with low power requirements, to be mounted on the motorized wheelchairs of potential users. Trials with 30 typically developed participants showed a decrease in time to accomplish a grasping action of 72.0% with a success rate of 90% .

6.2 Introduction

Assistive robotic manipulators (ARM) have seen their use increase in recent years due to increase in wellness and autonomy of people with disabilities such as muscular dystrophies [15–19]. Indeed, these robots, typically serial manipulators, are generally used to accomplish daily tasks such as grasping objects, opening doors and eating and drinking.

To do so, commercial ARMs generally are equipped with joystick-like devices as control interfaces [11–13, 130]. Since the robot can move in a six-dimensional space (3 translations and 3 rotations), the joystick which most often has 3 degrees of freedom (forward-back, left-right and twist) has to have multiple modes to allow the user to access the complete range of motion of the robot [16, 17].

As such, the joystick-based human-machine interface (HMI) can be limiting, especially to

users with low dexterity such as people with muscular dystrophy or cerebral palsy. Indeed, special filters and control algorithms have to be used to prevent involuntary, undesirable and potentially harmful movements of the robot due to spasms for example [169].

Additionally, some disabilities require other means of control when the user is not able to use the joystick, as it is the case for people with partial or complete upper limb paralysis, or simply not having the required dexterity level. In this case, several HMIs have been developed over the years, including tongue or chin controlled joysticks, voice activated interfaces and head controlled switches [16, 17, 23, 170].

Even with these new HMIs, there are still several issues highlighted in research works that require further development, especially of vision-based systems which can lead to drastic improvements in user loading in pick-and-place and grasping tasks [16, 17, 171]. Indeed, these HMIs still require the user to control the robot in its 6 dimensional motion space and thus are complex to use. They are also limited by the speed of the user sending motion commands through the HMI instead of automating the movement of the robot through object detection for example.

Given the limitations of joystick-based interfaces and other currently used HMIs and considering the fact that the vision of people suffering from even severe muscular dystrophies or tetraplegia remains functional [5, 6], multiple gaze estimation-based control systems for ARAs have been proposed as a possible solution [7–10]. Gaze estimation-based methods could thus allow people for whom no other HMIs are available, such as individuals suffering from tetraplegia, to use ARAs.

Therefore, the work described in this article builds upon previous eye tracking and gaze estimation control methods presented in [7, 8]. These have shown the feasibility of such systems and also demonstrated a significant reduction in the time needed to accomplish grasping tasks. In such systems, the user looks at an object and actuates a switch, after which it finds the gaze direction and position of the user and infers the point of gaze. This point is then used as the goal point for an ARM to accomplish a grasping motion.

The work described in [8] was accomplished in a way to maintain the overall cost of such system below 400\$, excluding the price of the required computer, and using non-intrusive gaze estimation techniques to improve accessibility. They achieved a 90% success rate and an average time to grasp of 17.3 seconds on trials with a single participant and with up to 3 objects in a single fixed position per object.

However, some issues reported by [8] have yet to be solved, including the ability to use the system in an outdoor setting, where infrared-based pupil center corneal reflection [52] such

as the eye tracker used in their proof of concept are unreliable due to the natural infrared illumination. This also includes indoor settings where the natural light of the sun is partly present, e.g., close to a window.

Also, the 3D scene reconstruction in [8] was prone to failure due to the absence of object detection or scene segmentation algorithms and due to the usage of a standard stereovision system consisting of two visible light grayscale cameras. These cameras were prone to block matching failures which lead to poor reconstruction results with low contrast surfaces such as tables and walls.

Finally, as is the case for most gaze estimation systems commercially available today, the proof of concept [8] required extensive calibration for each individual participant and for specific lighting conditions. This was not practical and as such the removal of calibration procedures, especially ones that are person and scene specific. This is in part because most commercial pupil center corneal reflection eye trackers generally only output 2D gaze point position on a screen at a fixed distance from the user, which leads to poor compensation of head movements and a complex calibration process which has to be repeated for every user and for different head positions.

6.2.1 Objective

Given that the system presented in this paper builds upon the proof of concept presented in [8] as previously stated, the objective is:

Develop an embedded gaze estimation-based solution for controlling assistive robotic arms for people with upper limb disabilities.

To ensure the solution is usable for daily living activities, several design constraints were established. Indeed, the solution should:

- be nonintrusive;
- be affordable;
- not require person specific calibration;
- function in unconstrained environment, including indoors and outdoors;
- detect and segment objects to allow adaptable and precise grasping;
- have low latency (real-time);
- be easy to use;

- avoid collision with the environment and with itself.

Validation of the objective should be performed through real-world trials with multiple participants.

6.2.2 Contributions

The contributions brought forward in this paper are as follows:

- A real-time open-source gaze estimation-based HMI for ARM including gaze tracking and augmentation, scene reconstruction, real-time path planning, object detection, scene segmentation and adapted user interface for multiple control modes. This system works in both indoors and outdoors scenarios and importantly does not require any person or scene specific calibration.
- A validation of the improvements in grasping tasks execution of the system via real world trials with 30 participants.
- A workflow which allows for easy integration of other ARM and on which to build other HMIs such as face tracking and automatic motions such as bringing an object close to the user's mouth.

6.3 Related Work

As introduced earlier, researchers have been developing alternative HMI for ARM to facilitate the use of robots with both the current users and potential users. Indeed, as was discussed earlier, some disabilities completely prohibit the user of joystick-based HMIs. The main focus in the literature seems to be on vision-based systems, where the ARM control interface is augmented and made smarter through the use of computer vision [171].

Most of these vision-based systems either often rely on a screen interface [36,37,46–49] using click or touch controls [36,37] or gaze estimation (also known as eye tracking) [46–49].

Considering that a screen-based HMI divert the user's attention towards the screen, narrowing their vision, and can be hard to look at for some due to the position of the screen off of the scene [10], researchers have also brought forward vision-based methods that do not rely on screens, for example using electroencephalogram (EEG) [172,173] or gaze estimation [8–10].

The usage of EEG for robotic control generally relies on classification of intentions to either select between a set of predetermined objects [172] or specific motion actions for the robot

which are augmented by computer vision [173]. Unfortunately, the accuracy of non-invasive EEG is low and it is prone to interferences while invasive EEG still poses high risk due to the surgical procedures and is therefore not easy to adopt [9].

On the other hand, head-free gaze estimation has shown great promise through the proof-of-concepts and preliminary trials [8–10]. Indeed, high success rates in the limited trials and high user satisfaction is reported [9, 10].

These methods generally rely on commercially available eye trackers which can be expensive (3,500 to 10,000 \$) [9, 10], require extensive calibration and setup especially for cheaper eye trackers [8, 9] and do not work reliably outdoors due the use of near-infrared illumination [8–10] which is drown out by sunlight in outdoor settings [8, 52].

Also, [9, 10] use commercially available eye tracking glasses which can be intrusive since they require the user to wear bulky glasses, especially when a 3D camera is mounted on them as is the case for [9]. They also often require long planning time for realtime use (several seconds) [9, 10] and the computation is done on a full desktop or laptop computer, which are both bulky and use a lot of power which is not particularly adapted to embedded use on for a wheelchair-mounted ARMs.

Furthermore, [9] also requires the application of specific tags on the table used for grasping objects, which is not adapted to most scenarios of activities of daily living as would be the case if such system was used in a real-world setting.

Finally, both [9, 10] do not implement any grasping method, instead relying on a magnetic attachment to pick up select objects, which also does not translate well to a real-world setting.

As such, there is a clear lack of nonintrusive and calibration-free gaze estimation control systems for ARMs which can be used in unconstrained environments without relying on screen-based or pupil center corneal reflection gaze trackers. Such a development would bring the gaze estimation-based control systems closer to being available for use in real-world scenarios and thus allow both current users of ARMs and users that are unable to currently fully benefit from ARMs due to their disability.

6.4 Methodology

To accomplish the aforementioned objectives, we designed a complete system capable of:

1. Estimating the gaze direction and face position of a user using a simple color webcam. (Section 6.4.1)
2. Reconstructing the scene in front of the user from a color and depth camera. (Section

- 6.4.2)
3. Detecting graspable objects in front of the user from a color and depth camera. (Section 6.4.3)
 4. Selecting the object to grasp from the detected objects and the estimated gaze information of the user. (Section 6.4.4)
 5. Planning a collision-free path to grasp an object from the reconstructed scene and selected object. (Section 6.4.5)

The inputs of the complete system are:

- A color image from a user-facing webcam (face camera).
- A color and depth image from a scene-facing depth camera (depth camera).
- The joint-space configuration (joint angle values) provided by the connected ARM.

The expected output is a list of joint-space configurations which, when sent to the robot, allows a collision-free motion from the initial position to the final position before grasping. A simple grasping motion is then executed to reach and grasp the desired object.

These capabilities are defined and explained in the following sections in the same order. Section 6.4.7 describes the control flow and HMI allowing a user to select objects and manipulate them. Figure 6.15 describes the interactions between each process of the system and can be referred to as needed.

Hardware and software setup The system hardware setup as can be seen on Figure 6.1 is composed of 5 distinct parts, namely:

- A Logitech C920 color user-facing camera (webcam).
- An Intel Realsense [26] D415 color and depth camera, capable of computing depth information with or without its infrared illuminator.
- A Kinova [130] JACO assistive robotic manipulator controlled through its USB interface.
- A rigid metal frame on which all the cameras and the robot are mounted, to simulate how the system would be mounted on a motorized wheelchair.

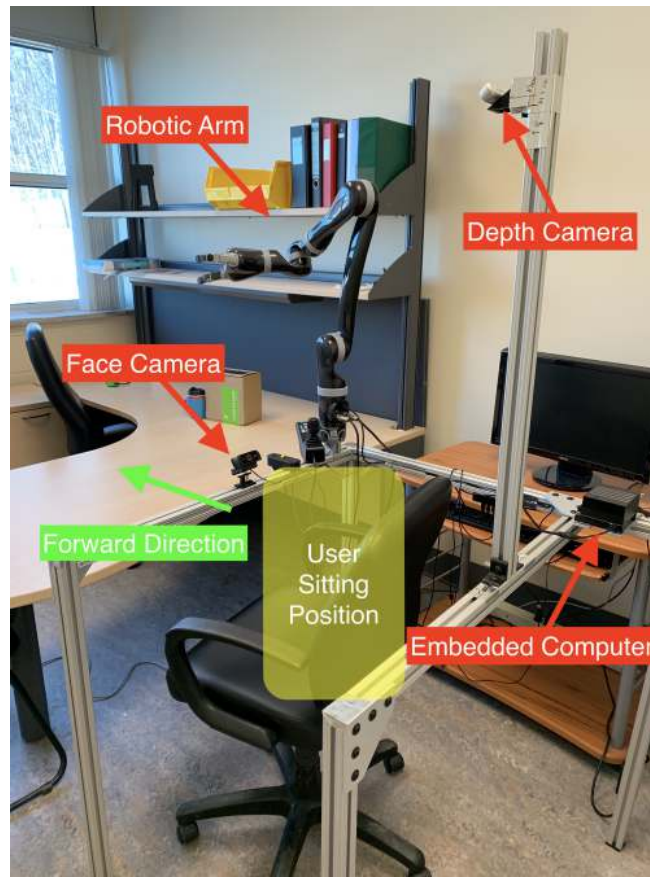


Figure 6.1 Photo of the complete system setup with annotated parts. Note that the screen and orange desk are not part of the system, but only used for debugging purposes.

- A Nvidia Jetson AGX Xavier development kit as the embedded computer used for all the required processing and on which the cameras and the robot are connected to.

It should be noted that no desktop computer is used in this system, as it is intended to be mounted on a motorized wheelchair. While the form factor benefit is a definite plus, the main reason to use an embedded computer in this case is the power consumption. Indeed, a typical desktop computer use around 500 Watts while the Jetson AGX Xavier uses at most 30 Watts, which is a clear benefit when using the wheelchair's onboard battery.

As for the software setup, most of the software written was done in Python using available high performance libraries such as Pytorch [152] and OpenCV [82]. Part of the code is written in C++ when necessary, but interfaced with Python. While this allows for a faster prototyping while retaining some performance, there is definitely space for improvement by using a higher performing language such as C++. The most critical places where the performance bottleneck of using Python is an issue are in the path planning and will be discussed in Section 6.4.5, but they are considered a limitation of this work.

Reference frames Several reference frames are used throughout the system workflow, which are depicted by Figure 6.2. To begin with, the *world* reference frame T_w is equivalent to the robot’s own reference frame, centered at the robot’s base with its x axis going toward the left when looking at the robot from its back, its y axis going back and its z axis going up.

Also, there is the reference frame proper to the *face camera* facing the user T_c , centered at its lens position with its x axis pointing towards the right of the camera when positioning ourselves as if looking through its lens, its y axis going down and its z axis aligned with the optical axis of the camera, going toward the user.

Finally, there is the reference frame proper to the *depth camera* T_d facing the scene. While this camera has several reference frames used internally since it contains 3 separate imaging sensors and an infrared dot illuminator, the reference frame used to align all images provided by the camera is the depth sensor’s. As such, in the case of our system, the depth sensor’s reference frame is used as the reference frame corresponding to the depth camera.

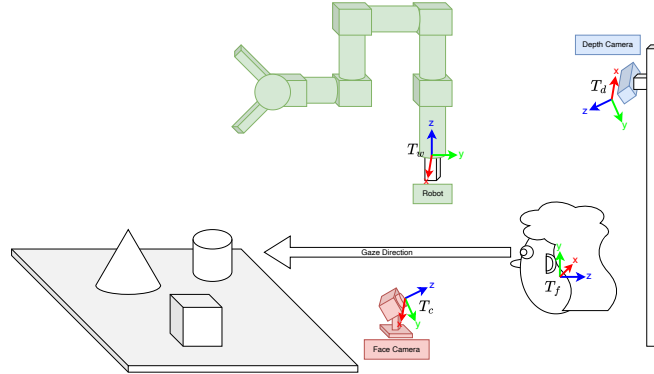


Figure 6.2 Approximate depiction of the reference frames of the different sensors and the robot.

6.4.1 Gaze Estimation

The gaze estimation part of the system takes as input an image from the color camera facing the user (webcam) and outputs the gaze direction and origin in the T_w reference frame. To do so, we use a novel fast gaze estimation workflow as described in [168], which relies on a face alignment convolutional neural network (CNN) to estimate the user’s face pose and a specifically trained gaze estimation CNN to output the gaze direction and origin, the latter of which is defined as the center point between the eyes. While the implementation details of this gaze estimation workflow are out of the scope of this paper, the details are available in [168].

However, to achieve a lower gaze estimation error in the case of this system, we trained the gaze estimation model using the method detailed in [174], which augments the available gaze estimation datasets by creating 3D meshes of the faces and rotating them to fit the expected pose of the user, while adding a random component to the rotation to avoid overfitting.

To train the gaze estimation model using the augmentation method described in [174], we used the MPIIFaceGaze dataset [1] which contains about 30,000 full face images of 15 different participants in several lighting conditions.

Finally, we added three steps to the gaze estimation workflow to improve the reliability of the gaze estimation by using multiple samples over time.

Online calibration

Initial tests of the gaze estimation part of the system showed that using person-specific calibration improves the results. Even a simple polynomial fitting as a person-specific calibration procedure can lead to significant improvements in gaze-estimation with less than 10 samples [72].

On the other hand, the system can greatly benefit from never requiring re-calibration, even if the user changes, or changes their appearance, e.g., growing a beard. Therefore, an online calibration procedure was implemented where each gaze estimation sample is added to a first-in, first-out list if the complete gaze estimation system was successful, i.e., the user confirmed that the object selected by the system was indeed the object they were looking at.

In the case where the selection by the system was successful, the selected object's centroid is used to compute the actual gaze direction $\vec{g}$ of the user, and added to the first-in, first-out list together with the initially estimated gaze direction $\hat{\vec{g}}$. We then define two rotation matrices given angles $\mathbf{R}_x(\theta)$ and $\mathbf{R}_y(\phi)$ about the *world* x and y axis with angles θ and ϕ .

Using a least-square regression fit, we find the angles θ and ϕ that minimize the following loss equation:

$$L_{calibration} = ||\vec{g} - \mathbf{R}_x(\theta)\mathbf{R}_y(\phi)\hat{\vec{g}}||^2 \quad (6.1)$$

Essentially, we find the best rotation matrices around the x and y axis that correct for any constant angular offset in the gaze estimation workflow.

To prevent possible outliers from completely throwing off the gaze estimation, we set the minimum number of samples to run the calibration to 4. As such, the angles θ and ϕ are set to 0 as long as the system has not reached at least 4 successful gaze estimation samples. This number was chosen by experience, being large enough to not update the calibration with a

potential outlier or user error, and small enough as to improve the gaze estimation results faster.

Finally, to allow the calibration to update more easily with changing conditions, we limited the first-in, first-out list to 10 samples, meaning any successful gaze estimation sample that has been obtained before the last 10 samples is not used for the online calibration. This generally translates into a decrease in gaze estimation error from 5% to 9%, as can be seen from Table 6.1.

Outlier rejection

Since the eyes can move quickly between sampled images and because the gaze estimation workflow might lead to large errors, for example due to motion blur, an outlier rejection method is needed. This is especially true if averaging over several samples, where an outlier can skew the average.

Although several methods are available in the literature, we selected a method that works reliably on small sample sets, which is our case. The method in [175] describes a modified Z-score called M using the median and median absolute deviation instead of the average and standard deviation respectively.

$$M = \frac{0.6754(x_i - \tilde{x})}{MAD} \quad (6.2)$$

x_i is the sample value, $\tilde{x}$ is the sample set median and MAD is the mean absolute deviation of the sample set. It is suggested in [175] that any sample with an M value above 3.5 is considered an outlier, which is the outlier rejection criterion used in this paper.

Variance weighted averaging

To further increase the accuracy of the gaze estimation one can use variance weighted averaging. That is, if the gaze estimation model outputs an estimated variance or uncertainty as well as the gaze direction, one can average over multiple samples and use the inverse of the estimated variance of each sample as a weight.

In doing so, the higher the variance, the lower the weight the sample has when computing the average. Essentially, the samples for which the gaze estimation model outputs a higher confidence level will have a larger impact on the resulting average.

The gaze estimation model described in [168] does not output an estimated variance level, and hence it had to be modified. The modifications are:

Table 6.1 Gaze estimation error decrease when using person-specific calibration compared to no calibration

# of calibration samples	Error decrease
4	5%
10	9%

The values are obtained using 90 gaze estimation samples obtained from 4 separate persons.

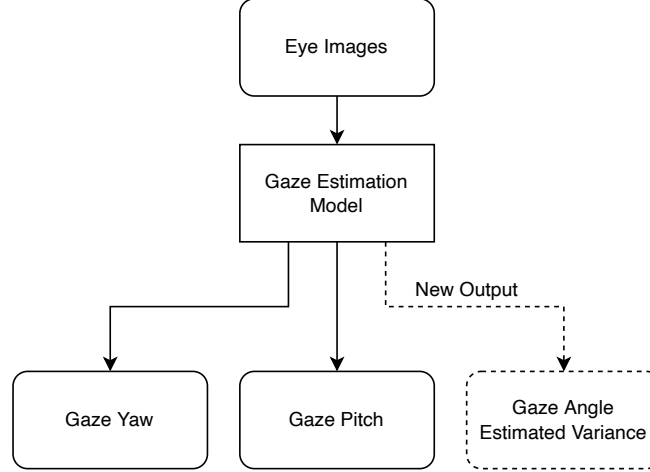


Figure 6.3 Added variance output dimension to the gaze estimation model.

- Add a third dimension that corresponds to the estimated variance besides the yaw and pitch gaze angles as output (see Figure 6.3).
- Use a loss function which includes the estimated variance term, such as the variance attenuation loss [176].

As such, the added third output dimension of the gaze estimation model is taken as the estimated variance $\hat{\sigma}^2(\mathbf{x})$ and the following loss expression is used during training:

$$L_{\text{variance}} = \frac{\log \hat{\sigma}^2(\mathbf{x})}{2} + \frac{(\hat{y} - y)^2}{2\hat{\sigma}^2(\mathbf{x})} \quad (6.3)$$

where $\mathbf{x}$ is a training input sample (eye images), y is the sample label (expected gaze angles) and $\hat{y}$ is the estimated sample label (estimated gaze angles). This loss term contains the mean squared error (MSE) term $(\hat{y} - y)^2$ to ensure the model learns the representation of the gaze angles y . The log of the estimated variance term will decrease with the decrease of the variance, but the variance term that divides the MSE will increase. Therefore, the bigger the MSE, the higher the variance has to be to lower the loss, whereas if the MSE is decreasing, the variance term is decreasing as well. The derivation of this loss and more details are given in [176].

Given a set of samples (eye images) taken one after the other x_i for $i = 1, 2, \dots, n$, we can compute their respective estimated gaze angles $\hat{y}_i$ and estimated variance $\hat{\sigma}^2(x_i)$ using our trained model. From that information, we can compute the following weighted average:

$$\hat{y}^* = \frac{\sum_{i=1}^n \hat{\sigma}^2(x_i)^{-1} \hat{y}_i}{\sum_{i=1}^n \hat{\sigma}^2(x_i)^{-1}} \quad (6.4)$$

These new averaged gaze angles $\hat{y}^*$ are then output by the gaze estimation part of the assistive solution.

6.4.2 Scene Reconstruction

The input of the scene reconstruction is a depth image and the output is a point cloud that will be used for object detection and selection as well as collision avoidance.

To infer potential objects to grasp from the scene as well as avoid collisions with the environment when moving the robot, the scene must be reconstructed in 3D. This is done through a depth camera such as the Intel Realsense D415 [26] used in this paper.

The selection of the Realsense D415 camera instead of other technologies such as LiDAR, also known as time-of-flight depth sensors, or structured light sensors was made due to its use of stereovision depth estimation. Indeed, it uses two near-infrared cameras and block-matching techniques to obtain the depth, relying mostly on the image sensors and not the infrared projector. In this case, the infrared projector is used to project a texture onto the scene, reducing the amount of ambiguity in the block-matching algorithm due to low-texture parts of the image.

The Realsense D415 camera was chosen because it is affordable (200-300 USD) and because of its convenient use of both stereovision and structured light via an infrared projector. The projector is used to project a texture onto the scene which improves the block-matching depth reconstruction. This is an important distinction with respect to the other depth imaging technologies, because it allows the camera to work with direct sunlight illumination. Indeed, even when the infrared projector's pattern is not visible due to infrared sunlight illumination, the camera can still provide accurate depth information by relying on the natural texture of the scene.

As obtaining 3D coordinates from a depth image is well defined in the literature and is directly available in the Intel Realsense's software development kit [177], the details of this procedure will not be covered in this paper. Essentially, the depth image is converted to a 3D point cloud, where each pixel of the depth image with a valid depth value is transformed

to a 3D point in the point cloud.

To decrease the computation time and thus the latency, we downsample the pointcloud using a voxel approach, in which we essentially group points within a cube voxel of 1 centimeter in size into a single point. From that smaller point cloud, we use the robot's current position, its forward kinematics definition and its collision bounding boxes to find which points are within the collision bounding boxes. These points are most likely parts of the ARM that is visible in the depth sensor's field of view and should be removed from the point cloud as they are invalid both for object detection and collision detection. Indeed, the collision checks are computed with respect to a virtual model of the robot against the real-world scene, and 3D points from the real-world scene that belong to the robot would always be in collision with the virtual model.

Indeed, we first define a link i of the n links of the ARM with the bounding box bounds $B_{i,min}$ and $B_{i,max}$, the forward kinematics transformation matrix ${}^WT_i(q)$ which describes the transformation from the reference frame of the link to the *world* reference frame as a function of q , the robot's current joint configuration and p a reconstructed point in *world* reference frame to test if it is part of the robot. We then compute the following algorithm for points of the pointcloud:

Algorithm 1 Algorithm to check if a point belongs to the robot collision bounding boxes.

```

 $i \leftarrow 0$ 
while  $i < n$  do
   ${}^ip = {}^WT_i(q) p$ 
  if any( ${}^ip > B_{i,min}$ ) then
    if any( ${}^ip < B_{i,max}$ ) then
      remove  $p$  from point cloud
    end if
  end if
   $i \leftarrow i + 1$ 
end while

```

The resulting downsampled and filtered point cloud is then used for the object detection and selection and collision avoidance during path planning.

6.4.3 Object Detection

The object detection part receives as input both the filtered point cloud from the 3D reconstruction and the color and depth images from the depth camera. The output is a list of graspable object centers and their bounding boxes. These objects will be used for the

object selection workflow using the estimated gaze of the user.

There are two separate methods used for the object detection that are run concurrently. The first method uses a horizontal flat surface in front of the user, e.g., a table. This method is very fast but relies on the presence of such a surface, making it less adaptable.

While the first method is processed, a separate, deep learning-based method is run. A Mask R-CNN instance segmentation [99] model trained on the LVIS instance segmentation dataset [104] is used on the color image from the depth camera to segment objects from the scene. While this second method does not rely on the presence of a table or flat surface, it is more computationally expensive, especially on an embedded platform. While the flat surface-based method takes less than half a second to process, the deep learning-based method takes over 2 seconds.

To reduce the system latency, we first start the deep learning-based method on a background thread and use the main thread to compute the flat surface-based method. If the segmented surface is either too far from the robot or does not represent a significant fraction of the points of the point cloud, then we wait for the deep learning-based method to finish and use its results for the object selection part.

If the detected flat surface is within a threshold distance of the robot's base and the fraction of points belonging to the surface is significant, we stop the background thread and use the results of the first method. The following sections will detail each method.

Flat surface-based method

This method uses the filtered point cloud obtained from the reconstruction step as input. The assumptions for this method is that the objects to grab with the ARM lay on a flat horizontal surface in front of the user.

Given this assumption, we use a RANSAC-based plane iterative plane fitting algorithm available in Open3D [178]. This allows us to find the plane equation that fits best the dominant plane of the point cloud. Knowing that plane equation, we can easily find which points lie above it. A point is determined to be above the plane if it satisfies the following inequality:

$$ax + by + cz + d > -l \quad (6.5)$$

a , b , c and d are the plane equation parameters, x , y and z are the components of the point being checked and l is the distance above the plane. If we simply want to know if the point

is above or below the plane, we use $l = 0$. Since doing so also includes a large part of the point belonging to the plane itself, we use $l = -0.04$ meters, meaning a point is considered above the plane if it is 4 centimeters or more above the plane. This filters out all the points belonging to the plane.

The remaining points belong to potential objects. To define the objects, we must cluster these points so that the ones that are close to each other are defined as belonging to the same object. We use the DBSCAN algorithm [179] to obtain distinct clusters of points, for which we compute the axis-aligned bounding boxes and find their geometric center.

To ensure that we do not cluster points that are irrelevant, we set the minimum number of points to define an object using DBSCAN at 15 points and we use a maximum distance between points of an object as 3 centimeters.

Also, to detect if a valid table or flat surface is present in the scene, we use the ratio between the number of points that are within a 1 centimeter distance of the plane equation and the total number of points in the point cloud. Through experimentation we found that a ratio above 15% is generally indicative of a valid flat surface dominating the scene.

Finally, to ensure that the flat surface is not the floor or an irrelevant surface, we also verify that the minimum distance to the plane from the robot’s origin, $(0, 0, 0)$ in *world* coordinates, is at most 25 centimeter. If not, the detected object is rejected and the deep learning-based object detection is used.

Deep learning-based method

This method relies on a Mask R-CNN [99] model trained on the LVIS dataset [104] from the pretrained models available from [180]. The Mask R-CNN model is useful in this context because it outputs masks of the detected objects instead of simple bounding boxes that can be applied on the depth image to reconstruct the points belonging to that specific object.

The LVIS dataset [104] on the other hand contains a large amount of categories (1200+) of daily objects, which makes it relevant for grasping tasks related to activities of daily living.

Given the detected segmentation mask of each object, we can then apply the mask from the color image to the depth image, assuming the color and depth are aligned using the software development kit of the Intel Realsense cameras [26]. By reconstructing in 3D space each depth point belonging to the mask, we obtain a small point cloud corresponding to the detected object. We can therefore compute the axis-aligned bounding box around these points and its center, required for the object detection module.

An issue that arises is that the mask around the object produced by the deep learning model

might not be exactly fitted to the actual object on the depth image, therefore including points that are part of the background that can be further away from the object. To filter out the background points, we use the DBSCAN clustering algorithm [179] to find the main cluster of point of the resulting mask point cloud. This main cluster is the one containing the most points and contains little to no background points.

Although the detection of a large variety of objects allows the system to better detect the graspable ones, it also leads to the detection of non-graspable objects such as a table or a carpet and can detect and classify a single object into multiple different segmented regions. As an example of the latter, a paper coffee cup with a lid can be detected as both a cup and a lid, where the upper region of the cup is classified as the lid, see Figure 6.4.

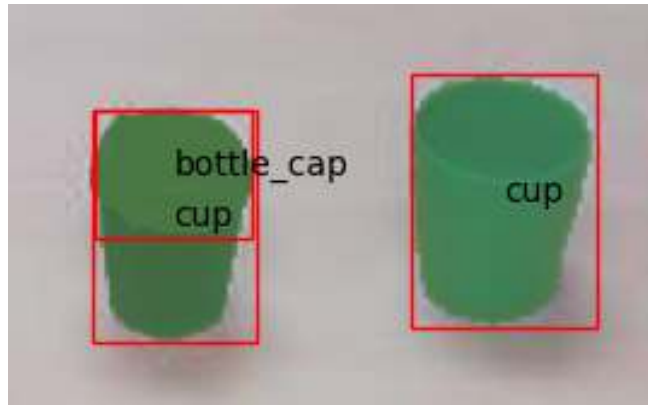


Figure 6.4 Example of object segmentation including the object and its lid as separate detections.

To filter out the non-graspable objects, we remove detections that are classified as a non-graspable object, such as a table, a tablecloth, a carpet, a mousepad, etc. We also remove detections using the size, where objects that are wider than the maximum gripper opened width of the ARM are filtered out. The width is defined as the maximum horizontal axis-aligned 3D bounding box dimension of the object, meaning the height is discarded since a tall but narrow object such as a water bottle can still be grasped.

To avoid the issue of multiple classes and segmentation masks being assigned to a single real life object, we combine the detected masks through an intersection-over-area approach. For every detected object, given their segmentation mask and bounding box on the image, we run the procedure described by Algorithm 2.

Considering the object masks A and B , we first check if the bounding box of A is mostly contained within the bounding box of B by computing the ratio of the area of the intersection of A and B over the total area of A . In the case where A is completely within the bounding box of B , the ratio will be equal to or above 1. The threshold is set to 80%. The selection

Algorithm 2 Algorithm to determine if two detection masks A and B should be combined.

```

procedure SHOULD_COMBINE( $A, B$ )
   $A_{box} \leftarrow \text{BoundingBox}(A)$ 
   $B_{box} \leftarrow \text{BoundingBox}(B)$ 
  if  $\text{Area}(A_{box} \cap B_{box}) / \text{Area}(A_{box}) > 0.8$  then
    if  $\text{Area}(A \cap B) / \text{Area}(A \cup B) > 0.1$  then
      return True
    end if
  end if
  return False
end procedure

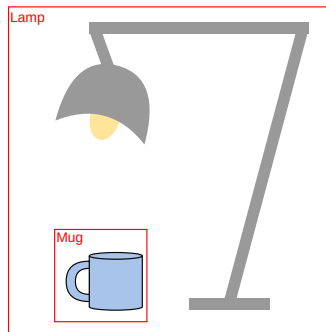
```

of this value was done through trial and error.

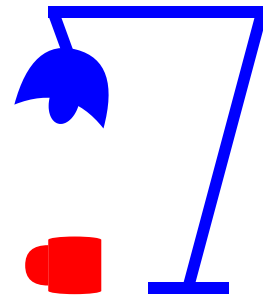
While this previously described threshold handles all convex objects, objects that are concave can be combined with another mask even though their mask do not overlap as is demonstrated by Figure 6.5. As an example, an inverted L-shaped lamp can appear as a very large bounding box which could completely encompass a mug under the lamp. To prevent these situations, we compute the Intersection-Over-Union of both masks, where it should be at least 10% to proceed with the combination of both detections.

6.4.4 Detected Object Selection

The detected object selection module takes as input the detected objects centers and bounding boxes and the gaze direction and origin and outputs the selected object center and bounding box from the gaze information.



(a) Bounding boxes



(b) Segmentation masks

Figure 6.5 Example of a concave object's detection bounding box containing another object without any overlap in their masks.

The selected object should be the object in the scene that most closely match the gaze information. Indeed, the gaze can be represented as line starting from the gaze origin, defined as the position between the user's eyes, and going in the direction of the estimated gaze vector. Although a simple approach would be to match the gaze line by finding the closest point in the reconstructed 3D scenes as in previous work [8], this raises several issues:

- If the gaze point is not at the center of the detected object, the grasping motion might be aimed at an offset from the object center, leading to knocking over of the objects or missing the object.
- The appearance-based gaze estimation is generally much less accurate than pupil center corneal reflection-based methods [52], with an accuracy of 3-5 degrees on average [168], the gaze vector might be offset from the actual object, meaning the closest point of the 3D scene might be on the table beside and behind the object for example.
- No width information can be extracted from a single point, as is needed to filter out objects that are too big to grasp.

Therefore, we use a sequential method where if a step of the method fails to find the object of interest, it computes the next step. Indeed, since the user expresses their intent to grasp an object explicitly by pressing a button as will be discussed in Section 6.4.7, we can safely assume that there *must* be an object the user is aiming to grasp.

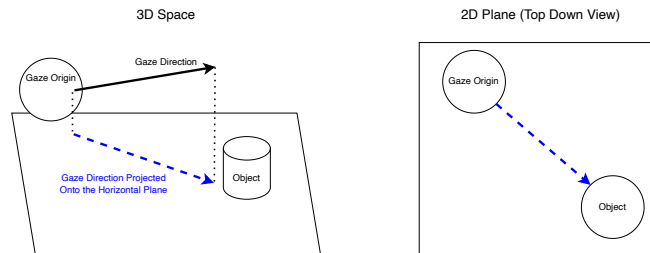


Figure 6.6 Reduction of the 3D object selection problem to 2D by projecting onto the horizontal plane.

The selection steps are as follows:

1. Find the objects for which the bounding box is intersected by the gaze line.
2. If the gaze line intersects more than one bounding box, select the object with the bounding box center the closest to the gaze line.

3. If no bounding boxes are intersected by the gaze line; find the object for which the gaze line is the closest to its bounding box center when the object centers and gaze line are projected onto the horizontal plane. See Figure 6.6.

Step 1 uses the fast axis-aligned bounding box ray intersection test presented by [181]. Steps 2 is done by simply projecting the vector between the object center o_c and the gaze origin g_o on the gaze direction vector g , and adding that vector to the gaze origin to find the closest point p_g on the gaze line to the object center, as given Equation 6.6.

$$p_g = \frac{o_c \cdot g_o}{||g||^2} g + g_o \quad (6.6)$$

Step 3 reuses Equation 6.6, with o_c and g_o points and g vector projected onto the horizontal plane, in our case the XY plane.

The main goal of the 3rd step is to correct for the most probable error in gaze estimation. Indeed, since the gaze estimation model makes use of simple images of the face to infer the horizontal and vertical direction of the gaze, the vertical component is generally much less accurate. In fact, we often experienced very accurate gaze estimation in the horizontal plane but with the vertical gaze direction constantly offset up to 10 or 15 degrees in some cases. This is due to the complexity of computing an accurate face pose in the pitch axis from a singular image, as the user's face can vary in shape as to appear angled down or up compared to another user. Since this head pose information is used by the gaze estimation model, this can lead to constant pitch (vertical) offsets.

While the online calibration can help alleviate this error after a few samples has been processed for a specific user, the first few samples must often use this horizontal-only gaze direction selection process (step 3) to find the object of interest.

6.4.5 Path Planning

The path planning module takes as input the selected object center and bounding box as well as the point cloud obtained during the scene reconstruction in Section 6.4.2 and outputs a list of collision-free joint positions for the robot.

While motion planning includes computing velocity profiles, in our case the robot used (Kinova JACO) is only compatible with joint or cartesian position commands through the open-source drivers available on Linux-based embedded computers. Therefore, we simply generate a list of joint angle positions and assume a constant velocity between those points.

The path planning module consists of a collision checking submodule as well as a path

generation submodule. Indeed, to generate a path, sampled joint positions must be checked for both self-collision and collision with the environment. The following subsections will detail each submodule.

Collision Checking

For a given joint angle configuration of the ARM, the collision checking submodule should be able to detect whether the robot is:

- in collision with the environment;
- in collision with itself;
- in a configuration outside of its joint limits;
- within a protection zone, such as being close to the user.

To do so, the collision checking submodule requires the following information about the robot:

- the mesh or bounding boxes of each link;
- the angular limit of each joint;
- the pairs of link for which no self-collision check should be done, either because it is impossible due to the joint limits or geometry or because the mesh of the robot overlap.

Generally, this information is distributed by the manufacturer of the ARM, often through a Unified Robotics Description Format (URDF) file, which details the link and joint properties. It is often bundled together with mesh files which define the geometry of each link of the robot, which is the case for the ARM used in this paper. The use of a URDF file with bundled meshes greatly simplifies the loading of the robot model as available libraries such as URDFpy [182].

To ensure the fastest collision checking available directly in Python, we used the Flexible Collision Library [183] through a modified version of Python-FCL [184] which includes Octomap's [185] octree implementation for collision checking. As opposed to other collision checking library like PyBullet [186] for which an octree-based collision check is not available, the use of octree allows for a natural representation of a point cloud without having to convert to a mesh or compute a collision check for every point.

Algorithm 3 details how the collision check logic is implemented. To reduce the computational burden, we use a simplified robot geometry made of bounding boxes of each link instead of the complete and detailed mesh. While this can be done when checking for collisions with the scene, we cannot use the simplified geometry when testing for self-collision, as the bounding box occupy by definition a larger volume than the mesh. This thus leads to false self-collision detections.

Algorithm 3 Algorithm to check for collision given a joint angle configuration q

```

procedure CHECKCOLLISION( $q$ )
   $N \leftarrow$  number of links of the ARM
   $L \leftarrow$  list of link meshes
   $W \leftarrow$  scene octree from point cloud
  if not WithinJointLimits( $q$ ) then
    return True
  end if
  for  $i \leftarrow 1$  to  $N$  do
     $l_i \leftarrow L[i]$ 
     $T_i \leftarrow$  ForwardKinematics( $q, i$ )
    ApplyTransformToMesh( $T_i, l_i$ )

    // Check for self-collisions
    for  $j \leftarrow i$  to  $N$  do
      if  $(i, j)$  is not in the skip self-collision list then
         $l_j \leftarrow L[j]$ 
         $T_j \leftarrow$  ForwardKinematics( $q, j$ )
        ApplyTransformToMesh( $T_j, l_j$ )
        if FCLCollisionCheck( $l_i, l_j$ ) then
          return True
        end if
      end if
    end for

    // Check for collision with world
    if FCLCollisionCheck( $l_i, W$ ) then
      return True
    end if
  end for
  return False
end procedure

```

Path Generation

The path generation submodule outputs a list of joint angle configurations for the robot to move from its current position to a grasping position. Iteratively, a grasping position around or on top of the object is generated and tested for kinematic validity and collision. If the inverse kinematics fails to converge within tolerance, is in collision or the path generation fails to find a path, another grasping position is generated until all grasping positions are tested.

Given the object center o_c and a pre-grasp offset vector d , we use a multi-step approach to find a potential grasping position. The pre-grasp offset vector d is a setting that can be changed depending on the robot which defines a fixed offset from the object to be grasped to which the path generation should lead. This position should be far enough from an object as to not be in collision with it because if the object center was taken directly as the goal to reach, the generation would never find a path.

In our case, we chose an offset of 13 centimeters before the object and 5 centimeters above it, which translates to an offset in the end effector reference frame of $d = [0, -0.05, 0.13]$ in meters. We define the offset in the reference frame of the end effector as this will make the first goal point be the closest to the current position and pose of the end effector.

Once the offset goal is reached, a direct line path from the offset goal to the object center is done without any collision checking.

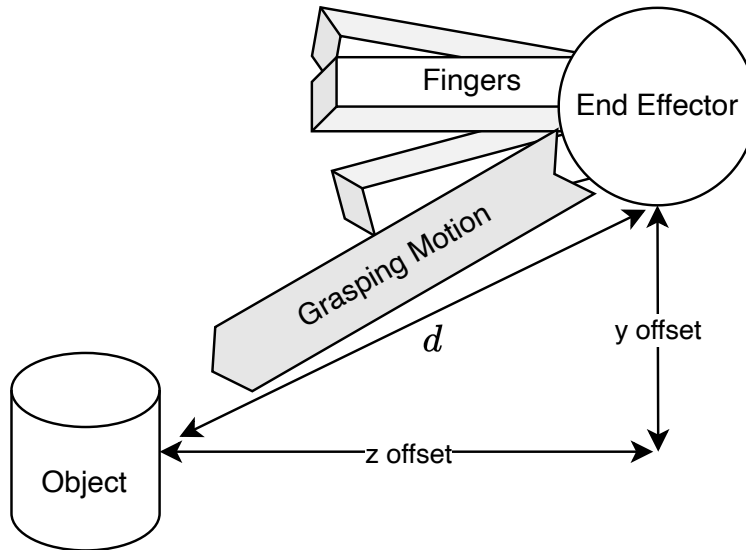


Figure 6.7 Depiction of the offset position and direction of the linear grasping motion.

To generate a potential grasping position, we rotate the offset around the object center until the position is kinematically feasible and not in collision. If the path generation fails, we

continue the rotation of the offset around the object. If a full circle is completed and no path could be generated, we try to grasp the object from the top. In this case, the offset d changes as to simply be 13 centimeters away from the object top ($d = [0, 0, 0.13]$). The top is found using the obtained bounding box of the selected object.

Algorithm 4 describes the method used to generate the angle to rotate around the selected object center. The parameter l is used to vary the spacing around the points. In our case, this is set to 1.6, meaning the distribution of points is denser close to the original pose of the end effector, and gets less dense the further away from it. Also, there is an alternance between a rotation counterclockwise and clockwise (see Figure 6.8 for a visual representation). This value was chosen as it is a good compromise between the density of poses closest to the original pose while still exploring poses that are positioned behind the object.

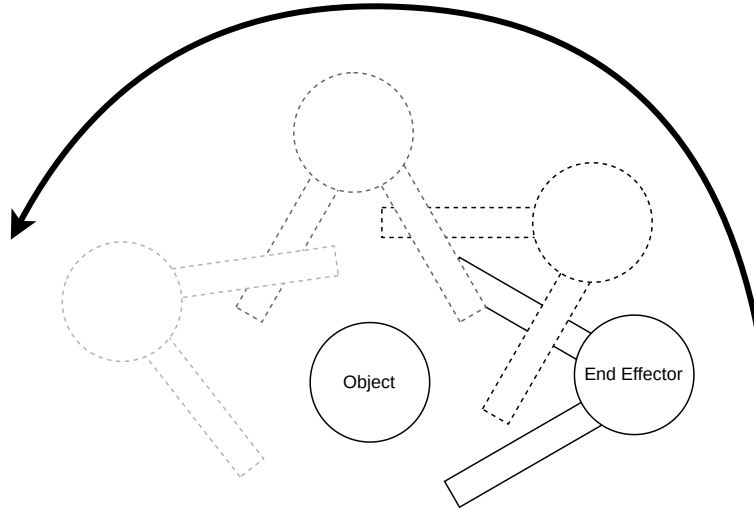


Figure 6.8 Possible grasping solution around the object. Note that this shows only one side of rotation around the object, but in reality this is mirrored to the other side. Also note the spacing between potential grasping poses that scales according to the power of l .

Given a goal position p_g at a rotated offset from the selected object center o_c , we define the goal pose of the end effector as positioned at p_g , with the Y axis of the end effector pointing up and the Z axis oriented from g_p to o_c . The goal pose X axis is thus the normalized cross-product of the Y and Z axis of the end effector.

In the case where no point around the object leads to a valid path, we set the goal pose as above the object as described previously, with the Z axis pointing down. The X and Y axis can be selected arbitrarily as the 6th joint of the robot (the wrist) can be rotated to any X and Y axis combination.

The path planning method used is Rapidly-exploring Random Trees (RRT) [187] which, while not being an optimal path planning method, is very fast to compute and hence led to a good

Algorithm 4 Algorithm to rotate around the offset vector around a circle.

```

procedure ANGLEAROUNDCIRCLEINRADIANS( $(i, n)$ )
  //  $i$  is the index of the point around the circle
  //  $n$  is the total number of points around the circle
  //  $l$  is a parameter to make the spacing between the points not be constant.
  if  $i$  is even then
    return  $(\text{Floor}(\frac{i}{2}) / \frac{n}{2})^l \pi$ 
  else if  $i$  is odd then
    return  $-(\text{Floor}(\frac{i}{2}) / \frac{n}{2})^l \pi$ 
  end if
end procedure

```

compromise between path quality and compute time in our case. We use the implementation of RRT by [188] in Python. We set a maximum planning time of 0.5 seconds, after which we consider that no path could be found for grasping and we continue with to another position. We also define the sampling interval as 0.1 radian, which although coarse, proved to be enough while allowing for faster computation.

The path is also smoothed and simplified by removing any middle point of the trajectory for which no collision occur by interpolating from the previous to the next trajectory point.

6.4.6 Additional control modes

While the comparative results presented in this paper are obtained for the automated grasping through detection of the gazed-at object, we also implemented other control modes which can facilitate the use of an ARM through gaze estimation.

Gaze Vector Control Mode

This mode controls the robot using the gaze direction of the user. Instead of trying to match a detected object to the gaze origin and direction of the user, the system plans a path towards the user's gaze vector and then moves along it. The motion is accomplished step-by-step, requiring the user's consent through a physical button to continue moving (see Figure 6.9 for an example of the movements accomplished by this control mode).

To do so, we set a gaze vector joining point p_j at a distance d_j of 0.5 meters from the user's face on the gaze vector line. This joining point will be the goal position for the end effector if the robot is closer to the user's face than d_j . Once the robot is past the d_j distance from the user, the gaze vector join point is set as the closest point on the gaze vector line, as in Equation 6.6. This is done so that if the gaze direction of the user changes, the robot does

not come back to the initial p_j point. Once the gaze vector joining point is reached, the robot moves along the gaze vector.

If the user requests the robot to move toward their face on the gaze vector instead of away, the criteria for joining the gaze vector at either a distance d_j or the closest point on the gaze vector line is flipped, i.e., the robot must be closer to the user than d_j .

This mode is intended to supplement the joystick by allowing automated but less precise movements of the robot by using the gaze direction of the user. It can allow for fast targeting of objects or light switches for examples, where the final approach and action motion of the robot is accomplished by the joystick instead.

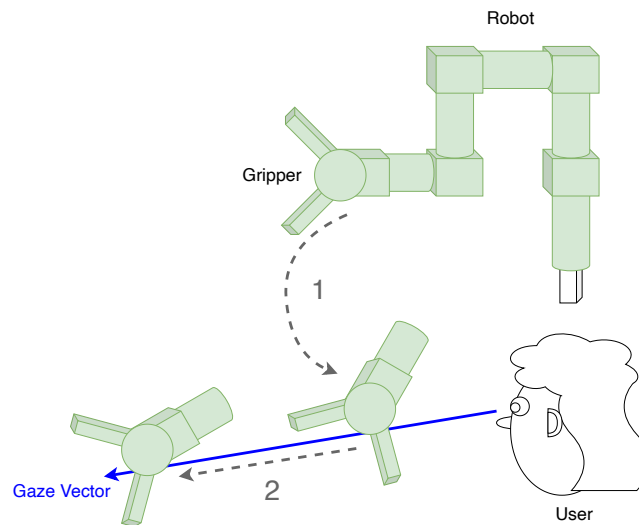


Figure 6.9 Depiction of the motion steps of the gaze vector control mode. The first step plans a motion step-by-step from the robot's initial position to the joining point (or closest point) on the gaze vector. Once reached, the robot then moves along the gaze vector. In this case, the forward motion is depicted.

Move to Face Control Mode

This control mode is used when an object is already grasped and the user requests the robot to automatically move it to a position to drink, eat or talk on the phone.

While preplanned paths are often programmed into the joystick as a selectable action, these motions do not consider possible obstacles and also do not account for any head positioning of the user.

As such we implemented a move to face automated motion mode, where the robot plans a path from the grasping position to a position at a specified distance from the detected face. This mode uses the same path planning method as in Section 6.4.5, but instead of planning

the path in the joint angle configuration space of the robot, it is planned in a 4 dimensional space of the end effector position and orientation, but keeps an upright direction.

Essentially, the search space is defined as the position (x, y, z) and angle of the end effector around the object (ψ) , to keep the object upright and avoid spillage of drinks or food.

A search space coordinate (x, y, z, ψ) is defined as collision-free if both an inverse kinematics solution exists and the solution is not in collision with any obstacle.

6.4.7 Control Flow



Figure 6.10 Remote controller used for controlling the robot with the labels for each button.

To allow the user to easily control the robot using their gaze, a simple remote is used. It is connected to the embedded computer through USB and possesses three buttons which we named: *Continue*, *Revert* and *Special* (see Figure 6.10 for a picture of the remote with the corresponding button labeled). The functions of each button is dependent on the state of the system and will be detailed in Section 6.4.7.

The general user flow is as follows:

1. The system processes the gaze of the user in a loop, waiting for them to press a button.
2. Once the *Continue* button is pressed, the system tries to detect, select and plan the grasping of the gazed-at object.

3. If the system is successful in selecting an object and a valid path is found, the robot starts moving toward the offset pre-grasp point. At that point, the user can either confirm (via the *Continue* button) that the right object was selected by looking at the robot's end effector and seeing if it is close to and aiming at the object that the user was looking at, or ask the robot to return to its home position and return to its initial state, waiting for another button press by the user.
4. If the user pressed *Continue*, the robot then grasps the object.

At any point, the user can ask the robot to cancel an action and return to home using the *Revert* button. The different states of the system and their interactions are detailed in the following section.

System States

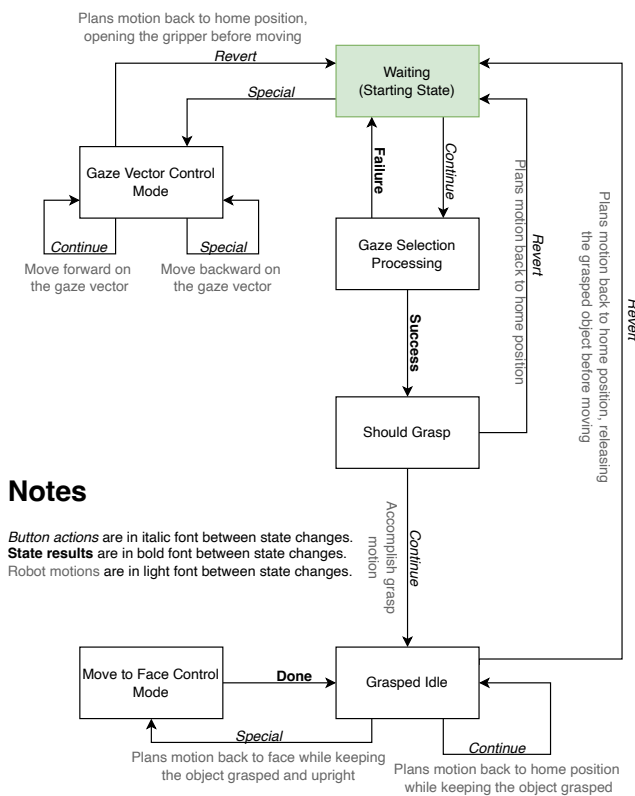


Figure 6.11 Progression of app states and robot motions given the button inputs and state results.

The system can reach several states depending on the user actions (button presses) and the success of object detection and selection and path planning. Figure 6.11 shows the actions and success or failure required to progress between states.

Waiting State: the system starts in a waiting state, where it awaits a user request to analyse the scene and select an object. In this state, each main program loop also processes the gaze and keep a first-in-first-out queue of gaze estimation directions and origins as described in Section 6.4.1. This makes it so all the required gaze estimation data is available at the time the user presses a button, and thus reduces the latency.

Gaze Selection Processing State: once the *Continue* button is pressed, the system starts by reconstructing the scene (Section 6.4.2), then tries to detect the objects in the scene (Section 6.4.3) and to select the best matching object (Section 6.4.4) and then plans a collision-free path to the offset pre-grasp point (Section 6.4.5). If any of the processing step fails, the system state is sent back to the *Waiting State*. If it is successful, the system is progressed to the *Should Grasp State*.

Should Grasp State: in this state the system awaits for the user to confirm that the right object was selected using the *Continue* button. If the *Revert* button is pressed instead, the robot moves back to its home position and to *Waiting State*. If the *Continue* button is pressed, the grasping motion is accomplished.

Grasped Idle State: once the object is grasped, the system is set to this state, which awaits for user input. In this state, the user can use the robot normally through the joystick. If the *Continue* button is pressed, a path is planned back to the robot's home position while still grasping the object. If the *Revert* button is pressed, the robot releases the object and moves back to its home position, and to *Waiting State*. This ensures that the system must have no object grasped for the next automatic gazed-at object grasping motion.

Gaze Vector Control Mode State: if the *Special* button is pressed from the *Waiting State*, the gaze vector control mode (Section 6.4.6) is entered and the system is set to this state. The *Continue* button is used to move the robot forward and the *Special* button is used to move the robot backward. The *Revert* button accomplishes the same action as in the other states, it simply plans a path back to the robot home position and back to *Waiting State*.

Move to Face Control Mode State: if the *Special* button is pressed when an object is grasped with the automated system, i.e., the system is in the *Grasped Idle State*, the motion mode described in Section 6.4.6 is processed. If no viable path is found, the system moves back to the *Grasped Idle State*.

6.5 Results

To validate the usefulness of the gaze control system described in this paper, device trials were done. As such, the following sections will describe the validation protocol.

6.5.1 Population

Table 6.2 shows the collected participant aggregate data. The participants were recruited both on and off campus, with a general bias towards on campus as can be seen from the age average.

The participants were all adults without any upper-limb disabilities that would be typical of ARM users, as the recruitment of as many participants as this study would have been very hard due to the limited availability of public contact information or central list and the limited population available.

6.5.2 Tasks

The trials were done in two parts: the first part using the joystick that comes with the ARM and the other using the gaze control system described in this paper. By working with the joystick first, users would be less inclined to try and mimic the robot motions as done by the gaze control system.

The tasks were to start from the ARM home position and move it to grasp a single specified object in front of the user. Several configurations of objects and obstacles were tested:

1. Two grasping tasks with a single object in the scene
2. Two grasping tasks with a single object and a single obstacle in the scene
3. Two grasping tasks with 2 objects in the scene
4. Two grasping tasks with 2 objects and an obstacle in the scene
5. Two grasping tasks with 3 objects in the scene
6. Two grasping tasks with 4 objects in the scene

We define an *object* as objects in the scene which could be grasped by a user such as a mug or a cup and an *obstacle* as a part of the scene that would not be grasped voluntarily by a user, a box larger than the gripper can grasp in our case. Therefore, for each grasping task,

Table 6.2 Participant data and distribution

	Average or Value	Median
Number of participants	30	-
Reported age	35.6 $\pm$ 16.2	28.0
Reported gender (proportion of males)	67%	-
Previous experience with ARM proportion	13%	-

a single object had to be grasped, with the obstacle or other objects all acting as obstacles and to verify the accuracy of the gaze selection in the case of the gaze control system (see Figure 6.14).

For each specific grasping task over all trials, the same configuration of objects and obstacles were preserved, but allowing for variability in the position, i.e., the objects were not fixed in place and no marker or measurements were done to place them at the same position.

The room used for the trial had a large window with direct sunlight illumination.

6.5.3 Data Collection

For each task of the trial, the time to grasp an object was measured from the start at the home position until the object is firmly grasped by the robot. Figure 6.12 and Table 6.3 show the average execution time for each task of the trials and overall respectively. Figure 6.13 also shows the different average times to grasp for each age group and control interface.

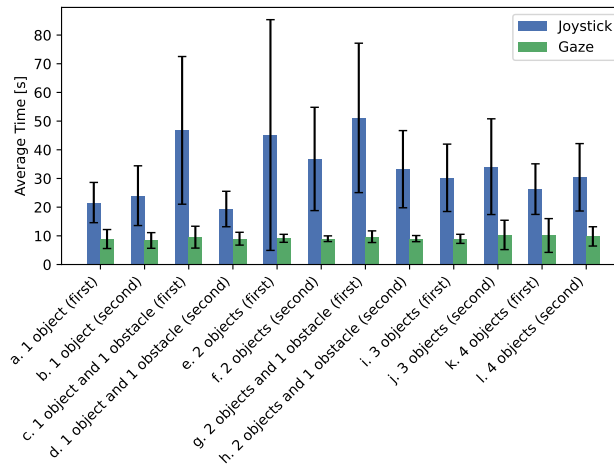


Figure 6.12 Average execution time per task and standard deviation for the joystick and gaze control system.

If the object to grasp fell over during the trial, it was classified as a failure and the trial was restarted from the home position. For the gaze control system, if the system was not able

Table 6.3 Overall Time to Accomplish a Task

	Average [s]	Median [s]
Joystick	33.3 $\pm$ 10.2	25.1
Gaze	9.3 $\pm$ 0.570	8.7

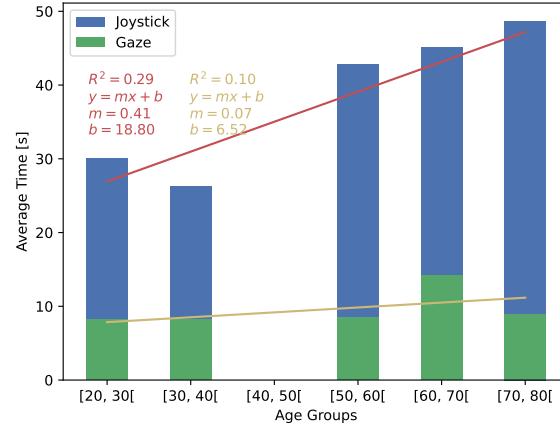


Figure 6.13 Average execution time per task for each age group.

to grasp the correct object after three times pressing the *Continue* button, the sample was classified as failed. Table 6.4 presents the success rates for each task and control method.

If any collision with an obstacle or object occurred, the trial was classified as *with collision*. The collision rates are presented in Table 6.5.

Every failure to select the right object was also noted. The gaze estimation system failed to select an object or the right object in 4.2% of the trials.

Finally, after the trials, the participants were asked to complete a short questionnaire to rate the gaze control system on a scale from 1 (worst) to 5 (best) on the following points:

- Perceived ease of use.
- Perceived quickness.
- Perceived efficiency.
- Perceived completeness, i.e., *does it accomplish the task you were trying to achieve?*
- Overall satisfaction.
- Perceived safety.

Table 6.4 Success Rates (%) Across All Trials

Task	Joystick	Gaze
a. One object (first)	93.8%	100.0%
b. One object (second)	100.0%	96.8%
c. One object and one obstacle (first)	93.8%	93.8%
d. One object and one obstacle (second)	96.8%	93.8%
e. Two objects (first)	90.9%	88.2%
f. Two objects (second)	90.9%	85.7%
g. Two objects and one obstacle (first)	96.8%	85.7%
h. Two objects and one obstacle (second)	85.7%	75.0%
i. Three objects (first)	96.8%	90.9%
j. Three objects (second)	96.8%	88.2%
k. Four objects (first)	100.0%	90.9%
l. Four objects (second)	100.0%	93.8%
Overall	95.0%	89.8%

Table 6.5 Collision Rates (%) Across All Trials

Task	Joystick	Gaze
a. One object (first)	3.3%	0.0%
b. One object (second)	3.3%	0.0%
c. One object and one obstacle (first)	0.0%	0.0%
d. One object and one obstacle (second)	6.7%	0.0%
e. Two objects (first)	6.7%	3.3%
f. Two objects (second)	13.3%	0.0%
g. Two objects and one obstacle (first)	13.3%	3.3%
h. Two objects and one obstacle (second)	13.3%	3.3%
i. Three objects (first)	20.0%	6.7%
j. Three objects (second)	10.0%	10.0%
k. Four objects (first)	10.0%	3.3%
l. Four objects (second)	10.0%	16.7%
Overall	9.2%	3.9%

The collected responses are presented in Table 6.6. The *average deviation* column represents the average score deviation from the average score given for all questions responded to by a participant. This latter column can thus be used to determine what are the attributes that have been consistently rated below the average score given by a participant.

Table 6.6 Responses about the perceived attributes of the gaze control system

Attribute	Average Score (/5)	Average Deviation
Perceived ease of use	4.9 $\pm$ 0.3	7.6%
Perceived quickness	4.7 $\pm$ 0.6	1.4%
Perceived efficiency	4.4 $\pm$ 0.5	-3.7%
Perceived completeness	4.5 $\pm$ 0.7	-3.1%
Overall satisfaction	4.6 $\pm$ 0.6	0.7%
Perceived safety	4.5 $\pm$ 0.6	-3.0%

6.6 Discussion

6.6.1 Interpretation

As can be seen from Figure 6.12 and Table 6.3, the gaze control system is generally much faster to use than the joystick for the grasping tasks accomplished in these trials. On average, this translates to a 72.0% reduction of the manual grasping time. The standard deviations are also of note, demonstrating that the gaze control system maintains generally the same time to grasp an object, whereas the time to grasp varies significantly more with the joystick.

Indeed, tasks which generally requires a change in the orientation of the robot's end effector (more than a single object in the scene, e.g., one object and one obstacle or two objects) have a significantly higher time to grasp with the joysticks, as can be seen for tasks *c* and *e* through *l* in Figure 6.12. The difference is marked with changes in orientation that are not a simple yaw rotation around the object, but generally requires multiple rotations such as is the case when an obstacle is present. This can be explained by the complexity of changing between the multiple modes (position, orientation and grasping) on the joystick leading to longer times.

Also, Figure 6.13, shows a correlation ($R = 0.54$) with a strong slope of 0.41 between the age and the time to grasp with the joystick, where older participants were slower to grasp objects on average than younger ones. This correlation is lower ($R = 0.32$) with the use of the gaze control system, which also shows a greatly decreased slope of 0.07.

Furthermore, as can be seen from the attributes score in Table 6.6, the overall positive perception of the gaze control system is high, with an overall average of $4.6/5 \pm 0.6$. And, more importantly, the perceived ease of use is rated higher than the average as can be seen from the *average deviation* column of Table 6.6.

Finally, Table 6.5 shows that the automatic path planning of the gaze control system is generally much better at avoiding collisions than when using the joystick.

On the other side, the success rate is lower than the joystick at 90% versus 95% . The main reasons are:

- inverse kinematic failures leading to grasping motions that reach short of the object (at the finger tips);
- failure to detect the object as a separate entity due to its closeness to an obstacle or other object using the flat surface method;
- manipulation error, pressing on the wrong button or at the wrong time or looking at the remote instead of the object;
- some objects have protrusions such as the handle of a mug which can offset the grasping center which then leads to the object falling over as the end-effector cannot hold on to it.

Furthermore, as can be seen from Figure 6.12, tasks for which several objects are present and closely spaced such as is the case for the tasks i , k and l , the standard deviation of the grasping time for the gaze control system is higher. This is due to some trials where the gaze estimation failed to select the right object the participant was looking at, leading to a longer grasping time since the robot had to come back to its home position.

Finally, as can be seen from Table 6.6, while the perceived efficiency and safety scores are high, they remain lower than the other attributes. Indeed, the comments received on those attributes are most often aimed at the step-by-step, jerky movements of the robot. This is due to the limitation of the communication interface and drivers available for the robot used, which only allows for sending positions and not velocities to the robot.

Another reason for the efficiency perception being slightly lower comes from the fact that the user currently has no control over the way the object is grasped, which is decided automatically. This is especially true when the system grasps the object from the top instead of from the sides which might give a sense of losing control over the grasping process.

6.6.2 Limitations and future work

There is more male participants (67%) than female participants (33%) which could create a bias towards male faces and eyes in the validation as we carry out facial landmarks detection and gaze estimation on eye images. This is also true for the unbalanced samples in terms of age. Indeed, while the average age is somewhat centered, the median demonstrates a tendency toward younger participants (between 20 and 30) with few older participants.

For the gaze estimation model described in [168], glasses can lead to poor gaze estimation results and thus participants were asked not to wear glasses. This is due to a lack of samples with glasses in the publicly available datasets used for training the model [168]. There is currently on-going work to use more diverse gaze datasets to train the gaze estimation model, which should include more samples with glasses with varying optical properties.

Furthermore, the study was done with typically developed participants and not the target user group for the usage of a gaze control system for ARM. While people with dystrophies or other disabilities might have more difficulties in using the joystick due to a lack of dexterity or spasms, they also might have more experience in using ARM with a joystick, as opposed to the limited proportion of participant with experience using ARM in this study (13%). This means that in some cases, they might be, on average, slower than typically developed participants, but in other cases faster.

As the gaze estimation model is less accurate in the pitch axis, most trial tasks did not have graspable objects exactly one behind the other from the user's perspective. While this is limiting in certain scenarios, the fact that the ARM is mounted on a motorized wheelchair means the user could simply take into account this issue when positioning their wheelchair in front of the object they want to grasp.

Another limitation of this study is that all trial tasks were done with the objects resting on tables, meaning in most if not all cases, the object detection method used was based on the detection of a flat surface (table).

Finally, none of the tasks done during the trials had objects touching eachothers, which could lead to issues in both object detection and path planning.

6.7 Conclusion

In conclusion, while there are still several limitations to the gaze control method described in this paper, device trials show promising results, especially with a reduction in time to grasp an object of 72.0% over using a joystick while the general perception of the system being very positive ($4.6/5 \pm 0.6$). This type of system does not require very expensive sensors, specific lighting scenarios and objects to grasp, intrusive gaze estimation glasses and power hungry computers as is the case for most systems described in the literature.

Considering the improvements achieved when comparing to the use of the joystick in terms of time to grasp an object, the satisfaction of the users and the high success rate of the developed system, further developments could lead to improved autonomy of current and new users. Indeed, current users could benefit from this technology by reducing the required

time for them to accomplish actions of daily living, while people that do not have the required dexterity to use a joystick or other forms of currently available interfaces could finally be able to use ARMs through the gaze estimation-based interface described in this paper. This could lead to drastic improvements in the quality of life of the users, which is one of the pivotal points of sustainable development.

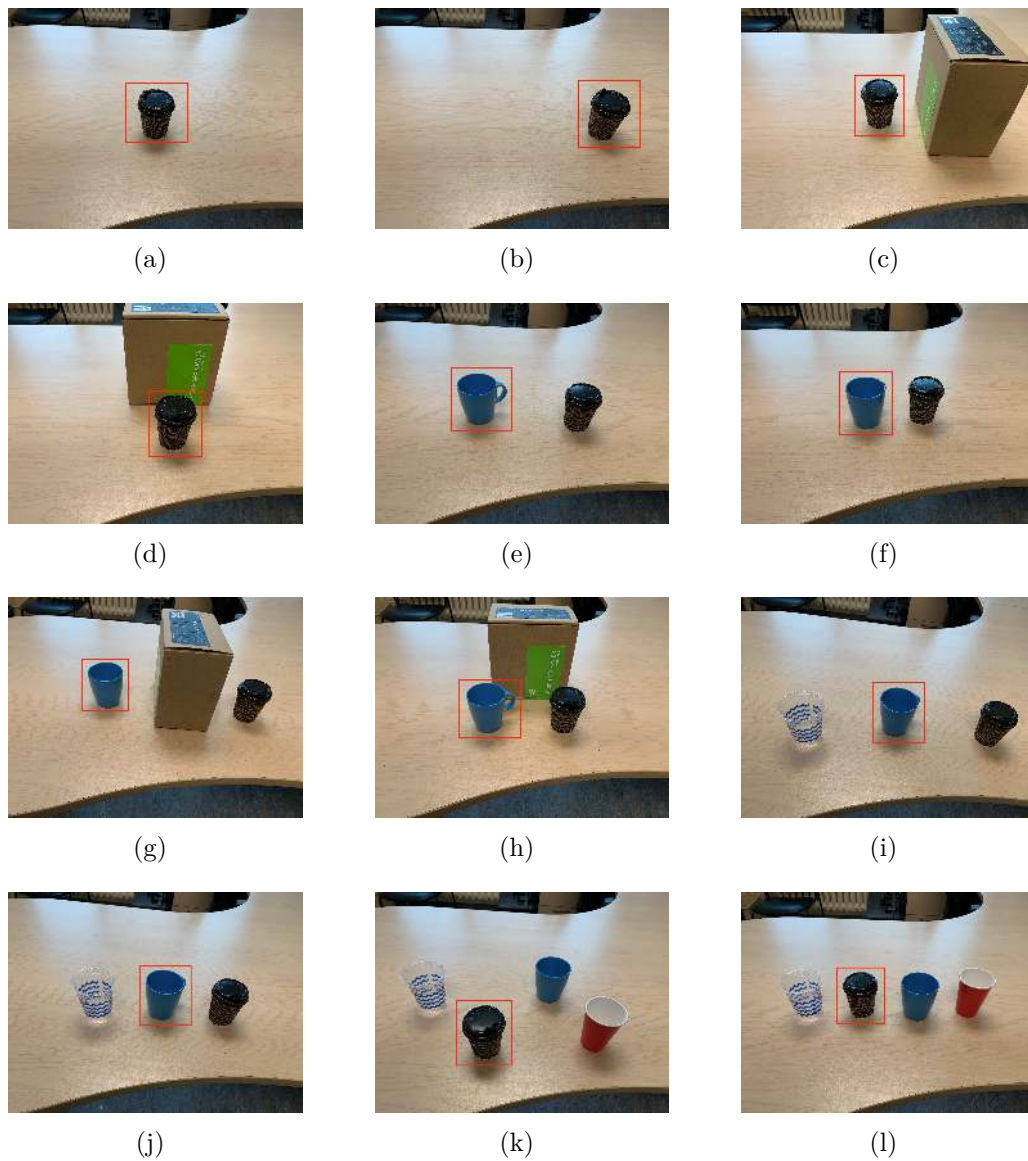


Figure 6.14 Approximate position of the objects for each task. The red boxes indicate the target object.

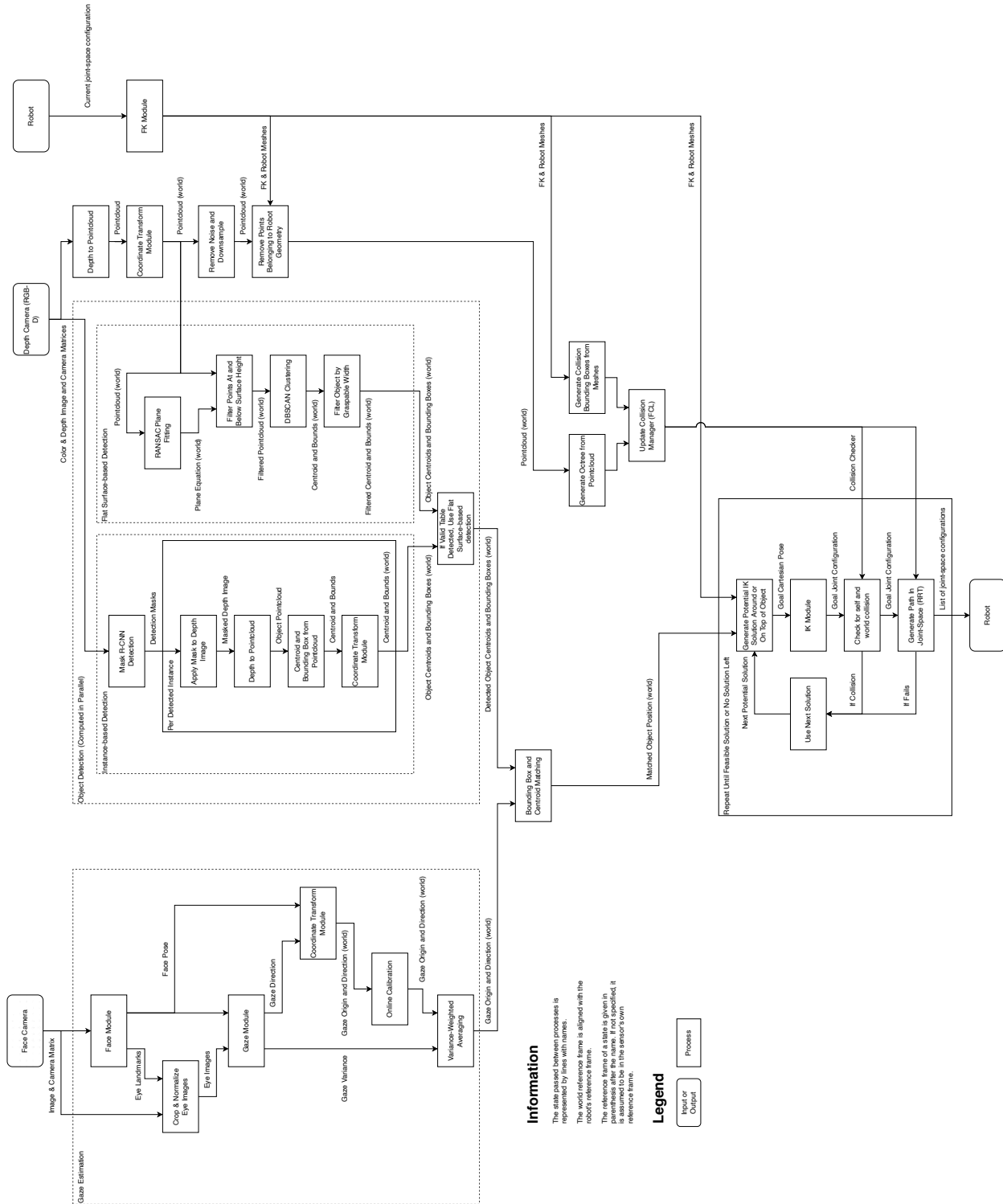


Figure 6.15 Process diagram of the complete gaze estimation, object detection and path planning workflow.

CHAPTER 7 GENERAL DISCUSSION

7.1 Improvements and Comparison to Previous Work

As first introduced by the literature review of Chapter 2, there is a clear lack of available HCIs for ARAs for people with upper limb disabilities, especially those who are not able to use a joystick controller with their hands.

As such, this thesis' work continues building upon the proof of concept described in [8]. As can be concluded from the articles in Chapters 4, 5 and 6, the prototype presented in this thesis is a completely new system with no codebase shared, and was built from the lessons learned.

The main differences from the previous proof of concept are as follows:

- No person-specific calibration is required. In this latest prototype, a user can simply sit and start using the system without any calibration. The only calibration needed is when the different sensors are mounted to the wheelchair or test setup, where the relative position and orientation of these sensors with respect to the robot must be calculated. This is only done once. The initial proof of concept required careful placement of a calibration screen to calibrate the eye tracker, which could take upwards of 30 minutes per user and was prone to being decalibrated by head movements.
- The gaze estimation subsystem works with significant sunlight illumination, as it does not rely on the PCCR method as opposed to the proof of concept and as was shown by the trials which had significant sunlight coming from a window. It is also much less sensitive to head movements, as can be demonstrated by the low failure rate of the gaze selection of 4.2% , even though the users could freely move their head.
- The 3D camera is now more robust to light changes and low texture objects by using an Intel Realsense D415 camera instead of a simpler stereo camera setup as was the case in the initial proof of concept.
- The object selection and grasping subsystems detect actual objects and not just points in the scene. Indeed, in the initial proof of concept, no proper object detection was done and instead the point of the pointcloud closest to the gaze direction was used. This meant that any inaccuracies in the eye tracker would lead to a failure to grasp the object, and the grasping method was static. In the new prototype, a valid, collision-free grasping angle is found depending on the object.

- The path planning is done in the joint configuration space instead of cartesian space for grasping paths, meaning the path is generally better optimized in terms of joint movements and also converges faster given that it does not rely on computing the inverse kinematics for each point when checking for collision.

7.2 Return on the Objectives

Considering the previous section, these differences mean the current prototype is much closer to a usable product than the initial proof of concept was as with real life tests, it would qualify as a Technology Readiness Level 4, which is the underlying goal of this project as to consider the general objective as attained. Indeed, as stated previously, the general objective is to:

Develop an embedded vision-based solution for facilitating the usage of assistive robotic arms for people with upper limb disabilities.

The following sections will explain how the developed system reaches the specific objectives.

7.2.1 Specific Objective 1

Develop a calibration-free, low cost, non-intrusive and embedded gaze estimation system.

Considering the system does not use the PCCR method and rather a regular webcam-style camera to estimate the gaze direction, we can therefore know that the gaze estimation works in unconstrained environments such as outdoors or lighting scenarios where direct sunlight illumination is present, which was not the case of the previous work and for several alternative ARA HCIs in the literature. This is also demonstrated by the device trials, which were done without imposing constraints on lighting conditions and still achieve a very low gaze estimation failure rate of 4.2% .

Moreover, the system does not require any person-specific calibration, only a single calibration is done to find the relative positions of the cameras with respect to the robot when doing the initial setup of the system on a motorized wheelchair, i.e., when assembling the system. A user can simply sit and start using the system immediately.

7.2.2 Specific Objective 2

Develop a real-time 3D scene reconstruction system using computer vision with object detection and segmentation.

The developed system make uses a commercial 3D camera (Intel Realsense D415 [26]) which works in both indoors and outdoors environment. It is then used to both generate the obstacles map for the path planning algorithm, but also for object detection and segmentation.

Indeed, the current system uses a hybrid approach to detect objects that runs in parallel. We first use a fast method based on table or flat-surface segmentation to detect objects sitting on those, while running in parallel a slower Mask R-CNN object segmentation model which does not use any flat-surface detection methods and instead relies on the appearance of the objects.

This allows for more robust detection, without relying on the presence of a table, which is the case for several alternative HCI systems in the literature.

7.2.3 Specific Objective 3

Develop a real-time path/trajectory planning algorithm taking the full robot geometry into account and leading to fast grasping times.

Considering the limitations to the available drivers for the ARA used in this project, only a real-time path planning module could be developed. It indeed takes into account the full geometry of the robot and the scene reconstruction as obstacles. It achieves real-time performance and efficient paths. This is demonstrated by the much faster grasping times when compared to the use of a joystick, with an average reduction in time of 72.0% . The average time to grasp an object, considering the time to process all modules of this system, and not just the path planning, is only 9.3 seconds on average.

It also achieves a high success rate, as is highlighted by the 90% success rate of the system to grasp objects.

7.2.4 Specific Objective 4

Validate the developed system with device trials with typically developed participants.

The obtained results from the device trials on 30 participants demonstrated a drastic improvement in grasping times while retaining a high success rate as in 90% versus 95% for the joystick. These trials also demonstrated that the system has a high satisfaction among participants, with a satisfaction score of 4.6 on 5 on average. This figure was obtained through a questionnaire completed by the participants at the end of their trials.

CHAPTER 8 CONCLUSION

As was described in Chapters 6 and 7, the current prototype of the gaze estimation workflow is much closer to a usable product for people with disabilities, although several elements remain to be improved.

8.1 Summary of Works

First, a novel gaze estimation workflow was created as to function with a simple color image of the user's face as would be available when using a webcam. This model was designed to be lightweight computationally so as to be used on an embedded computer. This led to a decrease in computation time of up to 91% (from 431 milliseconds to 38.3) when compared to other state-of-the-art gaze estimation models while achieving a comparable 4.5 degrees of error on average on the MPIIGaze dataset [71] and state-of-the-art 3.5 and 3.3 degrees on the UTMultiview [2] and GazeCapture [72, 133] datasets respectively. This is described in the article presented in Chapter 4.

Second, to prevent having to collect extensive gaze estimation samples to train the previously mentioned gaze estimation for the specific application of gaze estimation with an ARA, a novel data augmentation was developed which decreased the gaze estimation error for our application by an average of 47%. This is described in the article presented in Chapter 5.

Finally, a complete gaze control system for an ARA was proposed using the developed gaze estimation model and trained using the previously described data augmentation technique. This system includes the following subsystems:

- The previously described gaze estimation model, but with the added components of variance prediction used for weighted averaging with outlier rejection of gaze estimation samples and online calibration for correcting offsets caused by the differences between the users face and eyes physiognomy.
- A scene reconstruction subsystem which generates a point cloud using images obtained through a commercial 3D camera.
- An object detection subsystem which uses both a flat surface and the object's appearance to detect potentially graspable objects computed in parallel to speed up the processing.

- An object selection module which, given the detected object and estimated gaze, selects the best fitting object to grasp while accounting for possible reconstruction, detection and gaze estimation errors.
- A kinematics-only path planning module which allows for fast collision checking and generating a valid and collision-free path to the selected object.

The resulting gaze control system achieves a reduction in time to grasp objects of 72.0% from 33.3 to 9.3 seconds on average when compared to the use of the joystick that comes with the Kinova JACO robot used in the device trials. It also retains a fairly high success rate of 90% when compared to the joystick success rate of 95% . It also yielded a satisfaction score of 4.6 out of 5 on average for all participants.

8.2 Limitations

Although the gaze control system has many advantages, there are several limitations that should be addressed, especially if such a system is to become available to the general public.

First and foremost, the device trials detailed in Chapter 6 were conducted on participants without any upper limb disabilities. This was decided since the infrastructure available throughout this project was not adapted to those participants, but also because of the limited available time to recruit these participants. Therefore, device trials should be continued with participants with upper limb disabilities to further collect data but most importantly feedback on the usage of the system, to ensure no new issues would arise.

Also, another important limitation is that the system was only tested on participants not wearing glasses. From the limited pre-trial runs done, the gaze estimation failed several times when the users wore glasses. It was thus decided to ask participants to remove their glasses. This is most likely due to the limited availability of samples with glasses in the gaze estimation dataset used to train the model (MPIIFaceGaze [1]).

Furthermore, the system has limited usability when two objects are very close or touching each other, as the flat surface-based detection will fail to segment those as two separate objects. Only the image-based detection will segment each object individually, which can be limiting in cases where objects are in contact with each other.

Additionally, objects that are directly behind other objects are hard to select due to the limited accuracy in pitch of the gaze estimation model. While this can be alleviated by simply approaching the object of interest from a different angle, since the system would be mounted on a motorized wheelchair, it is still limiting.

Finally, the efficiency and safety of the system could be improved. Indeed, while users rated the perceived safety and perceived efficiency highly (4.5 and 4.4 on 5 respectively), there are still the lowest rated attributes of the system. Some of the most common comments are the jerky motions of the robot, which are due to the lack of trajectory planning. Indeed, these jerky movements occur mostly due to the fact that only a set of joint position points can currently be sent to the robot. This leads to high accelerations and decelerations, leading to a perceived lower safety and efficiency. The position of the buttons of the remote which could easily be pressed erroneously also contributed to this. The last component of this is also the fact that the user had no control over whether the system would grasp the object from the side or the top.

8.3 Future Research

Considering the previously described limitations, there is still future research and development to be made to optimize this gaze control system.

First of all, a solution to the lack of drivers available for embedded ARM-based processors should be found, either through reverse engineering or through support from the manufacturer. This would therefore allow for smoother and safer trajectories, which would also significantly reduce the probability of collisions.

Also, a faster face detection and gaze estimation workflow would further improve the accuracy and outlier rejection by allowing more samples to be gathered before selecting an object. This could be done through improvements in the models, porting to better performing frameworks such as TensorRT, using quantization and skipping the face detection step when a face was already detected once through object tracking (face tracking). Additionally, a more accurate gaze estimation model would allow for better selection of objects, especially for objects that are directly behind other objects.

Furthermore, porting the path planning and collision checking code to C++ would allow for much faster path planning, especially when combined with parallel processing which is much less efficient in Python. Faster path planning would allow for more optimal paths to be generated, but also more orientations and pre-grasping positions to be tested, allowing for planning a valid and collision-free path in environments that are even more cluttered.

Finally, improvements could be made to the control modes from the feedback received from the trials. This includes a better remote, step-by-step grasping controlled by the user, more specific object detection workflows, such as a mode dedicated to detecting and pressing elevator control panel buttons.

As a closing note, the use of very small and computationally inexpensive models in this thesis might allow for the use of a smartphone as the only required hardware of the complete system. Indeed, smartphones nowadays can include laser ranging equipment such as LiDARs and also contain a user-facing camera. As such, it can serve as both the 3D camera and face camera, while possessing the computational power to run the different models used in this PhD thesis.

REFERENCES

- [1] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “It’s Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation,” *arXiv:1611.08860 [cs]*, May 2017.
- [2] Y. Sugano, Y. Matsushita, and Y. Sato, “Learning-by-Synthesis for Appearance-Based 3D Gaze Estimation,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1821–1828.
- [3] A. M. Cook and J. M. Polgar, *Assistive Technologies: Principles and Practice*, fourth edition ed. St. Louis, Missouri: Elsevier/Mosby, 2015.
- [4] M. Chi, Y. Yao, Y. Liu, Y. Teng, and M. Zhong, “Learning motion primitives from demonstration,” *Advances in Mechanical Engineering*, vol. 9, p. 168781401773726, Dec. 2017.
- [5] V. Tanriverdi and R. J. K. Jacob, “Interacting with Eye Movements in Virtual Environments,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’00. New York, NY, USA: ACM, 2000, pp. 265–272.
- [6] E. B. Kelly, *Encyclopedia of Human Genetics and Disease*. ABC-CLIO, 2013.
- [7] M. Leroux, M. Raison, T. Adadja, and S. Achiche, “Combination of eyetracking and computer vision for robotics control,” in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, May 2015.
- [8] Y.-S. L.-K. Cio, M. Raison, C. L. Menard, and S. Achiche, “Proof of concept of an assistive robotic arm control using artificial stereovision and eye-tracking,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pp. 1–1, 2019.
- [9] B. Yang, J. Huang, M. Sun, J. Huo, X. Li, and C. Xiong, “Head-free, Human Gaze-driven Assistive Robotic System for Reaching and Grasping,” in *2021 40th Chinese Control Conference (CCC)*, Jul. 2021, pp. 4138–4143.
- [10] M.-Y. Wang, A. A. Kogkas, A. Darzi, and G. P. Mylonas, “Free-View, 3D Gaze-Guided, Assistive Robotic System for Activities of Daily Living,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 2355–2361.
- [11] “Exact Dynamics,” <http://www.exactdynamics.nl/site/>, (accessed 2019-09-09).

- [12] “Barrett Technology,” <https://advanced.barrett.com>, (accessed 2019-09-09).
- [13] “DLR - Institute of Robotics and Mechatronics - Regelung von Leichtbaurobotern,” https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-3803/6175_read-8963/, (accessed 2019-09-09).
- [14] “Robotic arm series,” <https://www.kinovarobotics.com/en/products/robotic-arm-series>, (accessed 2019-09-09).
- [15] A. Campeau-Lecours, V. Maheu, S. Lepage, H. Lamontagne, S. Latour, L. Paquet, and N. Hardie, “JACO Assistive Robotic Device: Empowering People With Disabilities Through Innovative Algorithms,” in *Rehabilitation Engineering and Assistive Technology Society of 516 North America (RESNA) Annual Conference*, Arlington, 2016.
- [16] C.-S. Chung, H. Wang, and R. A. Cooper, “Functional assessment and performance evaluation for assistive robotic manipulators: Literature review,” *The Journal of Spinal Cord Medicine*, vol. 36, no. 4, pp. 273–289, Jul. 2013.
- [17] S. S. Groothuis, S. Stramigioli, and R. Carloni, “Lending a helping hand: Toward novel assistive robotic arms,” *IEEE Robotics & Automation Magazine*, vol. 20, no. 1, pp. 20–29, Mar. 2013.
- [18] V. Maheu, J. Frappier, P. S. Archambault, and F. Routhier, “Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities,” in *2011 IEEE International Conference on Rehabilitation Robotics*. IEEE, Jun. 2011.
- [19] F. Routhier, P. S. Archambault, M.-C. Cyr, V. Maheu, M. Lemay, and I. Gélinas, “Benefits Of Jaco Robotic Arm On Independent Living And Social Participation: An Exploratory Study,” in *RESNA 2014*, Indianapolis, 2014.
- [20] S. Allin, E. Eckel, H. Markham, and B. R. Brewer, “Recent trends in the development and evaluation of assistive robotic manipulation devices,” *Physical Medicine and Rehabilitation Clinics of North America*, vol. 21, no. 1, pp. 59–77, Feb. 2010.
- [21] D. Ristic-Durrant, “Robust Stereo-Vision Based 3D Object Reconstruction for the Assistive Robot FRIEND,” *Advances in Electrical and Computer Engineering*, vol. 11, no. 4, pp. 15–22, Nov. 2011.
- [22] B. D. Argall, “Autonomy in Rehabilitation Robotics: An Intersection,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 441–463, May 2018.

- [23] I. Rulik, M. S. H. Sunny, J. D. Sanjuan De Caro, M. I. I. Zarif, B. Brahmi, S. I. Ahamed, K. Schultz, I. Wang, T. Leheng, J. P. Longxiang, and M. H. Rahman, "Control of a Wheelchair-Mounted 6DOF Assistive Robot With Chin and Finger Joysticks," *Frontiers in Robotics and AI*, vol. 9, p. 885610, Jul. 2022.
- [24] H. Ka, D. Ding, and R. A. Cooper, "Three Dimentional Computer Vision-Based Alternative Control Method for Assistive Robotic Manipulator," *Symbiosis*, vol. 1, no. 1, Feb. 2016.
- [25] "BlasterX Senz3D," <https://us.creative.com/p/web-cameras/blasterx-senz3d>, (accessed 2019-09-13).
- [26] "Intel® RealSense™ Technology," <https://www.intel.com/content/www/ca/en/architecture-and-technology/realsense-overview.html>, (accessed 2019-09-13).
- [27] "OpenRAVE | ikfast Module | OpenRAVE Documentation," <http://openrave.org/docs/0.8.2/openravepy/ikfast/>, (accessed 2019-09-13).
- [28] C. Bousquet-Jette, S. Achiche, D. Beaini, Y. S. Law-Kam Cio, C. Leblond-Ménard, and M. Raison, "Fast scene analysis using vision and artificial intelligence for object prehension by an assistive robot," *Engineering Applications of Artificial Intelligence*, vol. 63, pp. 33–44, Aug. 2017.
- [29] H. Ka, D. Ding, and R. A. Cooper, "ARoMA-V2: Assistive Robotic Manipulation Assistance with Computer Vision and Voice Recognition," in *The 9th Conference on Rehabilitation Engineering and Assistive Technology Society of Korea*. Goyang, Gyeonggi-do, Korea: RESKO, Nov. 2015.
- [30] H. Jiang, T. Zhang, J. P. Wachs, and B. S. Duerstock, "Enhanced control of a wheelchair-mounted robotic manipulator using 3-D vision and multimodal interaction," *Computer Vision and Image Understanding*, vol. 149, pp. 21–31, Aug. 2016.
- [31] "Kinect - Windows app development," <https://developer.microsoft.com/en-us/windows/kinect>, (accessed 2019-09-13).
- [32] H. Jiang, J. P. Wachs, and B. S. Duerstock, "Integrated vision-based robotic arm interface for operators with upper limb mobility impairments," *IEEE ... International Conference on Rehabilitation Robotics: [proceedings]*, vol. 2013, p. 6650447, Jun. 2013.
- [33] D. Kim, R. Lovelett, and A. Behal, "An empirical study with simulated ADL tasks using a vision-guided assistive robot arm," in *2009 IEEE International Conference on Rehabilitation Robotics*, Jun. 2009, pp. 504–509.

- [34] C. P. Quintero, O. Ramirez, and M. Jägersand, “VIBI: Assistive vision-based interface for robot manipulation,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4458–4463.
- [35] K. Tsui, H. Yanco, D. Kontak, and L. Beliveau, “Development and evaluation of a flexible interface for a wheelchair mounted robotic arm,” in *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2008, pp. 105–112.
- [36] S. Rakhimkul, A. Kim, A. Pazylbekov, and A. Shintemirov, “Autonomous Object Detection and Grasping Using Deep Learning for Design of an Intelligent Assistive Robot Manipulation System,” in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Oct. 2019, pp. 3962–3968.
- [37] G. Giuffrida, G. Meoni, and L. Fanucci, “A YOLOv2 Convolutional Neural Network-Based Human–Machine Interface for the Control of Assistive Robotic Manipulators,” *Applied Sciences*, vol. 9, no. 11, p. 2243, Jan. 2019.
- [38] R. Atienza and A. Zelinsky, “Intuitive interface through active 3D gaze tracking,” in *Proceedings of the 2005 International Conference on Active Media Technology, 2005. (AMT 2005).*, May 2005, pp. 16–21.
- [39] M. Buckley, R. Vaidyanathan, and W. Mayol-Cuevas, “Sensor suites for assistive arm prosthetics,” in *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)*, Jun. 2011, pp. 1–6.
- [40] U. Hassan, H. Mughal, I. Mohsin, and Z. H. Khan, “Real-time Control of a Mobile Robot using Electrooculogram based Eye Tracking System,” in *2018 5th International Multi-Topic ICT Conference (IMTIC)*, Apr. 2018, pp. 1–6.
- [41] S. Li, X. Zhang, and J. D. Webb, “3-D-Gaze-Based Robotic Grasping Through Mimicking Human Visuomotor Function for People With Motion Impairments,” *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 12, pp. 2824–2835, Dec. 2017.
- [42] R. O. Maimon-Mor, J. Fernandez-Quesada, G. A. Zito, C. Konnaris, S. Dziemian, and A. A. Faisal, “Towards free 3D end-point control for robotic-assisted human reaching using binocular eye tracking,” in *2017 International Conference on Rehabilitation Robotics (ICORR)*, Jul. 2017, pp. 1049–1054.
- [43] C.-C. Postelnicu, F. Girbacia, G.-D. Voinea, and R. Boboc, “Towards Hybrid Multimodal Brain Computer Interface for Robotic Arm Command,” in *Augmented Cognition*.

- 13th International Conference, AC 2019, Held as Part of the 21st HCI International Conference, HCII 2019, 26-31 July 2019*, ser. Augmented Cognition. 13th International Conference, AC 2019 Held as Part of the 21st HCI International Conference, HCII 2019. Proceedings: Lecture Notes in Artificial Intelligence (LNAI 11580). Springer International Publishing, 2019, pp. 461–70.
- [44] P. M. Tostado, W. W. Abbott, and A. A. Faisal, “3D gaze cursor: Continuous calibration and end-point grasp control of robotic actuators,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3295–3300.
 - [45] E. Zahir, M. A. Hossen, M. A. A. Mamun, Y. M. Amin, and S. M. Ishfaq, “Implementation and performance comparison for two versions of eye tracking based robotic arm movement,” in *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Feb. 2017, pp. 203–208.
 - [46] Z. Kong, S. Rao, H. Yang, W. Lan, Y. Leng, and S. Ge, “Eye-tracking-based robotic arm control system,” in *2022 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI)*, Jul. 2022, pp. 663–667.
 - [47] V. Krishna Sharma, K. Saluja, V. Mollyn, and P. Biswas, “Eye Gaze Controlled Robotic Arm for Persons with Severe Speech and Motor Impairment,” in *ACM Symposium on Eye Tracking Research and Applications*, ser. ETRA ’20 Full Papers. New York, NY, USA: Association for Computing Machinery, Jun. 2020, pp. 1–9.
 - [48] A. Dragomir, C. F. Pană, D. Cojocaru, and L. F. Manta, “Human-Machine Interface for Controlling a Light Robotic Arm by Persons with Special Needs,” in *2021 22nd International Carpathian Control Conference (ICCC)*, May 2021, pp. 1–6.
 - [49] J. R. Schultz, A. B. Slifkin, H. Yu, and E. M. Scheerer, “Proof-of-Concept: A Hands-Free Interface for Robot-Assisted Self-Feeding,” in *2022 International Conference on Rehabilitation Robotics (ICORR)*, Jul. 2022, pp. 1–6.
 - [50] Y.-S. Law-Kam Cio, “Asservissement d’un bras robotique d’assistance à l’aide d’un système de stéréo vision artificielle et d’un suiveur de regard,” Master’s thesis, École Polytechnique de Montréal, Aug. 2017.
 - [51] D. W. Hansen and Q. Ji, “In the Eye of the Beholder: A Survey of Models for Eyes and Gaze,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 478–500, Mar. 2010.

- [52] A. Kar and P. Corcoran, “A Review and Analysis of Eye-Gaze Estimation Systems, Algorithms and Performance Evaluation Methods in Consumer Platforms,” *IEEE Access*, vol. 5, pp. 16 495–16 519, 2017.
- [53] C. H. Morimoto and M. R. Mimica, “Eye gaze tracking techniques for interactive applications,” *Special Issue on Eye Detection and Tracking*, vol. 98, no. 1, pp. 4–24, Apr. 2005.
- [54] A. Poole and L. J. Ball, “Eye Tracking in Human-Computer Interaction and Usability Research : Current Status and Future Prospects,” 2004.
- [55] “Dual Purkinje Eyetrackers,” <https://ppw.kuleuven.be/home/english/research/lep/resources/purkinje>, (accessed 2019-11-05).
- [56] Y. Fu, “Regression Based Gaze Estimation with Natural Head Movement,” Ph.D. dissertation, 2015.
- [57] “Dark and bright pupil tracking,” <https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/what-is-dark-and-bright-pupil-tracking/>, 8/10/2015 6:22:21 PM, (accessed 2019-09-17).
- [58] J. Park, T. Jung, and K. Yim, “Implementation of an Eye Gaze Tracking System for the Disabled People,” in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, Mar. 2015, pp. 904–908.
- [59] P. Blignaut, “Mapping the Pupil-Glint Vector to Gaze Coordinates in a Simple Video-Based Eye Tracker,” *Journal of Eye Movement Research*, vol. 7, Mar. 2014.
- [60] J. J. Cerrolaza, A. Villanueva, and R. Cabeza, “Taxonomic Study of Polynomial Regressions Applied to the Calibration of Video-oculographic Systems,” in *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications*, ser. ETRA '08. New York, NY, USA: ACM, 2008, pp. 259–266.
- [61] C. Ma, K.-A. Choi, B.-D. Choi, and S.-J. Ko, “Robust remote gaze estimation method based on multiple geometric transforms,” *Optical Engineering*, vol. 54, no. 8, p. 083103, Aug. 2015.
- [62] E. D. Guestrin and M. Eizenman, “General theory of remote gaze estimation using the pupil center and corneal reflections,” *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 6, pp. 1124–1133, Jun. 2006.

- [63] C. Hennessey, “A Single Camera Eye-Gaze Tracking System with Free Head Motion,” in *In Proceedings of ETRA: Eye Tracking Research and Applications Symposium*. ACM Press, 2006, pp. 87–94.
- [64] A. Meyer, M. Böhme, T. Martinetz, and E. Barth, “A Single-Camera Remote Eye Tracker,” in *Perception and Interactive Technologies*, ser. Lecture Notes in Computer Science, E. André, L. Dybkjær, W. Minker, H. Neumann, and M. Weber, Eds. Springer Berlin Heidelberg, 2006, pp. 208–211.
- [65] Sheng-Wen Shih and Jin Liu, “A novel approach to 3-D gaze tracking using stereo cameras,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 234–245, Feb. 2004.
- [66] X. Zhou, H. Cai, Z. Shao, H. Yu, and H. Liu, “3D eye model-based gaze estimation from a depth sensor,” in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Qingdao, China: IEEE, Dec. 2016, pp. 369–374.
- [67] D. H. Yoo and M. J. Chung, “A novel non-intrusive eye gaze estimation using cross-ratio under large head motion,” *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 25–51, Apr. 2005.
- [68] P. Koutras and P. Maragos, “Estimation of eye gaze direction angles based on active appearance models,” in *2015 IEEE International Conference on Image Processing (ICIP)*, Sep. 2015, pp. 2424–2428.
- [69] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models,” in *Computer Vision — ECCV’98*, ser. Lecture Notes in Computer Science, H. Burkhardt and B. Neumann, Eds. Springer Berlin Heidelberg, 1998, pp. 484–498.
- [70] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, “Eye Tracking for Everyone,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 2176–2184.
- [71] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “MPIIGaze: Real-World Dataset and Deep Appearance-Based Gaze Estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 162–175, Jan. 2019.
- [72] S. Park, S. De Mello, P. Molchanov, U. Iqbal, O. Hilliges, and J. Kautz, “Few-Shot Adaptive Gaze Estimation,” *arXiv:1905.01941 [cs]*, Oct. 2019.

- [73] S. Park, A. Spurr, and O. Hilliges, “Deep Pictorial Gaze Estimation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 721–738.
- [74] A. Papoutsaki, P. Sangkloy, J. Laskey, N. Daskalova, J. Huang, and J. Hays, “WebGazer: Scalable Webcam Eye Tracking Using User Interactions,” Jul. 2016.
- [75] A. L. Yuille, P. W. Hallinan, and D. S. Cohen, “Feature extraction from faces using deformable templates,” *International Journal of Computer Vision*, vol. 8, no. 2, pp. 99–111, Aug. 1992.
- [76] V. Kazemi and J. Sullivan, “One millisecond face alignment with an ensemble of regression trees,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1867–1874.
- [77] F. Dornaika and J. Orozco, “Real time 3D face and facial feature tracking,” *Journal of Real-Time Image Processing*, vol. 2, no. 1, pp. 35–44, Oct. 2007.
- [78] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–511–I–518 vol.1.
- [79] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
- [80] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, Jun. 2005, pp. 886–893 vol. 1.
- [81] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object Detection with Discriminatively Trained Part-Based Models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
- [82] “OpenCV,” <https://opencv.org/>, (accessed 2019-09-19).
- [83] “Dlib C++ Library,” <http://dlib.net/>, Sep. 2019, (accessed 2019-09-19).
- [84] D. E. King, “Max-Margin Object Detection,” *arXiv:1502.00046 [cs]*, Jan. 2015.
- [85] A. Geitgey, “Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning,” <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>, Nov. 2018, (accessed 2019-09-20).

- [86] O. N. A. AL-Allaf, “Review of Face Detection Systems Based Artificial Neural Networks Algorithms,” *The International journal of Multimedia & Its Applications*, vol. 6, no. 1, pp. 1–16, Feb. 2014.
- [87] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, Jun. 2014, pp. 1701–1708.
- [88] A. Sharifara, M. S. M. Rahim, and Y. Anisi, “A general review of human face detection including a study of neural networks and Haar feature-based cascade classifier in face detection,” in *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, Aug. 2014, pp. 73–78.
- [89] S. Zhang, X. Wang, Z. Lei, and S. Z. Li, “Faceboxes: A CPU real-time and accurate unconstrained face detector,” *Neurocomputing*, vol. 364, pp. 297–309, Oct. 2019.
- [90] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, “BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs,” *arXiv:1907.05047 [cs]*, Jul. 2019.
- [91] C. Li, “A Gentle Introduction to Gradient Boosting,” p. 80.
- [92] S. Ren, X. Cao, Y. Wei, and J. Sun, “Face Alignment at 3000 FPS via Regressing Local Binary Features,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, Jun. 2014, pp. 1685–1692.
- [93] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, “Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs,” *arXiv:1907.06724 [cs]*, Jul. 2019.
- [94] “Face Mesh,” https://google.github.io/mediapipe/solutions/face_mesh.html, (accessed 2021-07-15).
- [95] “TensorFlow,” <https://www.tensorflow.org/>, (accessed 2019-09-26).
- [96] “Tensorflow/models,” <https://github.com/tensorflow/models>, (accessed 2019-09-26).
- [97] “YOLO: Real-Time Object Detection,” <https://pjreddie.com/darknet/yolo/>, (accessed 2019-09-26).

- [98] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” Jun. 2015.
- [99] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2980–2988.
- [100] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255.
- [101] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft COCO: Common Objects in Context,” Feb. 2015.
- [102] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes Challenge: A Retrospective,” *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.
- [103] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, “BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning,” Apr. 2020.
- [104] A. Gupta, P. Dollár, and R. Girshick, “LVIS: A Dataset for Large Vocabulary Instance Segmentation,” Sep. 2019.
- [105] Y. Zhong, D. Zhao, D. Cheng, J. Zhang, and D. Tian, “A Fast and Precise Plane Segmentation Framework for Indoor Point Clouds,” *Remote Sensing*, vol. 14, no. 15, p. 3519, Jan. 2022.
- [106] R. Szeliski, *Computer Vision: Algorithms and Applications*, ser. Texts in Computer Science. London: Springer, 2011.
- [107] “3D reconstruction,” *Wikipedia*, Sep. 2019.
- [108] H. Hirschmuller, “Stereo Processing by Semiglobal Matching and Mutual Information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb. 2008.
- [109] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. A Bradford Book, May 2005.

- [110] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path Planning and Trajectory Planning Algorithms: A General Overview," in *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, ser. Mechanisms and Machine Science, G. Carbone and F. Gomez-Bravo, Eds. Cham: Springer International Publishing, 2015, pp. 3–27.
- [111] S. M. LaValle, *Planning Algorithms*. Cambridge ; New York: Cambridge University Press, 2006.
- [112] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. New York: Wiley, 2004.
- [113] Yajue Yang, Jia Pan, and Weiwei Wan, "Survey of optimal motion planning," *IET Cyber-Systems and Robotics*, vol. 1, no. 1, pp. 13–19, Jun. 2019.
- [114] M. M. Costa and M. F. Silva, "A Survey on Path Planning Algorithms for Mobile Robots," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Apr. 2019, pp. 1–7.
- [115] T. Kröger, "Literature Survey: Trajectory Generation in and Control of Robotic Systems," in *On-Line Trajectory Generation in Robotic Systems: Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events*, ser. Springer Tracts in Advanced Robotics, T. Kröger, Ed. Berlin, Heidelberg: Springer, 2010, pp. 11–31.
- [116] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, Dec. 2016.
- [117] M. G. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, Feb. 2018.
- [118] I. Noreen, A. Khan, and Z. Habib, "Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions," 2016.
- [119] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of Robot 3D Path Planning Algorithms," *J. Control Sci. Eng.*, vol. 2016, pp. 5–, Mar. 2016.
- [120] G. Picard-Krashevski, "Optimisation des algorithmes de planification et d'évitement d'obstacle d'un bras robotique," Rapport, Aug. 2018.

- [121] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, “Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization,” in *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, Jun. 2013.
- [122] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, “Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs,” in *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016.
- [123] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574.
- [124] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “CHOMP: Gradient optimization techniques for efficient motion planning,” in *2009 IEEE International Conference on Robotics and Automation*. Kobe: IEEE, May 2009, pp. 489–494.
- [125] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, “CHOMP: Covariant Hamiltonian optimization for motion planning,” *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, Aug. 2013.
- [126] “CHOMP Planner - moveit_tutorials Kinetic documentation,” http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/chomp_planner/chomp_planner_tutorial.html, (accessed 2019-11-22).
- [127] “MoveIt Motion Planning Framework,” <https://moveit.ros.org/>, (accessed 2019-11-21).
- [128] “The Open Motion Planning Library,” <http://ompl.kavrakilab.org/>, (accessed 2019-11-21).
- [129] “Jetson Xavier AGX: Deploy AI-Powered Autonomous Machines at Scale,” <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/>, (accessed 2022-12-28).
- [130] “Kinova | Empowering humanity through robotics : Kinova,” <http://www.kinovarobotics.com/>, (accessed 2018-05-04).
- [131] “ROS: Home,” <https://www.ros.org/>, (accessed 2022-12-28).
- [132] “Fawkesrobotics/libkindrv: Kinova Jaco Driver,” <https://github.com/fawkesrobotics/libkindrv>, (accessed 2022-12-28).

- [133] K. Kraflka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, “Eye Tracking for Everyone,” *arXiv:1606.05814 [cs]*, Jun. 2016.
- [134] M. Karrenbach, D. Boe, A. Sie, R. Bennett, and E. Rombokas, “Improving Automatic Control of Upper-Limb Prosthesis Wrists Using Gaze-Centered Eye Tracking and Deep Learning,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 340–349, 2022.
- [135] D. P. McMullen, G. Hotson, K. D. Katyal, B. A. Wester, M. S. Fifer, T. G. McGee, A. Harris, M. S. Johannes, R. J. Vogelstein, A. D. Ravitz, W. S. Anderson, N. V. Thakor, and N. E. Crone, “Demonstration of a Semi-Autonomous Hybrid Brain–Machine Interface Using Human Intracranial EEG, Eye Tracking, and Computer Vision to Control a Robotic Upper Limb Prosthetic,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 784–796, Jul. 2014.
- [136] Y. Cheng, H. Wang, Y. Bao, and F. Lu, “Appearance-based Gaze Estimation With Deep Learning: A Review and Benchmark,” *arXiv:2104.12668 [cs]*, Apr. 2021.
- [137] E. Finnigan, “Eye gaze tracking for assistive devices,” Ph.D. dissertation, EECS Department, University of California, Berkeley, May 2019.
- [138] T. Fischer, H. J. Chang, and Y. Demiris, “RT-GENE: Real-Time Eye Gaze Estimation in Natural Environments,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 334–352.
- [139] S. Park, X. Zhang, A. Bulling, and O. Hilliges, “Learning to find eye region landmarks for remote gaze estimation in unconstrained settings,” in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, ser. ETRA ’18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 1–10.
- [140] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “Appearance-Based Gaze Estimation in the Wild,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4511–4520, Jun. 2015.
- [141] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks,” *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.
- [142] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556 [cs]*, Apr. 2015.

- [143] A. Adouani, W. M. B. Henia, and Z. Lachiri, “Comparison of Haar-like, HOG and LBP approaches for face detection in video sequences,” in *2019 16th International Multi-Conference on Systems, Signals Devices (SSD)*, Mar. 2019, pp. 266–271.
- [144] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” *arXiv:1704.04861 [cs]*, Apr. 2017.
- [145] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” *arXiv:1801.04381 [cs]*, Mar. 2019.
- [146] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, “MediaPipe: A Framework for Building Perception Pipelines,” *arXiv:1906.08172 [cs]*, Jun. 2019.
- [147] K. Sokal, “MediaPipe 3D Face Transform,” Sep. 2020.
- [148] X. Zhang, Y. Sugano, and A. Bulling, “Revisiting Data Normalization for Appearance-Based Gaze Estimation,” in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, ser. ETRA ’18. New York, NY, USA: Association for Computing Machinery, 2018.
- [149] —, “Evaluation of Appearance-Based Methods and Implications for Gaze-Based Applications,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’19. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 1–13.
- [150] A. Ablavatski, A. Vakunov, I. Grishchenko, K. Raveendran, and M. Zhdanovich, “Real-time Pupil Tracking from Monocular Video for Digital Puppetry,” *arXiv:2006.11341 [cs]*, Jun. 2020.
- [151] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.
- [152] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

- [153] W.A. Falcon and .al, “PyTorch lightning,” *GitHub. Note:* <https://github.com/PyTorchLightning/pytorch-lightning>, vol. 3, 2019.
- [154] G. Bradski, “The OpenCV library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [155] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015.
- [156] L. Wright and N. Demeure, “Ranger21: A synergistic deep learning optimizer,” *arXiv:2106.13731 [cs]*, Aug. 2021.
- [157] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, “On the Variance of the Adaptive Learning Rate and Beyond,” *arXiv:1908.03265 [cs, stat]*, Oct. 2021.
- [158] T. Fischer, “Tobias-Fischer/rt_gene,” https://github.com/Tobias-Fischer/rt_gene, (accessed 2021-07-28).
- [159] “NVlabs/few_shot_gaze,” https://github.com/NVlabs/few_shot_gaze, Jul. 2021, (accessed 2021-07-28).
- [160] N. Norori, Q. Hu, F. M. Aellen, F. D. Faraci, and A. Tzovara, “Addressing bias in big data and AI for health care: A call for open science,” *Patterns*, vol. 2, no. 10, p. 100347, Oct. 2021.
- [161] X. Zhang, S. Park, T. Beeler, D. Bradley, S. Tang, and O. Hilliges, “ETH-XGaze: A Large Scale Dataset for Gaze Estimation under Extreme Head Pose and Gaze Variation,” Jul. 2020.
- [162] P. Kellnhofer, A. Recasens, S. Stent, W. Matusik, and A. Torralba, “Gaze360: Physically Unconstrained Gaze Estimation in the Wild,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6912–6921.
- [163] J. Qin, T. Shimoyama, and Y. Sugano, “Learning-by-Novel-View-Synthesis for Full-Face Appearance-Based 3D Gaze Estimation,” Apr. 2022.

- [164] C. Álvarez Casado and M. Bordallo López, “Real-time face alignment: Evaluation methods, training strategies and implementation optimization,” *Journal of Real-Time Image Processing*, vol. 18, no. 6, pp. 2239–2267, Dec. 2021.
- [165] “OpenCV: Perspective-n-Point (PnP) pose computation,” https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html, (accessed 2022-07-01).
- [166] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, “The Ball-Pivoting Algorithm for Surface Reconstruction,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 04, pp. 349–359, Oct. 1999.
- [167] M. Matl, “Pyrender,” <https://github.com/mmatl/pyrender>, Jun. 2022, (accessed 2022-07-01).
- [168] C. Leblond-Menard and S. Achiche, “Non-Intrusive Real Time Eye Tracking Using Facial Alignment for Assistive Technologies,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 31, pp. 954–961, 2023.
- [169] A. Lebrasseur, Ms. Josiane Lettre OT, P. François Routhier PEng, P. Philippe S. Archambault OT, and P. Alexandre Campeau-Lecours PEng, “Assistive robotic arm: Evaluation of the performance of intelligent algorithms,” *Assistive Technology*, May 2019.
- [170] M. Hildebrand, F. Bonde, R. V. N. Kobborg, C. Andersen, A. F. Norman, M. Thøgersen, S. H. Bengtson, S. Dosen, and L. N. S. A. Struijk, “Semi-Autonomous Tongue Control of an Assistive Robotic Arm for Individuals with Quadriplegia,” in *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, Jun. 2019, pp. 157–162.
- [171] A. Mohebbi, “Human-Robot Interaction in Rehabilitation and Assistance: A Review,” *Current Robotics Reports*, vol. 1, no. 3, pp. 131–144, Sep. 2020.
- [172] S. Du, F. Wang, G. Zhou, J. Li, L. Yang, and D. Wang, “Vision-Based Robotic Manipulation of Intelligent Wheelchair with Human-Computer Shared Control,” in *2021 33rd Chinese Control and Decision Conference (CCDC)*, May 2021, pp. 3252–3257.
- [173] Y. Xu, C. Ding, X. Shu, K. Gui, Y. Bezsudnova, X. Sheng, and D. Zhang, “Shared control of a robotic arm using non-invasive brain-computer interface and computer vision guidance,” *Robotics and Autonomous Systems*, vol. 115, pp. 121–129, May 2019.

- [174] C. Leblond-Menard, G. Picard-Krashevski, and S. Achiche, “Semi-synthetic dataset augmentation for application-specific gaze estimation,” *unpublished*, Dec. 2022.
- [175] B. Iglewicz and D. C. Hoaglin, *How to Detect and Handle Outliers*. ASQC Quality Press, 1993.
- [176] M. Valdenegro-Toro and D. Saromo, “A Deeper Look into Aleatoric and Epistemic Uncertainty Disentanglement,” Apr. 2022.
- [177] “Developing depth sensing applications - collision avoidance, object detection, volumetric capture and more,” <https://www.intelrealsense.com/sdk-2/>, (accessed 2022-10-21).
- [178] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv:1801.09847*, 2018.
- [179] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD’96. Portland, Oregon: AAAI Press, Aug. 1996, pp. 226–231.
- [180] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, “MMDetection: Open MMLab detection toolbox and benchmark,” *arXiv preprint arXiv:1906.07155*, 2019.
- [181] T. Barnes, “Fast, Branchless Ray/Bounding Box Intersections - tavianator.com,” https://tavianator.com/2011/ray_box.html, (accessed 2022-11-23).
- [182] M. Matl, “Urdfpy,” <https://github.com/mmatl/urdfpy>, Nov. 2022, (accessed 2022-11-24).
- [183] J. Pan, S. Chitta, and D. Manocha, “FCL: A general purpose library for collision and proximity queries,” in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3859–3866.
- [184] C. Leblond-Ménard, “Cedriclmenard/python-fcl-octomap at features/octomap-pointcloud,” <https://github.com/cedriclmenard/python-fcl-octomap/tree/features/octomap-pointcloud>, (accessed 2022-11-24).
- [185] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “OctoMap: An efficient probabilistic 3D mapping framework based on octrees,” *Autonomous Robots*, 2013.

- [186] E. Coumans and Y. Bai, “PyBullet, a Python module for physics simulation for games, robotics and machine learning,” 2016.
- [187] S. M. LaValle, “Rapidly-exploring random trees : A new tool for path planning,” *The annual research report*, 1998.
- [188] C. Reed Garrett, “Motion-planners,” <https://github.com/caelan/motion-planners>, Nov. 2022, (accessed 2022-11-25).

APPENDIX A RESEARCH PROTOCOL (FRENCH)

Protocole de recherche

Comparaison de systèmes de contrôle d'un bras robotique d'assistance

Expérimentation menée à
Polytechnique Montréal

Étudiant:
Cédric Leblond-Ménard
Directeur :
Sofiane Achiche

4 novembre 2022

Protocole de recherche

Table des matières

1. Justification de l'étude	3
2. Objectifs de recherche	4
Objectif général	4
Objectifs spécifiques	4
3. Méthodologie	4
Échantillonnage et recrutement	4
Consentement	5
Lieu de l'expérience	5
Montage expérimental	5
Acquisition des données	6
Outils d'acquisition	6
Questionnaire	6
Confidentialité	7
Comparaison	7
Précautions	7
Suivi	7
Risques et inconvénients	7
Avantages et bénéfices	7
4. Contribution attendue	7
Résultats scientifiques	7
Résultats pratiques	8
5. Référence visuelle de l'installation	8
6. Références	9

Protocole de recherche

1. Justification de l'étude

Récemment, les bras robotiques d'assistance (BRAs) ont attiré l'intérêt des cliniques, des industries et des centres de recherche. Les BRAs les plus populaires sont le bras Manus qui est le prédécesseur du iARM (Exact dynamics, Didam, Pays-Bas) [1], le bras WAM (Barrett Technology, LCC, Newton, États-Unis) [2], le bras DLR LWR-III (DLR Robotics and Mechatronics center, Weichselbaum, Allemagne) [3] et les bras JACO et MICO (Kinova Robotics, Boisbriand, Canada) [4]. Les bénéfices de l'utilisation du BRA pour des personnes ayant des problèmes de motricités des membres supérieurs ont déjà été prouvés [5-9]. Néanmoins, un des défis principaux des BRAs est d'être le plus versatile possible afin d'assister le plus de personnes avec divers problèmes de motricité des membres supérieurs incluant les dystrophies musculaires, la paralysie cérébrale et la tétraplégie. Ainsi, il sera possible de les aider à retrouver la majorité de leur autonomie et d'améliorer leur qualité de vie.

Afin de surmonter ce défi, une interface humain-machine (IHM) adéquate doit être utilisée afin de contrôler le BRA. En ce moment, la technologie populaire pour contrôler les BRAs est une manette style « joystick » qui est idéal pour des individus ayant des problèmes de motricité moins sévère telle que les stades préliminaires de la dystrophie musculaire. Par contre, la technologie n'est pas adaptée aux individus ayant des problèmes de motricité des membres supérieurs sévères tels que les stades avancés de la dystrophie musculaire ou la spasticité sévère. En effet, la plupart des BRAs ont plus de degrés de liberté (DDL) que le nombre d'axes d'utilisation des manettes, c'est-à-dire, 6 DDL vs. 3 axes.

L'utilisation d'une manette apporte deux problèmes principaux. Le premier problème est que le BRA n'a aucune information de l'espace de travail. En permettant au BRA d'accéder à l'information de l'espace de travail, le BRA devient conscient des configurations possibles qui peuvent être adoptées ainsi que les contraintes qui doivent être respectées. Afin de remédier à ce problème, plusieurs types de capteurs peuvent être utilisés pour obtenir l'espace de travail en trois dimensions.

Le deuxième problème est le nombre d'axes d'utilisation de la manette. En contrôlant un BRA à 6 DDL à l'aide de 3 axes, l'utilisation se complique et l'exécution de tâche est grandement ralentit. Il est possible de réduire le nombre de modes de contrôle en remplaçant la manette par une IHM plus adéquate et intuitive, permettant de réduire de temps d'exécution d'une tâche, d'améliorer le taux de réussite et d'augmenter la satisfaction d'utilisation d'un bras robotique d'assistance.

L'étude présentée compare deux solutions permettant de contrôler un BRA, soit l'utilisation d'une manette commune telle que fournie avec les BRA et un système utilisant le suivi du regard de l'utilisateur, afin de trouver le système de contrôle optimal pour les personnes ayant des difficultés motricité des membres supérieurs.

Protocole de recherche

2. Objectifs de recherche

Objectif général

L'objectif principal est de réduire le temps d'exécution et d'augmenter le taux de réussite de tâches à l'aide d'un système d'acquisition de l'espace de travail et d'une IHM basée sur la direction du regard dans diverses situations naturelles (extérieures et intérieures).

Objectifs spécifiques

Les objectifs spécifiques sont :

1. Intégrer un système de reconnaissance et détection d'objets devant l'utilisateur.
2. Intégrer une IHM au BRA basée sur le suivi du regard.
3. Développer un système de contrôle du BRA à l'aide du système d'acquisition et de l'IHM.
4. Comparer la solution à celle commune (manette).

3. Méthodologie

Un système de contrôle d'un BRA a déjà été conçu au sein du laboratoire de Conception des Systèmes Intelligents et Mécatroniques (*CoSIM*), du professeur S. Achiche de Polytechnique Montréal. Afin de savoir à quel point ce système est viable, il faut le comparer avec d'autres systèmes tels que l'utilisation d'une manette.

Les participants de ce projet de recherche auront deux étapes à faire. La première étant d'effectuer une série de tâche à l'aide d'un système de contrôle. La deuxième étape est de refaire cette même série de tâche à l'aide d'un autre système de contrôle. Les essais seront supervisés par Cédric Leblond Ménard, candidat au doctorat. Finalement, un bref questionnaire sera fourni.

Échantillonnage et recrutement

L'échantillonnage ne comportera qu'un seul groupe. Celui-ci sera représenté par des individus âgés de plus de 18 ans sans handicap. La taille de l'échantillon sera de 30 personnes et le recrutement des personnes sera fait dans l'entourage des chercheurs, en demandant aux proches et amis. Ce choix de grandeur d'échantillon est fait pour limiter l'ampleur du projet et assurer sa complétion dans des temps raisonnables.

Considérant l'horizon à court terme du projet et la difficulté à recruter et à mettre en place les installations nécessaires pour accueillir ce type de participants, l'étude ne comportera aucun groupe d'individus ayant un handicap.

Le recrutement se fera par une annonce ou message Facebook contenant l'information suivante :

Bonjour,

Dans le cadre de mes études doctorales, notre laboratoire recherche des participants à une étude sur l'utilisation d'un système de suivi du regard pour les bras robotiques

Protocole de recherche

d'assistance. Il s'agit d'une expérience qui durera moins d'une heure et qui vous demandera de contrôler à l'aide d'une manette et d'un système de suivi du regard les mouvements d'un bras robotique d'adaptation, dans le but d'analyser la pertinence d'un tel système. Il s'agit d'une étude approuvée par le Comité d'Éthique de la Recherche de Polytechnique.

Pour plus d'information ou pour montrer votre intérêt, veuillez me contacter.

Consentement

Tout participant qui a été approché par le responsable et qui a montré un intérêt initial se verra recevoir le protocole de recherche ainsi que le formulaire de consentement. Celui-ci pourra alors le lire et prendre une décision informée. Si le potentiel participant décide d'accepter, le responsable ainsi que le participant se rencontreront pour des explications plus approfondies et pour répondre aux questions potentielles que pourrait avoir le participant. S'il décide d'accepter de participer, le formulaire de consentement sera alors signé par le participant ainsi que le responsable.

L'expérience à laquelle le participant sera présent sera au moins une semaine après la réception et signature du formulaire de consentement, pour assurer que celui-ci ait le temps nécessaire pour poser des questions et s'informer.

Une attention particulière sera portée pour s'assurer que la relation entre les participants et les chercheurs n'influencent pas le consentement en mettant bien au clair que la participation est complètement volontaire et n'est en aucun point requise.

Lieu de l'expérience

L'expérience sera faite en partie à Polytechnique Montréal dans les locaux du Pr. Achiche. L'environnement extérieur sera simulé en accomplissant les tâches proches d'une fenêtre (illumination du soleil).

Le participant sera seul avec les chercheurs lors de l'expérience, à moins qu'une autre personne accompagne le participant, à la demande de ce dernier.

Le local permet de garder la confidentialité du participant, car aucune caméra autre que celle nécessaire à l'expérience (la caméra 3D ainsi que la caméra regardant le visage du participant) n'est présente dans le local et aucune de celles présentes n'enregistrent d'image.

Montage expérimental

Le montage expérimental est une structure de métal sur laquelle sont montées : une caméra pointant vers l'utilisateur, une caméra de profondeur pointant vers l'avant, un bras robotique d'adaptation et sa manette ainsi qu'un petit ordinateur nécessaire au système de contrôle par le suivi du regard. Aucune partie de ce montage est en contact avec le participant et ce à tout moment de l'expérience, outre la manette qui devra être utilisée et un clavier qui sera utilisé pour activer le système de contrôle.

Protocole de recherche

Le bras robotique comporte des sécurités et ne peut se déplacer qu'à basse vitesse. Il s'agit d'un bras robotique d'assistance spécialement conçu pour ce type de tâche.

Acquisition des données

Une série de tâches devra être accomplie par le participant. Donc, pour chacune des tâches, le participant devra faire trois essais avec le premier système de contrôle. Par la suite, le participant devra refaire toutes les essais avec le deuxième système de contrôle. Pour tous les essais de l'expérience, le temps d'exécution et le taux de réussite de la tâche sera pris en note par le superviseur de l'expérience. Comme mentionner ci-haut, les systèmes de contrôle sont composés de dispositifs d'acquisition de l'espace de travail et de dispositifs d'IHM. Ces deux types de dispositifs sont non invasifs. L'IHM est dans un cas qu'une simple manette et dans l'autre cas une caméra qui n'enregistre aucune image sur le disque de stockage.

Outils d'acquisition

Les données seront colligées sur un tableur (Excel) et le temps d'exécution serait mesurée à l'aide d'un chronomètre.

Questionnaire

Les participants doivent à la fin de l'expérience répondre à quelques questions concernant leur expérience avec chacun des systèmes, soit les questions suivantes :

1. Sur une échelle de 1 à 5, 1 étant très difficile et 5 étant très facile, le système de contrôle est-il facile à utiliser?
2. Sur une échelle de 1 à 5, 1 étant très lent et 5 étant très rapide, le système de contrôle est-il rapide à utiliser?
3. Sur une échelle de 1 à 5, 1 étant pas du tout efficace et 5 étant très efficace, le système de contrôle vous paraît-il efficace?
4. Sur une échelle de 1 à 5, 1 étant aucunement et 5 étant complètement, le système de contrôle permet-il d'accomplir la tâche que vous vous attendiez?
5. Sur une échelle de 1 à 5, 1 étant aucunement et 5 étant complètement, êtes-vous satisfait quant à l'utilisation de ce système pour prendre des objets?
6. Sur une échelle de 1 à 5, 1 étant aucunement et 5 étant totalement, trouvez-vous le système de contrôle sécuritaire?
7. Avez-vous d'autres commentaires quant à l'utilisation.
8. Si vous êtes à l'aise à partager cette information, quelle est votre âge?
9. Si vous êtes à l'aise à partager cette information, quelle est votre genre?

Ces questions seront demandées à l'oral et colligées dans un tableur logiciel.

Protocole de recherche

Confidentialité

Aucune image ni information personnelle ne sera enregistrée lors de cette expérience. Seules les valeurs de temps et de succès ainsi que les réponses au questionnaire seront gardées, sans information personnelle attachée. Il ne sera donc pas possible d'identifier les participants.

Les documents de consentement seront détruits après la durée minimale requise de 7 ans.

Comparaison

Pour chaque tâche, une comparaison du temps d'exécution moyen et du taux de réussite sera faite entre les deux systèmes de contrôle. De plus, les informations de satisfaction seront colligées pour but de présentation dans les résultats des travaux.

Précautions

Aucune précaution préalable n'a besoin d'être prise par les participants.

Suivi

Aucun suivi ne sera nécessaire après l'expérience.

Risques et inconvénients

Cette expérience comporte très peu de risque. Dans les pires circonstances, le bras robotique pourrait être amené à s'approcher de l'utilisateur. Pour s'assurer qu'aucun contact ne soit possible, l'expérimentateur sera prêt à arrêter le bras robotique. Celui-ci est spécialement conçu pour des tâches d'assistance et se déplace à basse vitesse et possède des protections intrinsèques pour limiter sa force.

De plus, l'utilisateur aura accès à un bouton pour arrêter le bras robotique à tout moment.

Le temps de réalisation de l'étude est le seul inconvénient pour l'utilisateur. Celui-ci sera minimisé en s'assurant que l'expérimentateur soit bien préparé à la réalisation des essais.

Avantages et bénéfices

Aucun avantage et bénéfice monétaire ne sera accordé. Les participants contribueront au développement d'une solution qui pourra potentiellement aider les personnes atteintes de troubles moteurs des membres supérieurs.

4. Contribution attendue

Vue la pertinence du projet envers une clientèle cible, des résultats scientifiques et pratiques sont prévus.

Résultats scientifiques

La comparaison de systèmes de contrôle permettra de diffuser les résultats de la recherche par l'entremise d'articles dans des revues scientifiques. De plus, les résultats pourront être

Protocole de recherche

présentés à travers des congrès scientifiques ainsi que des présentations par affiches ou des présentations orales.

Résultats pratiques

Considérant que le système contrôle a été développé au sein de CoSIM, la présentation de résultats positives pourrait amener le perfectionnement du système de contrôle pour éventuellement le commercialiser et permettre aux personnes ayant des problèmes de motricité des membres supérieurs d'en bénéficier.

5. Référence visuelle de l'installation



Protocole de recherche

6. Références

- [1] Exact Dynamics. <http://www.exactdynamics.nl/site/>. Accessed 2017
- [2] Barrett Technology LCC: WAM Arm. <http://www.barrett.com/products-arm.htm>. Accessed 2017
- [3] DLR Robotics and Mechatronics Center: Light-Weight Robotics.
http://www.dlr.de/rmc/rm/en/desktopdefault.aspx/tabid-3803/6175_read-8963/. Accessed 2017
- [4] Kinova Robotics: Assistive robotics. <http://www.kinovarobotics.com/assistive-robotics/products/robot-arms/>. Accessed 2017
- [5] Routhier, F., Archambault, P., Cyr, M.-C., Maheu, V., Lemay, M., G  linas, I.: BENEFITS OF JACO ROBOTIC ARM ON INDEPENDENT LIVING AND SOCIAL PARTICIPATION: AN EXPLORATORY STUDY.
- [6] Maheu, V., Archambault, P.S., Frappier, J., Routhier, F.: Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities. In: Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on 2011, pp. 1-5. IEEE
- [7] Chung, C.-S., Wang, H., Cooper, R.A.: Functional assessment and performance evaluation for assistive robotic manipulators: Literature review. The journal of spinal cord medicine **36**(4), 273-289 (2013).
- [8] Groothuis, S.S., Stramigioli, S., Carloni, R.: Lending a helping hand: Toward novel assistive robotic arms. IEEE robotics & automation magazine **20**(1), 20-29 (2013).
- [9] Campeau-Lecours, A., Maheu, V., Lepage, S., Lamontagne, H., Latour, S., Paquet, L., Hardie, N.: JACO Assistive Robotic Device: Empowering People With Disabilities Through Innovative Algorithms. In: Rehabilitation Engineering and Assistive Technology Society of North America (RESNA) annual conference 2016