VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

CAPSTONE PROJECT

HAND GESTURE RECOGNITION FOR GAME-BASED HAND REHABILITATION

MAJOR: COMPUTER SCIENCE

COMMITTEE : COMPUTER SCIENCE 2
SUPERVISORS : ASSOC. PROF. QUAN THANH THO, PHD
MR. MAI DUC TRUNG, MSC
REVIEWER : DR. NGUYEN HUA PHUNG

STUDENT 1 : HO TRI KHANG 1952069
STUDENT 2 : TRAN TIEN PHAT 1952386
STUDENT 3 : VO NGOC SANG 1952430

HO CHI MINH CITY, JUNE 2023

HĐ2 CLC KHMT

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

KHOA:KH & KT Máy tính
BỘ MÔN:KHMT

NHIỆM VỤ LUẬN VĂN/ ĐỒ ÁN TỐT NGHIỆP

Chú ý: Sinh viên phải dán tờ này vào trang nhất của bản thuyết trình

HỌ VÀ TÊN: Ho Trí Kháng — MSSV: 1952069
HỌ VÀ TÊN: Trần Tiến Phát — MSSV: 1952386
HỌ VÀ TÊN: Võ Ngọc Sang — MSSV: 1952430
NGÀNH: Khoa học Máy tính (CLC) — LỚP: CC19KHM1, CC19KHM2

1. Đầu đề luận văn/ đồ án tốt nghiệp:
Hand gesture recognition for game-based hand rehabilitation

2. Nhiệm vụ (yêu cầu về nội dung và số liệu ban đầu):
- Conduct study on hand gesture recognition for game-based hand rehabilitation to design the system.
- Collect hand data to train and evaluate the system.
- Implement the system and apply it to play simple rehabilitation games.
- Develop interactive video games for hand rehabilitation.
- Evaluate the system based on selected criteria.
- Conduct a user study to evaluate the effectiveness of the system.

3. Ngày giao nhiệm vụ: 30/01/2023
4. Ngày hoàn thành nhiệm vụ: 09/06/2023
5. Họ tên giảng viên hướng dẫn: **Phần hướng dẫn:**
1) Quản Thành Thơ
2) Mai Đức Trung

Nội dung và yêu cầu LVTN/ ĐATN đã được thông qua Bộ môn.

Ngày ..... tháng ..... năm .....
CHỦ NHIỆM BỘ MÔN
(Ký và ghi rõ họ tên)

GIẢNG VIÊN HƯỚNG DẪN CHÍNH
(Ký và ghi rõ họ tên)

PGS. TS. QUẢN THÀNH THƠ

PHẦN DÀNH CHO KHOA, BỘ MÔN:
Người duyệt (chấm sơ bộ):
Đơn vị:
Ngày bảo vệ:
Điểm tổng kết:
Nơi lưu trữ LVTN/ĐATN:

HĐ2 CLC KHMT

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KH & KT MÁY TÍNH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Ngày tháng năm

PHIẾU ĐÁNH GIÁ LUẬN VĂN/ ĐỒ ÁN TỐT NGHIỆP

(Dành cho người hướng dẫn/phản biện)

1. Họ và tên SV: Ho Trí Kháng
MSSV: 1952069 Ngành (chuyên ngành): Khoa học Máy tính
Họ và tên SV: Trần Tiến Phát
MSSV: 1952386 Ngành (chuyên ngành): Khoa học Máy tính
Họ và tên SV: Võ Ngọc Sang
MSSV: 1952430 Ngành (chuyên ngành): Khoa học Máy tính

2. Đề tài: Hand gesture recognition for game-based hand rehabilitation

3. Họ tên người hướng dẫn/phản biện: Quản Thành Thơ

4. Tổng quát về bản thuyết minh:
Số trang: 162 Số chương: 6
Số bảng số liệu: 23 Số hình vẽ: 50
Số tài liệu tham khảo: 93 Phần mềm tính toán:
Hiện vật (sản phẩm)

5. Những ưu điểm chính của LV/ ĐATN:

- This work proposes a real-time hand gesture recognition system for game-based hand rehabilitation. By applying an enjoyable gamified approach, the proposed solution can be a great alternative to traditional hand rehabilitation approaches which are often tedious and unengaging.
- Compared to existing works, this work takes into consideration the factors that have large impact on hand rehabilitation, including the usability of the solution, the relevance of the adopted hand gestures to common hand rehabilitation exercises. This guarantees a positive impact on the recovery process of patients when they use the system.
- The proposed hand gesture recognition method is simple, real-time but still achieves impressive results on the self-generated datasets.
- The development of both the hand gesture recognition system and the interactive gaming application makes the system compact and ready for use, without relying on other systems.
- A user study was conducted on several elderly people to evaluate the effectiveness of the proposed system in real-life.
- The students have submitted a research paper for a student scientific conference

6. Những thiếu sót chính của LV/ĐATN:

- The design for game system should be given with more details
- The system evaluation needs to be elaborated with some metrics on machine learning aspect

7. Đề nghị: Được bảo vệ □ Bổ sung thêm để bảo vệ □ Không được bảo vệ □
8. Các câu hỏi SV phải trả lời trước Hội đồng:
a.

b.

c.

9. Đánh giá chung (bằng chữ: Xuất sắc, Giỏi, Khá, TB): Điểm : 9.5/10

Ký tên (ghi rõ họ tên)

Quản Thành Thơ

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KH & KT MÁY TÍNH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Ngày 11 tháng 06 năm 2023

PHIẾU CHẤM BẢO VỆ LVTN

(Dành cho người hướng dẫn/phản biện)

1. Họ và tên SV: Ho Tri Khang
   MSSV: 1952069 Ngành (chuyên ngành): KHMT
   Họ và tên SV: Tran Tien Phat
   MSSV: 1952386 Ngành (chuyên ngành): KHMT
   Họ và tên SV: Vo Ngoc Sang
   MSSV: 1952430 Ngành (chuyên ngành): KHMT
2. Đề tài: Hand Gesture Recognition for Game-Based Hand Rehabilitation
3. Họ tên người hướng dẫn/phản biện: Nguyễn Hứa Phùng
4. Tổng quát về bản thuyết minh:
   Số trang: 153 Số chương: 6
   Số tài liệu tham khảo: 93
5. Tổng quát về các bản vẽ:
6. Những ưu điểm chính của LVTN:

Students Ho Tri Khang, Tran Tien Phat and Vo Ngoc Sang implement the project to recognize hand gesture and apply it to implement a game for hand rehabilition. This group collects data from a device called Leap Motion Controller to extract hand feature. A component based on SVM is developed to classify poses which are passed to another component for recognizing hand gesture. This group also develop a simple game to apply these components for hand rehabilition. A simple study for the effectiveness of the game is also conducted.

7. Những thiếu sót chính của LVTN:

- The report just presented the UI of the game without the software development process and the students did not reply correctly about this process.

8. Đề nghị: Được bảo vệ ☒X Bổ sung thêm để bảo vệ ☐ Không được bảo vệ ☐
9. 3 câu hỏi SV phải trả lời trước Hội đồng:
10. Đánh giá chung (bằng chữ: giỏi, khá, TB):

Điểm :
Ho Tri Khang: 9
Tran Tien Phat: 8.5
Vo Ngoc Sang: 8.5

Ký tên (ghi rõ họ tên)

TS. Nguyễn Hứa Phùng

Declaration of Authenticity

We, hereby, declare that the work presented in this capstone project is the outcome of our research performed under the supervision and guidance of Assoc. Prof. Dr. Quan Thanh Tho and Mr. Mai Duc Trung, Faculty of Computer Science and Engineering, Ho Chi Minh University of Technology. The result of our research is legitimate and has not been published in any forms prior to this. All materials used within this researched are collected ourselves from various sources and are appropriately listed in the References section.

In any case of plagiarism, we stand by our actions and will be responsible for it. Ho Chi Minh city University of Technology therefore is not responsible for any copyright infringements conducted within our research.

Ho Chi Minh City, Jun 2023

Authors

Ho Tri Khang Tran Tien Phat Vo Ngoc Sang

Acknowledgements

First and foremost, we would like to express our gratitude to our supervisor, Assoc. Prof. Dr. Quan Thanh Tho, whose unending support and encouragement from the very beginning phase to the completion of the project have enabled us to develop a thorough understanding of the subject. This work would not have been possible without his insightful suggestions and motivations.

We would also like to send our gratitude to our classmates and friendly students from Ho Chi Minh University of Technology for their invaluable support, great ideas, and valuable suggestions throughout this work.

Our final gratitude is for our parents and family members who have raised us and supported us for all our pursues, with endless patience and inspiration.

Abstract

Hand rehabilitation is an influential aspect of post-injury recovery for many patients. During the hand rehabilitation process, the patients are required to practice repetitive movements to recover their hand functions. Traditional rehabilitation methods can be tedious and unengaging, leading to poor adherence to treatment plans. Video games have emerged as a potential tool to make rehabilitation more engaging and effective. Moreover, the rapid development of Artificial Intelligence and Machine Learning has led to the idea of applying hand gesture recognition in game-based hand rehabilitation. By playing interactive games, the patients will feel more enjoyable and motivated, thus encouraging the process of recovery. In this work, we propose a real-time skeleton-based hand gesture recognition system for game-based rehabilitation. The proposed system aims to support and encourage patients in practicing hand rehabilitation exercises via interesting rehabilitation games. Our system makes use of a Leap Motion Controller to extract skeletal information from the user's hand. A pose-based recognizer will then identify the key hand poses from the skeletal data to recognize the hand gesture. For the system to effectively support the hand rehabilitation process, a set of suitable hand gestures selected from common wrist and finger rehabilitation exercises is adopted. When a gesture is recognized, a corresponding action will be sent to the game environment to control the games. We also propose a gaming application consisting of three simple, interactive video games designed specifically for hand rehabilitation. Playing these games involves repetitively performing hand gestures related to rehabilitation exercises, which can help to speed up the recovery process. Our system provides an engaging and interactive way for patients to perform their rehabilitation exercises while also enjoying the benefits of

playing video games. We also conducted a user study with 10 individuals to evaluate the effectiveness of our system. The result showed that our system is interesting and is a potential solution for hand rehabilitation.

Keywords*: hand rehabilitation, hand gesture recognition, Machine Learning, interactive video games.*

Report Structure

Chapter 1: Introduction

Chapter 1 is an introduction of our work. This chapter presents the motivation for our hand gesture recognition system for game-based rehabilitaiton, the goals, scopes and the main contributions we made in this work.

Chapter 2: Literature Review

Chapter 2 presents a comprehensive literature review that explores previous research endeavors focused on the implementation of hand gesture recognition for game-based rehabilitation. The strengths and limitations in each study, and a comprehensive comparison of each study with our approach are also presented in this chapter.

Chapter 3: Theoretical Background

Chapter 3 provides essential background knowledge pertaining to our project. It delves into various aspects, including hand rehabilitation, mathematical principles, machine learning algorithms and evaluation metrics, all of which are directly applicable to the successful implementation of our project.

Chapter 4: Methodology

Chapter 4 elaborates our proposed hand gesture recognition system, presented our work on data collection and introduced the gaming application that we developed for game-based hand rehabilitation.

Chapter 5: Results

Chapter 5 provides a discussion of the obtained results, which includes various experiments on hand gesture recognition, test scenarios on the gaming application, and the user study results.

Chapter 6: Conclusion

Chapter 6 presents a thorough summary of our work, the limitations we faced and outlining our plans for future work.

References

References section presents all the documents cited in this work.

Appendix A: Assessment on the Working Process

Appendix A provides a detailed evaluation of the working process for each team member throughout the project.

Appendix B: Workload of Members

Appendix B presents the member workload in every parts of the project. The tasks performed by each members are clearly presented in this appendix.

Appendix C: Research Paper

Appendix C is the research paper we have submitted to the 12^{th} OISP Science and Technology Symposium for Students at Ho Chi Minh City University of Technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we present the problem statement and the motivation behind selecting this particular topic in Section 1.1, the goals we aim to achieve in Section 1.2, and the scopes of our project in Section 1.3. More importantly, we highlight the main contributions we have made throughout the course of this project in Section 1.4.

1.1 Problem Statement and Motivation

Hand rehabilitation has always been an important area of research in the field of physiotherapy [1, 2]. Hand injuries are quite common and can significantly affect a person's ability to perform daily activities, work, and participate in leisure activities.

Hand rehabilitation is an influential aspect of post-stroke recovery. According to the World Stroke Organization (WSO) [3] and the Global Burden of Disease (GBD) Study 2019 [4], stroke is the second leading cause of death (11.6% of total deaths globally) and the third leading cause of death and disability combined (5.7% of total DALYs[1]). As reported in [5], stroke can lead to upper limb dysfunction in up to 80% of survivors. Stroke survivors usually suffer from hand and upper extremity impairment which can significantly impede their ability to carry out daily activities, work, and leisure activities. Therefore, hand rehabilitation is necessarily important

[1]Disability-Adjusted Life Years.

for post-stroke patients to restore the strength, mobility, and coordination of their hands. However, post-stroke rehabilitation is a challenging process since it can be difficult, intensive and long depending on how adverse the stroke and which parts of the brain were damaged. Therefore, the process of recovery depends on how well the patients can commit to the treatment plan. Normally, stroke patients will follow a traditional rehabilitation method where they have to perform repetitive hand exercises under the instruction of a hand therapist. This approach, however, can be tedious and not engaging due to its repetitive nature. The patients may feel bored and lack of motivation to follow the treatment plan. Therefore, it is crucial to develop efficient and interesting rehabilitation methods that can engage to the patients and motivate them to follow the treatment plan.

In recent years, there is a growing interest in developing game-based approaches for hand rehabilitation [6, 7, 8]. By adopting modern technologies such as virtual reality [8, 9], augmented reality [10, 11] and advanced sensors for hand motion tracking such as Kinect [12, 13], LeapMotion [14, 15], game-based approaches are useful for creating an interactive environment where patients can enjoy the games while still participating in the hand function recovery process. This is done by performing repetitive hand movements while playing the games. Therefore, game-based rehabilitation is more enjoyable and interactive, enhancing patient engagement and participation in the treatment plan.

Hand gesture recognition has always been an active area of research in human-computer interaction. Recent studies have used hand gesture recognition as a key factor in game-based rehabilitation system to provide real-time and accurate recognition of gestures [16, 17]. With the advancement of Deep Learning recently, several deep neural network architectures have been developed for hand gesture recognition and achieved remarkable results on several benchmarks [18, 19]. However, in a system where real-time performance is strictly required, those models could be highly infeasible since they require large computational resources which can highly degrade the time performance of the system. Therefore, a simple yet effective recognition

approach could be a better choice for real-time systems.

In this work, we propose a real-time, skeleton-based hand gesture recognition for game-based hand rehabilitation. Our system uses a Leap Motion Controller to capture hand movement and extract hand skeletal data. Compared to several existing hand gesture recognition methods that rely on computational expensive deep learning models, we adopt a simple, real-time system which recognizes hand gestures by identifying key poses that appear during the gestures. Our pose-based recognition system can easily obtain real-time performance while still achieving high accuracy on our self-generated datasets. We also developed a gaming application consisting of 3 simple interactive games designed specifically for hand rehabilitation. Interacting with these games involves repetitively performing hand gestures, which simulates the common exercises used in hand rehabilitation. Therefore, patients can enjoy the games while still participating in the recovery process. Our work is inspired from a research called VirtualRehab [20], which contains several games using Kinect and Leap Motion for stroke patients to practice motor rehabilitation exercises.

In some of the previous work on hand gesture recognition for game-based rehabilitation, they use the hand gestures that have limited effects on hand recovery and are not suitable to apply in hand rehabilitation [21, 16, 22]. For example, in [21], they adopted the numeric gestures from “0” to “9” that are quite irrelevant to the gestures commonly encountered in hand therapy exercises. This introduces some drawbacks since performing the wrong movements can lead to even worse conditions. In our approach, we adopt a set of hand gestures that are carefully selected from a pool of common hand therapy exercises. Therefore our system is guaranteed to have positive effects on hand rehabilitation.

We also conducted a user survey with 10 individuals, both ordinary people and those who are having hand problems to verify the effectiveness of our system. The result shows that the proposed system is enjoyable and can be a potential solution to hand rehabilitation.

1.2 Goals

The main goal of this project is to propose a hand gesture recognition system for game-based hand rehabilitation. The system is designed to run in real-time and can recognize a set of commonly used exercises in hand rehabilitation. The system uses a Leap Motion Controller to capture hand movement and extract hand skeletal data.

Furthermore, we develop an engaging gaming application that features interactive games explicitly designed for hand rehabilitation. By combining the hand gesture recognition system with these specialized games, we aim to provide an innovative and effective solution for hand rehabilitation.

Through this project, we expect that our system will serve as a promising alternative to traditional hand rehabilitation approaches, providing a more immersive and effective experience for patients.

1.3 Scopes

It is important to list of the range of problems we are concerned with. The scopes of this project are as follows:

- Designing and implementing a skeleton-based, real-time hand gesture recognition algorithm that can recognize a set of commonly used hand gestures in rehabilitation exercises. The system uses a Leap Motion Controller to capture hand movement and extract hand skeletal data.
- Collecting hand gesture data to train and evaluate the performance of our system.
- Designing and developing a gaming application composed of interactive video games for game-based rehabilitation.
- Conducting a user study to evaluate the effectiveness of our system.

1.4 Contributions

The main contributionss in this work are as follow:

- We have successfully implemented a real-time hand gesture recognition system with all the predefined requirements. Our system works properly and achieved remarkable results on our two self-generated datasets.
- We have generated two hand datasets, a pose and a gesture dataset containing hand skeletal data recorded by the Leap Motion Controller to evaluate the performance of our system. These datasets can be contributed as an evaluation benchmarks for hand gesture recognition on Leap Motion data.
- We have conducted a user study with real users to study the effectiveness of our solution and achieved positive feedback.
- We have a paper submitted to the OISP Conference 2023. The paper abstract has been accepted by the conference.
- As game-based rehabilitation is still not popular in Vietnam, this project can be used as a great alternative approach to traditional rehabilitation methods in Vietnam.
- Compared with similar approaches in the world on hand gesture recognition for game-based rehabilitation, our system is **simple, fast and is designed to be more applicable to hand rehabilitation**. Moreover, the combination of both hand gesture recognition and interactive games into a single system makes our system an innovative and complete solution whereas some existing methods only focus on either hand gesture recognition or game development for hand rehabilitation.
- We already have a well-defined roadmap for expanding our research and enhancing our system beyond the completion of this capstone project. Our primary objective is to conduct extensive research aimed at refining our recognition system and documenting our findings through research papers, thereby making

valuable contributions to the research community. Additionally, we plan to scale up our work by conducting a larger and more comprehensive user study. To ensure the effectiveness of our solution, we intend to collaborate with hospitals and organizations specializing in hand rehabilitation in Vietnam. Through this collaboration, we aim to evaluate the impact of our solution and identify the areas of further improvement. Ultimately, we expect our work to make significant contributions to the field of hand rehabilitation not only in Vietnam but also on a global scale.

Chapter 2

Literature Review

This chapter presents the literature review on related works on hand gesture recognition in general and hand gesture recognition for hand rehabilitation specifically. Section 2.1 gives a general review on hand gesture recognition approaches. In Section 2.2, existing works on hand gesture recognition for hand rehabilitation are discussed. The advantages and disadvantages of each work are also presented.

2.1 Hand Gesture Recognition

Hand gesture recognition has always been an active research area in human-computer interaction[1] [23]. This section reviews numerous applications of hand gesture recognition and some common approaches to hand gesture recognition. At the end of this section, we also have some comments on existing hand gesture recognition approaches and the approach that we choose in this project. A summary of the applications and existing approaches to hand gesture recognition is shown in Figure 2.1

[1]Human-computer interaction refers to the multidisciplinary field that studies the design, development, and evaluation of interactive computing systems, focusing on the interaction between humans and computers. It investigates the ways in which users interact with computer systems, aiming to improve the usability, effectiveness, and overall user experience of such systems.

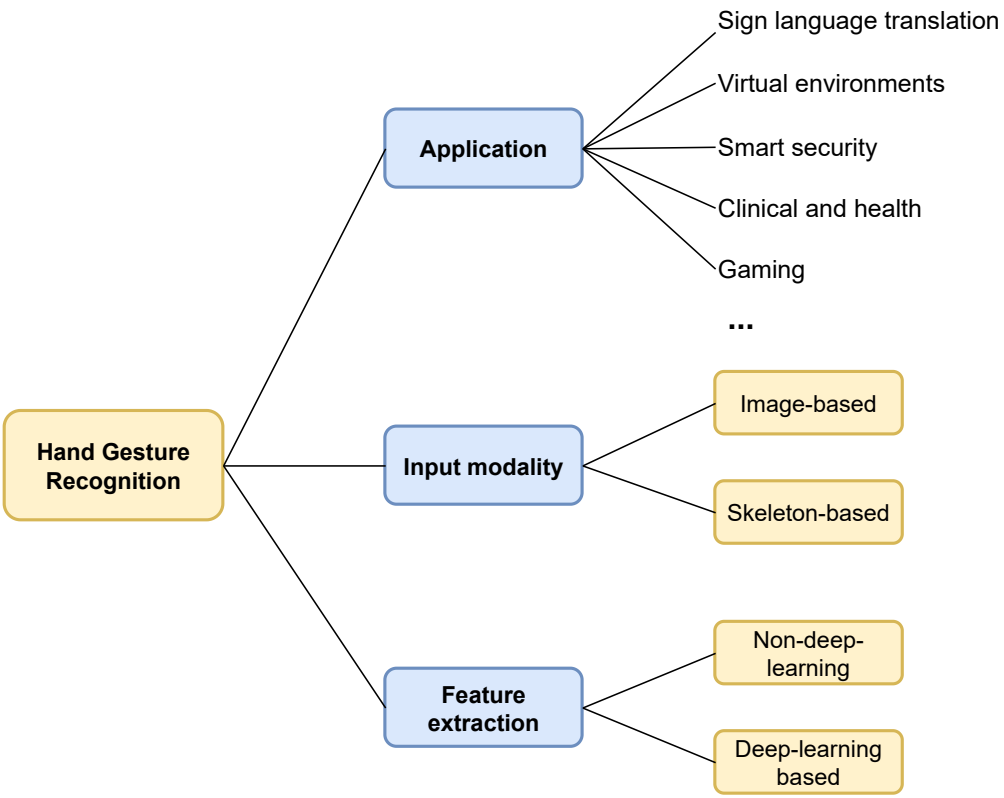


Figure 2.1: Summary on the applications and approaches to hand gesture recognition. There are several applications of hand gesture recognition. The approaches to hand gesture recognition can be classified based on the input modality or the feature extraction method.

2.1.1 Application of Hand Gesture Recognition

The application of hand gesture recognition systems has proven to be a valuable tool in a variety of fields, as evidenced by numerous studies [23, 24, 25]. Some of the domains in which hand gesture recognition has been utilized include sign language translation [26, 27, 28], clinical and health [25], virtual environment [29], robotic control [30], smart security [31, 32] and gaming [33, 34]. In the field of sign language translation, for example, hand gesture recognition can greatly facilitate the communication between ordinary people and individuals who are deaf or hard of hearing. Meanwhile, virtual environments can be made more immersive and interactive by incorporating hand

gesture recognition into their design. Hand gesture recognition can also improve the efficiency and effectiveness of smart security systems, enabling them to better detect and respond to potential security threats. In the realm of entertainment, hand gesture recognition can be applied to gaming to create an interactive environment and new interesting gaming experiences.

2.1.2 Approaches to Hand Gesture Recognition

Over the years, there have been several approaches proposed on hand gesture recognition. Hand gesture recognition methods can be classified into different categories based on the input modality and the method used for feature extraction.

Hand Gesture Recognition on Different Input Modalities

Based on the input modality, hand gesture recognition can be classified into image-based approach and skeleton-based approach. Image-based recognition takes RGB or RGB-D images as inputs and relies on image-level features for recognition. Although this is by far the most common approach used by many hand gesture recognition systems [23], the main challenge of image-based approach is to build models that are robust to the changes in background and lighting conditions. With the appearance of large open source datasets for hand gesture recognition in recent years [35, 36], training and benchmarking hand gesture recognition models have become a lot easier.

Skeleton-based approaches [15, 19], on the other hand, makes predictions by receiving hand skeletal data, such as 2D or 3D coordinates of hand joints as inputs. Compared with image-based methods, hand skeleton-based methods relieve the difficulty in recognizing a cluttered background, and requires lower computation cost, thus enabling real-time hand gesture recognition to be installed on small devices. Moreover, thanks to the development of smart, low-cost depth cameras and sensors, such as Kinect, Intel RealSense or Leap Motion, hand skeletal data can be easily obtained in real-time.

Hand Gesture Recognition on Different Feature Extraction Methods

Based on the approaches used for feature extraction, hand gesture recognition can be classified to non-deep-learning methods and deep-learning based methods. In non-deep-learning-methods, feature descriptors are often built to extract hand-crafted features from the image or the skeletal data. In [37], sequences of skeletal joints are represented as trajectories in an n-dimensional space. These trajectories are interpreted in a Riemannian manifold. The k-Nearest Neighbor (k-NN) is then used to classify an input sequence by computing the similarities between the shape of that sequence's trajectory with other trajectories in a Riemannian manifold. In [38], the authors used three different descriptors to describe the hand movement, hand location and the shape of the hand along with dynamic gestures. Each feature is then encoded in a final Fisher Vector representation which will then be fed into a Support Vector Machine (SVM) for gesture classification. Regardless of which feature descriptor is used, the extracted features are always fed to a classifier to perform the classification. Some frequently used classifiers [39] are Support Vector Machine (SVM), Dynamic Time Warping (DTW) algorithm, Hidden Markov models (HMM) or k-Nearest Neighbors (k-NN).

Deep-learning based-methods, on the other hand, adopt deep learning classification models to both learn the features to use for the classification and also the mapping from the feature space to the output classes. Therefore, they reveal themselves to be a very convenient class of models for classification. In [18], a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN), in particular, a Long Short-Term Memory (LSTM) are used for dynamic hand gesture recognition on both depth and skeletal data. While the CNN is used to learn the spatial features from depth images, the RNN is used to learn the temporal features from sequences of hand joints and feature maps. The use of RNN for hand gesture recognition can also be found in [40, 19]. In [41], a new skeleton-based hand gesture recognition approach based on a parallel CNN was introduced where sequences of hand-skeletal joints are processed by parallel convolutions, and the hand gestures are then classified. Another work is presented in [42] where they proposed a Dynamic Graph-Based Spatial Temporal

Attention (DG-STA) model for hand gesture recognition. The key idea is to perform a self-attention mechanism in both spatial and temporal domains to modify a unified graph dynamically in order to model different actions.

2.1.3 Comments on Related Approaches and Our Approach

Regarding the input modalities, both image-based and skeleton-based hand gesture recognition methods are widely applied nowadays. While image-based is the most common method with several open source datasets and benchmarks for evaluation, skeleton-based is getting more popular due to many advantages such as the robustness to varying background conditions, the lower computational cost, and the development of many devices for skeletal data extraction. In this work, we choose skeleton-based approach and use a Leap Motion Controller as the input device of our system. Thanks to its robustness in hand tracking, the Leap Motion Controller can capture hand movement and extract accurate hand skeletal data in real-time performance.

Regarding the approaches for hand gesture recognition, both non-deep-learning methods and deep-learning methods have their own advantages and disadvantages. Non-deep-learning methods with their simple design can be highly adaptable to real-time systems. However, non-deep-learning methods often require a large effort spent in feature engineering to manually select the features that result in the highest classification performance. Deep-learning methods, on the other hand, are more powerful methods which can learn both the feature extraction process and the mapping from the feature space to output classes. However, several works applying deep learning [40, 19] only focus on maximizing the performance of their approaches on popular benchmarks without considering the efficiency of their systems. Therefore, these approaches are often infeasible to run in real-time environment, especially on systems with limited computing power. Therefore, the trade-off between accuracy and time performance needs to considered when applying a hand gesture recognition to real-time systems.

In this work, we choose a non-deep-learning approach thanks to its simplicity and its feasibility in real-time systems. Our approach adopts a Support Vector Machine (SVM) classifier to identify hand poses and then recognizes gestures based on the sequence of identified key poses. Thanks to this simple design, our system can easily achieve real-time performance while still having high accuracy on our self-generated datasets.

2.2 Hand Gesture Recognition for Hand Rehabilitation

In this section, we review some existing works on hand gesture recognition for hand rehabilitation. Most of the works discussed in this section are applied in game-based rehabilitation in which the user indirectly performs hand exercises via playing interactive video games. We also have some comments at the end of this section and a comparison between existing works and our work.

2.2.1 Review on Related Works on Hand Gesture Recognition for Hand Rehabilitation

In [16], the authors presented their work on developing a game-based hand gesture recognition system for wrist rehabilitation. The system, named HandReha, is built to recognize 5 different hand poses: “Fist”, “Ok”, “Open Palm”, “PalmRight” and “PalmLeft” performed by the user in front of a computer webcam. Each captured video frame serves as input to a two-stage model, where the hand region is first segmented using background subtraction, and the binary mask of the hand is fed into a Convolutional Neural Network (CNN) for pose classification. The recognized pose is then used to control an avatar in a three dimensional maze run game. They also conducted a user study on 12 healthy participants and received positive responses from most of the users. Although they adopted an engaging game-based approach and achieved a high accuracy rate of 98.8% on a self-generated dataset, the set of static hand gestures (e.g. hand poses) that they used are loosely linked to common hand therapy exercises and hence have limited effects on hand rehabilitation. Also,

their approach recognizes static hand poses instead of dynamic hand gestures, which is quite inappropriate to be applied in hand rehabilitation since the main point in hand therapy is practicing hand movements via hand exercises. If a hand recognition system for rehabilitation only detects poses while missing the information regarding the hand movement, that system cannot check if a patient has completed a required hand movement or not. As a result, such a system has limited effects on the process of hand recovery. Moreover, the setting using web camera can introduce some difficulties as patients with hand problems cannot lift their hands up and keep their hands in the webcam view all the time. Therefore, the impact on the hand function recovery process might be reduced significantly.

In [17], the authors proposed a hand and fingertip detection system for game-based rehabilitation. In their approach, the hand is first detected using YOLOv3 object detection algorithm [43]. The cropped hand image is then passed to a trained Convolutional Neural Network to generate a heatmap containing the keypoints of the wrist and the 5 fingertips. Hand poses are then identified based on the keypoints' relative positions. The detected hand poses are used to control a game that they developed, called SuperFox. Their system recognizes a set of hand gestures selected from three common hand therapy exercises: "wrist ulnar/deviation", "wrist supination/pronation" and "hand/finger tendon glide" [44]. In their conducted user study, 10 healthy participants were invited to evaluate the effectiveness of the system. The authors received positive feedback thanks to their interesting approach that created motivation and enjoyable time for the participants. However, the proposed system still faces with some performance issues when the background is found to have a similar color to the skin color of the hand. Moreover, there are two limitations similar to the work in [16]. The first limitation is the recognition of hand poses instead of dynamic hand gestures in their approach. The second limitation is the setting in which the user needs to keep the hand in front of the webcam all the time, which might reduce the impact on the recovery process.

In [21], a hand gesture recognition system and an interactive hand writing game

were proposed to aid in the rehabilitation exercises of patients by improving their hand-eye coordination in a fun and engaging manner. The system obtains visual images from the computer camera and communicates with users through an interface. A hand joints detection module is adopted to extract hand skeleton from captured images. 10 static gestures from "0" to "9" are recognized by a lightweight residual graph convolutional neural network. Then, the trajectory processing module controls the generation of handwriting according to the hand behavior of the player and passes the image with the completed character to the character recognition module. The character recognition module then recognizes the images with characters and delivers them back to the interactive interface. The proposed system was tested with 4 healthy participants and the passing rate of each participants were recorded for evaluation. Although achieving high validation accuracy of 99.03%, there are still some limitations regarding the applicability of this system in hand rehabilitation, such as the inappropriate use of hand gestures, the setting using webcam to capture data and the lack of a user study to evaluate the effectiveness of the system.

Another study to be mentioned is [22]. In this work, they introduced a human computer interaction system by using hand gesture recognition to apply into game-based rehabilitation. Their system is applied in real-time application to contribute as a function of a virtual mouse. Several approaches were proposed, including hand region segmentation, hand palm partition and finger counting. Computer mouse is configured to move according to the location of hand palm region, and perform function of ordinary mouse according to the number of fingers detected. Although the effectiveness and and robustness in performing the mouse function were demonstrated, the system is far from being applied to hand rehabilitation due to the irrelevant choice of gestures, the robustness issues of the adopted color-based approach and the limitation in their user study.

Besides from image-based approaches that rely on computer webcam, skeleton-based hand gesture recognition is becoming more popular and has been applied in hand rehabilitation. Among several depth sensors and hand tracking devices, Leap Motion

Controller [45] stands out as one of the most robust hand tracking modules and is applied widely in several systems. Not only it is small, highly portable and easy to use, the Leap Motion controller is extremely suitable for hand rehabilitation. As the device can be placed below the hand to capture hand movement, users with hand problems can easily perform hand gestures while resting their forearms on a table or an arm support, without the need to lift their hands up as in the case of using webcam.

Li et al. [15] introduced a hand gesture recognition system for pose-stroke rehabilitation using a Leap Motion Controller. The system was developed to monitor hand exercises performed by patients at home. The skeletal data captured by Leap Motion is fed to a feature extractor module to extract distance-based and angle-based features. The feature vector then serves as the input to a Support Vector Machine to recognize hand gestures. The set of gestures were chosen from a pool of common hand exercises. The system achieves a high accuracy of 97.29% on a self-generated dataset. One drawback of this work is that their gesture recognition approach only identifies hand poses instead of taking the hand movement into consideration. Therefore this system might have wrong recognition in some real-life scenarios. Also, they did not conduct any user study to evaluate the effectiveness of their system on hand rehabilitation.

Another work using Leap Motion Controller was proposed in [14]. This work proposed an approach to monitor exercises for hand rehabilitation for post stroke patients. Angles of the joints are extracted as features, and the feature vector is fed into a Support Vector Machine for gesture recognition. From the recognition result, the system calculates the difference between the recorded gesture and the trained model to provide motion suggestions and feedback that help patients perform the hand exercises correctly. Similar to the work in [15], the approach proposed in this work does not take the dynamics of hand gestures into account, and no user study was conducted to evaluate the effectiveness of the system.

2.2.2 Comments on Related Works and Our System

From the review on existing works on hand gesture recognition for hand rehabilitation, advantages and limitations of each work were clearly discussed. Although most of the work achieved high accuracy and showed an interest in applying hand gesture recognition for rehabilitation, there are some limitations that need to be addressed, when it comes to the efficiency and the effects made on hand rehabilitation. For instance, in [21, 22], the set of gestures are quite limited and are not based on common hand rehabilitation exercise. Therefore, those systems have only limited effects on the process of hand recovery. In most of the mentioned work, the adopted approach only detects hand poses in order to recognize hand gestures. This can be a big drawback since only recognizing hand poses omits the information regarding hand movement, which is an important aspect in hand rehabilitation. Also the setting of using webcam, as in [16, 17, 21, 22], can cause some difficulties to users who have hand problems since this setting requires users to keep their hand in front of the webcam all the time. Finally, user study is another important factor to evaluate the effectiveness of the system, which was omitted in some studies, such as [14, 15, 21].

We have summarized the advantages and limitations of each study and listed in Table 2.1. Our method is also compared to these studies in this table. As can be seen, our approach has some advantages over previous work, in terms of applicability in hand rehabilitation, such as adopting a set of gesture from common hand exercises, using Leap Motion as an input device instead of webcam, recognizing hand gestures instead of hand poses, conducting a user study to evaluate the effectiveness of our system. Also, we carry out both the tasks of hand gesture recognition and game development in this work, which makes our system a complete solution to hand rehabilitation.

Work	**Input Device**	**Input Modality**	**Hand Exercises**	**Recognize Pose/Gesture**	**User Study**	**Game Dev**
Tan (2013)	Webcam	Image	No	Pose	3 healthy individuals	No
Potigutsai (2021)	Webcam	Image	Yes	Pose	10 healthy individuals	Yes
Farahanipad (2020)	Webcam	Image	No	Pose	10 healthy individuals	Yes
Chen (2022)	Webcam	Skeleton	No	Gesture	—	Yes
Cohen (2018)	Leap Motion	Skeleton	No	Pose	—	No*
Li (2017)	Leap Motion	Skeleton	Yes	Pose	—	No*
Ours	Leap Motion	Skeleton	Yes	Gesture	10 individuals	Yes

Table 2.1: Comparison between previous studies on HGR for hand rehabilitation and our approach. *: This work applies hand gesture recognition to monitor hand exercises instead of game-based rehabilitation.

Chapter 3

Theoretical Background

This chapter provides essential background knowledge pertaining to our project. It delves into various aspects, including hand rehabilitation in Section 3.1, hand gesture recognition in Section 3.2, mathematical principles in Section 3.3, machine learning algorithms in Section 3.4 and evaluation metrics in Section 3.5, all of which are directly applicable to the successful implementation of our project.

3.1 Hand Rehabilitation

In this section, we provide an overview of hand rehabilitation, covering fundamental concepts, detailed step-by-step processes, and the current global landscape.

3.1.1 Definition of Hand Rehabilitation

Hand rehabilitation, also known as hand therapy, is a specialized field within the broader domain of physical therapy that focuses on restoring optimal function and enhancing the quality of life for individuals with hand-related injuries, disorders, or disabilities [46, 47]. It encompasses a comprehensive range of therapeutic interventions designed to improve the strength, dexterity, coordination, and overall functionality of the hand.

Hand rehabilitation is rooted in a multidisciplinary approach, drawing from various

disciplines such as physical therapy, occupational therapy, orthopedics, neurology, and psychology. It integrates knowledge from these domains to develop tailored treatment plans that address the specific needs and goals of each individual.

The primary objective of hand rehabilitation is to promote the recovery of hand function, enabling individuals to perform daily activities and engage in meaningful occupations. This may include tasks such as writing, grasping objects, manipulating tools, performing self-care activities, and participating in recreational or vocational pursuits.

Hand rehabilitation interventions typically involve a combination of therapeutic exercises, manual techniques, adaptive devices, and assistive technologies. These interventions aim to improve muscle strength, joint range of motion, sensory perception, and proprioception.

3.1.2 Steps in Hand Rehabilitation

The process of hand rehabilitation involves a series of sequential steps designed to guide the assessment, treatment, and progress monitoring of individuals with hand-related injuries, disorders, or disabilities. These steps provide a structured framework for rehabilitation professionals to deliver effective and individualized care.

The following is an overview of the typical steps involved in hand rehabilitation:

- Evaluation and Assessment.
  The first step in hand rehabilitation is a comprehensive evaluation and assessment of the individual's condition. This includes gathering information about the individual's medical history, current symptoms, functional limitations, and personal goals. Physical examination techniques, such as range of motion measurements, strength testing, and sensory assessments, are conducted to identify specific impairments and determine the baseline status of hand function.
- Goal Setting.

Based on the evaluation findings and in collaboration with the individual, specific goals are established for the rehabilitation process. These goals should be realistic, measurable, and tailored to the individual's needs and aspirations. Examples of goals may include improving grip strength, increasing range of motion, enhancing fine motor skills, or regaining independence in performing daily activities.

- Treatment Planning.
  Once the goals are set, a customized treatment plan is developed. This plan outlines the specific interventions and strategies to be employed during the rehabilitation process. It takes into account the individual's impairments, functional limitations, preferences, and available resources. The treatment plan may include a combination of therapeutic exercises, manual techniques, splinting, modalities (such as heat or cold therapy), and assistive devices.

- Intervention.
  The intervention phase involves the implementation of the treatment plan. Rehabilitation professionals, such as physical therapists or occupational therapists, work closely with the individual to provide hands-on interventions and guidance. Therapeutic exercises are prescribed to improve strength, range of motion, coordination, and sensory integration. Manual techniques, such as joint mobilization or soft tissue mobilization, may be utilized to address specific impairments. Adaptations and modifications are made as needed to accommodate the individual's progress and changing needs.

- Education and Home Program.
  Throughout the rehabilitation process, patient education plays a crucial role. Individuals are educated about their condition, the rationale behind specific interventions, and strategies for self-management. They are provided with guidance on ergonomics, energy conservation, pain management, and proper body mechanics. Additionally, individuals are often given a home exercise program that includes exercises and activities to be performed independently to reinforce progress achieved during therapy sessions.

- Progress Monitoring and Reassessment.
  Regular reassessment is essential to monitor the individual's progress and make necessary adjustments to the treatment plan. Objective measurements, functional tests, and feedback from the individual are used to evaluate the effectiveness of the interventions. The frequency of reassessment depends on the nature of the condition and the goals set. Progress monitoring helps ensure that the rehabilitation process remains dynamic and responsive to the individual's changing needs.

- Transition and Maintenance.
  As the individual's hand function improves, the focus of rehabilitation shifts towards transitioning to functional activities and maintaining the achieved gains. This phase involves simulating real-life tasks, such as work or leisure activities, to facilitate the transfer of skills learned in therapy to everyday situations. The individual is encouraged to apply the strategies and techniques learned during rehabilitation to promote independence and long-term success.

In conclusion, the steps in hand rehabilitation involve a systematic approach to evaluate, treat, and monitor individuals with hand-related injuries, disorders, or disabilities. Through a combination of assessment, goal setting, treatment planning, intervention, education, progress monitoring, and transition to functional activities, hand rehabilitation aims to restore optimal hand function, promote independence, and improve overall quality of life.

3.1.3 Global Overview of Hand Rehabilitation

In the field of hand rehabilitation, a comprehensive understanding of the global landscape is essential for healthcare professionals and researchers alike. Hand rehabilitation encompasses a range of therapeutic interventions designed to restore or improve the functional abilities of individuals with hand impairments, injuries, or conditions.

First and foremost, hand rehabilitation is a multidisciplinary endeavor that draws

upon the expertise of various healthcare professionals, including occupational therapists, physiotherapists, hand surgeons, and rehabilitation specialists. These professionals work collaboratively to assess and treat patients with diverse hand-related challenges, such as those resulting from traumatic injuries, neurological disorders, or musculoskeletal conditions.

One significant trend in hand rehabilitation is the increasing emphasis on evidence-based practice [48, 49]. Researchers and clinicians are continuously striving to integrate the latest scientific knowledge and research findings into their treatment approaches. Evidence-based practice ensures that interventions are rooted in sound scientific principles and have demonstrated efficacy in improving patient outcomes. This approach also facilitates the standardization of treatment protocols across different healthcare settings, enabling better comparability and evaluation of intervention effectiveness.

Technological advancements have also greatly influenced the field of hand rehabilitation on a global scale. Innovations such as virtual reality, robotics, and wearable devices have opened up new possibilities for delivering engaging and targeted interventions [9, 50, 51, 52]. These technologies provide a means to simulate real-life scenarios, facilitate repetitive practice, and offer personalized feedback, all of which are crucial for maximizing therapeutic gains in hand rehabilitation. Furthermore, telehealth and telemedicine have gained prominence, allowing remote assessment, monitoring, and even treatment delivery, thereby expanding access to hand rehabilitation services, particularly in underserved areas [53].

Another important aspect of the global landscape of hand rehabilitation is the recognition of the importance of patient-centered care [54, 55]. The field has evolved from a purely biomedical approach to a more holistic model that considers individual preferences, goals, and psychosocial factors. Patient-centered care involves active collaboration between the patient and the healthcare team, empowering individuals to actively participate in their own recovery journey. This approach recognizes that each

person's experience and needs are unique, leading to more personalized and effective interventions.

Collaboration and knowledge exchange are crucial in advancing hand rehabilitation globally. Researchers, clinicians, and professional organizations actively engage in conferences, workshops, and research networks to share best practices, discuss emerging trends, and foster interdisciplinary collaboration. These collaborations help drive innovation, enhance treatment outcomes, and address the evolving challenges in hand rehabilitation.

To summarize, a global overview of hand rehabilitation reveals a dynamic and evolving field that integrates evidence-based practice, technological advancements, patient-centered care, and interdisciplinary collaboration. By staying informed about these global trends, healthcare professionals and researchers can contribute to the ongoing development and improvement of hand rehabilitation interventions, ultimately leading to enhanced functional outcomes and improved quality of life for individuals with hand impairments.

3.2 Hand Gesture Recognition

Hand gesture recognition is a prominent field in computer vision and human-computer interaction, aiming to enable natural and intuitive interaction between humans and machines. In this section, we aim to provide a concise overview of key concepts, applications, and challenges associated with hand gesture recognition.

3.2.1 Hand Poses

One crucial aspect of hand gesture recognition is the analysis and recognition of hand poses, which refers to the configurations and positions of the hand and fingers.

Hand poses pertain to the particular postures or arrangements of the hand and fingers. They encompass the static positioning and alignment of the hand, emphasizing the shape and orientation of individual digits, the thumb, and the palm. Hand poses find significant application in sign language, where distinct hand configurations symbolize specific words, letters, or concepts [56]. Moreover, disciplines such as anatomy and medical diagnostics utilize hand poses to examine patients, aiding in symptom identification and abnormality assessment [57]. By providing precise visual cues, hand poses convey explicit meanings and convey valuable information.

Hand poses play a fundamental role in understanding and interpreting human gestures. The human hand possesses remarkable dexterity, capable of expressing a wide range of meanings through intricate movements and configurations. Recognizing and comprehending these hand poses is essential for various applications, including sign language recognition, virtual reality, robotics, and human-computer interaction systems [58].

Accurate hand pose recognition involves multiple challenges due to the complexity and variability of hand configurations. The human hand consists of several interconnected bones, joints, and muscles, resulting in a vast number of possible hand poses. Additionally, factors such as occlusion, varying lighting conditions, and inter-class similarities further contribute to the complexity of the problem [59].

Hand poses and hand gestures are distinct elements of nonverbal communication. Hand pose refers to the static configuration of the hand and fingers, while hand gestures, mentioned in Subsection 3.2.2, encompass a wider range of dynamic movements. Hand pose is precise and often associated with sign language or specialized contexts, while hand gestures are versatile and culturally influenced. Both contribute to effective communication and enhance interpersonal interactions.

3.2.2 Hand Gestures

In the realm of human-computer interaction and computer vision, hand gestures play a pivotal role as a means of non-verbal communication between humans and machines. Hand gesture recognition involves the analysis and interpretation of various hand movements and postures to infer the intended meaning or command conveyed by the user.

Hand gestures refer to a wide range of dynamic movements and actions performed by the hand and fingers. They are a form of nonverbal communication used to convey meanings, emotions, or intentions [60]. Hand gestures can include movements such as pointing, waving, thumbs-up, or handshakes. They are culturally influenced and can vary in their interpretations across different societies. Hand gestures complement spoken language and enhance the overall expressiveness and effectiveness of communication [61].

Hand gestures possess inherent expressiveness, enabling humans to convey a rich set of instructions, emotions, and intentions. Recognizing and understanding these gestures is of utmost importance for numerous applications, including virtual reality, augmented reality, gaming, sign language recognition, and human-robot interaction [62]. By enabling intuitive and natural interaction, hand gesture recognition systems aim to bridge the gap between humans and machines, enhancing user experience and enabling novel modes of interaction.

The recognition of hand gestures is a challenging task due to the diverse nature of gestures and the complexity of hand movements [63, 64]. Hand gestures can vary widely in terms of hand shape, finger positions, temporal dynamics, and spatial trajectories. Additionally, the presence of occlusions, varying lighting conditions, and inter-class similarities further compound the difficulty of accurate gesture recognition.

Despite the significant progress in hand gesture recognition, several research challenges persist. One key challenge is the development of real-time systems capable of

recognizing gestures in real-world environments with limited computational resources [65]. In addition, addressing the issues of gesture ambiguity, dynamic hand movements, and individual variations in hand shapes and sizes remains an ongoing focus of research.

3.3 General Mathematical Knowledge

Mathematics is a discipline that encompasses the study of numbers, quantities, shapes, and patterns. It serves as a fundamental tool in various scientific and practical endeavors, providing a framework for logical reasoning and problem-solving. In this section, we explore some key concepts and principles that serve as the foundation of necessary mathematical knowledge.

3.3.1 Vectors

In mathematics, *vectors* are regarded as mathematical entities characterized by a dual nature consisting of both magnitude (or length) and direction. Their utility lies in their capacity to effectively represent a variety of quantities, including velocity, force, and displacement [66].

Vectors can be represented using coordinates, where each coordinate corresponds to a component of the vector in a particular dimension. For example, a two-dimensional vector can be represented as (x, y), where x and y are the components of the vector in x and y directions, respectively.

To represent a vector, the prevailing typographic convention is to use lower case, upright boldface type, as exemplified by $\mathbf{v}$. The International Organization for Standardization (ISO) suggests either bold italic serif, denoted as $\boldsymbol{v}$, or non-bold italic serif with a right arrow accent, such as $\vec{v}$.

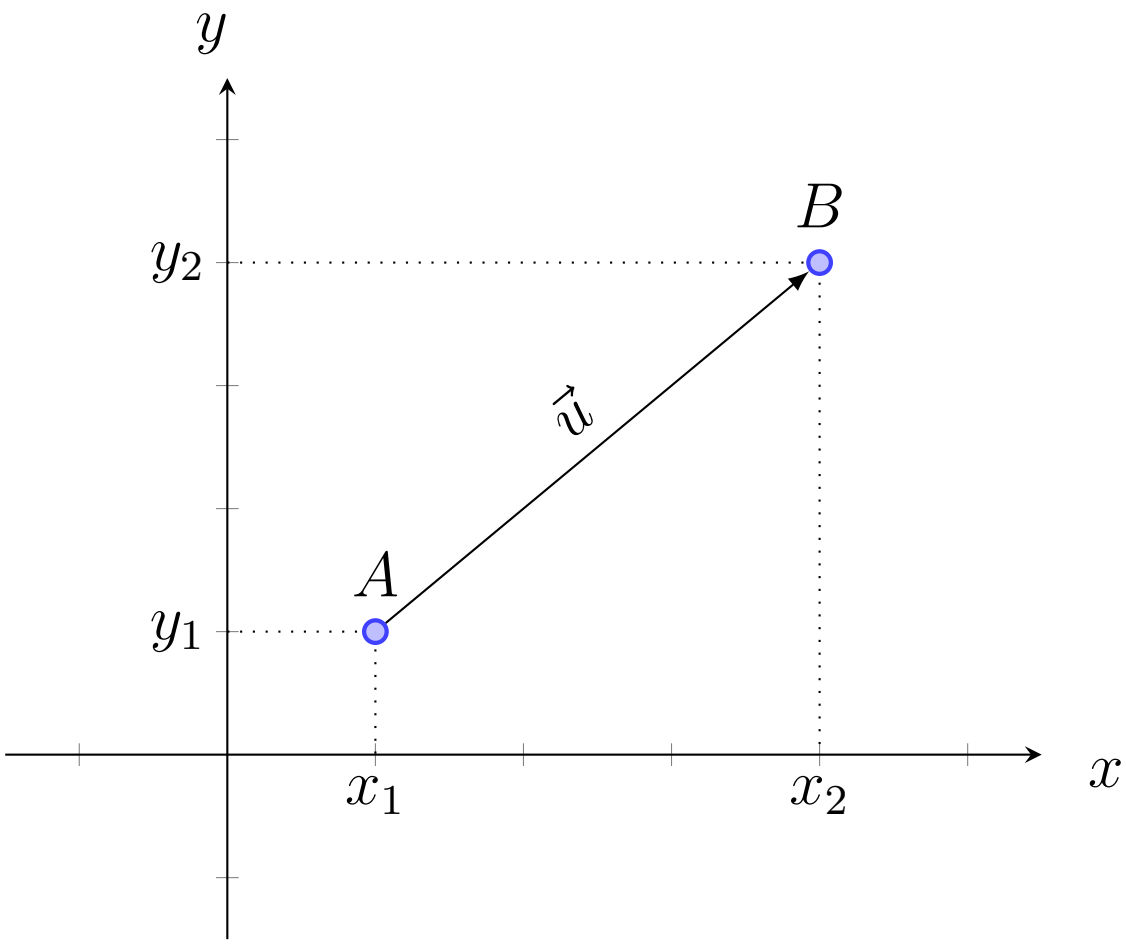


Figure 3.1: Example of a vector in 2-D space. The vector $\vec{u}$ is formed by connecting points A and B, with point A serving as the origin and defining the direction from A to B.

Coordinates

A *coordinate* is a value that specifies the position of a point in a particular space or plane [67]. Coordinates are often used in mathematics and geometry to describe the location of objects in a given system or reference frame.

In a two-dimensional coordinate system, a point is located by its position along two perpendicular axes, typically labeled the x-axis and the y-axis. The x-coordinate

gives the horizontal position of the point, while the y-coordinate gives the vertical position. The coordinates of a point are typically written as an ordered pair (x, y), where x is the horizontal coordinate and y is the vertical coordinate.

Figure 3.1 illustrates two points, A and B, in the 2-D Cartesian coordinate system. The coordinate pair $(1, 1)$ is assigned to point A, which is obtained by determining the intersection of two lines, namely, $y = 1$ and $x = 1$. The nomenclature of the coordinate pair $(1, 1)$ ascribed to point A conforms to the convention of Cartesian coordinates in analytic geometry. Similarly, point B has coordinates $(3, 3)$.

In a three-dimensional coordinate system, a point is located by its position along three perpendicular axes, typically labeled the x-axis, y-axis, and z-axis. The coordinates of a point in this system are typically written as an ordered triple (x, y, z), where x is the horizontal coordinate, y is the vertical coordinate, and z is the coordinate in the third dimension.

The following are some commonly used coordinate systems [68]:

- Cartesian coordinates.
  This is a fundamental coordinate system in analytic geometry, used to locate points in two or three dimensions using perpendicular axes. The axes are labeled x, y, and sometimes z, and the origin $(0, 0)$ is at the intersection of the axes. Points are located by measuring their distances from the origin along each axis. Figure 3.2a illustrates the three-dimensional Cartesian coordinate system.

- Polar coordinates.
  This coordinate system is used to locate points in two dimensions using a distance from a fixed point (the origin) and an angle measured from a fixed direction (usually the positive x-axis). Polar coordinates are particularly useful for describing circular or rotational motion. Figure 3.2b illustrates the three-dimensional Polar coordinate system.

- Spherical coordinates.

This coordinate system is used to locate points in three dimensions using a distance from a fixed point (the origin), an angle measured from a fixed direction (usually the positive z-axis), and an angle in a perpendicular plane (usually the xy-plane). Spherical coordinates are particularly useful for describing objects or systems with spherical symmetry. Figure 3.2c illustrates the three-dimensional Spherical coordinate system.

- Cylindrical coordinates.
  This system locates points in three dimensions using a distance from a fixed point (the origin), an angle measured from a fixed direction (usually the positive x-axis), and a height. Cylindrical coordinates are particularly useful for describing cylindrical objects or systems. Figure 3.2d illustrates the three-dimensional Cylindrical coordinate system.

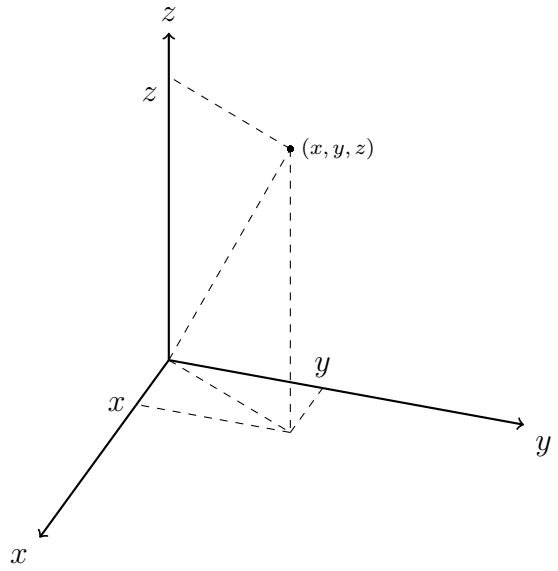
- Homogeneous coordinates.
  This system is used in projective geometry to represent points in two or three dimensions as vectors of four or five components. Homogeneous coordinates are particularly useful for describing transformations such as translations, rotations, and projections.

Coordinates are used to describe the location of objects in various fields of study, including mathematics, physics, and engineering. They can also be used to describe the position of a location on the Earth's surface using latitude and longitude coordinates.

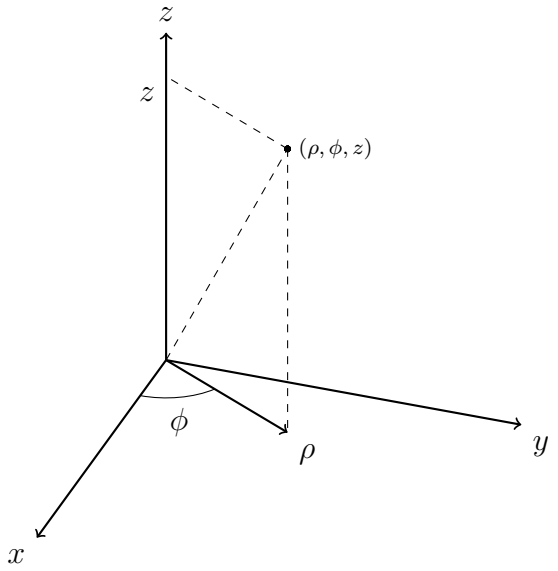
Basic Vector Properties

Basically, vectors, akin to other mathematical entities, manifest properties of equality and oppositeness:

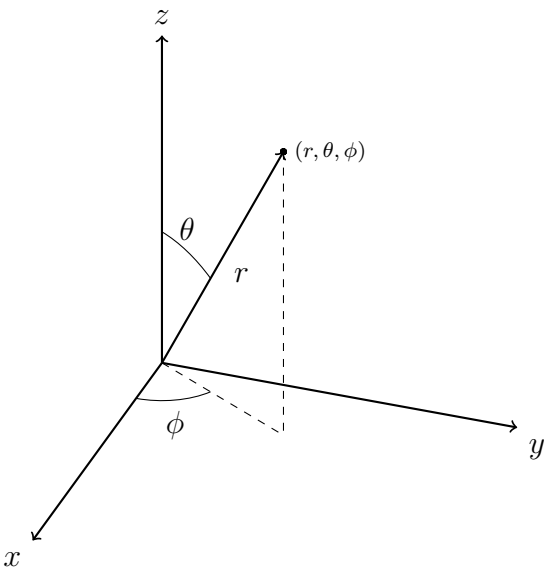
- Equality of Vectors.
  Two vectors are considered equal if their corresponding components are equal. For example, given two vectors $\mathbf{A} = \begin{bmatrix} A_x & A_y & A_z \end{bmatrix}^\top$ and $\mathbf{B} = \begin{bmatrix} B_x & B_y & B_z \end{bmatrix}^\top$, they are equal if $A_x = B_x$, $A_y = B_y$, and $A_z = B_z$.



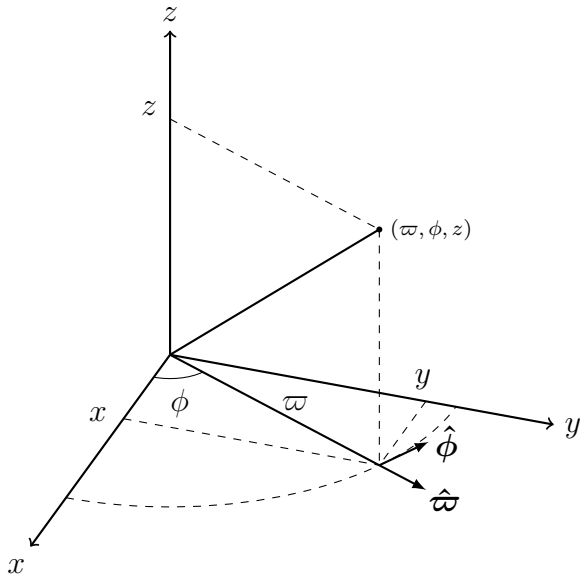
(a) Cartesian coordinate system.



(b) Polar coordinate system.



(c) Spherical coordinate system.



(d) Cylindrical coordinate system.

Figure 3.2: Mathematical illustration of some common coordinate systems. To facilitate observation, the viewing angle is configured in three dimensions.

- Opposite Vectors.

  Opposite vectors, also known as negation or additive inverse, are vectors that

have the same magnitude but opposite directions. For a vector $\mathbf{V} = \begin{bmatrix} V_x & V_y & V_z \end{bmatrix}^\top$, its opposite vector $-\mathbf{V}$ is obtained by changing the signs of its components:

$$-\mathbf{V} = \begin{bmatrix} -V_x & -V_y & -V_z \end{bmatrix}^\top$$

In addition, the realm of vectors extends to higher dimensions, encompassing various types such as unit vectors, zero vectors, and parallel vectors. These vectors possess several fundamental properties, which include:

- Vector Length (Magnitude).
  The length or magnitude of a vector $\mathbf{V} = \begin{bmatrix} V_x\ V_y\ V_z \end{bmatrix}^\top$ in three-dimensional space can be determined using the Euclidean norm. It is calculated as the square root of the sum of the squares of its components:

  $$|\mathbf{V}| = \sqrt{V_x^2 + V_y^2 + V_z^2}$$

  Example: Consider the vector $\mathbf{V} = \begin{bmatrix} 3 & -4 & 5 \end{bmatrix}^\top$. Its length is computed as:

  $$|\mathbf{V}| = \sqrt{3^2 + (-4)^2 + 5^2} = \sqrt{9 + 16 + 25} = \sqrt{50} \approx 7.07$$

- Unit Vector.
  A unit vector is a vector with a magnitude of 1. To obtain a unit vector in the same direction as a given vector $\mathbf{V}$, it can be normalized by dividing each component by its magnitude:

  $$\mathbf{V}_{\text{unit}} = \frac{\mathbf{V}}{|\mathbf{V}|}$$

  Example: Let $\mathbf{V} = \begin{bmatrix} 3 & -4 & 5 \end{bmatrix}^\top$. To find the unit vector $\mathbf{V}_{\text{unit}}$, divide $\mathbf{V}$ by its magnitude:

  $$\mathbf{V}_{\text{unit}} = \frac{\begin{bmatrix} 3 & -4 & 5 \end{bmatrix}^\top}{\sqrt{50}} \approx \begin{bmatrix} 0.424 & -0.566 & 0.707 \end{bmatrix}^\top$$

- Zero Vector.

  The zero vector, denoted as $\mathbf{0}$, has all its components equal to zero. It serves as the additive identity element, such that adding the zero vector to any vector $\mathbf{V}$ yields the same vector:

$$\mathbf{V} + \mathbf{0} = \mathbf{V}$$

- Parallel Vectors.

  Two vectors are parallel if they have the same or opposite directions. This means that the ratio between their corresponding components remains constant. For vectors $\mathbf{A} = \begin{bmatrix} A_x & A_y & A_z \end{bmatrix}^\top$ and $\mathbf{B} = \begin{bmatrix} B_x & B_y & B_z \end{bmatrix}^\top$ to be parallel, the ratios $\frac{A_x}{B_x} = \frac{A_y}{B_y} = \frac{A_z}{B_z}$ must hold, where the denominators are nonzero.

  Example: Let $\mathbf{A} = \begin{bmatrix} 3 & 6 & -2 \end{bmatrix}^\top$ and $\mathbf{B} = \begin{bmatrix} 2 & 4 & -1 \end{bmatrix}^\top$. These vectors are parallel since $\frac{3}{2} = \frac{6}{4} = \frac{-2}{-1} = \frac{3}{2}$.

- Antiparallel Vectors.

  Antiparallel vectors are vectors that have the same magnitude but opposite directions. In other words, they are parallel vectors with opposite senses. For vectors $\mathbf{A} = \begin{bmatrix} A_x & A_y & A_z \end{bmatrix}^\top$ and $\mathbf{B} = \begin{bmatrix} B_x & B_y & B_z \end{bmatrix}^\top$ to be antiparallel, the ratios $\frac{A_x}{B_x} = \frac{A_y}{B_y} = \frac{A_z}{B_z}$ must hold, where the denominators are nonzero, and the signs of the ratios must differ.

  Example: Consider $\mathbf{A} = \begin{bmatrix} 2 & -4 & 1 \end{bmatrix}^\top$ and $\mathbf{B} = \begin{bmatrix} -2 & 4 & -1 \end{bmatrix}^\top$. These vectors are antiparallel since $\frac{2}{-2} = \frac{-4}{4} = \frac{1}{-1} = -1$, and the signs of the ratios differ.

According to Figure 3.1, the process of creating a vector based on two points in a two-dimensional coordinate system, $A(1, 1)$ and $B(3, 3)$, can be outlined as follows:

- Identify the coordinates of the two points.

  Let the coordinates of the first point be denoted as $(x_1, y_1) = (1, 1)$. Let the coordinates of the second point be denoted as $(x_2, y_2) = (3, 3)$.

- Calculate the differences in coordinates.

Subtract the x-coordinates and y-coordinates of the two points to find the differences:

$$\Delta x = x_2 - x_1 = 3 - 1 = 2$$
$$\Delta y = y_2 - y_1 = 3 - 1 = 2$$

- Formulate the vector.
  A vector can be represented by an ordered pair or column vector. In this case, the vector can be expressed as:

$$\mathbf{V} = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix}^\top = \begin{bmatrix} 2 & 2 \end{bmatrix}^\top = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

- Understand the meaning of the vector.
  The vector $\mathbf{V} = \begin{bmatrix} 2 & 2 \end{bmatrix}^\top$ represents the change in position from the initial point $(1, 1)$ to the final point $(3, 3)$. It has a magnitude of $|\mathbf{V}| = \sqrt{\Delta x^2 + \Delta y^2} = \sqrt{2^2 + 2^2} = \sqrt{8} \approx 2.83$ units and is directed along the line connecting the two points.

- Optional - Normalize the vector.
  To obtain a unit vector (a vector with magnitude equal to 1) in the same direction, the vector can be normalized by dividing each component by its magnitude:

$$\mathbf{V}_{\text{normalized}} = \frac{\mathbf{V}}{|\mathbf{V}|} = \frac{\begin{bmatrix} 2 & 2 \end{bmatrix}^\top}{2.83} \approx \begin{bmatrix} 0.71 & 0.71 \end{bmatrix}^\top$$

Vector addition and subtraction serve as fundamental operations for manipulating vectors. In vector addition, corresponding components of two vectors are added together. Conversely, vector subtraction involves subtracting the corresponding components. Scalar multiplication is another operation defined for vectors, where each component is multiplied by a scalar, which is a real number. These operations adhere to essential properties, including the distributive law and the associative law,

ensuring mathematical consistency and coherence. To illustrate each case, consider the following examples:

- Vector Addition.
  Given two vectors $\mathbf{A} = \begin{bmatrix} A_x \\ A_y \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} B_x \\ B_y \end{bmatrix}$, their sum $\mathbf{C} = \mathbf{A} + \mathbf{B}$ can be determined by adding their corresponding components:

  $$\mathbf{C} = \begin{bmatrix} A_x + B_x \\ A_y + B_y \end{bmatrix}$$

  Example: Let $\mathbf{A} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} -1 \\ 4 \end{bmatrix}$. The sum $\mathbf{C} = \mathbf{A} + \mathbf{B}$ is computed as:

  $$\mathbf{C} = \begin{bmatrix} 2 & + & (-1) \\ 3 & + & 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 7 \end{bmatrix}$$

- Vector Subtraction.
  For vector subtraction, the corresponding components of the two vectors are subtracted from each other. Let $\mathbf{D} = \begin{bmatrix} D_x \\ D_y \end{bmatrix}$ be another vector. The difference $\mathbf{E} = \mathbf{D} - \mathbf{A}$ is calculated as:

  $$\mathbf{E} = \begin{bmatrix} D_x - A_x \\ D_y - A_y \end{bmatrix}$$

  Example: Consider $\mathbf{D} = \begin{bmatrix} 4 & 1 \end{bmatrix}$. The difference $\mathbf{E} = \mathbf{D} - \mathbf{A}$ is evaluated as:

  $$\mathbf{E} = \begin{bmatrix} 4 - 2 \\ 1 - 3 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

- Scalar Multiplication.
  Scalar multiplication involves multiplying each component of a vector by a scalar value, denoted by λ. If $\mathbf{F} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}$ is a vector, the scalar multiplication

$\mathbf{G} = \lambda\mathbf{F}$ is given by:

$$\mathbf{G} = \begin{bmatrix} \lambda F_x \\ \lambda F_y \end{bmatrix}$$

Example: Let $\mathbf{F} = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$ and consider $\lambda = 3$. The scalar multiplication $\mathbf{G} = \lambda\mathbf{F}$ is computed as:

$$\mathbf{G} = \begin{bmatrix} 3 & \cdot & 2 \\ 3 & \cdot & -3 \end{bmatrix} = \begin{bmatrix} 6 \\ -9 \end{bmatrix}$$

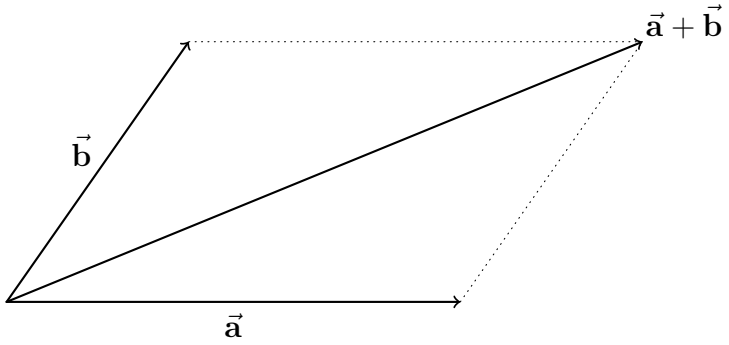


Figure 3.3: An illustration of adding two vectors in 2-D space. A new vector is created from two vectors $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$ based on the parallelogram rule.

Besides, vector products are fundamental mathematical operations that extend the capabilities of vectors beyond mere addition, subtraction, and scalar multiplication. These vector products offer profound insights into the geometric relationships and physical properties of vectors. The two primary vector products are the cross product and the dot product.

The dot product, also referred to as the scalar product, is an algebraic operation that combines two vectors and yields a scalar value. It quantifies the relationship between the magnitudes of the vectors and the angle between them. For two vectors

$\mathbf{A} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix}$, the dot product is defined in Equation 3.1:

$$\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}||\mathbf{B}| \cos(\theta), \tag{3.1}$$

where $|\mathbf{A}|$ and $|\mathbf{B}|$ represent the magnitudes of vectors $\mathbf{A}$ and $\mathbf{B}$, respectively, and θ denotes the angle between them.

To determine their dot product using the angle theta, knowledge of the magnitudes of the vectors and the angle between them is necessary. Assume $|\mathbf{P}| = 4$, $|\mathbf{Q}| = 3$, and the angle $\theta = 45$ degrees. Consequently, the dot product of two vectors $\mathbf{P}$ and $\mathbf{Q}$ can be calculated as follows:

$$\mathbf{P} \cdot \mathbf{Q} = 4 \cdot 3 \cdot \cos(45^\circ) = 6\sqrt{2}$$

The dot product can be computed by evaluating the sum of the products of corresponding components of the two vectors $\mathbf{A}$ and $\mathbf{B}$, illustrated in Equation 3.2:

$$\mathbf{A} \cdot \mathbf{B} = A_x B_x + A_y B_y + A_z B_z \tag{3.2}$$

For instance, consider the vectors $\mathbf{X} = \begin{bmatrix} 3 \\ -2 \\ 4 \end{bmatrix}$ and $\mathbf{Y} = \begin{bmatrix} 1 \\ 5 \\ -2 \end{bmatrix}$. To find their dot product by evaluating the sum of the products of corresponding components, the following can be calculated:

$$\mathbf{X} \cdot \mathbf{Y} = (3 \cdot 1) + (-2 \cdot 5) + (4 \cdot -2) = 3 - 10 - 8 = -15$$

The cross product, also known as the vector product, is an operation performed on two vectors in three-dimensional space. It results in a vector that is perpendicular to the plane formed by the original vectors. The magnitude of the cross product is equal to the product of the magnitudes of the vectors and the sine of the angle between

them. The direction of the cross product follows the right-hand rule. For two vectors $\mathbf{A} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix}$, the cross product is calculated as shown in Equation 3.3:

$$\mathbf{A} \times \mathbf{B} = |\mathbf{A}||\mathbf{B}| \sin(\theta)\mathbf{n}, \tag{3.3}$$

where $|\mathbf{A}|$ and $|\mathbf{B}|$ represent the magnitudes of vectors $\mathbf{A}$ and $\mathbf{B}$, respectively, θ is the angle between them, and $\mathbf{n}$ is a unit vector perpendicular to the plane formed by $\mathbf{A}$ and $\mathbf{B}$.

The cross product can also be written as Equation 3.4:

$$\mathbf{A} \times \mathbf{B} = (A_y B_z - A_z B_y)\mathbf{e}_1 + (A_z B_x - A_x B_z)\mathbf{e}_2 + (A_x B_y - A_y B_x)\mathbf{e}_3, \tag{3.4}$$

where $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ are the unit vectors in the x, y, and z directions, respectively.

Consider two vectors $\mathbf{M} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$ and $\mathbf{N} = \begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix}$. Hence, $|\mathbf{M}| = \sqrt{14}$, $|\mathbf{N}| = \sqrt{21}$, and the angle $\theta = 45$ degrees. The cross product can be calculated as follows, with $\mathbf{n}$ is the unit vector perpendicular to both $\mathbf{M}$ and $\mathbf{N}$:

$$\begin{aligned} \mathbf{M} \times \mathbf{N} &= (\sqrt{14} \cdot \sqrt{21}) \sin(45^\circ)\mathbf{n} \\ &= (\sqrt{14} \cdot \sqrt{21}) \frac{\sqrt{2}}{2}\mathbf{n} \end{aligned}$$

Another illustrative instance of computing the cross product between two vectors. Given two vectors $\mathbf{R} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$ and $\mathbf{S} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}$. The cross product of two vectors can be

calculated employing the following formula:

$$\begin{aligned}\mathbf{R} \times \mathbf{S} &= (R_y S_z - R_z S_y)i + (R_z S_x - R_x S_z)j + (R_x S_y - R_y S_x)k \\ &= ((3 \cdot 3) - (4 \cdot -2))i + ((4 \cdot 1) - (2 \cdot 3))j + ((2 \cdot -2) - (3 \cdot 1))k \\ &= 17i - 2j - 7k\end{aligned}$$

The computed cross product of vectors $\mathbf{R}$ and $\mathbf{S}$, denoted as $\mathbf{T} = \begin{bmatrix}17 & -2 & -7\end{bmatrix}^\top$, represents a vector in three-dimensional space. In this representation, the components in $\mathbf{T}$ correspond to the projections of the resultant vector onto the unit vectors i, j, and k, respectively. The unit vector i signifies the positive x-axis direction, j represents the positive y-axis direction, and k indicates the positive z-axis direction. Thus, the cross product $\mathbf{T}$ conveys not only the magnitude but also the direction of the resultant vector relative to the coordinate system, incorporating both geometric and quantitative information.

Summary and Application of Vectors

Vectors, a fundamental concept in mathematics, possess both magnitude and direction, making them a versatile tool in various domains. This section provides a concise summary of vectors and explores their applications across diverse fields.

A vector is a mathematical entity represented by an ordered set of components, typically in a multi-dimensional space. It possesses both a magnitude, which denotes its length or size, and a direction, which indicates its orientation in space. Vectors can be visualized as arrows, with their length representing magnitude and their direction depicting the orientation. They can be added, subtracted, scaled, and manipulated using vector operations, leading to numerous applications.

In physics, vectors find extensive use in describing quantities that possess both magnitude and direction. For instance, displacement, velocity, acceleration, and force can all be represented as vectors. By utilizing vector operations, such as addition and dot product, one can efficiently analyze and solve complex physical problems. Vectors

also play a crucial role in electromagnetic theory, where electric and magnetic fields are described using vector quantities [69].

In engineering and computer science, vectors serve as a powerful tool for representing and manipulating data. For example, in computer graphics, vectors are employed to describe the position, orientation, and size of objects in a virtual scene [70]. Vector graphics, characterized by smooth curves and scalability, rely on mathematical equations to define shapes and colors. In addition, algorithms based on vector operations are utilized in various computational tasks, such as optimization, machine learning, and computer vision [71, 72].

Vectors also have various applications within mathematics itself. They form the foundation of linear algebra, a branch of mathematics concerned with vector spaces and linear transformations. Concepts such as vector spaces, bases, and linear independence are essential in studying vectors and their properties. Furthermore, vectors are instrumental in calculus, particularly in the study of curves and surfaces, where tangent vectors and normal vectors are employed to determine rates of change and geometric properties.

In summary, vectors are an indispensable mathematical concept with a wide range of applications across various disciplines. Their ability to represent both magnitude and direction makes them particularly useful in physics, engineering, computer science, and mathematics. Whether in analyzing physical phenomena, manipulating data, or solving complex mathematical problems, vectors offer a versatile framework for understanding and modeling the world around us.

3.3.2 Angles

In the realm of mathematics, angles are fundamental geometric entities that play a crucial role in various branches of study, including geometry, trigonometry, and physics. An *angle* is defined as the measure of the rotation required to bring one line

segment, called a ray, into alignment with another [73].

Angles are typically denoted by a symbol or by using three points, with the middle point indicating the vertex of the angle. The unit used to measure angles is usually degrees (°) or radians (rad). A complete rotation around a point corresponds to an angle of 360 degrees or 2π radians.

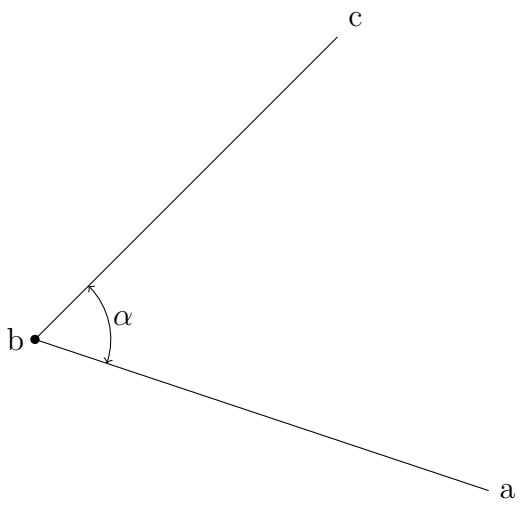


Figure 3.4: An illustration of an angle in 2-D space. α is the angle formed from two intersecting lines ab and bc at the intersection b.

Basic Angle Properties

There are several types of angles, each characterized by its measure and specific properties. The most common types include:

- Acute Angle.
  An acute angle is any angle whose measure is less than 90 degrees ($< 90°$) or $< \frac{\pi}{2}$ radians. Figure 3.4 depicts an acute angle α in two-dimensional space.
- Right Angle.
  A right angle is exactly 90 degrees (90°) or $\frac{\pi}{2}$ radians. It forms the basis for the

concept of perpendicularity, as two lines are considered perpendicular if they intersect at a right angle.

- Obtuse Angle.
  An obtuse angle is an angle greater than 90 degrees but less than 180 degrees ($90^\circ < \text{angle} < 180^\circ$) or $\frac{\pi}{2} < \text{angle} < \pi$ radians.

- Straight Angle.
  A straight angle measures exactly 180 degrees (180°) or π radians. It corresponds to a straight line, and its two rays are collinear.

- Reflex Angle.
  A reflex angle is an angle greater than 180 degrees but less than 360 degrees ($180^\circ < \text{angle} < 360^\circ$) or $\pi < \text{angle} < 2\pi$ radians. It extends beyond a full revolution.

- Complementary Angles.
  Two angles are said to be complementary if their measures add up to 90 degrees (90°) or $\frac{\pi}{2}$ radians.

- Supplementary Angles.
  Two angles are considered supplementary if their measures add up to 180 degrees (180°) or π radians.

Furthermore, angles can be classified based on their position relative to other angles or lines. These classifications include adjacent angles, vertical angles, alternate angles, corresponding angles, and interior and exterior angles. Some of these properties include:

- Vertical Angles.
  When two lines intersect, they form pairs of vertical angles that are congruent. Vertical angles are opposite each other and share the same vertex.

- Corresponding Angles.
  When a transversal intersects two parallel lines, corresponding angles are formed.

These angles are located in the same relative position at each intersection and are congruent.

- Alternate Interior and Exterior Angles.
  When a transversal intersects two parallel lines, alternate interior angles lie on opposite sides of the transversal and between the parallel lines. Similarly, alternate exterior angles lie on opposite sides of the transversal but outside the parallel lines. Alternate interior and exterior angles are congruent.

- Sum of Interior Angles in Polygons.
  In a polygon with n sides, the sum of its interior angles is given by $(n-2)\times 180°$. This relationship is useful in determining the number of sides in a polygon or finding the measure of a particular angle within the polygon.

- Angle Addition Postulate.
  The angle addition postulate states that the measure of the whole angle is equal to the sum of the measures of its parts. For example, if two angles α and β are adjacent, then the measure of the combined angle formed by them is equal to the sum of the measures of angle α and angle β.

- Congruent Angles.
  Congruent angles have the same measure. This property is vital in geometric proofs and establishing equality between angles.

- Angle Bisector.
  An angle bisector is a line, ray, or line segment that divides an angle into two congruent angles. It divides the angle into two equal parts.

- Complementary and Supplementary Angle Relationships.
  Complementary angles add up to 90 degrees (90°) or $\frac{\pi}{2}$ radians, while supplementary angles add up to 180 degrees (180°) or π radians. These relationships are essential in solving equations and finding missing angles.

Combined with vectors, the study of angles takes on additional depth and applicability in various mathematical and scientific disciplines. Vectors provide a powerful

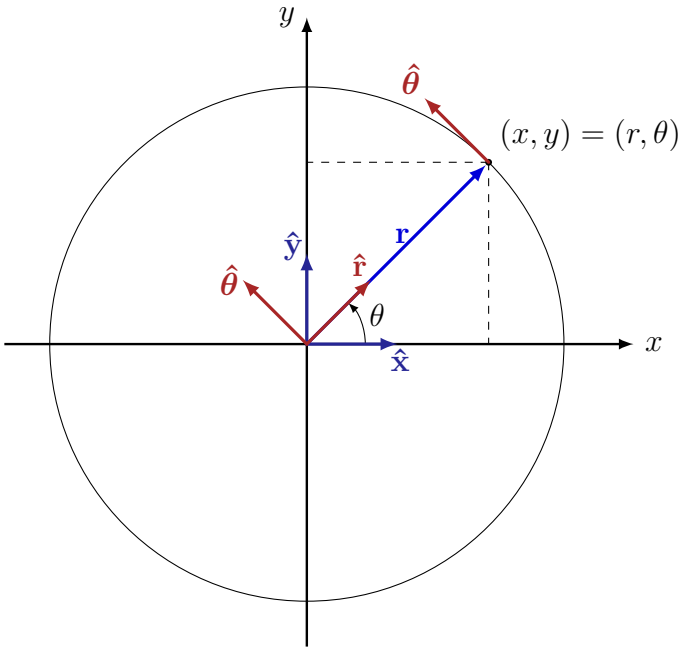


Figure 3.5: An illustration of an angle between 2 vectors in 2-D space. θ is the angle between the unit vector $\hat{\mathbf{x}}$ and the radius vector $\hat{\mathbf{r}}$ in the polar coordinate system.

mathematical framework for understanding physical quantities and transformations, and when applied to angles, they offer valuable tools for representing and analyzing rotational and directional aspects of geometric objects. By utilizing vector operations, such as addition, subtraction, and dot products, mathematicians can establish profound connections between angles and vectors. This integration enhances geometric reasoning, enabling a more comprehensive exploration of motion, forces, and transformations. Moreover, the application of vectors allows for a deeper understanding of the relationships between angles, providing a broader scope for solving complex problems in fields such as mechanics, engineering, and computer graphics. Figure 3.5 shows the close relationship between angle and vector.

Angles find wide-ranging applications in various fields, such as engineering, architecture, surveying, and physics. Trigonometric functions, such as sine, cosine, and tangent, are defined in terms of angles and are crucial in solving problems involving

triangles and periodic phenomena[1].

Understanding these properties and relationships of angles allows mathematicians and scientists to solve complex geometric problems, analyze shapes and figures, and make precise calculations in various applications.

In conclusion, angles form a cornerstone of mathematical knowledge, permeating numerous disciplines and applications. Their classification, properties, and relationships provide a robust framework for understanding geometric concepts and solving real-world problems. Mastery of angles and their associated principles equips individuals with essential tools for exploring the intricacies of mathematics and its practical applications.

3.3.3 Distance Metrics

Within mathematics, distance metrics hold significant importance in various branches like geometry, analysis, and data science. A *distance metric*, also known as a metric or a distance function, is a mathematical construct that quantifies the dissimilarity or similarity between two objects [74]. By assigning a numerical value to the concept of "distance", metrics enable the measurement and comparison of objects within a given space.

Formally, a distance metric on a set X is a function $d : X \times X \to \mathbb{R}$ that assigns a real number to any pair of elements in the set. This function d must satisfy the following properties for any elements $x, y, z \in X$:

- Non-negativity.
  The distance between any two elements x and y is always non-negative, denoted as $d(x, y) \geq 0$. In other words, the distance value is greater than or equal to

[1]Periodic phenomena refer to patterns or behaviors that repeat at regular intervals over time. In the context of angles, it implies that certain patterns or behaviors occur repeatedly as the angle changes. These phenomena can be observed in various natural and physical systems, such as waves, oscillations, and cyclic processes.

zero for any pair of elements in X. Furthermore, the non-negativity property ensures that the distance is zero if and only if the two elements are identical, i.e., $d(x, y) = 0$ if and only if $x = y$.

- Symmetry.
  The distance function is symmetric, meaning that the distance between x and y is the same as the distance between y and x. This property is expressed as $d(x, y) = d(y, x)$. Symmetry reflects the notion that the distance from x to y is the same as the distance from y to x, regardless of the specific elements x and y.

- Triangle inequality.
  The triangle inequality is a fundamental property of distance metrics. It states that the direct distance between two elements x and y is always shorter than or equal to the sum of the distances through an intermediate element z. Mathematically, it is represented as $d(x, y) + d(y, z) \geq d(x, z)$. In simpler terms, the distance traveled directly from x to y plus the distance traveled from y to z is always greater than or equal to the distance traveled directly from x to z. The triangle inequality ensures that the path taken between two points is never longer than a straight-line distance.

These properties ensure that the distance function behaves consistently and reflects the intuitive notion of distance. The non-negativity property asserts that distances are non-negative, with zero distance only occurring between identical objects. Symmetry dictates that the distance from object x to y is the same as the distance from y to x. The triangle inequality asserts that the direct path between two points is always shorter than or equal to any detour through a third point. They are collectively known as the axioms or requirements for a distance metric and establish the fundamental properties that a function must possess to qualify as a distance metric.

By adhering to these properties, a distance metric allows for meaningful comparisons, quantification of dissimilarity or similarity, and the development of mathematical frameworks such as metric spaces. The choice of a specific distance metric

depends on the context, problem domain, and desired properties or characteristics of the distances being measured. Different distance metrics capture different aspects of similarity or dissimilarity between elements, enabling tailored analyses and interpretations in various mathematical disciplines and real-world applications.

In practice, various distance metrics are utilized, each tailored to a specific application or problem domain. Some commonly encountered distance metrics include:

- Euclidean distance.
  The Euclidean distance is a widely employed metric for quantifying the dissimilarity between two points in an n-dimensional space. Its name pays homage to the ancient Greek mathematician Euclid, who made significant contributions to geometry and laid the foundation for this distance measure.

  Given two points $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, the Euclidean distance between them in an n-dimensional space is defined as Equation 3.5:

  $$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.5}$$

  The Euclidean distance corresponds to the straight-line distance between the points and is widely used in geometry and data analysis. Figure 3.6 depicts this line.

- Manhattan distance.
  The Manhattan distance, alternatively known as the taxicab or city block distance, is a prominent metric utilized for quantifying the dissimilarity between two points in an n-dimensional space. Its name derives from the resemblance to the path a taxicab would take when navigating through the city blocks of a grid-like city layout.

  In mathematical terms, consider two points in an n-dimensional space represented by $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$. The Manhattan distance

between these points can be calculated as the sum of the absolute differences between their corresponding coordinates, as shown in Equation 3.6:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.6}$$

The Manhattan distance can be conceptualized as the distance one would travel on a grid-like city layout, where only horizontal and vertical movements are allowed. It represents the sum of individual displacements along each dimension required to traverse from one point to another. This distance metric finds widespread application in diverse fields, including urban planning, transportation, and computer science.

- Hamming distance.
  The Hamming distance is a commonly used metric that is particularly valuable in computer science and information theory. It serves to quantify the dissimilarity between two strings of the same length. This distance is named after Richard Hamming, an esteemed mathematician and computer scientist who made noteworthy advancements in these fields.

  Given two strings of equal length n, string x and string y, the Hamming distance $d(x, y)$ is calculated by counting the number of positions at which the corresponding elements in the two strings differ. The Hamming distance can be expressed as Equation 3.7:

$$d(x, y) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{3.7}$$

Here, $\delta(x_i, y_i)$ represents the Kronecker delta function[2], defined as:

$$\delta(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{otherwise} \end{cases}$$

The Hamming distance is a measure that captures the dissimilarity between two strings by identifying the positions where their corresponding elements differ. It quantifies the number of substitutions needed to transform one string into another. This distance has broad applications in various fields, such as error detection and correction codes, DNA sequence analysis, and data mining. In the context of error detection and correction codes, the Hamming distance is instrumental in identifying and rectifying bit errors during data transmission. In DNA sequence analysis, it facilitates the assessment of genetic variation and the identification of evolutionary relationships. Additionally, in data mining, the Hamming distance is employed for tasks such as clustering, classification, and similarity analysis.

- Minkowski distance.
  The Minkowski distance is a versatile metric that generalizes both the Euclidean and Manhattan distances, incorporating them as special cases. Named after the eminent mathematician Hermann Minkowski, this distance measure finds wide applicability in various domains, including physics, machine learning, and pattern recognition.

  Consider two points in an n-dimensional space denoted as $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$. The Minkowski distance between these points is defined by Equation 3.8, where p is a positive parameter:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{3.8}$$

[2]The Kronecker delta function, denoted as $\delta(i, j)$, is a mathematical function defined as follows: $\delta(i, j) = 1$ if $i = j$, and $\delta(i, j) = 0$ if $i \neq j$. It is commonly used in mathematics and physics to represent the discrete nature of indices or to simplify expressions involving sums and products.

The Minkowski distance incorporates the absolute differences raised to the power of p for each corresponding coordinate, followed by a sum across all dimensions. Finally, the result is raised to the power of $\frac{1}{p}$. By varying the value of p, the Minkowski distance can exhibit different behaviors, encompassing various distance measures. For $p = 2$, it reduces to the Euclidean distance, while $p = 1$ yields the Manhattan distance. If p is adjusted to infinity, denote as $p \to \infty$, the Minkowski distance converges to the Chebyshev distance.

By adjusting the value of p in the Minkowski distance equation, a continuum of distance measures is obtained, allowing for fine-grained control over the trade-off between different aspects of dissimilarity assessment. The Minkowski distance's flexibility and adaptability make it a valuable tool in various fields, including physics simulations, feature extraction, and clustering algorithms.

- Chebyshev distance.
  The Chebyshev distance, also referred to as the chessboard distance or maximum metric, is a mathematical measure employed to quantify the dissimilarity between two points in a multidimensional space. It derives its name from the mathematician Pafnuty Chebyshev, who made significant contributions to various fields, including mathematics and physics.

  Mathematically, the Chebyshev distance between two points, denoted as $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, is defined as the maximum absolute difference between any coordinate of the two points. It can be expressed as Equation 3.9:

$$d(x, y) = \max_{i=1}^{n} |x_i - y_i| \tag{3.9}$$

  By considering the maximum discrepancy between corresponding coordinates, the Chebyshev distance captures the most significant disparity between the two points under consideration. This characteristic makes it particularly suitable for scenarios where the focus is on identifying the maximum deviation across

dimensions.

The Chebyshev distance finds applications in various domains, including game artificial intelligence, image processing, and pattern recognition. For instance, in the field of artificial intelligence applied to gaming, the Chebyshev distance allows for effective evaluation of possible moves by determining the maximum number of steps a chess king would need to traverse on a chessboard to move from one square to another. Similarly, in image processing and pattern recognition, this distance metric proves valuable in comparing objects based on their maximum deviation in any given dimension.

These are just a few examples from a vast array of distance metrics available in mathematics. The choice of the appropriate distance metric depends on the nature of the problem at hand and the specific requirements of the application. Different distance metrics may capture different aspects of similarity or dissimilarity between objects, allowing for more nuanced analysis and interpretation.

Beyond their fundamental role in mathematical theory, distance metrics find extensive applications across various domains. In geometry, distance metrics are crucial for measuring lengths, defining geometric shapes, and studying the properties of geometric objects. In optimization and numerical analysis, distance metrics often serve as error measures or convergence criteria for iterative algorithms. In machine learning and pattern recognition, distance metrics enable clustering, classification, and similarity-based retrieval of data instances. Furthermore, distance metrics play a vital role in data mining, image processing, signal analysis, and numerous other fields where quantifying the dissimilarity between objects is paramount.

Moreover, distance metrics facilitate the establishment of mathematical structures such as metric spaces. A metric space consists of a set of objects along with a distance metric defined on that set. By imposing the metric structure, a multitude of mathematical tools and techniques become applicable, allowing for rigorous analysis and reasoning about the underlying space.

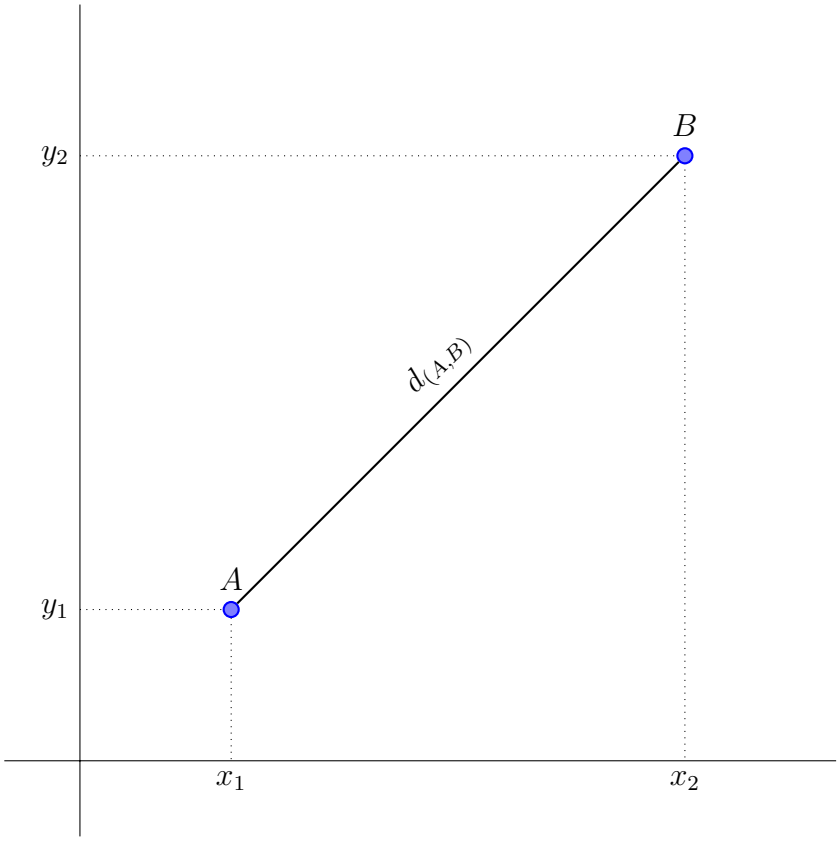


Figure 3.6: Euclidean distance in two-dimensional space in a Cartesian coordinate system. In Euclidean geometry, the line segment $d_{(A,B)}$ represents the shortest distance between two points, A and B. It serves as a direct connection between these points, providing a direct path with minimal length.

In conclusion, distance metrics form an indispensable component of general mathematical knowledge. They provide a mathematical framework to quantify the notion of distance or dissimilarity between objects, enabling comparisons, analysis, and interpretation within various mathematical disciplines and real-world applications. By understanding and utilizing the properties and applications of distance metrics, mathematicians, scientists, and researchers can unlock valuable insights and solve complex problems across a wide range of domains.

3.4 Machine Learning Algorithms

Machine learning algorithms utilize computational methods to efficiently process and analyze data, facilitating learning, and enabling accurate predictions or decisions without relying on explicit programming instructions. These algorithms employ mathematical and statistical techniques, leveraging computational resources to perform intricate calculations and operations on data. By leveraging these computational methods, machine learning algorithms excel in tackling complex tasks and managing vast datasets, empowering computers to autonomously discern patterns, extract meaningful insights, and make informed decisions. By leveraging this training data, these algorithms acquire the capability to make predictions or decisions without the need for explicit programming instructions. As a fundamental cornerstone of modern artificial intelligence systems, machine learning algorithms play a pivotal role in advancing the development of intelligent applications across diverse domains.

Supervised learning algorithms, one prominent type of machine learning algorithm, involve training a model on labeled examples, where the desired output is known. Through a process called training, the algorithm learns patterns and relationships within the data, allowing it to generalize and make accurate predictions on unseen data. Common supervised learning algorithms include decision trees, support vector machines, and multilayer perceptrons.

Unsupervised learning algorithms, on the other hand, work with unlabeled data, seeking to uncover inherent patterns or structures within the dataset. By analyzing the data's statistical properties or similarities, these algorithms can identify clusters, associations, or anomalies. Popular unsupervised learning algorithms include k-means clustering, principal component analysis (PCA), and self-organizing maps (SOMs).

Another category of machine learning algorithms is *reinforcement learning*, which focuses on learning optimal behavior through interaction with an environment. In this setup, an agent learns to take actions that maximize a reward signal, received

as feedback from the environment. Reinforcement learning algorithms have achieved remarkable success in complex tasks, such as game playing and robotics.

Furthermore, there are specialized algorithms designed for specific machine learning tasks. For example, recommendation systems employ collaborative filtering or content-based algorithms to provide personalized suggestions to users. Natural language processing algorithms enable machines to understand and generate human language, supporting applications like sentiment analysis and machine translation.

The choice of machine learning algorithm depends on the nature of the problem, the available data, and the desired outcome. Researchers and practitioners continuously explore and develop new algorithms, aiming to improve accuracy, efficiency, and interpretability. As machine learning continues to advance, these algorithms will play an increasingly significant role in enabling intelligent decision-making, automation, and transformative technological advancements across industries.

3.4.1 Logistic Regression

Logistic Regression is a machine learning algorithm that uses mathematics to find the relationships between two data factors [75]. It then uses this relationship to predict the value of one of those factors based on the other. Typically, the forecast has a limited number of possible outcomes, such as yes or no.

For example, suppose the request is to predict if a website user would use the checkout button after adding anything to their cart. Logistic regression analysis looks at past visitor behaviour, such as time spent on the website and the number of items in the cart. It determines that, in the past, if visitors spent more than five minutes on the site and added more than three items to the cart, they clicked the checkout button. Using this information, the Logistic Regression function can then predict the behaviour of a new website visitor.

Consider the mathematical equation between x and y, known as the logistic regression function or logit function, to understand how Logistic Regression operates. The logit function maps y as a sigmoid function of x. Equation 3.10 describes how the sigmoid function works.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.10}$$

By plotting this Logistic Regression equation, the S-curve is obtained as Figure 3.7.

The logit function returns only values between 0 and 1 for the dependent variable, irrespective of the values of the independent variable. This is how Logistic Regression estimates the value of the dependent variable. Logistic regression methods also model equations between multiple independent variables and one dependent variable.

In many cases, multiple explanatory variables affect the value of the dependent variable. To model such input datasets, Logistic Regression formulas assume a linear relationship between the different independent variables. The sigmoid function can be modified to compute the final output variable as in Equation 3.11:

$$y = f(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n) \tag{3.11}$$

The symbol β represents the regression coefficient. The logit model can reverse calculate these coefficient values when giving it a sufficiently large experimental dataset with known values of both dependent and independent variables.

The logit model can also determine the ratio of success to failure or log odds. The logistic function $f(x)$ could now be represented as log odds as shown in Equation 3.12:

$$f(x) = log(\frac{p}{1 - p}) \tag{3.12}$$

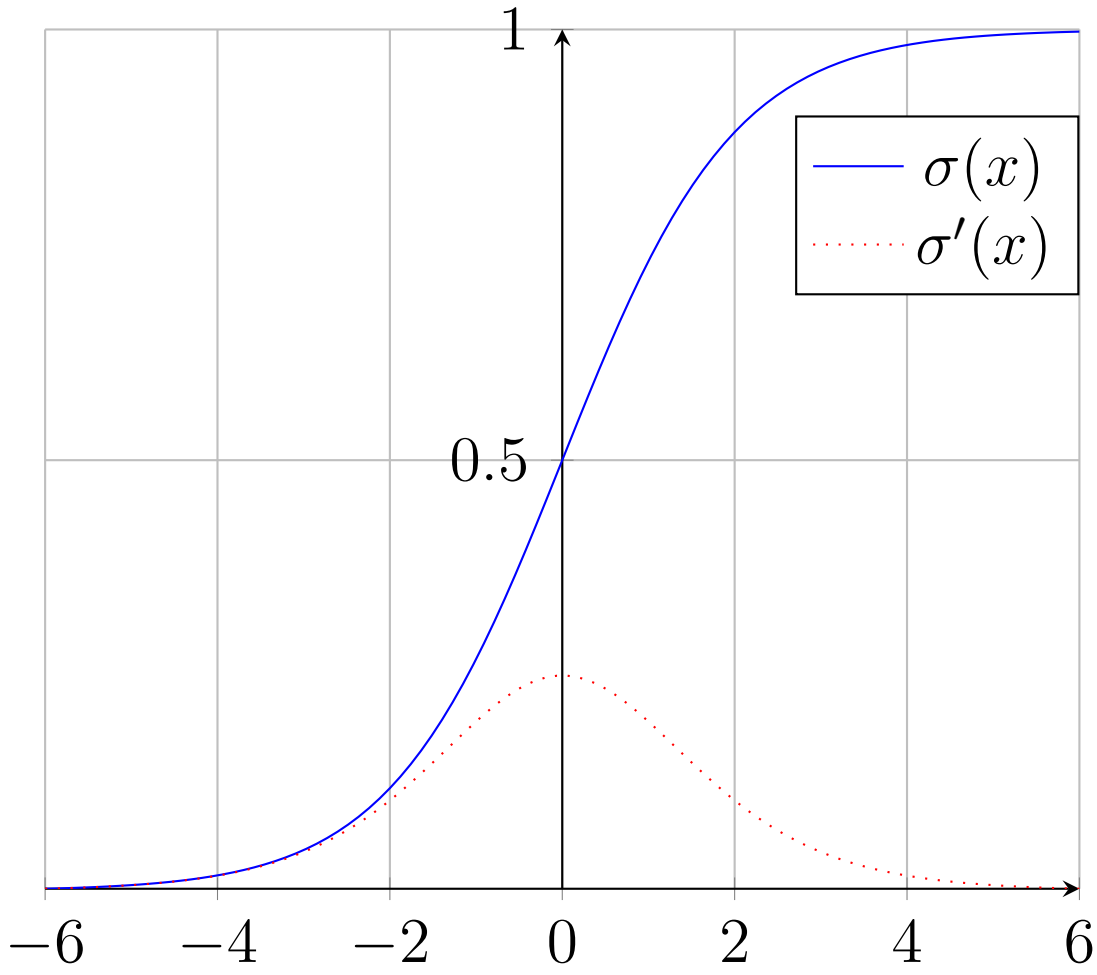


Figure 3.7: Plotting of a sigmoid function and its derivative. The return value (y-axis) of the Sigmoid $\sigma(x)$ function is in the range of 0 to 1.

There are three types to Logistic Regression analysis based on the outcomes of the dependent variable:

- Binary Logistic Regression.
  Binary Logistic Regression is a variation of the logistic regression model that is used when the response variable is binary, i.e., can only take one of two possible categories. Even though the logistic function calculates a range of values

between 0 and 1, the binary regression model rounds the answer to the closest values. Generally, answers below 0.5 are rounded to 0, and answers above 0.5 are rounded to 1, so that the logistic function returns a binary outcome. Assuming that the likelihood of a college basketball player being drafted into the NBA[3] can be predicted using three predictor variables - (1) points, (2) rebounds, and (3) assists, a sports data scientist intends to use a binomial logistic regression model due to the binary nature of the response variable (drafted or not drafted).

- Multinomial Logistic Regression.
  Multinomial Logistic Regression is an extension of the logistic regression model that is used when the response variable has three or more categories, and there is no inherent order among them. The logistic regression model operates by mapping outcome values to a continuous range of values between 0 and 1 using the logistic function. In the context of multinomial regression, this function is utilized to group the output to the nearest possible values. This is possible due to the model's ability to produce a range of continuous data, such as 0.1, 0.11, 0.12, and so on. Assuming that an individual's political preference can be predicted using two predictor variables - (1) annual income and (2) years of education, a political scientist aims to predict the probability of voting for one of four presidential candidates. Since there are more than two possible outcomes (there are four potential candidates) for the response variable and there is no natural ordering among the outcomes, the political scientist would use a multinomial Logistic Regression model.

- Ordinal Logistic Regression.
  Ordinal Logistic Regression is a type of logistic regression model that is used when the response variable has three or more categories, and these categories have a natural ordering or hierarchy among them. Suppose an academic advisor wants to use three predictor variables - (1) GPA[4], (2) ACT[5] score, and (3)

[3]National Basketball Association is a professional basketball league in North America.
[4]Grade Point Average.
[5]American College Testing.

SAT[6] score to predict the likelihood of an individual gaining admission to a university categorized as "bad", "mediocre", "good", or "great". Since the response variable has more than two possible outcomes and these outcomes have a natural ordering, an ordinal logistic regression model would be appropriate for the analysis.

	Bioniminal Logistic Regression	Multinomial Logistic Regression	Ordinal Logistic Regression
Number of categories for response value	2	3 or more	3 or more
Does the order of categories matter?	No	No	Yes

Table 3.1: Differences between types of Logistic Regression. The three types of Logistic Regression are different in terms of the number of response values and whether the order of the categories matter.

Table 3.1 provides a concise summary of the distinctions between the three types of Logistic Regression models, highlighting the differences in the number of response values and the relevance of the ordinal relationship among the categories.

3.4.2 Naive Bayes

Naive Bayes (or Naïve Bayes) is a probabilistic supervised machine learning algorithm that is based on Bayes' theorem [76]. It is used for classification tasks and assumes that the features are independent of each other. In other words, it assumes that the presence or absence of a particular feature in a class is unrelated to the presence or absence of any other feature. Despite this simplifying assumption, Naive Bayes has been shown to be effective in many real-world applications and is widely used in natural language processing, text classification, and spam filtering, among others.

Considering *Bayes theorem*, which finds the probability of an event occurring given

[6]Scholastic Assessment Test.

the probability of another event that has already occurred. Bayes theorem is stated mathematically as the following Equation 3.13:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.13}$$

where

- A and B are events.
- $P(B) \neq 0$.

Bayesian inference is a statistical framework for estimating the probability of an event A, given that event B is true, which is also known as evidence. The probability of event A before observing the evidence is referred to as the prior probability and denoted as $P(A)$. The evidence is typically an attribute value of an unknown instance, represented as event B. The probability of event A after taking into account the evidence is called the posterior probability and denoted as $P(A|B)$. Bayesian inference involves using Bayes' theorem to update the prior probability of event A based on the observed evidence, resulting in a new probability that reflects the degree of belief in event A given the available information.

Specifically, for a given dataset, Bayes' theorem can be applied using Equation 3.14:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{3.14}$$

where:

- y represents the class variable.
- X denotes a dependent feature vector in an n-dimensional space, given by $X = (x_1, x_2, x_3, ..., x_n)$.

Applying the premise of feature independence to the Bayes' theorem, the evidence can be partitioned into its independent components. Consequently, when considering

two independent events, denoted as A and B, Equation 3.15 is applicable:

$$P(A, B) = P(A) \times P(B) \tag{3.15}$$

In Equation 3.15, the joint probability of events A and B is expressed as the multiplication of their respective individual probabilities, $P(A)$ and $P(B)$.

Building upon the foundation of Bayes' theorem and the Naive Bayes algorithm, Equation 3.14 and Equation 3.15 can be combined to yield Equation 3.16, which provides the probability of class y, given the feature vector X:

$$P(y|x_1, x_2, x_3, ..., x_n) = \frac{P(x_1|y)P(x_2|y)P(x_3|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)P(x_3)...P(x_n)} \tag{3.16}$$

After fixing the denominator for a specific input, the probability of the input vector for all possible values of the class variable y needs to be computed in order to create a classifier model. The output with the highest probability is then selected. This can be expressed mathematically as shown in Equation 3.17:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_1|y) \tag{3.17}$$

Therefore, the remaining task is to compute $P(y)$ and $P(x_i|y)$.

As our input is coordinates of landmarks of hand skeletons, which are continuous variables, so we just consider the Gaussian Naive Bayes classifier in this section, which also deals with continuous input.

In *Gaussian Naive Bayes*, an assumption is made that the continuous values of each feature adhere to a Gaussian distribution, also known as a normal distribution. When graphed, this distribution forms a symmetrical bell-shaped curve centered around the mean of the feature values. Figure 3.8 provides a visual representation of this concept.

Since it is presumed that the likelihood of the features is Gaussian, the conditional

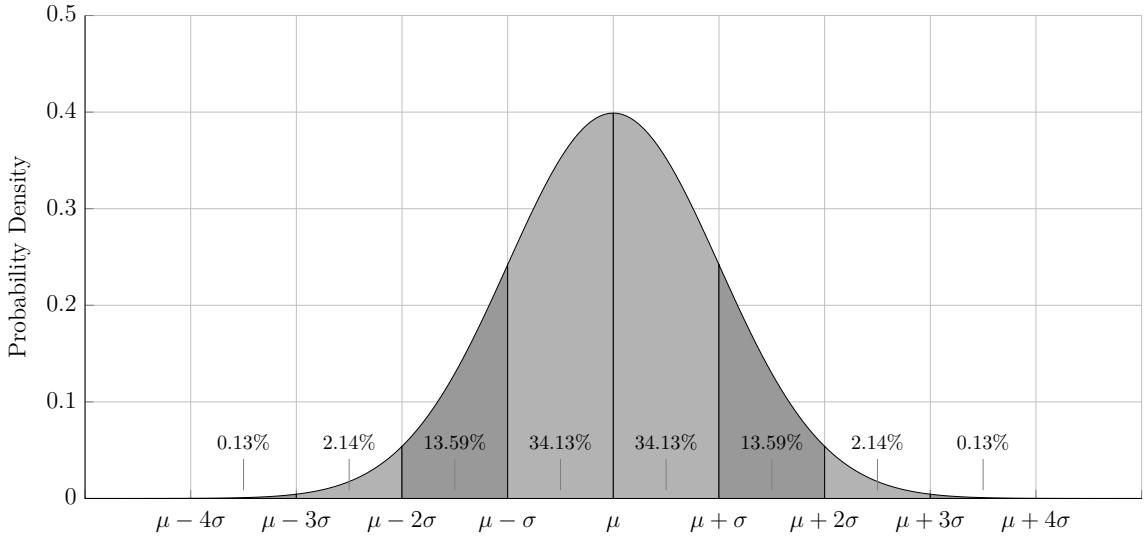


Figure 3.8: The Normal Distribution. A graph of a normal distribution with probabilities between standard deviations σ.

probability is given by using the Gaussian function which is described in Equation 3.18:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \nu_y)^2}{2\sigma_y^2}\right) \tag{3.18}$$

where:

- x_i is a feature of vector X.
- y is a specific class.
- σ_y is the standard deviation of feature x_i for the class y.
- ν_y is the mean of feature x_i for the class y.

3.4.3 Support Vector Machine

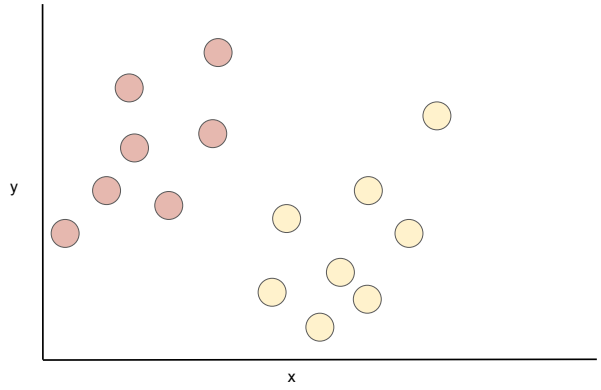
Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression [77]. Though we call them regression

problems, they are best suited for classification. The SVM algorithm's goal is to find a hyperplane in an n-dimensional space that clearly classifies the data points. The size of the hyperplane is determined by the number of features. If there are only two input features, the hyperplane is simply a line. If the number of input features is three, the hyperplane becomes a two-dimensional plane.

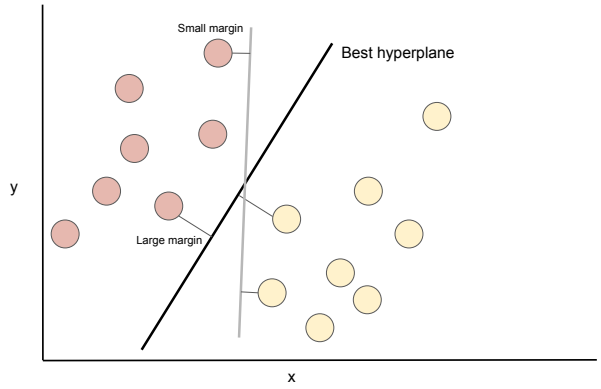
SVM is based on the idea of finding a hyperplane that best separates the features into different domains.

Linear SVM

Given a dataset containing two distinct tags, denoted as brown and yellow, as well as two corresponding features represented by the variables x and y, the task is to design a classifier capable of accurately determining the tag associated with a given pair of coordinates in the (x, y) space. To aid in this task, the labelled training data is plotted on a two-dimensional plane, as depicted in Figure 3.9a.



(a) An example of a 2-dimension setting.



(b) In a 2-D setting, the best hyperplane is simply a line that optimizes the margins from both tags.

Figure 3.9: A visualization of SVM in a simple set of 2-D samples. If the points belonging to different classes are linearly separable, the best hyperplane is simply a line that optimizes the margins from both tags

For SVM, it is the best hyperplane that optimizes the margins from both tags. In other words, the hyperplane (a line in the considered example) with the greatest distance to the nearest element of each tag. Considering our current example, the hyperplane is drawn as in Figure 3.9b.

Nonlinear SVM

The example illustrated in Figure 3.9 is deemed simple due to the fact that the dataset is linearly separable, hence, a straight line can be used to distinguish between the brown and yellow points. However, in reality, datasets are often more complex and challenging to separate. For instance, consider the scenario depicted in Figure 3.10a. As shown in Figure 3.10a, there is no linear decision boundary that can be used to effectively separate the two distinct tags. Despite this, it is apparent that the samples are still relatively easy to separate.

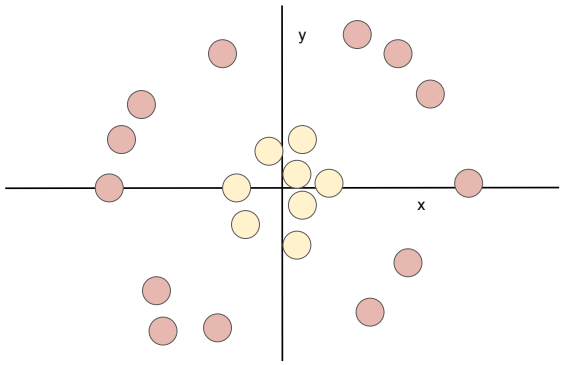
One approach to address this issue is to introduce a third dimension. In the existing two-dimensional space represented by the x and y axes, a new dimension denoted by the variable z can be defined. Specifically, the value of z can be calculated using the equation $z = x^2 + y^2$, which can be interpreted as the equation of a circle.

By introducing the new dimension z as defined in the previous statement, the resulting space becomes three-dimensional. A subset of this three-dimensional space is illustrated in Figure 3.10b when the x and z dimensions are plotted. As can be seen from the figure, the resulting coordinates are now linearly separable, making it easier for the SVM to effectively operate in this space.

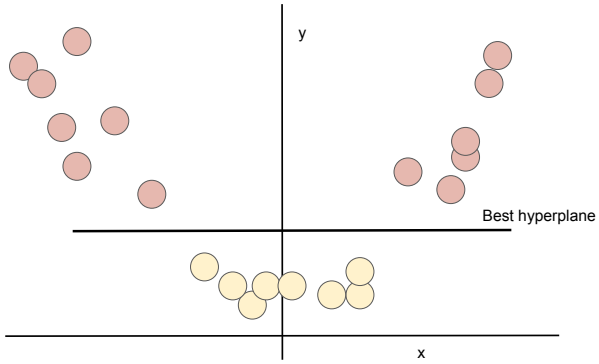
Types of Kernels

SVM techniques utilize a set of mathematical functions collectively known as the kernel. The primary role of the kernel is to accept input data and transform it into a higher dimensional space where it can be more easily separated in a linear manner. There are various types of kernel functions employed in different SVM algorithms, including linear, nonlinear, polynomial, Gaussian radial basis function (RBF), sigmoid, and others.

As a kernel, K takes x and y that are vectors in the original space as input and return an inner product in a feature space based on some mapping φ, as in Equation



(a) A more complex dataset. Considering this dataset, the points belonging to different classes are not linearly separable.



(b) The considered dataset when plotted by the two axes x and z, where $z = x^2 + y^2$. From a different perspective, the data is now in two linearly separated groups.

Figure 3.10: A visualization of SVM in a more complex set of 2-D samples. If the points belonging to different classes are not linearly separable, it is necessary to map the points to other dimensional spaces.

3.19:

$$K(x, y) = \langle \varphi(x)\varphi(y) \rangle \tag{3.19}$$

where:

- K denotes the kernel funcion.

- x and y are vectors in the original space.
- φ is a mapping function to map a vector from the original space to a new space.

We will be focusing on the polynomial and radial basis functions since its most commonly used.

- Polynomial kernel is a widely used kernel in image processing and can be defined using Equation 3.20:

$$K(x, y) = \left(x^\top y + 1\right)^d \tag{3.20}$$

  where x and y are input vectors or data points, and d is the degree of the polynomial which determines the complexity of the mapping.

- Gaussian RBF is a versatile kernel that is typically used when no prior knowledge about the data is available. It can be defined mathematically using Equation 3.21:

$$K(x, y) = \exp\left(-\gamma \|x - y\|^2\right) \tag{3.21}$$

  where γ is a hyperparameter that controls the width of the kernel and $\gamma > 0$, the most preferred value for γ is 0.1.

- Sigmoid kernel is sometimes used as a proxy for neural networks, and can be defined mathematically using Equation 3.22:

$$K(x, y) = \exp\left(\alpha x^\top y + c\right) \tag{3.22}$$

- ANOVA radial basis kernel is a type of kernel that can be used in regression problems, and can be defined using Equation 3.23:

$$K(x, y) = \sum_{k=1}^{n} \exp\left(-\sigma(x^k - y^k)^2\right)^d \tag{3.23}$$

3.4.4 Multilayer Perceptron

The *Multilayer Perceptron* (MLPs) consists of an input layer and an output layer which are fully connected [78]. MLPs have the same input and output layers but may have multiple hidden layers in between the aforementioned layers, as seen below.

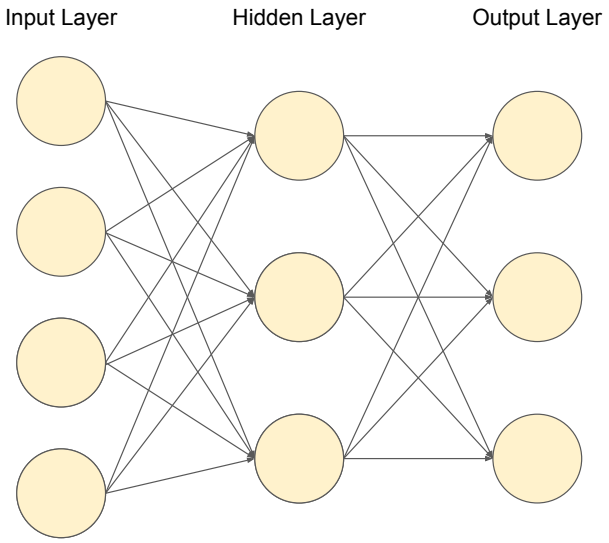


Figure 3.11: The illustration of a Multilayer Perceptron classifier. Adapted from Multilayer Perceptron, retrieved 05/05/2023, from https://deepai.org/machine-learning-glossary-and-terms/multilayer-perceptron.

The algorithm for the MLPs is as follows:

- Similar to the perceptron, the inputs are sent through the MLPs by computing the dot product of the input with the weights between the input layer and the hidden layer. This dot product yields a value at the hidden layer. However, unlike the perceptron, this value is not forwarded.
- Activation functions[7] are essential components in multilayer perceptrons (MLPs) as they introduce non-linearities into the network's computations. There are

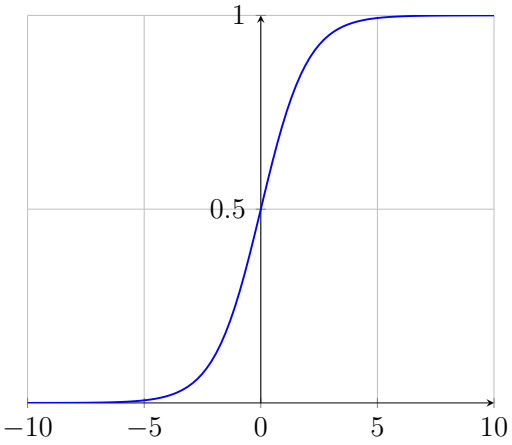
[7]Activation functions are mathematical functions that introduce non-linearity into the computations performed by artificial neural networks. They are applied to the outputs of individual neurons

several activation functions to consider, including rectified linear units (ReLU), the sigmoid function, and tanh. To apply an activation function, pass the computed output of the current layer through one of these functions. Figure 3.12 shows some of the popular activation functions.

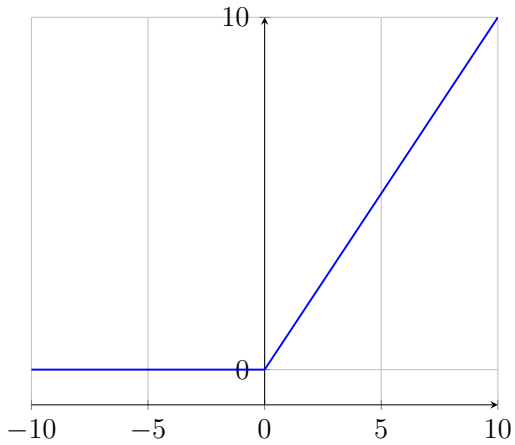
- After calculating the output at the hidden layer and processing it through the activation function, move it to the next layer in the MLPs by taking the dot product with the corresponding weights.
- Iterate steps two and three until the input has propagated through all layers, and the output layer has been reached.
- At the output layer, the calculations will be used either for a backpropagation algorithm that corresponds to the activation function selected for the MLPs (in the case of training) or for making a decision based on the output (in the case of testing).

---

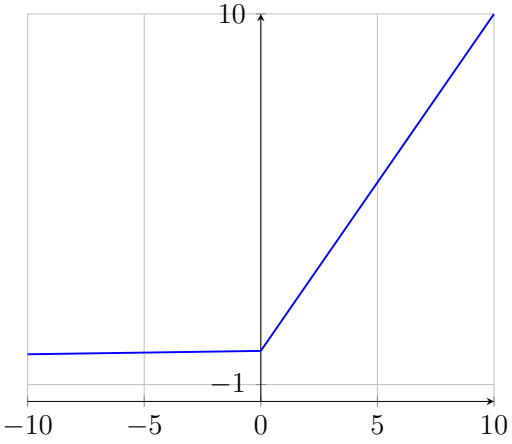
or the entire layers of a neural network. Activation functions play a crucial role in determining whether a neuron should be activated (fired) or not, based on the weighted sum of its inputs.



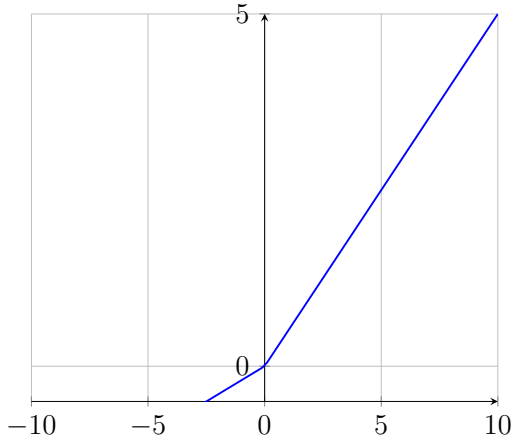
(a) Sigmoid function:
$\sigma(x) = \frac{1}{1+e^{-x}}$.



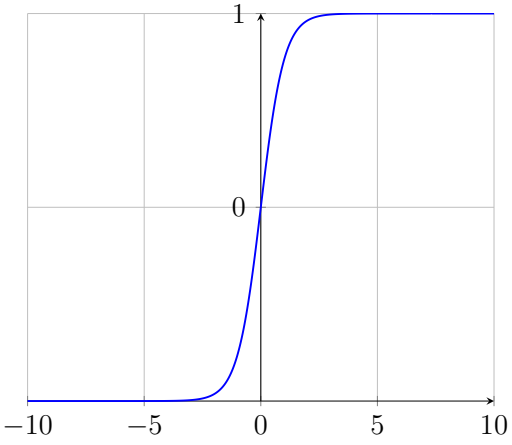
(b) ReLU function:
$f(x) = \max(0, x)$.



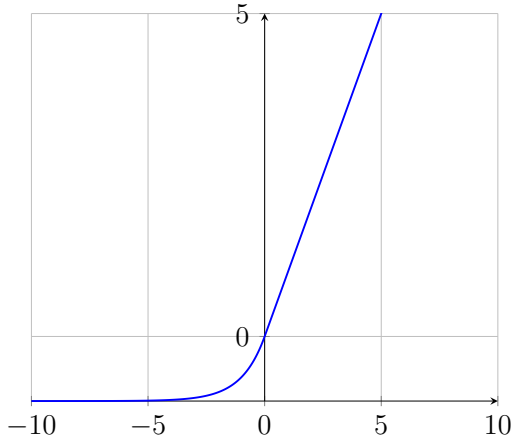
(c) Leaky ReLU function:
$f(x) = \max(\alpha x, x)$.



(d) Maxout function:
$f(x) = \max(w_1^\top x + b_1, w_2^\top x + b_2)$.



(e) Hyperbolic tangent function:
$f(x) = \tanh(x)$.



(f) ELU function:
$f(x) = x$ if $x \geq 0$, $\alpha\left(e^x - 1\right)$ if $x < 0$.

Figure 3.12: Illustration of some common activation functions.

MLPs are the foundation of all neural networks and have significantly enhanced computing capabilities when utilized for classification and regression tasks. With MLPS, computers can now learn intricate and complicated models and are no longer restricted to XOR cases.

3.4.5 Random Forest

For both classification and regression issues, supervised learning algorithms like *Decision Tree* and *Random Forest* are employed. Because Random Forest are a collection of Decision Tree combined, it is easiest to understand these two algorithms together [79]. Naturally, when developing and combining Decision Tree, there are a number of dynamics and parameters to take into account. We will describe the operation of Decision Tree and Random Forest in this section, along with the important factors to take into account when applying these models.

Iteratively asking questions to divide up the data is the foundation of a Decision Tree. This is a Decision Tree's visual representation to make it simpler to understand how the data is partitioned, which is illustrated in Figure 3.13.

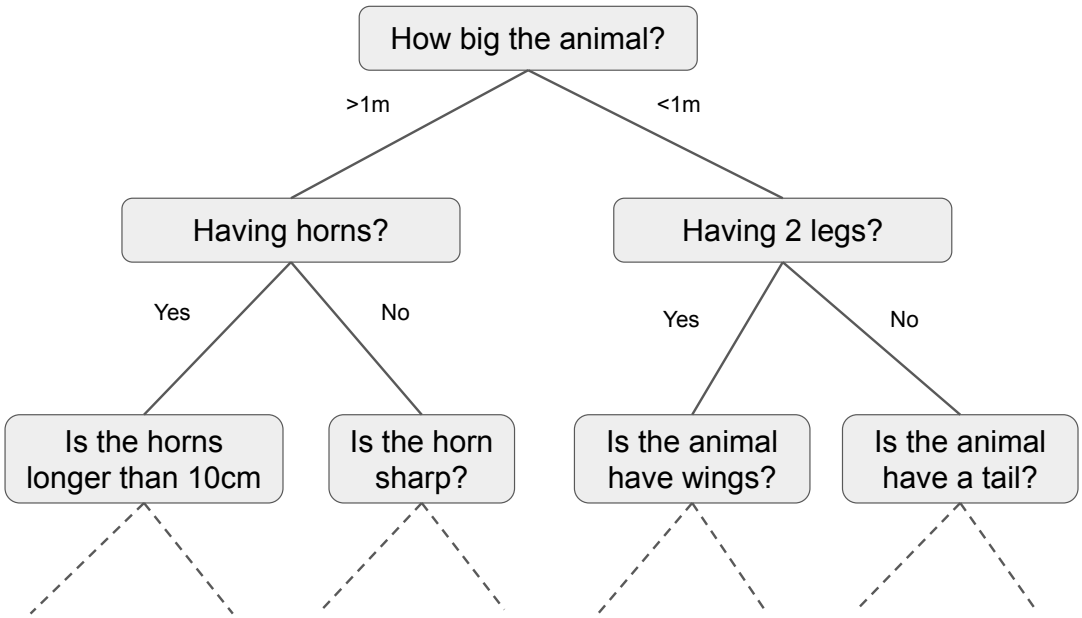


Figure 3.13: A visualization of how a Decision Tree works. From answering questions from general ones to those that are more specific.

This is an example of a Decision Tree used to categorize animals. The first division is based on animal size. Although the question appears to be "how big is the animal?" it is actually phrased as "is the animal bigger than 1m?" because we want to divide the data points into two groups at each step. As going deeper into the tree, the questions become more specific.

The ultimate objective is to enhance the predictiveness of the model at each partitioning so that it can further learn about the dataset. It is often the case that randomly splitting the features does not provide meaningful insights into the dataset. Instead, it is more informative to select splits that augment the purity of a node. A node's purity is determined by the distribution of different classes within that node. Therefore, the questions posed to the model are designed to either increase purity or reduce impurity.

Two common measures of split quality are Gini Impurity and Entropy. These metrics quantify the degree of impurity in a given split and produce comparable outcomes.

Gini Impurity

Gini Impurity is a metric that assesses the frequency at which an element selected randomly from a given subset would be misclassified if it were labelled randomly based on the distribution of labels in that subset. In essence, it quantifies the increase in impurity that results from an increase in randomness. For instance, imagine a box containing ten balls. If all the balls are of the same color, the randomness is absent, resulting in an impurity score of zero. Conversely, if there are five blue balls and five red balls, the impurity score would be one.

The formula for Gini Impurity is described as in Equation 3.24:

$$G = \sum_{i=1}^{C} p(i)(1 - p(i)) \tag{3.24}$$

where:

- C denotes all the unique labels.
- i is one specific label in C.
- $p(i)$ is the probability whether an element is labelled i.

Entropy and Information Gain

Entropy is a measure of randomness or uncertainty. The greater the randomness of a variable, the greater its Entropy. Entropy is greatest for variables with uniform distribution. Rolling a fair dice, for example, has six possible outcomes with equal probabilities, resulting in a uniform distribution and high Entropy.

The Entropy of the whole set of data S can be calculated by using the following

Equation 3.25:

$$H(S) = -\sum_{i=1}^{N} P_i \log_2(P_i) \tag{3.25}$$

where:

- S denotes the considered dataset.
- N is the number of unique values in S.
- P_i is the probability whether an element in S is labelled i.

Splits that result in more pure nodes are chosen. All these indicate *Information Gain* which is basically the difference between Entropy before and after the split, described as in Equation 3.26:

$$\text{Information Gain} = (\text{Entropy before split}) - (\text{Weighted Entropy after split}) \tag{3.26}$$

Or more specifically in Equation 3.27:

$$IG = Entropy(S) - \sum_{i=1}^{k} \frac{|S_i|}{|S|} Entropy(S_i) \tag{3.27}$$

where:

- IG is the Information Gain.
- $Entropy(S)$ is the Entropy of the original dataset S.
- k is the number of subsets created by the split.
- $|S_i|$ is the number of examples in the i-th subset.
- $|S|$ is the total number of examples in S.
- $Entropy(S_i)$ is the Entropy of the i-th subset.

Decision Tree Objectives

During feature selection for splitting, the Decision Tree algorithm aims to achieve the following objectives:

- Increase the predictiveness of the model.
  The Decision Tree algorithm selects features that help the model to make accurate predictions for the target variable. In other words, the model aims to split the dataset based on features that provide meaningful insights into the target variable. By doing so, the model can create subgroups of observations that have similar characteristics, leading to more accurate predictions.

- Minimize the level of impurity.
  Impurity measures the extent to which observations in a group belong to different classes. The Decision Tree algorithm aims to select features that result in splits that minimize impurity in each resulting subgroup. A subgroup with low impurity implies that most of the observations in that subgroup belong to the same class, which increases the accuracy of predictions.

- Decrease the amount of Entropy.
  Entropy is a measure of the level of impurity in a given group. The Decision Tree algorithm aims to select features that result in splits that decrease Entropy in each resulting subgroup. A lower Entropy score indicates that most of the observations in that subgroup belong to the same class, leading to more accurate predictions.

Random Forest Objectives

During the construction of a Random Forest model, the algorithm aims to achieve the following objectives:

- Increase the predictive power of the ensemble.
  The Random Forest algorithm aims to select features that enhance the overall predictive power of the ensemble model. It accomplishes this by considering

a subset of features for each individual decision tree within the forest. By combining the predictions of multiple trees, the Random Forest model can make accurate predictions for the target variable.

- Minimize the correlation among decision trees.
  Correlation among decision trees within a Random Forest can reduce the effectiveness of the ensemble. The algorithm aims to select features and construct decision trees in such a way that the trees are as uncorrelated as possible. This is achieved through random feature selection and random sampling of the training data for each tree. By reducing the correlation, the Random Forest can capture different aspects of the data, leading to improved generalization and robustness.

- Mitigate overfitting and improve generalization.
  Overfitting occurs when a model learns the training data too well, resulting in poor performance on unseen data. The Random Forest algorithm aims to combat overfitting by creating a diverse set of decision trees and aggregating their predictions. This ensemble approach helps to reduce the over-reliance on any individual tree and provides a more generalized model that performs well on unseen data.

- Identify important features for prediction.
  Random Forest models can provide insights into feature importance. By analyzing the average impact of each feature across the ensemble, the algorithm can identify which features contribute the most to prediction accuracy. This information can be useful for feature selection, as well as for understanding the underlying relationships between the features and the target variable.

By pursuing these objectives, the Random Forest algorithm constructs an ensemble model that leverages the strengths of individual decision trees to make accurate predictions while addressing issues such as overfitting and feature importance.

3.4.6 *k*-Nearest Neighbors

The *k-Nearest Neighbors algorithm*, also known as *k*NN or *k*-NN, is a type of supervised learning classifier. It is non-parametric, meaning that it does not make assumptions about the underlying data distribution, and uses proximity to make predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is most commonly used as a classification algorithm, based on the idea that similar data points are typically located near one another.

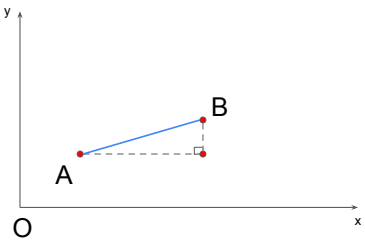
It is known that the *k*-NN algorithm helps to determine the closest points or groups to a query point. But to determine the closest groups or the nearest points for a query point we need some metric. Several distance metrics are available for this purpose:

- Euclidean Distance
- Squared Euclidean Distance
- Manhattan Distance
- Chebyshev Distance

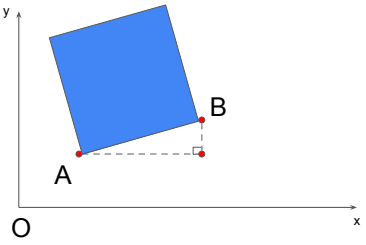
Considering Figure 3.14, there are some examples of popular distance metrics with their corresponding visualization.

The working of the *k*-NN algorithm can be explained through the following steps:

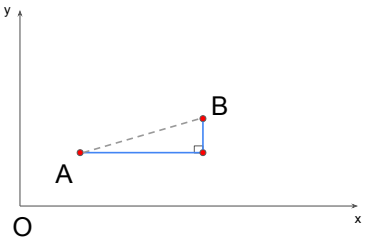
- Step 1: Select the number k of the neighbours.
- Step 2: Calculate the Euclidean distance of k number of neighbours.
- Step 3: Take the k nearest neighbours as per the calculated Euclidean distance.
- Step 4: Among these k neighbours, count the number of the data points in each category.



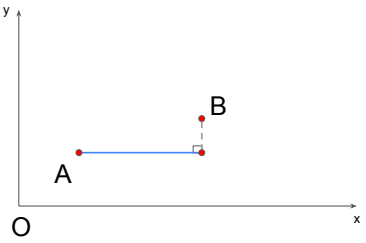
(a) Euclidean distance:
$d(u, v) = \|u - v\|_2$.



(b) Squared Euclidean distance:
$d(u, v) = \|u - v\|_2^2$.



(c) City Block (Manhattan) distance:
$d(u, v) = \sum_i |u_i - v_i|$.



(d) Chebyshev distance:
$d(u, v) = max_i|u_i - v_i|$.

Figure 3.14: Visualizations of some distance metrics with their corresponding formulas. The lines or areas coloured blue are the visualizations of the distance between A and B.

- Step 5: Assign the new data points to that category for which the number of neighbours is maximum.

Suppose a new data point needs to be categorized based on a given dataset. Consider the example dataset shown in Figure 3.15.

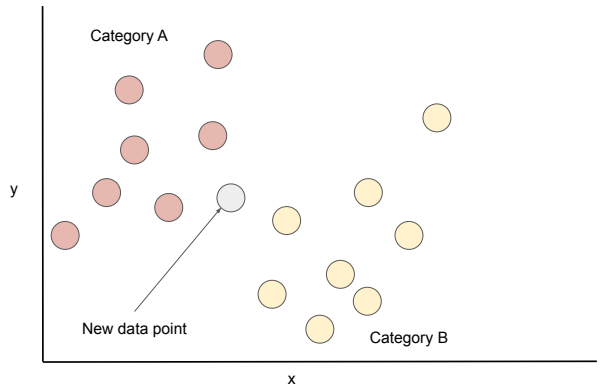


Figure 3.15: A sample data set for k-NN. The data set includes 2 categories A and B, which are red and yellow in colour, respectively.

To begin, the number of neighbors to consider is chosen, in this case, $k = 3$. Next, the Euclidean distance between the new data point and the existing data points is calculated. The Euclidean distance is a measure of distance between two points in space, commonly used in geometry, and can be calculated as $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Referring to Figure 3.16, by calculating the Euclidean distance, the algorithm identifies the two nearest neighbors in category A and one nearest neighbor in category B. Thus, the new data point is classified as belonging to category A.

3.5 Evaluation Metrics

This section introduces the evaluation metrics utilized to assess our solution. We will delve into the details of Precision, Recall, Accuracy, and F-score, providing practical examples for both binary classification and multiclass classification.

In the field of data analysis and machine learning, *evaluation metrics* are quantitative measures used to assess the performance and effectiveness of a model or algorithm

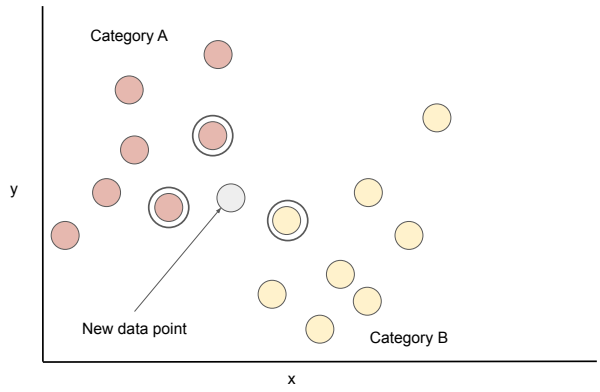


Figure 3.16: Choosing the category for the new data point using k-NN. As two nearest neighbours in category A and one nearest neighbour in category B, the new data point must belong to category A.

[80]. These metrics provide valuable insights into how well the model is performing and help in comparing different models or algorithms.

Evaluation metrics are typically tailored to the specific task at hand and can vary depending on the problem domain

3.5.1 Evaluation Metrics for Binary Classification

In light of the illustration depicted in Figure 3.17, an exemplification of the outcome of a binary-classification problem can be observed, featuring two distinct labels, namely "positive" and "negative". In other words, *binary classification* is a fundamental task in machine learning and data analysis that involves categorizing input data into one of two classes or categories. It is called "binary" because there are only two possible outcomes or classes.

Evaluation metrics such as accuracy, precision, recall and F_1 score are commonly used to assess the performance of binary classification models. These metrics provide insights into how well the model is distinguishing between the two classes and can

help gauge its effectiveness in real-world applications.

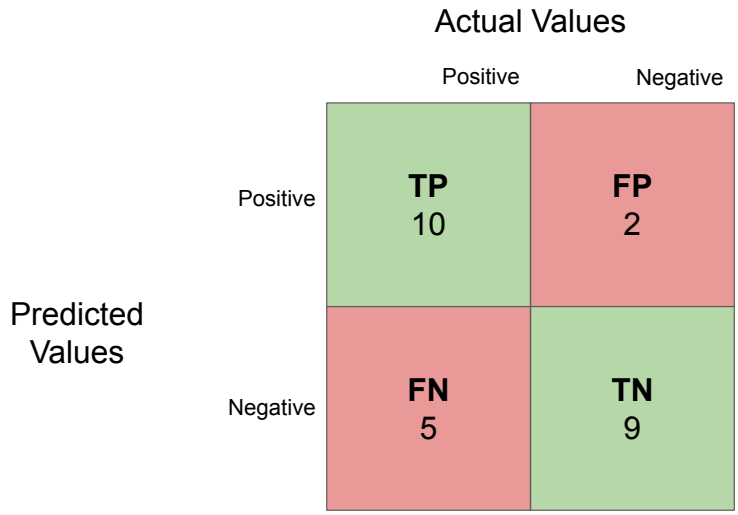


Figure 3.17: The result of a binary-classification problem. There are two labels: Positive and Negative.

Accuracy

Accuracy, as defined in the context of algorithmic predictions, quantifies the ratio of correctly predicted instances in relation to the total number of predictions. Mathematically, it can be computed by dividing the sum of true positives (TP) and true negatives (TN) by the sum of true positives, false positives (FP), true negatives, and false negatives (FN), as shown in Equation 3.28:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.28}$$

Applying this formula to the given example, where TP equals 10, FP equals 2, TN equals 9, and FN equals 5, the accuracy is calculated as follows:

$$Accuracy = \frac{10 + 9}{10 + 2 + 9 + 5} = 0.73$$

Thus, the accuracy of the binary classification problem in question is 0.73.

Precision

Precision, also referred to as positive predictive value, represents the ratio of true positives to the sum of false positives and true negatives. In the provided figures (refer to Figure 3.17), precision can be computed using Equation 3.29:

$$Precision = \frac{TP}{TP + FP} \tag{3.29}$$

By substituting the values from the example in Figure 3.17, where TP equals 10 and FP equals 2, the precision can be calculated as follows:

$$Precision = \frac{10}{10 + 2} = 0.83$$

Hence, the precision for the binary classification problem at hand is determined to be 0.83. Precision is a valuable metric as it illustrates the accuracy of positive predictions specifically, indicating how precise the predictions are within that category.

In cases where the cost of false sensitivity errors is considerably high, precision serves as a suitable measure of accuracy. For instance, in email spam detection, if an email that is not actually spam is incorrectly flagged as spam, it can lead to significant consequences. Users may lose important emails if the precision for spam detection is not sufficiently high. Therefore, precision plays a crucial role in such scenarios, ensuring that false positives are minimized to maintain a high level of accuracy.

Recall

Recall, also referred to as sensitivity or specificity, measures the ratio of correctly predicted outcomes (true positives) to the sum of true positives and false negatives. It can be calculated using Equation 3.30:

$$Recall = \frac{TP}{TP + FP} \tag{3.30}$$

By replacing the number with the data of the problem:

$$Recall = \frac{10}{10 + 5} = 0.66$$

Recall indicates the proportion of positive instances that our model correctly identifies as positives. When the cost of a false negative (misclassifying a positive instance as negative) outweighs the cost of a false positive (misclassifying a negative instance as positive), selecting the best model based on recall becomes crucial.

For instance, consider the context of fraud detection. If a transaction that is predicted as fraudulent turns out to be non-fraudulent for the bank, it can have serious consequences. In such scenarios, maximizing recall becomes essential, as it ensures that the model captures as many positive instances (fraudulent transactions) as possible, reducing the chances of missing potentially harmful cases.

By prioritizing recall, models can minimize false negatives and provide a higher level of confidence in correctly identifying positive instances, thereby mitigating risks and minimizing adverse consequences.

F-score

F-score[8], specifically the F_1 *score*[9], serves as a consolidated metric that combines precision and recall into a single value, providing a comprehensive assessment of a model's performance. It is derived from the harmonic mean of precision and recall. The F_1 score can be calculated using Equation 3.31:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.31}$$

Using Equation 3.31 with data from the problem:

$$F_1 = 2 \times \frac{0.83 \times 0.66}{0.83 + 0.66} = 0.74$$

The F_1 score ranges from 0 to 1, where a higher value indicates a better model performance. By incorporating both precision and recall, it provides a balanced evaluation of a model's ability to correctly identify positive instances (precision) while also capturing the full proportion of positive instances (recall).

3.5.2 Evaluation Metrics for Multiclass Classification

Multiclass classification is a machine learning task in which an algorithm or model is trained to classify instances into three or more distinct classes or categories. It is a type of supervised learning problem where the goal is to assign an input sample to one of several possible classes based on its features or attributes.

Consider Figure 3.18, there is a result of a multi-class classification, which has 4 classes: A, B, C, and D.

In multiclass classification, each instance or data point can belong to only one class

[8]The F-score is a metric that combines precision and recall into a single value using the harmonic mean. It provides an overall measure of a model's performance.

[9]The F_1 score is a specific variant of the F-score that balances the importance of precision and recall by giving them equal weight. It is commonly used when both precision and recall are considered equally important.

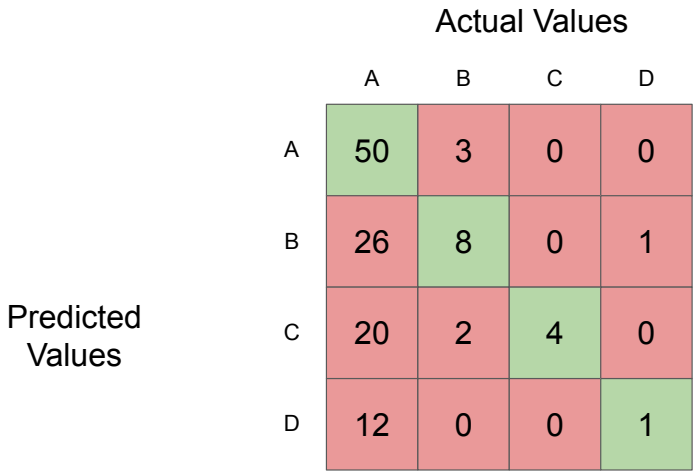


Figure 3.18: The result of a multiclass classification problem. There are 4 classes: A, B, C and D.

out of several possible classes. The model learns patterns and relationships in the input data during the training phase and then uses that knowledge to predict the class labels of unseen instances.

The accuracy for multiclass classification is calculated similarly by finding the ratio of correct predictions out of all predictions made by an algorithm. However, unlike binary classification, multiclass classification generates an F_1 score for each class separately.

$$Accuracy = \frac{50 + 8 + 4 + 1}{3 + 1 + 26 + 20 + 12 + 2} = \frac{63}{128} = 0.49$$

For a multiclass classification problem, the F_1 score is calculated per class using a one-vs-rest approach. This means that we evaluate the success of each class separately, treating them as if there are distinct classifiers for each class.

The precision of class, for example, is calculated as follow:

$$Precision(class = A) = \frac{TP(class = A)}{TP(class = A) + FP(class = A)} = \frac{50}{53} = 0.94$$

$$Recall(class = A) = \frac{TP(class = A)}{TP(class = A) + FN(class = A)} = \frac{50}{108} = 0.46$$

Then, we apply the formula for class A:

$$\begin{aligned} F_1(class = A) &= \frac{2 \times Precision(class = A) \times Recall(class = A)}{Precision(class = A) + Recall(class = A)} \\ &= \frac{2 \times 0.94 \times 0.46}{0.94 + 0.46} = 0.62 \end{aligned}$$

Similarly, we first calculate the precision and recall values for the other classes:

$$Precision(class = B) = \frac{8}{35} = 0.23 \quad Recall(class = B) = \frac{8}{13} = 0.62$$

$$Precision(class = C) = \frac{4}{26} = 0.15 \quad Recall(class = C) = \frac{4}{4} = 1.00$$

$$Precision(class = D) = \frac{1}{13} = 0.08 \quad Recall(class = D) = \frac{1}{2} = 0.50$$

The calculations then lead to per-class F_1 score for each class:

$$F_1(class = B) = \frac{2 \times 0.23 \times 0.62}{0.23 + 0.62} = 0.33$$

$$F_1(class = C) = \frac{2 \times 0.15 \times 1.00}{0.15 + 1.00} = 0.27$$

$$F_1(class = D) = \frac{2 \times 0.08 \times 0.50}{0.23 + 0.62} = 0.13$$

Chapter 4

Methodology

In this chapter, we introduce our hand gesture recognition system, the data collection process and the gaming application that we developed for hand rehabilitation. Firstly, Section 4.1 elaborates on our hand gesture recognition system. Next, Section 4.2 introduced the two datasets that we collected to evaluate the system. Finally, Section 4.3 details on the design and implementation of the gaming application.

4.1 Hand Gesture Recognition

This section details our proposed solution for hand gesture recognition. We first present the overview picture of how the system works in Subsection 4.1.1. In Subsection 4.1.2, we introduce the Leap Motion Controller, the Leap Motion skeletal data and how we extracted the data from the tracking service. Subsection 4.1.3 discusses on hand pose identification, including our strategy for feature extraction and the algorithm we adopted for pose classification. Finally, Subsection 4.1.4 introduces our pose-based approach for hand gesture recognition.

4.1.1 Overview

In this work, we propose a real-time, skeleton-based hand gesture recognition system that recognizes hand gestures based on key pose identification. Our approach is inspired from the work of [81] where they proposed a method that recognizes body

gesture from key body poses. In our approach, we use a different strategy for feature extraction and build a different key pose database that is suitable for hand gestures. For gesture recognition, we adopt a simple dictionary searching strategy instead of using Decision Forests as proposed in [81] to avoid adding more complexity to the system. Thanks to this simple design, our gesture recognition system can easily achieve real-time performance while still ensuring a high accuracy on our self-generated datasets. Our system uses a Leap Motion Controller (LMC) [45] as a hand tracking module to capture hand movement and extract the skeletal data from the user's hand.

The workflow of our system is summarized in Figure 4.1. Firstly, the Leap Motion Controller and the Leap Motion Hand Tracking Software [82] will capture hand movement and extract hand skeletal data containing the 3D coordinates of different hand elements from the user's hand. The data will then be sent to a feature extraction module to extract important features, such as distances and angles between hand joints. After that, the pose classifier will receive the feature vector and identify the hand pose in each frame. The gesture recognizer will then recognize gestures based on which hand poses have been detected, based on dictionary matching and pattern checking. If the sequence of identified hand poses is matched with any of the key pose sequences in the that build up a hand gesture in the key pose dictionary, and the valid gesture pattern is met, the found hand gesture will be recognized by the system. Whenever a hand gesture is recognized, a corresponding action will be sent to the game environment to control the game.

As the motivation of our system is to support hand rehabilitation via an enjoyable gamified approach, we adopted 9 hand gestures, all of which were selected from common hand, wrist and finger rehabilitation exercises [44, 83, 84] that help patients to recover their hand mobility, strength and flexibility [85, 86]. All of those gestures can be performed by one hand, either the left hand or the right hand. The list of gestures and their equivalent hand exercises are shown in Table 4.1.

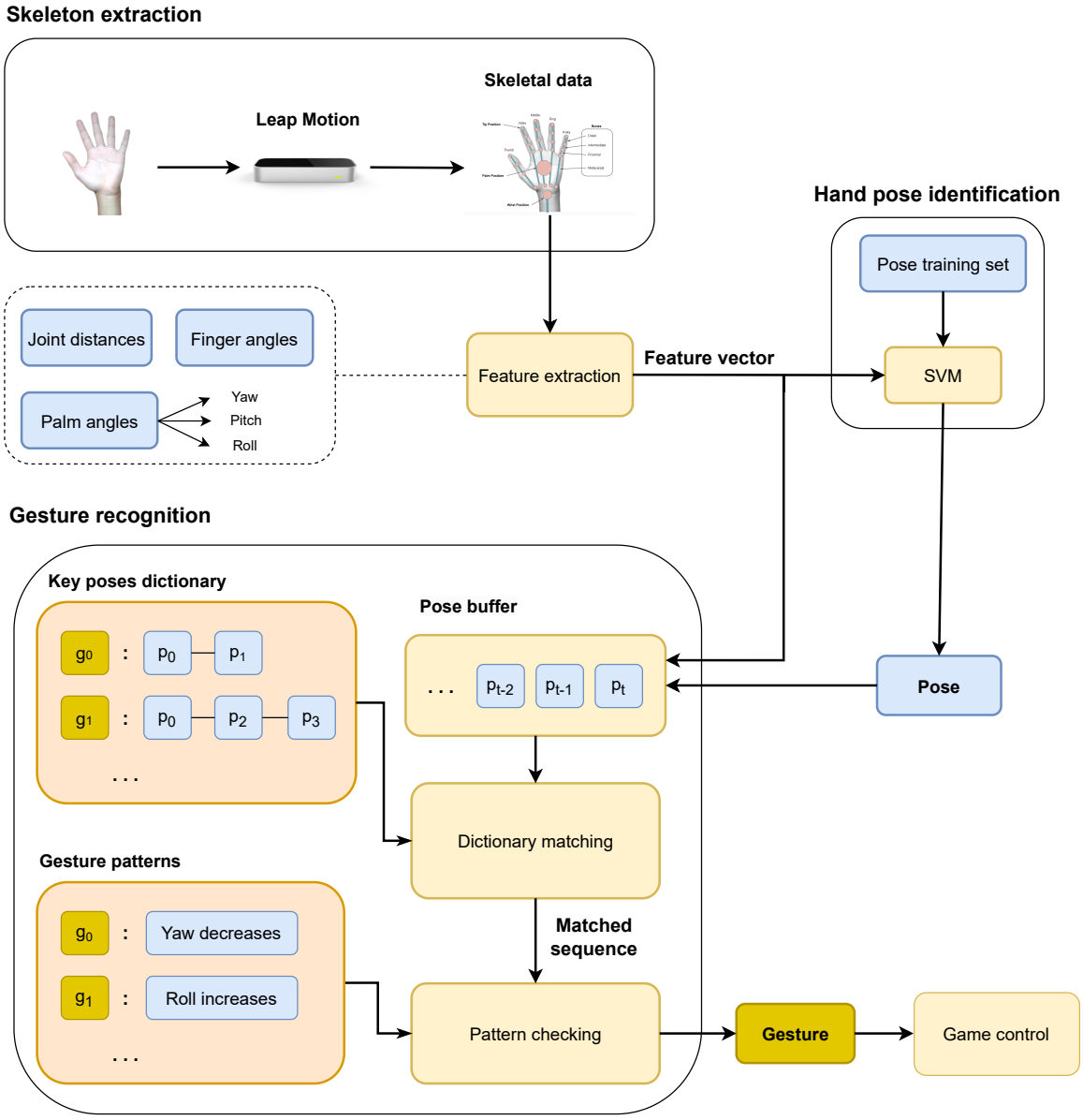


Figure 4.1: Overview of the proposed system. Our system extracts features from Leap Motion data and recognizes the gesture based on key poses classification.

Previous work on hand gesture recognition for hand rehabilitation [16, 87] often adopts a setting in which the patient needs to perform hand gestures in front of a web camera. This can introduce some difficulties as patients with hand problems might feel painful and find it difficult in lifting and keeping the hand in the webcam's

view. We adopt a better setting where the patient's forearm will be put on arm support, and the Leap Motion Controller will be placed below the hand to capture hand movement. By this way, the patient can find it easier and less painful while performing hand gestures, thus our system is more practical for hand rehabilitation. The illustration of this setting is shown in Figure 4.2.

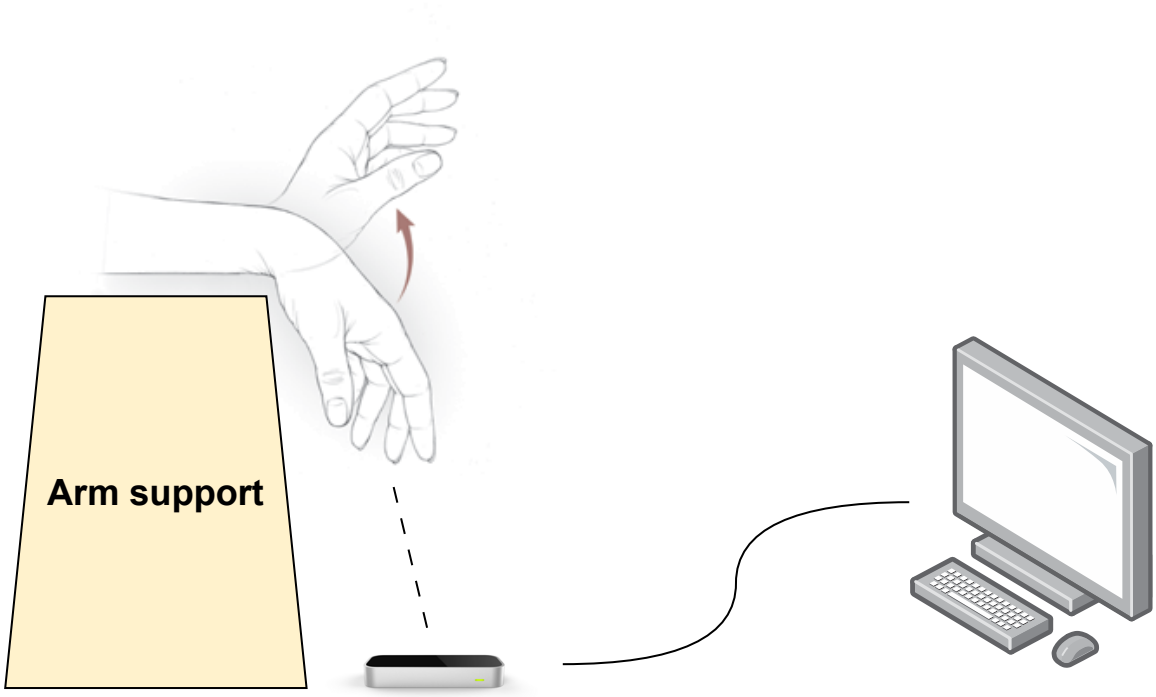


Figure 4.2: The camera view setting of our system. The user's forearm is placed on an arm support and the Leap Motion Controller is placed below the hand to capture hand movements and send the data to the computer for gesture recognition.

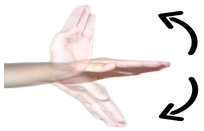
Gesture	Exercises	Health Effects
 Move left/right	Wrist ulnar/radial deviation	Increase mobility and flexibility. Improve grip and pinch strength.
 Move up/down	Wrist extension/flexion	Increase mobility and flexibility. Improve grip and pinch strength.
 Rotate left/right	Wrist supination/pronation	Increase mobility and flexibility. Improve grip and pinch strength.
 Close fist	Close fist	Improve finger mobility and reduce stiffness. Improve range of motion and flexibility of fingers.
 Stop	Spreading and closing fingers	Improve finger mobility and reduce stiffness. Improve range of motion and flexibility of fingers.
 Thumb in	Thumb stretch	Strengthen extensor pollicis longus, extensor pollicis brevis, and flexor pollicis brevis muscles of the thumb.

Table 4.1: List of hand gestures, their corresponding hand rehabilitation exercises, and the health effects of these gestures. Each gesture is equivalent to a hand, wrist or finger rehabilitation exercise.

4.1.2 Skeleton Extraction by Leap Motion Controller

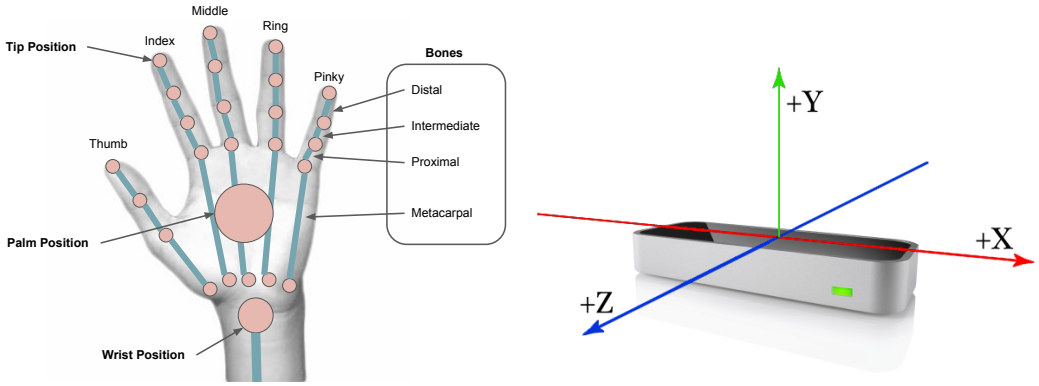
Leap Motion Controller and Leap Motion skeletal data

The Leap Motion Controller, manufactured by Ultraleap[1], is an optical hand-tracking module that is developed to capture the movement of users' hands and fingers. The controller is capable of tracking hands within a 3D interactive zone that extends up to 60 cm ($24''$) or more, extending from the device in a $140 \times 120^{\circ}$ typical field of view. The device operates at a frequency of 120 Hz, enabling it to capture images with exceptional speed and accuracy. Specifically, it can capture an image within $1/2000^{\text{th}}$ of a second. The Leap Motion Controller also comes together with a hand tracking software which is able to extract hand skeletal data from the raw sensor data captured by the controller. That skeletal data will serve as the input to our system. For each time a hand is captured, a skeletal data for that hand is extracted. Each hand skeleton consists of 3D coordinates of 27 distinct hand elements, including joints and bones. Each 3D coordinate is a tuple of three values x, y, z representing the position of an object in the Leap Motion's coordinate system. The Leap Motion's coordinate system sees objects above it in units of real-world millimetres. The origin is located at the top, centre of the hardware. The x-axis and the z-axis lie on the horizontal plane and the x-axis runs parallel to the long edge of the device. The y-axis is a vertical axis perpendicular to the horizontal plane, representing the height from the object to the origin of the device. The skeletal model and the Leap Motion's coordinate system are shown in Figure 4.3.

Data Extraction from the Leap Motion Hand Tracking Service

Ultraleap provides a hand tracking service that reads the data from the Leap Motion Controller and extracts hand skeletal data. In order for our system to read that data, we used Leap C which is a C API for accessing the Ultraleap's tracking data [88]. The LeapC library serves as a vital mediator facilitating the seamless interaction between an application and the hand tracking service. With its implementation

[1]Ultraleap is a technology company that specializes in developing and providing hand tracking and mid-air haptic solutions. Formerly known as Ultrahaptics, the company re-branded as Ultraleap in 2019 after acquiring a fellow haptics company called Leap Motion.



(a) Skeletal model **(b)** Coordinate system

Figure 4.3: The Leap Motion's skeletal model (left) and the Leap Motion's coordinate system (Right). The skeletal model is composed of 27 distinct hand elements. The coordinate system has the $x-$axis and $z-$axis on the horizontal plane and the $y-$axis is a vertical axis perpendicular to the horizontal plane.

of a message queue, LeapC effectively handles the transmission of tracking, image, and status messages originating from the service. This intermediary role assumed by LeapC establishes a reliable and efficient channel through which pertinent information is relayed between the application and the hand tracking service.

As we choose Python as the programming language to develop our system, it is necessary to have a mechanism to read the tracking data into Python. In the past, Leap Motion used to have an SDK to help app developers in Python [89]. But this API is no longer supported nowadays. Therefore, it was challenging for us to program and interact with Leap Motion Controller. Therefore, to use the latest Leap Motion Hand Tracking Service, we had to use LeapC. We faced several issues initially while trying to convert the tracking data from C to Python. Finally, we solved the problem by using CMake and SWIG to create a binding for Python programming language using LeapC. Thanks to that, we could read the tracking data easily from Python.

CMake, a widely used cross-platform build system, plays a significant role in the development and compilation of software projects [90]. It serves as a sophisticated build configuration tool that aids in the automation of the build process by generating

platform-specific build files. CMake enables developers to define the project's build requirements, dependencies, and compilation instructions in a concise and portable manner, abstracting away the intricacies of different operating systems and compilers. With its declarative syntax and modular design, CMake empowers researchers and software engineers to efficiently manage and organize complex projects, promoting code reusability, scalability, and maintainability. By leveraging CMake's capabilities, scholars can streamline the software development workflow, foster collaboration, and facilitate the seamless deployment of their scholarly endeavors across various computing environments.

SWIG, short for Simplified Wrapper and Interface Generator, is a highly esteemed tool widely used in software development and programming languages [91]. Its main purpose is to bridge the gap between various programming languages, making it easier to integrate software components written in different languages. With SWIG, researchers and software engineers can effortlessly create language bindings that enable seamless integration between high-level languages like Python, Java, or C# and lower-level languages such as C and C++. This interoperability empowers developers to leverage the unique strengths and features of different languages, thus enhancing the versatility and extensibility of their software projects. SWIG functions by automatically generating wrapper code, which serves as a translation layer between the desired programming language and the underlying C/C++ code base. Through this process, developers gain access to existing libraries and functionalities written in C or C++ and can effectively utilize them within their preferred high-level programming language.

In this work, we followed the instructions presented in the LeapCxx repository [92] provided by the manufacturer. The idea is to create a Python SDK from LeapC using SWIG and CMake so that our system can read the tracking data directly from Python environment. We also found another implementation of the Python SDK for Leap Motion on a Github repository named RoseMotion [93]. The SDK version we used for this project is Ultraleap 5.3 Gemini for Python 3.9.

4.1.3 Hand Pose Identification

Feature Extraction

From the skeletal data captured by the Leap Motion Controller, a set of important features are extracted for hand pose identification. All of these features were determined by our research on existing work, and mainly through our experiments. We first extract 13 local features, including the Euclidean distances between the palm center and the five fingertips, the distances between adjacent fingertips and the angles between adjacent fingers. Such local features not only extract useful hand information but also acts as a normalizer that helps remove the noise effects of varying hand positions and angles relative to the Leap Motion's sensors. In addition, three global features are added, including the yaw, pitch, roll angles of the hand. These features represent the global movement of the user's hand relative to the Leap Motion Controller. From our experiments, we observe that those features have significant impacts on the recognition of gestures that involve palm movement and rotation, such as `Move left` or `Rotation`. In total, there are 16 features extracted from the skeletal data. The mathematical process of calculating such features is as follow.

Let P denote the point representing the center of the palm within the Leap Motion's coordinate system. The five fingertips, denoted as F_i where $i \in [1, 5]$, correspond to the thumb, index finger, middle finger, ring finger, and pinky finger, respectively.

The distance-based features can be calculated as in Equation 4.1 and Equation 4.2:

$$D_i = ||F_i - P||, i \in [1, 5] \tag{4.1}$$

$$D_{F_i} = ||F_{i+1} - F_i||, i \in [1, 4] \tag{4.2}$$

where D_i represents the Euclidean distances between the palm center and the five fingertips, and D_{F_i} represents the distances between every pair of adjacent fingertips.

The angles between adjacent fingers are calculated as in Equation 4.3 4.2:

$$\alpha_i = \arccos\left(\frac{(F_i - P).(F_{i+1} - P)}{||F_i - P||.||F_{i+1} - P||}\right), i \in [1, 4] \tag{4.3}$$

where α_i represents the angles (in radians) between every pair of adjacent fingers.

The yaw, pitch and roll angles are the features that describe the hand movement. Such features are useful in distinguishing hand poses that differ in angles and direction, such as `Left`, `Right`, `Up`, `Down` or `Palm up`. The visualization of yaw, pitch, roll angles on the coordinate system of Leap Motion is shown in Figure 4.4.

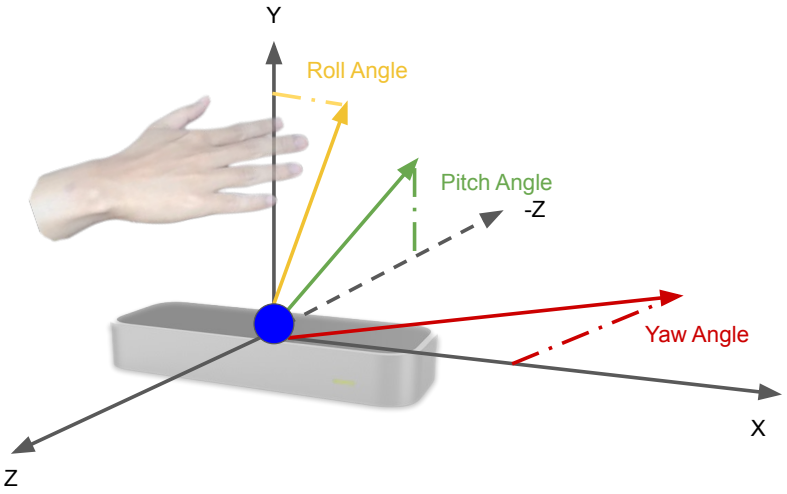


Figure 4.4: The visualization of yaw, pitch, roll angles on the coordinate system of Leap Motion.

Yaw is the angle describing the rotation of the hand around the y-axis. Yaw is calculated as the angle between the negative z-axis and the projection of the hand direction vector onto the Oxz plane. Yaw angle varies from $-\pi/2$ to $\pi/2$ radians when the

user performs the `Move left` or `Move right` gesture.

Pitch is the angle describing the rotation of the hand around the x-axis. Pitch is calculated as the angle between the negative z-axis and the projection of the hand direction vector onto the Oyz plane. Pitch angle varies from $-\pi/2$ to $\pi/2$ when the user performs the `Move up` or `Move down` gesture.

Roll is the angle describing the rotation of the hand around the z-axis. Roll is calculated as the angle between the negative y-axis and the projection of the palm normal vector onto the Oxy plane. Roll angle varies from $-\pi$ to π when the user performs the `Rotate left` or `Rotate right` gesture.

All of the yaw, pitch and roll angles are calculated by the Leap Motion Hand Tracking Software and are extracted as features for hand pose classification.

After feature extraction, we obtain a feature vector V consisting of 16 distinct features, as shown in Equation 4.4:

$$V = (D_1...D_5, D_{F_1}...D_{F_4}, \alpha_1...\alpha_4, yaw, pitch, roll) \tag{4.4}$$

The feature vector is then standardized and be used as the input for hand pose classification.

Hand Pose Classification

Following the process of feature extraction, the hand pose classifier will be employed for the purpose of hand pose identification. In this work, a set P of of 12 hand poses have been carefully selected and will be utilized for classification. These poses serve as the key components in the construction of hand gestures. The name of these poses and their visual representation can be observed in Table 4.2 and Figure 4.5.

For hand pose classification, several machine learning algorithms were tested. We

Pose	ID
Palm	p_0
Left	p_1
Right	p_2
Up	p_3
Down	p_4
Fist	p_5
Hook	p_6
Stop	p_7
Thumb in	p_8
Palm left	p_9
Palm right	p_{10}
Palm up	p_{11}

Table 4.2: 12 key poses and their IDs. Each pose serves as a key component in the construction of hand gestures.

found that Support Vector Machine (SVM) with radial basis function (rbf) kernel demonstrated the highest accuracy of 96.84% on our pose dataset (refer to Section 5.1 for more details). Therefore, we choose SVM with rbf kernel as the pose classifier of the system.

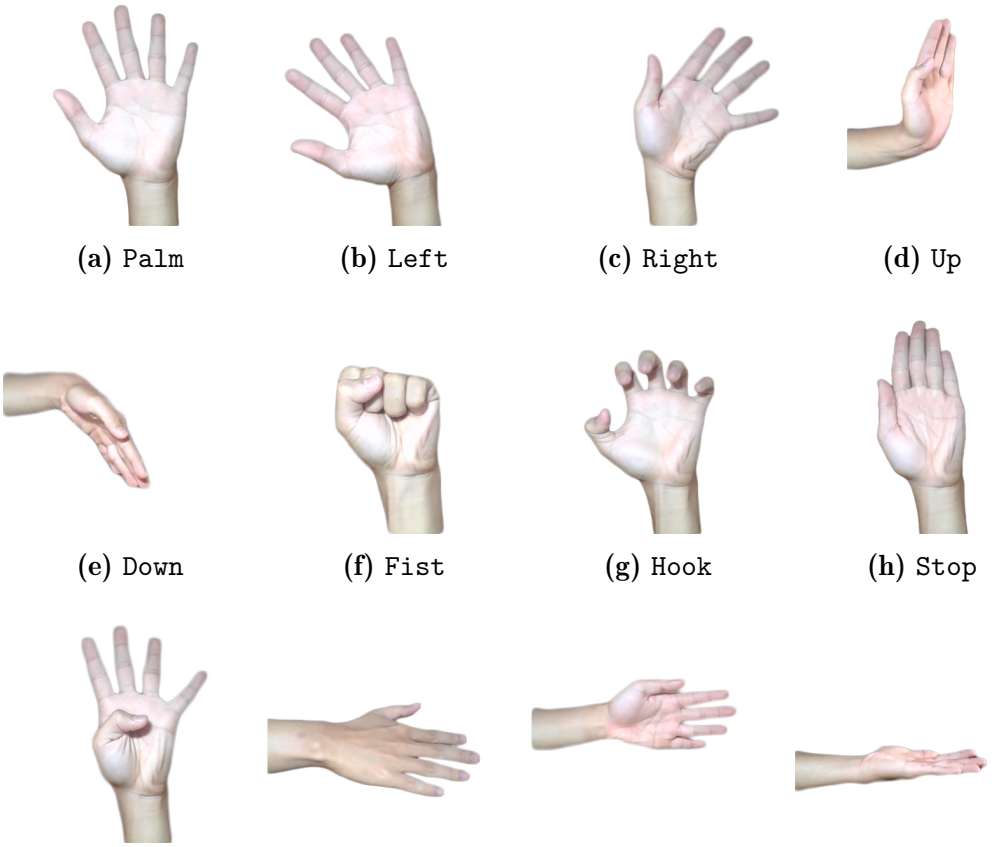
SVM Formulation

Given the training set $T = \{(v_1, y_1), (v_2, y_2), ..., (v_n, y_n)\}$ having n samples where v_i is the feature vector of the i^{th} sampple and c_i is the label of that sample, $y_i \in P$ of all possible hand poses. For each pose class $p \in P$, a decision function f_p (Equation 4.5) is constructed from the training samples to decides if a new sample v belongs to class p or not.

$$f_p(v) = \sum_{j \in SV} \alpha_j c_p(y_j) K(v_j, v) + b \tag{4.5}$$

where:

$$c_p(v) = \begin{cases} 1 & \text{if } y_j = p, \\ -1 & \text{otherwise} \end{cases} \tag{4.6}$$



(a) `Palm` **(b)** `Left` **(c)** `Right` **(d)** `Up`

(e) `Down` **(f)** `Fist` **(g)** `Hook` **(h)** `Stop`

(i) `Thumb in` **(j)** `Palm left` **(k)** `Palm right` **(l)** `Palm up`

Figure 4.5: The visual representation of the 12 selected key poses. Each pose serves as a key component in the construction of hand gestures.

SV is the set of support vectors that define function f_p, α_j are the dual coefficients, K is the kernel function and b is the intercept. The rbf kernel function K for two vectors v_1 and v_2 is shown in Equation 4.7:

$$K(v_1, v_2) = \exp\left(-\gamma||v_1 - v_2||^2\right) \tag{4.7}$$

where γ is a hyperparameter that defines how much influence a single training sample has. The larger γ is, the closer samples must be to be affected. In our experiments, setting $\gamma = 0.01$ gives the highest performance in our self-generated datasets.

As there are $|P|$ possible classes, $|P|$ decision functions are defined. The final hand pose will be determined based on the decision function that gives the highest confidence score among all the decision functions. Given a sample v, the class of that sample will be determined as Equation 4.8:

$$f(v) = \begin{cases} k = \arg\max_p f_p(v) & \text{if } f_k(v) > 0, \\ -1 & \text{otherwise} \end{cases} \tag{4.8}$$

where k is the pose class that has the highest confidence score. If all the decision functions return non-positive values, then the sample v does not belong to any classes in the training set.

4.1.4 Hand Gesture Recognition from Key Poses

Once a specific hand pose has been successfully identified, the system will proceed to recognize hand gesture from the sequence of identified hand poses.

Gesture Definition

In our setting, a gesture g is represented as a sequence of key poses $g = (p_1, p_2, ..., p_{g_n})$, where p_i belongs to a finite set of key poses P. Most of the gesture is composed of 2 to 3 key poses and can have different combinations of key poses. For example the gesture `Move left` is composed of either `Palm` $\rightarrow$ `Left` or `Stop` $\rightarrow$ `Left`, whereas the gesture `Close fist` is composed of either `Palm` $\rightarrow$ `Fist` or `Palm` $\rightarrow$ `Hook` $\rightarrow$ `Fist`.

We built a key pose dictionary that contains all the mappings from gestures to key pose sequences. Each record in the dictionary is a pair of key-value (k, v) where k is a hand gesture and v is the list containing all possible key pose sequences that can be used to build up that gesture. As there are several ways that a user can perform a gesture, different combinations of poses may appear during the gesture. To ensure the generalization of our approach, 6 subjects were asked to perform each gesture in several ways. We then captured the sequences of poses and manually inserted all possible key poses that build up a gesture into the key pose dictionary. The final key

pose dictionary are shown in Table 4.3.

Getsture	ID	Key Pose Sequences
Move left	g_0	(p_0, p_1), (p_7, p_1)
Move right	g_1	(p_0, p_2), (p_7, p_2)
Move up	g_2	(p_0, p_3), (p_7, p_3)
Move down	g_3	(p_0, p_4), (p_7, p_4)
Rotate left	g_4	(p_0, p_{10}, p_{11}), (p_7, p_{10}, p_{11})
Rotate right	g_5	(p_0, p_9, p_{11}), (p_7, p_9, p_{11})
Close fist	g_6	(p_0, p_5), (p_0, p_6, p_5), (p_0, p_8, p_6, p_5)
Stop	g_7	(p_0, p_7, p_0)
Thumb in	g_8	(p_0, p_8, p_0), (p_7, p_8, p_7)

Table 4.3: Key pose dictionary for gesture recognition. Each gesture can be defined by a finite sequences of key poses.

Dictionary Matching

When a hand pose p_t is identified by the pose classifier at time t, it will be inserted in a pose buffer β. The gesture recognizer will then read the sequence of all identified poses $s = (p_{t-n}, ..., p_{t-1}, p_t)$ from the buffer and search from the key pose dictionary to see if there is any key pose sequence $x = (p_1, ..., p_{g_n})$ contained in s. In our design, a gesture cannot be contained in another gesture. Therefore, the search from the dictionary will return at most one key pose sequence at a time. If a the match is found, the matched gesture g composed of the sequence x, and the matched hand pose sequence m inside the buffer will be outputted. The buffer will then be cleared and ready to store the next hand pose. If there is not match, nothing will be outputted. The buffer will continuously store new identified hand poses until a match is found.

Pattern Checking

Through experiments, we found that only using dictionary matching creates several false positive cases. This issue is mainly due to errors made by the pose classifier. When there is a false positive case recognized by the pose classifier (e.g. misclassifying hand poses), the wrong pose will be added to the buffer and might result in an unwanted match in a key pose dictionary. This lead to unwanted hand gestures being recognizes by the system, thus decreasing the overall performance.

To alleviate the issue, we added a pattern matching process to make sure the matched hand gesture follows a valid movement pattern. When adding a new identified hand pose into the pose buffer, some features extracted from that hand pose will also be inserted. These features include the yaw, pitch, roll angles, the average finger angles and the distance between the thumb tip and the palm. After dictionary matching, the matched sequence m and the matched gesture g will be checked against a set of pre-defined patterns. These patterns are specific to each gesture class and involve the valid movement required for each gesture. For directional hand gestures such as `Move left`, `Move up` or `Rotate right`, the valid pattern involves an increase/decrease in the yaw, pitch, roll angles and the increased/decreased value reaching a minimum threshold. Table 4.4 shows the hand gesture classes and the valid movement pattern that is checked for each hand gesture.

After pattern matching, if the matched pose sequence has a valid pattern, the gesture will be recognized. The recognized gestures will then be used to send control to the game environment. On the other hand, if the matched hand pose sequence fails to match the valid gesture pattern, the matched gesture will be discarded. More details on our gesture recognition algorithm is described in Algorithm 1.

Gesture	Feature to Check	Valid Pattern
`Move left`	Yaw angle	Decrease at least 5°
`Move right`	Yaw angle	Increase at lease 5°
`Move up`	Pitch angle	Increase at least 5°
`Move down`	Pitch angle	Decrease at least 5°
`Rotate left`	Roll angle	Increase at least 60°
`Rotate right`	Roll angle	Decrease at least 60°
`Stop`	Average finger angle	Decrease at least 2°, then increase at least 2°
`Thumb in`	Thumb palm distance	Decrease at least 20 mm, then increase at least 5 mm

Table 4.4: Valid pattern for each hand gesture. The matched gesture and matched hand pose sequence will be checked against these patterns.

Algorithm 1 Pose-based hand gesture recognition

$\beta \leftarrow ()$ ▷ Initialize an empty buffer
dictionary $\leftarrow$ load_key_pose_dictionary() ▷ Load the key pose dictionary
patterns $\leftarrow$ load_gesture_patterns() ▷ Load the gesture patterns
while True **do**
    $p_t \leftarrow$ hand_pose_identification() ▷ Identify hand pose at time t
    store the hand pose p_t into the pose buffer β
    $s \leftarrow$ read_buffer(β) ▷ Read the sequence of identified hand poses
    for (g, x) in dictionary **do** ▷ Start dictionary matching
        if x is contained in s **then**
            m = extract_matched_sequence(s, x)
            pattern = get_gesture_pattern(*patterns*, g)
            if is_pattern_matched(m, *pattern*) **then** ▷ Pattern checking
                recognize gesture g
            end if
            empty buffer β
            break
        end if
    end for
end while

4.2 Data Collection

We created two different datasets, a pose dataset for training and testing the pose classifier, and a gesture dataset to evaluate the approach we proposed for hand gesture recognition. There are 6 subjects, four men and two women, involved in the process of data collection. The datasets were collected at different locations, both indoor and outdoor to ensure the generalization to different background and lightning conditions. All the 6 subjects involved in data collection are healthy, belonging to different genders and different age groups. The general information of the 6 subjects is summarized in Table 4.5.

Subject	Gender	Age (years)	Status
1	Male	10-30	Healthy
2	Male	10-30	Healthy
3	Male	10-30	Healthy
4	Female	10-30	Healthy
5	Male	10-30	Healthy
6	Female	30-50	Healthy

Table 4.5: Information of the 6 subjects involved in data collection. The subjects are all healthy, belong to different genders and different age groups.

While collecting data, each subject placed their forearm on an arm support and generated samples of different poses and gestures that are necessary to evaluate the system. The Leap Motion Controller was placed below the hand to capture hand movement and send the data to the computer. After that, the Leap Motion hand tracking service read the sensor data and extracted skeletal information from the subject's hand. Finally, the skeletal data was recorded, cleaned and saved to the storage.

During data collection, all subjects were asked to generate add different variations to the samples to diversify the data. While a subject was generating data, the other ones monitored the process to avoid any mistakes and ensure the data accuracy.

4.2.1 Pose Dataset

The pose dataset was collected to train and evaluate the pose classifier of the system. This dataset consists of 12 distinct hand poses that are identified by the system. These poses have already been introduced in Subsection 4.1.3 and Figure 4.5. Each sample in the dataset is the skeletal data extracted in each time a hand was captured by the Leap Motion Controller.

Collecting Data

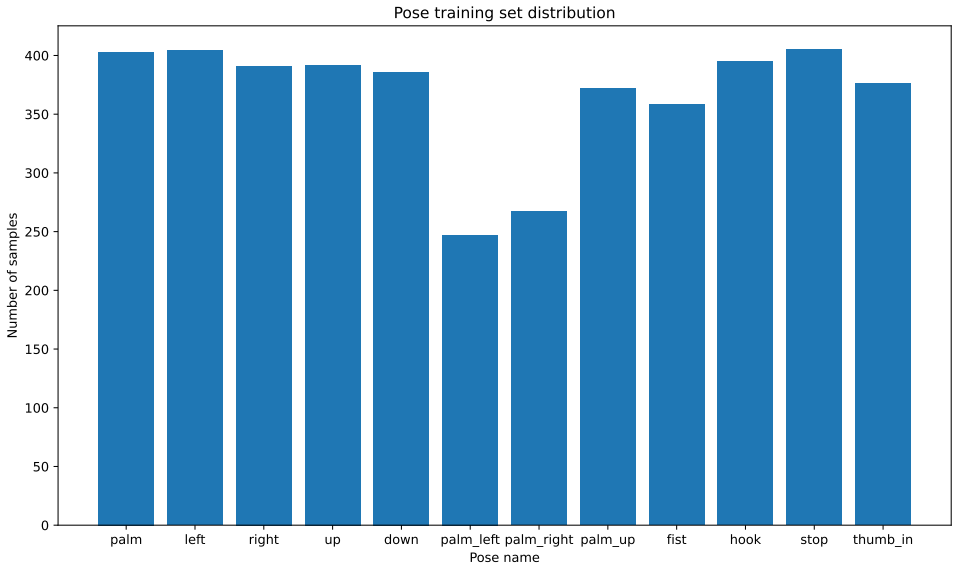
The dataset was generated by all the 6 subjects. For each hand pose, each subject was asked to make the hand pose and move their fingers and palm slightly to generate different variations of the pose. The system recorded 6 sequences of 200 frames, 3 performed in left hand and 3 performed in right hand, then 20 frames were sampled from each sequence. There are 120 samples per subject for each hand pose (60 in left hand and 60 in right hand). In total, there are 720 samples collected for each hand pose, except for the two poses `Palm left` and `Palm right`. As these two poses can only be performed by either left or right hand, we collected a smaller number of samples of 480 samples for each of these two poses.

For a robust evaluation, we splitted the dataset into two parts, a training set containing the samples collected from four subjects and a test set containing the samples collected from the other two. The training set is used to train the pose classifier and the test set is used to evaluate the trained model.

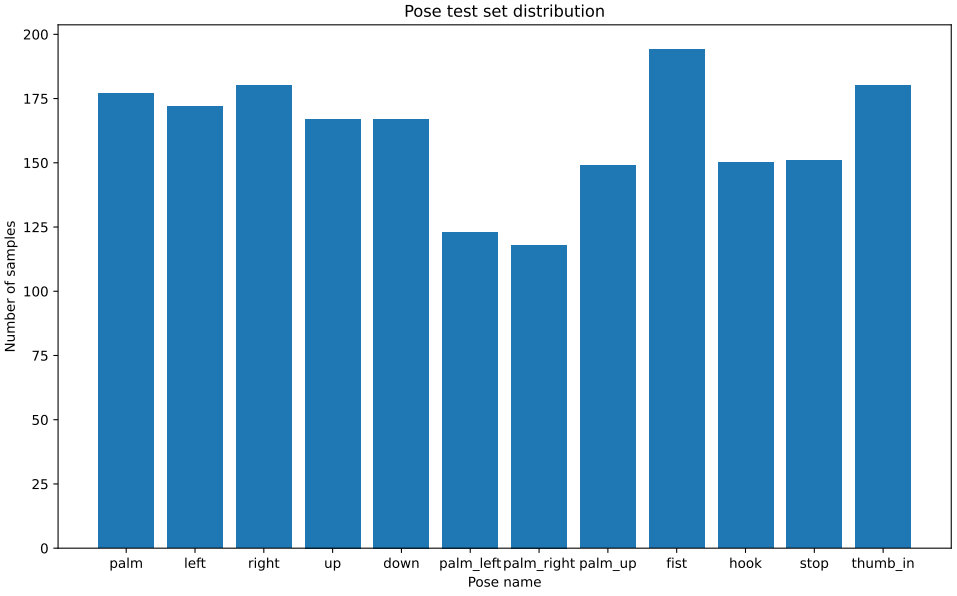
Cleaning Data

After data collection, we observed that there were many cases where the device failed to capture the hand, leading to the lost of skeletal data. We removed all of the empty samples in the dataset. Also, there are some mistakes made by the subjects while generating data, such as making the wrong poses or performing the hand gestures incorrectly. We also removed all of those cases.

Another issue is the wrong extracted skeletal data. This can be due to either the errors made by the hardware device or the errors of the hand tracking service. To overcome the issue, we used the row, pitch and yaw angles to double check the correctness of the data and rule out the incorrect cases. For example, we removed all of the `Up` samples with the pitch angle below 15 degrees. Similarly, all of the `Down` samples with pitch angle larger than 20 degrees were ruled out. We chose these thresholds from the average angles of each hand pose class in the dataset. All of the hand pose classes and their valid ranges of angles are shown in Table 4.6. Any sample with out of range angles is considered wrong and was removed from the dataset.



(a) Training set distribution.



(b) Testing set distribution.

Figure 4.6: Training set and test set distribution of the pose dataset. There are 12 hand pose classes in total.

Pose	Angle	Valid Range (°)
`Left`	Yaw	Greater than 15°
`Right`	Yaw	Less than -15°
`Up`	Pitch	Greater than 15°
`Down`	Pitch	Less than -20°
`Palm_up`	Roll	Greater than 140° or less than -140°

Table 4.6: Hand poses and their valid ranges of angles (in degree). Any sample out of these ranges is considered wrong and was removed from the dataset.

After data cleaning, there were a total of 4396 training samples and 1928 test samples. The training set and test set distribution of each class are shown in Figure 4.6. It can be seen that although several cases have been removed, the dataset is quite balance among all the classes in both the training set and the test set.

4.2.2 Gesture Dataset

The gesture dataset was created to evaluate the performance of the proposed hand gesture recognition approach. This dataset contains 9 hand gesture classes that are recognized by our system (See more detail in Subsection 4.1.1), along with an additional negative class that represents no gesture at all. The negative samples were created to further evaluate the precision rate of our method.

Collecting Data

The gesture dataset was collected from 5 subjects, each generating 10 samples per gesture, 5 in left hand and 5 in right hand. For each sample, the subject performed a hand gesture and the system recorded the sequence of extracted skeletal data during that gesture. Exceptionally, for the two gestures `Rotate left` and `Rotate right`, which can only be performed in either left or right hand, only 8 samples were generated per subject. In total, there are 40 samples for the two gestures `Rotate left` and `Rotate right` and 50 samples for the other gestures.

To ensure the generalization of the dataset, gesture samples were performed with

varying lengths. Table 4.7 provides a statistical description on the length of samples in the dataset. The shortest recorded sample belong to gesture `Move right`, lasting for 0.8167 seconds. Conversely, the longest recorded sample belongs to gesture `Rotate left`, lasting for 6.3667 seconds. Regarding the length variance, `Rotate left` is the gesture with the most varying length, with the standard deviation of 1.0030. In terms of all gestures, the average length is 2.6022, the average median is 2.45 and the average standard deviation is 0.7816. From these numbers, it can be seen that there are some variances in length of all the samples.

Gesture	Min	Max	Mean	Median	Standard Deviation
`Move left`	1.5667	3.9000	2.5882	2.3833	0.6747
`Move right`	**0.8167**	4.4833	2.4760	2.3583	0.7083
`Move up`	1.3000	4.8167	2.4078	2.3333	0.7333
`Move down`	1.3000	4.4000	2.4243	2.3250	0.6825
`Rotate left`	1.8167	**6.3667**	2.9992	2.6583	**1.0030**
`Rotate right`	1.6167	4.6000	2.9254	2.7833	0.8068
`Close fist`	1.8500	4.6500	2.8623	2.8333	0.7327
`Stop`	0.8667	3.9833	2.2189	2.0833	0.6020
`Thumb in`	1.2500	4.4333	2.5717	2.5667	0.7166
All gesture	0.8167	6.3667	2.6022	2.4500	0.7816

Table 4.7: Statistics on length of each hand gesture (in seconds). For each hand gesture class, we evaluate the mean, median, minimum, maximum, and standard deviation among all samples.

Cleaning Data

Similar to the pose dataset, during data collection, there were some missing data and wrong cases due to errors made by the input device and the hand tracking service. The wrong cases were also due to the mistakes made by the subjects while generating data. For example, while generating the negative samples, some subjects accidentally perform some valid hand gestures. We double checked all of the recorded samples carefully removed all the wrong cases.

After data cleaning, there are a total of 438 samples belonging to 9 hand gesture classes and an additional negative class. The distribution of the gesture dataset is shown in Figure 4.7.

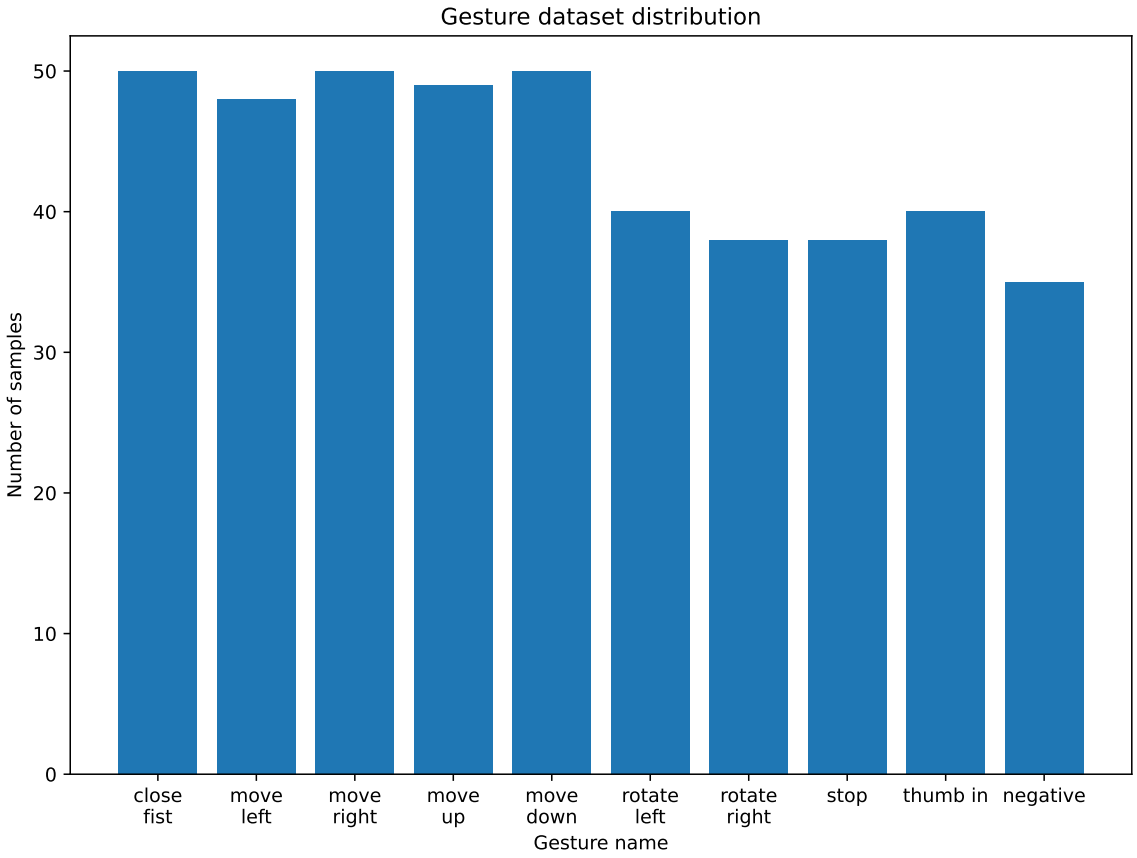


Figure 4.7: Distribution of the gesture dataset. There are 9 gesture classes along with a negative class that represents no gesture at all.

4.3 Gaming Application for Hand Rehabilitation

In this section, we introduce the gaming application that we developed for game-based rehabilitation. The application composed of 3 simple and interactive games:

- Shapes and Colors
- Eggs and Milk

- Dino Run

The gaming application is developed using Pygame version 2.3.0, a versatile Python module for cross-platform game development [94]. It incorporates the SDL[2] library version 2.24.2 and runs on Python version 3.11.3. The home screen interface of the application is depicted in Figure 4.8.

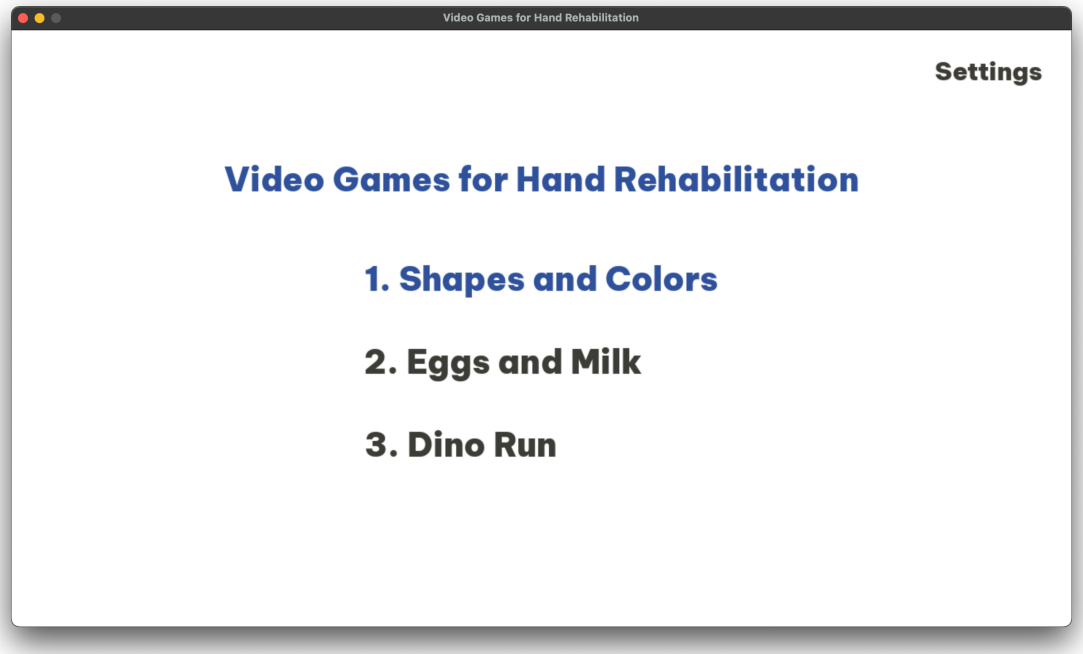


Figure 4.8: Home screen of the application. There are three interactive games that can be selected.

The home screen allows the player to select which game they wish to play among the three interactive games. Additionally, there is a `Settings` button where users can customize various options such as turning on background music or setting the language

[2]SDL stands for Simple DirectMedia Layer. It is a popular cross-platform multimedia library used for game development, providing low-level access to audio, keyboard, mouse, joystick, and graphics functionalities. SDL serves as a foundation for many game engines and frameworks, including Pygame.

of the system. The background music is included to enhance the user experiences while playing the games. Figure 4.9 displays the setting screen of our application.

Figure 4.9: Home setting screen. In this setting, we can modify music and language.

The application supports English and Vietnamese. The default language is English. To switch to Vietnamese language, the user can navigate to the `Settings` section and select the `Language` option. From there, user can switch to Vietnamese language and click `Save` to apply the changes. Figure 4.10 depicts the interface of the application in Vietnamese. Table 4.8 lists the hand gestures that the user needs to perform to send actions in the home screen.

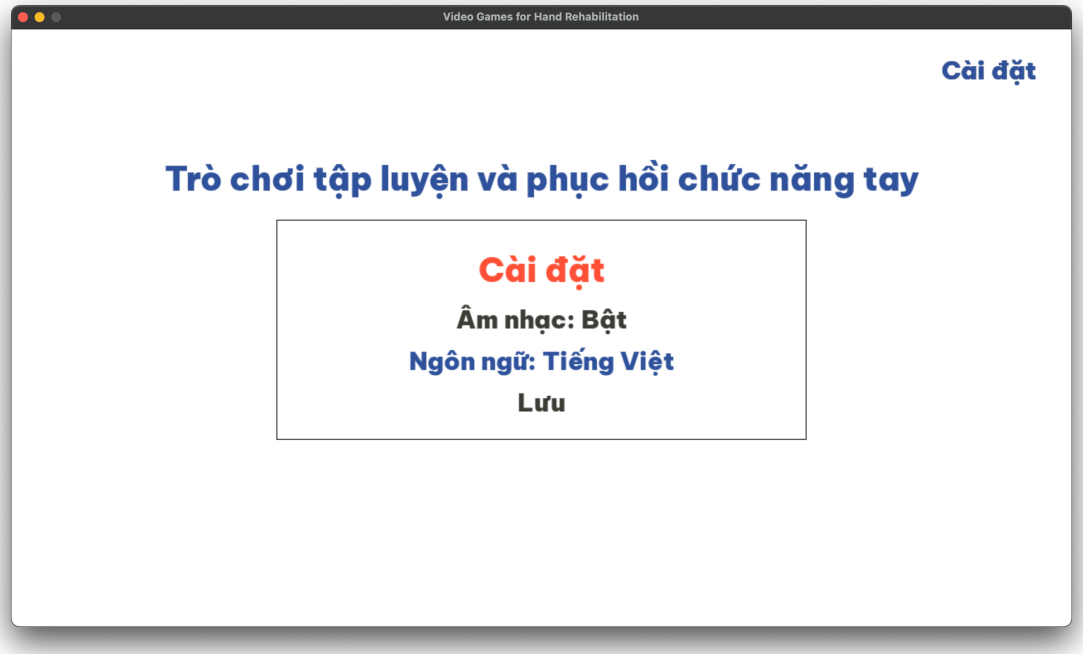


Figure 4.10: Vietnamese setting screen. This Vietnamese setting is developed for Vietnamese children and the elderly.

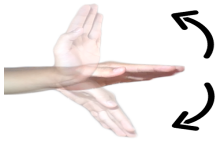
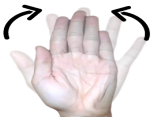
Gesture	Game Action
 Move left/right	Move the selection up/down.
 Close fist	Confirm the selection.
 Stop	Open the setting screen.

Table 4.8: List of hand gestures and corresponding actions in home screen.

4.3.1 Shapes and Colors

Shapes and Colors is an interactive game designed to aid in hand rehabilitation. The game features colorful shapes of varying sizes and colors that challenge the user to select the correct shape based on the on-screen description. Players will need to user their hands to move the selection box to the correct position containing the correct shape on the screen. The game gradually increases in difficulty as players progress, challenging them to sort shapes at a faster pace and with greater accuracy while the number of shapes and colors increases. With its engaging gameplay and therapeutic benefits, we believe that Shapes and Colors is an excellent tool for hand rehabilitation.

The game mechanics are straightforward, requiring players to select the shape with the appropriate color based on the on-screen descriptions provided during each turn. Following each turn, the associated shapes, colors, and descriptions are randomly reset to ensure a varied gameplay experience. This system encourages players to continuously adapt their hand-eye coordination and dexterity skills in order to perform well, while also promoting cognitive flexibility.

The player can adjust their selection by horizontally moving their palm to the left or right, which will shift the selection box correspondingly. To confirm their choice, they can make a fist. That offers a straightforward and user-friendly control system suitable for individuals with a variety of hand mobility limitations.

To enhance the enjoyment and engagement of players, we implemented a tiered level system ranging from easy to medium, culminating in a challenging hard level. In the easy level, players are introduced to three basic colors: red, green, and blue (the fundamental RGB colors), as well as three basic shapes: circle, square, and triangle.

Figure 4.12 depicts the gameplay at the easy level, wherein players are required to select one shape out of three that appear on the screen. The correct shape is indicated in the text situated above the shapes, providing players with clear instructions.

Figure 4.11: Help pop-up displayed on Shapes and Colors screen. In this game, we have to select the shape with the colour mentioned on the top of the screen.

The easy level is initiated when the player scores between 0 to 7 points. Upon successfully completing the 7^{th} round, players are then advanced to the intermediate (or medium) level, which presents a greater challenge. The medium level introduces additional challenges by incorporating an oval shape and two new colors, white and black. Moreover, players are presented with four options, in contrast to the three options in the easy level. Each round at this level consists of a random assortment of four shapes, including circles, squares, triangles, and ovals, accompanied by a total of five colors: red, green, blue, white, and black.

The medium level is unlocked when a player's score falls between 8 and 15 points. Figure 4.13 is an example of a medium level game. Upon successfully completing the 15^{th} round, players are then advanced to the final and most challenging level, i.e., the hard level. The hard level is the ultimate test of skill and endurance in this

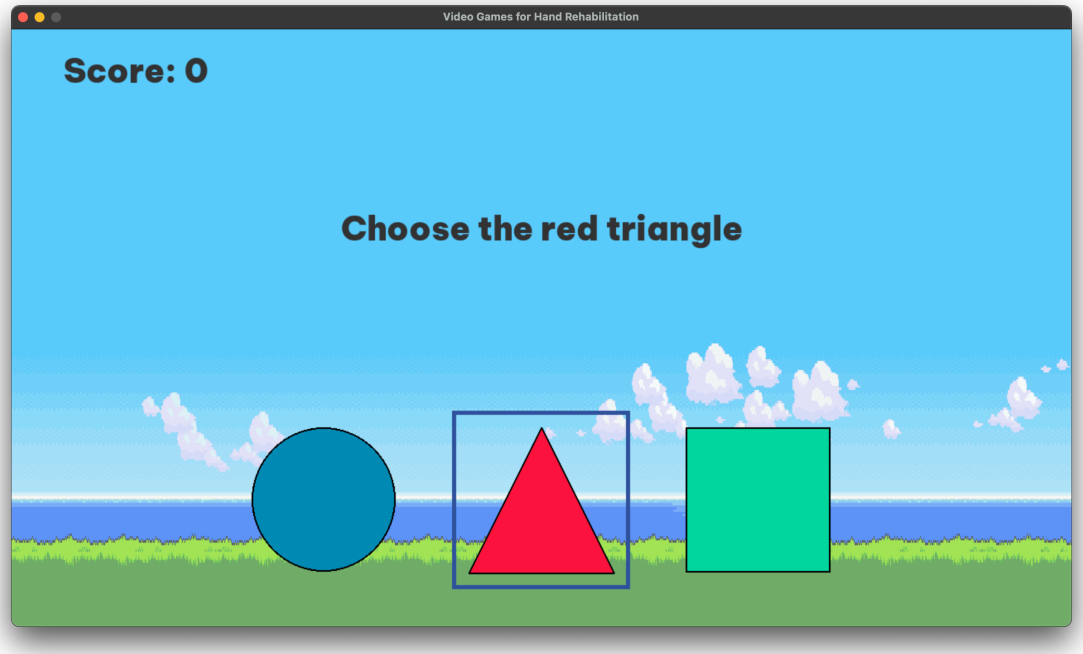


Figure 4.12: Easy level in Shapes and Colors. At this level, we have 3 options to select with a small range of colours.

game. Players can access this level only after successfully passing the first two levels. Once unlocked, players can play this level endlessly. The hard level includes several enhancements, such as an increase in the number of options to choose from during each round, from four to five. The new options include a star and diamond shape, in addition to the four shapes already present - circle, square, triangle, and oval. To add to the challenge, two new colors - purple and orange - have been added to the existing list of colors, which includes red, green, blue, white, and black. This brings the total number of shapes to six and the total number of colors to seven. Each round at this level presents a randomized selection of five shapes, each with a unique color. Players have only one opportunity to make the correct selection. Figure 4.14 exemplifies a hard game level.

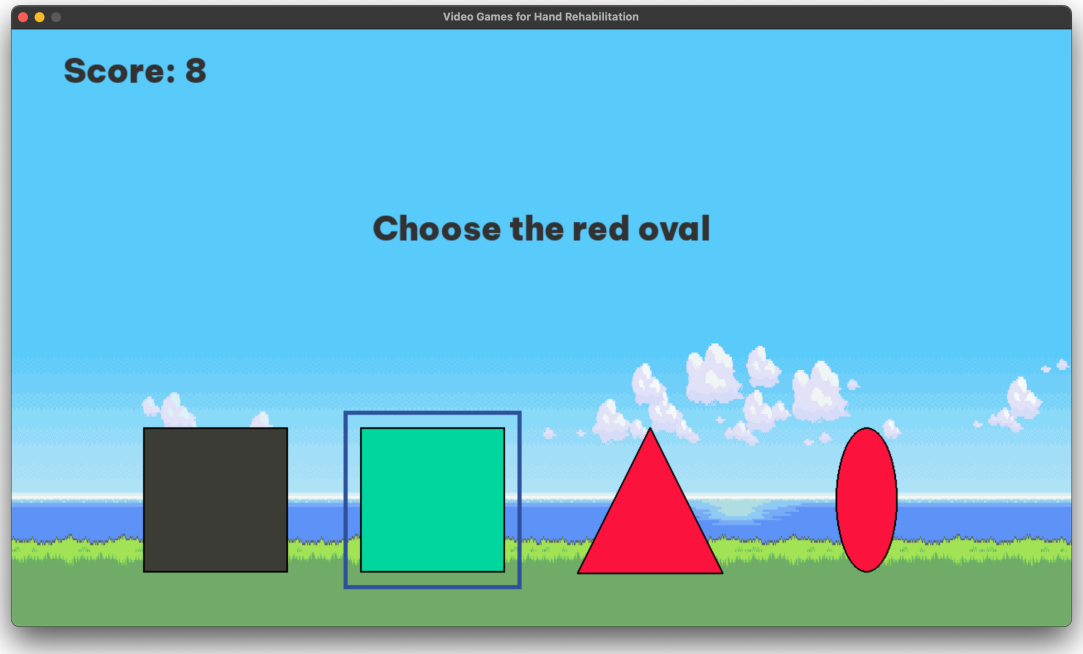


Figure 4.13: Medium level in Shapes and Colors. At this level, we have 4 options to select with a medium range of colours.

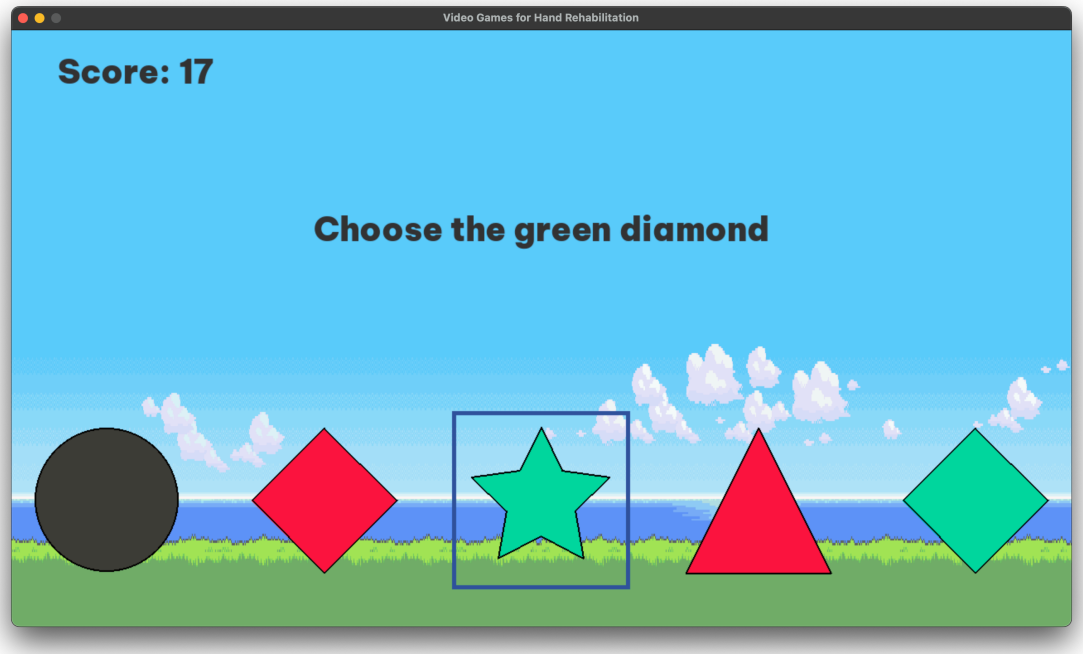


Figure 4.14: Hard level in Shapes and Colors. At this level, we have 5 options to select with a wide range of colours and shapes.

If the player selects an incorrect shape at any round that does not match the description, the game immediately ends. The end screen, as shown in Figure 4.15, offers players two options. The first option is `Replay`, which allows them to start the game from the beginning again. The score and difficulty level will be reset to their original values of zero and easy, respectively. The second option is `Home`, which brings players back to the home screen interface, refer to Figure 4.8.

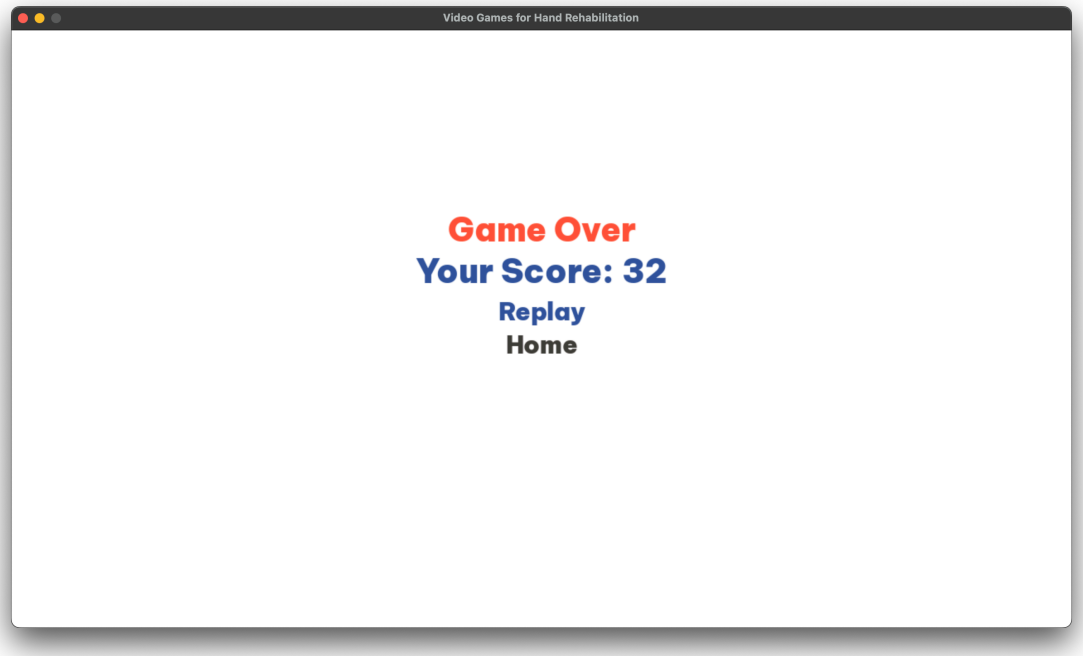


Figure 4.15: End screen in Shapes and Colors.

Additionally, we have designed a feature that allows players to pause the game if they need to. Once paused, players have two options to choose from. The first option is `Continue`, which resumes the game from where it was stopped. The second option is `Home`, which brings the player back to the home screen. To navigate between these options, players must use hand gestures. Moving their hand up and down changes the selected option, and making a fist confirms their decision. Figure 4.16 shows the paused game screen.

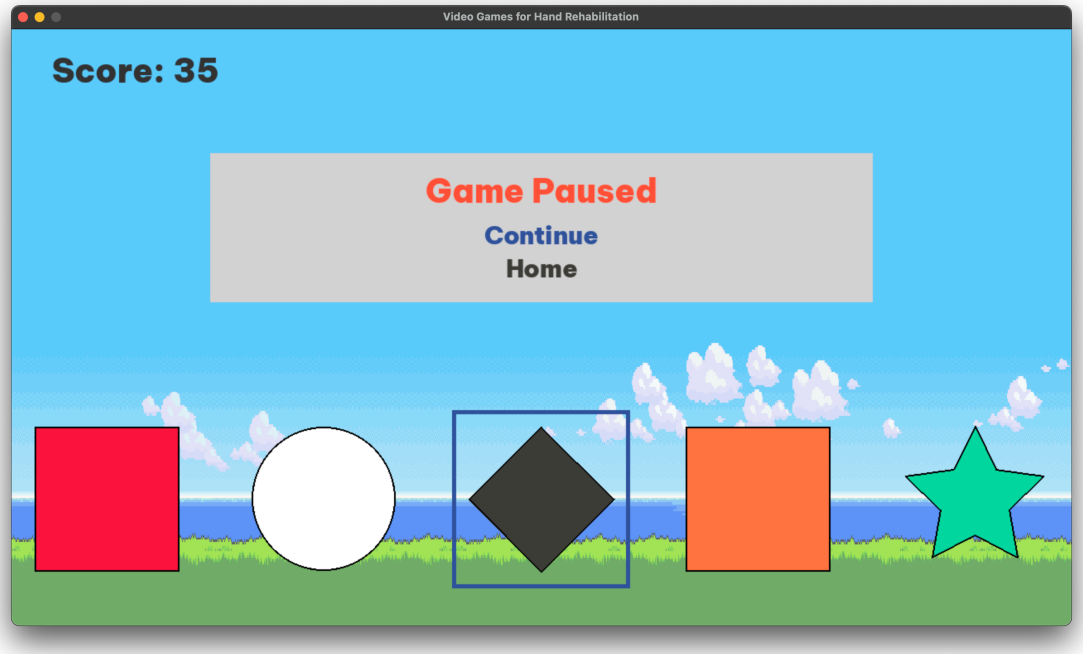


Figure 4.16: Paused pop-up in Shapes and Colors. At this state, we could go back to the Home screen or continue to play.

The Shapes and Colors game is an enjoyable and captivating means of assessing and enhancing a player's proficiency in identifying and differentiating various shapes and colors. This game not only serves as a form of entertainment but also provides valuable cognitive benefits. The game's mechanics require the player to make selections by moving their hand left or right, providing a beneficial exercise for their wrist and aiding in their recovery. This feature enhances the game's therapeutic potential, making it a valuable tool for rehabilitation purposes. Table 4.9 lists the hand gestures and corresponding actions in Shapes and Colors.

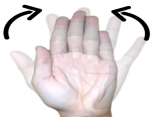
Gesture	Game Action
 Move left/right	Move the selection box to left/right.
 Close fist	Confirm the selection.
 Stop	Pause the game.
 Thumb in	Open the help pop-up.

Table 4.9: List of hand gestures and corresponding actions in Shapes and Colors.

4.3.2 Eggs and Milk

The Egg and Milk Game is a rehabilitation tool intended to improve the hand function of patients. The game is designed to challenge the player's ability to accurately collect eggs and milk, requiring them to collect each item in its corresponding container.

The rules of the game entail the acquisition of eggs by means of baskets and milk

bottles with milk containers, whereby the player must collect the descending eggs or milk bottles by manoeuvring the corresponding container to the designated drop zone. The game is designed to enhance hand-eye coordination and reaction time by utilizing physical movements to regulate the containers. These actions may also serve as a mode of physical exercise and recuperation for individuals with hand and wrist ailments. The primary objective of the game is to offer an enjoyable and interactive platform for physical therapy and the advancement of general motor skills.

The gameplay mechanics of this game are simple: the player's objective is to collect eggs using baskets and milk bottles with milk containers. To do so, the player must manually switch between the basket and milk container as objects randomly fall from the top of the screen to the bottom. It is crucial to note that the player must match the item with the corresponding container, as failing to do so will result in the game's termination. Additionally, allowing an item to drop to the bottom of the screen will also end the game. Players can read instructions on how to play as shown in Figure 4.17.

Similar to the Shapes and Colors game, we incorporated a level system in our design of Eggs and Milk. The level progression is determined by the player's score and comprises of three difficulty levels: easy, medium, and hard. The easy level aims to familiarize players with the game mechanics and the fundamental rules. In this level, the game item, which is represented by an egg, drops solely in the three designated squares located at the center of the screen, as illustrated in Figure 4.18. Moreover, the item type is limited to eggs only, and the falling speed is relatively slow to enable players to acclimate to the game's rhythm. Note that at the easy level, the item that appears randomly only encapsulates the egg type.

As depicted in Figure 4.18, the default game item for the easy level is the egg, which appears randomly at three fixed positions on the screen in each round. Players will commence the game at the easy level until their score reaches a threshold of 8. Upon reaching this score, they will advance to the medium level.

Figure 4.17: Help pop-up displayed on Eggs and Milk screen. In this game, we move the basket to catch the falling eggs.

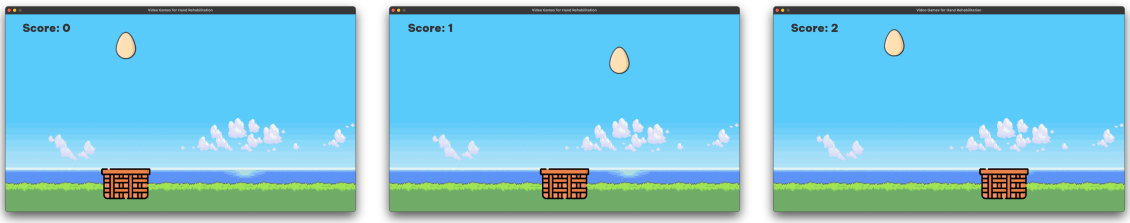


Figure 4.18: Easy level in Eggs and Milk. At this level, there are three discrete positions in which the user could move the basket.

The medium level is akin to the easy level, but with a slight modification - the introduction of the milk bottle item. This addition requires players to pay close attention to the item that appears on the screen and choose the corresponding container accordingly. It is essential to note that the egg basket cannot catch the milk bottle

and vice versa, thus maintaining the game's rules. Therefore, the items that appear on the screen will now be selected randomly between the egg and milk bottle types.

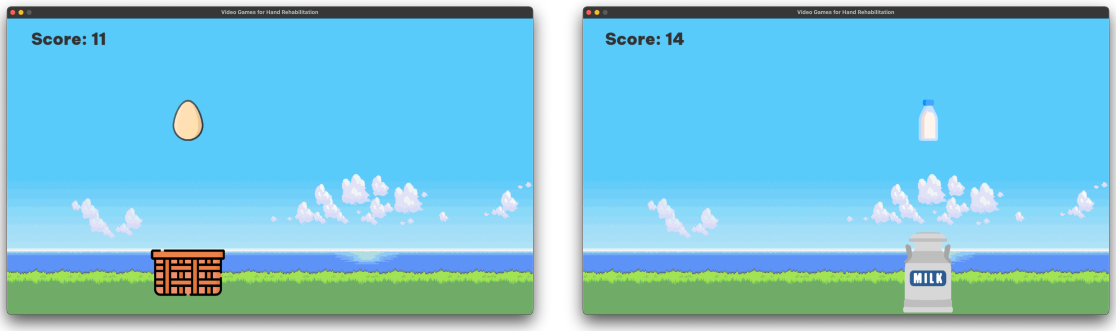


Figure 4.19: Medium level in Eggs and Milk. At this level, there are three discrete positions in which the user could move the basket, and the user can change to the milk container to catch the glass of milk.

Players will encounter a higher level of challenge when playing at the medium level until they reach a score of 16. Once the player reaches this score threshold, the game will automatically elevate the level of hard. At this stage, the game unlocks two additional area tiles situated on the left and right edges of the screen, as depicted in Figure 4.20. Moreover, to intensify the gameplay experience, the game's drop rate of items is slightly increased to elevate the game's speed. This adjustment is crucial in stimulating the player's competitive spirit and enhancing their overall engagement. It is important to note that the game will continue to stretch its level of difficulty to infinity, thereby providing players with an endless and challenging gameplay experience.

As previously mentioned, the game over scenario occurs when the item drops to the bottom of the screen. This can happen due to two primary reasons - the player's inability to move the catcher quickly enough or the use of the wrong catcher to catch the dropped item. Upon reaching a game over situation, the game will display the

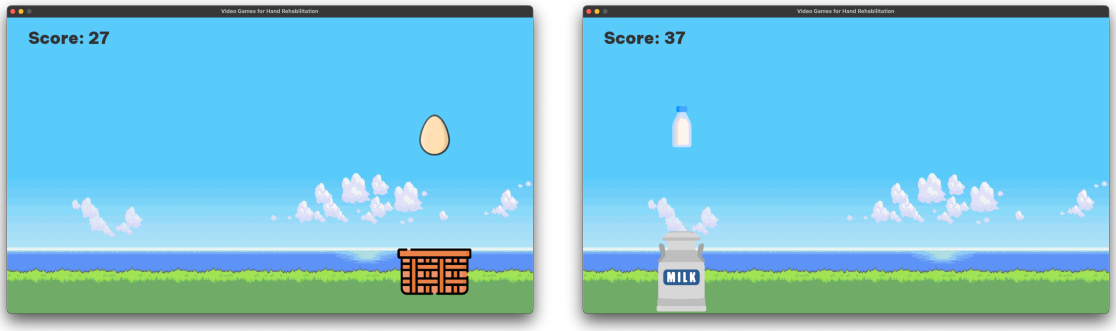


Figure 4.20: Hard level in Eggs and Milk. At this level, there are five discrete positions in which the user could move the basket, and the user can change to the milk container to catch the glass of milk.

end screen, similar to Figure 4.21. At this point, players are presented with two options - `Replay` and `Home`, which serve the same functions as in the Shapes and Colors game. The `Replay` option enables players to restart the game, while the `Home` option redirects them to the main screen, refer to Figure 4.8.

As the Eggs and Milk game is a continuous and timed game, it is essential to provide players with the option to pause the game when necessary. To this end, we have incorporated a stop feature in the game, allowing players to pause the game as needed. This feature comprises two options - `Continue`, which enables players to resume playing the game, and `Home`, which redirects them to the home interface screen. Figure 4.22 shows an example using the stop game function in the game Eggs and Milk.

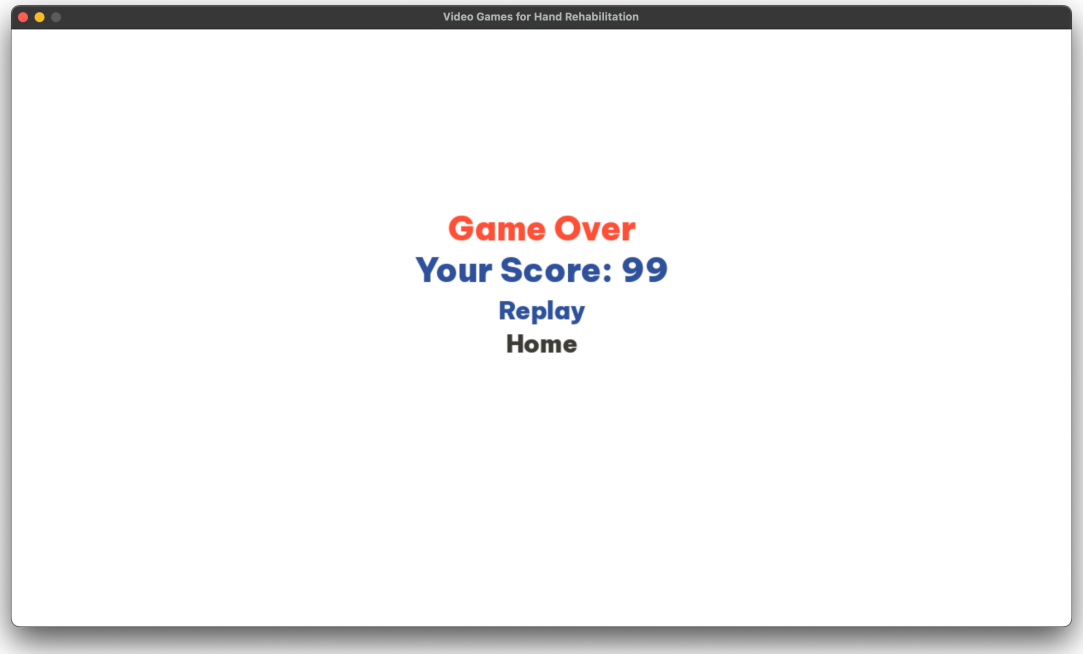


Figure 4.21: End screen in Eggs and Milk. At this state, we can go back to the Home screen or play again.

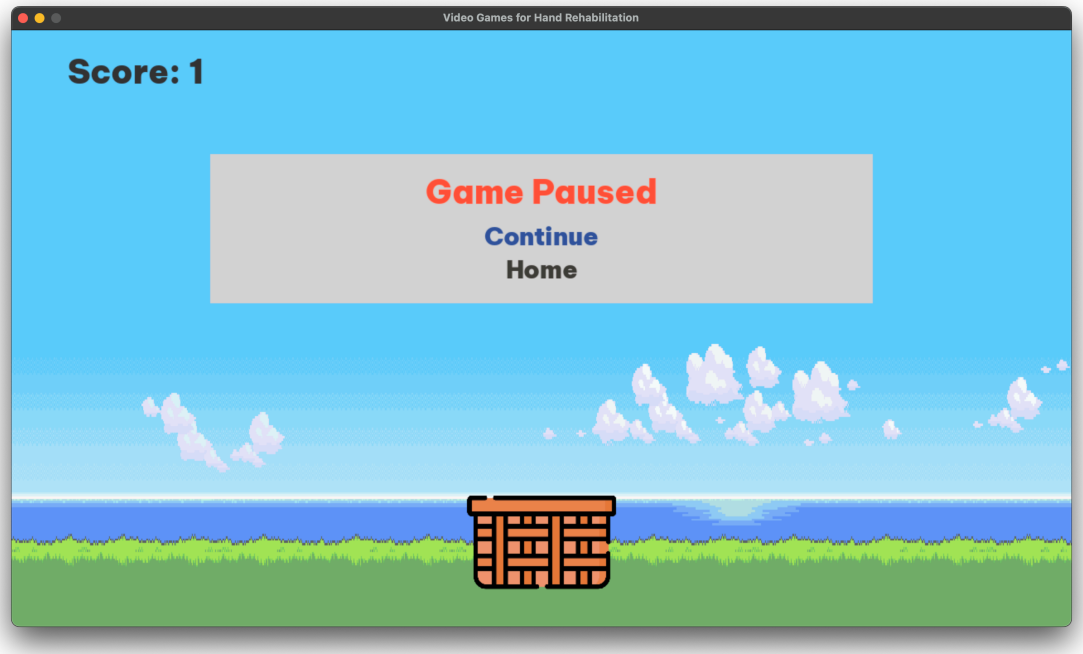


Figure 4.22: Paused pop-up in Eggs and Milk. At this state, we can go back to the Home screen or continue playing.

Overall, Eggs and Milk is an engaging game that challenges players to catch falling items in corresponding containers. The game has a level-based structure, with each level presenting unique challenges and introducing new gameplay mechanics. The game also includes a stop feature, allowing players to pause the game as needed. With an endless and challenging gameplay experience, Eggs and Milk is a fun and addictive game for players of all ages. Table 4.10 lists the hand gestures and corresponding actions in Eggs and Milk.

Gesture	Game Action
 Move left/right	Move the container to left/right.
 Rotate left/right	Change the container type.
 Stop	Pause the game.
 Thumb in	Open the help pop-up.

Table 4.10: List of hand gestures and corresponding actions in Eggs and Milk.

4.3.3 Dino Run

Dino Run is an exciting new side-scrolling platformer game that puts players in control of a lovable dinosaur on a quest to outrun extinction. With simple yet addictive gameplay and retro-inspired graphics, Dino Run is the perfect game for players of all ages and skill levels. We developed Dino Run based on the idea of Dinosaur Game, a classic game from Google that runs on Google Chrome browser. We developed Dino Run based on the idea of The Dinosaur Game, a well-known classic game from Google that runs on Google Chrome browser [95].

The objective of the game is to control a pixelated dinosaur that runs through a desert landscape, jumping over cacti and dodging obstacles such as pterodactyls. The game is endless, meaning that the player can continue to play until they make a mistake and the dinosaur collides with an obstacle.

The game's mechanics are based on simple principles of physics and timing. In the traditional version, players control the dinosaur's movement using the space bar to jump, with the length of the jump depending on how long the space bar is held down. However, the game mechanics have been altered by us to enable the control of the dinosaur via hand gestures, as opposed to the conventional utilization of the space-bar. The game's difficulty increases as the player progresses, with obstacles appearing at higher speeds and frequencies. Figure 4.23 instructs the player how to use hand gestures to play this game.

Dino Run game will have two main control ways: jump and duck. Figure 4.24 depicts these two movements. Figure 4.24a illustrates the dinosaur's upward jump, triggered by the player's execution of a palm-up gesture. In contrast, Figure 4.24b portrays the dinosaur's ducking, initiated by the player's palm-down gesture movement.

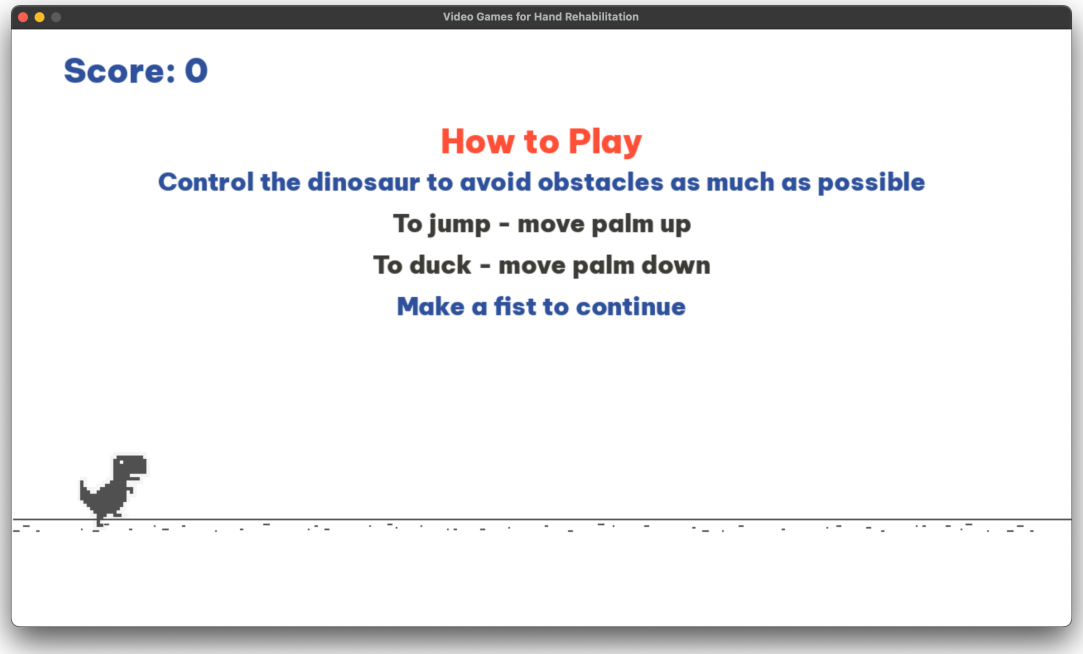
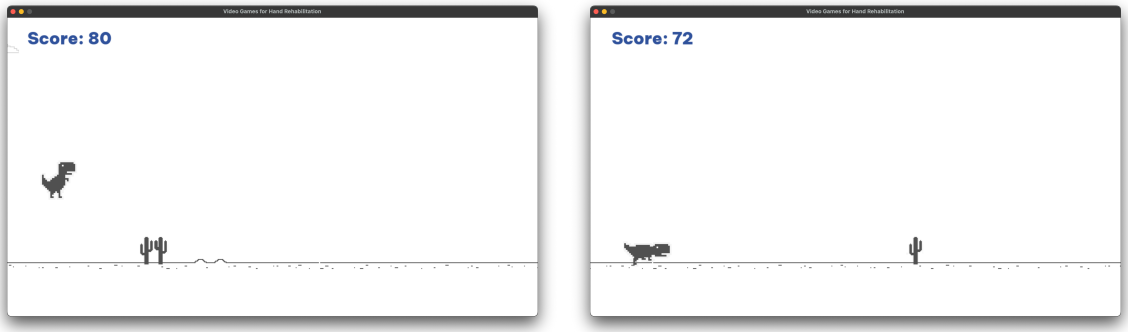


Figure 4.23: Help pop-up in Dino Run. In this game, we have to control the dinosaur to jump and bend properly to avoid obstacles.



(a) Jumping movement of the dinosaur. **(b)** Ducking movement of the dinosaur.

Figure 4.24: Two movements in the game Dino Run. The dinosaur could either run or jump.

It should be emphasized that the current version of the game diverges from the conventional one in terms of the dinosaur's crouching functionality being linked to

the duration of the player's pressing action. Notably, we have introduced a distinct temporal parameter, whereby the dinosaur remains in a crouched position for a fixed duration, irrespective of the player's behaviour. Specifically, each time the dinosaur engages in a crouching maneuver, it will maintain the crouched posture for a fixed interval of 2500 milliseconds or 2.5 seconds.

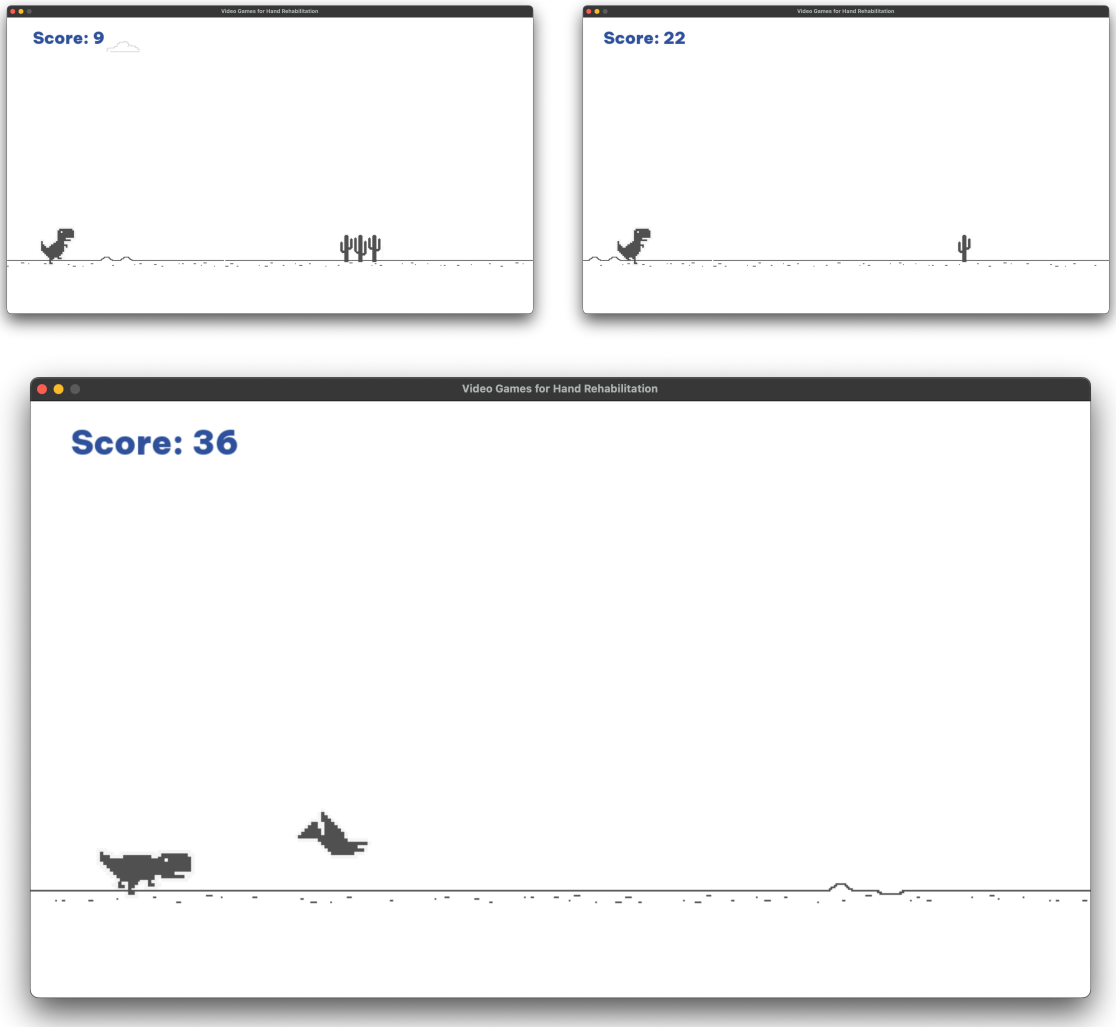


Figure 4.25: An example played in the game Dino Run.

As depicted in Figure 4.25, the game features several obstacles that players must

overcome, including large and small cacti, as well as birds. The cacti are fixed and permanently positioned on the ground, while the birds are randomly spawned from the ground up to an appropriate height. It is worth noting that the obstacles in the game are not only randomly spawned in terms of position but also in terms of their size.

Similar to other games in our application, the Dino Run game features a level system. However, unlike those games, the challenge system of Dino Run game lies in the speed of events in the game. As players progress through the game, they are met with increasingly higher game speeds that intensify as they cover more distance. This dynamic feature serves to heighten player engagement and excitement while simultaneously fostering and maintaining key cognitive skills, such as mental focus, observation ability, and hand-eye coordination.

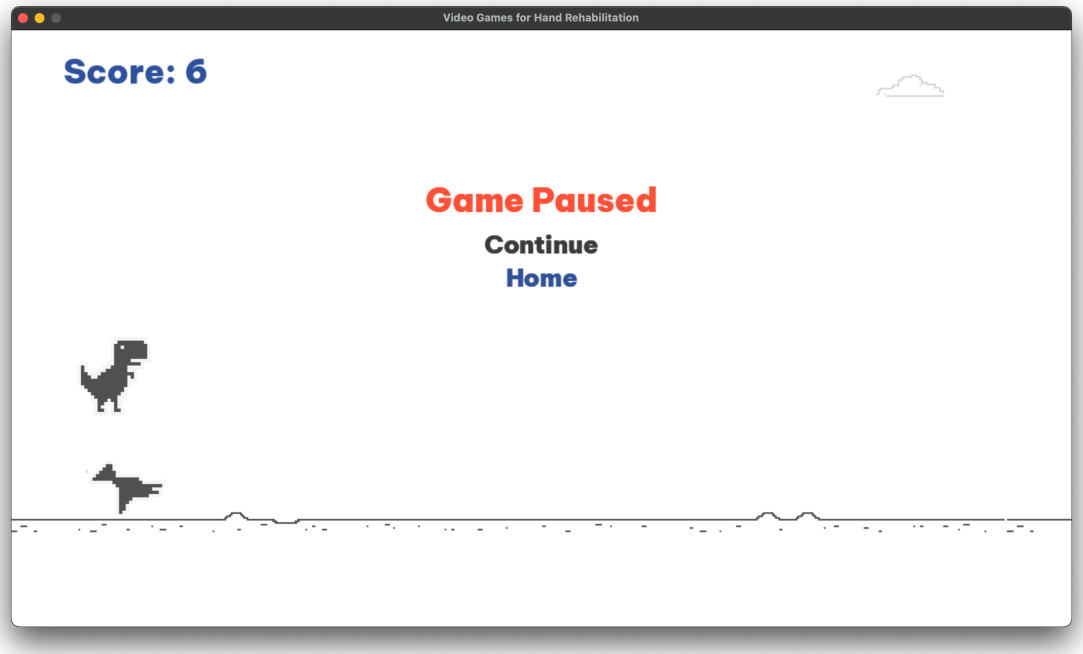


Figure 4.26: Paused pop-up in Dino Run.

Besides, the bonus speed is also subject to a proportional increase alongside the game speed. However, in order to ensure that the game remains playable and engaging for all players, we have established a limit on both the game speed and bonus speed, capping them at 100 points. Once this threshold is reached, the game speed and bonus speed will no longer increase, but remain fixed and constant. This design decision is intended to maintain the game's enjoyability and provide players with an endless source of entertainment, depending on their skill level and perseverance.

Given that this game emphasizes time as a key factor, we have incorporated a pause feature that allows players to halt gameplay at any time, based on their preferences. In addition, we have provided two options: `Continue` - which permits players to resume their endless journey, or `Home` - which enables them to exit the game and return to the main screen. The pause feature is illustrated more explicitly in Figure 4.26. In the event that the dinosaur collides with an obstacle and is unable to continue, the game will conclude, and the end screen will be displayed, depicting the player's final score. This feature is illustrated in greater detail in Figure 4.27.

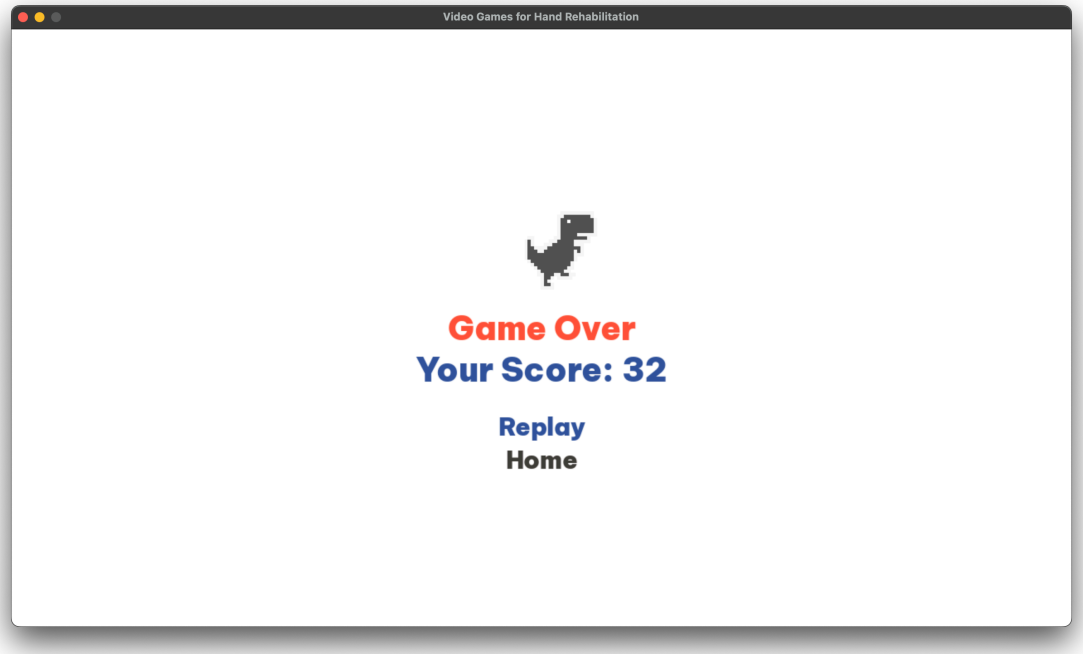


Figure 4.27: End screen in Dino Run.

At this point, players will have two options: `Replay` - which allows them to start the game anew, or `Home` - which enables them to exit the game and return to the home interface. These options are similar to those presented in other sections of the application, providing a consistent user experience throughout.

To sum up, Dino Run is an example of a simple yet entertaining game designed to test a player's hand-eye coordination and reaction time. The game features a dinosaur that must jump over obstacles such as cacti and birds while running endlessly forward. To increase the challenge, the game progressively increases in speed the further the player progresses. The game's interface provides options for pausing and resuming the game, as well as replaying or exiting the game upon completion. The game demonstrates how even simple games can be engaging and challenging, providing a fun and entertaining experience for players. This provides a positive and enjoyable experience for the patient when participating in hand physical therapy.

Table 4.11 lists the hand gestures and corresponding actions in Dino Run.

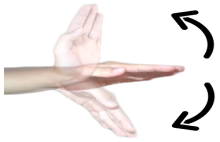
Gesture	Game Action
 Move up/Down	Make the dinosaur to jump/duck.
 Stop	Pause the game.
 Thumb in	Open the help pop-up.

Table 4.11: List of hand gestures and corresponding actions in Dino Run.

4.3.4 Software Development Process

To develop the gaming application, we followed some of the main stages in software development.

Requirement Analysis

In this stage, we evaluated the requirements, including the functional and the non-functional requirements of the application. As our goal is a simple and interactive gaming application that supports hand rehabilitation, some functional requirements need to be met, such as allowing the user to play the games using their hand gestures, allowing them to adjust some settings in the application. Non-functional requirements, such as the usability, and the responsiveness of the application is also

required. Table 4.12 and Table 4.13 shows the functional and the non-functional requirements of our gaming application, respectively.

No.	Name	Description
1	Game Play	Allow the user to use hand gestures to interact with the game environment.
2	Game Selection	Allow the user to choose any of the three games to play.
3	Pause and Resume	Allow the user to pause the game and resume it later from the same point.
4	Help pop-up	Allow the user to display the help pop-up before and during the game play.
5	Language Setting	Allow the user to switch between English and Vietnamese.
6	Score Tracking	The system should track the score of the player in each game.
7	Game Over	When the user loses, the game should display the final score and provide an option to restart the game.
8	Difficulty Level	The difficulty level of each game will increase when the user achieves higher scores.

Table 4.12: Functional requirements of our gaming application.

No.	Name	Description
1	Performance	The game should be responsive and run smoothly without any noticeable lag or delays.
2	Usability	The game should have a simple and intuitive interface, allowing users of all ages to understand and play the game easily.
3	Reliability	The game should be stable and free from crashes or unexpected errors during gameplay.
4	Responsiveness	The game should respond quickly to user input, providing immediate feedback to the player's actions.
5	Localization	The game should support multiple languages and cultural adaptations to cater to a global audience.

Table 4.13: Non-functional requirements of our gaming application.

Design

In this stage, we designed the application from the requirements specified in the previous stage. As part of the design process, we created UML (Unified Modeling Language) diagrams to represent the structure and behavior of the system. Figure 4.28, Figure 4.29, Figure 4.30, and Figure 4.31 show the UML use case diagrams of the home screen and the three interactive games, respectively. These diagrams illustrate the various user interactions and functionalities associated with each component of the application. The use case diagrams provide a high-level overview of how users will interact with the system and help in identifying the key features and requirements of the application.

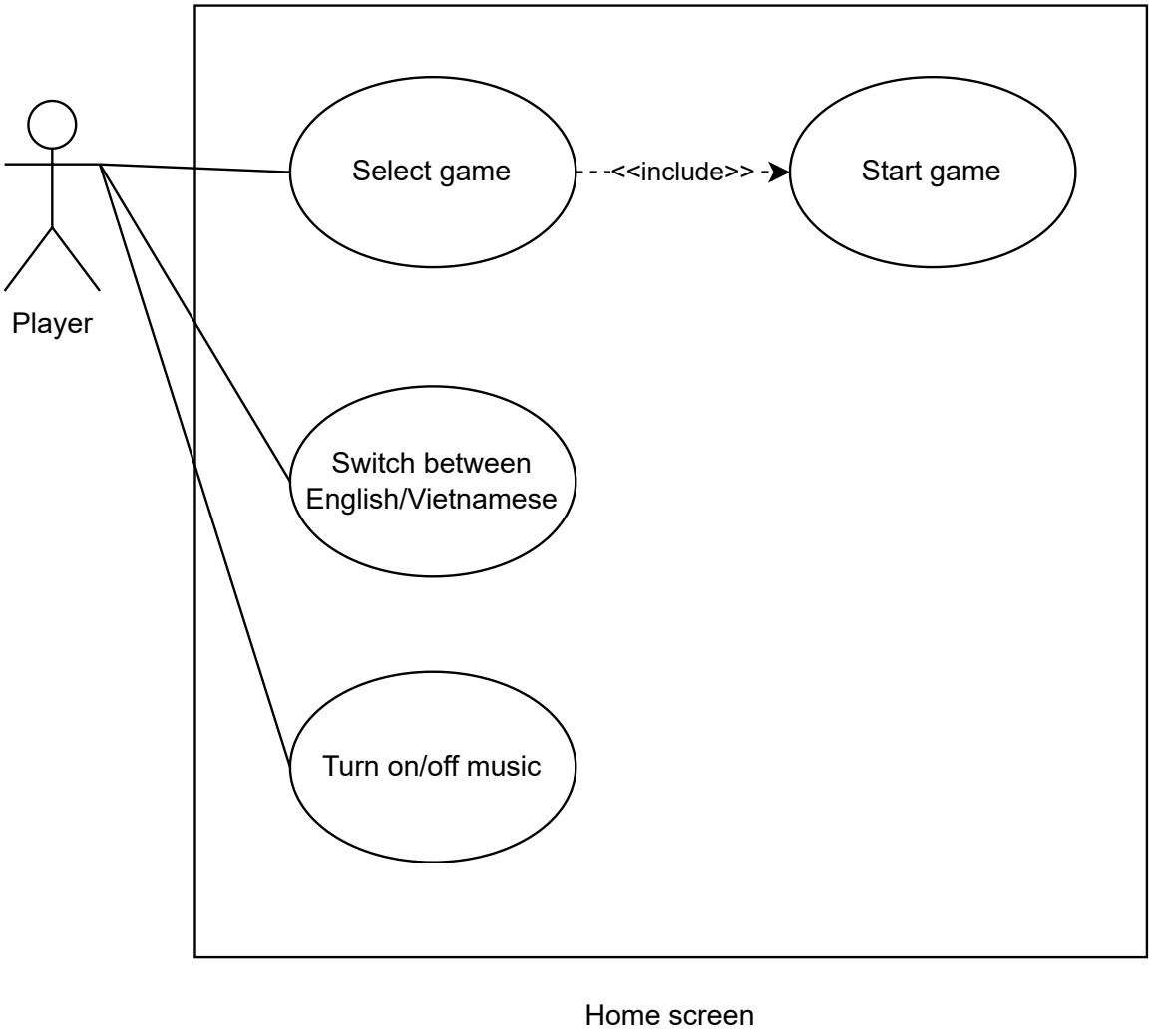


Figure 4.28: Use case diagram of the home screen.

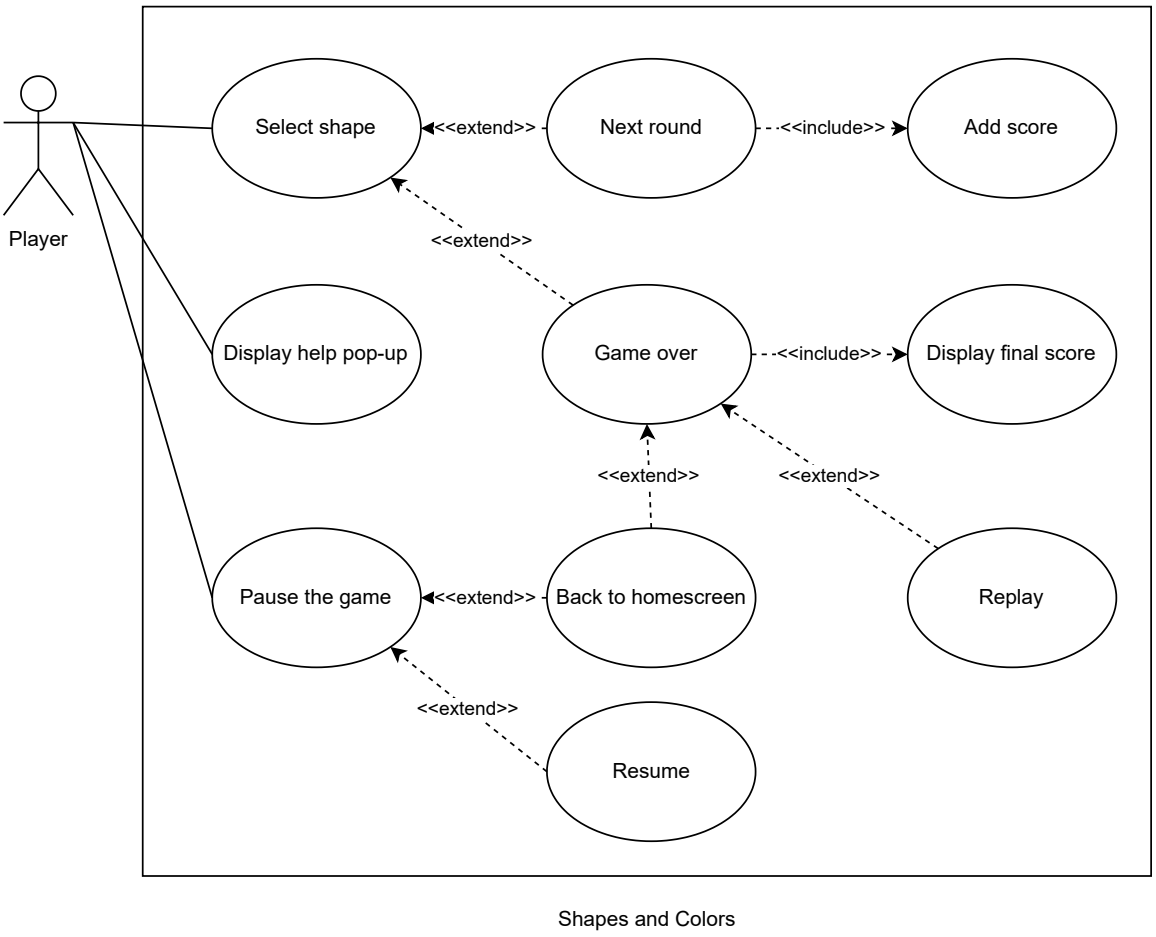


Figure 4.29: Use case diagram of the game Shapes and Colors.

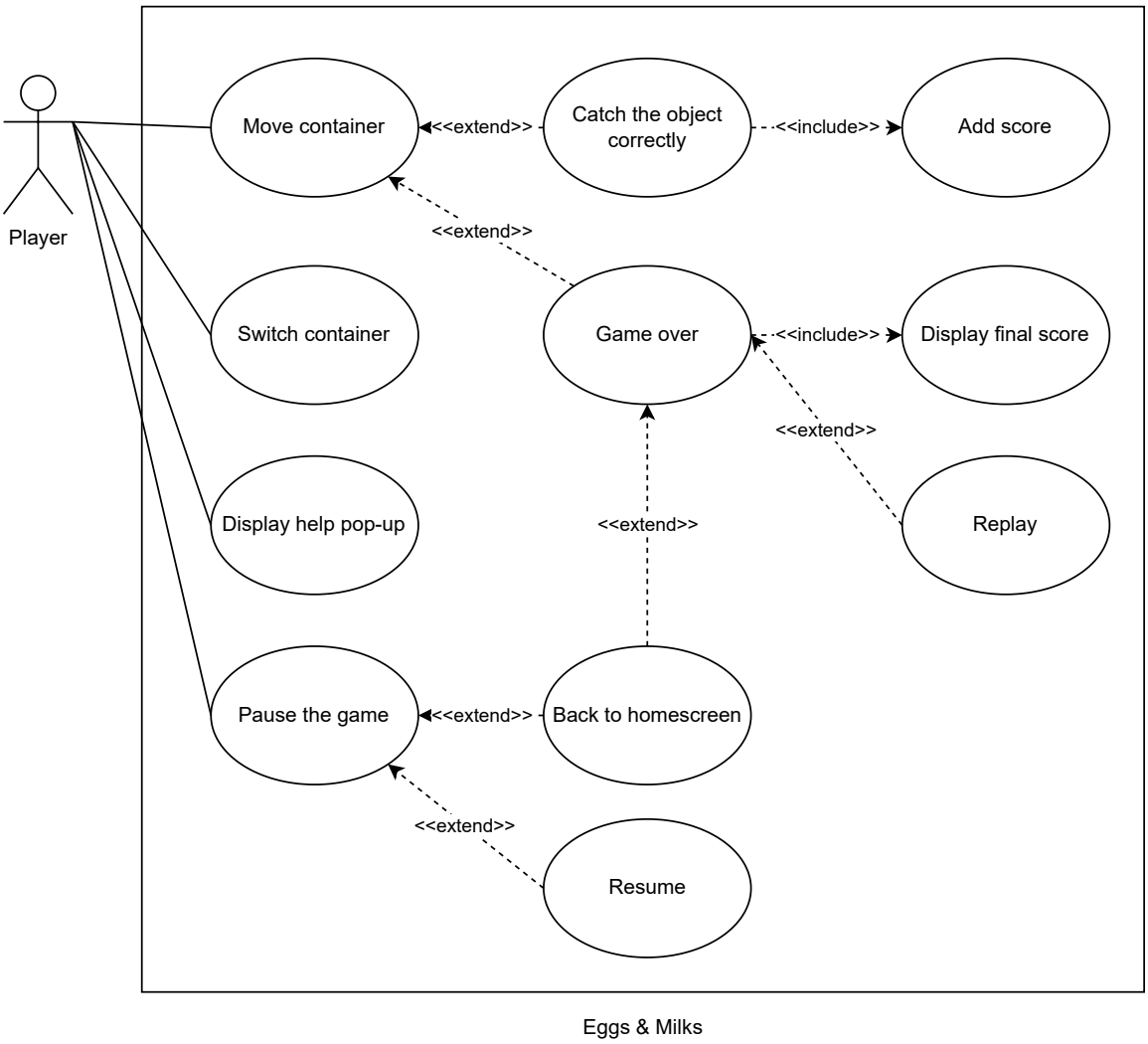


Figure 4.30: Use case diagram of the game Eggs and Milk.

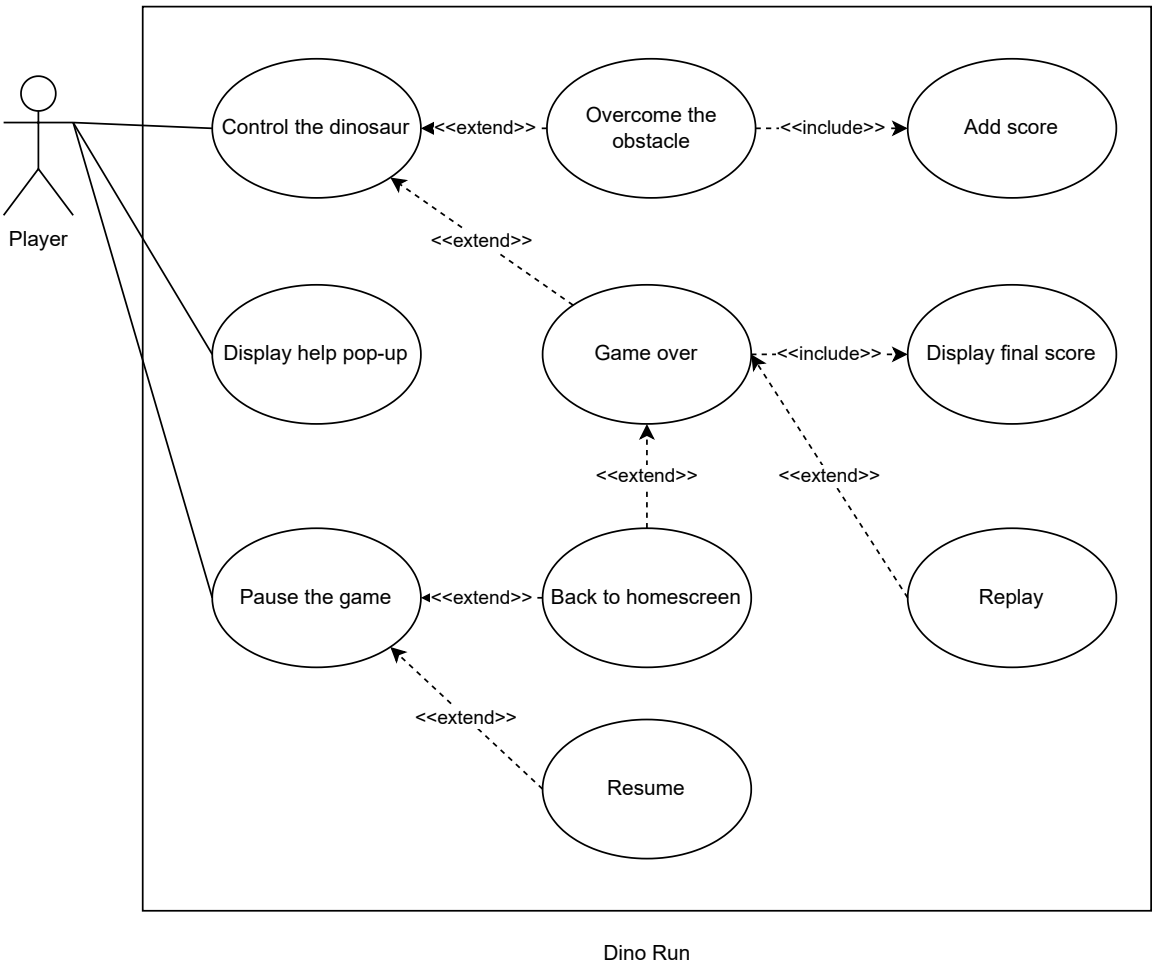


Figure 4.31: Use case diagram of the game Dino Run.

Figure 4.32 shows the class diagram of the application. This diagram is extremely important and acts as a blueprint to implement the application. We define several classes to implement the application. The `GameHandler` is the main entry point that manages the state of the entire application and manages the general setting of the app. There are classes that implement several screens of the application, including `HomeScreen`, `ShapesAndColors`, `EggsAndMilk`, `DinoRun`, `GameOver`. These classes keep track of the state of each window, for example, managing the state of a game, and draw out the user interace on the screen. All of the classes communicate and are connected to each other to create the overall gaming application.

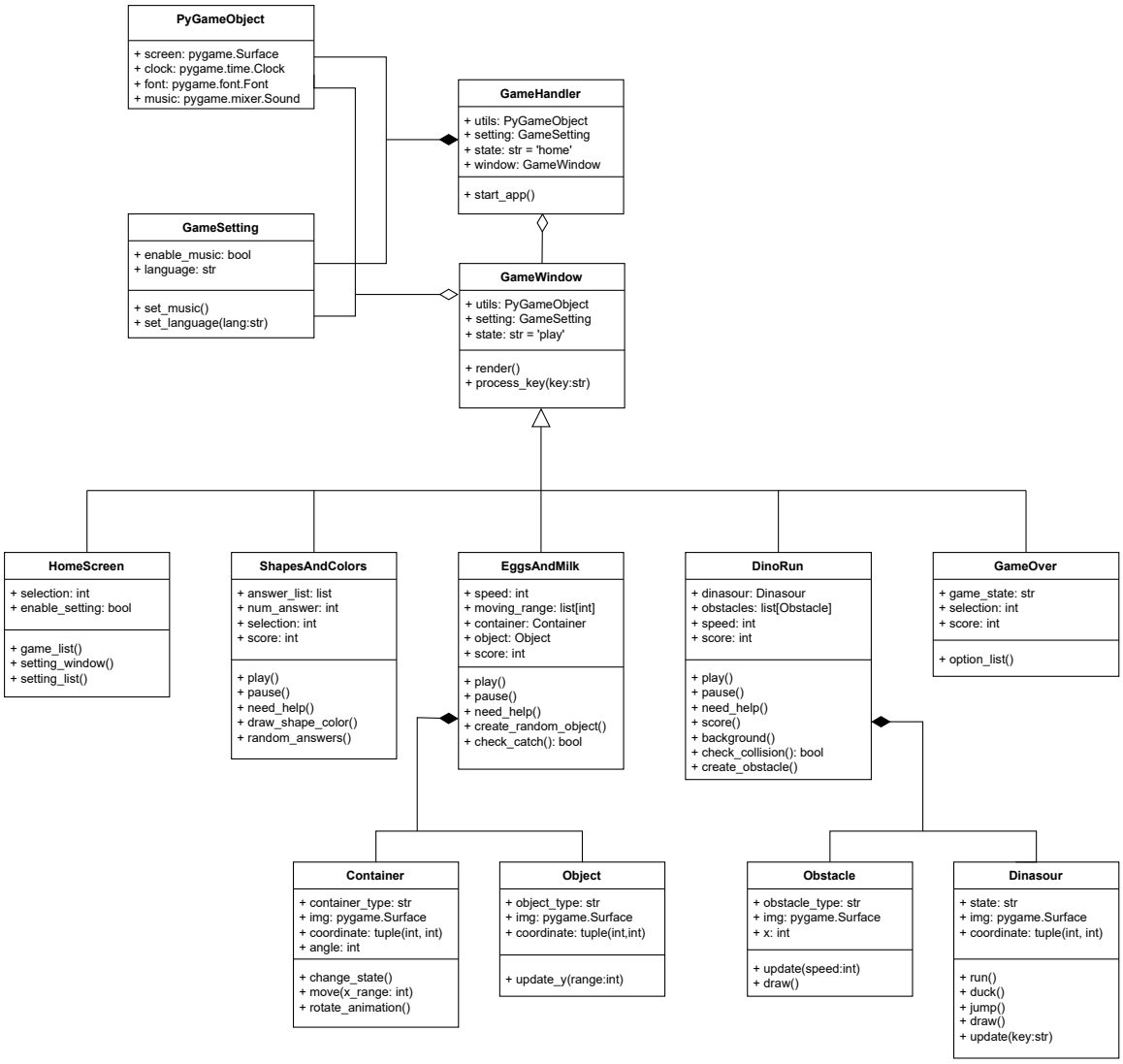


Figure 4.32: Class diagram of our gaming application.

Coding and Implementation

This stage implements the gaming application from the design in the previous stage. In our work, we adopted Python programming language (version 3.9.0) and PyGame [94] (version 2.3.0) to develop our gaming application. Python is a multi-functional programming language that offers several packages and libraries to achieve many different tasks, including game development. PyGame is a cross-platform set of Python

modules designed for writing video games. From the design of the gaming application in the previous stage, the actual application is implemented in this stage. By leveraging the power of Python and PyGame, we bring the envisioned gaming experience to fruition.

Testing

After implementing the application, we created several test scenarios to test the gaming applications. The scenarios were created based on the functional, non-function requirements and the use cases of the application. In overall, we created 15 test scenarios to test the three games, and some other functionalities of our application. More details on the testing process is discussed in Section 5.2.

Application Packaging

The final step of game development is to wrap everything into a single package in which the user can simply launch the application with a single click. Currently, because of the limited support of Leap Motion tracking service on other operating systems, our system only supports Windows. Therefore, the gaming application was bundled into a single package where the user can click a .exe file to launch the entire application.

We used PyInstaller to bundle the gaming application and all of its dependencies into a single package [96]. As there are several dependencies and media files needed for the application, `-onedir` is the most suitable option that provides the fastest time for launching up. After running PyInstaller, our application was bundled into a single directory, and a single .exe was generated for the user to launch the application with just a single click.

Chapter 5

Results

This chapter presents the results we obtained in this work. Section 5.1 presents the results and experiments on our hand gesture recognition system. Section 5.2 covers several test scenarios for our gaming applications. Section 5.3 shows the results of the user study we conducted in this work.

5.1 Results on Hand Gesture Recognition

In this section, we present the results and experiments conducted to validate the robustness and assess the performance of our hand gesture recognition system.

5.1.1 Experimental Setup

We utilized the two self-generated datasets mentioned in Section 4.2 to train and evaluate the proposed hand gesture recognition system. We used Python programming language and leveraged scikit-learn [97], a well-known machine learning library for the Python programming language that features various machine learning algorithms and useful tools to evaluate machine learning models. All the experiments were conducted on an ASUS laptop equipped with an Intel Core i5-8250U 1.6GHz CPU and 12GB RAM.

5.1.2 Hand Pose Identification

We performed a test on five different machine learning algorithms, including SVM (our choice), Multilayer Perceptron (MLP), Random Forest, Logistic Regression and k-Nearest Neighbors (k-NN).

To find out the optimal set of hyperparameters for each algorithm, we performed k-fold cross-validation [98] technique on the pose training set and apply Grid Search [99] to search from a large space of hyperparameter settings to find out the models that have the highest accuracies on the validation set. Through experiments, we choose $k = 10$ as the parameter to cross validate the pose training set. The best set of hyperparameter for each algorithm after applying Grid Search, together with their cross-validation accuracies, are shown in Table 5.1. SVM and MLP have the highest validation accuracies with 97.7 and 97.15, respectively.

Algorithm	**Best Hyperparameters**	**Validation Accuracy (%)**
SVM	kernel='rbf', $C = 100.0$, $\gamma = 0.01$	**97.70**
MLP (100,)*	activation='relu', $\alpha = 0.001$, learning_rate='adaptive'	**97.15**
Random Forest	criterion='gini, max_depth=8, n_estimators=200	96.15
Logistic Regression	$C = 1.0$, regularization='l2'	86.69
k-NN	$k = 3$	95.99

Table 5.1: The best hyperparameters and the validation accuracy for five different machine learning algorithms after hyperparameter tuning. SVM and MLP have the highest validation accuracies.
*: (100,) means there is one hidden layer with 100 units.

After finding the best hyperparameter settings, we re-trained the five models with the whole training set and evaluated those models with the pose test set. The result is shown in Table 5.2. The precision, recall, and F_1 score for the whole dataset is calculated as the weighted sums of the precision, recall and F_1 score on all the individual classes. The weight of a class is determined by the proportion of that class's samples on the training set. It can be seen from the result that SVM and MLP are the two models that have the best performance. Specifically, SVM is the best model with the highest accuracy (96.84%), precision (0.9698), recall (0.9684) and F_1 score

score (0.9684). Therefore, we adopt SVM as the pose classifier for our system.

Model	**Accuracy (%)**	**Precision***	**Recall***	**F_1 score***
SVM	**96.84**	0.9698	0.9684	**0.9684**
MLP (100,)	**96.47**	0.9686	0.9673	**0.9675**
Random Forest	91.34	0.9300	0.9134	0.9145
Logistic Regression	82.21	0.8336	0.8221	0.8211
k-NN	92.21	0.9323	0.9222	0.9225

Table 5.2: Evaluation results on pose test set of five classification models. *: The precision, recall and F_1 score is the weighted average from all classes.

To evaluate the performance of our pose classifier (SVM) on each hand pose class, the confusion matrix of the classifier on the pose test set is shown in Figure 5.1. The precision, recall and F_1 score on each class are depicted in Table 5.3.

As can be observed from Table 5.3, our pose classifier demonstrates outstanding performance on multiple classes. The classifier achieves exceptional precision scores on some classes (`Right`, `Fist`, `Thumb in`, `Palm up`), while having remarkable recall scores on some other classes (`Stop`, `Thumb in`, `Palm left`, `Palm right`). Notably, our system excels in accurately classifying the `Thumb in` class, achieving a flawless F_1 score of 1.0000.

Some minor mistakes made by the classifier can be observed in Figure 5.1. Approximately 9% of the `Palm up` samples were classified as `Palm left`, while 8% of the `Down` samples were classified as `Left`. As a result, the recall scores for the `Palm up` and `Down` classes (0.9060 and 0.8922, respectively) are slightly lower compared to the other classes. The two lowest precision scores belong to the `Left` class (0.9044) and the `Palm` class (0.9130). The lower precision score on the `Left` class can be attributed to the model's tendency to predict `Down` as `Left`. Meanwhile, the lower precision score on the `Palm` class can be explained by the occurrence of some false positive cases where the model misclassifies samples from other classes as `Palm`, as

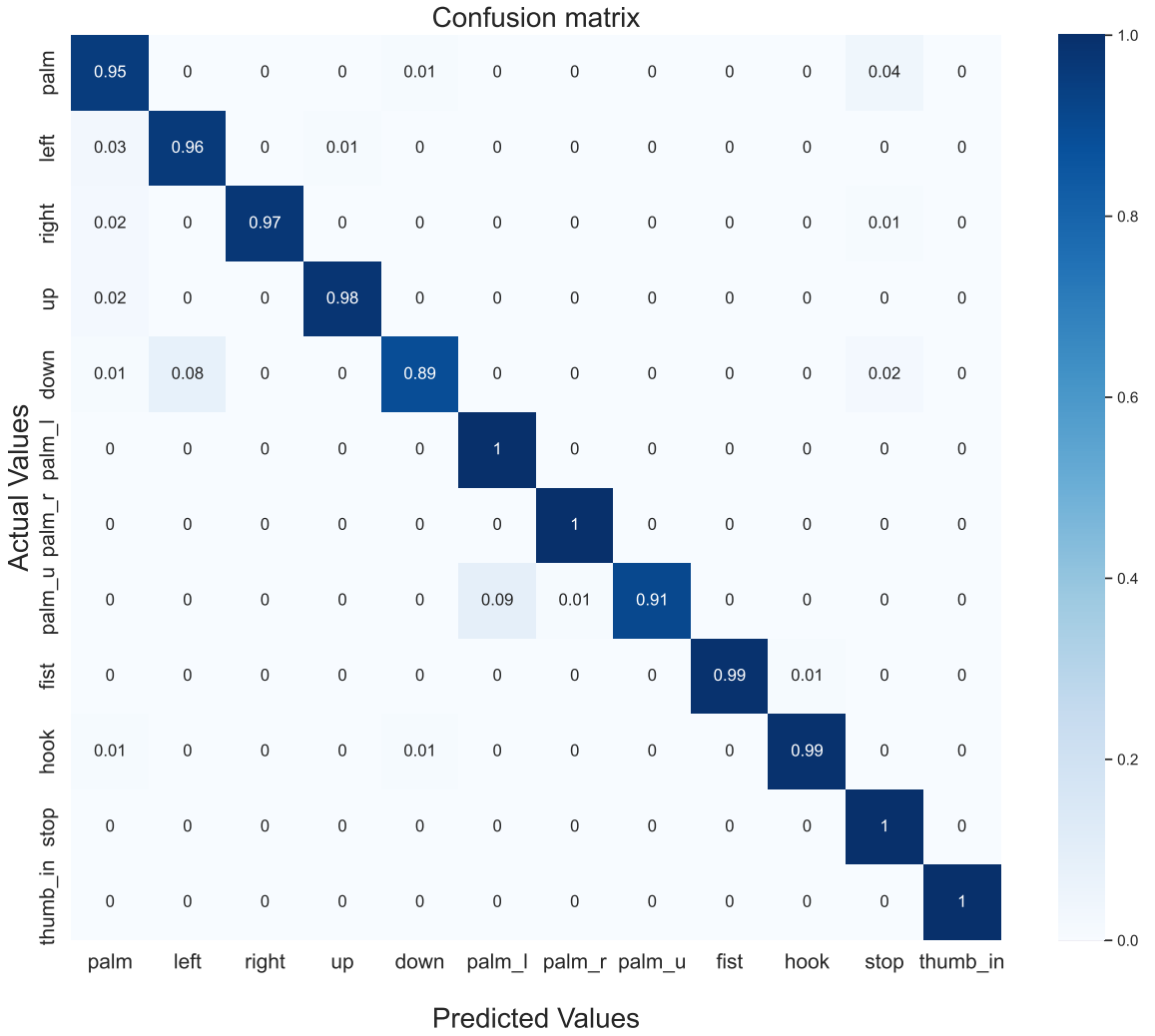


Figure 5.1: Confusion matrix of SVM on the pose test set. The SVM classifier achieves a high accuracy.

depicted in Figure 5.1.

We also conducted a study on the effect of feature extraction on the classification of hand poses. Table 5.4 presents the accuracy of the SVM and MLP models under different feature extraction settings. In the table, `Distances`, `Palm angles`, and `Finger angles` denote the inclusion of distance-based features, the three angles yaw, pitch, roll, and the angles between fingers, respectively.

Pose	Precision	Recall	F_1 score
Palm	**0.9130**	0.9492	**0.9307**
Left	0.9218	0.9593	0.9492
Right	**1.0000**	0.9722	0.9859
Up	0.9939	0.9760	0.9849
Down	0.9803	**0.8922**	0.9342
Fist	**1.0000**	0.9897	0.9948
Hook	0.9867	0.9867	0.9867
Stop	0.9321	**1.0000**	0.9649
Thumb in	**1.0000**	**1.0000**	**1.0000**
Palm left	**0.9044**	**1.0000**	0.9498
Palm right	0.9916	**1.0000**	0.9958
Palm up	**1.0000**	**0.9060**	0.9507

Table 5.3: The precision, recall and F_1 score on each hand pose class using SVM model.

For both models, the highest accuracy is achieved when all of the mentioned features are included (96.84% for SVM and 96.47% for MLP). `Palm angles` has a significant impact on pose classification, as omitting the features in this group results in a significant decrease in classification performance (66.8% for SVM and 69.5% for MLP). This is expected since the three global features (yaw, pitch, and roll) play an important role in recognizing poses involving hand movement, such as `Left`, `Right` or `Rotate left`, and `Rotate right`. Distance-based features also play a vital role in the models' performance, and excluding these features leads to a considerable drop

in accuracy (75.99% for SVM and 79.92% for MLP). Finally, using the raw skeletal data without feature extraction still yields acceptable performance, with an accuracy of 85.32% for SVM and 84.96% for MLP.

Feature Extraction Setting	**Accurracy (%)**	
	SVM	MLP
All features (Distances + Palm angles + Finger angles)	96.84	96.47
Without Palm angles	66.80	69.50
Without Distances	75.99	79.72
Without Finger angles	96.68	95.38
Raw skeletal data	85.32	84.96

Table 5.4: Performance of pose classifier on different feature extraction settings. The feature extraction of distances, palm angles and finger angles combined with SVM classifier achieve the best performance in terms of accuracy.

5.1.3 Hand Gesture Recognition

After finding the best model for hand pose classification, we tested the performance of our pose-based gesture recognition algorithm on the gesture dataset. The confusion matrix and the overall accuracy of our method on the gesture dataset are shown in Figure 5.2.

We achieved an accuracy of 97.95% on the gesture dataset and obtained impressive results for most of the gesture classes. However, there is a minor number of false positive cases and false negative cases that affect the system's performance. We observe that the false positive cases are due to the errors in hand pose classification, making the system recognize hand gestures where there are no gesture at all. On the other hand, the false negative cases happen when the pose classifier misses some key hand poses that are required to construct hand gestures, and when the subjects did not performed all the key poses that are required to construct hand gestures.

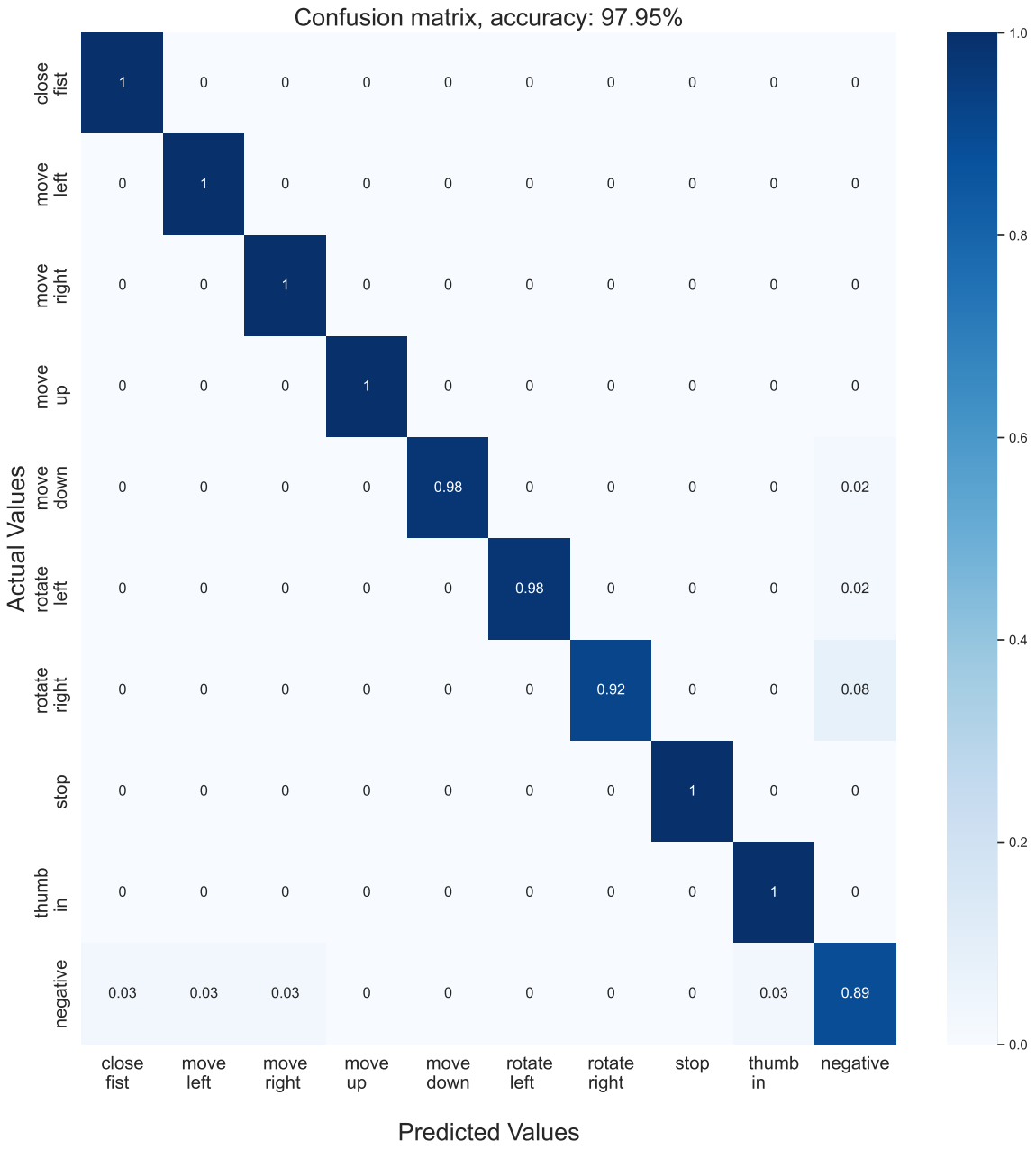


Figure 5.2: [Confusion matrix and accuracy score of our system on the gesture dataset. We achieved a high accuracy of 97.95% and impressive results on many classes. Performance issues are due to the false positive cases made by our method.

The precision, recall, and F_1 scores for each hand gesture are illustrated in Table

5.5. It can be observed that our approach achieved impressive results on several gesture classes, with the maximum precision score in classes such as `Move up` and `Stop`, and the maximum recall score on classes such as `Move left`, `Move right`, `Close fist` and `Thumb in`. The weighted precision, recall and F_1 score for all of the classes are 0.9903, 0.9876 and 0.9887, respectively.

However, there are still some robustness issues due to the lower recall rate for the gesture `Rotate right` (0.9211). The issues happen in some extreme cases when the subject has not performed the entire gesture properly. For example when a user wants to perform the `Rotate right` gesture but has not completed all the key poses required to construct that gesture (e.g. they have performed `Palm` and `Palm left`, but lack the `Palm up` pose to complete the gesture). All of the other wrong cases are due to the errors made in hand pose identification, when the pose classifier misses some key poses or misclassifies some poses as other poses, leading to the wrong construction and matching of hand gestures.

We have also applied several heuristics to improve the system's performance and alleviate the mistakes made in hand pose classification. These heuristics are responsible for validating the palm angles of the identified hand pose and making some correction if needed. For instance, if the `Left` pose is identified while the yaw angle has a small magnitude, the system will classify it as `Palm` instead. We found that setting 16° as the minimum value of the yaw angle for `Left` achieves the highest performance gain on our gesture dataset. We applied a similar strategy for all other classes. Additionally, we set the minimum confidence threshold to 0.2 for a hand pose to be successfully classified. If the confidence score for an identified pose is below this threshold, our system will discard the classification result.

The effects of heuristics on some of the gesture classes are shown in Table 5.6. As can be seen, there is an improvement in the recall and precision rates after adding heuristics to our system. The overall accuracy increases from 93.24% to 97.95%.

Gesture	Precision	Recall	F_1 score
`Move left`	0.9796	1.0000	0.9897
`Move right`	0.9804	1.0000	0.9901
`Move up`	1.0000	1.0000	1.0000
`Move down`	1.0000	0.9800	0.9899
`Rotate left`	1.0000	0.9750	0.9873
`Rotate right`	1.0000	0.9211	0.9589
`Close fist`	0.9804	1.0000	0.9901
`Stop`	1.0000	1.0000	1.0000
`Thumb in`	0.9756	1.0000	0.9877
Weighted average	0.9903	0.9876	0.9887

Table 5.5: Precision, recall and F_1 scores on the gesture dataset. High results are achieved in many classes. Robustness issues are due to the lower recall rate of `Rotate right`.

Gesture	Before Heuristics		After Heuristics	
	Precision	Recall	Precision	Recall
`Move left`	0.9792	0.9792	0.9796	**1.0000**
`Move right`	0.9231	0.9600	**0.9804**	**1.0000**
`Move up`	0.9074	1.0000	**1.0000**	1.0000
`Move down`	0.9412	0.9600	**1.0000**	**0.9800**
`Rotate left`	1.0000	0.9500	1.0000	**0.9750**
`Rotate right`	1.0000	0.8684	1.0000	**0.9211**
`Thumb in`	0.9744	0.9500	0.9756	**1.0000**
Accuracy (%)	**93.24**		**97.95**	

Table 5.6: The effects of heuristics to our system's performance. Adding heuristics improves the precision and recall rates on several classes and increases the accuracy from 93.24% to 97.95%

5.1.4 Time Performance

We evaluated the time performance of our hand gesture recognition system by calculating the average execution time per frame for each sample in the gesture dataset. The execution time per frame in a sample is calculated by dividing the total elapsed time by the number of frames in that sample. In our dataset, a frame represents the hand skeletal data extracted by the Leap Motion Controller. We then computed the average execution time across all 438 samples. The results are presented in Table 5.7. The average execution time for hand gesture recognition is 1.4 ms, equivalent to 833 frames per second (FPS).

It is important to note that the overall time performance of our system depends on the process of hand skeleton extraction, as the system needs to wait for the skeletal information extracted by the Leap Motion Hand Tracking Software. During our experiments, the Leap Motion's tracking service consistently operated at a frame rate of 60 frames per second (16.67 ms per frame). Since the execution time of hand gesture recognition is significantly lower than the time required for skeleton extraction (1.4 ms compared to 16.67 ms), the overall execution time of our system remains within 16.67 ms (60 FPS). Therefore, our system can easily achieve real-time performance.

Skeleton Extraction by LMC		**Hand Gesture Recognition**		**Overall**	
Time	FPS	Time	FPS	Time	FPS
16.67	60	1.4	833	**16.67**	**60**

Table 5.7: Execution time and frames per second of our system. Our system can easily obtain real-time performance at the frame rate of 60 FPS, note that execution time is measured in ms per frame.

5.2 Test Scenarios for Gaming Application

In software development, the process of testing holds paramount importance to ensure the quality and functionality of applications [100]. This is particularly true for

gaming applications, where user engagement and satisfaction are key factors for success. To effectively evaluate the performance and behavior of a gaming application, a comprehensive set of test scenarios must be devised. These test scenarios encompass a range of potential user interactions and system responses, facilitating the identification and resolution of any issues before the application is released to the public.

In this section, we meticulously devise multiple test scenarios to evaluate the performance and user experience of our gaming application. These tests are of utmost importance as they ensure smooth functionality and enhance user satisfaction. The test scenarios are outlined and organized in Table 5.8, Table 5.9, Table 5.10, and Table 5.11.

ID	Description	Precondition	Action	Result
101	The user moves his hand in the left or right direction. The selective square indicating the shape being chosen moves accordingly.	- User selected the Shapes and Colors game. - The gaming application and the hand gesture recognition system runs properly.	1. User moves his hand left or right. 2. The white square (chosen answer) moves in the same direction.	Passed
102	The user performs the gesture `Close fist`, to confirm the choosing answer of the shape inside the white square by the instruction sentence above.	- User selected the Shapes and Colors game - The chosen answer is being selected. - The gaming application and the hand gesture recognition system runs properly.	1. User performs the gesture `Close fist`. 2. The shape inside the white square (chosen answer) is selected.	Passed
103	The user moves his hand in the left or right direction slightly. The selective square indicating the shape being chosen moves accordingly.	- User selected the Shapes and Colors game. - The gaming application and the hand gesture recognition system runs properly.	1. User moves his hand left or right slightly. 2. The white square (chosen answer) moves in the same direction.	Failed
104	The user performs the gesture `Close fist` fast, to confirm the choosing answer of the shape inside the white square by the instruction sentence above.	- User selected the Shapes and Colors game - The chosen answer is being selected. - The gaming application and the hand gesture recognition system runs properly.	1. User performs the gesture `Close fist` fast. 2. The shape inside the white square (chosen answer) is selected.	Passed

Table 5.8: Test cases for game Shapes and Colors.

ID	Description	Precondition	Action	Result
201	The user moves his hand in the left or right direction, then the basket moves accordingly.	- User selected the Eggs and Milk game. - The gaming application and the hand gesture recognition system runs properly.	1. User moves his hand left or right. 2. The basket moves in the same direction.	Passed
202	The user rotates his hands, then the game changes the container type.	- User selected the Eggs and Milk game. - The gaming application and the hand gesture recognition system runs properly.	1. User rotates his hand. 2. The game adapts the container.	Passed
203	The user moves his hand in the left or right direction slightly. The basket moves accordingly.	- User selected the Eggs and Milk game. - The gaming application and the hand gesture recognition system runs properly.	1. User moves his hand left or right slightly. 2. The basket moves in the same direction.	Failed
204	The user rotates his hands fast, then the game changes the container type.	- User selected the Eggs and Milk game. - The gaming application and the hand gesture recognition system runs properly.	1. User rotates his hand fast. 2. The game adapts the container.	Passed

Table 5.9: Test cases for game Eggs and Milk.

ID	Description	Precondition	Action	Result
301	The user performs the gesture `Move up` to make the Dino jump up to avoid the obstacles.	- User selected the Dino Run game. - The gaming application and the hand gesture recognition system runs properly.	1. User performs the gesture `Move up`. 2. The Dino jumps for a period of time to avoid certain obstacles.	Passed
302	The user performs the gesture `Move down` to make the Dino bend down to avoid the bird.	- User selected the Dino Run game. - The gaming application and the hand gesture recognition system runs properly.	1. User performs the gesture `Move down`. 2. The Dino bends down for a period of time to avoid birds.	Passed
304	The user perform the gesture `Move up` or `Move down` with wrong starting pose.	- User selected the Dino Run game. - The gaming application and the hand gesture recognition system runs properly.	1. User performs the gesture `Move up` or `Move down` with wrong starting pose. 2. The Dino reacts to gestures and avoids obstacles with specific actions.	Failed

Table 5.10: Test cases for game Dino Run.

ID	Description	Precondition	Action	Result
401	The user performs the gesture `Stop`, then the game will be paused and a number of navigation is presented	- User is playing for any of the three games. - The gaming application and the hand gesture recognition system runs properly.	1. User performs the gesture `Stop`. 2. The game will be paused, then instructions about the playing game and options for navigation are presented.	Passed
402	The user performs the gesture `Move up` or `Move down`, then the selected navigation is changed according to the users movement.	- The game is being paused. - The gaming application and the hand gesture recognition system runs properly.	1. User performs the gesture `Move up` or `Move down`. 2. The selected navigation is changed according to the users movement.	Passed
403	The user performs the gesture `Close fist`, then the user is navigated to the selected navigation.	- The game is being paused. - The selected navigation is being chosen. - The gaming application and the hand gesture recognition system runs properly.	1. User performs the gesture `Close fist`. 2. The user is navigated to the selected navigation.	Passed
404	The user loses the game, then the game-over interface is presented.	- User loses the game.	1. User loses the game. 2. The game-over screen is presented with the score that the player has achieved.	Passed

Table 5.11: Test cases for other functions.

5.3 User Study

We conducted a user study to better understand how individuals interact with our system. Our primary goal is to examine and focus on the varied responses and experiences of users. This would help us gain a more comprehensive understanding of their needs and generate ideas for refining and tailoring our solution to better suit them.

5.3.1 Methodology

We selected a diverse group of participants who have some connections with hand rehabilitation, such as individuals who had previously undergone hand rehabilitation, caregivers of patients, experts in the field of hand rehabilitation, as well as normal individuals with no prior experience or relation to the subject.

The study was conducted in two locations: a residential area with a high population density of elderly individuals, and a private home. The residential area was selected for its large elderly population, making it more likely to encounter individuals who were in need of hand rehabilitation or had relevant experiences to contribute. This facilitated the recruitment of suitable participants and provided a more realistic and practical environment for the study. Figure 5.3 depicts the locations where our user study takes place.

By focusing on this diverse group of participants and conducting the study in real-world settings, we were able to gather valuable insights into how our solution was perceived and experienced by different individuals. This information will prove invaluable in guiding the further development and optimization of our solution, ultimately enhancing its effectiveness and usability for a wide range of individuals in need of hand rehabilitation.

The user study was structured in two distinct stages to ensure a comprehensive understanding of participants' experiences and gather valuable insights into the effectiveness

Figure 5.3: Locations to conduct our user study. The left picture is captured in a private home, while the right one is captured in a residential area.

of our hand rehabilitation solution:

1. Pre-survey.

   The primary objective of this stage is to gather relevant data from the participants, both directly and indirectly related to hand rehabilitation. This involves obtaining information about their background, previous experiences with hand rehabilitation, expectations, and any specific concerns or requirements they might have. Pre-survey data provided us with a solid foundation to better understand the participants' individual perspectives and allows us to tailor the study to address their unique needs and expectations.

2. Post-survey.

   Upon completion of the pre-survey stage, participants were given the opportunity to experience our gaming application, consisting of three interactive games. Once they had finished playing the games, we collected their feedback regarding our solution in the post-survey stage. This feedback allowed us to gain insights

into the perceived opportunities, efficiency, and effectiveness of our solution from the participants' standpoint.

The combination of these two stages enable us to paint a complete picture of how our system was perceived by the participants, taking into account their initial expectations, their experiences with the game, and their overall impressions. This invaluable feedback will be instrumental in guiding the future development and refinement of our solution, ensuring it better serves the needs of a diverse range of individuals requiring hand rehabilitation.

5.3.2 Pre-survey

The pre-survey stage helps us to understand more of the background of the invited participants. There were 10 participants and most of them are elders. As our solution aims to support people with weak status in both mental and physical health, elders are the great users to test the effectiveness of our solution. The pre-survey result is presented in Table 5.12.

Randomly, our participants gender was distributed 50% for two genders. This gave us equal insights into them, therefore we could understand how different the two genders' opinions are to develop the solutions to be more effective and suitable.

To gain insights into the participants' familiarity with hand rehabilitation, we inquired about their prior experiences in this regard. Out of the participants, two individuals responded affirmatively. One of them had undergone rehabilitation due to a traffic accident, while the other had previously worked as a long-distance driver. This information is of significant importance for our study as it provides valuable insights into how individuals who have encountered hand rehabilitation expect from our solution.

We surveyed the participants on whether they had played any interactive games before and their frequencies of playing video games. The result shows that only 3

Variable		n
Age (years)	15-30	1
	30-45	1
	45-60	1
	> 60	7
Gender	Male	5
	Female	5
Have experienced physical therapy?	Yes	2
	No	8
Have played hand-interactive games?	Yes	3
	No	7
Gaming frequency per week?	0	6
	1-5	2
	5-10	1
	> 10	1

Table 5.12: Demographic characteristics of the study population. This table presents the population distribution for various questions that participants must answer before playing the game, with `n` denoting the number of people.

people had experienced hand-interactive games prior to our study. This information is useful since we set different expectations for each participant according to whether they had ever played games or not.

5.3.3 Post-survey

In this section, we present the post-survey outcomes of our user study on the 10 participants. Each participant was asked some questions about their experiences while playing the games. They were also asked to provide feedback that help improve the effectiveness of our system. The post-survey result is shown in Figure 5.4

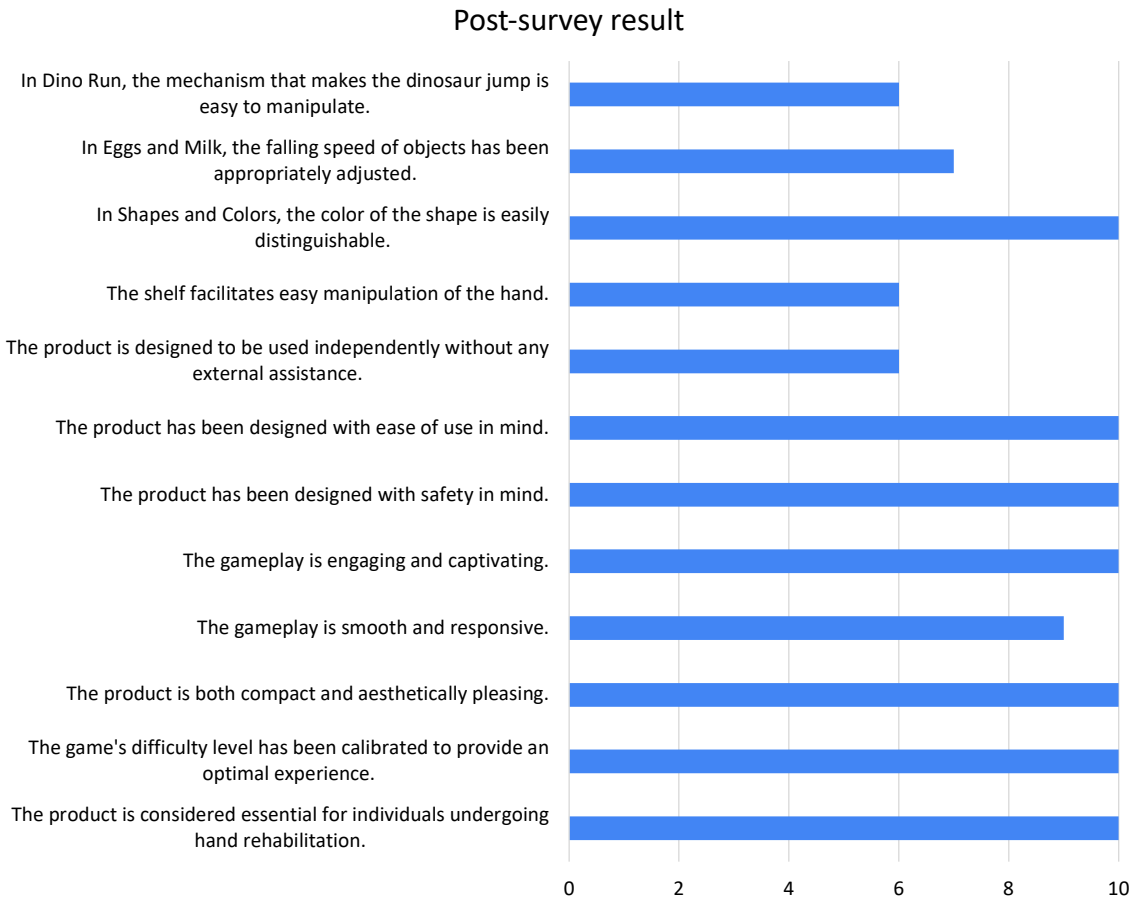


Figure 5.4: Result of the post-survey. After playing our games, the participants were asked to complete a questionnaire regarding the effectiveness of the solution.

Overall, the participants expressed favourable feedback regarding our system. Most of them felt that our solution is compact, lightweight, enjoyable and easy to use.

Moreover, the aesthetic design of the game plays a crucial role in ensuring that individuals are motivated to use it regularly and create a great user experience. From the post-survey result, all the participants found that our application is aesthetically pleasing. Additionally, the game's engaging design and acceptable difficulty levels were praised by the participants.

Most importantly, all of the participants found that our gaming application is an interesting solution for hand rehabilitation. Most of them agreed that our application brings a new experience and an interactive environment that motivated them to play and enjoy the games. Therefore, our system could be a potential solution to replace traditional hand rehabilitation approaches.

We also evaluated how well the participants can play each game by recording the maximum score that each participant achieved in each game. The objective of this evaluation is to know whether the difficulties level that we defined are suitable or not and from that we can adjust the games more appropriately. The evaluation result is shown in Figure 5.5. In each game, there are three levels, Easy, Medium and Hard. In the Shapes & Colors and Eggs & Milk games, the Easy, Medium and Hard level correspond to 0-7 scores, 8-15 scores and $>$ 15 scores respectively. In the Dino Run game, the score ranges for each level are 0-50 scores, 50-100 scores and $>$ 100 scores, respectively.

In the first game, Shapes & Colors, 5 patients reached the Medium level while the others could only play in the Easy level. In the second game, Eggs & Milk, all the participants reached the Medium level, which is a good response to our system. In the Dino Run game, only 2 participants were at the Easy level whereas the others could easily reach the Medium level. Most of the participants struggled to reach the Hard level. In fact, there was no participants who could reach the Hard level in all the three games. From the results, we can conclude that the Hard level in all three games need to be adjusted to be easier for the users.

The study did reveal some concerns regarding the product. One participant reported experiencing lag while playing the game, and four participants reported needing assistance with operating the product. These concerns will require further investigation and action to address and enhance the product's overall user experience.

5.3.4 Evaluating the Difficulty of Our Gaming Application

In order to further assess the difficulty of our gaming application more, we created hypothesis tests for a more systematic evaluation. As we only have 10 participants, the sample size is quite limited to have any significant conclusion. However, we want to create a general evaluation framework that can provide a statistical description of the data and can be applied in future when we get more data from larger user studies.

Our hypothesis is that more than 50% of users should be able to reach the medium level in each game. To test this hypothesis, we will use a statistical approach called the One-Proportion Z-Test. This is a statistical hypothesis test used to determine whether a proportion observed in a sample is significantly different from a specified population proportion. It is based on the normal distribution and is particularly applicable when working with categorical data.

A One-proportion Z-test is used to compare an observed proportion to a theoretical one. The test statistic is calculated as in Equation 5.1:

$$z = \frac{\hat{p} - p}{\sqrt{\frac{p \times q}{n}}} \tag{5.1}$$

where:

- x is the number of successes or observations
- n is the sample size.
- p is the population proportion.
- n is the sample proportion, given by $\hat{p} = \frac{x}{n}$.
- q is the complement of the population proportion, given by $q = 1 - p$.
- α is the significance level.

Once the Z-statistic is calculated, it can be compared to the critical values of the standard normal distribution to determine the statistical significance. The Z-statistic measures the number of standard deviations that a sample proportion deviates from the expected proportion under the null hypothesis. By comparing the Z-statistic to the critical values, we can evaluate whether the observed proportion is significantly different from the expected proportion.

The critical values are typically obtained from a Z-table, which provides the cut-off points corresponding to specific significance levels. Alternatively, statistical software can be used to calculate the critical values. These critical values determine the boundaries for rejecting or failing to reject the null hypothesis based on the level of significance chosen for the test.

The One-Proportion Z-Test is a statistical test commonly employed in various fields. It is particularly useful when comparing a sample proportion to a known population proportion or when assessing the effectiveness of interventions or treatments. In market research, for example, the One-Proportion Z-Test can be used to determine if the proportion of consumers with a certain characteristic is significantly different from a predetermined value. In public health, it can be applied to evaluate the success of a public health intervention by comparing the proportion of individuals affected before and after the intervention. Similarly, in quality control, the test can be used to assess whether a process is meeting quality standards by comparing the proportion of defective items to a specified target.

We have gathered data from a small sample of 10 participants and have established a significance level of 0.05 for the p-value. In this study, we are interested in determining the proportion of individuals who have reached the medium level, denoted as p. The collected data and evaluation results are presented in Table 5.13.

To investigate the relationship between the proportion p and the medium level, we have formulated the following hypotheses:

- Null Hypothesis (H_0): $p \leq 0.5$.
- Alternative Hypothesis (H_1): $p > 0.5$.

These hypotheses will help us assess whether the observed proportion significantly exceeds 0.5 or not.

Game	**Number of Players at Medium Level**	p**-value**
Shapes and Colors	5	0.5
Eggs and Milk	10	**0.0**
Dino Run	9	**0.0000124**

Table 5.13: Results of the One-Proportion Z-Test for a sample of 10 participants reaching the medium level with a p-value threshold of 0.05.

The results show that the p-values for Eggs and Milk and Dino Run are less than 0.05. This indicates that our hypothesis of more than 50% of users reaching the medium level in these games is supported by the data. In other words, these two games seem to have an appropriate level of difficulty for users.

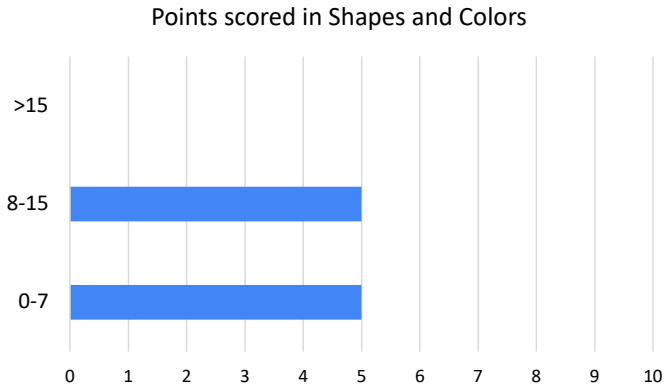
However, the p-value for Shapes and Colors is 0.5, which is larger than our threshold of 0.05. This suggests that the proportion of users reaching the medium level in the game does not align with our hypothesis. In this case, we fail to reject the null hypothesis, meaning that the difficulty of Shapes and Colors might be higher than intended.

Based on our analysis, we see that the game Shapes and Colors needs to be easier for users to reach the medium level. This could involve adjusting the game mechanics, providing clearer instructions, or offering more accessible challenges. By making these changes, we can ensure that this game is more in line with our initial hypothesis and users can have a more enjoyable and engaging experience across all three games.

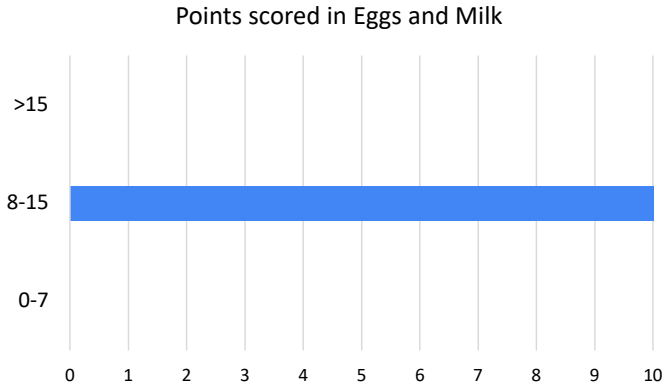
5.3.5 Discussion on Survey Results

The user study has generated insightful feedback regarding our solution. In general, there was a good response from the participants where most of them reported that the hand gesture-based game was effective, exciting and comfortable. Most of the participants reported that they felt no pain and difficulty in performing gestures to control the games. Most importantly, all of the users felt that our system is a potential solution for hand rehabilitation.

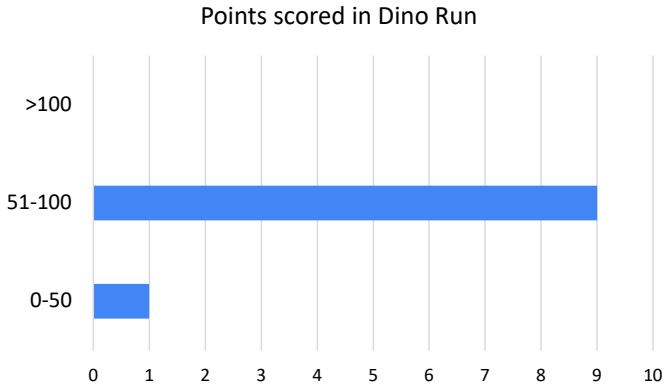
Nevertheless, there were a few concerns raised that require attention, two of which are the problem of lag and the issue of requiring assistance to use the product. Therefore, we will need to have a review on the technical aspects of our system and design a clearer instruction in future user studies to help the participants remember clearly all the hand gestures before let them experience our application. Also, the difficulty level of the games needs to be adjusted to be easier so that most of the users can enjoy the game without any problems.



(a) User's performance in Shapes and Colors.



(b) User's performance in Eggs and Milk.



(c) User's performance in Dino Run.

Figure 5.5: User's performance on the three interactive games. Each game is accompanied by a chart that displays the number of individuals who attain scores falling within a particular range. The ranges are typically defined as: 0-7, 7-15, and above 15.

Chapter 6

Conclusion

In this chapter, we conclude our work in this capstone project. Section 6.1 is a summary of our work. Section 6.2 evaluates the advantages and limitations of the proposed system. Finally, Section 6.3 presents our future plan after this capstone project.

6.1 Summary

In this project, we proposed and implemented a real-time hand gesture recognition system for game-based hand rehabilitation. We created two datasets, a pose dataset and a gesture dataset, using the Leap Motion Controller, and conducted some experiments to evaluate the performance of our approach on those datasets. The results show that the proposed hand gesture recognition approach achieve high accuracy rates of 96.84% on the our pose dataset and 97.95% on our gesture dataset.

We also developed a gaming application consisting of three interactive video games. These games are simple and specially designed for hand rehabilitation. Each game has different levels that can be used to assess the hand dexterity of patients. By playing video games, patients will feel more engaging while still participating in the process of recovery. This is done by repetitively perform hand gestures to interact with the game environment.

Lastly, we conducted a user study with 10 individuals who have different demographic characteristics to collect their opinions regarding our work and evaluate the effectiveness of our solution on real users. The post-study results show that our solution is interesting and highly potential to be applied in hand rehabilitation.

6.2 Solution Evaluation

In this section, we thoroughly assess our solution and outline its advantages and limitations.

6.2.1 Advantages

There are some advantages of our system compared to similar existing work on hand gesture recognition for game based rehabilitation:

- Firstly, we adopt an enjoyable gamified approach that can be a great solution to replace boringly traditional hand rehabilitation approaches. Through using our system, the user can feel more enjoyable and engaging playing interactive video games while still participating in the process of hand recovery.
- Secondly, as we select a set of hand gestures that are commonly used in hand rehabilitation, and adopt a setting where users can rest their hands on the arm support, our system has a more significant impact on rehabilitation compared to most of the existing works on hand gesture recognition for hand rehabilitation.
- Thirdly, the integration of hand gesture recognition and three interactive video games into a single system makes our system a unique and complete solution for game-based hand rehabilitation. This can be an advantage over some existing works since they focus on either hand gesture recognition or game development for hand rehabilitation.
- Fourthly, the pose-based approach for hand gesture recognition allows a simple system that can easily obtain real-time performance. Although prioritizing

efficiency, our system still achieves remarkable results on our two self-generated datasets.

- Last but not least, compared to other existing works, we have a user study conducted on real users with hand problems, which is an important factor to evaluate the effectiveness of our system in reality.

6.2.2 Limitations

Although achieving impressive results, there are some limitations in this work:

- Firstly, there are limited benchmarks for hand gesture recognition on Leap Motion's data, so it is difficult to compare the robustness of our method against existing works on the same research area. Furthermore, the hand gestures that we selected from common hand rehabilitation exercises are different from other studies, which makes it nearly impossible to have any benchmarks for comparison with other methods.

- Secondly, our datasets were collected from a small number of individuals. In reality, people can have several different hand shapes and hand sizes, which can affect the recognition accuracy of our system. Therefore, larger and more extensive datasets are required to ensure the generalization of our system on new users.

- Thirdly, the design using hand poses to recognize hand gestures are limited to simple gestures composed of distinctive key poses. For gestures that require extremely accurate movements, our system may have some trouble. Moreover, our design relies heavily on hand pose identification, so a minor error on this stage can significantly effect the overall performance of the system. Also, the proposed system is strongly dependent on the manual process of building the key pose dictionary. Therefore, it can be challenging to scale up this design to a broader set of hand gestures.

- Finally, the user study is conducted with only 10 individuals, which is still a limited number to accurately evaluate the effectiveness of our solution. Since

most of the individuals live in the same residential area, the result might be biased and not reliable enough. Therefore, a large scale user study should be conducted in future to have more reliable statistics.

6.3 Future Work

With the future potential of our system and some of the limitations that have already been discussed, we list out the following goals for future work:

- We aim to extend the project to a larger scale by writing research papers, attending start-up competitions and contacting with hospitals and organizations for hand rehabilitation to gain a better insight and upgrade our system. As a result, we can have larger datasets and resources to enhance the robustness and the generalization of our system.
- An alternative solution to the pose-based hand gesture recognition method is necessary to make our system more robust and can adapt to a broader set of hand gestures. Time series approaches can be a more potential solution for hand gesture recognition. More research needs to be done for an accurate yet effective method for hand gesture recognition.
- A larger-scale user study will be conducted to provide more accurate opinions from the users. Specifically, real patients with hand problems should be targeted as they will give more relevant insights and feedbacks regarding the effectiveness of our system. Therefore we can adjust the system to suit the needs of real patients.
- We are planning to have a copyright registration for our system in Vietnam. Registering a copyright will make our solution become more popular and avoid any illegal use of our idea for commercial use and bad purposes.

References

[1] N. M. Tuah, F. Ahmedy, A. Gani, and L. N. Yong, "A Survey on Gamification for Health Rehabilitation Care: Applications, Opportunities, and Open Challenges," *Information*, vol. 12, no. 2, 2021.

[2] P. D. E. Baniqued, E. C. Stanyer, M. Awais, A. Alazmani, A. E. Jackson, M. A. Mon-Williams, F. Mushtaq, and R. J. Holt, "Brain–computer interface robotics for hand rehabilitation after stroke: a systematic review," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, no. 1, p. 15, 2021.

[3] V. L. Feigin, M. Brainin, B. Norrving, S. Martins, R. L. Sacco, W. Hacke, M. Fisher, J. Pandian, and P. Lindsay, "World Stroke Organization (WSO): Global Stroke Fact Sheet 2022," *International Journal of Stroke*, vol. 17, no. 1, pp. 18–29, 2022.

[4] GBD 2019 Stroke Collaborators, "Global, regional, and national burden of stroke and its risk factors, 1990-2019: a systematic analysis for the Global Burden of Disease Study 2019," *Lancet Neurol*, vol. 20, no. 10, pp. 795–820, 2021.

[5] D. Nichols-Larsen, P. Clark, A. Zeringue, A. Greenspan, and S. Blanton, "Factors Influencing Stroke Survivors Quality of Life During Subacute Recovery," *Stroke; a journal of cerebral circulation*, vol. 36, pp. 1480–1484, 2005.

[6] I. Ayed, A. Ghazel, A. Jaume-i Capó, G. Moyà Alcover, J. Varona, and P. Martinez Bueso, "Vision-Based Serious Games and Virtual Reality Systems for

Motor Rehabilitation: A Review Geared Toward a Research Methodology," *International Journal of Medical Informatics*, vol. 131, 2019.

[7] J. Byra and K. Czernicki, "The Effectiveness of Virtual Reality Rehabilitation in Patients with Knee and Hip Osteoarthritis," *Journal of Clinical Medicine*, vol. 9, no. 8, 2020.

[8] R. M. Al-Whaibi, M. S. Al-Jadid, H. R. ElSerougy, and W. M. Badawy, "Effectiveness of virtual reality-based rehabilitation versus conventional therapy on upper limb motor function of chronic stroke patients: a systematic review and meta-analysis of randomized controlled trials," *Physiother Theory Pract*, vol. 38, no. 13, pp. 2402–2416, 2021.

[9] K. E. Laver, B. Lange, S. George, J. E. Deutsch, G. Saposnik, and M. Crotty, "Virtual reality for stroke rehabilitation," *Cochrane Database Syst Rev*, vol. 11, no. 11, p. CD008349, 2017.

[10] A. Da Gama, P. Fallavollita, T. Chaves, L. Silva Figueiredo, A. Baltar, M. Ma, N. Navab, and V. Teichrieb, "MirrARbilitation: A clinically-related gesture recognition interactive tool for an AR rehabilitation system," *Computer Methods and Programs in Biomedicine*, vol. 135, 2016.

[11] E. R. Ramírez, R. Petrie, K. Chan, and N. Signal, "A Tangible Interface and Augmented Reality Game for Facilitating Sit-to-Stand Exercises for Stroke Rehabilitation," in *Proceedings of the 8th International Conference on the Internet of Things*, IOT '18, Association for Computing Machinery, 2018.

[12] Y. Bouteraa, I. B. Abdallah, and A. M. Elmogy, "Training of Hand Rehabilitation Using Low Cost Exoskeleton and Vision-Based Game Interface," *Journal of Intelligent & Robotic Systems*, vol. 96, no. 1, pp. 31–47, 2019.

[13] H. Kayama, K. Okamoto, S. Nishiguchi, M. Yamada, T. Kuroda, and T. Aoyama, "Effect of a Kinect-based exercise game on improving executive cognitive performance in community-dwelling elderly: case control study," *Journal of Medical Internet Research*, vol. 16, no. 2, p. e61, 2014.

[14] M. W. Cohen, I. Voldman, D. Regazzoni, and A. Vitali, "Hand Rehabilitation via Gesture Recognition Using Leap Motion Controller," in *2018 11th International Conference on Human System Interaction (HSI)*, pp. 404–410, 2018.

[15] W. Li, C. Hsieh, L. Lin, and W. Chu, "Hand gesture recognition for post-stroke rehabilitation using leap motion," in *Proceedings of the 2017 IEEE International Conference on Applied System Innovation* (T.-H. Meen, A. Lam, and S. Prior, eds.), pp. 386–388, Institute of Electrical and Electronics Engineers Inc., 2017.

[16] F. Farahanipad, H. R. Nambiappan, A. Jaiswal, M. Kyrarini, and F. Makedon, "HandReha: Dynamic Hand Gesture Recognition for Game-based Wrist Rehabilitation," in *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pp. 1–9, 2020.

[17] N. Potigutsai and O. Sornil, "Hand and Fingertip Detection for Game-Based Hand Rehabilitation," in *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 36–43, IEEE, 2021.

[18] K. Lai and S. N. Yanushkevich, "CNN+RNN Depth and Skeleton based Dynamic Hand Gesture Recognition," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 3451–3456, 2018.

[19] Y. Li, Z. He, X. Ye, Z. He, and K. Han, "Spatial temporal graph convolutional networks for skeleton-based dynamic hand gesture recognition," *EURASIP Journal on Image and Video Processing*, vol. 2019, pp. 1–7, 2019.

[20] V. Rehab, "Psychological Rehabilitation For Vulnerable Populations." `https://www.virtualrehab.co/`. Accessed: 2023-05-15.

[21] J. Chen, S. Zhao, H. Meng, X. Cheng, and W. Tan, "An interactive game for rehabilitation based on real-time hand gesture recognition," *Frontiers in Physiology*, vol. 13, p. 2250, 2022.

[22] C. W. Tan, S. W. Chin, and W. X. Lim, "Game-based human computer interaction using gesture recognition for rehabilitation," in *2013 IEEE International Conference on Control System, Computing and Engineering*, pp. 344–349, IEEE, 2013.

[23] M. Oudah, A. A. Al-Naji, and J. Chahl, "Hand Gesture Recognition Based on Computer Vision: A Review of Techniques," *Journal of Imaging*, vol. 6, p. 73, 2020.

[24] J. S. Sonkusare, N. B. Chopade, R. Sor, and S. L. Tade, "A review on hand gesture recognition system," in *2015 International Conference on Computing Communication Control and Automation*, pp. 790–794, 2015.

[25] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-Based Hand-Gesture Applications," *Commun. ACM*, vol. 54, no. 2, pp. 60–71, 2011.

[26] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, 2007.

[27] R. Z. Khan and N. A. Ibraheem, "Hand Gesture Recognition: A Literature Review," *International Journal of Artificial Intelligence & Applications*, vol. 3, no. 4, p. 161, 2012.

[28] X. Chu, J. Liu, and S. Shimamoto, "A Sensor-Based Hand Gesture Recognition System for Japanese Sign Language," in *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*, pp. 311–312, IEEE, 2021.

[29] R. Kehl and L. Van Gool, "Real-Time Pointing Gesture Recognition for an Immersive Environment," in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pp. 577–582, IEEE, 2004.

[30] A. K. Malima, E. Özgür, and M. Çetin, "A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control," in *2006 IEEE 14th Signal Processing and Communications Applications*, pp. 1–4, 2006.

[31] J. Xu, X. Zhang, M. Zhou, and I. You, "A High-Security and Smart Interaction System Based on Hand Gesture Recognition for Internet of Things," *Security and Communication Networks*, vol. 2018, 2018.

[32] A. Gupta, Y. Kumar, and S. Malhotra, "Banking security system using hand gesture recognition," *2015 International Conference on Recent Developments in Control, Automation and Power Engineering (RDCAPE)*, pp. 243–246, 2015.

[33] N. V. Le, M. Qarmout, Y. Zhang, H. Zhou, and C. Yang, "Hand gesture recognition system for games," in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pp. 1–6, 2021.

[34] M. R. Islam, R. Rahman, A. Ahmed, and R. Jany, "NFS: A Hand Gesture Recognition Based Game Using MediaPipe and PyGame," 2022.

[35] H. Vaezi Joze and O. Koller, "MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language," in *The British Machine Vision Conference (BMVC)*, 2019.

[36] A. Kapitanov, A. Makhlyarchuk, and K. Kvanchiani, "HaGRID - HAnd Gesture Recognition Image Dataset," *arXiv preprint arXiv:2206.08219*, 2022.

[37] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, and A. Del Bimbo, "3D Human Action Recognition by Shape Analysis of Motion Trajectories on Riemannian Manifold," *IEEE Transactions on Systems Man and Cybernetics*, 2014.

[38] Q. De Smedt, H. Wannous, and J.-P. Vandeborre, "Skeleton-Based Dynamic Hand Gesture Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1206–1214, 2016.

[39] R. Nogales and M. E. Benalcázar, "A Survey on Hand Gesture Recognition Using Machine Learning and Infrared Information," in *Applied Technologies* (M. Botto-Tobar, M. Zambrano Vizuete, P. Torres-Carrión, S. Montes León,

G. Pizarro Vásquez, and B. Durakovic, eds.), (Cham), pp. 297–311, Springer International Publishing, 2020.

[40] X. Chen, H. Guo, G. Wang, and L. Zhang, "Motion feature augmented recurrent neural network for skeleton-based dynamic hand gesture recognition," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 2881–2885, 2017.

[41] G. Devineau, F. Moutarde, W. Xi, and J. Yang, "Deep Learning for Hand Gesture Recognition on Skeletal Data," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pp. 106–113, 2018.

[42] Y. Chen, L. Zhao, X. Peng, J. Yuan, and D. N. Metaxas, "Construct Dynamic Graphs for Hand Gesture Recognition via Spatial-Temporal Attention," in *BMVC*, 2019.

[43] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[44] H. H. Publishing, "5 exercises to improve hand mobility." `https://www.health.harvard.edu/pain/5-exercises-to-improve-hand-mobility`. Accessed: 2023-05-05.

[45] Ultraleap, "Leap Motion Controller." `https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf`. Accessed: 2023-05-08.

[46] R. J. Saunders, R. Astifidis, S. L. Burke, J. Higgins, and M. A. McClinton, *Hand and Upper Extremity Rehabilitation: A Practical Guide*. Churchill Livingstone, 4th ed., 2015.

[47] J. Hunter, E. Mackin, A. Callahan, T. Skirven, L. Schneider, and A. Osterman, *Rehabilitation of the Hand and Upper Extremity*. Elsevier, 7th ed., 2016.

[48] D. U. Jette, K. Bacon, C. Batty, M. Carlson, A. Ferland, R. D. Hemingway, J. C. Hill, L. Ogilvie, and D. Volk, "Evidence-based practice: beliefs, attitudes, knowledge, and behaviors of physical therapists," *Phys Ther*, vol. 83, no. 9, pp. 786–805, 2003.

[49] J. C. MacDermid, "An introduction to evidence-based practice for hand therapists," *J Hand Ther*, vol. 17, no. 2, pp. 105–117, 2004.

[50] C. D. Takahashi, L. Der-Yeghiaian, V. Le, R. R. Motiwala, and S. C. Cramer, "Robot-based hand motor therapy after stroke," *Brain*, vol. 131, no. 2, pp. 425–437, 2007.

[51] M. Sivan, J. Gallagher, S. Makower, D. Keeling, B. Bhakta, R. J. O'Connor, and M. Levesley, "Home-based Computer Assisted Arm Rehabilitation (hCAAR) robotic device for upper limb exercise after stroke: results of a feasibility study in home setting," *J Neuroeng Rehabil*, vol. 11, p. 163, 2014.

[52] M. A. Vélez-Guerrero, M. Callejas-Cuervo, and S. Mazzoleni, "Artificial Intelligence-Based Wearable Robotic Exoskeletons for Upper Limb Rehabilitation: A Review," *Sensors*, vol. 21, no. 6, 2021.

[53] L. Kozden, T. Pritchett, N. Tyagi, and C. F. D. Leochico, "Chapter 21 - Telerehabilitation for Hand and Upper Extremity Conditions," in *Telerehabilitation: Principles and Practice* (M. Alexander, ed.), pp. 309–317, Elsevier, 2022.

[54] S. Rosewilliam, C. A. Roskell, and A. D. Pandyan, "A systematic review and synthesis of the quantitative and qualitative evidence behind patient-centred goal setting in stroke rehabilitation," *Clin Rehabil*, vol. 25, no. 6, pp. 501–514, 2011.

[55] C. Rathert, M. D. Wyrwich, and S. A. Boren, "Patient-Centered Care and Outcomes: A Systematic Review of the Literature," *Medical Care Research and Review*, vol. 70, no. 4, pp. 351–379, 2013.

[56] M. M. Damaneh, F. Mohanna, and P. Jafari, "Static hand gesture recognition in sign language based on convolutional neural network with feature extraction method using ORB descriptor and Gabor filter," *Expert Systems with Applications*, vol. 211, p. 118559, 2023.

[57] J. Stenum, K. M. Cherry-Allen, C. O. Pyles, R. D. Reetzke, M. F. Vignos, and R. T. Roemmich, "Applications of Pose Estimation in Human Health and Performance across the Lifespan," *Sensors*, vol. 21, no. 21, 2021.

[58] M.-Y. Wu, P.-W. Ting, Y.-H. Tang, E.-T. Chou, and L.-C. Fu, "Hand pose estimation in object-interaction based on deep learning for virtual reality applications," *Journal of Visual Communication and Image Representation*, vol. 70, p. 102802, 2020.

[59] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, "BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2605–2613, 2017.

[60] S. Goldin-Meadow and M. W. Alibali, "Gesture's role in speaking, learning, and creating language," *Annu Rev Psychol*, vol. 64, pp. 257–283, 2012.

[61] P. Wagner, Z. Malisz, and S. Kopp, "Gesture and speech in interaction: An overview," *Speech Communication*, vol. 57, pp. 209–232, 2014.

[62] T. Wang, Y. Li, J. Hu, A. Khan, L. Liu, C. Li, A. Hashmi, and M. Ran, "A Survey on Vision-Based Hand Gesture Recognition," in *Smart Multimedia* (A. Basu and S. Berretti, eds.), pp. 219–231, Springer International Publishing, 2018.

[63] A. Choudhury, A. K. Talukdar, and K. K. Sarma, "A Review on Vision-Based Hand Gesture Recognition and Applications," in *Advances in Systems Analysis, Software Engineering, and High Performance Computing*, pp. 256–281, IGI Global, 2015.

[64] M. Yasen and S. Jusoh, "A systematic review on hand gesture recognition techniques, challenges and applications," *PeerJ Comput Sci*, vol. 5, p. e218, 2019.

[65] S. Zhao, H. Cai, W. Li, Y. Liu, and C. Liu, "Hand Gesture Recognition on a Resource-Limited Interactive Wristband," *Sensors*, vol. 21, no. 17, 2021.

[66] M. L. Boas, *Mathematical Methods in the Physical Sciences*. Wiley, 3rd ed., 2005.

[67] D. C. Lay, S. R. Lay, and J. J. McDonald, *Linear Algebra and Its Applications*. Pearson, 5th ed., 2019.

[68] J. Stewart, *Calculus: Early Transcendentals*. Wiley, 12th ed., 2019.

[69] D. J. Griffiths, *Introduction to Electrodynamics*. Cambridge University Press, 4th ed., 2017.

[70] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*. Pearson, 3rd ed., 2013.

[71] S. Bubeck, "Convex Optimization: Algorithms and Complexity," *Found. Trends Mach. Learn.*, vol. 8, no. 34, pp. 231–357, 2015.

[72] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[73] R. Hartshorne, *Geometry: Euclid and Beyond*. Springer, 2000.

[74] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson, 2nd ed., 2021.

[75] A. Agresti, *Categorical Data Analysis*. Wiley, 3rd ed., 2013.

[76] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[77] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[78] S. Haykin, *Neural Networks and Learning Machines*. Pearson, 3rd ed., 2009.

[79] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[80] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

[81] L. Miranda, T. Vieira, D. Martinez, T. Lewiner, A. W. Vieira, and M. F. M. Campos, "Real-Time Gesture Recognition from Depth Data through Key Poses Learning and Decision Forests," in *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pp. 268–275, 2012.

[82] Ultraleap, "Leap Motion Hand Tracking Software." `https://developer.leapmotion.com/tracking-software-download`, 2023. Version 5.7.2. Accessed: 2023-05-11.

[83] N. inform, "Exercises for wrist, hand and finger problems." `https://www.nhsinform.scot/illnesses-and-conditions/muscle-bone-and-joints/exercises/exercises-for-wrist-hand-and-finger-problems`. Accessed: 2023-05-15.

[84] F. Rehab, "39 Restorative, Strengthening Hand Therapy Exercises to Try at Home." `https://www.flintrehab.com/hand-therapy-exercises`. Accessed: 2023-05-05.

[85] L. Lamoreaux and M. M. Hoffer, "The effect of wrist deviation on grip and pinch strength," *Clin Orthop Relat Res*, no. 314, pp. 152–155, 1995.

[86] D. Kristen Gasnick, PT, "Hand and finger exercises to ease arthritis pain." `https://www.verywellhealth.com/finger-exercises-5120395`. Accessed: 2023-05-15.

[87] K. Morikawa, A. Matsumoto, S. Patki, B. Grundlehner, A. Verwegen, J. Xu, S. Mitra, and J. Fenders, "Compact wireless EEG system with active electrodes

for daily healthcare monitoring," in *2013 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 204–205, IEEE, 2013.

[88] UltraLeap, "LeapC." `https://docs.ultraleap.com/tracking-api/leapc-guide`. Accessed: 2023-05-01.

[89] UltraLeap, "Leap Motion Python SDK Documentation." `https://developer-archive.leapmotion.com/documentation/python`. Accessed: 2023-05-01.

[90] I. Kitware, "CMake Documentation." `https://cmake.org/documentation`. Accessed: 2023-05-01.

[91] S. Developers, "SWIG Documentation." `https://www.swig.org/doc.html`. Accessed: 2023-05-01.

[92] Leap Motion, "LeapCxx." `https://github.com/leapmotion/LeapCxx`, 2018.

[93] Sean Schneeweiss, "RoseMotion." `https://github.com/seanschneeweiss/RoSeMotion`, 2022.

[94] Pygame, "Pygame Documentation." `https://www.pygame.org/docs/`. Accessed: 2023-05-11.

[95] Google Inc., "Chromium Dinosaur Game," 2014. Accessed: 2023-05-05.

[96] PyInstaller, "PyInstaller Manual." `https://pyinstaller.org/en/stable/`. Accessed: 2023-05-11.

[97] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[98] J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation." `https://machinelearningmastery.com/k-fold-cross-validation/`, 2018. Accessed: 2023-05-05.

[99] Scikit-learn contributors, "Scikit-learn - Tuning the hyperparameters of an estimator." `https://scikit-learn.org/stable/modules/grid_search`. Accessed: 2023-05-11.

[100] I. Sommerville, *Software Engineering*. Pearson, 10th ed., 2015.

Appendix A

Assessment on the Working Process

During this project, all team members made great contributions and were always active in brainstorming new ideas, as well as finishing the deadlines on time. Our team regularly met up with the supervisor to update the progress and ask for advises to improve our work.

Regarding the workload of members, all members contributed equally to all parts of the project. This can ensure that every members have a good understanding on the project. The tasks were assigned based on the strengths and weaknesses of each member. This makes the working process become more productive and effective.

Appendix B

Workload of Members

Task	Assignee			Start	Due	Status
	Khang	Phat	Sang			
1. Plan workload for each member.	■	□	□	Jan 9	Jan 15	Done
2. Define problem and conduct literature review.	■	■	■	Jan 16	Feb 26	Done
3. Prepare theoretical background.	■	■	■	Jan 16	Feb 26	Done
4. Research and work with Leap Motion Controller.	■	■	□	Feb 27	Mar 10	Done
5. Plan and participate in data collection.	■	■	■	Mar 13	Mar 26	Done
6. Implement hand gesture recognition system.	■	■	■	Mar 27	Apr 5	Done
7. Develop interactive games for rehabilitation.	□	■	■	Mar 27	Apr 16	Done
8. Prepare and conduct user study.	■	□	■	Apr 17	Apr 24	Done
9. Perform experiments and evaluate results.	■	■	■	Apr 10	May 1	Done
10. Prepare materials for presentation (report, slides).	■	■	■	May 1	May 14	Done

■ Main person in charge

□ Supporter & Reviewer

Figure B.1: Assignment schedule. The workload of each member in the project is presented.

Appendix C

Research Paper

A research paper we submitted to the 12th OISP Science and Technology Symposium for Students.

Hand gesture recognition for game-based hand rehabilitation
(Nhận diện cử chỉ tay cho việc chơi trò chơi hỗ trợ phục hồi chức năng tay)

Ho Tri Khang[1,3,4]*, Tran Tien Phat[1,3,4], Vo Ngoc Sang[1,3,4] , Nguyen Ngoc Thanh Xuan[1,3,4], Le Gia Phat[1,2,4], Quan Thanh Tho[1,4]

[1] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam

[2] School of Industrial Management, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam

[3] Office for International Study Programs, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam

[4] Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam

*Corresponding author: khang.ho.0@hcmut.edu.vn

Abstract

Hand rehabilitation is an influential aspect of post-injury recovery for many patients. During the hand rehabilitation process, the patients are required to practice repetitive movements to regain the loss of hand functions. Traditional rehabilitation methods can be tedious and unengaging, leading to poor adherence to treatment plans. Video games have emerged as a potential tool to make rehabilitation more engaging and effective. Moreover, the rapid growth of Artificial Intelligence and Machine Learning recently has led to the idea of applying hand gesture recognition in game-based hand rehabilitation. By playing interactive games, the patients will feel more enjoyable and motivated, thus encouraging the process of recovery. In this work, we propose a real-time skeleton-based hand gesture recognition for game-based rehabilitation. The proposed system aims to support and encourage patients in practicing hand rehabilitation exercises via interesting rehabilitation games. Our system makes use of a Leap Motion controller to extract 3D hand skeletons from the user's hand. A pose-based recognizer will identify the key hand poses from hand skeletal data to recognize the hand gesture. For the system to effectively support the hand rehabilitation process, a set of suitable hand gestures selected from common wrist and finger rehabilitation exercises will be adopted. When a gesture is recognized, the corresponding action will be sent to the game environment to control the game. We also propose three simple, interactive video games designed specifically for hand rehabilitation. Playing these games involves repetitively performing hand rehabilitation exercises, which can help to speed up the recovery process. Our system provides an engaging and interactive way for patients to perform their rehabilitation exercises while also enjoying the benefits of playing video games.

Keywords: *hand rehabilitation, hand gesture recognition, Machine Learning, interactive video games*

1. Introduction

According to the World Stroke Organization (WSO) [1] and the Global Burden of Disease (GBD) Study 2019 [2], stroke is the second-leading cause of death globally, accounting for 11.6% of total deaths. It is also the third-leading cause of death and disability combined, accounting for 5.7% of total DALYs (Disability-Adjusted Life Years). Stroke survivors often experience hand and upper extremity impairment, which significantly impedes their ability to carry out daily, work, and leisure activities. Therefore, post-stroke hand rehabilitation is essential to restore the strength, mobility, and coordination of the patient's hands.

Post-stroke rehabilitation, however, is a challenging process, since it can be difficult, intensive, and lengthy, depending on the severity of the stroke and which parts of the brain were damaged. Therefore, the success of the recovery process depends on how well patients can commit to the treatment plan. Traditionally, stroke patients follow a rehabilitation method that involves repetitive hand exercises under the guidance of a hand therapist. However, this approach can be tedious and unengaging due to its repetitive nature, leading to the poor adherence of patients to the treatment plan. Therefore, it is crucial to develop an efficient and interesting rehabilitation method that can engage patients and motivate them to follow the treatment plan.

In recent years, there is a growing interest in developing game-based approaches for hand rehabilitation [3], [4]. By adopting modern technologies such as Virtual Reality [5], [6], Augmented Reality [7] and advanced sensors for hand motion tracking such as Kinect [8], and Leap Motion [9], game-based approaches are useful for creating an interactive environment where patients can enjoy the games while still participating in the hand function recovery process. This is done by performing repetitive hand movements while playing the games. Therefore, game-based rehabilitation is more enjoyable and interactive, enhancing patient engagement and participation in the treatment plan.

In this work, we propose a real-time, skeleton-based hand gesture recognition system for game-based hand rehabilitation. Our system uses a Leap Motion Controller to capture hand movement and extract hand skeletal data. We also developed a gaming application consisting of 3 simple interactive games designed specifically for hand rehabilitation. Interacting with these games involves repetitively performing hand gestures, which simulates the common exercises used in hand rehabilitation. Therefore,

patients can enjoy the games while still participating in the recovery process. Compared to previous work on hand gesture recognition in hand rehabilitation, our approach is simpler, faster and is more practical by adopting the set of hand gestures from common hand therapy exercises. We also conducted a user survey with 10 individuals to evaluate the effectiveness of the system on real users. The result shows that the proposed system is enjoyable and can be a potential solution to hand rehabilitation.

2. Methods and Implementation

2.1 Overview

We propose a real-time, skeleton-based approach that recognizes hand gestures based on the idea of key pose identification. The workflow of the system is summarized in Figure 1. Generally, the system workflow contains three main stages:

- **Skeleton extraction**: In this stage, the Leap Motion Controller extracts hand skeletal information from the user's hand and sends the data to the next stage to recognize hand gestures.
- **Hand pose identification**: Important features are extracted from hand skeletal data and a trained Support Vector Machine (SVM) will identify hand poses.
- **Gesture recognition from key poses**: The system reads the sequence of identified hand poses and checks if the sequence matches with any valid key pose sequences to construct hand gestures. If there is a hand gesture found, the system will output the result and the corresponding action will be sent to the game environment.

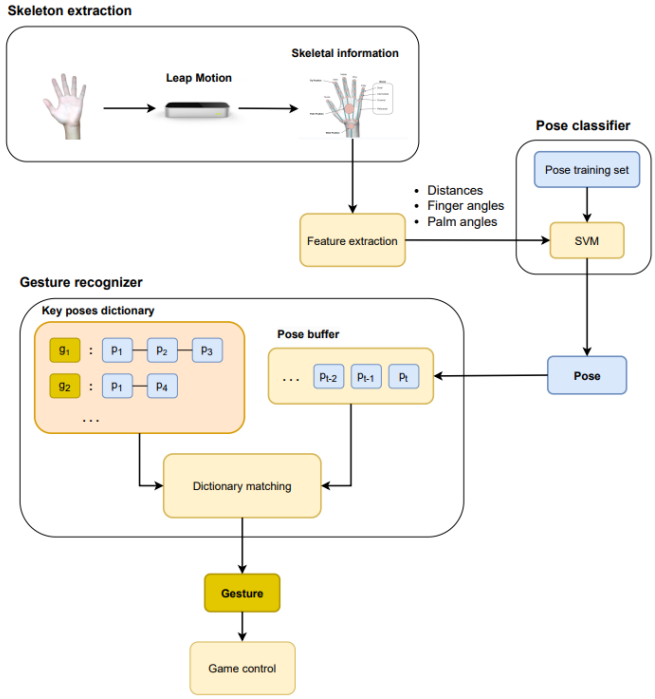


Fig 1. Overview of the proposed system. Our system extracts features from Leap Motion data and recognizes the gesture based on key pose identification.

We adopted 9 hand gestures, all of which were selected from common hand, wrist and finger rehabilitation exercises [10], [11] that help patients to recover their hand mobility, strength and flexibility. All of the gestures can be performed by one hand, either the left hand or the right hand. The list of gestures and their equivalent hand exercises are shown in Table 1.

Table 1. List of hand gestures and their corresponding hand exercises.

Gesture	Exercises
Move left/right	Wrist ulnar/radial deviation
Move up/down	Wrist extension/flexion
Rotate left/right	Wrist supination/pronation
Close fist	Close fist
Stop	Spreading and closing fingers
Thumb in	Thumb stretch

2.2 Datasets Collection

We collected two datasets, a pose dataset to train and evaluate the hand pose classification model, and a gesture dataset to evaluate the proposed hand gesture recognition approach.

Leap Motion Data: The hand skeletal data extracted by the Leap Motion Controller consists of 3D coordinates of 27 distinct hand joints and hand elements from the user's hand. Each 3D coordinate is a tuple of three values x, y, and z representing the position of an object in the Leap Motion's coordinate system. The Leap Motion's coordinate system sees objects above it in units of real-world millimetres. The origin is located at the top, centre of the hardware. The x-axis and the z-axis lie on the horizontal plane and the x-axis runs parallel to the long edge of the device. The y-axis is a vertical axis perpendicular to the horizontal plane, representing the height from the object to the origin of the device. The skeletal model and the Leap Motion's coordinate system are shown in Figure 2.

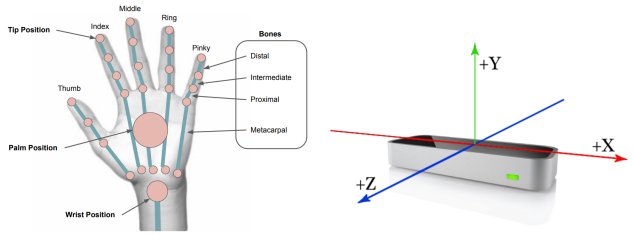


Fig 2. The Leap Motion's skeletal model (left) and the Leap Motion's coordinate system (Right).

Pose Dataset: The pose dataset consists of 12 distinct hand poses that are identified by the system (Figure 3). These poses are the main components that are necessary to construct hand gestures. The dataset was collected by six subjects, four men and two women. For a robust evaluation, the training set was collected from four subjects while the test set was collected from the other two. Each sample contains the hand skeletal information captured by the Leap Motion Controller in each frame. The samples are collected by both left and right hands, and from different angles and positions relative to the device. In total, there are 4396 training samples and 1928 test samples.

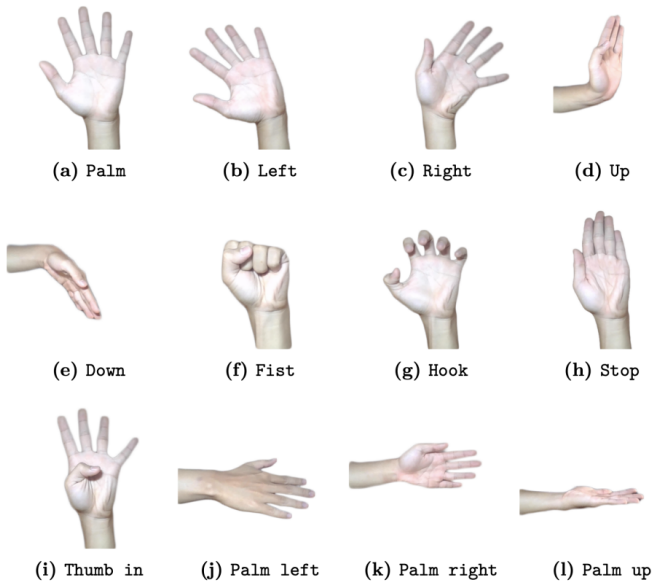


Fig 3. The visual representation of the 12 selected key poses.

Gesture Dataset: The gesture dataset was generated by 5 subjects, each generating 7 to 10 samples per gesture, using both their left and right hands. Each sample represents one gesture and consists of the sequence of hand-skeletal data recorded for each frame during that gesture. In total, there are 443 samples of 9 hand gestures that are recognized by the system (Table 1).

2.3 Hand Pose Classification

Feature Extraction: From the skeletal data captured by the Leap Motion Controller, we perform feature extraction to extract important features that will be used for pose identification. We first extract 13 local features, including the Euclidean distances between the palm center and the five fingertips, the distances between adjacent fingertips and the angles between adjacent fingers. Such local features not only extract useful hand information but also acts as a normalizer that helps remove the noise effects of varying hand positions and angles relative to the Leap Motion's sensors.

Let P denote the point representing the palm centre within the Leap Motion's coordinate system. The five fingertips, denoted as F_i where i in $[1, 5]$, correspond to the thumb, index finger, middle finger, ring finger, and pinky finger, respectively. The distance-based features can be calculated as follow:

$$D_i = ||F_i - P||,\ i \in [1,5]$$
$$D_{F_i} = ||F_{i+1} - F_i||,\ i \in [1,4]$$

where D_i represents the Euclidean distances between the palm centre and the five fingertips, and D_{F_i} represents the distances between every pair of adjacent fingertips.

The angles between adjacent fingers will be

$$\alpha_i = arccos\left(\frac{(F_i - P)\cdot(F_{i+1} - P)}{||F_i - P||\cdot||F_{i+1} - F_i||}\right), i \in [1,4]$$

where α_i represents the angles (in radians) between every pair of adjacent fingers.

Next we extracted three global features, including the yaw, pitch, and roll angles of the hand. These features represent the global movement of the user's hand relative to the Leap Motion Controller. After feature extraction, the final feature vector V consists of 16 distinct features:

$$V=(D_1 \dots D_5, D_{F_1} \dots D_{F_4}, \alpha_1 \dots \alpha_4, yaw, pitch, roll)$$

This feature vector is standardized and fed to the pose classifier for hand pose classification.

Pose Classification:

In this stage, the feature vector is classified as one of the hand poses that were introduced in Figure 3. Several machine learning algorithms were tested. We found that Support Vector Machine (SVM) with radial basis function (rbf) kernel demonstrated the highest accuracy of **96.84%** on our pose dataset (see Section 3 for more detail). Therefore, we chose SVM with the rbf kernel as the pose classifier of the system.

2.4 Gesture Recognition from Key Poses

In our setting, a gesture g is represented as a sequence of key poses $g = (p_1, p_2, \dots, p_{g_n})$, where p_i belongs to a finite set of key poses p. Most of the gestures are composed of 2 to 3 key poses and can have different combinations of key poses. We built a key pose dictionary that contains all the mappings from gestures to key pose sequences. Each record in the dictionary is a pair of key-value (k, v) where k is a hand gesture and v is the list containing all possible key pose sequences that can be used to build up that gesture. To ensure the generalization of our approach, 6 subjects were asked to perform each gesture in several ways. We then captured the sequences of poses and manually inserted all possible key pose sequences into the key pose dictionary. The final key pose dictionary is shown in Table 2.

Table 2. Key pose dictionary for gesture recognition. Each gesture can be defined by a finite sequence of key poses.

Gesture	ID	Key Pose Sequences
Move left	g0	(p0, p1), (p7, p1)
Move right	g1	(p0, p2), (p7, p2)
Move up	g2	(p0, p3), (p7, p3)
Move down	g3	(p0, p4), (p7, p4)
Rotate left	g4	(p0, p10, p11), (p7, p10, p11)
Rotate right	g5	(p0, p9, p11), (p7, p9, p11)
Close fist	g6	(p0, p5), (p0, p6, p5), (p0, p8, p6, p5)
Stop	g7	(p0, p7, p0)
Thumb in	g8	(p0, p8, p0), (p7, p8, p7)

When a hand pose pt is identified by the pose classifier at time t, it will be inserted into a pose buffer β. The gesture recognizer will then read the sequence of all identified poses $s = (p_{t-n}, \dots, p_{t-1}, p_t)$ from the buffer and search from the key pose dictionary to see if there is any key pose sequence $x = (p_1, \dots, p_{g_n})$ contained in s. If the match is found, the gesture g composed of the sequence x will be recognized. The buffer will then be cleared and ready to store the next

hand pose. If no match is found, nothing will be recognized. The buffer will continuously store newly identified hand poses until a match is found. More details of the gesture recognition process are described in Algorithm 1.

Fig.4. Shapes and Colors

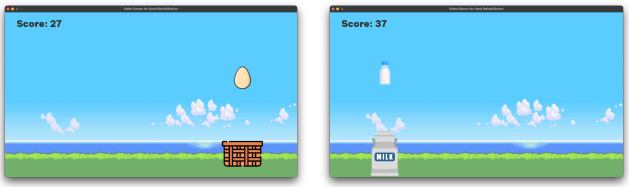


Fig 5. Eggs and Milk

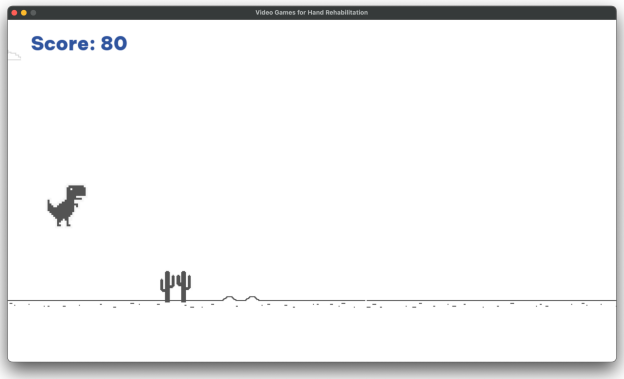


Fig. 6: Dino Run

2.5 Interactive Games for Hand Rehabilitation

We developed 3 interactive games for hand rehabilitation: Shapes and Colors, Eggs and Milk and Dino Run, as shown in Figure 4, Figure 5 and Figure 6, respectively. These three games were developed using Python 3.9.0 and PyGame 2.3.0. In the first game, Shapes and Colors, the player needs to select the correct shape and colour according to the question prompted on the screen. In the second game, Eggs and Milk, the player needs to move the container to catch the falling object from the top of the screen. Finally, the Dino Run game requires controlling the dinosaur to avoid as many obstacles as possible. In all of the three games, the player has to perform hand gestures to interact with the application. Therefore, the application helps improve hand mobility and flexibility and guarantees a positive effect on hand rehabilitation.

3. Results and Discussion

3.1 Hand Gesture Recognition

We performed some experiments to evaluate the performance of the system on our pose and gesture datasets. All experiments were conducted on an ASUS laptop equipped with an Intel Core i5-8250U 1.6GHz CPU and 12GB RAM.

Table 3. Evaluation results on the pose test set of five classification models. * : The precision, recall and F1 score is the weighted average from all classes.

Model	Accuracy (%)	Precision*	Recall*	F1 score*
SVM	**96.84**	0.9698	0.9684	**0.9684**
MLP	**96.47**	0.9686	0.9673	**0.9675**
Random Forest	91.34	0.9300	0.9134	0.9145
Logistic Regression	82.21	0.8336	0.8221	0.8211
k-NN	92.21	0.9323	0.9222	0.9225

To find out the best pose classifiers, we trained 5 different machine learning models, including Support Vector Machine (SVM), Multilayer Perceptron (MLP), Random Forest, Logistic Regression and *k*-Nearest Neighbors (*k*-NN) on our pose training set. During the training process, we used k-fold cross-validation with k = 10 and applied Grid Search [12] to find the best hyperparameter settings for each model. After that, we evaluated the performance of these models on the pose test set. From the result (Table 3), we found that SVM has the highest test set accuracy (96.84%), so we chose SVM as the pose classifier of the system.

The precision, recall and F1 score on each hand pose class using SVM are depicted in Table 4. Our pose classifier achieves outstanding results on all of the classes (all larger than 0.9). Some of the classes have maximum precision (Right), recall (Stop) and F1 score (Thumb_in).

Table 4. The precision, recall and F1 score on each hand pose class using the SVM model.

Pose	Precision	Recall	F1 score
Palm	**0.9130**	0.9492	**0.9307**
Left	0.9218	0.9593	0.9492
Right	**1.0000**	0.9722	0.9859
Up	0.9939	0.9760	0.9849
Down	0.9803	**0.8922**	0.9342
Fist	**1.0000**	0.9897	0.9948
Hook	0.9867	0.9867	0.9867
Stop	0.9321	**1.0000**	0.9649
Thumb_in	**1.0000**	**1.0000**	**1.0000**

Palm_left	**0.9044**	**1.0000**	0.9498
Palm_right	0.9916	**1.0000**	0.9958
Palm_up	**1.0000**	**0.9060**	0.9507

We also studied the effect of different feature extraction settings on hand pose classification using SVM and MLP (Table 5). We found that Distances (distance-based features) and Palm angles (Yaw, pitch, and roll angles) are the two most important groups of features, omitting one of which can lead to a significant drop in classification accuracy.

Table 5. Performance of pose classifier on different feature extraction settings.

Feature extraction setting	Accuracy	
	SVM (rbf)	MLP (100,)
Distances + Palm angles + Finger angles	96.84	96.47
Distances + Finger angles	66.8	69.5
Palm angles + Finger angles	75.99	79.72
Distances + Palm angles	96.68	95.38
Raw skeleton data	85.32	84.96

After finding the best hand pose classifier, we evaluated our pose-based gesture recognition approach on the gesture dataset. The precision, recall and F1 score on each gesture class are shown in Table 6. We achieved an accuracy of 94.46% on the gesture dataset and obtained impressive results for many gesture classes. However, there are still many false positive cases in which the system recognizes non-existent hand gestures or misclassifies a gesture as another one.

Table 6. Precision, recall and F1 scores on the gesture dataset.

Gesture	Precision	Recall	F1 score
Move left	1.0000	0.9792	0.9895
Move right	0.9245	0.9800	0.9514
Move up	0.8909	0.9800	0.9333
Move down	0.7539	1.0000	0.8596
Rotate left	1.0000	0.9750	0.9873
Rotate right	1.0000	**0.8684**	0.9296
Close fist	0.9259	1.0000	0.9615
Stop	**0.8444**	1.0000	0.9157
Thumb in	0.9250	0.9250	0.9250
Weighted average	0.9159	0.9702	0.9393

3.2 Time Performance

We also evaluated the time performance of the system by calculating the average elapsed time per frame of the system. The result is shown in Table VII. Our system relies on the hand skeleton tracking process from Leap Motion, which operates consistently at 16.67ms (60FPS) in our experiments. Meanwhile, the execution time per frame for hand gesture recognition (1.2 ms) is significantly lower than the time required for skeleton extraction (16.67 ms). Therefore, the system's overall execution time remains within 16.67 ms (60 FPS), meaning that the system can achieve real-time performance.

Table 7. Execution time and frame per second of our system.

Skeleton Extraction by LMC		Hand Gesture Recognition		Overall	
Time	FPS	Time	FPS	Time	FPS
16.67	60	1.2	833	16.67	60

3.3. User Study

We conducted a user study with 10 individuals to study the effectiveness of our system on real users. The study was conducted in both a residential area and a private home. There are two main stages in our study, a pre-survey study and a post-survey study.

In the pre-survey stage, we collected background information of the 10 participants, including their age, sex and general experiences to hand rehabilitation. The pre-survey results are shown in Table 8. From the result, it can be seen that the majority of participants are elderly (70% of them are over 60 years old). Two of them have experienced hand therapy before and three of them have played hand-interactive games before.

Table 8. Demographic characteristics of the study population.

Variable		n
Age (years)	15-30	1
	30-45	1
	45-60	1
	> 60	7
Gender	Male	5
	Female	5
Have experienced physical therapy?	Yes	2
	No	8
Have played hand-interactive games?	Yes	3
	No	7
Gaming frequency per week?	0	6
	1-5	2
	5-10	1
	> 10	1

In the post-survey stage, we collected their feedback after they had tried our solution. The pose-survey result is shown in Figure 7. Overall, all participants expressed favourable feedback, finding the solution compact, enjoyable, and easy

to use. Aesthetic design is crucial in motivating individuals to use the game regularly, and all participants found our application aesthetically pleasing. And most importantly, all the participants found our application is meaningful and necessary for hand rehabilitation.

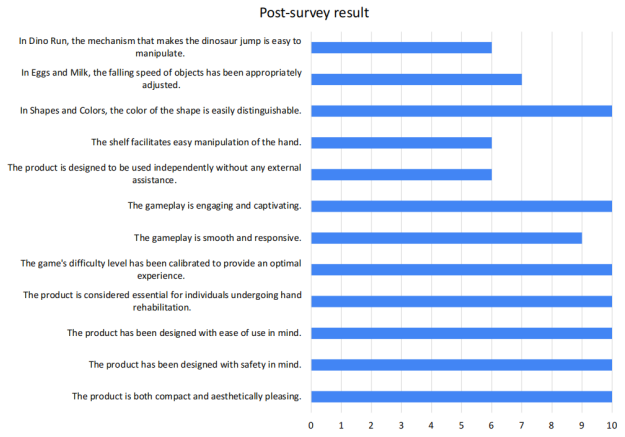


Fig. 7. Post-survey results.

4. Conclusions

In this work, we have implemented a real-time, skeleton-based hand gesture recognition system for hand rehabilitation and achieved remarkable results on our two self-generated datasets (96.84% on the pose dataset and 94.46% on the gesture dataset). We also developed a gaming application for hand rehabilitation and conducted a user study to collect feedback from real users. Most of the users agree that our system is interesting and is a potential solution for hand rehabilitation.

However, there is still some room for improvement in our work. Firstly, our self-generated datasets are small and our user study is still limited to only 10 individuals, which means the system might not be generalized enough to new data, and the collected feedback from the users may still be biased. Secondly, the pose-based gesture recognition approach can face some robustness issues when there are some errors in hand pose classification, and the design using hand poses may not be a good choice for more complex hand gestures which require exact hand movement from the user. These limitations will be addressed in our future work.

Acknowledgement: This research is funded by Office for International Study Programs (OISP), Ho Chi Minh City University of Technology (HCMUT), VNU-HCM under grant number **OSTS-2023-KH&KTMT-01**. We acknowledge the support of time and facilities from HCMUT, VNU-HCM for this study.

5. References

[1] V. L. Feigin, M. Brainin, B. Norrving, S. Martins, R. L. Sacco, W. Hacke, M. Fisher, J. Pandian, and P. Lindsay, "World Stroke Organization (WSO): Global Stroke Fact Sheet 2022," International Journal of Stroke, vol. 17, no. 1, pp. 18–29, 2022.

[2] GBD 2019 Stroke Collaborators, "Global, regional, and national burden of stroke and its risk factors, 1990-2019: a systematic analysis for the Global Burden of Disease Study 2019," Lancet Neurol, vol. 20, no. 10, pp. 795–820, 2021.

[3] I. Ayed, A. Ghazel, A. Jaume-i Capo, G. Moy ´ a Alcover, J. Varona, and ` P. Martinez Bueso, "Vision-Based Serious Games and Virtual Reality Systems for Motor Rehabilitation: A Review Geared Toward a Research Methodology," International Journal of Medical Informatics, vol. 131, 2019.

[4] J. Byra and K. Czernicki, "The Effectiveness of Virtual Reality Rehabilitation in Patients with Knee and Hip Osteoarthritis," Journal of Clinical Medicine, vol. 9, no. 8, 2020.

[5] R. M. Al-Whaibi, M. S. Al-Jadid, H. R. ElSerougy, and W. M. Badawy, "Effectiveness of virtual reality-based rehabilitation versus conventional therapy on upper limb motor function of chronic stroke patients: a systematic review and meta-analysis of randomized controlled trials," Physiother Theory Pract, vol. 38, no. 13, pp. 2402–2416, 2021.

[6] K. E. Laver, B. Lange, S. George, J. E. Deutsch, G. Saposnik, and M. Crotty, "Virtual reality for stroke rehabilitation," Cochrane Database Syst Rev, vol. 11, no. 11, p. CD008349, 2017.

[7] E. R. Ram´ırez, R. Petrie, K. Chan, and N. Signal, "A Tangible Interface and Augmented Reality Game for Facilting Sit-to-Stand Exercises for Stroke Rehabilitation," in Proceedings of the 8th International Conference on the '18. Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3277593.3277635

[8] Y. Bouteraa, I. B. Abdallah, and A. M. Elmogy, "Training of Hand Rehabilitation Using Low Cost Exoskeleton and Vision-Based Game Interface," Journal of Intelligent & Robotic Systems, vol. 96, no. 1, pp. 31–47, 2019.

[9] W. Li, C. Hsieh, L. Lin, and W. Chu, "Hand gesture recognition for post-stroke rehabilitation using leap motion," in Proceedings of the 2017 IEEE International Conference on Applied System Innovation, T.-H. Meen, A. Lam, and S. Prior, Eds. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 386–388.

[10] H. H. Publishing, "5 exercises to improve hand mobility," https: //www.health.harvard.edu/pain/5-exercises-to-improve-hand-mobility, accessed: 2023-05-05.

[11] N. inform, "Exercises for wrist, hand and finger problems," https: //www.nhsinform.scot/illnesses-and-conditions/muscle-bone-and-joints/ exercises/exercises-for-wrist-hand-and-finger-problems, accessed: 2023-05-15.

[12] Scikit-learn contributors, "Scikit-learn - Tuning the hyperparameters of an estimator," https://scikit-learn.org/stable/modules/grid search, accessed: 2023-05-11