TRANSPORT AND TELECOMMUNICATION

PROJECT REPORT

Identification of Finger-based Signatures Using Computer Vision Techniques

Supervised by
Prof. Alexander GRAKOVSKI

April 6, 2023

Contents

1 Introduction

1.1 Purpose of the Document

The purpose of this document is to outline the project plan for the development of an application that uses computer vision techniques to identify finger-based signatures. The application will be designed to address the challenge of signing documents in a world where physical contact is discouraged due to the COVID-19 pandemic.

This document will serve as a reference guide for the project team, detailing the project objectives, scope, and context, as well as the group composition and the technologies that will be used. It will also provide a description of the methodology that will be followed to achieve the project objectives and a summary of the expected outcomes.

The intended audience for this document includes the project team members, the supervisor, and any other stakeholders who may be interested in the project's progress and outcomes.

1.2 Objectives

The main objectives of this project are to:

1. Develop an application that uses computer vision techniques to identify a user's finger and capture the depth of the finger for use in creating a digital signature.

2. Implement a solution to recognize and transcribe the user's signature in 3D and 2D.

3. Test the application to ensure it is accurate and reliable in identifying and transcribing finger-based signatures.

4. Develop a user-friendly interface to allow for easy signature creation and storage.

These objectives will be achieved through a series of tasks and subtasks that will be outlined in the methodology section of this document. The successful completion of these objectives will result in an application that is both useful and innovative, providing a new solution for individuals and organizations seeking a secure and contactless way to sign documents.

1.3 Scope

The application will be designed to operate on both desktop and mobile devices with a camera, allowing users to create digital signatures on the go. The application will also include a user-friendly interface for easy signature creation and storage.

The project will focus on the development of the software application and will not involve the creation of any hardware components. The application will be tested to ensure it is accurate and reliable in identifying and transcribing finger-based signatures.

The project will be completed within the allocated time frame and with the resources available to the project team. Any changes to the scope of the project will be evaluated and approved by the project supervisor before implementation.

1.4 Technologies

To implement the proposed solution, we will leverage existing technologies and frameworks that are commonly used in computer vision, image processing, and software development. In this section, we will discuss some of the potential technologies that we could use to implement the different components of our application.

1.4.1 Technologies and frameworks

OpenCV

OpenCV (Open Source Computer Vision) is an open-source library of computer vision algorithms and tools that can be used for image processing, machine learning, and other computer vision applications. OpenCV has a variety of built-in functions that can be used for feature detection, image segmentation, and object recognition, making it an ideal technology for identifying fingers in images captured by the camera.

TensorFlow

TensorFlow is a popular open-source framework for building machine learning models. It provides a variety of tools for training and deploying machine learning models, as well as pre-trained models that can be used out-of-the-box. In our application, we could use TensorFlow to train a machine learning model that can accurately identify finger positions and orientations in images captured by the camera.

PyTorch

PyTorch is an open-source machine learning library that is widely used for building and training deep learning models. It offers a dynamic computational graph that allows for real-time experimentation and a wide range of built-in modules and tools for data loading, model building, and optimization. PyTorch supports both CPU and GPU acceleration and is highly optimized for distributed computing, making it ideal for large-scale applications. It has a large and active community that provides extensive documentation and support, as well as a vast library of pre-trained models and tools.

Unity

Unity is a popular game engine that can be used for developing 3D applications and games. It provides a variety of tools and features for creating 3D environments, rendering graphics, and handling user input, making it an ideal technology for rendering and animating 3D finger-based signatures in real-time.

Qt

QT is a cross-platform framework for building GUI applications. It provides a variety of tools and features for designing and implementing user interfaces, as well as support for multiple platforms and devices. In our application, we could use Qt to develop a user-friendly and intuitive interface for capturing and displaying finger-based signatures.

1.4.2 Advantages and Disadvantages

Each of the above technologies has its own advantages and disadvantages. OpenCV, for example, provides powerful tools for image processing and feature detection, but may require significant effort to integrate into a larger application. TensorFlow, on the other hand, provides pre-trained models and high-level APIs for building machine learning models, but may require more specialized knowledge and resources to train custom models. Unity is a powerful game engine with extensive 3D rendering capabilities, but may not be optimized for real-time performance in non-gaming applications. Finally, Qt provides a robust toolkit for building user interfaces, but may not be as flexible or customizable as other technologies.

1.4.3 Technology Selection

Based on our analysis, we have selected OpenCV and TensorFlow as the primary technologies for implementing our application. OpenCV will be used for finger detection and tracking, while TensorFlow will be used for machine learning-based finger identification. We believe that these technologies provide a good balance of functionality, performance, and ease of integration, and will enable us to achieve our project goals in a timely efficient manner.

In addition to these primary technologies, we will also leverage other tools and frameworks as needed, including Unity for 3D rendering and Qt for user interface development. We will continually evaluate and adjust our technology choices as we progress through the project to ensure that we are meeting our goals and requirements.

Overall, we believe that our technology choices will enable us to develop a robust and effective solution for finger-based signature recognition that can be used in a variety of setti Overall, we believe that our technology choices will enable us to develop a robust and effective solution for finger-based signature recognition that can be used in a variety of settings and applications. By combining the strengths of multiple technologies and frameworks, we are confident that we can deliver a high-quality and innovative product that meets the needs of our users.

2 Work Schedules

In this section, we describe the work schedules for the two main groups within our team: the finger recognition group and the signature capture group. The finger recognition group will focus on developing the algorithms and software necessary to detect and track a user's finger in real time, while the signature capture group will focus on capturing the user's signature using their finger and obtaining depth information to create a 3D representation of the signature.

The overall schedule for the project is as follows:

- Part 1: Research and familiarization with relevant technologies and software libraries.
- Part 2: Prototype finger detection and signature capture software.
- Part 3: Refine software and integrate the two main groups' work.
- Part 4: Testing, debugging, and finalization of the software.

This schedule is subject to change based on the progress of the project and the specific needs of each group. However, we believe that this schedule provides a reasonable timeline for completing the necessary work and achieving our project goals.

3 State of the art

The state of the art concerning air signing with fingers involves the use of computer vision and machine learning techniques to recognize and interpret hand gestures. This technology is typically referred to as computer vision-based sign language recognition, and it relies on complex algorithms to detect and track the movement of hands and fingers in real-time.

Recent advances in deep learning and neural networks have greatly improved the accuracy and speed of sign language recognition systems. These systems can recognize and translate sign language into spoken language, enabling communication between people who are deaf or hard of hearing and those who are not.

In addition, wearable devices such as gloves and bracelets with embedded sensors can capture hand movements.

Overall, the state of the art in air signing with fingers is rapidly advancing, and there is a great deal of potential for this technology to improve the lives of people.

Air finger drawing and tracking to sign typically require some form of technology or device to accurately track and interpret hand movements. While AI may be used in some cases to improve accuracy or recognize specific gestures, it is not necessarily required.

One example of a non-AI-based approach to air finger drawing and tracking to sign is using a motion tracking device such as a Microsoft Kinect or Leap Motion. These devices use infrared cameras to track the movement of your hands and fingers in real-time, allowing you to draw or sign in the air and have your movements translated into digital form.

Another approach is using a touchless interface that can detect the movement of your fingers through sensors or motion detectors. This can include devices like the Myo armband, which uses electromyography sensors to detect muscle movements in your arm and translate them into commands for controlling a computer or device.

It is important to note that while these non-AI-based approaches can be effective, they may not be as accurate or reliable as AI-based methods, especially for more complex gestures or sign language. Additionally, the technology required for these approaches may be less common or more expensive than AI-based alternatives.

There are several traditional methods of hand recognition that do not rely on AI, including:

Template matching: This method involves comparing the shape of a hand to a pre-defined set of templates to determine the most likely match. The templates can be created based on the average shape of a hand or specific hand shapes of interest.

Edge detection: This method involves detecting the edges of a hand in an image and using them to define the shape of the hand. Various edge detection algorithms can be used to accomplish this, including the Canny edge detector and the Sobel operator.

Contour analysis: This method involves extracting the contour of a hand in an image and analyzing its shape and features. Various techniques can be used to extract the contour, including the watershed algorithm, the active contour model, and the level set method.

4 Finger Recognition and drawing with finger

We made a solution to recognise the hand and each finger and then we wanted to add the features to draw following our index.

4.1 Functioning

This code uses OpenCV and MediaPipe to create a program that detects the position of the index finger in the webcam feed and uses it to draw a curve on the screen.

Here's a brief explanation of each module used in the code:

- Cv2: OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision.
- Mediapipe: MediaPipe is a cross-platform framework for building multimodal applied machine learning pipelines.

- Numpy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

4.1.1 How it works

MediaPipe Hands is a hand tracking solution developed by Google as part of the MediaPipe framework. It uses a combination of computer vision techniques and machine learning models to detect and track the positions of individual hand landmarks in real-time video streams.

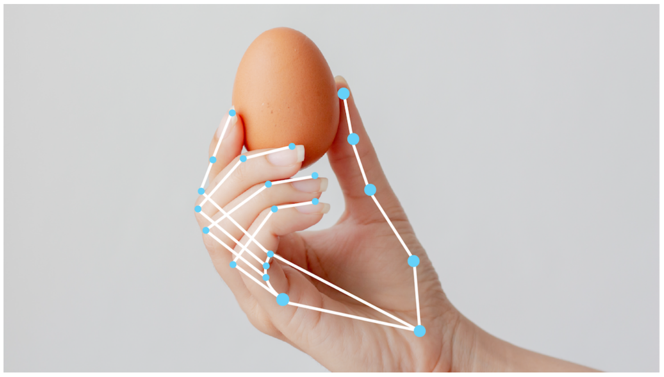

Figure 1: Hand detected with MediaPipe

The MediaPipe Hands pipeline consists of several stages:

- Hand Detection: The first stage involves using a machine learning model to detect whether there is a hand in the input video stream. The model identifies potential hand regions in the video frames and generates a hand detection bounding box.

- Hand Landmark Estimation: The second stage involves using another machine learning model to estimate the 3D positions of 21 key landmarks on the detected hand. These landmarks include finger tips, joints, and the palm center.

- Tracking: The third stage involves using a Kalman filter to track the 3D positions of the hand landmarks over time. The filter predicts the next location of each landmark based on the previous positions and updates the predictions based on the new input data.

- Gesture Recognition: The final stage involves using the tracked hand landmarks to recognize specific hand gestures. For example, it can detect whether the hand is making a fist or an open palm.

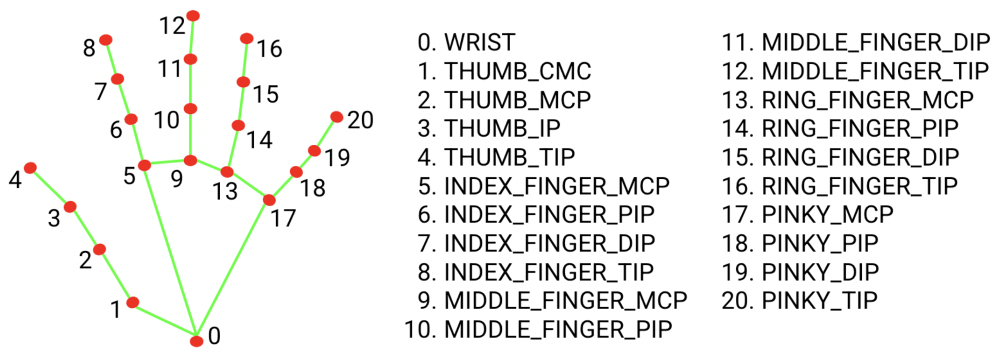

Figure 2: Hand Landmark Estimation

MediaPipe Hands uses a deep neural network architecture called BlazePalm for hand detection and a second architecture called BlazeHand for landmark estimation. Both models are lightweight and optimized for real-time performance on mobile and embedded devices.
Palms can be modelled using only square bounding boxes and therefore reducing the number of anchors by a factor between 3 and 5. Then an encoder-decoder feature extractor for a larger scene-context awareness even for small objects. Lastly, it minimize the focal loss during training to support a large amount of anchors resulting from the high scale variance

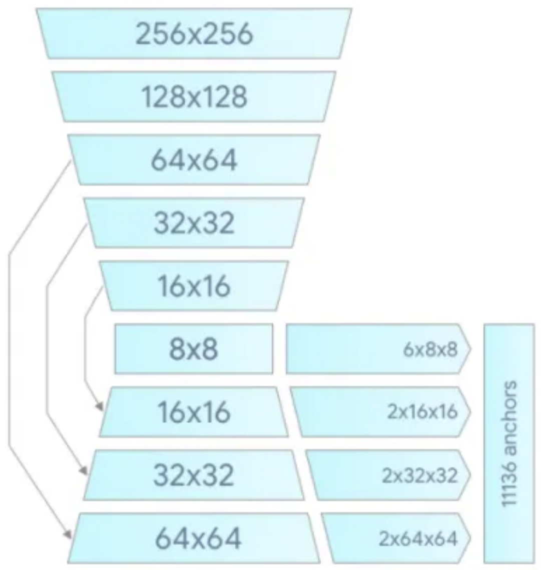

Figure 3: BlazePalm model architecture [6]

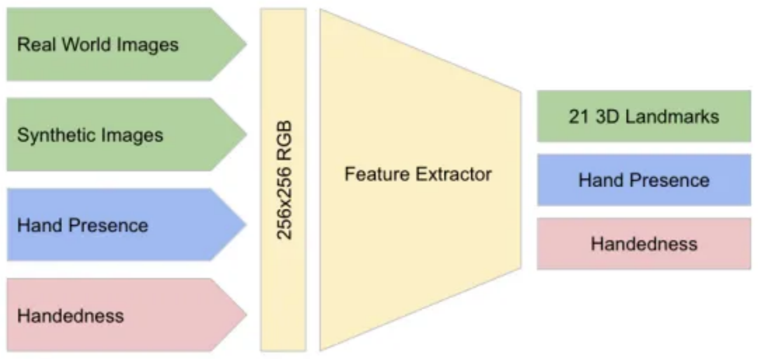

Figure 4: Hand Landmark model architecture [6]

MediaPipe Hands also supports multi-hand tracking, meaning it can track the positions of multiple hands in a video stream. Additionally, it provides various configuration options that allow developers to customize the pipeline for specific use cases and hardware platforms.

Model Variation	Average Precision
No decoder + cross entropy loss	86.22%
Decoder + cross entropy loss	94.07%
Decoder + focal loss	95.7%

Table 1: Model variation on precision. [6]

Overall, MediaPipe Hands is a powerful and efficient solution for hand tracking that can be used in a variety of applications, such as virtual reality, augmented reality, sign language recognition, and gesture-based interfaces.

4.1.2 The process

Here's how our algorithm works:

1. First, we use cv2 to provide computer vision capabilities for video processing. We continuously captures frames from the video stream.
2. Then, we initializes MediaPipe's hands module to detect hand landmarks in the video stream. We then draw the landmarks and the connections between them.
3. If the program detects a hand in the captured image, it gets the landmark coordinates of the index finger that we also mark with a circle to be sure of his position.
4. We then draw a curve on the screen following the index finger and finally get an output image of our drawn signature.

This code can be modified to draw various shapes or lines. With this we can air-sign with our index and see our signature.

5 Depth estimation for writing on and off

The objective is to estimate the depth of the writing hand in order to determine whether the finger movement should be taken as writing or not, in the same way as when you lift the pen from the paper.

5.1 ANN Model - MiDaS

This solution uses Pytorch's MiDaS to generate live depth detection. MiDaS (Mixed Depth Estimation of absolute Scale) is a deep learning-based approach for monocular depth estimation, which means it can estimate the depth of an image from a single 2D image input, without requiring stereo or other depth input.
MiDaS is based on a convolutional neural network (CNN) architecture that has been trained on a large dataset of approximately 2 million images with depth information with more than 70 million parameters. The network takes an input image as input and produces a corresponding depth map, which represents the distance from the camera to each pixel in the image. This depth map can be used for a wide range of computer vision tasks, such as virtual and augmented reality, autonomous driving, robotics, and more.

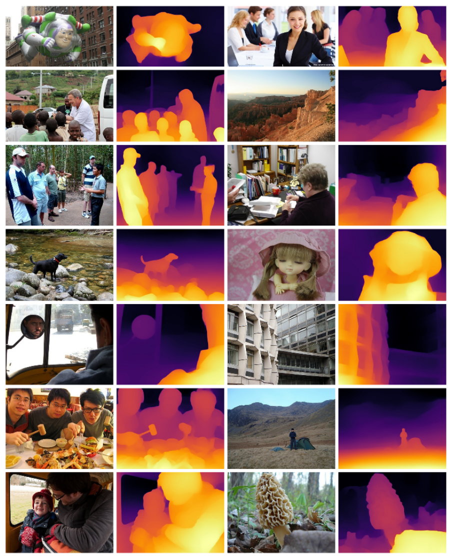

Figure 5: MiDaS Samples

What sets MiDaS apart from other depth estimation techniques is that it uses a mixed-scale approach, where the network learns to estimate depth at multiple scales simultaneously, and combines these estimates to produce a final depth map. This allows MiDaS to capture both large-scale and small-scale depth features in the image, resulting in more accurate depth estimates. MiDaS has a lightweight version, MiDaS Small, that offer the opportunity to compute real-time depth estimation. With only 1 million parameters, that makes it much faster to run on devices with limited computational resources. However, this also reduces the accuracy, as MiDaS Small cannot capture as much fine-grained detail in the depth map as the larger model.

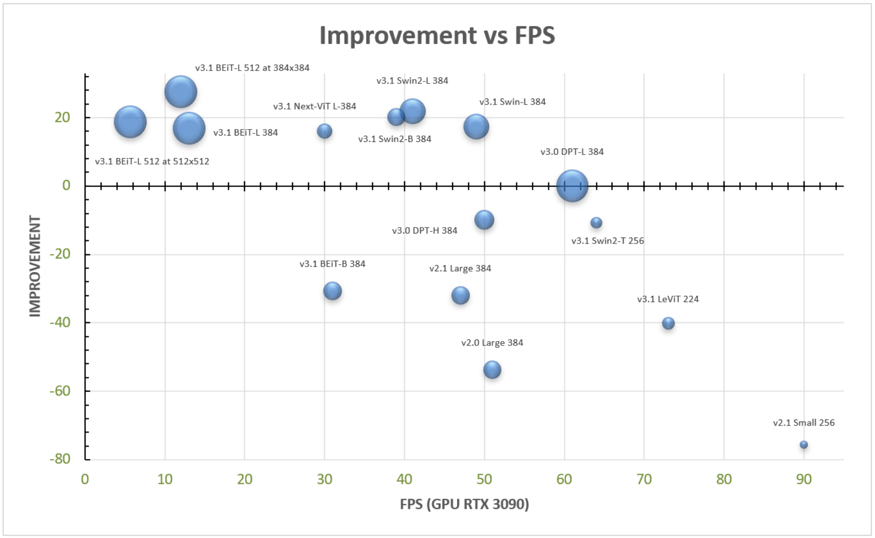

Figure 6: MiDaS version comparison

5.2 Threshold determination

5.2.1 Face depth detection

5.2.2 Hand depth detection

6 Team Composition

The project team is composed of four members, each with unique skills and expertise in software development, computer vision, and user interface design.

Member 1

Augustin MASCARELLI
Artificial intelligence enthusiast, with a master's degree in data science and 3 years of professional experience. During my engineering studies, I acquired skills in machine and deep learning through numerous projects in classification, image analysis, natural language processing.

Member 2

Yona BITTON
Business Intelligence professional with a master's degree in Data Analysis and AI. Through my experience, I have developed a strong skillset in Machine Learning, Deep Learning projects. My internship as a JavaScript developer has given me valuable technical exposure and effective communication skills.

Member 3

Quentin AMIEL

As an engineer specialized in Data Science and Artficial Intelligence, I have a strong expericence and in-depth knowledge in Data field especially in Computer Vision and Neural Networks. I can bring my leadership, my creativity and my technical skills to help the group go further.

Member 4

Kelig LEFEUVRE

Having a master's degree in data science and being passionate about artificial intelligence, I have developed an expertise in this field. My very good analytical skills are an asset for the group.

Member 5

Enzo COGNÉVILLE
Quantitative developer with a background in computer science and mathematics. I have more than 2 years of experience in data analysis, programming in Python, and creation of algorithms for analysis or financial risk management.

Member 6

Elias DÈVE
I have a Master's degree in data science and two years as a Data Engineer. My expertise includes Machine Learning, Deep Learning, data ingestion, processing, and indexing using tools such as Python, Hadoop, Spark or ELK stack. I am a detail-oriented problem solver with strong communication skills and a passion for delivering effective data-driven solutions.

References

1. OpenCV: Bradski, G. (2000). Open Source Computer Vision Library. Willow Garage, Menlo Park, CA. `https://opencv.org/`
2. TensorFlow: Abadi, M., Agarwal, A., Barham, P., et al. (2015). An End-to-End Open Source Machine Learning Platform. Google Brain Team, Mountain View, CA. `https://www.tensorflow.org/`
3. PyTorch: Paszke, A., Gross, S., Massa, F., et al. (2016).An Open Source Machine Learning Framework. Facebook AI Research, Menlo Park, CA. `https://pytorch.org/`
4. Unity: Lange D., et al. (2016). A Game Engine for Developing 3D Applications and Games. Unity Technologies, San Francisco, CA. `https://unity.com/`
5. Qt: Nord, H., Chambe-Eng, E. (1995). A Cross-Platform Framework for Building GUI Applications. Trolltech, Oslo, Norway. `https://www.qt.io/`
6. MediaPipe Hands: On-device Real-time Hand Tracking, F.Zhang, V.Bazarevsky, A.Vakunov. `https://arxiv.org/pdf/2006.10214.pdf`