**Final Project Documentation: Contactless Document Signing via Hand Tracking**

**Table of Contents**

1. [Project Overview](https://chat.openai.com/c/164c7992-4c54-4b4f-afa8-4eaa9347a58d#project-overview)
2. [Algorithm Description](https://chat.openai.com/c/164c7992-4c54-4b4f-afa8-4eaa9347a58d#algorithm-description)
3. [Results](https://chat.openai.com/c/164c7992-4c54-4b4f-afa8-4eaa9347a58d#results)
4. [Codebase and Repository](https://chat.openai.com/c/164c7992-4c54-4b4f-afa8-4eaa9347a58d#codebase-and-repository)
5. [Challenges and Limitations](https://chat.openai.com/c/164c7992-4c54-4b4f-afa8-4eaa9347a58d#challenges-and-limitations)
6. [Conclusion](https://chat.openai.com/c/164c7992-4c54-4b4f-afa8-4eaa9347a58d#conclusion)
7. [Appendices](https://chat.openai.com/c/164c7992-4c54-4b4f-afa8-4eaa9347a58d#appendices)

**Project Overview**

**Objective**

The primary goal of this machine learning project is to develop a robust hand tracking algorithm capable of accurately detecting and computing the path of a hand within the field of vision of a camera. This technology is intended to facilitate contactless document signing by tracking the movement of a hand as it signs a document in the air, eliminating the need for physical contact with pen and paper. This innovation not only enhances computer vision and hand recognition capabilities but also pioneers the development of contactless interfaces for computer applications, providing a sanitary and technologically advanced solution for various industries.

**Background**

This project was initiated in response to the global COVID-19 pandemic, which highlighted the need to minimize physical contact in daily activities as a precaution against the spread of viruses and bacteria. One common yet critical point of contact is document signing, a process traditionally requiring shared pens and physical documents. By developing a contactless method of signing, this technology addresses both immediate and future needs for hygiene and safety in document handling.

The initiative builds upon previous work in the field of hand tracking and gesture recognition undertaken by a team of students from the previous year at the Transport and Informations Institute in Riga. This project aims to enhance the earlier model's capabilities, achieving greater accuracy and reliability in hand path computation. It leverages advancements in machine learning, specifically in computer vision, to create a solution that could be pivotal in future interactions with digital systems, reducing the reliance on physical interfaces.

**Algorithm Description**

**Model Type**

The hand tracking functionality is powered by a pretrained Google model known as MediaPipe. This model is specifically developed for detecting human features such as heads, hands, and bodies, making it highly suitable for applications that require precise human feature recognition. MediaPipe was selected due to its efficiency and ease of use, as well as its robust performance in tracking dynamic objects. Additionally, this model was utilized in previous projects, providing a foundation for its adoption and further enhancement in our project.

**Features**

The primary feature tracked by our model is the index finger's position, which is crucial for computing the hand's path during the signature motion. This positional data is directly provided by the MediaPipe model, which detects and tracks the hand in real-time or in pre-recorded video feeds. The focus on the index finger's position is essential as it gives the signer a clear indication of where their signature is being placed, mimicking the motion of writing with a pen on paper.

**Data Sources**

The data for this project comes from video feeds, which can be either live or pre-recorded. This flexibility allows the model to be trained and tested in a variety of scenarios, enhancing its robustness and adaptability.

**Preprocessing**

Preprocessing of the video data is tailored to meet the input requirements of the MediaPipe model. This includes converting the color coding of the video from BGR (Blue, Green, Red) format to RGB (Red, Green, Blue), which is necessary for accurate hand feature detection by the model. Additional preprocessing experiments were conducted to explore the optimization of hand recognition by focusing the model's attention on likely hand locations. Techniques such as cropping the video to include only the expected hand area or applying a mask to the rest of the video were tested. These methods aim to reduce computational load and evaluate the potential improvements in processing time and recognition accuracy by focusing analysis on the most relevant parts of the video.

**Results**

**Performance Summary**

The pretrained model demonstrated generally good performance in tracking hand movements but struggled with recognizing blurred hands, particularly when the hand moved quickly. Benchmark testing, conducted on a specific benchmark video, revealed that the model processed normal and masked frames at about 52 frames per second (fps), while cropped frame processing varied between 40 and 52 fps. Interestingly, masked frame processing showed a slight improvement in framerate, approximately 1 fps higher than the other methods.

In terms of hand detection accuracy during these tests, the model achieved around 80% on full frame processing. This accuracy dropped in the cropped frame scenarios and improved slightly with masked frame processing. Despite these variations, the overall quality of the signature curve was notably better when the full frame was processed compared to either the masked or cropped frames.

**Visualizations**

**Une image contenant texte, capture d’écran, diagramme, Caractère coloré

Description générée automatiquement**

Figure 1 Methods' benchmark results

There are multiples codes here. We can read the code as follows: {Methods}\_{Detection Threshold Change}\_{Interpolation Method}

In the {Methods} category, we can have ‘C’ for cropping method, ‘F’ for full picture method (or no processing method) and ‘M’ for masking method.

In the {Detection Threshold Change} category, the attribute scale will modify two things in the program. The first thing is the cropped or masked area. The smaller the scale attribute, the smaller the considered area. But also, the smaller the scale attribute, the smaller the hand detection threshold. This was with the idea of improving detection when there was less information on the picture. As such, the code ‘DTC’ means “Detection threshold changing” referring to the second part and ‘DTF’ means “Detection threshold fixed” to 0.7. Incidentally, when the scaling is of 100, the detection threshold is of 0.7 too.

The last part, {Interpolation Method} category, consider the methods to interpolate the next hand position.

Two algorithms were considered, the first one:



Figure 2 First interpolation algorithm

This algorithm would consider the last position of the hand, and the last hand’s velocity and compute the next position. It would then add a fixed correction as an area around the position.

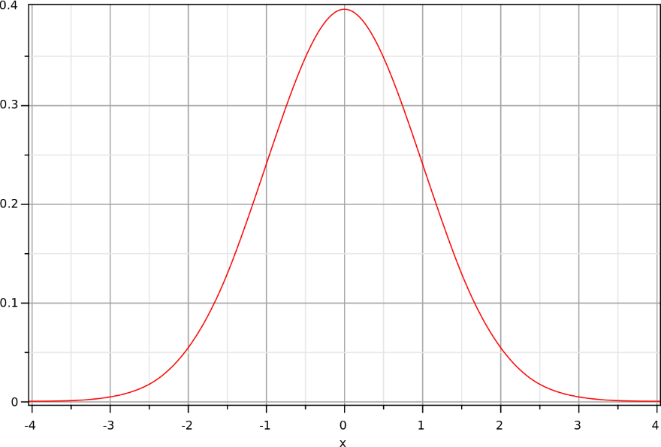
The second algorithm:

Figure 3 Second interpolation algorithm

The second algorithm would use a modified Gaussian function dependent on the last position, the velocity, and the scaling to compute a possible area where the hand could be.

Considering these two methods, the codes become ’DMD’ when the density function was used as the interpolation algorithm, and ‘DMS’ when the simpler one was used.

Incidentally, the comparison between the two methods gives the density one better.

Une image contenant texte, diagramme, ligne, Tracé

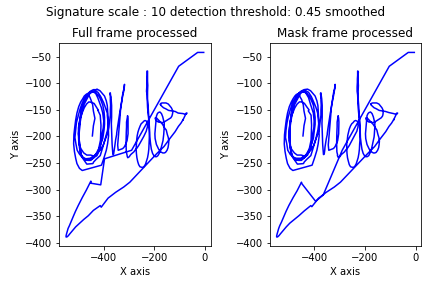
Description générée automatiquement

Figure 4 Comparison between the interpolation algorithms

On the signature results, here is the signature on the benchmark video.

0,7

Figure 5 Benchmark signature using the full picture and the masked picture



The right picture is noisier than the left one even when it had better measurements before.

Visual aids included in the documentation illustrate the average frame rate and detection efficiency for full, cropped, and masked frame processing, alongside plots depicting the resulting signature curves. These visualizations are crucial for understanding the practical impacts of different preprocessing methods on the model’s performance.

**Interpretation**

The results indicate that preprocessing the frames to potentially enhance hand tracking did not yield the expected improvements. In fact, the modifications slightly degraded the performance, particularly affecting the quality of the signature curve. This suggests that while preprocessing can influence detection efficiency and processing speed, its impact on the overall application effectiveness—especially for precise tasks like signature tracking—can be counterproductive.

**Unexpected Outcomes**

One interesting finding was that preprocessing techniques, which theoretically should have simplified the model's task by isolating the hand region, actually resulted in worse performance outcomes. This was unexpected, particularly since the visual differences between the full and modified frames were not substantial. This highlights the complexity of hand tracking in dynamic conditions and suggests that further research might be needed to explore other methods of improving recognition accuracy without compromising the quality of the output.

**Codebase and Repository**

**Repository Structure**

The project repository is organized into several key directories and programming files to streamline development and testing:

* **Signature CSV:** Stores CSV files of signature point coordinates.
* **Signatures Plots:** Includes plots of the signatures derived from the benchmark tests.
* **Video:** Houses various benchmark videos used for testing.
* **StereoDepth:** House various test with stereo vision. The file inside are not documented.
* Programming files are located outside these folders and are essential for running different components of the project.

**Primary and Secondary Files**

* **Secondary Files:**
  + **Result\_plotting:** Responsible for displaying live demo signatures.
  + **Signature\_graphs:** Generates plots for benchmark signatures.
* **Primary Files:**
  + **Camera: M**anages camera-related tasks.
  + **Hand:** Handles the hand recognition model tasks.
  + **Point:** Manages saving of point data and interpolation of the next point positions.
  + **Depth:** Attempts to handle depth-related tasks for computing the monocular depth of the finger position.
  + **BezierCurves:** Responsible for smoothing the signature curves using Bézier curve techniques.
  + **Main:** Integrates all components and manages the overall process.

These files are mostly documented.

**Dependencies**

The project utilizes several external libraries and frameworks:

* **MediaPipe:** For the hand recognition model.
* **OpenCV2:** For camera management.
* **Bezier, Simpy:** For generating Bézier curves.
* **Numpy, Torch, Scipy:** Used primarily for depth calculation tasks.
* **Matplotlib:** For generating plots and visualizations.
* **Pandas:** For exporting data to CSV format.
* **time, threading:** For managing timing and parallel processes.

**Setup Instructions**

To set up the project environment, follow these general steps:

1. Ensure all dependencies are installed. This can typically be done via pip, e.g., `**pip install mediapipe opencv-python bezier simpy numpy torch scipy matplotlib pandas**`.
2. Clone the repository to your local machine.
3. Navigate to the project directory.

**Running the Project**

* To start the project, run the`**Main**` script. This script integrates all components and initiates the hand tracking and signature processing functionalities.

**Challenges and Limitations**

**Known Issues**

While there are no bugs in the implementation, several issues impact the reliability and effectiveness of the project:

* **Path Detection**: The hand path detection isn't 100% reliable, occasionally resulting in noisy signatures.
* **Hand Detection**: Limited visibility or clarity of the hand reduces the hand detection rate.
* **Depth Computation**: Challenges persist in accurately calculating the depth of the hand in the scene.
* **Curve Smoothing**: Current methods for smoothing the signature curve need refinement.
* **Signature Distortion**: Distortions in the signature occur due to inadequate camera calibration.

**Areas for Improvement**

* **Hand Detection and Path Prediction**: Enhanced research or additional training may improve hand detection accuracy. However, predicting the hand's next position during random movements like signatures remains challenging.
* **Depth Estimation**: Three methods were explored:
  1. **Stereo Vision**: Initial attempts using two cameras with epipolar geometry failed, likely due to mismatched camera specifications.
  2. **Monocular Depth Estimation**: While this approach provided a depth map, it was either imprecise or computationally intensive for live applications.
  3. **Geometric Estimation**: Using hand size and bounding box geometry proved unreliable, especially with varied hand orientations.

Further attempts might reconsider the stereo vision approach, focusing on using identical cameras as studies seems to show this approach possible. Otherwise, the add of a third camera and triangulation will of course work…

* **Curve Smoothing**: Further experimentation with various smoothing techniques could enhance the quality without losing significant data.
* **Camera Calibration**: Improvements in camera calibration, potentially using a professionally manufactured chessboard, could reduce signature distortions.

**Limitations**

The project faces inherent technological limitations:

* **Hardware Requirements**: The need for one or two high-quality cameras and the computational power to run complex models adds significant cost.
* **Model Costs**: Licensing or development costs associated with advanced models may be prohibitive for live applications.
* **Economic Constraints**: In a cost-sensitive industry, these financial and technological barriers significantly challenge the widespread adoption and practical implementation of this technology.

**Conclusion**

This documentation has detailed the development and deployment of a machine learning project aimed at enabling contactless document signing through hand tracking. Utilizing the pretrained MediaPipe model by Google, the project focused on accurately tracking the movement of a hand to simulate the process of signing a document in air. Despite the high potential of the chosen model, challenges such as path detection reliability, hand visibility, depth computation, and curve smoothing were encountered, which occasionally affected the output quality.

**Achievements and Impact:** The project successfully implemented a system capable of hand tracking with an approximate 80% accuracy in optimal conditions. It demonstrated the feasibility of using advanced computer vision techniques for real-world applications like contactless interfaces, which could be particularly useful in enhancing sanitary conditions and reducing physical contact in various environments.

**Future Work:** Future enhancements could include refining hand detection algorithms, exploring more reliable depth estimation methods, and improving the calibration of the system to reduce distortions. These improvements could further enhance the system's accuracy and reliability, making it more adaptable to diverse application scenarios. Additionally, the exploration of cost-effective hardware solutions could make the technology more accessible for widespread use.

**Appendices**

* **Glossary**

No information to add.

* **References**

This report was written using ChatGPT4. <https://chat.openai.com/share/9508c10c-368f-4635-91ce-d10752986b71>

A trello was used for paper information sharing <https://trello.com/b/d19CRSw7/signature-recogition>

Another one for task advancement sharing <https://trello.com/b/QR4xOhmA/remotescan-recognition>

The previous project work github <https://github.com/FallenElias/Finger_AirSigning_Project>

Stereovision approach possibility <https://learnopencv.com/depth-perception-using-stereo-camera-python-c/>

* **Contact Information**

This work was done under Alexander Gravoski tutoring.

* **Group teamwork**

Testing, Code, Repository architecture, Documentation: Victor