BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION Computer Science in English

DIPLOMA THESIS

AI-Driven Hand Tracking Application for Real-Time Drawing

Supervisor
[Grad, titlu și Tudor-Dan Mihoc]

Author
Sali Arnold

2024

ABSTRACT

Contents

Chapter 1

Introduction

Chapter 2

Related Work

The field of computer vision has evolved from natural vision. Its main purpose is to allow machines to understand visual information based on the natural way humans and other animals gain information from visual input.

This branch of computer science is widely used to gain insight into the real world, through different algorithms and computational techniques, ranging from simpler problems such as object detection in an image or video to more complex ones such as scene understanding and image generation. The solutions from this field open the door for new innovations, which are already present in some capacity in today's society such as a simple social media filter or self-driving cars.

The evolution of computer vision can easily be followed along with the evolution of computational power, given the high requirements of image processing. In the earlier days, such as the 1960s and 1970s, the first algorithms were very limited by the processing power available, but as time passed, more sophisticated ones were created, such as Hough Transform, the Viola-Jones face detection and machine learning.

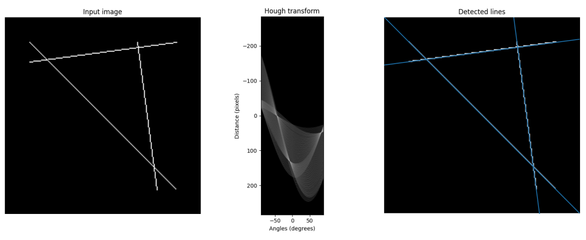

Figure 2.1: Hough transform calculation to find the lines of a triangle [Unk]

2.1 Image segmentation algorithms

2.1.1 Hough Transform

This technique was proposed by Paul Hough in 1962, as a method to identify patterns in images. [DH72]

The Hough Transform is used in computer vision and image processing. The core idea behind it is a voting process made by the curves in the transform space, creating cluster points or local maxima. These points represent the existence of a shape, and since the curves contain the parameters of the shapes, from those parameters the position of the shape in the input image can be detected, like it is shown in Figure 2.1.

The main strength of this method is that it is capable of identifying shapes even in slightly obscured or noisy data. This made this procedure a massive breakthrough in image processing. Since it's powerful to be able to detect shapes, the method is used in many sectors, from robotic navigation, by identifying edges and lines in the environment, to medical imaging, by detecting circular shapes potentially depicting different biological conditions and many more, where image processing can be utilized.

2.1.2 Viola-Jones Face Detection

Another breakthrough in computer vision was the face detection procedure proposed by Paul Viola and Michael Jones in 2001. [VJ01]

The main idea behind their approach lies in the usage of Haar-like feature and Adaptive Boosting.

The Haar-like features are simple rectangular regions in the input image data. The main goal of using these patterns is first to cut down on the necessary calculations by working with pixels, and second is to get a better understanding about certain regions in the image. By calculating the differences in the sum of pixel intensities between the regions, a contrast can be learned, an incredibly useful information which allows the detection of basic patterns, like edges, lines and textures.

Adaptive boosting is a machine learning algorithm used for boosting weak classifiers into a strong one. The main principle of this method is to iteratively train weak classifier, each focusing on different features. At each iteration pass, the algorithm selects the best performing classifier and merges it with the final strong one, weight proportionally to its performance. It also weighs incorrectly classified instances more for the next iteration for the purpose of shifting the focus on the most challenging parts.

With the help of these two procedures the method of Viola and Jones achieves great accuracy in real-time, more specifically at 15 frames per second on a 700 MHZ Intel Pentium III, using approximately 50 thousand parameters. Since its low hardware requirements, this way of face detection is still used on very weak devices.

2.2 Deep learning algorithms

With the increase in computational power, image processing algorithms have also evolved in tandem. As seen in the Viola-Jones face detection algorithm, machine learning were already used in the early 2000s, be it at a smaller scale.

Since these algorithms now are able to utilize more resources, they can work with way more parameters, going from thousands to millions, allowing them to learn more complex features than before.

My choice for the following three deep learning architectures is that, they were significant achievements in the field of computer vision, while also resembling my chosen base model the MobilNet.

2.2.1 YOLO

YOLO (You only live once) is an object detection method, based on a Convolutional Neural Network backbone, the building blocks of which can be seen in Table 2.1, containing convolutional layers of varying sizes.

The main selling point of this algorithm, is that it achieves real-time performance with a single-pass approach. It divides the image into a grid, each predicting a bounding box, with an associated confidance score and class probability. Utilizing these parameters and various anchor boxes, which are bounding boxes with a predefined shape, size and aspect ratio, the model predicts the final bounding box for the searched cell. [RF18]

Since its real-time performance the method has been utilized in many projects, from autonomous driving, to surveillance systems.

Type	Filters	Size	Output
Convolutional	$64 - 1024$	$3x3/2$	$128x128 - 8x8$
Convolutional	$32 - 512$	$1x1$	
Convolutional	$64 - 1024$	$3x3$	
Residual			$128x128 - 8x8$

Table 2.1: Building blocks for the YoloV3 CNN backbone, Darknet-53 [RF18]

2.2.2 RESNET

ResNet is a convolutional neural network proposed by Kaiming He and co. in 2015, achieving great performances in various computer vision tasks.

The main strength of this model is that it addressed the vanishing gradient problem, via the introduction of skip connections. The residual blocks, the main building blocks of this model, contain the shortcut connections, skipping one or more layers. With this, the network is able to learn residual functions, basically the difference between the desired output and the input data. This feature allows the ResNet architecture to increase the depth of the model, without the gradient vanishing. [HZRS15]

As can be seen in Table 2.2, the complexity of this model can grow to incredible levels, containing as much as 19.7 million parameters, meaning the number of calculation for one single pass through of this architecture is around that number.

Number of layers	Number of parameters
20	$0.27M$
32	$0.46M$
44	$0.66M$
56	$0.85M$
110	$1.7M$
1202	$19.7M$

Table 2.2: Correlation between the size of the model and the number of parameters used [HZRS15]

2.2.3 MOBILENET

MobileNet is a convolutional neural network designed specifically for mobile and other devices with limited computational power.

Model	ImageNet Accuracy	Parameters
Conv MobileNet	71.7%	$29.3M$
MobileNet	70.6%	$4.2M$

Table 2.3: Difference in parameters between Depthwise separable and full convolutional MobileNet architecture [HZC$^+$17]

At its core the architecture is built using depthwise separable convolutions. The standard convolutions is separated into two separated operations. The first one being a depthwise convolution, in which a single convolution filter is applied for each input channel, capturing features separatly. The second is pointwise operations, which applies a 1x1 filter to the outputs of the the previous calculations, combining the results. This allows the model to significantly reduce the number of parameters needed, compared to normal convolution, thus decreasing the computational power needed. [HZC+17]

In Table 3.1 the incredible difference between the complexity of the two architectures. For a 1.1 percent loss the exponential decrease in parameter count is a fantastic exchange.

Chapter 3

Approaches for better performance with machine learning models

As the learning capabilities of deep neural networks increased over the years, so have their sizes in terms of the number of parameters. With this the resources needed to make the algorithms learn and to utilizes them have also increased, to the point where utilizing them in real-time applications have become somewhat challenging on an everyday computer.

To be able to utilize a deep neural network algorithm in real-time, an architecture has to be chosen, which minimizes the number of parameters, hence also minimizing the number of computations for each pass through the network, while also maintaining a useful learning capacity.

To be able to compare the running time of some models, I am going to use the running time of one pass through the architecture in milliseconds. The goal is to use them in a real-time video processing application, with a capped frame rate of 24 frames per second, would give a running time needed of around 41 milliseconds. Achieving this with a model complex enough to somewhat accurately identify hand positions and gesture, might prove difficult, since most architectures have the number of parameters in the millions. To combat this, only every second image will be processed, creating a more comfortable window.

3.1 Model comparisons

Starting with a simpler architecture in terms of the deep learning neural network, in the study of John and co. [JBM$^+$16] a high accuracy was achieved in real-time using representative frames, hence selecting better data for the machine learning model to predict from. They algorithm clearly separated the two parts, the frame extraction running in around 70 milliseconds and the classification running in around 40 milliseconds, achieving an accuracy of 91 percent.

3.1.1 YOLOv3 perfatic ormance

The YOLOv3 model is an efficient deep learning architecture, making it more than useful in achieving real-time performance on hand gesture detection and identification problems. In the approach of Mujahid, Awan and co. [MAY+21], this algorithm was thought from scratch on the Mindst dataset [Den12], achieving good results. They proposed a lightweight architecture which is built on the YOLOv3 model, achieving impressive results, with a approximate accuracy rating of 98 percent in real-time, although time specifications were not provided.

3.1.2 MobileNet performance

The approach of Wanga, Hua and Jina [WHJ21] consists of utilizing the architecture of MobileNet and Random Forest to identify hand gestures. They utilized the pretrained parameters of MobileNet on the ImageNet dataset [DDS+09], than taking the output and running it through a Random Forest model to better extract features from the images. Their paper does not focus on performance in the context of the time needed for a prediction, that being said, since they use the MobileNet architecture as their backbone, which in a configuration of around 3.5 million parameters can achieve a pass-through time of around 35 milliseconds, it is safe to assume, that near real-time performance is achievable with their approach.

In Table 3.1 the comparison of the two models can be seen with regard to their accuracy ratings on three different hand based image datasets.

Model	SLD Dataset Accuracy	SLGI Dataset Accuracy	Fingers Dataset Accuracy
MobileNet	74.25%	94.12%	97.02%
MobileNet-RF	80.97%	95.12%	99.72%

Table 3.1: MobileNet and MobileNet-RF accuracy, from the study of Wanga and co. [WHJ21], on the Sign Language Digital Dataset (SLD) [KSH22], Sign Language Gestures Image Dataset (SLGI) [VMGMST+23] and the Fingers Dataset

3.2 Transfer learning

Transfer learning is a machine learning technique where a pretrained model on a specific task and dataset is repurposed and fine-tuned for a different but similar problem. The pretrained values are taken for a new task, since the learned functions are usable in the new context, like in the example of identifying animals and human faces, the pretrained values will help to extract certain features which characterizes these objects, like facial line structures.

Using this method the deep neural network will start in closer to optimal position when learning features, jump starting the process, and enhancing the overall accuracy of the model. It is also a very useful tool, when dealing with a smaller dataset, or when collecting more data proves to be more challenging, allowing the architecture to learn with less data. This is also the reason why this approach is used in this application, since collecting thousands of images and labeling every one of them would take up an incredible amount of time.

Chapter 4

Machine learning model specifications

4.1 Modified MobileNet

4.1.1 Transfer learning specifications

4.1.2 Model output

4.2 Data

4.2.1 Data specification

4.2.2 Data preprocessing

Chapter 5

Application requirements and specifications

The main goal of the application is to create an alternative way of interacting with a live video. For this purpose a deep learning neural network is used, more precisely a MobileNet model pretrained on the ImageNet dataset, with custom final layers for identification. The machine learning algorithm will identify hand gestures, acting as the controlling tools for the application.

5.1 Application requirements

5.1.1 Functional requirements

The main focus of the application is the alternative way of interacting with the live video feed, providing feature for drawing with a finger, using simple hand gestures for clearing the drawings, for zooming in on a specific section of the video, for changing the volume of the recording and for taking a screenshot.

Besides the hand driven way of interactions, the zooming, the volume settings and the screenshot features all have a specific buttons and toggles, so they can be worked with normally with mouse actions.

The application can also be used as a simple video recording software, providing settings options for camera and microphone selection, changing the folder to which save the video files and screenshots and video file extension options, such as mp4 and avi.

5.1.2 Non-functional requirements

The application only has one version for the windows operating system, the results of running it on Linux, MacOS or any other operating system is undefined, they have not been test.

The processing of the input images from the video feed is done in real-time with a locked frame rate of 24 frames per second. The hand gesture identification and finger tracking is done on every second or third frame, depending on if the the 24 frames per second is obtainable.

The saving of the video files and images are done by only accessing those particular folders, with a naming convention of Recording
Screenshot with the current date and time, so conflicts with other existing files is almost impossible, at the very least highly unlikely.

5.1.3 System requirements

The application only runs Windows operating system, needing the Windows 10 or 11 64-bit version, running on other versions may result in undefined behaviour.

In terms of the CPU the application was tested on Ryzen 7 4800h with a clock speed of 2.9 Gigahertz, from which 20-25 percent was used while running. From this data, an assumption can be drawn that as a baseline, a hardware similar to an intel core i7-9750h with a clock speed of 2.6 Gigahertz is advised. By having a weaker CPU the application may run into lagging issues when it comes to the rel-time performance of image processing.

The RAM used by the application is relatively small, using only 300 megabytes from a 3200 Megahertz unit.

In terms of storage, the application only needs around 300 Megabytes. The real impact will come from the saved video and screenshot files, depending on the file type in which they are saved.

5.2 Technical specifications

For the purpose of easier integration of the deep learning model, the language used to develop the application is python, since most frameworks used for machine learning are written in this.

For creating and working with the MobileNet architecture, I use the TensorFlow [AAB+15] and Keras framework [C+15]. TensorFlow is an open-source project developed by Google giving an easy API for developing machine learning models efficiently, by wrapping up the more complex C++ and CUDA core functionalities. It also supports Keras, which is an open-source neural network library, providing an easy interface for building and teaching models, while also containing several known once, so they can be accessed with ease.

For the video capturing and the opencv library [Bra00], which is open-source computer vision library written in mostly C++ and it is often used in machine learning projects. Because of that reason, the data, which is given back is easy to process and use in different models, making the development process that much easier.

For the creation of the graphical user interface, the cross-platform Qt framework is used [Gro95]. It is a wildly used service for creating high quality interfaces and is written in C++, providing good performance while supporting multiple languages like python. It provides various tools and modules, providing a relatively easy environment for UI development.

Besides the major libraries and framework, the win32api package [Mic] is also used to access certain computer specifications, like the width and height of the monitor screen, to align the application at launch.

To store the settings of the user, the json format is used, so the structure of the storage is easy to read for both the application and the user, in case a manual change would be wanted. For this the built in json library [Fun] is used in python.

Chapter 6

Application design and implementation

6.1 Graphical User Interface

6.2 File Repository

6.3 Image processing

Chapter 7

Application testing

7.1 Testing methods

Chapter 8

Performance metrics

8.1 Computing machine specifications

8.1.1 Training specifications

8.1.2 Running specifications

8.2 Training metrics

8.3 Real-time performance

Chapter 9

Future Work

9.1 Machine Learning Model Improvements

9.2 GPU Utilization

Chapter 10

Conclusions

Bibliography

[AAB$^+$15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[C$^+$15] Francois Chollet et al. Keras, 2015.

[DDS$^+$09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Den12] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[DH72] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, jan 1972.

[Fun] Python Software Fundation. Json library.

[Gro95] Qt Group. Qt, 1995.

[HZC+17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[JBM+16] Vijay John, Ali Boyali, Seiichi Mita, Masayuki Imanishi, and Norio Sanma. Deep learning-based fast hand gesture recognition using representative frames. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, 2016.

[KSH22] Maria Kopf, Marc Schulder, and Thomas Hanke. The Sign Language Dataset Compendium: Creating an overview of digital linguistic resources. In Eleni Efthimiou, Stavroula-Evita Fotinea, Thomas Hanke, Julie A. Hochgesang, Jette Kristoffersen, Johanna Mesch, and Marc Schulder, editors, *Proceedings of the LREC2022 10th Workshop on the Representation and Processing of Sign Languages: Multilingual Sign Language Resources*, pages 102–109, Marseille, France, 2022. European Language Resources Association (ELRA).

[MAY+21] Abdullah Mujahid, Mazhar Javed Awan, Awais Yasin, Mazin Abed Mohammed, Robertas Damaševičius, Rytis Maskeliūnas, and Karrar Hameed Abdulkareem. Real-time hand gesture recognition based on deep learning yolov3 model. *Applied Sciences*, 11(9), 2021.

[Mic] Microsoft. Win32api.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[Unk] Unknown. Straight line Hough transform. `https://scikit-image.org/docs/stable/auto_examples/edges/plot_line_hough_transform.html`. Online; accessed 23 March 2024.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[VMGMST$^+$23] María Villa-Monedero, Manuel Gil-Martín, Daniel Sáez-Trigueros, Andrzej Pomirski, and Rubén San-Segundo. Sign language dataset for automatic motion generation. *Journal of Imaging*, 9(12), 2023.

[WHJ21] Fei Wang, Ronglin Hu, and Ying Jin. Research on gesture image recognition method based on transfer learning. *Procedia Computer Science*, 187:140–145, 06 2021.