Efficient vision-based hand gesture recognition on low-power devices

Rodrigue GASPARD

November 4, 2024

Abbreviations

HGR = Hand gesture recognition CNN = convolutional neural network CPU = central processing unit GPU = graphical processing unit EMG = Electromyography SBC = Single-board computer IoT = Internet of Things ToF = Time of Flight IR = Infrared FPS = Frame per second CFS = Correlation-based feature selection

Abstract

This is the abstract

1 Introduction

1.1 Problem statement

With the democratization of handheld devices, smart home appliances, virtual and augmented reality environments, and the emphasis on accessibility in software development, new ways to interact with technology are required in order to bridge the gap between man and machine, and to capitalize on the portable access of these devices.

Several methods and technologies aims to tackle this issue, such as voice-assist, motion capture, gesture recognition, eye-tracking technology, and so on.

However, compared to traditional computer peripherals, those often requires specific hardware requirements, extensive data as well as complex and resource-intensive algorithms. For facial or hand-gesture recognition, a graphical processing unit is often necessary for real-time applications. Unfortunately, this increase manufacturing costs significantly, and requires more power, the latter being a heavy constraint for embedded devices, single-board computers and other low-power devices. Thus, optimizing existing algorithms for CPU-only environments might allow those devices to perform more tasks and make them more accessible. Most research done on those algorithms use powerful GPUs in order to drastically increase performance and reduce training time, sometimes by several orders of magnitude. As a result, this creates a research gap when it comes of benchmarking deep-learning models on CPU-only environments.

Thus, this paper will explore the performances increases between various optimization techniques for vision-based hand gesture detection convolutional neural network models on CPU-only environments, in order to highlight where the focus should be when implementing such systems.

1.2 Background

To interface with machines, humans have used a plethora of peripherals of techniques in the past. The keyboard, which is straightforward and was already familiar to users as it used for typewriters and pointer devices such as the computer mouse. Then came remote controllers, either by radio or infrared signals, or by Bluetooth/Wi-Fi technology, which allows more freedom of movement to the user, instead of being constrained to the close locality of the machine being used.

However, those methods often required some kind of training from the user in order to be able to use the machine. The idea of controlling devices remotely using body language or voice commands, which offers both flexibility and user-friendliness compared to the previously mentioned techniques, became a viable option around the end of the century.

Voice-assisted technology allows users to interact with devices and software through spoken commands. It usually involves speech recognition, which converts spoken words into text or commands, by detecting and tokenizing phonemes, words, and sentences. Text-to-speech, also known as TTS, converts text input into spoken language with the help of synthesizers to produce speech, sometimes conveying emotions with impressive results, as demonstrated by OpenAI's ChatGPT-4o. A notable example for voice recognition was Dragon NaturallySpeaking, a proprietary software recognition package released by Dragon Systems in 1996, which was one of the first voice assistants available to the public, and allowed for continuous recognition.

Hand and body gesture recognition refers to the analysis of movements made by the hands, arms, and body to understand the intended actions or expressions. It can involve only specific parts of the body, for example the face, to detect sentiments, for facial recognition by focusing on key features and cross-reference them in a database in order to get the individual's identity. It can also be used to capture input and mimic them in a skeleton-based approach to a 3d model for animation/special effects purposes. Gesture recognition can also involve the whole body, usually for motion capture, gait analysis, or for various security reasons (concealed weapon and theft detection).

The focus of this paper is hand gesture recognition, as it provides several advantages compared to full-body or facial motion capture. First, hand gestures can convey a lot of information easily, as demonstrated by the several visual languages, also called sign languages. The most full-fledged sign language is often considered to be American Sign Language (ASL). ASL is a complete, complex language with its own grammar, syntax, and vocabulary, distinct from English. It is widely used within the deaf community in the United States and parts of Canada. ASL also integrates fingerspelling, where each letter of the Latin alphabet is represented by a distinct hand gesture or sign. It is typically used to spell out proper nouns, technical terms, or words that do not have an established sign in a particular sign language. Fingerspelling allows users to communicate words or concepts that may be difficult to represent with standard signs, such as names, addresses, or specialized vocabulary. Thus, one can convey rich and complex ideas using only their hands, without requiring additional context or verbal and non-verbal cues. Hand gesture recognition is also more flexible and natural than full-body or facial recognition, it does not require as much space as full-body recognition and can be used more easily and naturally than facial recognition (it does not require looking straight at the camera with a specific intent). It also allows for people with mobility issues to use this system with ease, without straining their body or exerting too much energy. Furthermore, hand gestures can take many forms (fist clenched, palm opened, any number of fingers extended or retracted), thus are very clearly identifiable between one another. This can help to lower the computational complexity required to detect the gestures and increase the mean accuracy of the detections, making the technology more reliable when applied as a human-machine interface. Finally, multi-user interfacing, that is detect several hand signals at the same time from multiple agents is easier to implement, as the requirements for efficient detection are low compared to full-body or facial detection.

1.2.1 Static and dynamic hand gestures

Hand gestures can be classified in two categories : static and dynamic. In static hand gesture recognition, the system identifies gestures that involve a single posture or shape of the hand that remains relatively fixed in space. It focuses on recognizing hand configurations or positions without considering movement over time. For example, recognizing a hand gesture showing a specific number of fingers (a thumbs-up, the "peace" sign, or the number four). These gestures can be simply detected using techniques such as shape analysis or contour detection from single frames, as there is no notion of time tied to the meaning of the hand gesture. Thus, it's often considered simpler to implement than dynamic recognition, as it only requires capturing a single frame or static pose. It's also well-suited for tasks where the hand gesture is briefly held (recognizing numbers or simple commands, such as manipulating basic software). However, for more complex applications (sign language interpreter, complex software interactions such as drawing or gaming), this approach might not be enough to convey the actions of the user in detail. Dynamic hand gesture recognition involves detecting and interpreting gestures that include hand movement over time. This type of recognition analyzes not only the shape and position of the hand but also the trajectory, speed, and changes in hand posture across a sequence of frames. Waving "goodbye", pointing in a specific direction, pinching, or drawing shapes using your hands are all examples of dynamic hand gestures. Dynamic gestures require tracking the hand over multiple frames to capture motion, often using optical flow or skeletal tracking, the former being the apparent motion of objects in a visual scene caused by the relative movement between the camera (or observer) and the objects, and the latter being a computer vision technique used to detect and track the position of a human body or hand by identifying key points (or joints) that make up the "skeleton" of a person or limb. While dynamic hand gesture recognition can capture more complex and rich gestures, and can also be more suitable for real-time applications, such as controlling devices through continuous hand gestures (navigating a menu by swiping or controlling a robot or a drone), it is more computationally intensive than static hand gesture recognition because it requires processing and analyzing multiple frames, which can be challenging for real-time applications, and it can be sensitive to speed variations in the gesture execution, requiring robust algorithms and fine-tuning to account for temporal differences.

We will focus on static hand gesture recognition for this thesis. As we're focusing on low-power devices with limited computational power, static hand gesture recognition is more suited for these conditions, and it can still provide enough information for basic applications, such as manipulating a video player.

1.2.2 Detection techniques

In order to capture the movements of the human body and translate it into input for a computer, special hardware is often required. We can classify those sensors into two broad categories : sensor-based and vision-based [**?**]. The sensor-based approach uses a variety of different techniques, namely :

- Glove-based approach, which uses gloves with a wide range of sensors, that the user must wear in order to input finger or hand movement for detection. These sensors include, but are not limited to flex sensors, which records the bending and movement of individual fingers, accelerometers and gyroscopes which track the movement and rotation of the hand and pressure/force sensors, which can be used to detect clenching or pinching. While they are considered the most reliable way to detect hand gestures due to the rich and precise data they provide, they're not always suited for most applications as you need to wear the gloves at all times in order to detect hand gestures. They're also not the most cost-effective way of making detections, as the gloves requires special, often expensive sensors and specific tailoring for the user.

- Electromyography, or EMG, can also be used for detection, by measuring the electrical activity produced by skeletal muscles when they contract. By placing electrodes on the surface of the skin at specific points, it is possible to capture the electrical signals generated by muscle activity in the hand and forearm. These signals vary depending on the type of movement or gesture being performed. Much like the data gloves, it can achieve extreme levels of accuracy, but suffers from the same problems, the equipment required is not readily available and can be expensive, and the user has to have the electrodes at all times during the detection process. Furthermore, EMG signals can wildly vary between individuals depending upon several factors (age, physical fitness, medical conditions, medications, etc.) resulting in noise and false readings. Finally, correct electrode placement can be tedious without prior medical anatomical and medical knowledge.

- Detecting using radio or Wi-Fi waves, which leverages the way wireless signals interact with human movement. Both technologies exploit changes in the reflection, absorption, scattering or frequency (Doppler shift [**?**]) of wireless signals caused by hand movements to recognize gestures. While these techniques might not always require wearable sensors, can be readily available and cost-efficient in the case of Wi-Fi detection, and have decent range and accuracy, they are still subjected to perturbations like signal congestion, variations of signal strength, and in the case of radar detection, can be power-hungry and a concern for low-power devices.

The vision-based approach refers to using computer vision techniques to detect, track, and recognize hand movements or postures from visual data, typically captured by cameras or optical sensors. This approach relies on image or video processing algorithms to interpret gestures in real-time or from recorded footage. The entire process revolves around using almost exclusively visual data rather than external sensors like gloves or electrodes, which makes it non-intrusive and more convenient for real-world applications. However, challenges such as lighting conditions, background complexity, and occlusions can impact the accuracy of this method. It is generally considered less accurate overall than the sensor-based approach.

For this paper, I chose to explore the vision-based approach over the sensor-based one for detection for several reasons :

- The vision-based approach is easier to implement than other methods, as it only relies on visual data, typically in the visible or infrared spectrum. There are several techniques available for detection and there are a lot of datasets in relation to hand gesture recognition available, making the machine learning and deep learning detection approach simple to implement.
- Vision-based systems capture gestures from a distance without the need for wearable devices like gloves or sensors. This makes the interaction more natural and intuitive for users. Also, in relation to low-power and embedded devices which often need to be as compact and portable as possible, this approach is more suited as it only requires a camera.
- Multiple users can perform gestures at the same time, and vision-based systems can handle these inputs concurrently without interference, which is more difficult with sensor-based approaches.
- Vision-based systems can scale to large spaces and public installations, such as in smart homes, interactive displays, or public settings, where sensor-based systems might be impractical due to range and hardware limitations.
- Vision-based techniques approximate how humans perceive gestures, making it easier to design systems that feel intuitive and human-centered. The raw data input is also more intuitive than the sensor-based techniques, where prior knowledge in physics (radar or Wi-Fi detection makes use of wave theory, and EMG requires basic understanding on how muscles works)

Detecting and processing hand gestures in real-time, as in interpreting the contents over a live video feed and outputting a prediction in a reasonable short interval in order to interact in a fluid manner with a computer, presents several challenges to overcome. In real-time systems, gestures must be processed almost instantaneously to ensure a seamless user experience. Any delay between the gesture being made and the system's response can cause frustration and disrupt the interaction flow. Devices with limited computational power, such as mobile phones or embedded systems, can struggle to handle the real-time processing demands. Running complex algorithms or machine learning models on such hardware without introducing lag is a key challenge. Extracting the necessary data through feature extraction from video frames quickly and accurately is critical for real-time recognition. Advanced algorithms like deep learning may offer high accuracy but are computationally demanding, making it challenging to run them in real-time without optimization. Such models must be lightweight enough to run in real-time without overloading the processor (and also permits the user to run additional programs on top without issues, in order to actually use the detection model), but robust enough to accurately classify gestures in varying condition. Real-time detection is critical for most applications where hand gesture recognition is needed. This is why optimizing the various steps of the model and highlighting the impacts of optimization techniques will be important.

The focus of this paper being low-power devices, where power efficiency being crucial, it is important to talk about the hardware constraints. Most low-power devices do not come with the processing power necessary for complex tasks, such as training deep learning models. Most SBCs only comes with a central processing unit and no graphical processing unit, due to the power restrictions. The central processing unit, also called CPU, is the primary component of a computer responsible for carrying out general-purpose tasks, including arithmetic, logic, control, and input/output operations. It is designed to handle a wide range of tasks sequentially, making it a versatile processor for a variety of computing needs. Modern CPUs have multiple cores, allowing them to handle some parallelism, but they are still optimized for serial tasks. The graphical processing unit, or GPU, on the other hand is specialized hardware originally designed to accelerate the rendering of images and videos, especially for gaming and other graphical tasks. Unlike CPUs, GPUs have a large number of smaller, more efficient cores designed to handle many parallel tasks simultaneously. This architecture makes them highly effective for tasks that require processing large amounts of data in parallel, such as matrix and vector operations. Unlike CPUs, GPUs typically have hundreds to thounsand smaller cores, each with different functions. The NVIDIA RTX 4090 has more than 16,384 CUDA cores, design to accelerate parallel processing tasks. It's generally considered essential for training deep learning models, large datasets, and high-dimensional computations and can dramatically speed up the training time for deep learning models, usually by one of two orders of magnitude.

CPUs comes in different core architectures, which is the underlying design and structure of a CPU's processing core, which is responsible for executing instructions and performing computations in a computer. Different core architecture have different instruction set architectures (ISAs) and optimizations and special support for AI-specific instructions. The ISA defines the set of instructions the CPU can execute, such as arithmetic operations (add, subtract, etc.), memory operations (load, store), and control operations (branching). Common ISAs include x86, used by Intel and AMD processors, and ARM, used in mobile devices and some SBCs such as the Raspberry Pi. Intel processors usually have better single-core performance than AMD processors, but they also have special features such as DL Boost. Deep Learning Boost (DL Boost) introduced in recent Intel processors is designed to accelerate AI workloads, especially inference, by optimizing low-precision calculations (INT8). This feature is useful in speeding up inferencing for ML models like neural networks. On the other hand, AMD usually offer more cores and threads at a lower price point than Intel, and are often the choice for embedded or mobile devices. More cores can be helpful when training ML models, especially for large datasets and parallelized workloads. AMD does not have a direct analog to Intel's DL Boost, however.

1.3 Motivation

Single board computers (or SBCs) are complete computers built on a single circuit board, integrating components like processor, memory, and storage. They are compact, cost-effective, and easy to integrate into various applications. Some notorious examples are the Raspberry Pi series, or the Arduino series, which are most commonly used for embedded devices, such as home security, weather stations or small drones. These devices gained popularity during the last decade due to the aforementioned benefits, fostering a rich and active community of developers, and created many popular community projects such as the CinePi, which is a open source high-end cinema camera using a Raspberry Pi [?] or the Pi-Hole, which is a network-wide ad and tracker blocker [?].

Pairing vision-based hand gesture recognition to single-board computers and other embedded devices offers a lot of advantages compared to other human-machine interfaces such as a keyboard or wireless mouse. First, single-board computers are compact, and often needs to remain so making vision-based gesture-controlled systems portable and easy to integrate, as it only requires a camera which may already be part of the setup (in the case of smartphones for example). There is no need to add extra hardware in most cases, whereas keyboards, mouses and other peripherals are very often not included in a basic SBC setup, and are often more cumbersome than the computer itself. A typical camera will usually cost around the same as a keyboard and mouse combination. Taking the Raspberry Pi for example, with the Raspberry Pi Camera Module 3 : The standard version (12MP) is around \$25–\$35, the wide (12MP, with a wider field of view) is around \$35–\$45, and finally, with autofocus or noIR (infrared for night vision), it is slightly higher in price, around \$30–\$45, depending on the configuration. These prices may vary depending on the retailer, shipping costs, and region. Online platforms like Amazon, Shopee, or official Raspberry Pi resellers usually stock these products. The price range for a wireless keyboard or mouse combo is around \$20-\$40 for a very basic setup, which is arguably more cumbersome to use, and each device needs to be recharged individually. The keyboard (and even the mouse in most cases) is very often bigger than the SBC itself, so the portability of the setup is dramatically reduced. Furthermore, Open-source platforms like Raspberry Pi allow for extensive customization, making them ideal for DIY projects, experimentation, and specialized applications. If designed properly, it would be easy to customize HGR software to map new commands to new or existing hand gestures, making it very flexible and adaptable to the user. Leveraging low-cost hardware, these systems can be employed in both hobbyist and professional environments, creating solutions that are affordable, portable, and often energy-efficient.

Hand gesture recognition holds significant promise in improving accessibility for individuals with disabilities, particularly those who have difficulties with traditional input devices like keyboards or touchscreens. This technology might be more suited for elderly people and people suffering from age-related mobility or coordination challenges. As motor skills decline with age, larger and simpler gestures can be easier than using small keycaps or touchscreens. Pain and limited dexterity from arthritis make precise movement difficult and painful, but gestures can be a simpler alternative. For people with movement disorders that involve tremors or unintentional movements, gesture recognition can be adapted and can differentiate between intentional movements and tremors, allowing individuals to control devices without false inputs. Individuals with cognitive disabilities, including those who may struggle with traditional interfaces, can benefit from gesture-based interaction, which can be more intuitive and accessible. Some individuals on the autism spectrum, especially when nonverbal, may find gestures easier to use for communication and interaction compared to text or speech-based systems, and it can also be useful for individuals struggling with dyslexia or other learning disabilities. HGR can provide alternative input methods that reduce reliance on written or typed text, making digital interaction more accessible. HGR can also be made to recognize sign language and translate it into spoken or written communication, bridging the gap between deaf individuals and those who don't know sign language.

Vision-based hand gesture recognition also offers some interesting ventures and possible applications that can extend the usefulness of single-board computers and other embedded devices. Hand gestures can be used to control various IoT devices such as lighting or air conditioning. For example, waving your hand can turn lights on/off ,adjust brightness or activate remotely several smart home appliances such as ovens or laundry machines, providing a seamless way to interact with home automation systems. Possible uses can extend outside homes, and into security systems. SBCs can be used to enable gesture-based access control, such as unlocking doors or triggering alarms based on specific hand movements. Users could control TVs, projectors, or audio systems through gestures, for example by swiping left or right to change tracks or adjusting the volume by moving the hand up or down. Finally, In industrial settings, hand gesture recognition can allow workers to interact with cobots, instructing them to perform specific tasks through simple gestures, enhancing efficiency and safety. Cobots, or collaborative robots, are robots designed to work alongside humans in shared workspaces, assisting with tasks while ensuring safety through sensors and other safety features. They are typically used to enhance efficiency in industries like manufacturing without replacing human workers.

1.4 Research objective

This research will try to answer several questions related to vision-based hand-gesture recognition on CPU-only, or CPU-focused environments, that will hopefully lead to further research and development on this topic. - CNN of SVM? CNN (explain why)

- Details on the optimization techniques

- Explore differences between sensors Then, this thesis will explore the differences between two sensors

- Difference between chip architectures Finally, I will try to highlight the subtleties and possible optimizations and performance gains between two different CPU microarchitectures, namely the Intel CPU microarchitecture and the AMD CPU microarchitecture.

2 Literature review

2.1 Research methodology

- Research methodology, selection criteria and resources used for reproducability

The following key-words were selected for the research, some words were only search in context, and in relation with other terms :

- ”real-time”
- ”low-power”
- ”CPU”
- ”image recognition”
- ”hand gesture recognition” and its abbreviations
- ”vision-based”

The following key-words were excluded from the research :

- ”electromyography” and its synonyms and abbreviations. While a popular hand gesture detection method, it is not vision-based and not related to this topic.
- ”glove” or ”glove-based”

2.2 Data acquisition

First, choosing a data source, which will be used to capture the hand gestures, is primordial. For vision-based hand detection, the use of cameras is very common. Typically, RGB cameras, depth cameras, infrared (IR) sensors, or specialized devices like the Microsoft Kinect or Leap Motion are used. These can capture hand movements in 2D (RGB) or 3D (depth or IR). For vision-based detector using a standard camera, the detection can be simplified if we can assume the hand covers most of the screen and is easily distinguishable for its surroundings by its color, for example using the YCbCr skin detection algorithm [**?**]. However, in real-life situations you cannot expect for this to always be the case (for example: the user might put his hand in front of his face, or be in front of someone’s else). Some papers recommend the use of sensors with depth perception capabilities [**?**] [**?**].

There are various ways to accomplish depth perception:

- Binocular cameras, which imaging devices that utilize two camera lenses to capture images or video from slightly different angles, mimicking human binocular vision. This setup allows for the creation of three-dimensional (3D) images or videos by simulating depth perception. The dual lenses can be arranged in various configurations, depending on the intended use, and the resulting imagery can provide a more immersive experience compared to traditional single-lens cameras. In the case of hand gesture recognition, this can help differentiate a hand from the rest of the body in the background, however since those are often RGB cameras, the skin detection can prove hazardous when the hand is at the same depth as the user's body. A noteworthy example would be the Leap Motion Controller 2, made by Ultraleap [?] which uses two monocular IR sensors to capture depth.

- RGB-D Cameras, which are imaging devices that capture both color (RGB) and depth (D) information in a single frame. They combine traditional color camera capabilities with depth-sensing technology, allowing for the creation of 3D representations of the environment. The depth data can be acquired using methods such as structured light or time of flight (ToF), enabling the camera to understand the spatial relationships and distances between objects. Structured light depth perception is a technique used to capture depth information by projecting a known pattern of light onto a scene and analyzing how this pattern deforms when it encounters surfaces. The changes in the projected pattern allow the system to calculate the distance to various points in the scene, creating a depth map. Time of Flight (ToF) depth perception is a technique used to measure distances by calculating the time it takes for a light signal—typically infrared light—to travel from a sensor to an object and back again. This method enables the creation of detailed depth maps, which are essential for understanding the three-dimensional structure of a scene. The Microsoft Kinect, developed by Microsoft, used the structured light approach for its first iterations when released along the Xbox 360, while newer versions of the device implemented ToF depth perception.

The frame rate of the camera, which will affect the complexity of the data stream and real-time performances, is also an area where special precaution must be taken. The frame rate must be high enough to capture the movement accurately, usually 30 frames per second (FPS) or higher, depending on the application, but low enough so that it does not make real-time detection impossible.

2.3 Preprocessing

This step helps prepare the raw visual data for gesture classification, ensuring that the model can efficiently recognize gestures with greater accuracy. It removes noise, distortions, or irrelevant information to make the data cleaner and more reliable for further processing, it simplifies the input data to reduce computational load and make pattern recognition easier and standardizes the input to ensure consistent and reliable feature extraction, leading to better recognition performance. For real-time applications, preprocessing is paramount, as exposing a non-processed video stream for complex pattern recognition and making a prediction in less than a second (which would be necessary for a comfortable user experience) might be computationally unfeasible for the system.

2.3.1 Segmentation

Segmentation is a critical preprocessing step in hand-gesture recognition systems, as it isolates the region of interest (the hand) from the background. This is essential because it ensures that only the relevant features of the hand gesture are extracted and processed by the recognition algorithm. A robust segmentation process enhances the accuracy and efficiency of gesture recognition by simplifying the complexity of the scene and removing unwanted noise. Segmentation helps by reducing computational complexity, first by focusing only on the segmented hand region, it reduces the data the system needs to process. It also improves feature extraction: Accurate segmentation ensures that only the hand features (like edges, contours, and textures) are extracted, leading to more reliable recognition results. Finally, a well-designed segmentation method can adapt to different hand sizes, positions, and skin tones, dramatically increasing accuracy and make it more viable for everyday uses.

Several segmentation techniques can be used in hand-gesture recognition systems :

- Skin color-based segmentation is one of the earliest and simplest techniques for hand segmentation. It relies on identifying pixels in the image that match the color of human skin and separating them from the rest of the scene. The image is converted to a color space that is better suited for distinguishing skin tones. HSV (Hue, Saturation, Value) can be used and is typically more robust against lighting variations than RGB. YCrCb (Luminance, Chrominance) which is netter at separating chrominance (color) from luminance (brightness), is also commonly used. A predefined range of skin color values is used to threshold the image, creating a binary mask where skin-colored pixels are marked as the foreground (hand), and everything else is background. Minimal computational resources are required, making it suitable for real-time applications, and it works well in environments with stable lighting conditions and simple backgrounds. However, skin color changes with different lighting, which can lead to poor segmentation in varying environments, and people with different skin tones, or wearing gloves, might not be segmented correctly unless the algorithm is calibrated to account for the full range of human skin colors. Also, the edge extraction might be inaccurate, resulting in a decrease in accuracy without subsequent processing.

- Background subtraction involves segmenting the hand by comparing the current frame with a reference background. This method assumes that the background remains static while the hand moves in front of it. A background model (a reference frame without the hand) is either captured beforehand or built dynamically. For each new frame, the pixel-wise difference between the current frame and the background is computed. A threshold is applied to the difference image to identify foreground objects (in this case, the user's hands), generating a binary mask where foreground pixels are marked. Obviously, it works extremely well in environments and controlled situations where the background remains constant (for example in front of a TV or in a particular room, the background is not expected to change). Therefore, it is more suited for indoor uses, such as gaming or some smart home applications. It is also relatively fast and can be implemented with simple image differencing, making it useful for real-time applications. However, when faced with dynamic backgrounds, typical of outdoors situations, or uneven backgrounds resulting in shadows, reflections or glares, it is not applicable as it can severely impact the accuracy of the system. It is not as flexible as other techniques and requires specific, controlled environments.

- Depth-based segmentation utilizes depth information from cameras like the Microsoft Kinect or Intel RealSense, which capture not only the color (RGB) of each pixel but also the distance of each pixel from the camera. The depth camera provides a depth map where each value corresponds to the distance between the object and the camera, on top of pixel color information. A depth threshold is set to segment objects that are within a specific distance range. Usually we assume that the user will put his hand forward, pointed at the sensor, in order to input commands. The hand is then separated from the background by selecting the pixels that fall within the defined depth range. Since it relies on infrared light, it is resistant to conditions where luminosity may be variable and works well even in low-light or high-contrast environments since depth data is independent of color. When used correctly, this technique can precisely segment objects based on their distance from the camera, reducing errors from background clutter or shadows. In addition, with efficient hardware, depth-based segmentation can be performed in real-time. It's a popular technique that is used in a wide variety of devices. Of course, since it requires specialized depth sensors, it may increase the cost and complexity of the system, and is not as readily accessible as other techniques relying solely on visible light.

- Edge and contour-based segmentation uses edge detection algorithms to identify boundaries between the hand and the background. Once edges are detected, contour detection is applied to outline the hand. Edge detection algorithms (such as Sobel, Prewitt, or Canny edge detectors [**?**]) are applied to the image to identify areas with high-intensity gradients, which often correspond to object boundaries. Contours are extracted by grouping edge pixels into continuous lines or curves, representing the hand's boundary. The largest or most relevant contour is selected as the hand region, and the rest of the image is discarded. It excels in shape-based recognition and works well in applications where the shape of the hand is important, such as very distinct hand signs (flat palm, closed fist) or when certain fingers are raised, such as hand signals representing numbers. Edge detection can be affected by noise in the image, leading to incomplete or false edges, and only performs well when there is a strong contrast between the hand and the background, which may not always be the case in real-world environments.

- Machine learning techniques, particularly deep learning, have revolutionized segmentation in recent years by learning to automatically segment the hand from large datasets of labeled examples. A model, for example a convolutional neural network is trained on a dataset of labeled images where the hand region has already been segmented. The model learns to predict the segmentation mask directly from input images, classifying each pixel as belonging to the hand or background. The trained model is then used to segment the hand from new, unseen images, automatically. Deep learning models can achieve very high segmentation accuracy, especially when trained on diverse datasets, leveraging ata augmentation techniques. It can handle a wide range of skin tones, lighting conditions, hand poses, and complex backgrounds, and is capable of segmenting hands in different environments and contexts, more than any other technique, provided there is sufficient training data. As this technique relies almost exclusively on the quality of the dataset, often requiring a large and very diverse dataset of hand images for training, which can be time-consuming to acquire and to train the model with without proper equipment. Deep learning models are computationally expensive to train and run, especially for real-time applications. Optimization is therefore extremely essential for this technique. Finally, the model may overfit to the specific data it is trained on, leading to poor generalization on unseen scenarios without proper regularization or data augmentation.

In addition, there exists different techniques of gesture tracking, specific to dynamic hand gesture recognition. They often rely on several frames over a video feed, as those hand gestures are expressed temporally. Here is a quick overview of different techniques :

- Optical flow tracks the motion of pixels between consecutive video frames to detect movement patterns. As it captures the whole scene, it can adapt to a wide variety of scenarios, but lacks accuracy in scenarios with dynamic backgrounds or fast hand movements.
- The frame differential method compares consecutive frames and segments regions where pixel values differ significantly. This method is easy to implement and has low computational complexity. However, non-static background can lead to false positives.
- The Kalman filter technique predicts the next position of the hand based on past movement, correcting for noise and small disruptions. It's efficient for tracking smooth, continuous gestures, but not as efficient with erratic or fast movements and is sensitive to large jumps in motion.
- The MeanShift technique, which is a simple, fixed-size region tracker that shifts the search window based on the highest feature density. It is suitable for scenarios where object size is static. It is relatively fast and boasts decent real-time performance.
- The CAMShift (Continuously Adaptive MeanShift) algorithm is an adaptive version of MeanShift that adjusts the search window size based on object size, making it better for tracking objects that change in scale, in the case of hand gesture recognition, when the hand is closer or farther from the visual sensor. While it is more accurate overall, it is more computationally intensive and might not be suited for scenarios where the user might stay at the same place, reducing the size variations of the hands.
- The particle filtering, a probabilistic method that tracks objects by maintaining multiple hypotheses (particles) about the object's location. It's robust to noise, occlusions, and non-linear object motion, but computationally more intensive.

2.3.2 Optimization techniques

Reducing computational complexity in vision-based hand gesture recognition is critical, especially when dealing with real-time systems that must process data efficiently. Preprocessing techniques play a major role in simplifying the data and reducing the amount of computation required without compromising the recognition accuracy. Below is a list of possible techniques that can be used on their own or in conjunction with other techniques.

- Reducing the size of the input image can significantly lower the computational load since the amount of pixel data decreases. For example, processing a 640x480 image is much faster than processing a 1920x1080 image. It is often the first step of image pre-processing done. It can be done very efficiently using simple interpolation methods like bilinear or nearest-neighbor. Of course, while resizing lowers complexity, it can also reduce the level of detail in the image, which may affect the accuracy of gesture recognition. For this technique, balance is key between accuracy and computational load, and the type of sensor, and expected distance between the user and the sensor must be considered.

- Region of Interest (ROI) cropping is also an important pre-processing step, that can also be done early. Instead of processing the entire frame, only the region of interest is processed, drastically reducing the number of pixels to analyze. After detecting the hand through techniques like background subtraction or motion detection, a bounding box is drawn around the hand, and the rest of the image is discarded. While this technique is more complex than image rescaling, this significantly reduces the amount of data to be processed, while also discarding irrelevant data, allowing for faster analysis and recognition.

- Grayscale conversion, also called image graying, converts an image from RGB (three channels) to grayscale (one channel), reducing the amount of data to process by a factor of three. While losing color information, which means it cannot be used with certain segmentations techniques that rely on skin color, this step preserves critical details like edges, contours, and hand shape, which are sufficient for gesture recognition in most cases. It is typically applied after image acquisition and before other operations like edge detection or segmentation.

- Background subtraction eliminates unnecessary parts of the image that are irrelevant to hand gesture recognition, leaving only the hand for further processing. It can be done very early in the pre-processing stage, just after image acquisition. Frame differencing which compares the current frame with a background model or a reference frame, or Gaussian Mixture Models (GMM), which creates a statistical background model and subtracts this from the current frame to isolate the hand, can be used for this process. Additionally, if the sensor has depth perception (RGB-D, or binocular), this step can be done extremely fast and easily.

- Image thresholding converts the image into a binary format (black and white), where the hand is represented as a white region and the background is black. This drastically reduces the data complexity and makes further processing (like contour or edge detection) faster. Thresholding can be either global (a single threshold value is applied to all pixels, converting them to either black or white) or adaptive (The local thresholding value varies depending on local pixel density, making this technique more useful in varying lighting conditions). This technique can be considered a more extreme version of grayscale conversion. Binary images are far less complex to process since each pixel contains only one bit of information (black or white), as opposed to grayscale or RGB images. This step is best used after background subtraction or segmentation, and before shape/contour analysis.

- Dimensionality reduction aims to reduce the number of features or dimensions to lower the computational load for further processing, such as classification. It usually leverages two main techniques : PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis). CA is a technique used to reduce the number of features or dimensions in a dataset while preserving as much information as possible. It does this by transforming the data into a new coordinate system where the new axes (called principal components) are ordered by the amount of variance (information) they capture from the original data. The first principal component captures the most variance, and the subsequent ones capture less. Meanwhile, LDA is a technique used for both dimensionality reduction and classification. Unlike PCA, which focuses on maximizing variance, LDA tries to find the feature space that best separates different classes (categories) in the data. It projects the data into a lower-dimensional space where the separation between different classes is maximized. It is best used after feature extraction, especially when dealing with high-dimensional data like feature vectors from images.

- Morphological operations like erosion and dilation are used to clean up binary images by removing noise (small unwanted pixels) or filling gaps in segmented hand images. Erosion is a morphological operation that shrinks or ”erodes” the boundaries of foreground objects (usually white regions in a binary image). It works by removing pixels from the edges of objects. Dilation is the opposite of erosion; it grows or ”dilates” the boundaries of the foreground objects. It works by adding pixels to the edges of objects. These operations simplify the hand shape, making it easier to process, without needing to analyze complex pixel data. It’s often used after segmentation or thresholding in order to clean up the image.

- Downsampling is critical for real-time applications. As not every frame is necessary, especially for static hand gesture recognition, reducing the number of frames lowers computational requirements while maintaining gesture information. This is done directly after capturing the video feed.

- While primarily a training technique, data augmentation can help reduce overfitting, which can indirectly reduce the need for large-scale computations during model inference. Rotation, flipping, and scaling of training data can simulate different viewpoints or hand sizes, reducing the need for a massive dataset and complex models. It allows simpler models to perform well on complex tasks, reducing the computational load during inference.

- Feature selection comes as an extra step after feature extraction. After feature extraction, not all extracted features are important. Feature selection techniques aim to select only the most relevant features, reducing the complexity of the recognition model and the classification process. Popular techniques used for feature selection includes correlation-based feature selection. also referred as CFS, and mutual information. Correlation-based feature selection (CFS) is a feature selection method that evaluates and selects features based on their correlation with the target class and their correlation with other features. CFS selects features that are highly correlated with the target (outcome) variable and minimally correlated with each other. This way, it chooses features that provide unique, relevant information about the target without redundancy. In a dataset with multiple redundant features (for this application, the wrist or arm is extremely redudant and useless for classification), like different measurements of the same property, CFS would help identify the most important and independent ones. Mutual information calculates the amount of information a feature shares with the target variable. Features with high mutual information with the target are selected, as they provide significant insights for predicting the outcome. In a classification problem, mutual information can help identify which features contribute the most to classifying samples correctly by quantifying their relevance.

2.4 Gesture recognition and classification

Gesture classification is the process of identifying and categorizing specific hand movements into predefined gesture classes or categories. - machine learning vs deep learning

- Several methods exist to detect the hand : by skin-color, by edge, using depth, using deeplearning..

This paper proposed by Sahoo et al. [**?**] seems to suggest that developing a CNN from scratch might be too tedious, time-consuming, and have diminishing returns compared to fine-tuning a pre-existing image classification model to focus on hand gesture detection. The proposed techniques achieved between 5.85% and 8.01% increase in mean accuracy compared to similar models.

2.5 Object detection

Highlight the various techniques to detect patterns (objects) in larger images

2.6 Training

-Talk about dataset

-Optimizers

-Data augmentation

2.7 Research gaps

- No specific research on vision-based HGR for CPU-only environments

- No particular study explores the possible differences between various processor architectures (Intel, AMD)

3 Methodology

3.1 Experiment design

This experiment will consist of benchmarking several pre-trained KerasCV convolutional neural networks, varying in datasets used for training, optimization and detection techniques. Those convolutional neural networks will also be benchmarked on different hardware, namely a mid-range Dell Latitude laptop and a Raspberry Pi Model 4B, in order to highlight the possible differences in accuracy or overall performance between the two architectures, and the practicality of running local deep learning networks for hand gesture detection on low-power devices, in real time.

The Python project will make use of several libraries. Here's a detailed breakdown of each major library used for the project:

- TensorFlow
- Keras
- KerasCV
- OpenCV, used for live camera feed and image preprocessing.
- MatPlotLib
- sci-kit (?)
- pandas (?)

TensorFlow library, the pandas library, the sci-kit library as well as the MatPlotLib library for graph plotting. In order to test performances over a live webcam feed, OpenCV will also be used. (describe each lib precisely and provide links)

- Define each project, Julia and Python (Do I have the time for Julia?)
- Preprocessing with the precise actions
- Many different models will be pre-trained with different configurations?
- Architecture of the model used, with explanation for each layer
- Fine-tuning
- Optimization techniques tested
- Testing procedure

The projects will be benchmarked on both Intel and AMD processors, particularly a Dell Latitude 7490 (Intel i5-8350U with 8 cores @ 3.6 GHz) and a Raspberry Pi Model B (Quad core Cortex-A72 ARM v8 64-bit SoC @ 1.8GHz), in order to gauge the impacts of each optimization techniques on both architectures.

The results will then be compared to pre-trained general object detection models such as YOLOv3.

3.2 Data extraction

4 Results

5 Conclusion

6 Future research

7 Appendix 1 : Hardware specifications

- Benchmarking notes, RAM, etc.. (also wattage) for both laptop and raspberry pi
    - Sensors used