Smart Gesture-Based Device Control System.

Shaik Sadik (Student)  
Department of CSE  
GMR Institute Of TechnologyRajam, Andhra Pradesh, India.  
shaiksadik954@gmail.com

Shaik Sumsuddin (Student)   
Department of CSE  
GMR Institute Of TechnologyRajam, Andhra Pradesh,, India.  
sumsuddin28s@gmail.com Adarsh Sailaka (Student)  
Department of CSE  
GMR Institute Of TechnologyRajam, Andhra Pradesh,, India.  
sailakaadarsh@gmail.com

Gangu Dharmaraju(Assistant Professor)  
Department of CSE  
GMR Institute Of TechnologyRajam, Andhra Pradesh,, India.  
dharmaraju2606@gmail.comVuriti Varshini(Student)  
Department of CSE  
GMR Institute Of TechnologyRajam, Andhra Pradesh,, India.  
varshinivuriti38@gmail.com

*Abstract*—With the growing need for touchless interaction, gesture-based control systems have emerged as promising solutions for tasks like mouse navigation, key presses, and virtual typing. While these systems have shown notable progress, most existing approaches focus on individual tasks rather than offering a unified framework, resulting in fragmented and inconsistent user experiences. Additionally, many of these solutions rely on computationally heavy models that are unsuitable for real-time applications, especially on low-power devices. This project aims to overcome these challenges by developing a lightweight and integrated system that combines multiple functionalities —such as mouse control, clicking, and typing—into a single, seamless interface. The proposed system leverages deep learning techniques to track hand movements and identify keypoints, enabling smooth and accurate gesture recognition. It is optimized for performance under varying lighting conditions and hand orientations while remaining efficient enough to run on resource constrained devices. Beyond basic controls, the project also explores additional features such as gesture-based typing to enhance usability and convenience. The expected outcome is a practical and responsive solution that simplifies human-computer interaction, offering a contactless alternative to traditional input devices. Its versatility makes it suitable for a wide range of applications, including assistive technologies, smart home systems, and virtual reality interfaces, providing an intuitive and accessible way to control devices through gestures.

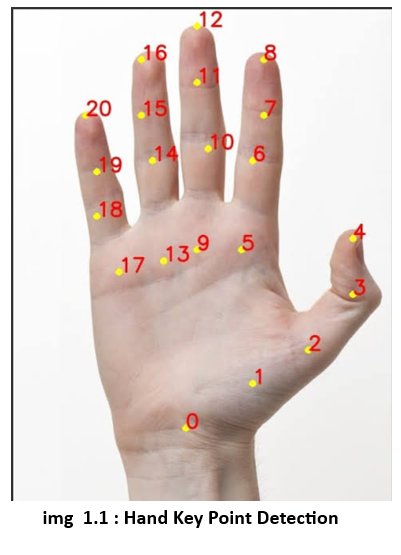
*Keywords— Motion Tracking, Object Detection , Feature Extraction , Hand Keypoint Detection , Gesture Recognition , Image Processing.*

# Introduction

* 1. Over the past few years, gesture control systems have drawn considerable interest as an intuitive, touch-free solution to conventional input devices. The growing need for contactless interactions, fueled by developments in artificial intelligence and deep learning, has spurred the creation of several solutions for tasks like mouse navigation, virtual typing, and key presses. Most current solutions are narrow in scope, focusing on a single task instead of offering a unified and complete framework. This lack of integration causes uneven user experiences and diminishes the overall practicality of gesture-based interfaces. Additionally, one of the key challenges in real-time gesture recognition is the computational complexity of deep learning models, which renders them unsuitable for deployment on low-power devices. High processing demands tend to create latency problems, restricting responsiveness and usability. To overcome these issues, this project suggests a light and integrated system that can manage multiple functionalities—mouse control, clicking, and typing—under one seamless interface. By using deep learning methods for hand tracking and keypoint detection, the system provides smooth and precise gesture recognition. It is optimized to work under different lighting conditions and hand orientations with minimal computational overhead. In addition, the project investigates advanced features like facial recognition for user login and gesture typing to enhance accessibility and ease of use. The solution proposed seeks to make human-computer interaction more natural and efficient by simplifying it. Its flexibility enables it to be used in assistive technologies, smart home automation, and virtual reality applications, offering a convenient and accessible alternative to traditional input.

# Related Works

To have a clearer understanding of the methodology and workflow involved in the project, we believe that it is important to learn the following concepts prior to the literature study:

* 1. **Hand Keypoint Detection** : To identify the gesture demonstrated by the user and then translate that to a specific action, we require a model that is capable enough to detect the user's hand key points efficiently and then based on the state of the user, which can be deduced from the keypoint information, we can determine the gesture and then translate it to a specific action. A number of hand keypoint detection methods have been investigated in previous work, from heavy deep neural networks to light real-time systems. One of them is OpenPose [1], which employs a multi-stage CNN pipeline to detect 2D pose and hand keypoints. Although OpenPose has high accuracy and can detect multiple hands in dense scenes, it is computationally expensive and less ideal for real-time applications on resource-limited devices. Likewise, HRNet [2], which retains high-resolution representations across the network, reports state-of-the-art performance on hand pose benchmarks. Yet, its inference time and resource usage are more appropriate for offline scenarios. Another notable approach is the MediaPipe BlazePalm + Hand Landmark Model, which we eventually used. But prior to this finalizing, we considered VNect [3]'s Hand Keypoint Detection via Pose Estimation model, which predicts 3D human pose from a monocular RGB image. While useful for whole-body tracking, its accuracy on fine-grained hand movements was poor.More current work such as Keypoint R-CNN [4] (an expansion of Mask R-CNN), also performs hand keypoint estimation but also adds a lot of latency to the system, particularly when on non-GPU hardware. Likewise, Graph-Based Networks like GCNs used on hand skeletons have shown strong occlusion robustness but need preprocessed skeletal data, which makes real-time integration cumbersome .After comparing these and taking into account constraints such as latency, accuracy, hardware support, and ease of integration, Mediapipe's hand detection pipeline proved to be the most efficient and robust solution. It delivers real-time performance with robust tracking even on CPU-only environments, and provides a simple abstraction for recovering hand landmarks (21 keypoints) without deep model fine-tuning or the need for a GPU.
  2. **Object Detection :** : Object detection is useful in the first step of our integrated hand gesture control system by facilitating person localization in the camera frame. This step makes the system process only on the proper cropped area where the user is, thus improving the accuracy of the following hand keypoint detection and saving unnecessary computational burden. Precise and effective object

detection enables strong performance even in cluttered backgrounds or multi-person scenes, and thus it is a building block for reliable gesture-based control.

Over the past few years, many object detection frameworks have been introduced, mostly belonging to two broad categories: CNN-based detectors and transformer-based detectors.

* 1. **DETR (DEtection Transformer** [1], proposed by Facebook AI, was a major departure from conventional object detectors by employing transformers for end-to-end object detection. DETR eliminates the requirement of handcrafted parts such as anchor boxes and NMS (non-max suppression), and it has a conceptually clean design. Nevertheless, its inference latency is significantly worse, and it needs large-scale data and extensive training time, which makes it less ideal for real-time applications, particularly for low-resource platforms such as CPU-only systems.
  2. Conversely, **YOLO (You Only Look Once)** models, most notably the newly introduced YOLOv11, have also shown impressive real-time object detection efficiency. The models balance between speed and precision by employing CNN architectures optimized for dense detection. Nonetheless, although they are efficient, even light-weight versions such as YOLOv11n require a lot of GPU power to run smoothly, and execution on typical CPU configurations (such as laptops or embedded systems) can result in perceptible lag or frame drops, which is not good for real-time gesture control. Alternative CNN-based detectors like SSD (Single Shot Detector) and Faster R-CNN offer high accuracy but suffer from either reduced frame rates or increased complexity during deployment. Lightweight versions of these models have also been explored but often compromise on localization accuracy, especially when the scene contains multiple subjects or low lighting conditions.

Due to the performance constraints of these widely used object detection pipelines in our real-time, CPU-intensive environment, we chose Mediapipe's BlazePose + Region of Interest (ROI) tracking pipeline for person detection.

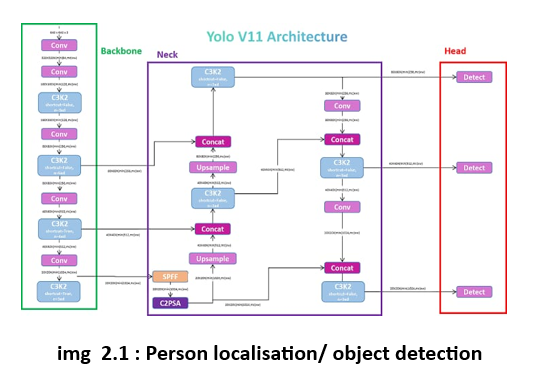
Though not a general-purpose object detector, it is very well optimized for tracking a human body and gives a tight bounding box over the detected subject with low latency. The model can run at real-time frame rates even on CPU, without the explicit need for GPU acceleration or external dependencies. This selection enables us to lower computational overhead, prevent flickering or frame stuttering, and provides a consistent user experience, which is particularly important in the case of gesture-based interfaces.

* 1. **Image Denoising :** In actual uses of vision-based systems—particularly those intended for low-power or embedded devices—image quality may be a constraint on performance. Cameras on such devices tend to generate noisy, grainy, or low-resolution images, particularly under low lighting conditions or lower frame rates.

In our gesture control pipeline, this degradation in image quality can have a drastic impact on the performance of both the person localization and hand keypoint detection models, causing misdetections, jitter, or lost gestures. To address this

problem, we investigated image denoising as a preprocessing technique to improve the input frames prior to feeding them to downstream processes. Denoising has been an extensively researched problem in computer vision, and a number of deep learning-based methods have shown excellent performance in noise removal with semantic detail preservation. In the deep learning age, models such as DnCNN [3] proposed convolutional neural networks designed specifically for removing Gaussian noise.

Subsequent to this, FFDNet [4] and MemNet [5] provided better results using noise level maps and memory blocks, respectively.

Recent work has also investigated transformer-based models for denoising, including Restormer [6], which uses self-attention mechanisms to more effectively capture long-range dependencies in noisy images. Although these models achieve state-of-the-art performance on benchmarking datasets such as DND and SIDD, they are computationally expensive, utilizing GPUs and large memory bandwidth that makes them not suitable for our target deployment platform. With our need for low-latency CPU-friendly inference, we decided not to employ these heavyweight pre-trained models. We instead implemented a custom light-weight denoising autoencoder based on the U-Net architecture [7]. U-Net, first suggested for biomedical segmentation, provides a compact encoder-decoder model with skip connections that assists in maintaining spatial details during recovery from noise. We borrowed this idea to construct a Denoising U-Net Autoencoder and, deliberately narrowed its depth and channel widths for the requirements of real-time performance.

# Methodology and Architecture

**3.1 Live input capture using OpenCV :**

The initial and basis element of our system is acquiring the video frames from the user's camera in real-time. This process is vital since it serves as input to the entire gesture recognition and control pipeline.

In order to achieve cross-platform compatibility and low-latency frame grabbing, we used OpenCV (Open Source Computer Vision Library), a popular, open-source computer vision library that offers strong support for webcam and video stream interfacing.

Through OpenCV's cv2.VideoCapture() API, we opened the default system webcam.

This API provides frame-by-frame access to the video stream in real time.

A critical optimization step in our pipeline is the localization of the individual in the camera frame. Instead of using the following detection and control models on the whole frame— which can be computationally costly and noisy—we first localize the user and dynamically crop the frame to concentrate on the area of interest. This makes the downstream hand keypoint detection and the interpretation of the gestures not just faster but also considerably more accurate, particularly in situations where the user is at a distance from the camera or simply takes up a small amount of frame space.

**3.2 Person localisation using media pipe :**

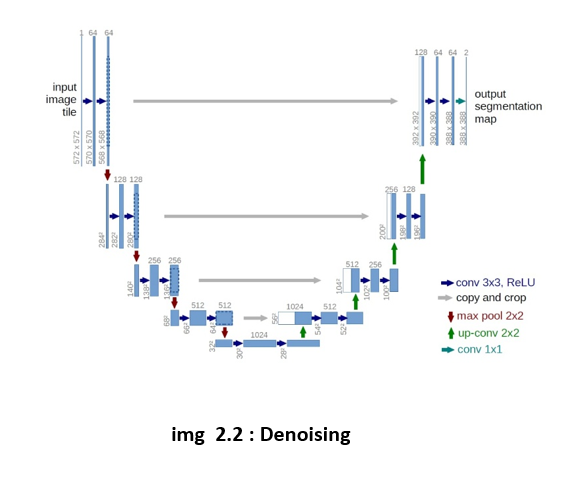
Person localisation using media pipe : During our exploratory phase, we tested three current state-of-the-art person localization models: DETR (DEtection TRansformer), YOLOv11, and Mediapipe's holistic tracking system. Both DETR and YOLOv11 were highly accurate in person detection but were associated with significant drawbacks in the real-time interaction and low power deployment scenarios: • DETR, as a transformer-based model, had high latency due to its self-attention mechanism, rendering it less effective for live processing on CPU-constrained devices. • YOLOv11, while much

quicker than DETR and object-detection-optimized, was also computationally intensive when run consecutively at high frame rates, especially in the absence of GPU acceleration on devices.

Also, its iterative detection resulted in constant and erratic re-cropping of the frame, resulting in a flicker-y visual output and unstable gesture areas.

Eventually, we employed Mediapipe's holistic model, which best balanced speed, reliability, and smoothness. Mediapipe provides a light pose estimation pipeline that detects human body landmarks, including the important joints like shoulders and torso. These landmarks are much more stable than hand gestures or facial points that are dynamic, allowing the model to track the user's location in the frame more consistently. This approach allowed us to design a cropping mechanism that updates only when there's a significant shift in the user's position, thereby avoiding the disruptive effect of constant re-cropping. The resulting experience is visually smoother and operationally more efficient. In contrast to DETR and YOLO, which identify the bounding box of the whole person, Mediapipe's continuous landmark tracking allows us to smartly center and crop the frame around the torso, which is relatively stable in relation to the extremities.

**3.3** **Denoising with custom model :**

Denoising with custom model : After the person localization process, we have a cropped image that concentrates on the user only. While this cropping greatly enhances the speed and accuracy of subsequent gesture recognition models, it brings a new challenge with it—the quality of the cropped image, particularly on low-power devices equipped with low-resolution cameras, tends to be noisy, grainy, or dim. Such visual artifacts can detract from the performance of hand keypoint detection models, which are extremely sensitive to small pixel-level changes in the input. In order to counter this problem, we present a light-weight custom denoising module motivated by the design of the U-Net architecture. The intention of this module is to improve the visual quality of the cropped area in real-time without any added substantial computational burden. U-Net, initially developed for biomedical image segmentation, is famous for its encoder-decoder design and skip connections, making it extremely powerful for image-to-image translation problems, such as denoising.

Our model is a denoising autoencoder. It learns to denoise a clean representation of a noisy image by minimizing the difference between the original clean image and the denoised output. Nevertheless, to render this model applicable for real-time use, we perform several architectural optimizations:

• We decrease the depth and the number of filters in the encoder and decoder blocks to reduce latency. • All computations remain completely convolutional, enabling the model to process input images of any spatial extent—this is particularly useful since the size of the cropped image can change depending on the distance of the user from the camera.

• The model makes predictions directly over the cropped image, rather than the entire camera frame, and this reduces computations dramatically and leads to low inference time even on CPUs.

**3.4** **Hand Keypoint detection with Mediapipe :**

After the denoised and cropped image is acquired, the second important step in the system pipeline is hand keypoint detection, which is the foundation for gesture control—both virtual mouse and keyboard. For this, we use Mediapipe's Hand Tracking solution, a light and highly optimized model that does real-time multi-hand keypoint estimation with high precision.

Mediapipe employs a two-stage pipeline which includes:

1. A palm detector to find and crop the region of interest with the hand.

2. A hand landmark model that traces 21 accurate hand keypoints on this cropped image.

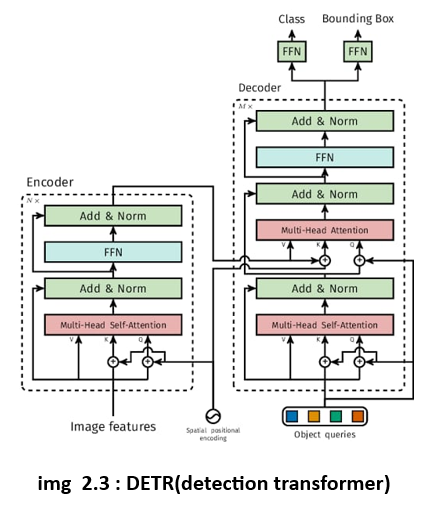
This cascaded structure enables the system to bypass unnecessary computation across the whole frame and concentrate on just the region where the hand has been detected, which has very low latency and rapid inference rates. Thus, the model stays strong even on non-GPU accelerated systems—especially suitable for our intended application scenario of real-time control on low-power hardware.

For our project, these 21 keypoints are utilized to monitor finger motion and calculate useful gestures like:

• Position of index finger for moving the cursor.

• Pinch of index and middle fingers for clicking the mouse.

• Diffract fingers' raised, complex combinations in order to find mode switching or key presses. These motions make up the backbone of the common hand-gesture control pipeline. In an attempt to identify the most appropriate model for this step of the pipeline, we considered several object detection and landmark estimation models such as:

• DETR (DEtection TRansformer), which provides very high accuracy using its transformer based global attention mechanism. Its inference speed is very slow, however, particularly on edge devices, and thus impractical for real-time frame-by-frame keypoint estimation.

• YOLOv11, a recent entry in the YOLO line famous for real-time and high-accuracy object detection. YOLOv11 is great at detecting and localizing hand or person, but it has no native capability for fine-grained keypoint detection and so needs an additional model or a post-processing stage—therefore contributing to latency and complexity.

• Mediapipe which in the end turned out to be the most suitable because of its specialized hand-tracking pipeline that natively outputs precise keypoints with little overhead.

In addition, its tracking process is optimized to preserve temporal consistency between frames, with much less jitter and more reliable gestures.

With these factors in mind, we determined that Mediapipe provides the optimal balance between speed, accuracy, and deployment simplicity.

Its capability of keeping hand keypoint tracking stable over time even in moderately noisy environments makes it perfect for use in gesture recognition systems.

Smooth control is enabled by the consistency of landmark detection, which is crucial for natural user interaction.

**3.5 Gesture Mapping using pynput :**

To convert recognized hand landmarks into actions in the physical world, we add the Pynput library, which exposes high-level abstractions for the simulation of mouse and keyboard events. This provides a connection from gesture recognition to user control where the hand serves as a generic input device. After Mediapipe outputs the 21 significant landmarks of the recognized hand, we utilize the coordinates of certain fingertips and joints to infer useful gestures. These gestures are then translated to move the mouse cursor, click, or type on a virtual keyboard displayed on screen.

Mouse Control through Right Hand Gestures : In Mouse Mode, we employ the following mapping strategy: • Cursor Movement: The system keeps tracking the index fingertip (landmark 8) of the user's right hand at all times. Its location within the camera frame is interpolated and smoothed to screen resolution, yielding a smooth and responsive cursor movement. A small buffer (frameR) is kept along the screen borders to prevent jitter on the boundaries.

• Left Click Gesture: Upon the approach of the index finger and middle finger (landmarks 8 and 12), i.e., when the Euclidean distance between them goes below a specified limit, it is recognized as a left click. This gesture is natural and reduces accidental clicks in overall navigation.

• Drag or Scroll Extensions : The gesture semantics can be extended to accommodate drag-and-drop or scrolling through the recognition of certain finger patterns or gestures like all fingers folded except index and thumb for drag.

This mapping makes mouse control intuitive and hands-free, allowing users to do most pointing and clicking with minimal training.

Keyboard Control through Virtual Interface : Rather than assigning a distinct gesture to each key— which would be hard to learn, unuitive, and error-prone—we created a more scalable method with an on-screen virtual keyboard interface.

This interface consists of:

• Alphabet Keys (A-Z)

• Number Keys (0-9)

• Navigation Keys (Arrow keys)

• Special Keys such as Space, Backspace, etc.

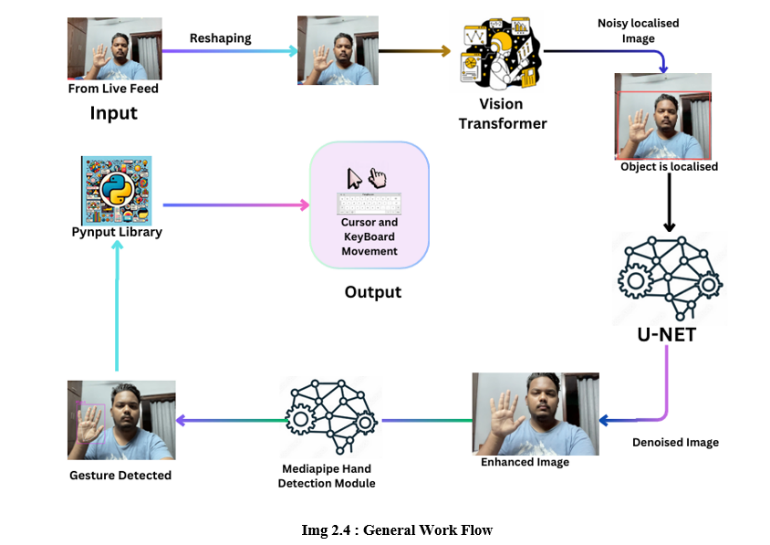
Each key is rendered on the screen as a semi-transparent rectangle with OpenCV. The index fingertip (landmark 8) position is used to hover over keys. When the fingertip enters the bounding box of a key and stays there for a brief period (e.g., 1 second), the key is deemed "selected" and pressed using Pynput's keyboard.press() and keyboard.release() functions.

This approach enables the user to type naturally by tapping at the target key—nearly replicating touchscreen use. It also decreases the cognitive and computational complexity over direct gesture-to-key mappings substantially.

Mode Switching : To switch between Mouse Mode and Keyboard Mode, we introduce a special gesture with the thumb (landmark 4) and pinky (landmark 20):

• If the thumb and pinky fingers are both raised (i.e., their landmark points are above the hand baseline) and all other fingers are clenched, it is taken as a mode toggle gesture.

When this gesture is detected, the mode is toggled:

• From Mouse to Keyboard, the virtual keyboard UI is displayed.

• From Keyboard to Mouse, the screen reverts back to pointer mode.

# Result

**4.1 Outcomes :**

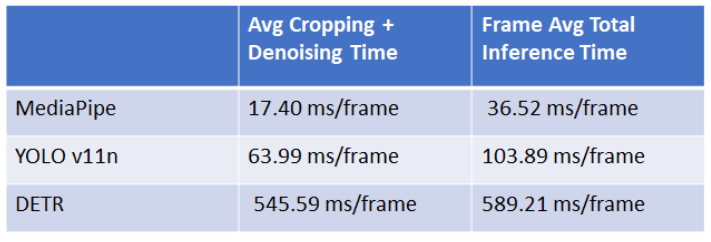
In order to evaluate the effect of various person localization backbones on the performance of our multimodal hand gesture-based input system, we conducted comparative comparisons against three broadly recognized models: YOLOv11, DETR, and Mediapipe. Each one of these models was plugged in separately into our pipeline without changing the rest of the architecture, in an attempt to examine the effect of person localization as a standalone component on downstream hand keypoint detection

**4.2 Evaluation Dataset :**

Our evaluation targets the following most important metrics:

1.Localization Inference Time (ms/frame) Tracks the average time spent by the person localization module in detecting and cropping the person out of the frame. Lower is better.

2.Overall Pipeline Inference Time (ms/frame) The average end-to-end time from raw frame input to keypoint output after denoising and cropping. Lower is better.

**Inference Times of the model with different backbones for localizations:**

3. Keypoint Detection Accuracy Computed as mean percentage of keypoints correctly predicted (w.r.t. Euclidean distance from ground truth). Better is higher.

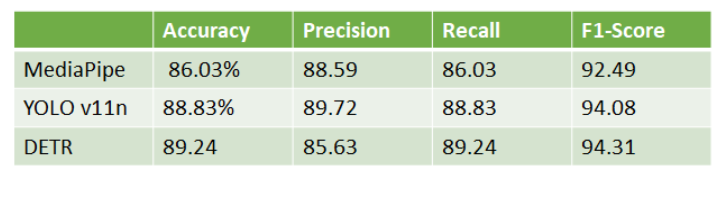
Keypoint Detection Accuracy Computed as mean percentage of keypoints correctly predicted (w.r.t. Euclidean distance from ground truth). Better is higher.

**4.3 Evaluation Metrics :**

Our evaluation focuses on the following key metrics:

1. Localization Inference Time (ms/frame) Measures the average time taken by the person localization module to detect and crop the person from the frame. Lower is better.
2. Overall Pipeline Inference Time (ms/frame) The average end-to-end time from raw frame input to keypoint output after denoising and cropping. Lower is better.
3. Keypoint Detection Accuracy Calculated as the mean percentage of correctly predicted keypoints (based on Euclidean distance from ground truth). Higher is better.

Keypoint Detection Accuracy Calculated as the mean percentage of correctly predicted keypoints (based on Euclidean distance from ground truth). Higher is better.

****

# Conclusion and Future Work

**5.1 Conclusion :**

We have in this project designed and successfully implemented a real-time unified hand gesture based input control system with the ability to mimic both keyboard and mouse functionality.

The system is based on the use of an efficient computer vision and deep learning methods such as person localization using Mediapipe, denoising an image using a compact U-Net-based model, and detecting the hand key points using Mediapipe's landmark tracker in order to build a hardware-independent intuitive input solution.

By means of intelligent gesture mapping via the pynput library and an interactive virtual keyboard overlay, we have provided for smooth interaction with the system by natural hand gestures alone.

One of the main contributions of our work is to maximize the computational pipeline for real-time performance on low-power hardware, i.e., ordinary laptops without special GPUs. Our analysis demonstrates that applying Mediapipe for person localization really improves performance, both in inference speed and in keypoint detection accuracy, and hence is very well-suited for interactive use cases.

**5.2 Future Work :**

Though the existing system shows strong performance in single-user scenarios, it lacks strength in terms of operation in multi-user settings. As part of future work, we will incorporate facial recognition and re-identification methods to selectively monitor and react to gestures from a designated authorized user, even when there are several users in the frame.

##### References

1. T. Y. Kim, A. Niaz, J. S. Choi and K. N. Choi, "One-Stage Detection Model Based on Swin Transformer," in IEEE Access, vol. 12, pp. 60960-60972, 2024, doi: 10.1109/ACCESS.2024.3393152.
2. Miah, A.S.M., Hasan, M.A.M., Tomioka, Y., & Shin, J. (2024). Sign language recognition using graph and general deep neural networks. IEEE Open Journal of the Computer Society, 5, 144–155
3. Hu, Dunli & Zhu, Jun & Liu, Jiayu & Wang, Jiaju & Zhang, Xiaoping. (2022). Gesture recognition based on modified Yolov5s. IET Image Processing. 16. 10.1049/ipr2.12477.
4. Sánchez-Brizuela, G., Cisnal, A., de la Fuente-López, E. et al. Lightweight real-time hand segmentation leveraging MediaPipe landmark detection. Virtual Reality 27, 3125–3132 (2023).
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. International Conference on Learning Representations (ICLR)
6. Kumar, A., Singh, R., & Gupta, P. (2023). Computer vision-based hand gesture recognition for human-robot interaction. Complex & Intelligent Systems.
7. O. Köpüklü, A. Gunduz, N. Kose and G. Rigoll, "Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks," 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019), Lille, France, 2019.
8. Carion, Nicolas, et al. "End-to-end object detection with transformers." European conference on computer vision. Cham: Springer International Publishing, 2020.
9. Sarma, D., & Bhuyan, M.K. (2024). Real-time hand gesture recognition. Cybernetics and Information Technologies, 24, 167–182.
10. Wang, Ao & Chen, Hui & Liu, Lihao & Chen, Kai & Lin, Zijia & Han, Jungong & Ding, Guiguang. (2024). YOLOv10: Real-Time End-to-End Object Detection. 10.48550/arXiv.2405.14458.
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS), 30.
12. Carion, N., Massa, F., Synnaeve, G., Deshpande, A., Usunier, N., et al. (2020). End-to-End Object Detection with Transformers. European Conference on Computer Vision (ECCV).
13. Chen, X., Yin, Z., Li, Z., Zhang, Y., & Li, J. (2021). VIDT: An Efficient and Effective Fully Transformer-Based Object Detector. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(11), 3885–3899.
14. Khanam, R., & Hussain, M. (2024). YOLOv11: An Overview of the Key Architectural Enhancements. arXiv preprint arXiv:2410.17725.

hu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2021). Deformable DETR: Deformable Transformers for End-to-End Object Detection. ICLR 2021.