Budget-Aware Keyboardless Interaction

Quang-Thang Nguyen[1,2], Gia-Phuc Song-Dong[1,2], and Trung-Nghia Le[1,2(✉)]

[1] University of Science, Ho Chi Minh City, Vietnam
{22120333,22120282}@student.hcmus.edu.vn
[2] Vietnam National University, Ho Chi Minh City, Vietnam
ltnghia@fit.hcmus.edu.vn

Abstract. Interacting with computers typically relies on traditional input devices such as keyboards, mice, and monitors, which can be cumbersome for users seeking greater mobility. Virtual keyboards have been explored to address these limitations, but they often involve complex setups or expensive equipment. This paper proposes a novel virtual keyboard system that leverages only a standard camera and a paper with a printed keyboard layout. Unlike previous methods requiring complex calibration or special lighting conditions, our approach can work on standard environment using modern computer vision technologies. Combining modern segmentation and detection models with traditional image processing algorithms, we efficiently identify the keyboard region. Touch detection is performed using an algorithm analyzing the color of the user's fingernail. Experiments demonstrated a promising results our proposed solution of keyboard and keystroke detection for practical applications. Participants attended our user study also found the proposed system interesting.

Keywords: Human-computer interaction · Virtual keyboard · Keystroke recognition

1 Introduction

Traditionally, interacting with computers has depended on keyboards, mice, and monitors, which can be cumbersome and inconvenient for users requiring greater mobility. To address this challenge, various innovative methods have been proposed. For instance, virtual keyboards can now be utilized with flat surfaces [5,15], or in AR/XR environments [8,17,22]. The TapType [19] system enables users to type using two wristbands, eliminating the need for a physical keyboard. Yıldıran [22] uses virtual hands and a virtual keyboard in an XR environment, and allows users to adjust the keyboard's size.

There are two primary types of keyboardless input approaches: Dynamic Bayesian Networks [7,8,19] and projected virtual keyboards [5,15,17,22]. Dynamic Bayesian Networks may struggle with non-standard typing patterns

Q.-T. Nguyen and G.-P. Song-Dong—Contributed equally to this research.

W. Buntine et al. (Eds.): SOICT 2024, CCIS 2352, pp. 257–271, 2025.
https://doi.org/10.1007/978-981-96-4288-5_21

(a) Camera and keyboard setting up (b) Keyboard-printed paper

Fig. 1. Setup of the proposed system, which leverages only a standard camera and a keyboard image without any markers.

since they guess keystroke by analyzing finger's position when typing. The same position of finger returns different key in different keyboard layout. Paper interaction system proposed by Adajania et al. [1] requires markers printed on keyboard images that constrain the accessibility of the system. Meanwhile, projected virtual keyboards often require bulky or expensive equipment such as projectors or AR headsets. An AR headset is priced around 400 USD, while a projector costs 200 USD, and both are cumbersome to transport and use. Therefore, these systems are difficult to deploy for practical use.

In this paper, we introduce a novel approach for virtual keyboard using only a standard camera and a sheet of paper with a printed keyboard layout. In our system, a camera required to record the typing user is readily available on any laptop or smartphone, therefore it is much cheaper than existing systems relying on multiple cameras [10] or special camera [2,5,21]. Our proposed system supports a variety of keyboard layouts, including QWERTY and AZERTY, as well as keyboards from MacOS, like the Magic Keyboard, and those from Windows. Specially, unlike the virtual keyboard proposed by Adajania et al. [1], we do not need to mark anything on the printed image.

Figure 1(a) shows the setup of our proposed virtual keyboard system. A standard camera is position in a frontal, slightly angled view, pointing down towards the keyboard area. The system can be easily run on a personal computer with a webcam. By detecting the keyboard region beforehand, users can type on the printed keyboard without any hands in the frame, allowing real-time practical deployment.

The proposed system consists of two main modules: the keyboard processing module identifies the keyboard region and its keys and the touch processing module detects whether a touch is made. YOLOv8n-seg and YOLOv8n [9] are utilized for keyboard and keystroke detection. The printed keyboard is transformed to orthogonal view using homography transformation [4] to improve the detection performance. On the other hand, YOLOv8n-seg [9] is used to segment fingernails for color analysis, followed by a color analysis algorithm [13] to

identify whether fingers are pressing. The users have to apply a moderate amount of pressure while pressing key to ensure the method works. To expedite the process, we analyze fingertip movement using the Google Mediapipe library [12] and proceed with segmentation only for suspected candidates.

We conducted extensive experiments and user study to analyzing the proposed system. The experimental results show that we achieved high AP of 92% for keyboard detection and 70% for detecting keys on that keyboard. The touch detection accuracy is only around 36% due to affects of light intensity in the experiments. The results of the user study show that the majority of participants found our proposed system interesting, as they can input text with just a piece of paper and a camera.

Our contributions are as follows:

- We present a novel method for constructing a virtual keyboard using a sheet of paper that has a printed keyboard layout, all without the need for additional markings.
- We train YOLO models to segment keyboard region and detect keystrokes.
- We employ the fingertip color analysis to detect touch utilizing only a single standard camera.

2 Related Work

2.1 Virtual Keyboard

Adajania et al. [1] used a paper sheet with a printed keyboard where the endpoints were highlighted in blue, aiding in their identification during thresholding and keyboard recognition. Posner et al. [15] developed a virtual floating keyboard for mobile phones, designed to function on any surface by utilizing the phone's single 2D camera to capture and interact with the keyboard. Du et al. [5] employed a projector to display the virtual keyboard on a surface. Yildiran et al. [22] employed AR/VR Head-Mounted Displays (HMDs) to render a virtual keyboard and two virtual hands within the display, allowing users to interact with the keyboard entirely in a virtual environment. Shatilov et al. [17] introduced MyoKey, a text entry system for XR headsets that relies on inertial measurement units (IMUs) and myoelectric signals instead of virtual objects, thereby reducing the number of gestures needed and improving text input efficiency.

With the advancement of deep learning, methods that enable typing without relying on physical or virtual keyboards began to emerge in the 2020 s. Fu and Xi [7] introduced a deep-learning approach to infer keystrokes from media channels captured by AR headsets. Concurrently, Gu et al. [8] developed QwertyRing, a text entry technique that utilizes signals from a finger-worn 6-axis IMU along with a Bayesian decoder, making it compatible with external displays such as AR/VR headsets or smart TVs. Additionally, the Taptype system by Streli et al. [19] enabled typing using inertial sensors embedded in a wristband, allowing users to type without needing to remove their mobile phones from their pockets.

Different from existing systems, our proposed system employs a printed keyboard layout on a single sheet of paper, which can be in black and white or color, without requiring any extra markings. For keystroke inference, a cheap camera on any commercial laptop or mobile phone is sufficient to capture both the hands typing and the keyboard, resulting in a highly cost-efficient virtual keyboard system.

2.2 Touch Detection

Touch detection could be achieved based on depth sensors or stereo cameras. Yamamoto et al. [21] utilized a high-speed vision camera to concentrate on the high-frequency component that arose when a fingertip contacted an object. Du et al. [5] proposed a 3D optical ranging-based virtual keyboard system that uses a pattern projector and a 3D range camera. Their system reconstructs typing events by analyzing both gray-scale and depth information from the scene, specifically examining the depth curve of the finger to detect touch. With an overhead camera and a side-mounted camera, Katz et al. [10] was also able to calculate the three-dimensional coordinates of the fingertips and the surface. Agarwal et al. [2] introduced a computer vision algorithm that uses an overhead stereo camera to detect touch with high precision by aggregating stereo cues from several fingertip points. However, these methods necessitate special or multiple cameras, leading to higher costs or greater complexity during setup.

Meanwhile, shadow analysis algorithms were represented. Song et al. [18] used one camera and one projector overlooking the tabletop. When two fingertips of the real finger and its shadow created by the projector merged into one, a touch was detected. Posner et al. [15] used the similar idea to create their virtual keyboard by detecting the fingertip and its shadow's tip using environment light and a standard 2D camera. Also based on shadow analysis, Adajania et al. [1] proposed the system that detects touch by analyzing the ratio of white pixels, which represent areas not covered by shadows, to black pixels, which represent shadowed areas. Although these approaches required no additional hardware beyond the web camera, their sensitivity to lighting direction can sometimes lead to insufficient shadow changes, making touch detection less reliable.

On the other hand, Marshall et al. [13] developed a method to detect finger pressure by analyzing color changes in the fingertip, which occurred due to blood displacement when pressing against a hard surface. Their approach employed a standard camera and computer to visually capture these color variations, allowing for pressure and multi-touch sensing on diverse surfaces without altering the physical object. Inspired by the work of Marshall et al. [13], we improve their algorithm by excluding the normalization step to speed up the nail color analysis.

3 Proposed System

3.1 Overview

Our system is represented in Fig. 1(a), where the camera is positioned in front and angled upward, so that it clearly captures the keyboard and the hands typing on it. The camera angle should not be too vertical, as it will make it difficult to observe the nails. At the same time, it should not be too low, as this may impair the ability to recognize the keyboard (Sect. 4). It should be between 45° and 60°. We use a sheet of paper with a printed keyboard design instead of a physical keyboard or a virtual keyboard projected by a projector. The image can be printed in color or black and white. The size should be large enough to comfortably type by hand (i.e., A4 paper). Only one standard camera is needed, and this can be found on any laptop or smartphone. When pressing a key, users need to apply sufficient pressure for the color of the fingernail to change noticeably, allowing the system to function properly.

Our system consists of two primary modules: Keyboard processing module and Touch processing module, as illustrated in Fig. 2. The keyboard processing module identifies the bounding boxes of the keys. Meanwhile, the touch processing module detects the location where the finger touches or releases the key. This position is then used to determine which key has been pressed or released.

3.2 Keyboard Processing Module

Figure 3 illustrates overview pipeline of keyboard processing module. First, we draw a quadrilateral that closely surrounds the edges of the keyboard. Subsequently, the keyboard area is transformed to a top-down orthogonal view using homography transformation. Finally, the positions of the keys on the keyboard are identified utilizing a finely-tuned YOLO model 9. The details of the steps are presented as follows.

Keyboard Region Segmentation. We utilize a semantic segmentation model to locate the positions of keyboards within image frames. Prior to processing, each image frame is converted to gray-scale, as color can introduce noise, especially if keyboards are inconsistently colored. The processed frames are then fed into the model to segment the keyboards. The results are stored as coordinate masks for each keyboard present in the frame.

Implementation: We employ the YOLOv8s-seg.pt pre-trained model and train it for 30 epochs. For fine-tuning, we use one label (keyboard). The learning rate is set at 0.01, and the batch size is 16.

Segmentation Refinement. With a top-down and angled view, the keyboard image is not rectangular but trapezoidal. This makes it difficult for the model to recognize the keys' position on the keyboard. Therefore, we use the convex hull of these points to create a convex polygon, which maps the image back

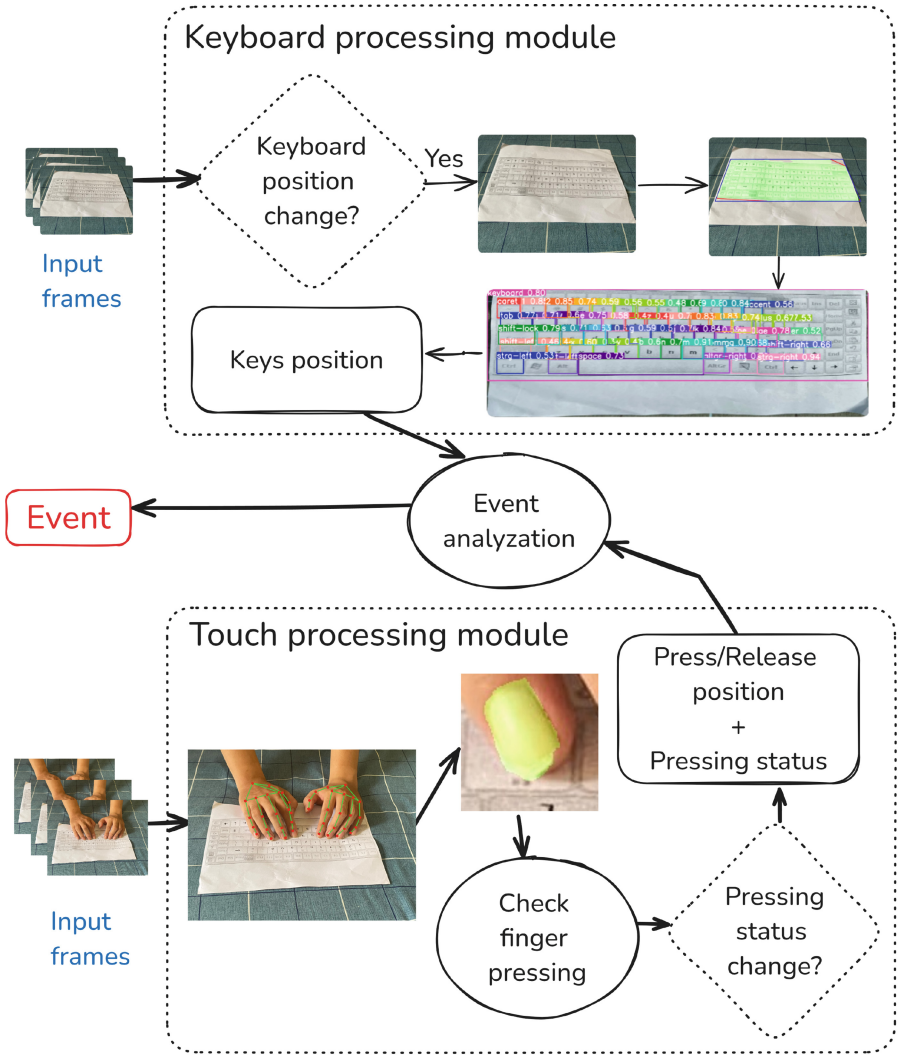


Fig. 2. Flowchart of the proposed keyboardless interaction system.

to a perpendicular projection angle. Furthermore, we construct a quadrilateral encompassing the keyboard area by utilizing the Minimum Area Enclosing Polygon algorithm [3].

Homography Transformation. Once the four vertices of the keyboard are identified, we employ Homography [4] to transform the image from an oblique perspective into a rectangular keyboard image with a top-down orthogonal view. With this perspective, recognizing individual keys on the keyboard becomes

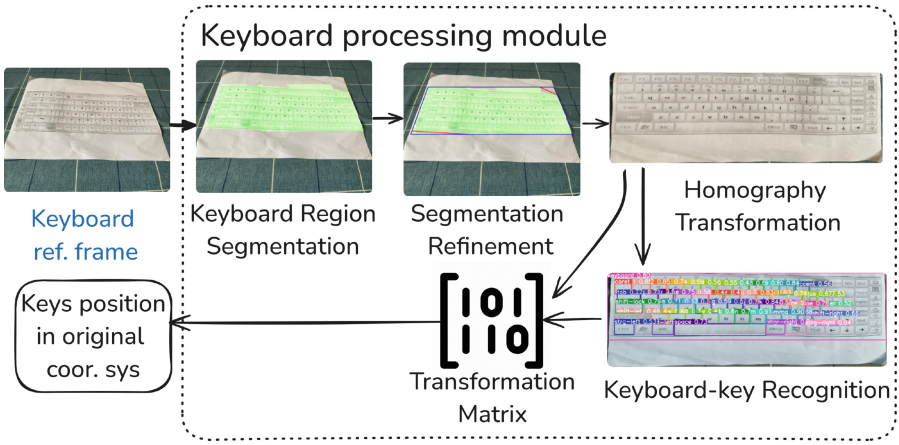


Fig. 3. Pipeline of keyboard processing module.

simpler. Particularly, characters on the keys undergo minimal distortion, facilitating the key identification process as bounding boxes do not overlap as much as in other viewing angles. Additionally, each point pressed by the user's hand corresponds to a unique bounding box. This enables to swiftly determine which key is pressed, optimizing pressed key search.

Keyboard-Key Recognition. We utilize an object detection model to pinpoint the positions of individual keys on the keyboard, resulting in a clearly delineated keyboard with each key's location distinctly marked, avoiding overlap (see Fig. 4).

Implementation: We utilize the pre-trained YOLOv8n.pt model and train it for 18 epochs. Our fine-tuning process involves 60 labels, corresponding to the 60 keys on a standard keyboard. We employ a batch size of 16, with a learning rate set at 0.01 to optimize the training process.

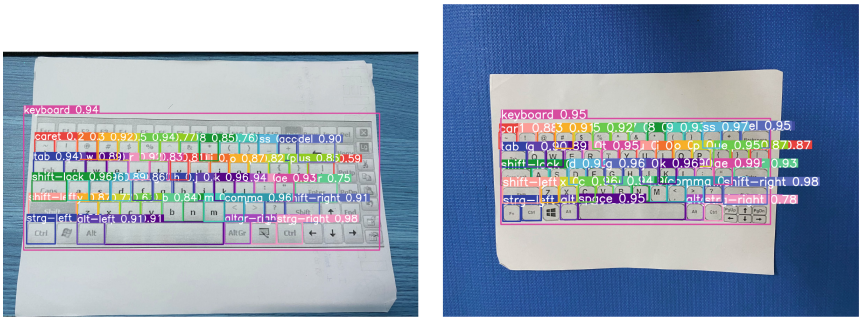


Fig. 4. Keyboard with keys are detected and marked.

3.3 Touch Processing Module

Figure 5 illustrates the pipeline for touch detection. First, the video frame is passed through the hand landmarks detection model [12] to determine the position of the fingertips . Then we compare position of the fingertips within a sufficiently small neighborhood of current frame and return one of these event: press a key or keep holding a key (i.e., touch down), release a key (i.e., touch up), and press nothing (i.e., finger is in the air).

With the touch up event, we set pressing status to False, and jump to the last step of this module. With the key down event, we analyze color of the fingertips to make sure there is a touch down event with higher confident level.

To determine whether the finger is pressing a key, we segment the fingernail and analyze its color. Particularly, we extract a small image region whose center at the position detected by the hand landmarks detection model and has a predefined size. This frame is then passed through the segmentation model to detect the actual nail area. Similar to the work of Marshall el al. [13], we calculate Hue values as follows:

$$MeanHue = \arctan\left(\left[\sum_{1}^{n}\cos(Hue)\right], \left[\sum_{1}^{n}\sin(Hue)\right]\right), \tag{1}$$

$$VarHue = \frac{1}{n}\sum_{1}^{n}\min\left((Hue - MeanHue)^2, (360 - (Hue - MeanHue))^2\right), \tag{2}$$

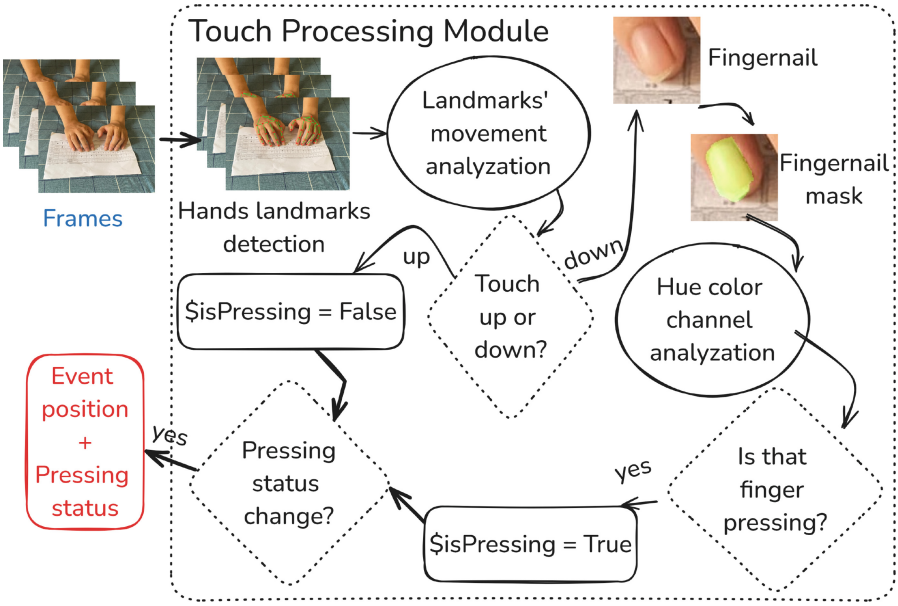


Fig. 5. Pipeline of touch processing module.

Base on the $VarHue$ value, we check whether this is indeed a press event or not. If there is a Touch down event (i.e., a finger is pressing a key), we set pressing status to True and move to last step.

Finally, the pressing status is check whether changed or not. If it did, the user had just pressed or released a key. We combine the coordinate of this event with data from keyboard processing module to determine which key is pressed or release.

To avoid excessive segmentation of the fingernail, which is a slow process, we do a calculation to find out if there is a touch down or touch up with low confident level or not. Let $|\Delta x|$ and $|\Delta y|$ represent the changes in the position of the fingertip in the horizontal and vertical directions, respectively. We checks how the new position differs from the positions of previous frames via four cases:

- If $|\Delta x|$ is too big, there might be something wrong in the hand landmarks detection model (e.g., the finger is covered by the others). We assumes the finger just release the key if it is pressing a key.
- If $\Delta y > 0$ and the finger is not pressing any key, it might be about to press a key. We note that and wait for next frame to know the finger keeps moving downward or stop when meeting a keyboard surface.
- If $\Delta y < 0$ and the finger is pressing a key, it might be about to release a key.
- If $|\Delta y| < \varepsilon$, where ε is the movement threshold, we assumes that different is caused by the unstable when detecting hand landmarks, and the finger is considered not moving. We now use the color processing layer to determine whether there is a press event or not.
- In other cases, nothing changes. We go to the next video frame.

4 Experiment

4.1 Experimental Settings

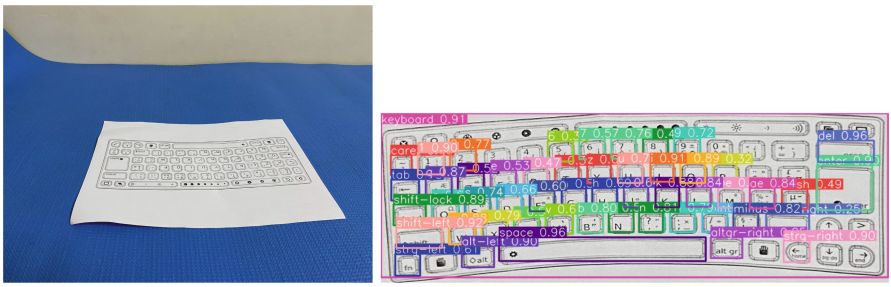
We set up our system in natural light environment for experiments because it is difficult to observe the color of the nail polish under artificial light. A back camera in iPhone 13 Pro Max was used to record typing videos.

Before typing, users should leave the keyboard stationary on a flat surface for a moment to allow the system to detect the positions of the keys. Next, they need to position their fingers in front of the camera without pressing any keys, enabling the system to capture the initial color of their fingers.

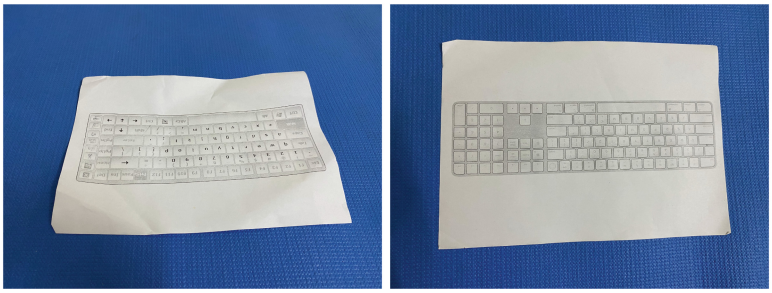
For keyboard region detection, we utilized 2,007 images for training and 388 images for validation. The dataset is aggregated from various sources, such as Microsoft COCO [11] (1,790 images for training and 388 images for validation), E-waste [20] (130 images for training), and MieKeyboard [14] (130 images for training). For keyboard-key recognition, we used "Keyboard Key Detection Dataset" [6] (5,050 images for training, 198 images for validating). For fingernail segmentation, we used the dataset from user "Personal Projects" on Roboflow [16] (4,369 images for training, 16 for validating, 12 for testing).

4.2 Experimental Results

Keyboard-Key Detection. We took 12 photos of keyboard 3 with frontal view, in three different angles, as illustrated in Fig. 6. Regardless captured images were not clear, keyboard region was detected with acceptable AP around 90%, as showed in Table 1. Average accuracy of keys on keyboard is about 70%. This detection performance is not so high because the dataset contains many keyboard layouts (e.g., QWERTY, AZERTY, Windows keyboard layout, Mac keyboard layout).



(a) Windows AZERTY keyboard with 30° view



(b) Windows QWERTY 60° view and Apple QWERTY 90° view

Fig. 6. Illustration of keyboard layouts from different perspectives.

Nail Segmentation. We captured two images of typing hands from different angles, as shown in Fig. 7. The result is represented in Table 2. The thumbs and little fingers often have a lower recognition rate because their angles are usually not aligned with the camera angle.

Touch Detection. We recorded several videos pressing a surface to validate the performance of touch detection. Results are showed in Table 3. Two index fingers and two middle fingers have high accuracy, meanwhile two ring fingers and two little fingers have high wrong probability, since they move a lot and they are not observed from a direct view.

Table 1. Keyboard detection results.

Type of keyboard	View angle	AP of "keyboard" class	Average accuracy of keys
Mac QWERTY	Frontal	0.95	0.8
Mac QWERTY	60	0.95	0.67
Mac QWERTY	45	0.94	0.7
Mac QWERTY	30	0.96	0.77
Windows AZERTY	Frontal	0.89	0.64
Windows AZERTY	60	0.89	0.64
Windows AZERTY	45	0.88	0.57
Windows AZERTY	30	0.91	0.69
Windows QWERTY	Frontal	0.91	0.81
Windows QWERTY	60	0.93	0.71
Windows QWERTY	45	0.92	0.71
Windows QWERTY	30	x	x

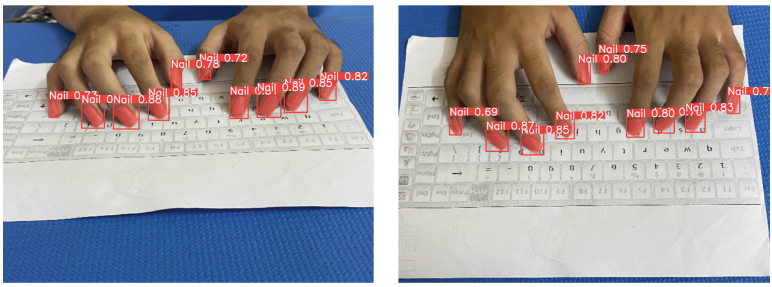


Fig. 7. Hands with fingers segmented.

Table 2. Nail segmentation result.

View angle	Mean prob.	Min. prob.	Min. finger	Max. prob.
30	0.82	0.72	Little fingers and thumbs	0.89
70	0.79	0.70	Little fingers	0.87

Table 3. Touch detection result.

Finger	No. pressing time	True Positive	False Negative
Right thumb	3	3	1
Right index finger	9	8	1
Right middle finger	7	5	2
Right ring finger	6	3	1
Right little finger	7	3	2
Left thumb	3	1	0
Left index finger	8	6	0
Left middle finger	8	5	1
Left ring finger	7	4	0
Left little finger	6	4	3

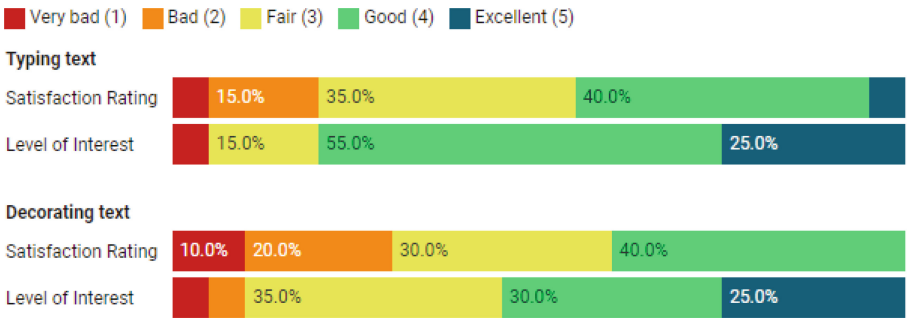


Fig. 8. User study result.

5 User Study

We invited participants to experience our system and share their feedback. The participants include 20 people (70% of whom are male) with age between 10 and 51 (most of them are from 18 to 30.)

We asked each participant to complete two tasks. The first task is to type a piece of text (using one key at a time). The second task involves formatting the text (using a combination of multiple keys such as bold, italic, copy, paste, etc.).

After that, we surveyed users about several aspects of the system, including interest level and satisfaction (i.e., accuracy and smoothness). The scale ranges from 1 (very bad) to 5 (very good). The results presented in Fig. 8 shows that participants highly evaluated our proposed system in both tasks.

Additionally, we gather user feedback on the system's limitations and what can be improved. The majority of users request improvements in smoothness and accuracy, as they sometimes experience lag in the system.

6 Conclusion

In this paper, we introduced a new approach to construct a low-cost virtual keyboard system by combining traditional computer vision algorithms and advancements in deep learning. We fine-tuned YOLOv8 models to detect and segment keyboard region and keyboard keys with high confident level, with the accuracy of 92% for keyboard region, and 70% for keys. We utilized the traditional algorithms to analyze the color of fingernail for touch detection with the accuracy about 36%. This result was not high since there are many factors affect the system (i.e. lighting conditions, camera angles, pressing force and fingernail of users). We carried out a pilot study to obtain initial qualitative insights into the usability of our system. The findings demonstrated the advantages of the proposed system, particularly its efficiency in typing without the need for specialized equipment. Participants also offered useful feedback, pointing out areas for enhancement, including the need to improve the system's smoothness and accuracy, as well as the suggestion that the application should have built-in guidance rather than relying on manual instructions.

We have plan to enhance touch detection algorithms to achieve higher performance. We will combine OCR technique to classify keyboard layout better. Therefore we can fine-tune models for each keyboard layout. Moreover, we will integrate LLM models to auto-correct text while typing.

Acknowledgement. This research is supported by research funding from Faculty of Information Technology, University of Science, Vietnam National University - Ho Chi Minh City. We would like to thank Assoc. Prof. Minh-Triet Tran for contributing to the shaping of the initial idea and some of the methods.

References

1. Adajania, Y., Gosalia, J., Kanade, A., Mehta, H., Shekokar, N.: Virtual keyboard using shadow analysis. In: 2010 3rd International Conference on Emerging Trends in Engineering and Technology, pp. 163–165 (2010). https://doi.org/10.1109/ICETET.2010.115
2. Agarwal, A., Izadi, S., Chandraker, M., Blake, A.: High precision multi-touch sensing on surfaces using overhead cameras. In: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07), pp. 197–200 (2007). https://doi.org/10.1109/TABLETOP.2007.29
3. Aggarwal, A., Yap, C.: Minimum area circumscribing polygons. Vis. Comput. **1**, 112–117 (1985). https://doi.org/10.1007/BF01898354
4. Babbar, G., Bajaj, R.: Homography theories used for image mapping: a review. In: 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 1–5 (2022). https://doi.org/10.1109/ICRITO56286.2022.9964762
5. Du, H., Oggier, T., Lustenberger, F., Charbon, E.: A virtual keyboard based on true-3D optical ranging. In: Proceedings of the British Machine Vision Conference, pp. 27.1–27.10. BMVA Press (2005). https://bmva-archive.org.uk/bmvc/2005/papers/paper-151.html

6. Farmermatzle: keyboard key detection dataset (2024). https://www.kaggle.com/datasets/farmermatzle/keyboard-key-detection, visited 06 Aug 2024
7. Fu, X., Xi, M.: Typing on any surface: Real-time keystroke detection in augmented reality. In: 2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR), pp. 350–354 (2024). https://doi.org/10.1109/AIxVR59861.2024.00060
8. Gu, Y., Yu, C., Li, Z., Li, Z., Wei, X., Shi, Y.: QwertyRing: text entry on physical surfaces using a ring. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. **4**(4) (2020). https://doi.org/10.1145/3432204
9. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics YOLOv8 (2023). https://github.com/ultralytics/ultralytics
10. Katz, I., Gabayan, K., Aghajan, H.: A multi-touch surface using multiple cameras. In: Blanc-Talon, J., Philips, W., Popescu, D., Scheunders, P. (eds.) Advanced Concepts for Intelligent Vision Systems, pp. 97–108. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74607-2_9
11. Lin, T.Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision - ECCV 2014, pp. 740–755. Springer International Publishing, Cham (2014)
12. Lugaresi, C., et al.: MediaPipe: a framework for building perception pipelines. CoRR abs/1906.08172 (2019). http://arxiv.org/abs/1906.08172
13. Marshall, J., Pridmore, T., Pound, M., Benford, S., Koleva, B.: Pressing the flesh: sensing multiple touch and finger pressure on arbitrary surfaces. In: Indulska, J., Patterson, D.J., Rodden, T., Ott, M. (eds.) Pervasive Computing, pp. 38–55. Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79576-6_3
14. Mie-University: keyboard dataset (2023). https://universe.roboflow.com/mie-university/keyboard-v2itg, visited 14 Jun 2024
15. Posner, E., Starzicki, N., Katz, E.: A single camera based floating virtual keyboard with improved touch detection. In: 2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel, pp. 1–5 (2012). https://doi.org/10.1109/EEEI.2012.6377072
16. Projects, P.: Nails segmentation dataset (2024). https://universe.roboflow.com/personal-projects-jfbag/nails_segmentation, visited 06 Aug 2024
17. Shatilov, K.A., Kwon, Y.D., Lee, L.H., Chatzopoulos, D., Hui, P.: MyoKey: inertial motion sensing and gesture-based qwerty keyboard for extended realities. IEEE Trans. Mob. Comput. **22**(8), 4807–4821 (2023). https://doi.org/10.1109/TMC.2022.3156939
18. Song, P., Winkler, S., Gilani, S.O., Zhou, Z.: Vision-based projected tabletop interface for finger interactions. In: Lew, M., Sebe, N., Huang, T.S., Bakker, E.M. (eds.) Human–Computer Interaction, pp. 49–58. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75773-3_6
19. Streli, P., Jiang, J., Fender, A.R., Meier, M., Romat, H., Holz, C.: TapType: ten-finger text entry on everyday surfaces via Bayesian inference. In: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems. CHI '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3491102.3501878
20. TRCProject: E-waste detection model dataset (2023). https://universe.roboflow.com/trcproject/e-waste-detection-model, visited 14 Jun 2024

21. Yamamoto, K., Ikeda, S., Tsuji, T., Ishii, I.: A real-time finger-tapping interface using high-speed vision system. In: 2006 IEEE International Conference on Systems, Man and Cybernetics, vol. 1, pp. 296–303 (2006). https://doi.org/10.1109/ICSMC.2006.384398
22. Yıldıran, N.F., Meteriz-Yildiran, Ü., Mohaisen, D.: AiRType: an air-tapping keyboard for augmented reality environments. In: 2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), pp. 676–677 (2022). https://doi.org/10.1109/VRW55335.2022.00189