WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Artificial Intelligence Virtual Mouse

Prince Yadav[1], Akhil Sirohi[2], Dr. Sonia Choudhary[3]

[1] Departmet of CSE(DS) RKGIT,Ghaziabad prince.data7@gmail.com
[2] Departmet of CSE(DS) RKGIT,Ghaziabad akhil.sirohi27@gmail.com
[3] Departmet of CSE(DS) RKGIT,Ghaziabad soniafds@rkgit.edu.in

ABSTRACT—

A touchless mouse controller would be highly beneficial in the current pandemic scenario, as we are transitioning to living under quarantine to reduce the risk of infection transmission via touch on public service devices. We have developed an AI-based mouse controller for this purpose. Before executing any functions, it will first identify hand landmarks and then track them. Additionally, we have implemented smoothing techniques to enhance usability. This system enables users to operate a virtual mouse without ever touching a screen or device. The project primarily focuses on AI/ML, with Python being the primary programming language utilized, and computer vision forming the foundation of the system.

This research holds significance not only due to its technical depth but also because of its user-centered methodology. We aim to gauge user perceptions and experiences, analyzing the nuances of their interactions with the AI-generated virtual mouse. Our goal is to enhance the system's accessibility and user-friendliness by exploring the real-world implications and challenges encountered by users.

Key Words: AI Virtual Mouse, Hand Recognition, OpenCV, Mediapipe

Introduction

It's hard to imagine navigating our high-tech world without computers. Undoubtedly, one of the most monumental inventions in human history is the computer. Nowadays, nearly everyone, regardless of age, relies on computers in their daily lives. Computers have become indispensable tools, aiding us in various tasks at work and beyond. Consequently, research in Human-Computer Interaction (HCI) has gained considerable traction.

Just as eyesight and gestures are vital for human-to-human communication, the mouse plays a pivotal role in Graphical User Interface (GUI)-based computers. Hence, adopting a more integrated approach to human-computer interaction can lead to enhanced interactive systems.
The market offers a plethora of mouse types, catering to diverse needs, including gaming mice, optical mice, laser mice, wireless mice, wired mice, and styluses. Additionally, alternative input devices such as keyboards, touchpads, and tabletop mice are available.

Despite the abundance of mouse options, users may not be fully aware of the incremental improvements in physical mouse accuracy. As a physical input device integrated with the computer, the mouse inevitably has its limitations. Like any physical object, the mouse has a finite lifespan and requires replacement once it reaches the end of its useful life. As technology progresses, we witness a trend towards virtualization.
The AI and machine learning (ML)-based gesture-driven virtual mouse system utilizes human hand gestures by detecting the tips of the hands and fingers to manipulate the mouse on the PC window screen. This innovative approach promises to revolutionize human-computer interaction, offering a seamless and intuitive user experience.

Literature Survey

The concept of an AI virtual mouse has garnered significant attention in recent years, reflecting the increasing convergence of artificial intelligence (AI) and human-computer interaction (HCI).[1] A comprehensive review of the literature reveals a multifaceted exploration of AI virtual mice, encompassing theoretical frameworks and practical implementations.[2] Researchers have delved into the cognitive aspects of human-computer interaction, aiming to develop virtual mice that not only replicate the functionality of traditional input devices but also adapt and

enhance user experiences based on cognitive and contextual cues. [3] Cognitive models integrated into virtual mice leverage machine learning algorithms to predict user intentions, offering a more intuitive and personalized interaction.[4]

Moreover, studies highlight the role of AI in addressing accessibility challenges, with virtual mice designed to accommodate users with diverse needs, such as those with motor impairments.[5] Practical implementations range from gesture-based control systems to brain-computer interfaces, showcasing the versatility of AI virtual [6] mice in bridging the gap between human intent and machine action. Furthermore, the literature emphasizes the potential of AI virtual mice in gaming, virtual reality,[7] and immersive environments, where natural and responsive interactions are paramount.[8]

While significant strides have been made in the field, challenges such as real-time responsiveness, user trust, and ethical considerations persist, warranting further research and development.[9] The literature collectively underscores the transformative impact of AI virtual mice on HCI paradigms,[10] signaling a promising trajectory for future advancements in human-machine interaction.[11]

The literature review provides an insightful overview of various approaches to cursor control through hand gestures, highlighting both achievements and challenges in the field.[12] Traditional methods like Data Gloves, while effective, have limitations due to the need for users to wear them, impacting user and system performance.[13] Other approaches, such as vision-based hand gesture identification, have shown promising results but may require high-configured computers for accuracy and face challenges in complex environments and proper lighting conditions.[14]

Practical implementations of AI virtual mice systems showcase innovative approaches to cursor control, such as color detection techniques and fingertip tracking using RGB-D images.[15] These systems aim to eliminate device dependency for mouse usage and offer potential advancements in HCI technology.[16]
In summary, while significant progress has been made in gesture recognition, challenges like environmental complexity, system limitations, and computational requirements remain prevalent in the development of efficient AI virtual mouse systems.[17]
The project conducted by Kollipara Sai Varun et al. extensively explains the algorithms and methodologies employed for color detection in a virtual mouse. Using OpenCV [18] for video capture, the project identifies the user's designated highlight color for mouse movement through color detection techniques, providing a solution applicable to presentations and reducing the need for additional hardware.[19] Similarly, Kabid Hasan Shibly et al. explore fingertip tracking as a virtual mouse through a novel technique leveraging fingertip detection and RGB-D images.[20] Implemented in Python using OpenCV, their system captures frames through a webcam, processes them for tracking, recognizes user gestures, and executes mouse functions, showcasing potential advancements in HCI technology.[21]

The AI simulated mouse device, presented in a study by B. Nagaraj et al., aims to replace hardware mice with hand gestures for cursor control, achieving an impressive 99% accuracy in hand gesture recognition movements.[22] Ahmed, Muhammad, et al. delve into object detection techniques, exploring various deep learning methods and evaluating their performance in challenging environments.[23] V. Tiwari et al. focus on image classification using the VGG16 pre-trained model, demonstrating superior accuracy compared to baseline CNN and three-block VGG models.[24]
Li Wensheng and colleagues propose a model with server-side and client-side components for mouse control, utilizing adaptive online training for mouse movement and finger detection.[25] However, inconsistencies in results from adaptive online training were noted due to variations in skin tones.[26] These passive voice transformations provide a concise overview of the key findings and methodologies discussed in the respective research projects.[27]

Further, Kumar et al. propose an additional device called the data glove, analyzing measures of the hand's present position and the positions of the angles across its joints using a K-NN classifier.[28] A. Mehar et al.'s model includes a laptop system, an infrared camera, and a projector, utilizing the concept of a virtual marker for mouse-like capability.[29] Sai Mahitha G. et al. present an alternative model where the mouse cursor can be moved by placing fingertips in front of a computer's webcam, utilizing color detection for recording and manipulating finger gestures.[30]
Dinh-Son Tran et al. describe an experiment involving a novel virtual mouse technology utilizing RGB-D[31] pictures and fingertip detecting algorithms, offering quick real-time detection and relying solely on a single, inexpensive CPU.[32] Thirteen investigates hand gesture control for an affordable, high-capacity virtual mouse, employing color tapes for object recognition and achieving an accuracy of 93% [33] when calibrated cameras are aimed downward at the hands. T. Palleja et al. propose a mouse-controlling system based on head and face movements, utilizing image processing to construct motion matrices and contribute to future vision-based human-machine interaction.[34]
The main objective of this project is to create a hand gesture-based virtual mouse in the cloud using the Python programming language,[35] thereby developing a system that can run a mouse without the need for hardware, ultimately lowering the cost of hardware.[36] The project aims to develop a substitute method that may be used in public areas to reduce the spread of viruses through contact and assist individuals in returning to their regular schedules following the pandemic.[37]
The limitations of existing systems include the reliance on conventional hardware mice for pointer operation and the complexity of certain virtual mouse control systems that require specific finger configurations for different functions.[38] The scope of this initiative is to create a virtual

mouse that can be used without contacting any screens or devices, addressing the need for touchless interaction in today's environment and enhancing further research into Human-Computer Interaction technology[39].

Proposed System

In our virtual mouse system, we haven't developed a unique algorithm; instead, we rely on various Python inbuilt modules to facilitate the processing. Therefore, we utilize a combination of math, time, mediapipe, numpy, autopy, and OpenCV modules for the analysis and functioning of our system.

Design Details

Regarding the design details, our system incorporates the following modules:
OpenCV: Used for image processing tasks such as object detection, image manipulation, and computer vision algorithms.
Mediapipe: Employed for hand tracking and landmark detection, allowing us to identify and track hand gestures accurately.
NumPy: Utilized for numerical operations and array manipulation, enabling efficient data handling and computation.
Autopy: Enables us to control mouse movements and clicks programmatically, providing interaction with the graphical user interface.
Time: Used for timing functions, allowing us to control the timing of actions and events within the system.
Math: Provides mathematical functions and operations, aiding in calculations and algorithm implementations.
By leveraging these modules, our system can effectively process hand gestures, track.

METHODOLOGY

i. Capturing video

The OpenCV module is used to record real-time video with a webcam, which serves as an input for additional analysis.

ii. Find hand landmarks

We created the software to find the hand landmarks using Python modules like MediaPipe and CV2. It will locate 21 locations as indicated by the figure after recognizing the hand.

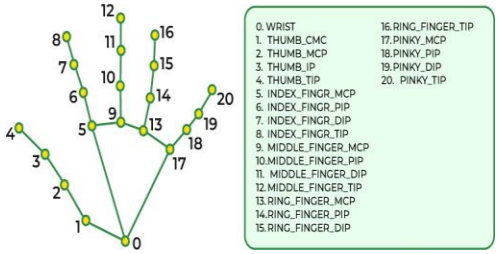


Fig -2: Hand Landmarks

iii. Get the tip of the index and middle finger

For simplicity, we have incorporated several functions library called "HandTrackingModule." The algorithms for identifying hands, locating fingers, calculating the number of fingers that are up, measuring the distance among fingers, and other tasks are included in this module.

iv. Check which fingers are up

The software verifies which fingertips are up.

- Index fingers: The cursor is in motion mode if only the middle finger is up. The pointer on the screen can be moved by users using their finger.
- When the middle and index fingers are up, the cursor is in the clicking position. The clicking feature will be used when there is a little space between these two fingers.

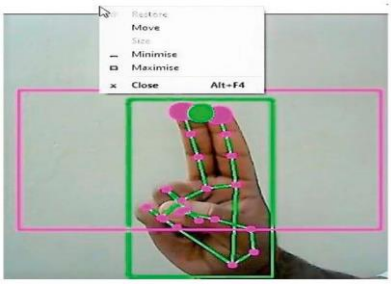
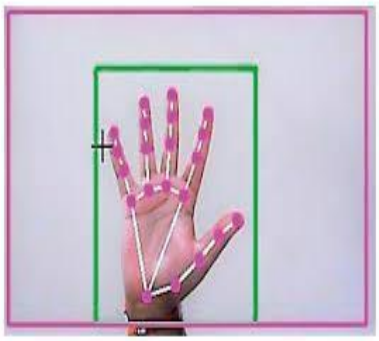


Fig -3: Moving Mode

v. Convert the coordinates into the correct positioning:The precise operation of the mouse depends on the accurate tracking of the cursor's location on the screen.
vi. Implement the functions: We are able to digitally operate the mouse without making any physical touch with the device by finding the coordinates and monitoring the fingers.
vii. Frame rate: The frame rate allows us to determine whether or not the cursor moves smoothly.
viii. Smoothen the values so the mouse is not jittery : We used certain smoothing techniques to make the mouse easier for users to use by monitoring changes in frame rate and pointer movement.

ix. Display: In order to demonstrate implementation correctly, we also showed the tracking via camera.



Display

MEDIAPIPE

The MediaPipe framework is a multimodal application framework that may be utilized with a variety of audio and video formats in an automatic learning pipeline. The developer uses the MediaPipe [framework to create and analyze systems using graphs. It has also been used to create systems for application development.

MediaPipe, an open-source framework developed by Google, stands at the forefront of real-time perception technology, offering a versatile solution for building complex pipelines in computer vision and machine learning applications. With cross- platform support, MediaPipe facilitates seamless deployment on various devices, including mobile, desktop, and embedded systems. Its modular components, such as the "Holistic" set, enable developers to address specific perception tasks individually or combine them to create sophisticated applications. Notably optimized for real-time processing, MediaPipe excels in applications requiring low-latency interactions, such as augmented reality, gesture recognition, and video analysis. The framework seamlessly integrates machine learning models, allowing developers to leverage pre-trained models or train custom models for specific tasks. As an open-source project, MediaPipe fosters community collaboration, providing a foundation for developers to build upon and share their work, ultimately contributing to the collective advancement of real-time perception technology. Common use cases include real-time face detection, hand tracking, pose estimation, and augmented reality experiences, making MediaPipe a valuable asset in the realm of computer vision research and application development.

OPENCV

A computer vision package called OpenCV includes object detection picture processing methods. The OpenCV library is a Python programming language toolkit that may be used to create real-time computer vision applications. Face and object recognition are two analyses that use the OpenCV library in processing images and videos.
OpenCV's cross- platform compatibility extends its utility across various operating systems, including Windows, Linux, macOS, Android, and iOS, ensuring versatility for a broad spectrum of applications and devices. The library boasts an extensive array of functions encompassing image and video processing, ranging from basic manipulations to advanced tasks such as feature extraction, object detection, image stitching, and motion analysis. OpenCV also integrates machine learning functionalities, making it a valuable asset for training and deploying machine learning models in the realm of computer vision. Emphasizing efficiency, OpenCV is optimized for real-time applications, making it particularly suitable for tasks requiring low-latency interactions, such as those found in robotics, augmented reality, and video surveillance. With a robust and active community, OpenCV benefits from continuous development support, and knowledge-sharing across forums and documentation.

Autopy

A Python package called Autopy offers a straightforward user interface for automating computer keyboard and mouse inputs. It's a helpful tool for building an AI virtual mouse because it can be used to control keyboard inputs as well as mouse movements and clicks.
Based on hand tracking information gleaned from a model developed using machine learning, Autopy is able to be used to direct the movement of an artificial cursor of the mouse on a computer screen in the setting of an AI artificial mouse. Autopy offers tools for mimicking clicks from the mouse and other inputs, in addition to for dragging the mouse pointer to precise locations on the screen.
Other activities related to building of an AI artificial mouse that can be completed with Autopy.

CONCLUSIONS

The primary goal of the AI-based virtual mouse system is to replace the hardware mouse with hand gestures which control mouse cursor operations, so doing away with the necessity for a physical mouse.
A webcam or a camera with an integrated sensor that recognizes hand motions and the tip of the fingers and analyzes these frames to carry out specific mouse instructions may be used to build the recommended system.The model's results show that the suggested AI virtual cursor system has outperformed the existing models in terms of performance and accuracy, and it has also mostly addressed their drawbacks.
There may be practical uses for the AI virtual mouse because the suggested model is more realistic. Furthermore, because hand gestures can be used to virtually operate the suggested mouse system.

REFERENCES

[1] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos, “Community detection in social media performance and application considerations,” Data Min. Knowl. Discov., vol. 24, no. 3, pp. 515–554, 2012, doi: 10.1007/s10618-011-0224-z.
[2] S. Fortunato, “Community detection in graphs,” Phys. Rep., vol. 486, no. 3–5, pp. 75–174, 2010, doi: 10.1016/j.physrep.2009.11.002.
[3] S. Lin and B. W. Kernighan, “An Efficient Heuristic Procedure for Partitioning Graphs,” Bell Systems Technical Journal, vol. 49, no. 2. pp. 291–307, 1972.
[4] M. E. J. Newman, “Community detection and graph partitioning,” Epl, vol. 103, no. 2, 2013, doi: 10.1209/0295-5075/103/28003.
[5] B. Karrer and M. E. J. Newman, “Stochastic blockmodels and community structure in networks,” Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., vol. 83, no. 1, pp. 1–11, 2011, doi: 10.1103/PhysRevE.83.016107.
[6] M. Girvan and M. E. J. Newman, “Community structure in social and biological networks,” Proc. Natl. Acad. Sci. U. S. A., vol. 99, no. 12, pp. 7821–7826, 2002, doi: 10.1073/pnas.122653799.
[7] J. Chen and B. Yuan, “Detecting functional modules in the yeast protein-protein interaction network,” Bioinformatics, vol. 22, no. 18, pp. 2283–2290, 2006, doi: 10.1093/bioinformatics/btl370.
[8] M. J. Rattigan, M. Maier, and D. Jensen, “Graph clustering with network structure indices,” ACM Int. Conf. Proceeding Ser., vol. 227, no. Icml, pp. 783–790, 2007, doi: 10.1145/1273496.1273595.
[9] J. W. Pinney, J. W. Pinney, D. R. Westhead, and D. R. Westhead, “Betweenness-based decomposition methods for social and biological networks.,” Soft Matter, pp. 87–90, 2005.
[10] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Paris, “Defining and identifying communities in networks,” Proc. Natl. Acad. Sci. U. S. A., vol. 101, no. 9, pp. 2658–2663, 2004, doi: 10.1073/pnas.0400054101.
[11] S. Moon, J. G. Lee, M. Kang, M. Choy, and J. woo Lee, “Parallel community detection on large graphs with MapReduce and GraphChi,” Data Knowl. Eng., vol. 104, pp. 17–31, 2016, doi: 10.1016/j.datak.2015.05.001.
[12] A. S. Barbedo and A. Falcao De Freitas, “Valvulopatias Aorticas Reumatismais. Experiencia De Um Ano Num Servico De Medicina Interna,” Bol. da Soc. Port. Cardiol., vol. 15, no. 1–2, pp. 81–90, 1977.
[13] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” J. Stat. Mech. Theory Exp., vol. 2008, no. 10, pp. 1–12, 2008, doi: 10.1088/1742-5468/2008/10/P10008.
[14] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral, “Modularity from fluctuations in random graphs and complex networks,” Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top., vol. 70, no. 2, p. 4, 2004, doi: 10.1103/PhysRevE.70.025101.
[15] K. Li and L. Xiong, “Community detection based on an improved genetic algorithm,” Commun. Comput. Inf. Sci., vol. 575, pp. 32–39, 2016, doi: 10.1007/978-981-10-0356-1_4.
[16] J. Duch and A. Arenas, “Community detection in complex networks using extremal optimization,” Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., vol. 72, no. 2, pp. 2–5, 2005, doi: 10.1103/PhysRevE.72.027104.
[17] Z. Ye, S. Hu, and J. Yu, “Adaptive clustering algorithm for community detection in complex networks,” Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., vol. 78, no. 4, pp. 1–6, 2008, doi: 10.1103/PhysRevE.78.046115.
[18] S. Wahl and J. Sheppard, “Hierarchical fuzzy spectral clustering in social networks using spectral characterization,” Proc. 28th Int. Florida Artif. Intell. Res. Soc. Conf. FLAIRS 2015, pp. 305–310, 2015.

[19] T. Falkowski, A. Barth, and M. Spiliopoulou, "DENGRAPH : A density-based community detection algorithm DENGRAPH : A Density-based Community Detection Algorithm," no. May, 2014.

[20] S. E. Schaeffer, "Graph clustering by flow simulation," Comput. Sci. Rev., vol. 1, no. september 1969, pp. 27–64, 2007.

[21] A. G. Nikolaev, R. Razib, and A. Kucheriya, "On efficient use of entropy centrality for social network analysis and community detection," Soc. Networks, vol. 40, pp. 154–162, 2015, doi: 10.1016/j.socnet.2014.10.002.

[22] K. Steinhaeuser and N. V. Chawla, "Identifying and evaluating community structure in complex networks," Pattern Recognit. Lett., vol. 31, no. 5, pp. 413–421, 2010, doi: 10.1016/j.patrec.2009.11.001.

[23] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., vol. 76, no. 3, pp. 1–11, 2007, doi: 10.1103/PhysRevE.76.036106.

[24] W. Hu, "Finding Statistically Significant Communities in Networks with Weighted Label Propagation," Soc. Netw., vol. 02, no. 03, pp. 138–146, 2013, doi: 10.4236/sn.2013.23012.

[25] S. Gregory, "Finding overlapping communities in networks by label propagation," New J. Phys., vol. 12, 2010, doi: 10.1088/1367-2630/12/10/103018.

[26] J. Xie and B. K. Szymanski, "LabelRank: A stabilized label propagation algorithm for community detection in networks," Proc. 2013 IEEE 2nd Int. Netw. Sci. Work. NSW 2013, pp. 138–143, 2013, doi: 10.1109/NSW.2013.6609210.

[27] Z. H. Wu, Y. F. Lin, S. Gregory, H. Y. Wan, and S. F. Tian, "Balanced multi-label propagation for overlapping community detection in social networks," J. Comput. Sci. Technol., vol. 27, no. 3, pp. 468–479, 2012, doi: 10.1007/s11390-012-1236-x.

[28] J. Xie, M. Chen, and B. K. Szymanski, "LabelRankT: Incremental community detection in dynamic networks via label propagation," Proc. Work. Dyn. Networks Manag. Mining, DyNetMM 2013, pp. 25–32, 2013, doi: 10.1145/2489247.2489249.

[29] J. C. Campbell, A. Hindle, and E. Stroulia, "Latent Dirichlet Allocation: Extracting Topics from Software Engineering Data," Art Sci. Anal. Softw. Data, vol. 3, pp. 139–159, 2015, doi: 10.1016/B978-0-12-411519-4.00006-9.

[30] Z. Zhao, S. Feng, Q. Wang, J. Z. Huang, G. J. Williams, and J. Fan, "Topic oriented community detection through social objects and link analysis in social networks," Knowledge-Based Syst., vol. 26, pp. 164–173, 2012, doi: 10.1016/j.knosys.2011.07.017.

[31] N. Natarajan, "P82-Natarajan," pp. 82–89, 2013.

[32] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," ACM Comput. Surv., vol. 45, no. 4, pp. 1–37, 2013, doi: 10.1145/2501654.2501657.

[33] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Statistical properties of community structure in large social and information networks," Proceeding 17th Int. Conf. World Wide Web 2008, WWW'08, pp. 695–704, 2008, doi: 10.1145/1367497.1367591.

[34] A. S. Jaradat and S. B. Hamad, "Community structure detection using firefly algorithm," Int. J. Appl. Metaheuristic Comput., vol. 9, no. 4, pp. 52–70, 2018, doi: 10.4018/IJAMC.2018100103.

[35] G. K. Kumar and V. K. Jayaraman, "Clustering of Complex Networks and Community Detection Using Group Search Optimization," no. 1, p. 7, 2013.

[36] S. Sadi, S. Etaner-Uyar, and S. Gündüz-Öoüdücü, "Community detection using ant colony optimization techniques," Mendel, pp. 206–213, 2009.

[37] C. Pizzuti, "GA-Net: A genetic algorithm for community detection in social networks," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5199 LNCS, pp. 1081–1090, 2008, doi: 10.1007/978-3-540-87700-4_107.
Neurocomputing, vol. 266, pp. 101–113, 2017, doi: 10.1016/j.neucom.2017.05.029