TOUCHLESS AND GO

A FEASIBILITY STUDY OF IMPLEMENTING GESTURE CONTROL AT THE OPERATING SYSTEM LEVEL

Presented by Triet Ngo, Align MSCS in Computer Science, Khoury College of Computer Science
CS 8674: Master's Project - Spring 2024 - Dr. Scott A. Valcourt

SUMMARY

Human-computer interaction (HCI) is an active research area that explores new and improved ways people can interact with computers and similar technologies. As technology steadily permeates human societies, with the latest trends and innovations incorporating various forms of artificial intelligence (AI) into existing software and hardware systems in order to better assist people as they interact with technology. Indeed, as technology becomes more sophisticated with advanced features, making technology easier to use therefore has positive accessibility implications. The Master's project included as part of this report attempts to explore the implementation of hand gesture control as part of the operating system (OS), and not as a separate application. Throughout the project, it is clear that the implementation of OS-level gesture control is theoretically possible, but due to the proprietary nature of commercial OSs combined with the complexity and inconsistencies of open-source OS, the project only succeeded in simulating gesture-based control as a C++ program on Microsoft Windows. Nonetheless, discoveries made within the duration of the project provide an opportunity to explore how computer vision can be incorporated as part of the OS and how it can contribute to and elevate HCI accessibility research.

DISCOVERIES AND CAVEATS

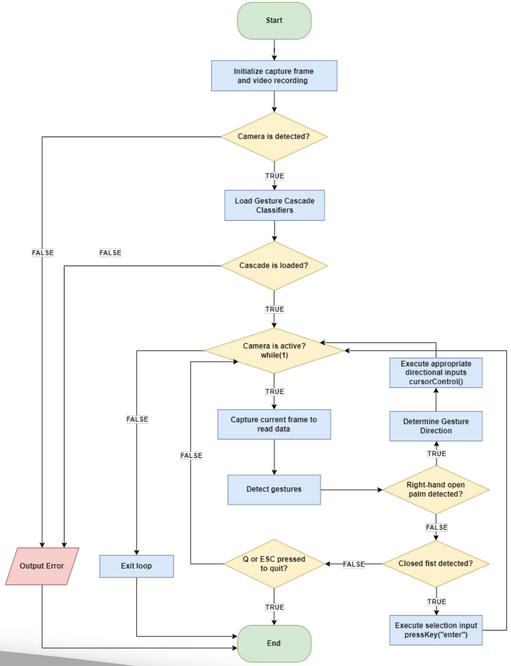
One major component of vision-based gesture control is pattern recognition, which is not trivial to implement without abstractions. Coupled with time constraints, the principal investigator (PI) relied on the latest available version of the **OpenCV** library which contains optimized computer vision functions in order to streamline the development processes related to machine perception.

OpenCV's increasing prevalence in projects, companies, and government bodies around the world effectively solidifies its status as the default computer vision library. As commercial OSs such as Microsoft Windows and Apple macOS are proprietary by nature, an OS-level program will inherently introduce low-level changes to the operating system whose source code is not available publicly and whose modification privileges are not accessible to the regular user. While Linux was considered, the PI was unable to install OpenCV on Linux due to unexpected and unexplained errors, which hindered development.

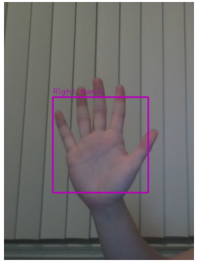
Since the library itself is currently written natively in C++, the PI determined that C++ is the most appropriate language as the documentation is the most up-to-date. This language, as well as its corresponding OpenCV library, have consequently influenced the final selection of the target platform, which is Microsoft Windows, thanks to its accessible C++ libraries needed to control inputs.

METHODOLOGY

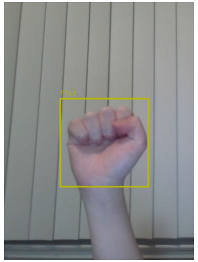
Gesture Control Algo. Flowchart


Start
Initialize capture frame and video recording
Camera is detected?
TRUE
Load Gesture Cascade Classifiers
Cascade is loaded?
TRUE
Camera is active? while(1)
TRUE
Capture current frame to read data
Detect gestures
Right-hand open palm detected?
TRUE
Determine Gesture Direction
Execute appropriate directional inputs cursorControl()
FALSE
Closed fist detected?
TRUE
Execute selection input pressKey("enter")
FALSE
Q or ESC pressed to quit?
TRUE
End
FALSE
FALSE
Exit loop
FALSE
FALSE
Output Error

Hand Gesture Detection



Right-Hand Open Palm



Closed Fist

Acknowledgement: Gesture model trained by Sandeep Sthapit.
From: github.com/Sandeep-Sthapit/HandGestureDetection

Helper Functions

cursorControl(): Takes in two 2-dimensional coordinates as four parameters at each frame. The first coordinate, that of the current on-screen cursor, represents the position of the open palm at the end of the previous frame. The second is derived from the detected open palm in the current frame. The function then calculates vertical and horizontal changes, and whichever is greater takes precedence in terms of direction. The cursor is then temporarily reset to the origin point using *cursorRefreshTemp()*, and the appropriate command is passed onto *pressKey()* to execute the associated virtual key press.

pressKey(): Takes a string as a command from the *cursorControl()* function, performs a specific virtual Winuser keyboard press an input event depending on the command. For example, if the command is the string "left", the input event will be assigned the virtual key VK_LEFT, which corresponds to a typical keyboard's left-arrow button and the input event will then be passed into the *SendInput()* function to simulate pressing and releasing the button.

cursorRefreshTemp(): Sets the cursor position using the Windows function SetCursorPos() to position (0, 0), or the top-left corner of the screen, and execute a left-mouse click event consisting of pressing and releasing actions using the mouse_event() utility function.

LIMITATIONS

1. **Not truly operating system level:** OpenCV, an external computer vision library, is required to compile and run.
2. **Gesture detection is limited to two gestures:** users cannot reliably navigate windows or apps other than the desktop.
3. **Sizable gesture datasets required:** lots of data is needed to train a good recognition model, which may be taxing to older systems.
4. **Arm strain and fatigue:** narrow field of view and detection errors require repeated gesturing.
5. **No cross-platform compatibility:** time constraints and restrictions placed on proprietary operating systems, combined with unexplained package installation peculiarities restrict the development to Microsoft Windows

SUGGESTIONS FOR FUTURE RESEARCH

1. **Explore OS cross-compatibility:** implement distributions with integrated gesture control on open-source OS such as Linux
2. **Add more gestures:** include actions such as exiting a program, rewind or move forward within a video, etc.
3. **Bring gesture control to more devices:** explore implementations beyond personal computers and integrate gestures as part of operating heavy machinery
4. **Wait for big companies to do it:** developers with privileged OS development access can implement gesture control

CONCLUSIONS

The resurgence of touchless interaction, propelled with a marked improvement in processing power and artificial intelligence, represents the latest ongoing effort in human-computer interaction research aimed at making technologies more accessible to all. And with tangible and demonstrable benefits to users, especially those with limited mobility, integrating touchless user interface into current operating systems is no doubt a worthwhile, if not trivial, endeavor.

Building upon the work of many developers in the past, the principal investigator has been able to implement a vision-based gesture control prototype on Microsoft Windows as a proof of concept for low-level implementation. Specifically, this prototype is developed using as few external libraries as possible, namely the ubiquitous computer vision library OpenCV, while leveraging built-in hardware, such as the camera, and existing Windows system-level utilities to achieve hands-off control.

This research is hoped to provide some guidance for future developers who wish to elevate human-computer interaction and make technologies more accessible and easy-to-use. Indeed, as technology continues to dig its heels further into human societies, enabling more people to use these products of progress is not only a lofty milestone, it is a moral imperative.