

Armchair Interface: A Framework for Hand Tracking for Hands-Free Human-Computer Interaction

Jeffrey A. Delmerico, Daniel R. Schlegel

Abstract—The current ubiquity of webcams offers an opportunity to create computer vision systems which can enable novel new methods for human-computer interaction. To that end, we present a system which allows the user to control the operating system cursor in a hands-free way by gesturing in mid-air. Our system leverages *OpenCV* and the *X* windowing system to track the index finger and thumb of a user using a webcam; the user can motion in the direction she wishes the cursor to move, and can execute mouse operations by “pointing” toward the camera and breaking a “plane of interaction”. We perform tracking using the Continuously Adaptive Mean Shift algorithm [Bra98], which uses a modified version of Mean Shift to find and track the modes of a color-based probability density. Since we are relying on color for tracking, we require the user to wear gloves with colored fingertips in order to distinguish the desired targets. The “gesture controller” portion of our system is designed in such a way as to be easily expandable to recognize many more gestures besides mouse interactions. Our goal with this work is to demonstrate a proof-of-concept system for enabling a new method of controlling and interfacing with a computer using commodity webcams.

I. INTRODUCTION

Visions of hands-free computer interfaces have permeated science fiction (i.e. *Minority Report*) for many years, but we are now on the cusp of being able to implement systems which capture at least the same spirit as the fictional ones using off-the-shelf webcam technology and open source libraries. Although much work has been done in both hand tracking and gesture recognition, to our knowledge, no one has yet married the two in a way that allows the user to override the mouse interface with a hands-free one. Our system represents a first step in that direction, as a proof-of-concept demonstration that the user can interface with an operating system using hand tracking and gesture recognition via the webcam. Much work remains to be done to make it user-friendly enough that a person might opt to use it *instead of* the mouse, but we’ve demonstrated that the principle is sound, and with some refinement, could expand the realm of computer interfaces beyond mice, keyboards, and trackpads.

II. RELATED WORKS

Although we know of no other work with the same goals and using the same methods, our project has built on a large body of research in both tracking and gesture recognition.

A. Tracking

A number of different methods have been introduced in recent years for color-based hand tracking. One such system uses

color to segment the arms of the user, and then incorporates a physical model to identify and track a finger [AWdVL00]. Although their results seem robust, it is unclear whether they can be implemented in real-time, which is a prerequisite for an effective user interface. A number of other methods involve the application of mean shift to tracking, or use color-based particle filters. Mean shift has been shown to be effective in finding the modes of a multimodal density, in this case the color space of an image [CM02]. By extending that concept to sequences of images, an iterative mean shift algorithm has been implemented for real-time color-based tracking in video [DCM00]. The same group augmented these methods to improve target localization by masking the target with an isotropic kernel before performing mean shift on the smoothed feature space [DCM03]. An algorithm called Continuously Adaptive Mean Shift (CAMSHIFT) further extends this concept to handle the dynamic color space of a video sequence [Bra98]. Particle filters have also been shown to be effective at color-based tracking in video, and are a robust method for dealing with occlusion, complex backgrounds, and variable numbers of tracked objects [JCM05]. However, they make no claim to accomplishing this in real-time. Another approach incorporates both of these methods by embedding mean shift tracking within a particle filter to combine the “best of both worlds”, apparently in real-time [CSO04].

B. System

Over the years there has been a great deal of interest in vision gestures for specific computing tasks. These have ranged from television control to computer games [WTFW99] and more recently to augmented reality [JHYS06]. Some attempts have been made at a simplistic control model using marking menus. This performs a certain action when a hand is detected in a certain region of the cameras field of view [SLE02]. This of course does not allow generalized control, but allows an interface for some tasks which are easily distilled down to a small set of options.

Some work has been done in gestures for general computing but there has been no clearly successful method. Work has been done in using eye tracking [Kum07] and even nose-tracking to move a mouse on the screen [LZY07]. Nose tracking tends to use gestures that may be unnatural to click the mouse (such as “dwell time” [JGF00] which could be a problem if the user is performing some action which doesn’t involve head movement). These two interfaces are more useful

for persons with disabilities who lack hand control. In addition there has been work in tracking a hands movement on a surface or within a 2D plane which would simulate a mouse [BMP]. Kleek, Robertson, and Laddaga produced a system in 2004 which uses gestures for general computer use. Their system uses a "thumbs-up" gesture to enter the state where the mouse is being moved, and a fist gesture is used to click the mouse. It is unclear how the system deals with the transition between the two states though (ie. does the mouse still move while the hand is moving towards being a fist) [PRK04].

III. METHODS

After much experimentation, we settled on a color-based mean shift approach to finger tracking, as well as a more natural "pointing" gesture for mouse actions. Our methods allowed us to leverage open source libraries in order to speed development, enhance robustness, and at least leave open the possibility for cross-platform deployment. Additionally, the use of gloves with colored fingertips allowed us to more effectively perform tracking than if we had relied on a model-based approach such as [AWdVL00] with skin-tone detection.

A. Tracking

Some initial tests with particle filters indicated that they would be effective for our purposes, but they were far too computationally expensive if further processing would be necessary for gesture recognition. A more lightweight solution was to use OpenCV's native CAMSHIFT method, which is based on [Bra98]. It proved to be fairly robust, and left a lot of computational room for further components of our system. One aspect of this method which required significant investigation (which is ongoing) is the problem of reinitialization. Moving too quickly, moving out of the frame, occlusion, or a similarly colored background all proved to be challenges for for the CAMSHIFT algorithm. When one of these things occurred, the algorithm would lose track of the finger it was tracking, and would either begin tracking a different mode from the background, or would not be able to find a stationary point in the distribution and would get "stuck" at the last point in the frame for which it had tracked the object. In either of these cases, the system would need to find the tracked finger again. Although it was often possible to move back to the previous location to lock the tracker back onto the desired finger, we experimented with a number of approaches for reinitialize tracking so that it could be done automatically. The CAMSHIFT algorithm works by using the previously converged upon mode as the initial guess for the finger location in the next frame. We made several attempts to predict the location of the next observation for that finger from previously acquired data, in order to reinitialize a lost tracker for the next frame. In an initial test, we provided the algorithm with the entire image as the initial region of interest, so that it would reinitialize by finding the dominant mode in the entire image. We also used the last known position and velocity to estimate the new location in a linear way. In addition, we attempted to use the other finger on the same hand to constrain the

possible positions of the lost finger. None of these methods were entirely satisfactory in their performance. Although we have yet to find a method which is robust enough to make our system user friendly, we settled on reinitialization using the last verified *observation*, which is described below. We only recorded these observations when several conditions were met, in order to ensure that valid data was being used in our gesture processing, so by reverting to the last known "good" observation, we could at least guarantee valid data for the reinitialization. The results are improved, but still not robust enough.

We also experimented a bit with the glove design. Initially, we tried using ink to color the finger tips, but due to irregularities in how the ink was absorbed, the histograms which resulted were not always consistent. Despite apprehension that using colored electrical tape to distinguish the fingertips would provide too much specularly, this approach resulted in very consistent histograms within the HSV color space, and as a result, provided consistency for tracking.

B. Gesture Recognition

The Armchair Interface system was designed from the start to be an extremely modular system. At its core are two subsystems: the tracker and the gesture controller. The tracker submits what are called *observations* to the gesture controller. These observations contain the location of each of the fingers being tracked along with the radius of the observation area for each. Calculations are then performed on the observation to determine relative distance between fingers on the same hand, velocity of movement, and coordinates for the fingers are converted from absolute (screen) coordinates to relative coordinates. Observations are then submitted to all registered gesture recognizers.

Gesture recognizers are included in the system for detecting mouse clicks. The first of these is a "pinching" gesture of the forefinger and thumb. When the fingers touch a mouse-down event is registered, when they separate again, a mouse-up event is registered.

This first attempt at a gesture to click did not appear to be natural. Since the hands are being tracked in real time as the pinch gesture occurs the forefinger moves down to meet the thumb, thereby moving the mouse down as well. This did not appear to be something people easily adapted to having to deal with. It is unclear from their paper whether the similarly gestured system by Kleek, Robertson, and Laddaga dealt with this kind of issue as well.

In an attempt to rectify these usability concerns the idea of using a *plane of interaction* is introduced. All of the systems previously mentioned (including our own initial implementation) assume a two dimensional mapping. Having a plane of interaction introduces a third dimension. In this paradigm moving the mouse and clicking can be done with only a single interaction finger. The user moves the mouse around by moving their pointer finger through the air at some distance away from the screen, and when the finger moves close enough to the screen (into the interaction plane) a mouse down is

registered. Similarly a mouse up event is registered when the finger is moved out of the plane. This is analogous to tapping on a touch screen.

One issue we experienced with this is that of course this interaction is really the closer you get to the *camera* the closer you are to the plane of interaction, not the *screen* per se. If the camera is at any angle relative to the screen the perceived distance in the *z* axis from the camera will be different from those the user would perceive as different in the *z* axis from the screen. This type of issue can be accounted for through the use of an affine transform in the calibration to distort the radius of the observations at a given point. (A further option is technology such as that developed at MIT to place a camera directly behind a screen [Lab09]).

A calibration system is designed to create a mapping between the camera and the screen. The user can choose to calibrate the camera so that when they gesture at the screen they are gesturing at a place logical to them in relation to what is happening on the screen. The currently implemented system only scales the appropriate pixels from the camera to those of the screen. It also allows the user to note the location of the plane of interaction for a more natural experience. As noted above, expanding this to use an affine transform instead of simple scaling may have a great impact on usability.

It is very easy to add new gestures to the system. One need only create a function which accepts the new observations as they arrive and register it with the gesture controller. The new gesture will be queried each time a new observation is submitted. Gestures can be conceived of which use the relationships between finger locations stored in the observations or even ones using machine learning techniques.

IV. RESULTS

As a proof of concept, our results indicate that our approach has merits, and with further experimentation and refinement, could provide a usable computer interface.

Although still not robust enough to be user-friendly, the tracking portion of our project, in principle, offers a lightweight method for finger tracking, when used with the colored-fingertip gloves. A more general approach is certainly desirable, but in order to leave enough headroom for the other components of the system, having to wear the gloves is a small inconvenience. The reinitialization problem still needs much more work in order for this tracking system to be reliable enough for operating system control. However, it is encouraging that some of the experimentation in this area resulted in improvement.

In addition to the issues mentioned previously there are other inherent issues with the use of off the shelf webcams. There is a noticeable decrease in resolution of mapping a normal webcam to the display screen. This is because most webcams today are still 0.3MP cameras operating at a resolution of 640x480. The average computer monitor today operates at resolutions of 1024x768 or higher, meaning that there are twice or more pixels visible on the screen as there are available on the webcam. Additionally monitors frequently use a 16:10

display ratio vs. the 4:3 common in cameras. This causes this issue to be amplified in the horizontal axis. The second issue can be corrected via the use of a calibration system as discussed earlier, but this only serves to cause equally bad precision on both axes, and doesn't improve the resolution matching. For this a higher resolution webcam is necessary - even 1MP would create a nearly 1-1 mapping in most cases. This issue (among others) is also mentioned in [KH01].

V. CONCLUSIONS & FUTURE WORK

In the tracking domain, further work is necessary to improve reinitialization. Another proposed improvement would be to provide priors for the histograms of the colors that the system is to track, and then learn the histograms of the actual fingertips one tracking commences. This would avoid the manual initialization that is present in the system. Certainly, some of the other methods discussed in the related work section could be implemented to expand the effectiveness of our tracking method, hopefully without having a negative impact on real-time performance. Mean shift has also apparently been implemented on GPUs with CUDA, so an interesting investigation for the future might be to incorporate that to reduce the tracking overhead.

We have started to implement a Hidden Markov Model for gesture learning and recognition in hopes to be able to create better and more natural gestures. The gestures currently implemented all use a heuristic approach. HMMs have been used extensively for gesture recognition in pen computing [DT04] and in vision [ER98] before. Using a type of machine learning instead of heuristics for a gesture recognizer is no more difficult to have interact with our system.

REFERENCES

- [AWdVL00] Mubarak Shah Andrew Wu and N. da Vitoria Lobo. A virtual 3d blackboard: 3d finger tracking using a single camera. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 536–543, 2000.
- [BMP] Dan Crosta Ben Mitchell and Zach Pezzementi. Camera based mouse. <http://www.cs.jhu.edu/ben/vision/lab5/index.html>.
- [Bra98] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. Technical report, Intel Technology Journal, Q2 1998.
- [CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619, 2002.
- [CSO04] Tieniu Tan Caifeng Shan, Yucheng Wei and Frederic Ojardias. Real time hand tracking by combining particle filtering and mean shift. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 669–674, 2004.
- [DCM00] Visvanathan Ramesh Dorin Comaniciu and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 142–149, 2000.
- [DCM03] Visvanathan Ramesh Dorin Comaniciu and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 564–577, 2003.
- [DT04] Richard Mann David Tausky. Categorization and learning of pen motion using hidden markov models. In *Proceedings of the 1st Canadian Conference on Computer and Robot Vision (CRV'04)*, pages 488–495, 2004.

- [ER98] S. Eickeler and G. Rigoll. Continuous online gesture recognition based on hidden markov models. Technical report, Gerhard-Mercator-University Duisburg, 1998.
- [JCM05] Branko Ristic Jacek Czyz and Benoit Macq. A color-based particle filter for joint detection and tracking of multiple objects. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, pages 217–220, 2005.
- [JGF00] Margrit Betke James Gips and Peter Fleming. The camera mouse: Preliminary investigation of automated visual tracking for computer access. In *Proceedings of RESNA 2000*, pages 98–100, 2000.
- [JHYS06] Jong-Seung Park Jong-Hyun Yoon and Mee Young Sung. Vision-based bare-hand gesture interface for interactive augmented reality applications. In *Proceedings of ICEC 2006*, 2006.
- [KH01] Rick Kjeldsen and Jacob Hartman. Design issues for vision-based computer interaction systems. In *Proceedings of PUI 2001*, 2001.
- [Kum07] Manu Kumar. *Gaze-Enhanced User Interface Design*. PhD thesis, Stanford University, May 2007.
- [Lab09] MIT Media Lab. Computing with the wave of the hand, December 2009.
- [LZY07] Weixian Li Lumin Zhang, Fuqiang Zhou and Xiaoke Yang. Human-computer interaction system based on nose tracking. *HCI*, pages 769–778, 2007.
- [PRK04] Robert Laddaga Paul Robertson and Max Van Kleek. Virtual mouse vision based interface. In *Proceedings of IUI04*, pages 177–183, January 2004.
- [SLE02] Lars Bretzner Sren Lenman and Bjrn Eiderbck. Computer vision based recognition of hand gestures for human-computer interaction. Technical report, University of Stockholm, January 2002.
- [WTFW99] H. Kage K. Tanaka K. Kyuma W. T. Freeman, P. A. Beardsley and C. D. Weissman. Computer vision for computer interaction. *SIGGRAPH Compute Graphics*, November 1999.