Application Control Through Accurate Finger Tracking

Daniel Bentall
Department of Mechatronics Engineering
University of Canterbury
Christchurch, NZ 8042
Email: djb216@uclive.ac.nz

Richard Green
Department of Computer Science
University of Canterbury
Christchurch, NZ 8042
Email: richard.green@canterbury.ac.nz

***Abstract*—An application of vision-based computer interfaces is explored, where the Printed User Interface (PUI) controls a computer program. The usability is evaluated based on frame processing rate, computer resource consumption and reliability while interfacing with the image editing program Photoshop. The user's hands are extracted from over a sheet of paper, and each visible fingertip is tracked. Polygons representing interaction zones are defined for the paper, which would typically represented by an image of a button or similar. Commands are specified for each polygon; when a finger is over a polygon the corresponding command is sent to the target application. The goal is to use the output of this project to create a GUI authoring tool which a user can use to create their own PUI customised to suit their experiences with their chosen program.**

I. Introduction

Computer hardware and software are developing rapidly, while common input methods such as the mouse and keyboard set-up remain largely unchanged. For many advanced computer programs the constraints on input are a massive bottleneck on productivity. In some applications the lack of suitable input devices severely limits or even prohibits certain usage altogether. Examples of this are digital audio workstation programs such as FL Studio, which have the option of live control of the music, typically with many parameters to be controlled at once. However, without expensive hardware such as MIDI controllers, controlling the music live is extremely limited. Additionally, an ideal hardware set-up will vary greatly from song to song, with vastly different mixtures and layouts of binary, discrete and continuous controls. To achieve the necessary flexibility would require a range of costly hardware devices and potentially as many interfacing difficulties.

The application developed in this paper was tested with the image manipulation program Adobe Photoshop CS3. The application communicated with Photoshop by creating keyboard events. This simple control method minimised development while providing a widely useful control interface. Any program which uses hot-keys can be controlled using this method, however the control is limited to actions that hot-keys can perform. An additional advantage is the simplicity of setting the actions of the interface; a user only needs to know the required key-presses. For more powerful control methods a program-specific interface would be required.

The majority of computer vision related papers focus on new techniques and algorithms for achieving more robust tracking and better performance. Few investigate use with standard computer programs and the potential for enhancing productivity and usability. Some novel interaction devices have been developed, including various Tangible User Interfaces. Examples of this are standalone tables including the Microsoft Surface and DiamondTouch, the haptic input device PHANTOM Omni, and vision-based devices such as the Microsoft Kinect. In general the devices do not provide the flexible interface and low cost which this project aims to achieve.

This paper investigates the possibilities for a simple vision-based configurable interface program. The only hardware requirements for the interface would be a laptop or desktop computer and a standard USB webcam.

Many computer vision applications seek only to run at or near real-time on a modern inexpensive computer system. To be useful as an input method to other computer programs the application must not consume all of the system's resources. Thus this research project sought to create a specialised hand tracker: one which operates consuming minimal resources, accurate and fast enough to be useful as an input device, and within the constrained but realistic setting of a colourless sheet of paper. Research indicates that when an action is followed within 50ms by its reaction, a user perceives them as occurring immediately [1]. Thus if a finger action is detected in one frame, then the output should have been produced within 50ms. A frame rate of 20Hz (50ms) should be sufficient for smooth interactions. The system should be limited to this frequency to minimize resource consumption. This leads to the following project evaluation metrics:

- Computational requirements
- Speed
- Accuracy
- Multi-finger tracking
- Cost
- Range of applications

II. Related Works

Relevant topics within the field of computer vision:

- Object recognition and tracking
- Detecting finger "presses"
- Gesture recognition
- Interfacing with target applications

A. Computer Vision Research

Research in the area of vision-based computer interfaces typically involves new or improved tracking methods. A promising algorithm is known as Predator or Tracking-Learning-Detection (TLD). This combines binary classification with online learning, so that the tracker tends to improve over time [2], [3], [4]. When compared with state-of-the-art trackers, Predator was found to perform significantly better in a range of situations. Predator was tested for use with this project, but was found to perform poorly tracking hands and fingers, and does not have the means to track multiple fingers.

One study which matched the set-up of this project relatively closely was the "Finger Mouse" project. In this project the index finger was tracked over a blank tabletop, using the output to control the mouse [5]. Clicks were supported by recognising the thumb extension gesture. The project used the CAMSHIFT algorithm, so could operate well over any backgrounds that didn't contain too much skin-like colour. The limitations of this project were one-hand and one-finger interaction only, and limited computer interaction. The position of the finger was measured relative to the camera co-ordinates, which did not allow accurate mapping to any real-world object as was required by this project.

Another study investigating partitioned sampling was tested with a drawing package [6], as was used for this project. Partitioned sampling avoids the large computational cost of particle filters while retaining their benefits [7]. A black sheet of paper was used as a mobile workspace. The paper was tracked using a Kalman filter, and allowed natural manipulations similar to traditional drawing on paper. The study also investigates the use of partitioned sampling for tracking articulated objects. The tracker was able to determine the pose of each thumb joint with reasonable accuracy, which was again used to determine when the finger was active. The method for tracking the paper, while probably less robust than the OPIRA tracking library used in this project, is likely to be less computationally expensive. In the constrained setting of this project this method could be reliable enough.

Farinella and Rustico [8] used two webcams in an uncommon configuration; positioned laterally to the camera's view, at table height. One camera would view the x-coordinate of the finger while the other would view the y-coordinate. The cameras monitored three virtual lines parallel to the projection surface, to detect when fingers or other pointing objects were near the surface or touching the surface. This allowed tablet pen style input, where holding the pen near the surface controls the cursor, and pressing the pen to the surface clicks the mouse button. This system would not naturally work with multiple fingers, as fingers would have to be correlated between the two cameras.

Two similar studies involved the combination of projector and camera mounted side by side and pointed at the working surface. The first utilised cross-correlation for template matching, and addressed a number of issues regarding template matching in their scenario [9]. The second study [10] used an algorithm by Hardenberg [11] which was developed for tracking fingers and detecting hand postures. The distortion of the projection due to any viewing angle was compensated for, allowing the projector to be mounted out of the way of the user.

Hardenberg's algorithm mentioned above uses a 'smart' difference imaging approach to extract the fingers. The current frame is compared with a reference frame which is a running average of the background elements of the scene. This means that changes in lighting do not affect the output. Another interesting technique is the click interaction. When a user's finger stays in roughly the same position for 0.5s a click command is generated. Demonstration videos show good performance on a 1GHz Pentium 4 computer, indicating that the algorithm should run easily on modern computers.

B. Alternative Hardware Options

The popularity of computer gaming has led to the most commercial development of new computer input devices, including specially designed gaming keypads and joysticks. These often incorporate programmable keys and application specific settings, giving them qualities of an adaptable interface device. Hardware like this is always limited to its physical layout and often costs significantly more than standard input devices.

Tablets and tablet PCs are becoming increasingly popular as an alternative to mice and keyboards for non-text input situations. In the graphic design and digital arts communities graphics tablets such as those created by Wacom are widely used. In addition to the pressure-sensitive pen input area, there are usually a few programmable auxiliary controls. This hardware has been shown to be faster for users for the low-level action of dragging items with the cursor [12], and provides a more intuitive interface. The programmable buttons allow the user to quickly access common commands. However these systems are limited by a small number of controls and a fixed physical set-up.

Mobile computing platforms integrating touchscreens represent the most successful advances in user-interface technology, such as many smartphones and the iPad. These devices allow a range of interactions that were not previously possible using hand touches and gestures on the screen, often combined with other sensors such as accelerometers to provide more possibilities. However these devices do not currently provide the flexibility, low cost and wide applicability that is possible with this project.

III. System Description

A. Set-up

The user positions a webcam facing a flat surface within reach of their computer. A guide page comprising of an image header and a set of interaction zones - buttons, sliders and pads - is placed within the field of view of the camera. When the system is running the user places their fingers over the interaction zones and commands are sent to the target computer program.

The initialising stages of the program start with reading the image files defining the marker image, rescaling the image to make it smaller if it is larger than a threshold size and reading a vector image file defining the interaction zones.

B. Computer Vision Processing

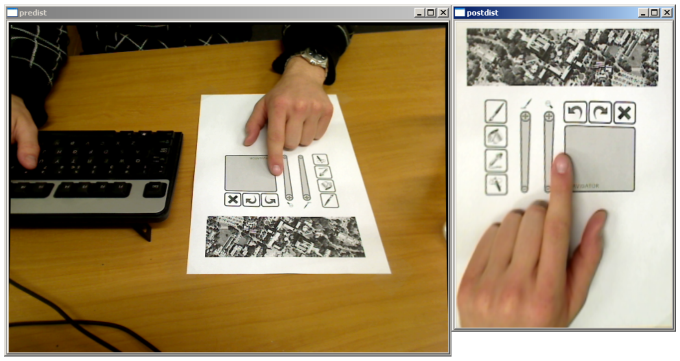

Fig. 1. An example of the perspective transformation resulting in a plan view of the page. At the low angle of roughly 45° the distortion to the hand is evident.

The first part of the process is tracking of the page using the OPIRA registration algorithm [13]. Using the homography information from OPIRA, the video stream is then transformed to remove the perspective of the camera's view of the page (Fig. 1). This gives a plan view of the page and the user's hand with nothing visible beyond the edges of the page. Initially the OPIRA registration was run for every frame that was processed, however this caused a drastic reduction in performance. As a result the registration step was removed from the main loop, so that it only occurs during the initial stages of the program and thereafter only when the user presses the 'spacebar'. Each new frame is then transformed using the results of the initial registration. As a result the camera and page should not be too easy to move, so that usual user motions don't change the perspective. Deliberate repositions are fine as the user can update the registration as needed.

The plan view of the page is then processed using simple computer vision techniques with the OpenCV (Computer Vision) library. A saturation image is constructed with the image header masked out, then smoothed, thresholded, dilated and eroded. Due to the constrained nature of the situation this simple method reliably and quickly extracts the hand, as bare hands are universally far more saturated than colourless paper. Next the centroid of the hand is calculated, and the corners of the hand detected using OpenCV's corner minimum eigenvalue function. This operation highlights the fingertips and webbing between the fingers, as the outline of the hand changes direction rapidly, like a corner. These areas are then extracted through thresholding and morphology operations, and their individual centroids calculated. This gives an array of fingertip and webbing points. The average distance from the centroid of the hand is calculated, and those points further than the average are considered fingers, while those closer are considered webbing points (Fig. 2).

The speed and CPU consumption of the process are controlled by concluding the main loop with a wait-for-key instruction. The program idles for a set time or until a key is pressed. The set time can be used to increase the execution time of a loop, while giving the CPU more time for other tasks.

C. Application Control

In the current prototypical version of the software only single-finger interactions are used. The finger with the greatest distance is considered the primary pointer, and each interaction zone is checked to see whether this finger is within its bounds. If the finger is, and another 'clicking' finger is also extended, then the button is considered 'pressed', and the zone's key press is generated. Depending on the zone type, either a single key press or a repeated key press is generated until the pointer leaves the button or the 'clicker' finger is retracted.

The different types of interaction zone planned are:

- Buttons, which execute a single command when pressed.
- Buttons, which repeat a single command when pressed.
- Discrete sliders, which have commands mapped to different parts of their length.
- Continuous sliders, which control some continuous parameter, and can be either absolute or relative.
- 2D pads, which control two parameters based on finger x-y position, and two more parameters using multi-touch detection.

Currently only buttons are supported with a single pointing finger. Another planned feature is providing an additional degree of freedom utilising the thumb extension. An extra parameter could be specified which relates to the angle the thumb is extended to. In an image editing program a 2D pad might be used like a tablet pen input, while the thumb extension would control the brush size or opacity.

IV. Results

A. Test Platforms

Testing of the program was performed on the desktop computer on which most of the development was performed. 32-bit Microsoft Windows XP Service Pack 3 was operating on an Intel Core 2 Quad CPU running at 2.4GHz with 4GB of RAM (3.25GB effective), and an NVIDIA GeForce 8800 GTS graphics card. Early testing was performed on a laptop with Windows 7 64-bit, running on an Intel Core i7 ULV (Ultra Low Voltage) CPU with 2GB of RAM and no video adaptor.

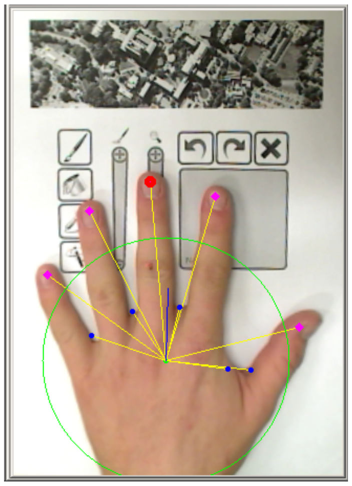

Fig. 3. A demonstration of the multiple finger tracking potential of the system.

B. Test Results

Early versions of the program did not contain the OPIRA tracking library, and the camera was simply positioned above a blank white sheet of paper, and zoomed in until the page filled the camera view. The hand-tracking algorithm developed was fast and reliable at detecting the fingertips of the user, as long as the parameters of the algorithm were correctly set. At a capped rate of 20 frames per second the program only consumed around 10% of the CPU and 55MB of the memory on both testing computers. The dimensions of the processed area were 662 by 542 pixels.

When the OPIRA tracking library was integrated into the project the performance dropped greatly. The CPU consumption climbed to around 80%, while the frame rate dropped down to around 10 fps. This drop in performance negated two of the projects key objectives, which were to achieve low CPU consumption and high speeds. As a result, the OPIRA tracking was made to only occur at the beginning of the program and when the user requested an update. After this change the performance returned to previous levels.

TABLE I
FRAMES PER SECOND (FPS) AND CPU CONSUMPTION AT DIFFERENT WAIT PERIODS

Wait (ms)	Frame Rate (Hz)	CPU Usage (%)
100	7	5
90	8	7
80	8	7
70	9	7
60	10	9
50	10	9
40	12	12
30	12	12
20	12	14
10	12	13
1	12	13

All visible fingers were tracked (Fig. 3), although the demonstration code only utilised one finger at a time.

The effects of changing the wait period on the processing rate and the CPU consumption were tested (Table I). As expected, there was a negative correlation between the loop wait time and both other parameters, however the results at low wait times were unexpected. Instead of frame-rate and CPU usage continuing to rise, both metrics became roughly stationary at a wait period of around 40ms. The reason for this phenomenon was not clear.

V. DISCUSSION

In order to make this program a useful tool for creating customised interfaces, a Graphical User Interface (GUI) must be created. This GUI would allow a user to choose a marker image, lay out the PUI with the marker image and the interaction zones, and set the commands and parameters for the zones. This would make the system accessible to many more users, while also making the creation of new interfaces easier.

An interface more integrated with the target applications would give the system more flexibility and usefulness, but

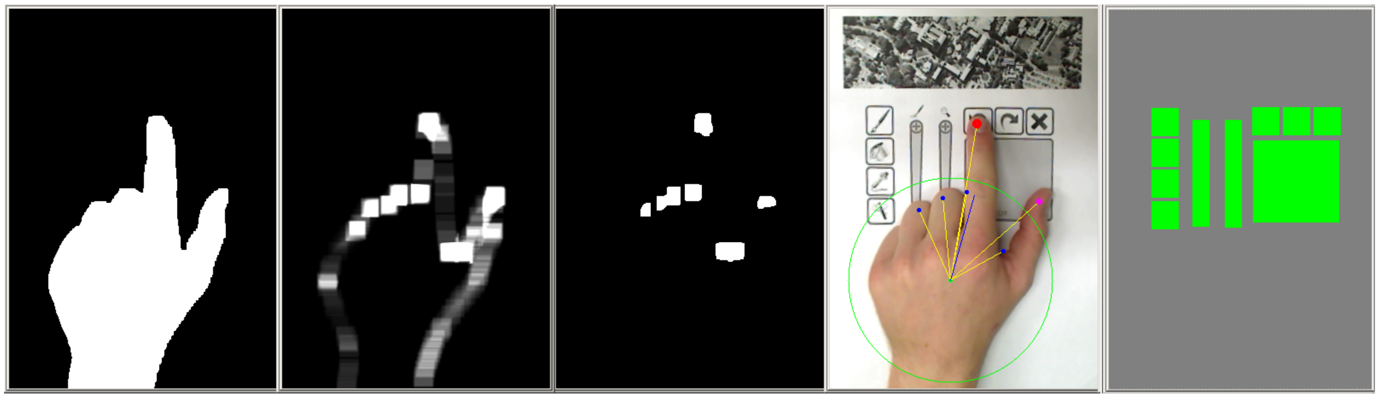

Fig. 2. The stages of the computer vision processing. The 4th image displays the results of the processing overlaid on the transformed camera feed. The red circle marks the centroid of the pointing finger and the green dot marks the hand centroid. The green circle represents the average distance of the corner points from the hand centroid. The last image shows the interaction zones as defined in the .SVG vector image.

would also require specific interfacing efforts for each application. The currently implemented key-emulation is universal, and should continue to work even as target applications progress, whereas a tailored integration might by version-specific. The current implementation is limited by what can be achieved with hot-keys.

The program could be made to integrate with multiple programs simultaneously, to the point where no other input devices would be required for certain workflows. Key presses can be sent to specific programs even when they are not in the window-manager's focus. Additionally, multiple guide pages could be set up and registered to the tracker, allowing either quick changes between workflows or a greater range of commands.

The image header at the top of the guide page should not be necessary for page tracking, assuming the page contrasted sufficiently with the table underneath it. This would allow a greater operational area for the same visual field of the camera. The visible edges of the page could be detected and extrapolated until they intersected, and then the homography calculated directly from the resulting quadrilateral.

The response of the system would not have been fast enough to be usable with real-time applications such as controlling music playback live, but was fast enough to act as an input device for Photoshop. Neither code analysis nor extensive optimisations were performed. Using a profiler may have uncovered inefficient areas of the code, or shed light on the disparity between the frame rates at different loop wait times and the CPU usages.

The current method of deciding whether a corner represents a finger appears to be the largest cause for false positive results. The half of all of the corner points which are further from the hand centroid than the average are considered fingers. This method performs poorly when there is an imbalance in finger points and non-finger points, especially when points are near the average. One alternative would be to decide on the boundary radius based on the magnitude of the palm area only. The fingers can be removed by eroding and then dilating the image, and then the smallest circle which fits around the remaining palm area can be used to determine which points are fingers.

Various changes to the computer vision aspects of the project could be made. The thresholding during the hand extraction stage could be made to be adaptive thresholding. Lighting changes can significantly effect the perceived saturation of an image, so a dynamic threshold point could make the program more robust. Circular morphology and corner detection kernels could be used instead of the default square kernels, which would reduce the variations in results due to rotation of the hand relative to the page. The registration that needs to occur with a change in camera or page position could be made automatic. When the difference between the current and previous image is large, it is likely that a perspective shift has occurred, and this could be used to trigger a new registration. Alternatively, a less computationally expensive tracking algorithm could be used continuously.

Currently the application is optimised for hands at a certain distance from the camera. The testing was performed with A4 sized paper; if the user was to use larger sheets, the camera would have to be mounted further from the table to be able to view the entire sheet. The techniques employed in the algorithm are scale-sensitive, so the performance of the program would degrade. The scale-dependent parameters could be automatically changed to suit different scales by measuring the moment of the palm area of the hand, and applying a scale-factor. The larger sheets of paper would have to be scaled down to be processed as quickly by the system, which could result in loss of accuracy.

VI. Conclusion

The system described in this paper demonstrates the potential of a vision-based system for use as an additional input device allowing more intuitive, natural computer interactions and greater productivity. According to the evaluation metrics defined earlier the project was successful. The system tracked the user's hand and all visible fingers consuming less than 10% of the computer's CPU, with reasonable response and accuracy. Finally the hardware cost of the system to many computer users would be zero due to the system's reliance on only standard computers and webcams. This interface has the potential for a wide range of applications, improving the productivity and workflow of many computer programs.

Acknowledgement

References

[1] S. Card and T. Moran, "The psychology of human-computer interaction," 1983.

[2] Z. Kalal, K. Mikolajczyk, and J. Matas, "Face-TLD: Tracking-Learning-Detection Applied to Faces," *International Conference on Image Processing*, 2010.

[3] ——, "Forward-Backward Error: Automatic Detection of Tracking Failures," *International Conference on Pattern Recognition*, 2010.

[4] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints," *Conference on Computer Vision and Pattern Recognition*, 2010.

[5] T. Singh, "Visual Interfaces To Computers," 2004, columbia University.

[6] J. MacCormick and M. Isard, "Partitioned Sampling, Articulated Objects and Interface-Quality Hand Tracking," Ph.D. dissertation, University of Oxford, 2000.

[7] J. MacCormick and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," *Int. J. Comput. Vision*, vol. 39, pp. 57–71, August 2000. [Online]. Available: http://portal.acm.org/citation.cfm?id=360088.360095

[8] G. M. Farinella and E. Rustico, "Low Cost Finger Tracking on Flat Surfaces," *Eurographics Italian Chapter Conference*, 2008.

[9] J. L. Crowley, F. Brard, and J. Coutaz, "Finger tracking as an input device for augmented reality," 1995, pp. 195–200.

[10] A. D. Wilson, "Playanywhere: a compact interactive tabletop projection-vision system," in *Proceedings of the 18th annual ACM symposium on User interface software and technology*, ser. UIST '05. New York, NY, USA: ACM, 2005, pp. 83–92. [Online]. Available: http://doi.acm.org/10.1145/1095034.1095047

[11] C. Hardenberg and F. Brard, "Bare-hand human-computer interaction," in *Proceedings of the ACM Workshop on Perceptive User Interfaces*, 2001, pp. 1–8.

[12] I. S. MacKenzie, A. Sellen, and W. A. S. Buxton, "A comparison of input devices in element pointing and dragging tasks," in *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, ser. CHI '91. New York, NY, USA: ACM, 1991, pp. 161–166. [Online]. Available: http://doi.acm.org/10.1145/108844.108868

[13] A. Clark, R. Green, and R. Grant, “Perspective correction for improved visual registration using natural features,” 2008, pp. 1–6.