A CAMERA-BASED POINTING INTERFACE FOR MOBILE DEVICES

Orazio Gallo, Sonia M. Arteaga, and James E. Davis

School of Engineering, University of California, Santa Cruz.

ABSTRACT

As the applications delivered by cellular phones are becoming increasingly sophisticated, the importance of choosing an input strategy is also growing. Touch-screens can simplify navigation by far but the vast majority of phones on the market are not equipped with them. Cameras, on the other hand, are widespread even amongst low-end phones: in this paper we propose a vision-based pointing system that allows the user to control the pointer's position by just waving a hand, with no need for additional hardware.

Index Terms ***—*** *Machine vision, mobile devices, pointing systems, computer interfaces, tracking.*

1. INTRODUCTION

After over two decades virtually dominated by the use of mice and keyboards, the last few years have been characterized by the rise of alternative input devices, generally designed for improved ergonomics. In particular, the crucial ability to control a pointer on the screen has driven the study and development of a large number of devices such as pen-tablets, touch-pads, touch-screens, and TrackPoints. Developing a sensible pointing strategy is especially important for hand-held devices, such as cellphones. The very nature of these devices, in fact, requires that both access and manipulation of information be as quick and efficient as possible.

High-end phones frequently utilize touch-screens as a strategy to solving this problem. Touch-based technology provides a very intuitive interface for navigation but does suffer from limitations: for non stylus-based touch-screens, the user's finger size can bound the pointing accuracy and generate even more frustration than a traditional input method. Moreover, for relatively small screens such as those available on many mobile devices, the area of most interest might be partially occluded by the fingers. For stylus-based products, on the other hand, the need for an additional device itself may attenuate the benefits of the approach.

We propose to take advantage of the fact that screen and camera are usually on opposite sides of the phone. With this setup, the user can look at the screen while the movement of a finger or a hand is captured by the camera, tracked by simple yet robust algorithms, and finally used to control the pointer position. This method builds on a tool available on virtually all cellphones; moreover, because it does not require the finger to be moved on a physical surface, the spatial sensitivity can be adjusted by varying the distance between phone and finger. Finally, because the hand is constantly behind the screen, there is no risk of occlusions.

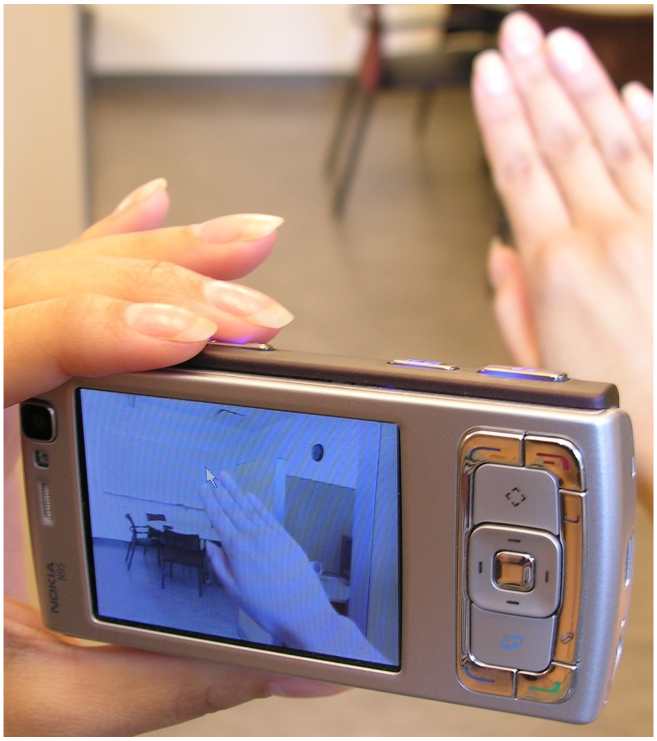

Figure 1: *The user can control the pointer by simply moving his or her hand behind the phone.*

Our contribution lays in the definition of a simple yet effective pointing strategy that only requires hardware available on most devices. We also present a fast algorithm that is able to deal with factors neglected by other approaches, such as camera shake or poor picture resolution. Finally, we performed a user study that confirmed that our algorithm, capable of real-time execution on a Nokia N95 phone, overcomes some of the limitations of other pointing strategies.

2. RELATED WORK

Recent advancements in computer vision have paved the way for designing natural control of the cursor's position; many systems are available that allow for simply moving a finger in 2D on a generic surface, or even in 3D. The common denominator to these techniques is the need for finger tracking. However, despite the interest generated by this subject, none of the many strategies proposed so far has become a de facto standard. What follows is a review of existing methods, most of which are successful but do not satisfy one or more of the constraints of our problem.

A common approach to finger tracking is marker-based. Many augmented reality applications rely on gloves with embedded retro-reflective markers for high accuracy and reliability as in Ulhaas et al. [1]. Zeng et al. use a simple color patch to ensure trackability of the fingers as they move over other parts of the body [2]. The main drawback of marker-based methods is the need to "wear" special aids.

A second class of tracking algorithms exploits temperature as a means of segmenting objects. The EnhancedDesk uses a simple threshold on the infrared image obtaining very robust results [3]. The ThermoTablet extends this idea to track fingers, airbrushes, or even brushes dipped in warm water, as they move on a surface upon which images are retro-projected [4]. Though fairly robust, thermal-based methods require expensive infrared cameras that are not available on commercial cellular phones.

A larger set of methods for bare hand tracking is based on visible light: contour-based methods are very popular [5, 6, 7, 8], other approaches such as correlation [9] and wavelets-based methods are also common [10, 11]. These methods are generally computationally intensive and some even require multiple cameras. Less computationally intensive approaches use color segmentation in the appropriate color space [12, 13, 14]; however, most of these are still beyond the capabilities of a mobile device. In addition, none of the above techniques addresses the problem of hand-held cameras which are affected by blur and relative motion with the scene background.

3. METHODS

For a pointing strategy to work, real-time execution is crucial and this, in turn, poses strict requirements on the computational complexity and memory efficiency of our tracking algorithm. In every frame, the aim is to detect the position of a hand placed in front of the camera (i.e. behind the screen). In order to achieve this efficiency, we have designed an algorithm suitable for uncluttered backgrounds. In the following we describe how we perform the tracking to achieve the required robustness at the desired frame-rate.

3.1. Algorithm description

Our algorithm detects fingers using image gradients in a color space relevant to human skin. The algorithm's flow is shown in figure 2. Skin pigmentation is well represented in the *YCrCb* color space: by working in the red chrominance channel, we can expect that the magnitude of the gradient be higher around the hand's contours than anywhere else. We calculate row and column gradient images, and then sum along each axis. In order to find a consistent pointer location

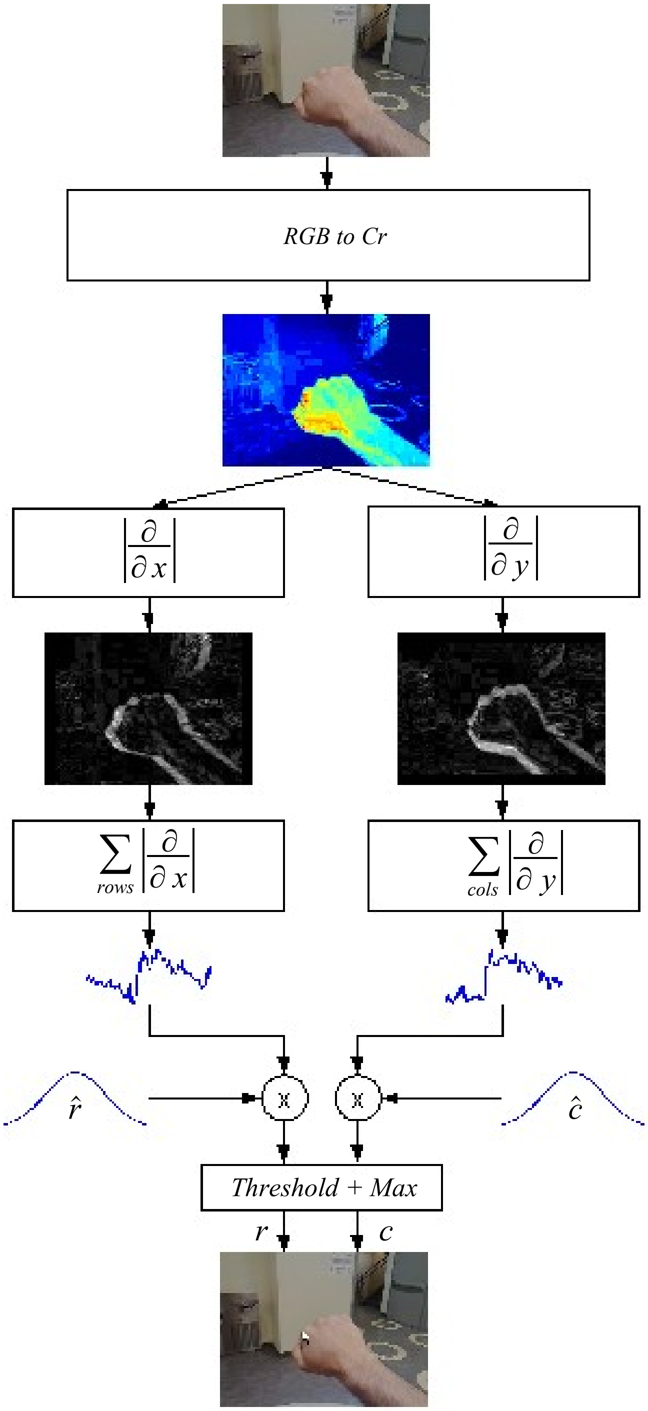
RGB to Cr
$\left|\frac{\partial}{\partial x}\right|$
$\left|\frac{\partial}{\partial y}\right|$
$\sum_{rows}\left|\frac{\partial}{\partial x}\right|$
$\sum_{cols}\left|\frac{\partial}{\partial y}\right|$
$\hat{r}$
$\hat{c}$
Threshold + Max
r
c

Figure 2: *Flow chart of the algorithm. Horizontal and vertical gradients of the red chrominance (Cr) are accumulated along rows and columns, and are weighted with Gaussian distributions centered around the coordinates predicted from previous frames. The first maximum above threshold provides the coordinates of the finger.*

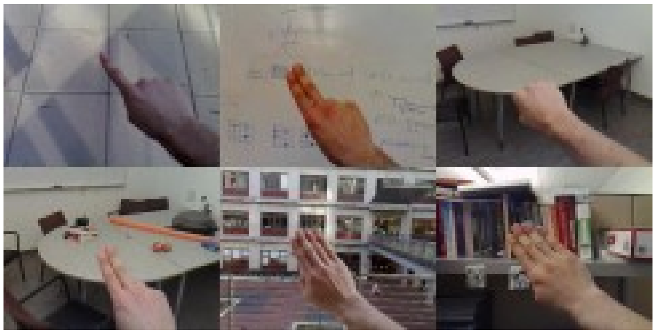

Figure 3: *The proposed pointing interface can work in different environments. The top row shows a sample of backgrounds that are suitable for use with our interface. Note that the results are not affected by the hand's pose, and both pointing fingers and fists can be used. As the background becomes cluttered, however, the presented algorithm is insufficiently robust (bottom row).*

with relation to the hand, the finger tip is localized by detecting the first maximum from the top and from the left in the summed gradient vectors.

A small memory footprint is achieved by directly calculating the gradient vectors, instead of computing the whole gradient image. This approach is similar to that of Adams et al. [15]. We create two mono-dimensional vectors, gradX and gradY. Given a column *c* and a row *r*, the gradient vectors are computed as:

$$gradY(c)=\sum_i \left|I_{Cr}(i,c)-I_{Cr}(i-n,c)\right| ,$$

$$gradX(r)=\sum_j \left|I_{Cr}(r,j)-I_{Cr}(r,j-n)\right|$$

where I_{Cr} is the red chrominance channel of the image and n is the distance, in pixels, across which the difference is computed. The use of $n>1$ is motivated by the fact that the transition from background to foreground is not abrupt and takes, in fact, a few pixels, primarily due to camera optics and motion blur.

In a noise-free image, the first peaks in gradX and gradY would provide the coordinates (c,r) of the fingertip. To attenuate the effect of noise, we employ a fixed threshold (about 15% of the typical maximum value) so that only significant pixel differences contribute to the gradient.

Inter-frame information is exploited by predicting the position of the finger based on the information collected in previous frames. Since the finger trajectory for the purpose of controlling a pointer can be very well approximated by a piecewise linear function, we use a simple linear predictor:

$$\hat{c}_t=2\cdot c_{t-1}-c_{t-2} ,$$

$$\hat{r}_t=2\cdot r_{t-1}-r_{t-2}$$

where *c* and *r* are the coordinates of the point and the subscripts refer to the frame index. We then weigh the gradients with a Gaussian distribution centered on the predicted position and with standard deviation 1.5 times the size of the image; the large variance makes the Gaussian a vague prior, important to allow unexpectedly large movements. The finger location is finally determined as the first maximum that is higher than a percentage of the highest value in the resulting vectors. Lastly, we smooth the location of the detected position by averaging it over multiple frames.

3.2. Implementation

We used a Nokia N95 for development and testing. On this device, the acquisition frame-rate can be as fast as the viewfinder, which runs at 30 fps providing 320 by 240 (QVGA) images. The core of the algorithm was implemented in Symbian C++, using exclusively fixed point arithmetic, while interface and camera control are managed through Python for S60.

4. RESULTS

4.1. Performance

The algorithm takes about 20 ms to run on a single viewfinder image, allowing real-time operation. As long as the exposure is short enough to eliminate motion blur, the hand can be waived as fast as needed.

The tracking robustness of the algorithm depends on the background: figure 3 shows a collection of situations of increasing difficulty. The algorithm was designed to work on reasonably uncluttered backgrounds, and works well in the situations shown in the top row. We have found that acceptable background environments are nearly always available, often a wall or ceiling. Of course, the algorithm's performance degrades as the scene gets cluttered by the presence of objects whose color presents a strong gradient in the red chrominance channel.

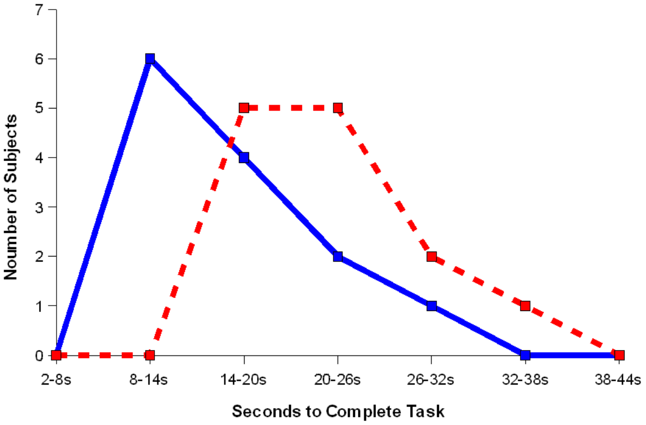

Figure 4: *Distribution of task completion times. The blue solid line refers to our approach while the red dotted line refers to the button-based method. Note that the mean is significantly lower for the vision-based interface: 16.3s vs 23.2s.*

4.2. Usability

In order to evaluate the proposed pointing strategy, we designed a test representative of a common task on mobile devices: navigation of an image too large to fit on the display. Using our interface and moving his or her hand behind the phone, a user can drag the image around to visit its different regions. We asked several subjects to browse pictures both with our method, and by using the phone's buttons, which provide an inherently discrete input strategy. We used pictures of groups of people and measured the time that was necessary for them to count the number of people with red eyes. Red eyes were added to randomly selected faces in the pictures. Before taking the test, the subjects were allowed to familiarize themselves with both pointing strategies as well as the original pictures and the task. Showing the images beforehand allowed us to focus our results on *navigation* since it provided an initial estimate of the position of the different faces, thus reducing the effect of different search strategies.

A total of 13 subject were used. The results of the comparison are shown in figure 4. The average time to complete the task was 16.3s (std 6.1s) with our approach and 23.2s (std 5.9s) with the discrete strategy, thus indicating that our vision-based approach is effective. The experiment also showed that the proposed interface feels fairly natural to a new user: it took on average a minute for our users to be ready to take the test.

5. LIMITATIONS AND FUTURE WORK

In order to achieve an efficient implementation, the tracking algorithm was designed for uncluttered backgrounds. Future work could seek to improve the level of clutter in which hand tracking is possible.

The proposed approach proved to be robust across different hand configurations: a fist, an open hand, or a single finger all get tracked with the same reliability. This suggests that gesture recognition can be integrated to add capabilities such as clicking or dragging.

6. CONCLUSIONS

In this paper we presented a novel pointing interface for mobile devices. Our approach exploits the ubiquity of cameras and the naturalness of controlling a pointer by freely moving a hand. The algorithm runs in real-time on a cellphone and we have verified that the proposed method provides better usability than buttons for a common navigation task. A video showing the algorithm in action is available at:
http://www.soe.ucsc.edu/~orazio/demoICIP08.html

7. ACKNOWLEDGMENTS

The authors would like to thank Nokia for providing the phones used for this paper. Natasha Gelfand and Andrew Adams provided invaluable support for getting started with developing on the N95. Mariano I. Lizarraga helped greatly with stimulating discussions about both the algorithm and its implementation.

8. REFERENCES

[1]: K. Dorfmueller-Ulhaas and D. Schmalstieg, "Finger Tracking for Interaction in Augmented Environments", Proc. of the 2nd ACM/IEEE Int'l Symposium on Augmented Reality, pp. 55-64, 2001.

[2]: J. Zeng, Y. Wang, M. Freedman, and S.K. Mun, "Color-Feature-Based Finger Tracking for Breast Palpation Quantification", Proc. Int'l Conf. on Robotics and Automation, pp. 2565-2570, 1997.

[3]: H. Koike, Y. Sato, and Y. Kobayashi, "Integrating Paper and Digital Information on EnhancedDesk: A Method for Realtime Finger Tracking on an AugmentedDesk System", ACM Transactions on Computer-Human Interaction, pp. 307-322, 2001.

[4]: D. Iwai and K. Sato, "Heat Sensation in Image Creation with Thermal Vision", Proc. of the ACM SIGCHI Int'l Conf. on Advances in Computer Entertainment Technology, pp. 213-216, 2005.

[5]: H. Zhou and Q. Ruan, "Finger Countour Tracking Based on Model", Proc. of the Conf. on Computers, Comunications, Control and Power Engineering, pp. 503-506, 2002.

[6]: J.M. Rehg and T. Kanade, DigitEyes: Vision-Based Human Hand Tracking, Tech. report, CMU-CS-93-220, 1993.

[7]: J. MacCormick and M. Isard, "Partitioned sampling, articulated objects, and interface-quality hand tracking", European Conference on Computer Vision, pp. 3-19, 2000.

[8]: J. Segen, "Gesture VR: Vision-Based 3D Hand Interface for Spatial Interaction", ACM Multimedia Conference, pp. 455-464, 1998.

[9]: R. O'Hagan and A. Zelinsky, "Finger Track - A Robust and Real-Time Gesture Interface", Australian Joint Conference on Artificial Intelligence, pp. 475-484, 1997.

[10]: J. Triesch and C. von der Malsburg, "Robust Classification of Hand Postures against Complex Backgrounds", International Conference on Authomatic Face and Gesture Recognition, pp. 170-175, 1996.

[11]: R.R. Brooks, L. Grewe and S.S. Iyengar, "Recognition in the Wavelet domain: A survey", Journal of Electrical Imaging, pp. 757-784, 2001.

[12]: M. Bencheikh-el-hocine, M. Bouzenada, and M.C. Batouche, "A New Method of Finger Tracking Applied to the Magic Board", Proc. of the Int'l Conf. on Industrial Technology, pp. 1046-1051, 2004.

[13]: J. Letessier and F. Berard, "Visual Tracking of Bare Fingers for Interactive Surfaces", Proc. of the ACM Symposium on User Interface Software and Technology, pp. 119-122, 2004.

[14]: W. M. Tsang and K. Pun, "A Finger-Tracking Virtual Mouse Realized in an Embedded System", Proc. Int'l Symposium on Intelligent Singal Proc. and Commu. Systems, pp. 781-784, 2005.

[15]: A. Adams, N. Gelfand, and K. Pulli, "Viewfinder Alignment", Eurographics, 2008.