Visual Tracking of Bare Fingers for Interactive Surfaces

Julien Letessier, François Bérard
CLIPS-IMAG
BP 53
38041 Grenoble Cedex 9
FRANCE
julien.letessier | francois.berard@imag.fr

ABSTRACT
Visual tracking of bare fingers allows more direct manipulation of digital objects, multiple simultaneous users interacting with their two hands, and permits the interaction on large surfaces, using only commodity hardware. After presenting related work, we detail our implementation. Its design is based on our modeling of two classes of algorithms that are key to the tracker: Image Differencing Segmentation (IDS) and Fast Rejection Filters (FRF). We introduce a new chromatic distance for IDS and a FRF that is independent to finger rotation. The system runs at full frame rate (25 Hz) with an average total system latency of 80 ms, independently of the number of tracked fingers. When used in a controlled environment such as a meeting room, its robustness is satisfying for everyday use.

Categories and Subject Descriptors: H.5.2 **[User Interfaces]**: Input devices and strategies; I.4.8 **[Scene Analysis]**: Tracking -- *fingers*

Additional Keywords and Phrases: finger tracking with computer vision, large interactive surface, multi-user multi-hand interaction

INTRODUCTION
We present a system that tracks the 2D position of the tips of bare fingers on a planar display surface. The tracker is founded on computer vision techniques that process a video stream in real time. The position of the tip of the fingers is used to control a Graphical User Interface (GUI) projected on the surface. The maximum number of tracked fingers is only constrained by the space on the surface, thus the system supports multi-handed, multi-finger interaction of a group of users. The provided interaction is also more direct than the interaction offered by the mouse: the input device acquisition phase (grasping the mouse) is suppressed because the finger *is* the input device.

This work is aimed at providing a low-level, event-based input mechanism for a range of novel interactive system that supports multiple users interacting with a single system. Examples of such systems include the "Magic Table" [1] and the "RoomPlaner" [9].

UIST '04, October 24-27, 2004, Santa Fe, New Mexico, USA.

We focus on the implementation of a computer vision system that processes the video stream of a single standard video camera and doesn't require any equipment of the surface. The key advantages of this particular setup are the following:
– computer vision hardware is *off-the-shelf* and *affordable*; we use low-end Firewire (IEEE 1394a) cameras, a standard video projector and a recent PC;
– the setup is *transportable* and *compact*; the PC used can be a laptop, thus making the whole hardware under 5 kg, and the whole system can be ceiling-mounted;
– the hardware is *physically robust* when setup out of reach from the users;
– *richer data* than simple fingertip location can be provided to the application developer using the same setup and video stream; for instance an object tracker (for graspable interfaces), or a surface digitizer [1].

The design choices of the system are justified by a number of human-centered requirements (described in [6] and [1]): low system latency, static stability, millimetric resolution, and robustness to our setup conditions. We also emphasis our work on *autonomy*: the system should not require complex setup or user maintenance.

RELATED WORK
Most finger tracking for Human-computer interaction make use of equipped surfaces. Commercial products include *DViT* from Smart Technologies (smarttech.com/dvit) that can only track 2 fingers and would be confused by a cup laid on the surface. Research prototypes such as the *DiamondTouch* [3] and the *SmartSkin* [5] can track many fingers but all surface-equipped solutions share some limitations: user interaction is required for calibration, the hardware is at reach from the users, and the device has to be as large as the interactive surface, which limits its portability. It should also be noted that the various research prototypes rely on custom hardware: reproducing them represents a non-trivial effort.

Our approach is to track fingers from a standard camera view, without using custom or expensive hardware (e.g. a thermoscopic camera such as in [4]). Aspect-oriented segmentation methods (such as color models, region-growing or image differencing) are sensitive to camouflage and occlusion. We choose to use image differencing because it can be made autonomous, provides good resolution, and allows to balance performance and robustness.

SYSTEM DESCRIPTION

Our system is founded on Image Differencing Segmentation (IDS). IDS generates a *similarity map*: an image where the value of each pixel is the probability that the pixel represents an object of the foreground. Difference maps must be post-processed in order to extract finger position. Following other shape filtering proposals [4][6] we propose an original shape filtering algorithm that is optimized in processing and invariant to finger orientation.

Our approach is a four-step process: (*a*) foreground extraction using IDS yields a grayscale difference map, (*b*) automatic thresholding converts it into a binary map, (*c*) shape filtering extracts the fingertip positions, and (*d*) association generates high-level events (such as "motion") for the client application. We detail each of these steps in the following subsections.

Foreground extraction

The literature abounds with variants of IDS. In order to characterize and compare these variants, we first propose a general model of IDS. An IDS variant can be modeled as the combination of three choices:

– the form of the background model,
– the metric used to compare the background model and the current image,
– the background maintenance process.

Background model. Background modeling is achieved by modeling each pixel variations. This pixel-level model can be as complex as a mixture of Gaussians in the Hue Saturation Value (HSV) color space [8] or as simple as a simple luminance value [6]. In our operating conditions, pixel variations are due to camera and lighting noise, and to background changes; they are roughly the same over the image. Consequently, we model the background as a simple image: each pixel of the model is the mean over time of the measured pixel values.

Comparison metric. We note that hand and fingers cast a shadow on the surface when they are close to it. If shadows are segmented as part of the foreground, they make the task of finger shape recognition very difficult because they extend the finger silhouette to arbitrary shapes and connect fingers together [6].

We propose a new metric that is purely color-based (hence naturally robust to shadows) and which simplicity supports computational optimization. It is the Euclidian distance between corresponding pixels in the (r,g) normalized chromaticity plane. We name it Chrominance Euclidian Distance (CED). The similarity between pixels p and p' is computed as $d(p, p')$ given by

$$p = [R,G,B]$$

$$[r, g] = [R/(R + G + B), G/(R + G + B)]$$

$$d(p, p') = \|[r, g] - [r', g']\|$$

Figure 1 show an example of the similarity map obtained using the CED metric.

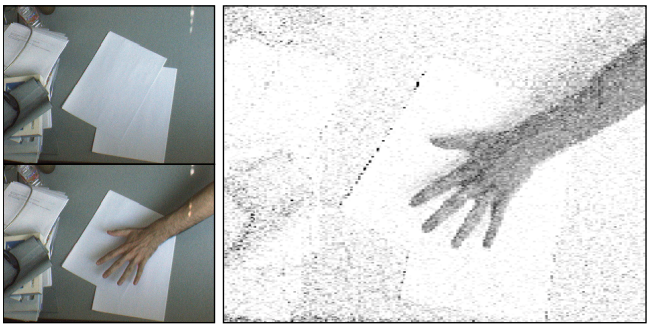

Fig. 1. Segmentation example using the CED metric. Background model (a), current frame (b), similarity map (c). The map is noisy but insensitive to lighting and shadows, and provides high-resolution contours.

Background maintenance. In order to adapt the segmentation process to a changing background, we maintain the background model *Bg* by computing the average of recent pixel measurements. This is approximated by a running average. Let the current frame be *Im*; the background model at time $t + 1$ is given by

$$Bg^{t+1}_{(x,y)} = \alpha^{t}_{(x,y)} \cdot Im^{t}_{(x,y)} + (1 - \alpha^{t}_{(x,y)}) \cdot Bg^{t}_{(x,y)}$$

The learning rate $\alpha^{t}_{(x,y)}$ is biased by the IDS results, i.e. given small values over segmented objects, and high values over the background.

Dealing with the projected feedback. In the case of an interactive surface, we project visual feedback on the surface. The feedback changes the appearance of the background and thus causes a lot of false alarms in the difference map. As the projection on a white surface is much brighter than the physical objects in the scene, we can eliminate this effect by augmenting the camera gain (causing overexposure on purpose).

Automatic thresholding

The shape filtering stage presented below requires the similarity map to be transformed into a binary image. Since our segmentation method ignores luminance variations over the image, we use a uniform threshold θ^t. We determine θ^t automatically from the similarity map. A typical histogram of the similarity map exhibits two modes. The lower mode represents the background noise; it contains roughly 80% of the pixels. Ideally θ^t should be chosen to eliminate this first mode while preserving the other mode(s). We approximate the first mode's moments by its median (m_0) and median absolute deviation (m_1); θ^t is then given by

$$m_0 = \text{median}_{(x,y)} d^t(x, y)$$

$$m_1 = \text{median}_{(x,y)} \left| m_0 - d^t(x, y) \right|$$

$$\theta^t = m_0 + 4 \cdot m_1$$

d^t being the similarity map at time t. The computed threshold is stable and empirically close to manually chosen thresholds.

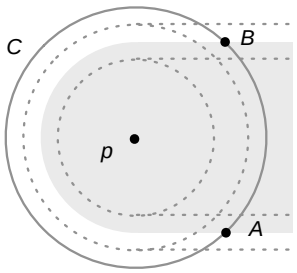

Fig. 2. A simple geometric model of the fingertip, used in our Fast Rejection Filter. (*gray*): the observed finger; (*dashed*) the smallest and largest finger models; (*full circle*) the scanned contour.

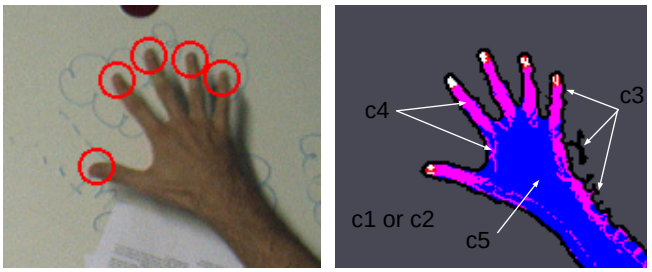

Fig. 3. Typical output of our Fast Rejection Filter. (*right*) rejection map: grey pixels where rejected by c_1 or c_2, black by c_3, pink by c_4, blue by c_5. The detected pixel clusters are white, their centers are represented by red circles on the left image.

Shape filtering

Shape filtering for fingertip detection (as introduced in [4]) is a method that scales to any number of fingers with no performance hit because it processes the input image as a whole. The idea behind shape filtering is to use simple geometric criteria that characterize the searched object, and verify these criteria for each pixel in the (binary) image.

Formally, a geometric model can be presented as a set of binary characteristics,

$$\{c_1, \ldots c_n\} \qquad c_k(x, y) \in \{0, 1\}$$

where 0 stands for "the pixel doesn't match" and 1 for "the pixel possibly matches the model". The c_k are independent, and ordered according to both their performance (low cost characteristics first) and the number of pixel they reject (characteristics that reject many pixels first). Once this is defined, the filter algorithm is trivial; we call its instances Fast Rejection Filters (FRFs):

```
for each pixel (x,y):
  for each k:
    if c_k(x,y) = 0, skip pixel;
  mark (x,y) as a candidate;
return the map of candidates.
```

A simple geometric model for the finger is a long rectangle ended by a half disc which size can vary between the smallest and the largest possible visible finger size (∅ 9 to 20 mm). This model is illustrated by figure 2. Using the FRF formalism, we propose and implement a set of characteristics inspired by [4] and [6] but improved by the use of a circle instead of a bounding rectangle. This way, we insure that our filter is not sensitive to finger orientation. Our filter characteristics are:

– c_1: the pixel p is classified as foreground in the binary map;
– c_2: p is within a region of connected pixels that is large enough to be a hand (20 sq. cm);
– c_3: p is surrounded by a fully segmented disc (∅ 9 mm);
– c_4: while scanning the contour C (∅ 20 mm) around p, exactly one connected component is encountered;
– c_5: the distance AB is coherent with the size of a finger.

The successive rejection of pixels is illustrated on figure 3. The pixels that are not rejected by the filter are then clustered into groups of connected pixels. The output of the FRF is the list of (x,y) coordinates of the vertical and horizontal medians of these clusters. The algorithm parameters can be determined using only the view scale. In particular the perspective deformation of the camera view does not significantly hinder the filter.

Association

The output of the FRF has to be translated into events for the client application. The difficulty is to identify the fingers so that the same ID parameter is provided with the *appear*, *motion* and *disappear* events that concern one particular finger. We implement a very simple closest neighbor algorithm detailed in [1]. After receiving the FRF result, the algorithm executes the two following steps:

– each detected fingertip at frame t is matched to the closest memorized finger at frame t-1. If there is no memorized finger or the closest one is further than the distance threshold , we generate an *appear* event with a new finger ID. Otherwise, if the matched finger has moved of more than a fixed motion threshold, we generate a *motion* event. The motion threshold is introduced in order to satisfy the stability requirement of the tracker's output.
– memorized fingers that were not matched to any fingertips in the previous step are forgotten and disappear events are generated with their IDs.

Since the speed of fingertips is typically under 2.5 m/s, we choose the distance threshold as the maximum finger movement per frame: for quarter-PAL frames at 25 Hz, we use 40 image pixels as the threshold. We actually add a 50 ms. time window before generating an *appear* event and a 200 ms. time window before *disappear* events in order to add tolerance to false alarms and misdetections. The 200 ms. time window allow the system to successfully track finger that are momentarily hidden to the camera by an occluding object.

Our system has no way to detect the contact of a finger with the surface corresponding to a mouse *button* event. We add a spatiotemporal filter on top of the association component of the system. This filter detects pauses in finger trajectories and reports a *button* event when a pause lasts more than 300 ms.

All events are complemented by the (*x*,*y*) location of the finger at the time of the event, in display coordinates. The conversion between camera and display coordinates is achieved using an off-line, automatic calibration process. A set of 9 bright disks is projected on the surface at known coordinates. Their location is easily recovered in the camera image using automatic thresholding and connected component analysis, and the set of corresponding points is used to compute the projective transformation between camera and projector image [2].

EVALUATION SUMMARY

We evaluated the performances of the system as well as its usability during one formal (Fitts-like) and one informal user experiment. The following is a summary of the results of the evaluations.

Autonomy. When used in a stable, controlled lighting environment, the system is almost autonomous thanks to the automatic thresholding and camera-projector calibration algorithms. The only parameter is the view scale, which has to be determined manually.

Resolution and Stability. The PAL video stream is processed at quarter definition (388x284). While the output of the shape filter has sub-pixel accuracy, the instability of the video signal forces us to set a 1.6 pixel motion threshold before reporting a motion event. In our setup, the camera view is set to encompass the 100 cm x 75 cm surface. The resolution is thus approximately 4 mm which, with a 1024x768 display, translates to 4 display pixels.

Latency. On a 1.4 Ghz PowerPC G4 machine, we measure an average latency of 80 ms with a standard deviation of 18 ms. This is not ideal when considering the 50 ms requirement expressed in [7] but it is close to ideal.

Robustness. The typical robustness failures are taken care of by the system:

– Occlusions, as caused by the body or a limb, are rare in our setup (users are sitting around the table). On the other hand occlusions between different users' hands are frequent. We compensate this problem by allowing a 200 ms time window before considering that a finger has actually "disappeared".
– In spite of the simplicity of the "association" algorithm presented above, finger aliasing rarely occurs during tracking. This allows unconstrained user movements (except extremely fast ballistic gestures).
– Finally, the performance of the system doesn't degrade when the number of tracked fingers increases.

Usability. During the formal experiment, we observed that our finger tracker supports target acquisition tasks with an efficiency that is in the same range as with a mouse. In the informal experiment, subjects were asked to spatially reorganize a set of photographs. Users easily interacted with the system. Bi-manual interaction was natural and frequently used. The main problem was to detach a finger from a photograph: the finger had to be hidden from the system. It seems suitable to introduce a 300 ms. pause for detaching (i.e. the same as for attaching).

CONCLUSION

Our finger tracker mostly satisfies the requirements of Human-Computer Interaction. It provides the location of more than 20 fingers at 25 Hz with 80 ms average latency, using commodity hardware. The output is stable and the tracker is robust to typical usage in a controlled lighting environment. User experiment shows that the system support efficient interaction. Our contributions include the modeling of two families of vision algorithms that are key to finger tracking, and a set of design choices for their instantiation. Our Image Differencing Segmentation (IDS) is insensitive to shadows and our Fast Rejection Filter (FRF) is insensitive to finger orientation.

This work can be improved in many ways. Its robustness should be maintained in a less controlled environment such as a train station or an airport hall. Also, providing more information (such as finger orientations and connection to the hand) will increase the design space of client developers.

REFERENCES

1. Bérard, F. The Magic Table: Computer-Vision Based Augmentation of a Whiteboard for Creative Meetings. CD-ROM proceedings of the *IEEE International Conference in Computer Vision, Workshop on Projector-Camera Systems* (PROCAMS'03), Nice, France (2003).
2. Criminisi, A. Reid, I. Zisserman, A. A plane measuring device. *British Machine Vision Conference*, UK (1997).
3. Dietz, P. and Leigh, D. DiamondTouch: A Multi-User Touch Technology. *ACM Symposium on User Interface Software and Technology*, Orlando, Florida, pp 219-226 (2001).
4. Koike, H. Sato, Y. Kobayashi, Y. Tobita. H. and Kobayashi, M. Interactive textbook and interactive Venn diagram: natural and intuitive interfaces on augmented desk system. *SIGCHI conference on Human factors in computing systems*, The Hague, The Netherlands, pp. 121-128 (2000).
5. Rekimoto, J. SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. *SIGCHI conference on Human factors in computing systems*, Minneapolis, Minnesota, pp. 113-120 (2002).
6. Von Hardenberg, C. and Bérard, F. Bare-hand human-computer interaction. *Workshop on Perceptive User Interfaces*, Orlando, Florida (2001).
7. Ware, C. and Balakrishnan, R. Reaching for Objects in VR Displays: Lag and Frame Rate". *ACM Transactions on Computer-Human Interaction (TOCHI)*, Vol. 1, No. 4, pp 331-356 (1994).
8. Wren, C., Azarbayejani, A. Darrell, T. and Pentland, P.: Pfinder: Real-time tracking of the human body. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, (1997).
9. Wu, M. and Balakrishnan, R. Multi-Finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. *ACM symposium on User Interface Software and Technology*, Vancouver, Canada, pp. 193 - 202 (2003).