Real-Time 3D Hand Interaction: Single Webcam Low-Cost Approach

Florin Duca, Jonas Fredriksson, Morten Fjeld

TableTop Interaction Lab (http://t2i.se/), CSE
Chalmers University of Technology
Gothenburg, Sweden

ABSTRACT

The purpose of this project is to create a library that will allow its users to control 3D applications by using one or both of their hands. The final product could easily be incorporated into 3D applications, each customized to utilize a set of poses. Even though off-the-shelf motion capture gloves have reached lower prices in recent years, they are still expensive for home users. The algorithm suggested is based only on a single webcam combined with coded palm and fingers. Users should be able to code one or more of the fingers. One webcam is still somewhat constraining as two should ideally be used for 3D mapping of the hand, but by additionally using palm and finger coding we can greatly improve precision and, most importantly, reduce the processing power required for feasible real-time 3D interaction.

1 INTRODUCTION

Tracking articulated structures in a reasonable amount of time is a complex task due to the system's multiple dimensionalities. Several research projects have previously introduced different particle filter algorithms [1, 2, 3, 4, 5]. Implementations of these algorithms have proven to be too slow for real-time tracking of the hand (more than 20 seconds processing time per frame). Better frame rates were obtained using multiple cameras like in Stenger, Mendonça, and Cipolla's work [6, 7], but the required processing power still prevents their results from being used in real-time applications.

Our goal is to develop an algorithm capable of real-time performance at a rate of several frames per second using minimal system resources in order to achieve real-time one- and two-handed 3D desktop input. We also want to provide a viable solution that could be used by average home-users, which rules out access to sophisticated cameras. To achieve this goal the dimension of the system needs to be reduced while keeping essential information. We aim to reach this goal by reducing the hand model's total number of degrees of freedom (DOF). Our hand model has 16 DOF (six for the palm and two for each finger). Combining the use of a simplified hand model with a heuristic approach for coordinate determination, we expect to keep error rates relatively low while significantly reducing computation cost.

A user hand's palm and fingers are color coded as shown in Figure 1. A webcam typically sits on top of the computer screen looking down towards the user's hands. XYZ is the webcam coordinate system, where the X and Y axes are parallel to the

duca@student.chalmers.se
jonafred@dtek.chalmers.se
morten@fjeld.ch

camera view plane while the Z axis runs along the camera's line of sight. This coordinates system is drawn in red (Figure 1, right). An issue when using only one webcam is that depth measurement (Z-axis) cannot be performed very accurately. This will affect the design of transfer functions mapping user hand movements onto digital interaction and navigation spaces.

The study of real-time 3D hand interaction based on a single webcam combined with low-cost computation is a research project in the field of tabletop interaction that we recently started[1]. Besides desktop screens, we conjecture that our approach may benefit interaction with large vertical and horizontal displays. We also believe that collaborative and remote applications may benefit from the approach presented.

The remainder of this paper offers a section on color coding, followed by a description of finger tracking by blob detection, an overview of palm and finger coordinate determination, and is rounded off with a discussion and outlook section.
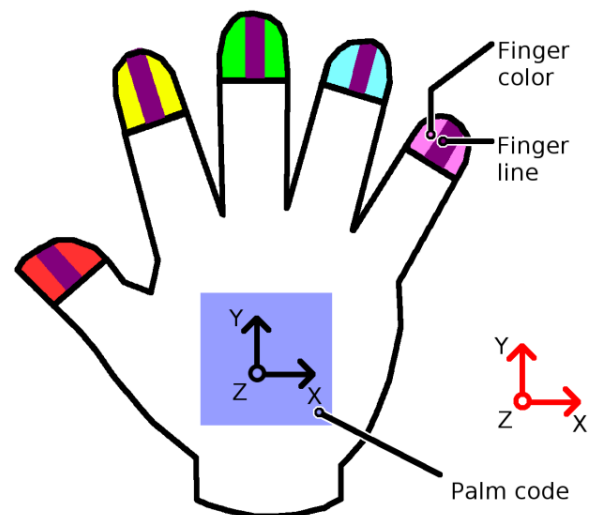

Figure 1: A user's hand with color coded fingers. Webcam coordinate system is shown in red (right) and palm coordinate system in black (center). The finger colors are (from left to right) red, yellow, green, cyan, and magenta. Finger lines on the inside of the hand share the same color (purple); finger lines on the outside of the hand share another color. The palm and the back of the hand are coded with the same color. Any additional hand gets new color codes for the fingers, the palm, and the back of the hand; finger line colors are reused.

2 COLOR CODING

We suggest an algorithm for object tracking based on colors, optionally combined with patterns. Pattern tracking is often more

---

[1] The project web site offers video documentation and ongoing work: http://www.t2i.se/projects/ht.php

exact than color tracking. Patterns can easily be fitted onto the palm and the back of a user's hand. However, as user's fingers are too small to allow for pattern tracking with typical low-end cameras. Therefore, our algorithm uses pattern-based coding of the hand's palm and back combined with color-based finger coding.

2.1 Color model

The color model utilized separates the chromatic and luminescent components of the image. We have chosen the YCrCb model where Y represents luminance and Cr and Cb represent the color's chromatic components. To determine if a pixel is part of a certain finger's color code from a chromatic perspective (not yet looking at geometric positions) the two chromatic components (Cr and Cb) must be taken into account.

2.2 Determining the best colors to use

We must take it into account that different types of webcams with different qualities and resolutions could be used. This is why, in a calibration phase, users must determine the appropriate colors to use in their environment combined with their webcam. Therefore users must print an image with the complete CrCb spectrum (Figure 2) and display it to the webcam (Figure 3). Then, after identifying the image with the complete spectrum in the captured webcam image, the system creates a histogram of the captured spectrum (Figure 4). By applying data mining classification on the histogram, the algorithm can determine the optimal colors given the environmental conditions, printer output, and webcam mode.
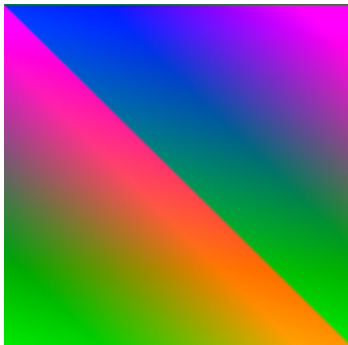

Figure 2: Full CrCb spectrum generated
with 8 bits for each of the two color components.

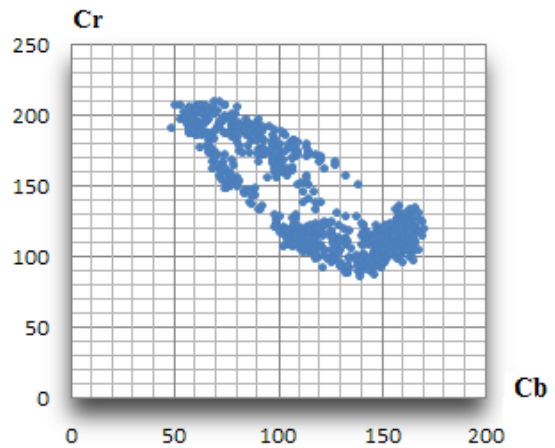Figure 3: Webcam captured image of the full CrCb spectrum
(shown in Figure2).

Figure 4: Histogram of the captured image (shown in Figure 3).

3 FINGER TRACKING BY BLOB-DETECTION

We define a blob as a region in the image that possesses certain properties as described in the following sub-section. The algorithm utilizes color matching to identify blobs, each color corresponding to a finger tip or the palm-square. As discussed below, we use lines inside the blobs to determine blob coordinates, hence making the algorithm independent of exact blob positions. Since the Gaussian noise distortion that is present in most webcams becomes more apparent in the CrCb channels, we apply a Gaussian blur filter to decrease its impact on the detection process. The finger tracking we suggest here (Figure 10) consists of a first phase and a second phase. The first phase includes blob detection while the second phase includes blob tracking over time.

3.1 First Phase: Determining approximate blobs

To determine approximate blob positions for the color codes, we use the grid decomposition technique and segment the captured image into fixed-size squares. A region (square) is part of a blob if at least one of its edges has enough pixels (specified by a threshold) in the color interval which we are currently tracking. For example, when using images with dimensions of 320 by 240 pixels with squares of size 15 by 15 pixels, a feasible threshold is 7 pixels for an edge. The edges that comply with that threshold are colored in red (Figure 5).

Figure 5: Grid-edge detection applied to webcam image. Edges complying with the threshold are colored in red (middle finger).
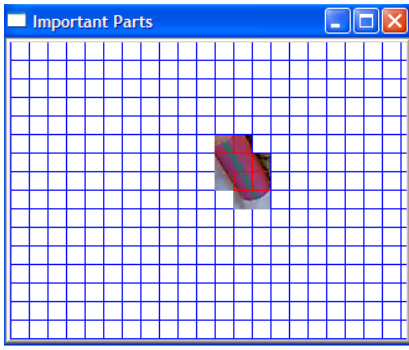
Figure 6: Image grid with approximated blob grid-parts with Figure 5 as input.

After having successfully determined the approximate blobs for a certain color code, the algorithm then goes through each detected blob and searches for the number of finger line colored pixels. If the number of found pixels is equal or higher than a specified threshold, the blob is said to be certified as a finger blob.

3.2 Second phase: Tracking blobs over time

If a finger blob was present in the preceding frame, the algorithm only looks for changes in the edge grid-parts in order to save time; otherwise the algorithm analyzes the complete image.

We define a point as a unit square of the image grid (see Figure 6). The algorithm classifies each point of the image grid as one of the following:

- **Internal point** - blob image part that is surrounded at left/right and up/down by other blob parts
- **Marginal point** - blob image part that is not surrounded by blob parts on all sides
- **Consideration point** - non-blob image part that has at least one marginal point at left/right or up/down
- **Non-important point** - neither part of a blob image part nor is it situated in adjacent image parts

The result of applying this classification using the output from the previous step (Figures 5 and 6) as input is shown below (Figure 7).
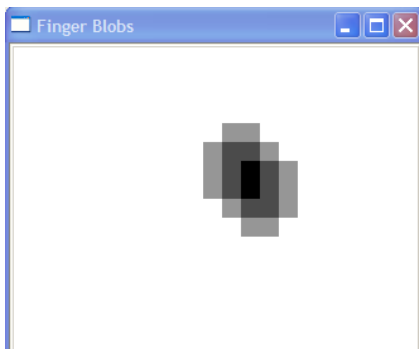

Figure 7: Detected blob regions in the previous image (Figure 6). Internal points are black, marginal points dark grey, consideration points light grey, and non-important points are white.

Taking detected blob regions (Figure 7) as input and tracking them over time, the algorithm determines a unique blob corresponding to each finger. That is, at most one candidate blob (one if finger and its finger line are visible in the image and none if the finger is obstructed or out of the field of view of the webcam) will have a finger line with more pixels than a predefined threshold and is called a finger blob. In the second phase, finger blobs are tracked over time. Neither creation of new blobs nor merging of existing blobs is allowed in this phase. If a finger blob disappears from the image or becomes too small, the algorithm returns to the first phase.

3.3 Second phase: Adapting the searched color interval

A major issue with color blob detection using a webcam is variations in color and light levels in the environment (due to, for example, light changes or color reflections). To deal this issue, the YCrCb color model was used, which isolates most of the luminance component into the Y value. However, the color model chosen cannot completely separate luminance from chromatic information, so the same finger blob in different positions will have slightly different CrCb values. To overcome this variable, we gradually adapt the color interval that we are tracking.

After having found a blob corresponding to a specific finger, we create its histogram. By adapting the color interval we can find the blob even if its overall color changes slightly between frames. To reduce histogram processing time we use quantization. For example, if we use 16 bits for the color components (8-bits Cr and 8-bits Cb) and quantize this to 8-bits (4-bits Cr and 4-bits Cb), only 256 of the 65536 possible values for the color components are utilized, as can be seen in the CrCb histogram of a blob below (Figure 8).
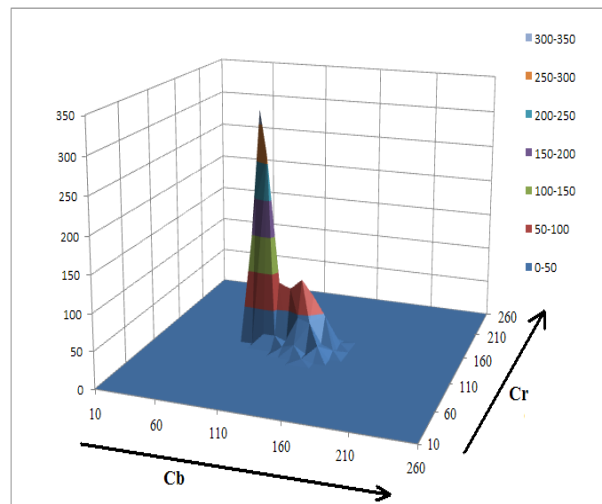

Figure 8: Quantified CrCb histogram for a certified finger blob.

4 PALM AND FINGER COORDINATE DETERMINATION

In the first step, the palm's coordinates and each finger in the webcam coordinates system are determined. Then the coordinates of each finger in the palm coordinate system are computed (see Figure 1). The complete finger position determination is shown in Figure 11 and a detailed presentation of the finger tracker is offered in Figure 10.

4.1 The position of the palm

To correctly measure hand-webcam distance with high precision a large symbol must be used. Hence, the algorithm employs colored markers placed at the center of a user hand's palm. By using a square we have a simple means to determine the position of the palm in 6 degrees of freedom (DOF). The algorithm requires one colored marker to be placed on each side of a hand to correctly measure hand rotation.
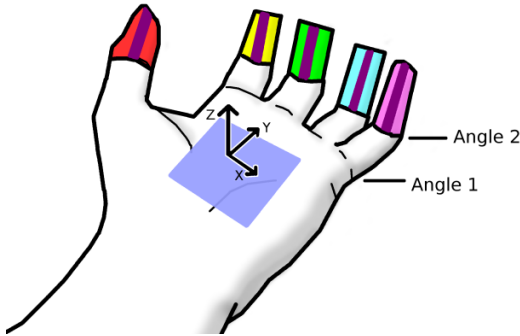

Figure 9: Angles 1 and 2 for phalanxes 1 and 2 in the palm coordinate system (2 DOF).

4.2 The position of the fingers

While each finger has 6 DOF, each finger-tip only has 2 DOF in the palm coordinate system. So by using relative finger positions with respect to the palm, we can deduce the position of each finger in 2 DOF. The 2 DOF are the angles of the first and the second phalanxes in relation to the palm, which are found by utilizing outside and inside finger lines (Figure 9). After locating each finger-tip's approximate position by using the least squares linear trend line approximation, we find the slope (Equation 1) for a set of points (x_i, y_i), i = [1 … N], sampled along the finger line.

$$slope = \frac{(\sum_{i=1}^{N} x_i) * (\sum_{i=1}^{N} y_i) - N \sum_{i=1}^{N}(x_i * y_i)}{(\sum_{i=1}^{N} x_i)^2 - N \sum_{i=1}^{N}(x_i{}^2)}$$

(Equation 1)

The subsequent computations are carried out in the webcam coordinate system. First, the angle of the finger line is computed by the arctangent of the slope (Equation 1). By relating this angle to the length of the finger line and the coordinate values of the finger line center of gravity, we determine the finger coordinate values. These values are then transformed into the palm coordinate system.
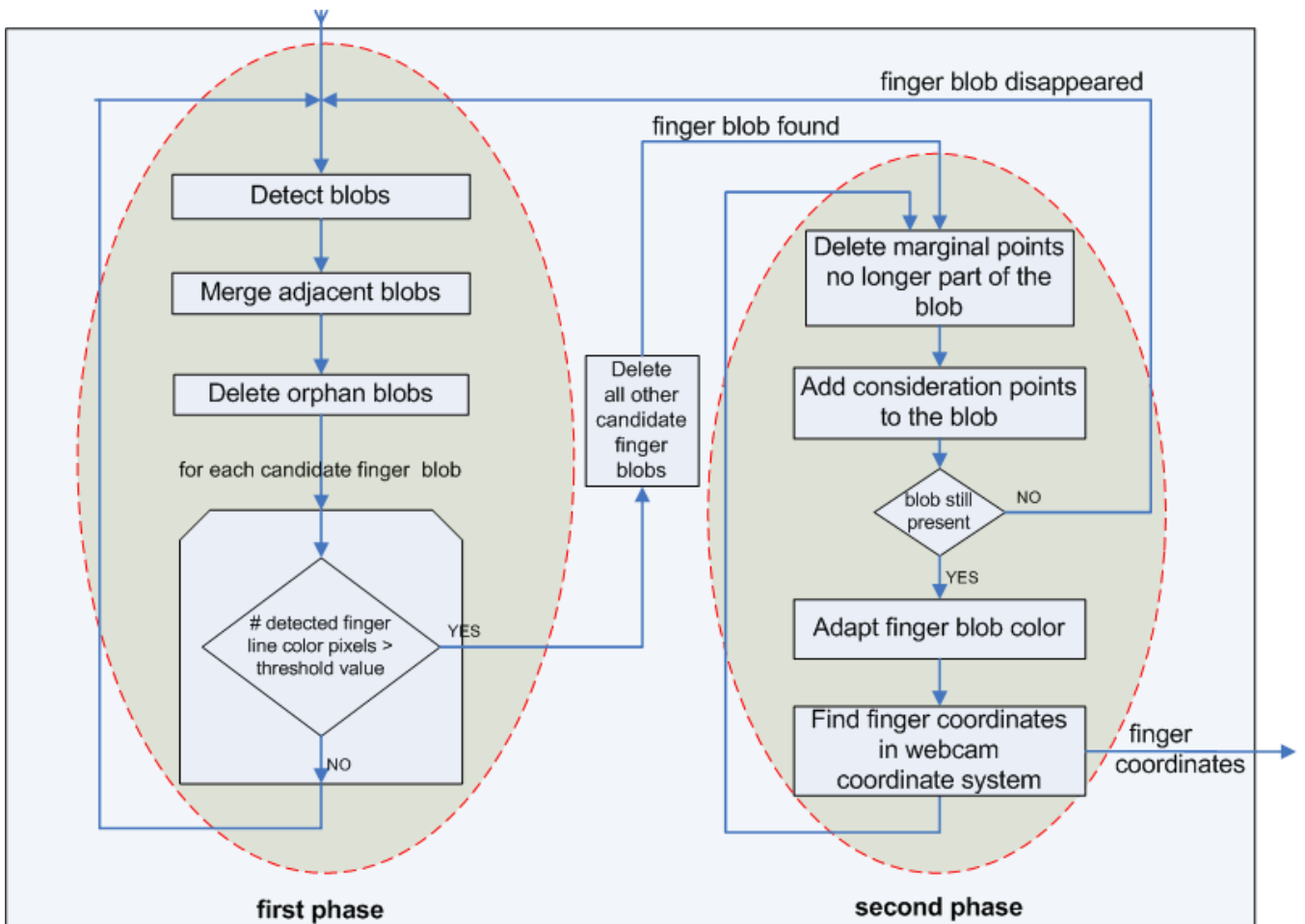

Figure 10: Finger Tracking including First and Second Phase as described above.
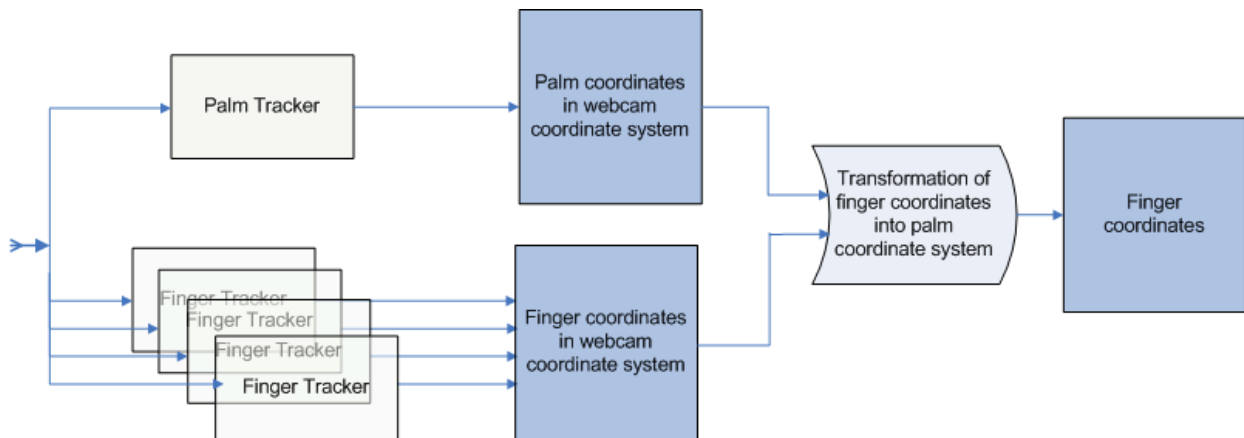
Figure 11: Complete coordinate detection for one hand's palm and fingers.

5 DISCUSSION AND OUTLOOK

We have presented an algorithm to detect one or both of a user's hands utilizing palm and finger coding. Fingers are color coded and the palm is pattern- or color-coded. Previous gesture recognition approaches have employed different means of hand pose classification in the webcam coordinate system. Our approach uses hand pose classification in the hand coordinate system which is rotationally independent in the webcam system. Hence, our approach requires less computation, no training, and so, supports hand pose configuration in a more straightforward way than previous approaches.

An inconvenience in using only one webcam is the existence of blind spots. That is, when the webcam line of sight is included in the palm plane, the palm code is not visible to the webcam. A p problem less related to the use of one webcam is the various occlusion effects, caused by clenched fingers or other body parts than the hand being tracked. Finally, we believe that the algorithm suggested opens possibilities for a range of potentially interesting applications such as free-form shaping, tabletop interaction, edutainment, and remote collaboration. To support such uses, we plan to offer an Application Programming Interface (API) which will include a webcam, a printer, environment color calibration, and hand pose configuration.

REFERENCES

[1] M. Bray, E. Koller-Meier and L. Van Gool, "Smart Particle Filtering for High-Dimensional Tracking", Computer Vision and Image Understanding, Springer LNCS, 2007, in press.

[2] M. Bray, E. Koller-Meier, N.N. Schraudolph and L. Van Gool, "Fast stochastic optimization for articulated structure tracking", Image and Vision Computing, pages 351-363, 2006.

[3] M. Bray, E. Koller-Meier, P. Mueller, L. Van Gool and N. N. Schraudolph, "3D Hand Tracking by Rapid Stochastic Gradient Descent Using a Skinning Model", 1st European Conference on Visual Media Production (CVMP), March 2004.

[4] H. Sidenbladh, M. J. Black, and D. J. Fleet. "Stochastic tracking of 3D human Figures using 2D image motion". In ECCV, pages 702–718, 2000.

[5] M. Isard and A. Blake. "Condensation – conditional density propagation for visual tracking", International Journal on Computer Vision, 29(1):5–28, 1998.

[6] B. Stenger, P. R. S. Mendonça, and R. Cipolla, "Model-Based 3D Tracking of an Articulated Hand", Proc. CVPR, Vol. II, pages 310-315, Kauai, USA, December 2001.

[7] B. Stenger, P. R. S. Mendonça, and R. Cipolla, "Model-Based Hand Tracking Using an Unscented Kalman Filter", Proc. BMVC, Vol. I, pages 63-72, Manchester, UK, September 2001.