Real-Time Two Hands Tracking System

Nianjun Liu and Brian C. Lovell

Intelligent Real-Time Imaging and Sensing (IRIS) Group
School of Information Technology and Electrical Engineering
The University of Queensland ,Brisbane,Australia 4072
Email: {nianjun, lovell}@itee.uq.edu.au

Abstract: *The paper introduces a novel system of two hands real-time tracking based on the unrestricted hand skin segmentation by multi color systems. After coror-based segmentation and pre-processing operation, a label set of regions is created to locate the two hands automatically. By the normalization, template matching is used to find out the left or right hand. An improved fast self-adaptive tracking algorithm is applied and Canny filter is used for hand detection.*

Key words: HSV-YUV colour transform, Normalization, Morphology, Blur, Template matching.

Introduction.

A vast number of potential applications in gesture interface for HCI have been designed in last decade. Not only hand gesture as a mode of HCI can enhance the interaction in "classical" desktop computer applications by replacing the computer mouse and keyboard as well as joysticks and control buttons, but also they can be used to help physically impaired people communicate more easily with others. Nevertheless, the major impetus for the development of gesture interfaces has come from the growth of applications in virtual environments.
Human gestures naturally use both hands. Yet, almost all of the vision-based gesture systems focus their attentions on single-hand gestures. The first reason is that the analysis technique requires that the hands be extracted from global images. If two-handed gestures are allowed, several ambiguous situations that do not occur in single-hand case may have to be dealt with, such as occlusion of hands, distinguishing between or the indexing of left/right hands. Secondly, the major drawback is the computation speed. Hence, in order to adequately exploit the interface of two-handed gestures in the future, more effective analysis techniques should be considered. These techniques should not only rely on the improvements of the classical techniques used in single-hand gesture, but also exploit the interdependence between the two hands performing a gesture.
This paper presents a system for two-handed tracking and edge detection. On the basis of this work, the future work of recognition and controlling will be outlined.

System overview

In the tracking system, the hand gestures are captured by a cheap web camera and a standard Intel Pentium-based personal computer. Skin color segmentation based on multi-color systems has been applied. Pre-processing (Morphological) operations are used to smooth the image and remove the noise. Normalized template match is used for searching the location of left or right hand. An improved Camshift algorithm has achieved a good performance in tracking the hand. Canny edge detection can extract robust and efficient hand contours.

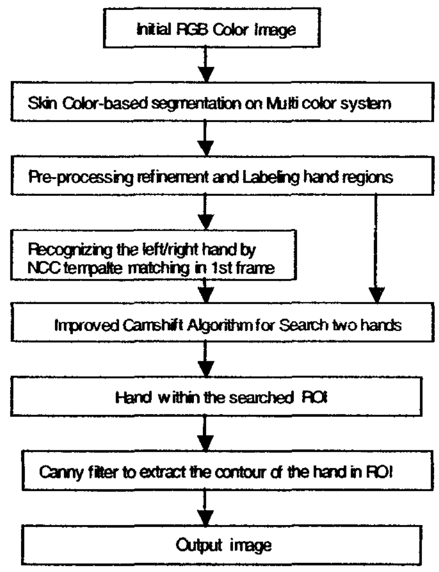

Initial RGB Color Image
Skin Color-based segmentation on Multi color system
Pre-processing refinement and Labeling hand regions
Recognizing the left/right hand by NCC tempalte matching in 1st frame
Improved Camshift Algorithm for Search two hands
Hand within the searched ROI
Canny filter to extract the contour of the hand in ROI
Output image

Figure 1 Flow chart of the real-time hand tracking system

The diagram Figure 1 depicts the system flow chart. When initial Color images are input, the first step is to convert RGB to both the HSV and the YUV color systems. Because the H and UV values of human skin color are in the invariant ranges, the probability distribution image is discerned using a predefined lookup table for skin color hue on the base of Histogram. This step is followed by Pre-processing refinement, in which morphological operations are used to remove the noise, with the hand regions of interest being sorted and labeled to locate the region of the two hands. An improved Continuously Adaptive Mean-shift algorithm (Camshift) algorithm [2] is applied to determine a region of interest (ROI) surrounding the hand in each successive frame in order to track the hand. Finally, canny edge detection on the grayscale ROI determined in the previous step) is used to extract the contour of the hand, and this edge map is combined with the skin-color probability distribution image to determine a reliable contour.

Color-based segmentation

An improved color-based segmentation has proved to be an available method for segmenting the hand in an unrestricted environment. We apply two color systems, one being HSV and the other being YUV. Both HSV and YUV colour system have their own advantages and disadvantages for hand segmentation. After the experiment, if applying them separately, it can be seen that either of them will include noise and lose the real skin dots. Therefore, the two systems are combined together, which achieves better performance than could be achieved by applying either of them individually after many experiments.

Colour segmentation and skin detection is the first module confronted by any input image. The colour content of each pixel is analyzed and determined to be either skin or non-skin. It has been demonstrated, regardless of the differences in skin color of different people and different races, that the chrominance of skin is relatively consistent. The major difference between skin tones is intensity, as dark-skinned people have more skin saturation than light-skinned people. YUV and HSV color systems have been demonstrated to be good methods for separating the chrominance and intensity. It was found in YUV or HSV color representation, that human skin colors remain distributed over a very small region on the chrominance plane.

While the input images are generally input as Red, Green and Blue (RGB) values, it is also possible to represent colour using other colour coordinate systems such as HSV or YUV. The HSV system is a polar coordinate system with H denoting hue values, S denoting the saturation and V denoting the intensity values. YUV is similar to HSV with Y denoting the intensity, while the UV components (also known as the chrominance components) specify the hue and saturation in Cartesian coordinates. The equation and function converting RGB to YUV coordinates is [3] is:

RGB To YUV(YC_rC_b)	RGB To HSV
$Y=0.3R+0.6G+0.1B$ $U=B-Y$; $C_b=0.5(V-1)$ $V=R-Y$; $C_r=(V/1.6)+0.5$	Matlab Function Rgb2hsv();

We apply both HSV and YUV color space and create each model of the desired chrominance value using a color histogram. The skin segmentation is made independant of illumination changes and different skin tones using an illumination normalized color space and a skin color probability distribution that is generated from a range of different ethnic groups. In the system, the raw RGB image is converted to HSV or YUV color space first, and then the known skin color model is used to convert H-space(or UV) image to a color probability distribution image. In this way, the skin color regions have been separated from the background. A lookup table is employed to classify each pixel, where each intensity is tested to see if it lies in the range of skin colour and be associated a binary value of one if it is and zero if not. This is done after the colour image has been mapped into a binary image of ones and zeros representing skin and non-skin regions.

Preprocessing and label hand location

It is common during the colour segmentation to return values that are close to skin but are non-skin, or other skin-like coloured regions that is not part of the body. These noisy erroneous values are generally isolated pixels or group of pixels that are dramatically smaller than the total hand regions with a big connected region in the binary image. Pre-processing refinements are applied to the binary output in order to reduce some of the effects of these noisy pixels. We apply the morphological operation, using opening first (eroding first and then dilating) and closing (dilating first and then eroding) in order to remove the spurious noises which are generally much smaller than the hand region itself and not change the shape of main object(such as hand and face). After it, the imfill function is used to fill the holes inside the hand. The next step is to automatically locate the hands' region and remove all other "similar skin" parts. We use label function to label each part inside the binary image and sort them by area. We only select the 4 or 5 top ones and remove all other parts. Because the hands are always the big connected regions, the left 4 or 5 regions should includes one or two hands and/or face,etc. Finally, template matching is used to locate the left or right hand. Because the boundary of region of interests(hand or face) is very curved, we will use the imfilter function to smooth the boundary. After normalisation, we can use template matching to recognise the hand or face.

The following shows the segmentation images. Figure 2-1 is the original image, Figure 2-2 is the searched hand region in the whole image after pre-processing refinement.

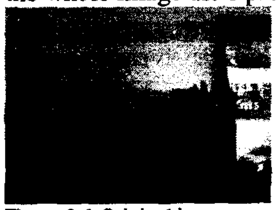


Figure 2-1 Original image

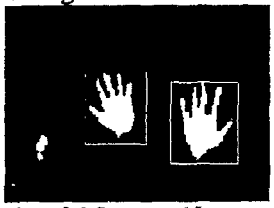


Figure2-2 Segmented Image

Recognizing left and right hands by NCC

In order to distinguish between the left and right hands, we use Eigenvalues and Normalized Cross Correlation(NCC) to define the hands' templates. The first step is to rotate and scale the searched region to achieve normalized status, and then we apply pre-matching method and NCC to compare the regions with the templates to define the left hand, right hand or face.

Normalisation is the essential preparation before doing template matching. The success of the NCC is heavily dependent on the accuracy of the normalization. As in the case of the output binary image from segmentation, normalization includes the rotation and scaling.

Rotation Normalisation: In a video sequence, the hands in the scenes will typically be subjected to many different rotational orientations. Due to the three dimensional rotary motion of hands, forward tilting, planar rotations (sideway tilting of hand) and turning rotations (twisting of hands) are all major problems encountered by hand normalization. However, due to the complexity, only planar rotation–sideways tilting of the hand – has been investigated in this system.

Rotational analysis is based on the binary mapping produced from the skin detection stage. The aim is to compute the angle of rotation of the hand block in the binary image. This is achieved by wrapping an ellipse around the hand region so that the second-order moment of the ellipse is the same as the second-order moment of the binary mapping. The angle formed between the major axis of the calculated ellipse and the horizontal x-axis will consequently represent the hand's angle of tilt. Rotation normalisation can be achieved by rotating the image by the negative of that angle. The following Equations illustrates how the rotational angle is computed.

The 2D orientation of the binary mapping image is also easy to obtain by using the second moments, where(x,y) range over the search window, and I(x,y) is the pixel(probability) value at (x,y). The second moments are:

$$M_{20} = \sum_x \sum_y x^2 I(x,y); \qquad M_{02} = \sum_x \sum_y y^2 I(x,y)$$

Then the object orientation (major axis) is

$$\theta = \frac{\arctan\left(\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)} \right)}{2}$$

The first two eigenvalues (major length and width) of the hand block may be expressed in the following closed form:

$$a = \frac{M_{20}}{M_{00}} - x_c^2, \; b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right), \; c = \frac{M_{02}}{M_{00}} - y_c^2$$

Then the length l and width w from the centroid are

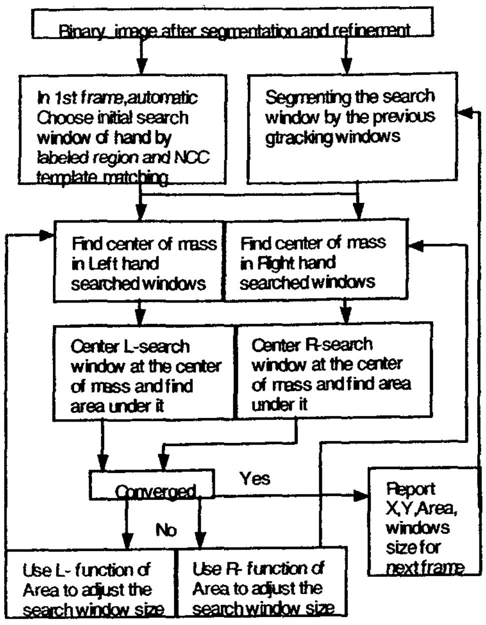
$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}}; \; w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}}$$

Scaling Normalisation:
This method involves making full use of the skinned region information obtained from the colour segmentation stage. Based on the binary mapping of the hand region in the image, the size of the extracted box is compared to the dimensions of the template. Two scaling factors are then determined, one being a ratio of width and the other, a ratio of height. Selection is achieved by choosing the larger ratio between the two ratios when scaling down, and selecting the smaller ratio when scaling up. This process of using the same ratio for both dimensions is chosen in order not to morph the hand to a different set of dimensions so that the relative variances between the pixels is not altered. Once the ratio has been determined, normalisation can be achieved by adjusting the dimensions of the extracted box so that the hand in the image is now in the same scale as the templates. Under dynamic mode, where real-time recognition is performed on video input, hands of all scales will be present. The task of scaling normalisation involves ensuring that the scaling of all the hands in each input frame captured conforms to a standardised size determined by the samples. This is accomplished by computing the scaling correction factor from the bounded region of the binary image produced from the skin detection stage.
Template matching to find left and right hands:
Before using template matching, a pre-matching method is applied by comparing the area of the normalized region with one of the templates. If the area's difference is not in the defined range, which means this region is certainly not the hand object, the next region will be calculated, otherwise, the region will be matched by the template. If all of the searched 5 regions from the pre-processing step are not qualified, we will discard this frame, use the better one in next frames, because there are dozens of frames during the initial time(1 or 2 seconds).
Normalized Cross Correlation (NCC) involves finding the best match between a template and a sequence of windows. The NCC search requires a template; therefore, a typical hand or average hand needs to be available in order for NCC to work. The basic principle behind NCC is to compare two windows of the same size and measure its correlation or how closely related each pixel from one window is from its corresponding pixel in another window. A maximised correlation value therefore means that the two windows under examination, has each of their corresponding pixels matching the closest from all the other windows under testing. Hence, the best candidate for the template is the average or typical hand. It is a standard template representing the most basic features of hands and contains as little biased additions as can be found of any hand image. With the template selected, the process of hand detection with NCC remains merely a matter of applying the equation of normalised cross correlation. The normalized skinned region of the image will become the test window; the modified average hand on the other hand will form the template window. The test window, of the same dimensions as the template, will then be continuously extracted and correlated with the template. The extracted region with the highest correlation value corresponds the closest to the hand template.

Improved continuous adaptive meanshift algorithm.

Improved Camshift adaptive algorithms to track the real-time hand region. The Continuously Adaptive Mean-shift algorithm, or CamShift algorithm is a generalization of the Mean-shift algorithm which is designed for static distributions. CamShift operates on a 2D color probability distribution image produced from histogram back-projection[2]. It is designed for dynamically changing distributions. These occur when objects in video sequences are being tracked and the object moves so that the size and location of the probability distribution changes in time. The Camshift algorithm adjusts the search window size in the course of its operation. For each video frame, the color probability distribution image is tracked and the center and size of the color object as found via the Camshift algorithm. The current size and location of the tracked object are reported and used to set the size and location of the search window in the next video image. The process is then repeated for continuous tracking. Instead of a fixed, or externally adapted window size, Camshift relies on the zeroth moment information, extracted as part of the internal workings of the algorithm, to continuously adapt its windows size within or over each video frame. Figure 3 shows CamShift Algorithm Process:

Step 1: Convert image from RGB to HSV color system. In the first frame of video, we segment the whole image, use the method introduced in previous segmentation section and then we will automatically find the rectangle hand region.
Step 2: Run the mean shift algorithm to calculate the


Binary image after segmentation and refinement
In 1st frame,automatic Choose initial search window of hand by labeled region and NCC template matching
Segmenting the search window by the previous gtracking windows
Find center of mass in Left hand searched windows
Find center of mass in Right hand searched windows
Center L-search window at the center of mass and find area under it
Center R-search window at the center of mass and find area under it
Converged
Yes
No
Report X,Y,Area, windows size for next frame
Use L- function of Area to adjust the search window size
Use R- function of Area to adjust the search window size
Figure 3: Camshift Algorithm of finding the hand region

centroid of mass with the search window and center search window at the centroid of the mass and find the area under it. If the centroid and the center of search window is not converged, we use a function of area to adjust the search window size and repeat the loop. If converged, we store the windows size and location for next frame. The centroid within the search window is found by the zeroth and first moments.

The zeroth and 1st moment for x and y:

$$M_{00} = \sum_x \sum_y I(x,y) \qquad M_{10} = \sum_x \sum_y xI(x,y); \qquad M_{01} = \sum_x \sum_y yI(x,y)$$

The centroid then is found by:

$$x_c = \frac{M_{10}}{M_{00}}; \qquad y_c = \frac{M_{01}}{M_{00}}$$

Where I(x,y) is the pixel value in the position(x,y) in the image, and x and y range over the search window.

Step 4: For the next video frame, center the search window at the stored location stored in step 3, and repeat the loop from step 2.

The demo of hand tracking is shown below.

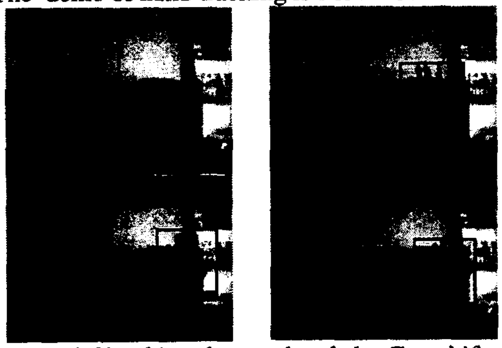
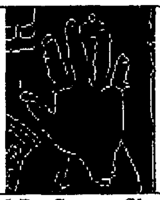


Figure 4: Tracking the two hands by Camshift

Real-time hand edge extraction

Canny edge detection is a very good method for detecting edges, and was suggested by J.Canny [7]. It takes a grayscale image on input and returns a bi-level image where non-zero pixels mark dectected edges. In the system, I apply Canny filtering to detect contours in the region of interest (rectangle region). Figure 5 is the output of only applying Canny filter in the whole image and the rectangle region of interest.

The following summarize the steps. The original image data in ROI is first converted to greyscale and then smoothed by a Gaussian function of width specified by the parameter. The smoothed image is differentiated with respect to the directions x and y. We join the smoothing and differentiation in a Sobel operator. After the gradient has been calculated at each point of the image, the edges can be located at the points of local maximum gradient magnitude. The Canny operator sets an upper and lower edge value limit to threshold edges and it drastically reduces the likelihood of streaking. The ratio of the high limit to the low limit is in the range of three to one based on predicted signal-to-noise ratios.

	
Figure 5-A Canny filter on the whole image	Figure 5-B Canny filter on the region of interest

Final Output Image and Video

The system is a combination of all of the above approaches, which is better than applying each of them individually. CamShift Tracking is used to define the searching region. The left/right hand region is found out using color-based segmentation, pre-procession and NCC template matching. Canny edge detectors are used for extracting the hand contours. Finally, the noises inside and outside the hand contour are removed by comparing the edge image with the segmenting binary one. The method is high speed, computationally efficient, with a tracking rate is 20 fps on a standard PC. An example of the video output can be found on our group web site. The images below are samples of the output videos. Figure 6 is the output image of two hands. The black box is the Camshift tracking region. Figure 7 is the extracted hand contour after removing the noise of edges outside and inside the hands.

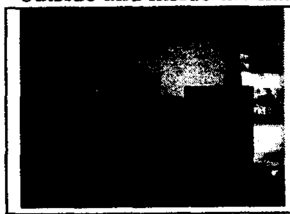
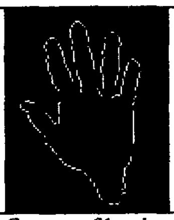
	

Figure 6 :Two hands' output	Figure 7:Contour of hand

References

[1] D. Heckenberg and B. C. Lovell, "MIME: A Gesture-Driven Computer Interface", Proceedings of Visual Communications and Image Processing, SPIE, V4067, pp 261-268, Perth, 2000.

[2] Intel Open Source Computer Vision Library Reference Manual. www.intel.com/research/mrl/research/opencv