Hand Recognition in Live-Streaming Video

Mikhail Belov
Department of business-informatics
Higher School of Economics
Moscow, Russian Federation
mpbelov@gmail.com

Abstract— **The article describes the algorithmic component of the pattern recognition method for extracting hand patterns from a video stream. Methods removing excess information from frames, localizing fragments with a hand and extracting hand contours to classify them are described.**

Keywords-pattern recognition; Hu invariants; Canny detector; video stream processing

I. INTRODUCTION

One can input data into a computer in a form of graphical information. There are methods for processing the graphical information and for treating it not only as a set of dots with color codes, but also as a container for another data. This fact gives an opportunity to extend the number of human computer interaction (HCI) ways. Such systems are described in [5] and [6].

It is planned to develop a system prototype for direct and online controlling the graphical objects displayed on a screen. Currently this idea has been implemented in two types of systems: sensor screens and "smart boards". In the first case, a transparent sensor pad is placed over the screen, which catches the user's touches and translates them into control signals to the processor. Due to an existing technology such screens are expensive and produced mostly in small and medium formats. In case of the "smart boards" the projector's light is not focused on a usual board, but aimed to a special sensor surface. Unlike sensor screens, there are large "smart boards" because of projector, but this approach remains rather expensive and not suitable as a mass solution. Instead, building such a HCI system based on a video camera and a projector will reduce the dependency of the cost from the display size.

The article describes the algorithmic component of the pattern recognition method for extracting hand patterns from a video stream.

II. PREPARING A FRAME

To remove excess information from a frame one can use the Histogram Backprojection method. In this case a hand is being searched by its color characteristics. The method can be applied to search for pixels satisfying the histogram, or to search a pattern (of a hand image) by shifting the pattern w.r.t. the initial image. A frame fragment containing the hand image should be used as a template histogram.

In Histogram Backprojection the model (target) and the image are represented by their multidimensional color histograms M and I as in Histogram Intersection. A ratio histogram R, defined as $R_i = min\left(\frac{M_i}{I_i}, 1\right)$, is computed from the model and image histograms. It is this histogram R that is backprojected onto the image, that is, the image values are replaced by the values of R that they index. The backprojected image is then convolved by a mask, which for compact objects of unknown orientation could be a circle with the same area as the expected area subtended by the object. The peak in the convolved image is the expected location of the target, provided the target appears in the image [1].

III. LOCALIZING A FRAME FRAGMENT WITH A HAND

We can locate a fragment with hand by calculating the difference image characteristics [4]. We have to use an image template, which contains a hand picture.

One of possible methods is a square difference matching method. Perfect match leads to 0 result. Large result means bad match:

$$R_{sq_{diff}}(x,y) = \sum_{x',y'} [T(x',y') - I(x+x',y+y')]^2$$

Correlation matching methods multiplicatively match the template against the image so a perfect match will be large and bad matches will be small or 0.

$$R_{ccorr}(x,y) = \sum_{x',y'} [T(x',y') \cdot I(x+x',y+y')]^2$$

Correlation coefficient matching methods match a template relative to its mean against the image relative to its mean, so a perfect match will be 1 and a perfect mismatch will be –1; a value of 0 simply means that there is no correlation (random alignments).

$$R_{ccoeff}(x,y) = \sum_{x',y'} [T'(x',y') \cdot I'(x+x',y+y')]^2$$

$$T'(x',y') = T(x',y') - \frac{1}{(w \cdot h)\sum_{x'',y''} T(x'',y'')}$$

$$I(x+x,y+y) =$$

$$= I(x+x,y+y) - \frac{1}{(w \cdot)\sum_{x,y} I(x+x,y+y)}$$

These factors may be normalized [4]. The normalized methods are useful because they can help reduce the effects of lighting differences between the template and the image. In each case, the normalization coefficient is the same:

$$Z(x,y) = \sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}$$

IV. Extracting Hand Contours

Then we extract contours from the image using the Canny detector [2]. The Canny algorithm runs in 5 separate steps.

A. *Smoothing: Blurring of the image to remove noise*

It is inevitable that all images taken from a camera will contain some amount of noise. To prevent that noise is mistaken for edges, noise must be reduced. Therefore the image is first smoothed by applying a Gaussian filter.

B. *Finding gradients: The edges should be marked where the gradients of the image has large magnitudes*

Gradients at each pixel in the smoothed image are determined by applying what is known as the Sobel-operator.

C. *Non-maximum suppression: Only local maxima should be marked as edges*

The purpose of this step is to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges. Basically this is done by preserving all local maxima in the gradient image, and deleting everything else. The algorithm is for each pixel in the gradient image:

1) Round the gradient direction θ to nearest 45 $^\circ$, corresponding to the use of an 8-connected neighbourhood;

2) Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient direction. I.e. if the gradient direction is north (theta = 90 $^\circ$), compare with the pixels to the north and south;

3) If the edge strength of the current pixel is largest; preserve the value of the edge strength. If not, suppress (i.e. remove) the value.

D. *Double thresholding: Potential edges are determined by thresholding*

Edge pixels stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.

E. *Edge tracking by hysteresis: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge*

Strong edges are interpreted as "certain edges", and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. The logic is of course that noise and other small variations are unlikely to result in a strong edge (with proper adjustment of the threshold levels). Thus strong edges will (almost) only be due to true edges in the original image. The weak edges can either be due to true edges or noise/color variations.

V. Classifying the Found Contour

We classify found contours after the extraction. Hu invariant moments of contours are used for this. Moment is a gross characteristic of the contour computed by integrating over all of the pixels of the contour [3]. The (p, q) moment of a contour is defined as:

$$m_{p.q} = \sum_{i=1}^{n} I(x,y) x^p y^p$$

Here p is the x-order and q is the y-order, whereby order means the power to which the corresponding component is taken in the sum just displayed. The summation is over all of the pixels of the contour boundary (denoted by n in the equation).

The moment computation just described gives some rudimentary characteristics of a contour that can be used to compare two contours. However, the moments resulting from that computation are not the best parameters for such comparisons in most practical cases. In particular, one would oft en like to use normalized moments (so that objects of the same shape but dissimilar sizes give similar values). Similarly, the simple moments of the previous section depend on the coordinate system chosen, which means that objects are not matched correctly if they are rotated.

A central moment is basically the same as the moments just described except that the values of x and y used in the formulas are displaced by the mean values:

$$\mu_{p,q} = \sum_{I=0}^{n} I(x,y)\left(x_{avg}\right)^p \left(y - y_{avg}\right)^q$$

where $x_{avg} = \frac{m_{10}}{m_{00}}$ and $y_{avg} = \frac{m_{01}}{m_{00}}$.

The normalized moments are the same as the central moments except that they are all divided by an appropriate power of m_{00}:

$$\eta_{p,q} = \frac{\mu_{p,q}}{{m_{00}}^{(p+q)/2+1}}$$

The Hu invariant moments are linear combinations of the central moments.

The following factors are used to detect similarity between two contours [4]:

$$I_1(A,B) = \sum_{i=1}^{7} \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right|$$

$$I_2(A,B) = \sum_{i=1}^{7} \left| m_i^A - m_i^B \right|$$

$$I_3(A,B) = \sum_{i=1}^{7} \left| \frac{m_i^A - m_i^B}{m_i^A} \right|$$

Here m_i^A and m_i^B are defined as:

$$m_i^A = sign(h_i^A) \cdot log\left|h_i^A\right|,$$

$$m_i^B = sign(h_i^B) \cdot log\ |h_i^B|,$$

where h_i^A and h_i^B are the Hu moments of A and B, respectively.

After classification system performs the action associated with a certain gesture.

VI. Future Work

The described image processing methods may be used with various parameters. It is planned to implement the investigated algorithm and to choose the best methods among the available alternatives at the next step of the research. It is also needed to assess the capabilities of usage Hu moments for hand contour comparison and similarity detection. As further research of the described method it is required to compare the contour similarity coefficient metrics listed above based on Hu moments by quality of classification result.

References

[1] M. Swain, D. Ballard, "Color Indexing". International Journal of Computer Vision, 7:1, Kluwer Academic Publishers, Manufactured in The Netherlands, 1991, pp. 11-32.

[2] J. Canny. "A computational approach to edge detection. Pattern Analysis and Machine Intelligence", IEEE Transactions on, PAMI-8(6), Nov. 1986, pp. 679–698.

[3] M. Hu, "Visual pattern recognition by moment invariants" IRE Transactions on Information Theory 8, 1962, pp. 179‑187

[4] G. Bradski, A. Kaehler. "Learning OpenCV". O'Reilly Media. 2008. Pages: 576.

[5] P. Garg, N. Aggarwal, S. Sofat. "Vision Based Hand Gesture Recognition", World Academy of Science, Engineering and Technology – 49, 2009.

[6] C. Keskin, A. Erkan, L. Akarun, "Real Time Hand Tracking and 3D Gesture Recognition for Interactive Interfaces Using Hmm", ICANN/ICONIPP, 2003.