FACIAL FEATURES DETECTION USING TEXTURE HOUGH TRANSFORM

V. S. Gorbatsevich

The Federal State Unitary Enterprise «State Research Institute of Aviation Systems» Russia, Moscow - gvs@gosniias.ru

Commission V, WG V/5

KEY WORDS: Local Binary Pattern, Object detection, facial features detection, Hough transform

ABSTRACT:

The paper presents an original method for object detection. The “texture” Hough transform is used as the main tool in the search. Unlike classical generalized Hough transform, this variation uses texture LBP descriptor as a primitive for voting. The voting weight of each primitive is assumed by learning at a training set. This paper gives an overview of an original method for weights learning, and a number of ways to get the maximum searching algorithm speed on practice.

1. INTRODUCTION

One of the most popular issue in technical vision is the issue of identification of certain points on the image of a face (mouth, nose, eyes, eyebrows, etc). These special points which align with different part of the face are used, more or less, in almost every application developed in the digital face image processing sphere. For example, there is an abundance of methods for comparing images of faces, pose evaluation algorithms, algorithms for composing 3D-models of faces and many others. This being said, besides the quality, the computation speed is very important for the methods of finding the special points. Currently, there are a lot of ways to find the points, from already classical Voila-Jones(Viola 2001) method, to Hough Forest(Gall 2010). All these methods give us proper means to deal with the identification issue; however the problem of quality/speed ratio is still relevant.

This work makes an attempt to look into a matter of objects identification from the perspective of the concept of generalized Hough transform and voting weights training. It also suggests modification of Hough transform, as well as follow-up methods of evidence analysis.

2. TEXTURE HOUGH TRANSFORM

2.1 Hough transform

Nowadays Hough transform is one of the most popular methods of analytically defined objects identification. Its popularity arises from the excellent ratio of quality and speed. The central idea of Hough transform is voting of some primitives for hypotheses about the position of the sought object. The hypotheses that have gathered the votes above a certain threshold are considered true. To illustrate the central idea of Hough transform, let’s take a look at the simplest equation for identification of a black line on a binary image. As is known, the equation of line takes the form of:

$$y = k * x + b$$

where K – slope of the line
X, y – coordinates
B - is the y-intercept of the line

While in actual practice the following parameterization is more commonly used:

$$r = x \cos\theta + y \sin\theta$$

where R - radius vector
Θ – angle between the line and the horizontal axis

In this case every hypothesis appears as a pair of parameter values (R, Θ). A range of all these hypotheses are combined into a so-called Hough space. Therefore every point in the Hough space corresponds to the certain line. Points different from the background (in this case – black) are used as primitives. For every fixed point at the coordinates (x_p, y_p), there are infinite straight lines coming through it. In Hough space these lines are assigned via the sinewave, derived from:

$$r(\theta) = \boldsymbol{x_P} \cos\theta + \boldsymbol{y_P} \sin\theta \qquad (1)$$

So, set of points of some line (r, Θ) corresponds to the set sinewaves of the form of (1), which intercross at the point [r, Θ]. Thus, the line identification reduces to the voting of every point for all hypotheses in the line of equation (1). In actual practice a special accumulator-array is used for hypotheses storage, and the process of voting is reduced to summing.

Figure 1 shows an example of a line and a corresponding accumulator.

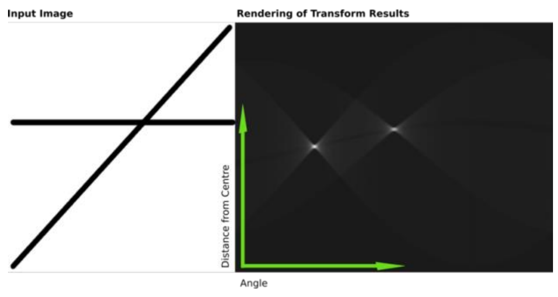

Figure 1. An example of Hough transform for lines identification. Initial image (on the left) and accumulator (or the right).

Hough transform is used the same way for identification of any curves defined by analytical equations.

2.2 Generalized Hough Transform

Generalized Hough Transform (GHT) is used for detection binary objects of an arbitrary shape. Unlike the classical variation of the transform, the generalized Hough transform is used in situations when the desired objects cannot be defined by any analytical equations. However, if the center is set manually, a convex curve can be defined as a look-up table. Therefore, it is possible to make a process of voting of edge points for possible center for simple distortion models simple. In this case a Hough space is a space of possible positions of center, and an accumulator-array is an array aligned with the size of an image. Figure 2 shows an example of a voting process for identification of an object of an arbitrary shape.

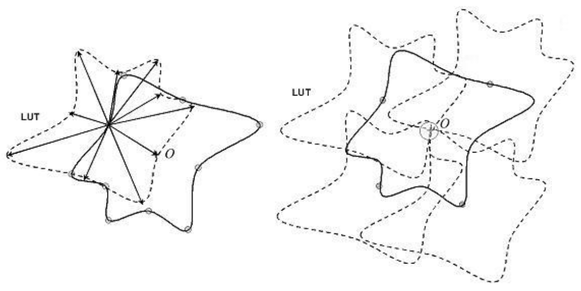

Figure 2. An example of Hough transform for lines identification. Initial image (on the left) and accumulator (or the right).

This example of a process of identification of some object in the image does not account for possible distortions although it does demonstrate the basic ideas of generalized Hough transform. Currently there are a lot of GHT modifications, which are capable to work with different distortions.

2.3 Texture Hough Transform

And now we are back to facial features identification. Unlike the previous cases, our current example is greyscale, namely, a greyscale image of an eye. A figure 3 shows a greyscale image of an eye.

Figure 3. A greyscale image of an eye.

Unfortunately, it is impossible to apply Hough transform directly because:

Firstly, in this case we work not with just an object, but with the whole class of objects, the appearance of which can vary depending on race, gender, emotional state, and other conditions;

Secondly, preliminary appliance of highly-accurate segmentation and edge detection algorithms is needed;

Thirdly, there is no “object” – “background” image mapping. However, the ideas of generalized Hough transform can still be applied. Let’s take a look at some texture descriptor. Let this descriptor be able to assume a limited set of values. Then we can take some specific value of this descriptor as a primitive. Then, just like with GHT, the Hough space consists of many possible object centers. Thus, the accumulator is a rectangular array, the size of which corresponds to the size of the image.

Let the size of an object be fixed at the [W*H], so that for some primitive with coordinates [X, Y] the area of possible center positions of the object takes a rectangular shape with the size of [W*H] and the center at the [X, Y], as shown in the figure 4.

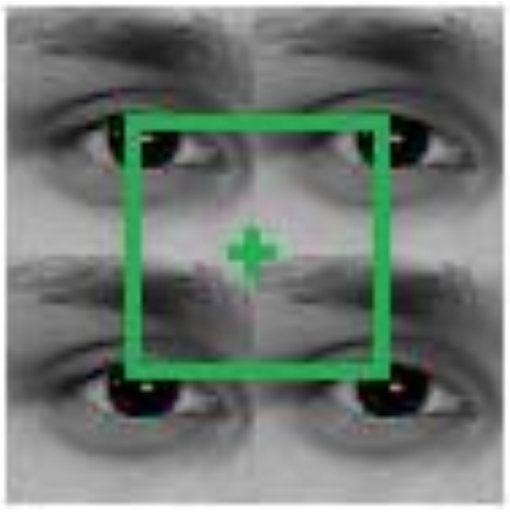

Figure 4. An area possible object center positions (green rectangular) for some primitive position (green cross).

Therefore, just a small voting weights map is required for every descriptor value, describing possible center positions of the desired object relative to the descriptor position.

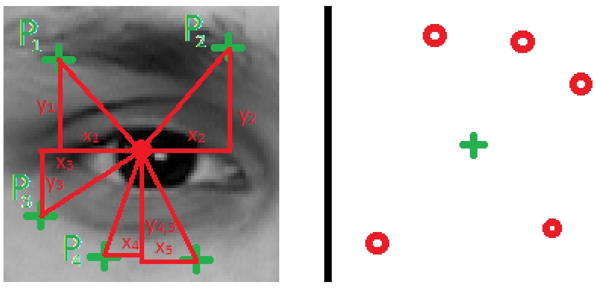

Figure 5. An example of weights array for some descriptor value. Descriptor with some fixed value (on the left). Weight map with possible center positions (on the right).

The voting algorithm looks like this:

```
For every point in the image (x,y):

    for dx = -W/2 to W/2
        for dy = -H/2 to H/2                                  (2)
            $Akk_{x+dx,y+dy} = Akk_{x+dx,y+dy} + Weigth_{Diskr_{x,y},dx,dy}$
```

where $Akk_{x,y}$ – accumulator array
$Weight_{VAL,dx,dy}$ – weight mapping for VAL descriptor value
$Diskr_{x,y}$ – value of descriptor in a point [x,y]

Detected objects are defined by:

$$\text{Centers} = \{(x,y): Akk_{x,y} > Thr\}$$

where Thr – threshold

Therefore, the accumulator filling occurs just only with summing operations. This will be discussed in the following section in more detail.

Albeit being really similar to the classical GHT, the described method of texture Hough transform has a number of important differences. All points in the image are used in the voting, and not just those on the edges. On one hand it makes the identification of objects with any texture and shape possible, but in the other this method causes the problem of assigning voting weights and, consequently, forming the corresponding weight maps.

3. LBP DESCRIPTOR

This work uses the LBP descriptor as a texture descriptor. This descriptor is one of the most well-known texture descriptors. Currently, there is an abundance of texture descriptors, like SIFT(Lowe 2001), SURF(Bay 2006), G-SURF(Pablo 2013), and many others. And yet the LBP has a number of attributes which make it more preferable for texture Hough transform:

1. Resistance towards additive and multiplicative brightness distortion
2. Very high speed of composition
3. There are only 256 descriptor value units
4. Every LBP descriptor matches with only one pixel

The process of getting a LBP descriptor can be briefly described as follows: a circular neighborhood of a certain radius is considered for every point in the image, and then every point is compared to a value of a current point of the image:

$$LBP(p) = \sum_{i=0}^{n} 2^i \begin{matrix} 1: I(p) \geq I(o_i) \\ 0: I(p) < I(o_i) \end{matrix}$$

where N – number of neighborhood points
O_i – neighborhood points
P – current point of the image

The result of comparison is recorded as binary vector (see Figure 4).

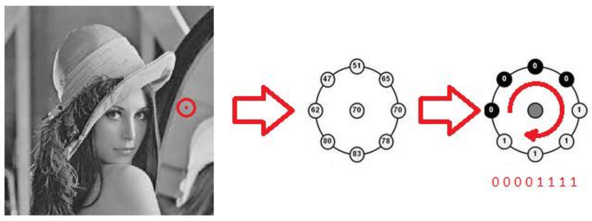

Figure 4. An example of LBP descriptor calculation for a point in the image

4. WEIGHTS MAPS LEARNING

The texture Hough transform described in the section 2.3 implies that there are preset weights maps for every value of a texture descriptor. It is the weights values that assign search object model and background model, and that's why the task of weights assigning is really important in using texture Hough transform. This work suggests using original learning procedure for weights value calculation. In order to do that we would need to rewrite the earlier mentioned texture transform as Viola-Jones style strong binary classifier:

$$\boldsymbol{C(x,y) = \begin{cases} object: val(x,y) \geq Thr \\ background: val(x,y) < Thr \end{cases}}$$

$$\boldsymbol{val(x,y) = \sum_{dx=-W/2}^{+W/2} \sum_{dy=-H/2}^{+H/2} \sum_{LBPVAL=0}^{255} Weigth_{LBPVAL,dx,dy}\, S(x+dx, y+dy, LBPval)} \qquad \textbf{(3)}$$

Where

$$\boldsymbol{S(x,y,LBPval) = \begin{cases} 1: Diskr_{x,y} = LBPval \\ 0: Diskr_{x,y} \neq LBPval \end{cases}}$$

W,H – object sizes

$Weight_{LBPVAL,dx,dy}$ – weight map for descriptor value = *LBPval*
$Diskr_{x,y}$ – LBP_{val} value weight map
Thr – some threshold

It is easy to verify that using such classifier in a sliding window is identical to the voting process from 2.3. Therefore, the assigning/setting the voting weights mirrors the training of such classifier. The usual method for training such classifiers is boosting (Rojas 2009). However, in this case the situation is slightly different from usual process. This not something for Haar-like characteristics. The main difference is that there is just a small number of classifiers that can be used for every position, namely, only W*H classifiers. This gives us an opportunity to use new unexpected algorithm for weights calculation.

Imagine a small subset Base. It is possible to use quality quadratic functional for evaluation of classifier's quality:

$$Q(\mathbf{Base}\,, \boldsymbol{Weigths}) = \frac{1}{2} \sum_{a \epsilon \mathbf{Base}} (val(a, \boldsymbol{Weigths}) - Ans(a))^2$$

where val(a) – classifier's response for example a (see 2)
Ans(a) – trainer's response:

$$\boldsymbol{Ans}(\boldsymbol{a}) = \begin{cases} \mathbf{1}: \boldsymbol{a} - \boldsymbol{object} \\ \mathbf{0}: \boldsymbol{a} - \boldsymbol{background} \end{cases}$$

$\boldsymbol{Weigths}$ –weight maps

The learning procedure can be represented as minimization problem:

$$\boldsymbol{ResW} = \mathbf{argmin}_{\boldsymbol{Weigth}}\, \boldsymbol{Q}(\,\mathbf{Base}, \boldsymbol{Weights}\,)$$

For minimization we would use a stochastic gradient descend. E. g. for every training example a_i ϵ Base, changes are iteratively made in the weights value, of the form:

$$\Delta = -\boldsymbol{\alpha}\, \frac{\partial \boldsymbol{q}(\boldsymbol{a}, \boldsymbol{Weigths})}{\partial \boldsymbol{Weights}} |_{a=a_i} \qquad (3)$$

where $\mathbf{q}(\boldsymbol{a}, \boldsymbol{Weigths}) = (\boldsymbol{val}(\boldsymbol{a}, \boldsymbol{Weigths}) - \boldsymbol{Ans}(\boldsymbol{a}))^2$

$\boldsymbol{\alpha} - speed\ parameter$

Nevertheless it should be noted that the number of nonzero elements at Δ for every training example is equal to W*H, which allows for a high speed of the process of weights correction. All in all this approach is basically similar to back propagation, commonly used in neural networks training.

As a first approximation, the probability estimate is used in a form of:

$$\boldsymbol{Weigth}_{\boldsymbol{LBPVAL,dx,dy}} = \boldsymbol{TP}/(\boldsymbol{TP} + \boldsymbol{FP}) \qquad \mathbf{(4)}$$

where TP – true positives
FP – false positives

Then the $\boldsymbol{Weigth}_{\boldsymbol{LBPVAL,dx,dy}}$ must be normalized, in such a way as to val (3) function highest value is no more than 1.0. Such approximation allows for getting a stable result for most real objects.

On completion of weights values learning, the value of strong classifier threshold is calculated. The point corresponding to the threshold is determined by a FAR/FRR graph, made for validation frame based on specific standards for the final classifier.

5. IMPLEMENTATION SPECIFICS

5.1 Learning

During the implementation of the described method in actual practice, a number of specifics should be used for the best results.

Since the assumed descriptors are less resistant towards the scale change than classical Haar-like features, the training selection should include artificial scale and rotation angle distortion.

As "background" examples should be used a small random subset of real images cuts. This set iteratively grown during learning process trough adding wrong classified examples from huge dataset of real data.

5.2 Scale invariance

As aforesaid, some small scale fluctuations are recorded directly into the training selection. A classical method provided for an image pyramid is used to record more tangible distortions.

5.3 Cascades

It is possible to use cascades technique using the discussed classifiers in a similar way to Viola-Jones (Viola 2001) cascade classifiers. The cascades tuned to different scales and working with different level of the pyramid are the most appropriate.

5.4 Voting process

The voting is the most resource-hungry task in Hough texture transform operation. Let us consider the formula (2). The sum operation is the only operation used there. The algorithm structure allows for the full use of SSE2 and AVX vector commands. It should be noted that it is possible to use 16-bit целое представление for weights maps and for accumulator with little to no losses of detection quality, which makes using vector commands to the full extent possible.

Another opportunity for huge performance improvement is using weights maps for two nearby descriptors. Such weights maps can be easily extracted from the initials. Although it leads to the huge increase in the used memory, it makes the voting really faster.

6. TESTING RESULTS

A number of classifiers was trained for the identification of 24 basic points on the digital image of a face (see figure 5) with the help of such method. "Faces in the Wild" database (LFW from "http://vis-www.cs.umass.edu/lfw/") was used in this testing. The special property of this database is that all its images were taken in unregulated conditions. The manually labeled images subset were used as a reference. The response is considered true if identified coordinates value's deviation on from the reference image is no more that 5% of the space between the eyes. The testing results are shown in the table 1.

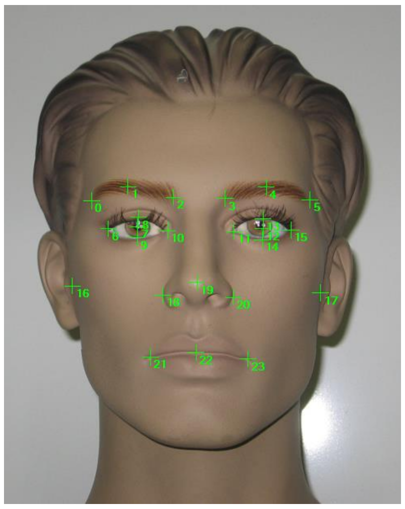

Figure 5. Facial features.

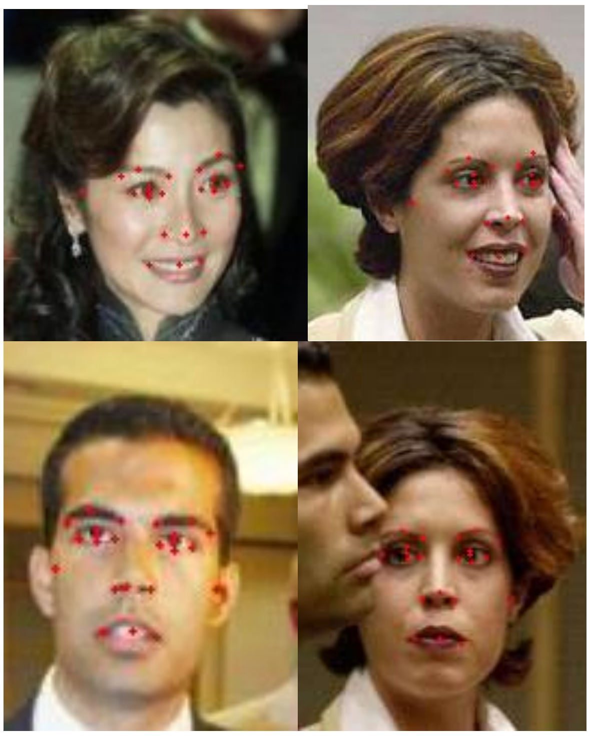

Figure 6. Detection results.

Feature point id (see Fig 5)	TPR(FPR = 0.99)
0/5	0,81
1/4	0,83
2/3	0,92
6/15	0,88
7/12	0.99
8/13	0,95
9/14	0,95
10/11	0.96
16/17	0,72
18/20	0,97
19	0,96
21/23	0,97
22	0,95

Table 1. Results on LFW Dataset

All in all the results of classifiers' work are on a par with modern methods of identification of the facial features.

7. CONCLUSIONS

This article discusses the unique method of the objects identification in the image. The main feature of this meth is using Hough-like voting process. Texture LPB-descriptors are used as voting primitives. Voting weights are defined by training in some training selection. The work describes the training algorithm in detail, and gives recommendations on its implementation. The method was tested by the identification of the special points on the faces from the LFW database. The results confirm the efficiency of the method and its actionability.

REFERENCES

Viola, Jones: Robust Real-time Object Detection, //IJCV 2001 p 1-25

Juergen Gall, Victor Lempitsky: Class-Specific Hough Forests for Object Detection, //CVPR 2009 pp. 1022-1029

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool: SURF: Speeded Up Robust Features // Computer Vision and Image Understanding (CVIU).Vol. 110, No. 3, pp. 346–359, 2008.

Lowe, David G. "Object recognition from local scale-invariant features"//International Conference on Computer Vision - ICCV , vol. 2, pp. 1150-1157, 1999

Pablo F. Alcantarillaa, Luis M. Bergasab, Andrew J. Davisonc: Gauge-SURF Descriptors // Image and Vision Computing 2013

Rojas, R. AdaBoost and the super bowl of classifiers a tutorial introduction to adaptive boosting. //Freie University, Berlin, Tech. Rep. 2009