Computer Science Journal of Moldova, vol.**11**, no.2(32), 2003

Combined Randomized-Local Hough Transform versus UpWrite Transform in stamp detection

Daniela Tondini

Abstract

The conventional Hough Transform used for detection of objects with known shape and size has proved its robustness. One typical task for this transform can be the detection of stamp(s) on an envelope. Unfortunately, the Hough Transform has an important drawback: the heavy computational effort and, as consequence, a big execution time. This paper introduces a variant of Hough Transform that speeds up the process. One important aid is given by a filtering step based on a fast analysis of a rough deformation model. This method is a combination of Randomized and Local Hough Transform. Experiments were made comparing the modified Hough Transform approach with the UpWrite Transform and they proved that the first approach preserves the quality of the Hough Transform results at a higher speed.

Keywords: Hough Transform, Randomized Hough Transform, Local Hough Transform, UpWrite Transform, stamp detection, fast algorithm.

1 Introduction

Both the Hough Transform (HT) and UpWrite Transform are efficient methods to detect patterns from noisy images. The Hough Transform involves building a two-dimensional voting array using all pixels of the image, which requires much computing time. Thus, many variations (the Local HT, the Randomized HT, the Probabilistic HT, the Fast HT etc.) were created with the declared purpose to reduce the amount of computation. The Local Hough Transform (LHT) rapidly detects lines using local polar coordinates with the pole being a point within the image, using only the deviation angle as a parameter in the linear equation. This means that a one-dimensional voting array will suffice to detect lines, and thus there is a substantial reduction of computing load. The Randomized Hough Transform (RHT) doesn't examine all the combinations of pixels needed, but randomly chooses for examination sets of pixels (the number of pixels examined at a time depends upon the shape to be detected). In order to increase the speed of the detection, the author combined the LHT with RHT, obtaining a better breed of algorithm. This new method was compared with the Up Write Transform on stamp detection.

2 Hough Transform

The Hough Transform (HT), invented by Hough in 1962, is a well-accepted method used for object detection. It requires that we explicitly choose a class of objects for recognition (e.g. lines, circles etc.) and a parameterization of this class. This parameterization describes all possible "ideal" instances of the object. However, images are rarely ideal and the parameterization offers no explicit mechanism for tolerating noise. To overcome this inherent difficulty, in practice the computational results from a Hough Transform are recorded in a histogram, providing a rough quantization of any computations. This allows some tolerance to noise by implicitly assuming that any noise in the image will only affect computations by "perturbing the numbers a little". While for some simple forms of noise (e.g. speckle noise) this may hold, for many others (like variations in curvature) it fails. Under such noise, the Hough Transform's accuracy may degenerate.

Hough Transform Principle. The principle is described as it is used for line detection, in order to keep the introductory sections clear and short. The circle detection will be detailed in the section explaining how the local and the randomized approaches were combined. Any black pixel in a black/white image has indefinitely many straight lines that could pass through it, one for every possible angle. To describe an ideal line it can be used the slope-intercept form of the equation for a line:

$$y = mx + b,$$

with (x, y) being the coordinates of the pixel involved, m being the slope of the line, and b representing the coordinate on Y axis where the line intersects the axis. By interpreting the equation differently, so that x and y are constants and m and b are coordinates, the equation can be recognized as:

$$b = -xm + y,$$

which is the equation of a line in the (m, b) space. Thus, each point in the two-dimensional image space (X, Y) corresponds to a straight line in (m, b) coordinates. More important, the points in the (m, b) space (called Hough space) where two lines intersect correspond to collinear points in image space. This is useless if only two points are taken into consideration, but this result proves useful when considering multiple intersections: if N straight lines in Hough space that correspond to N given pixels in image space intersect at a point, then those N pixels reside on the same straight line in image coordinates. The parameters of that line correspond to the Hough coordinates (m, b) of the point of intersection. This is the basis for the Hough transform: all pixels are converted into lines in (m, b) space, and the points of intersection of many lines are collected into line segments. When implementing the Hough transform, a degree of quantization in (m, b) coordinates is decided upon in advance, and a Hough image is created. For each pixel in the original image, the line in Hough space is computed, and each pixel on that line in the Hough image is incremented. After all pixels have been processed, the pixels in the Hough image that have the largest values correspond to the largest number of collinear pixels in the original image. The slope-intercept line equation is unable to deal with vertical lines because the slope becomes infinite. There are other forms of the equation of a line that do not have this pitfall, including the normal form:

$$\rho = x * \cos\Phi + y * \sin\Phi$$

where ρ is the perpendicular distance from the origin to the line, and Φ is the angle of the perpendicular line to the X axis.

2.1 Local Hough Transform (LHT)

2.1.1 Representation of lines in polar coordinates

Line equation in polar coordinates. Consider a system of polar coordinates with the pole being the origin O of the orthogonal XY coordinates, and the baseline being the X axis. Denoting the radius vector by r, and deviation by Φ, an arbitrary point P on line l is represented by polar coordinates (r, Φ). (ρ_0, θ_0) are the coordinates of the foot of a perpendicular drawn from the pole to the line l. The equation of the line l in polar coordinates is:

$$r \cos(\Phi - \theta_0) = \rho_0 \tag{1}$$

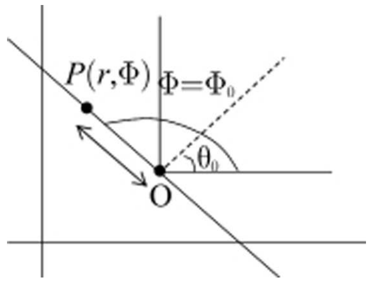


Figure 1. Local polar coordinates

Line equation in local polar coordinates. In the above specific case, it is considered a system of coordinates with the pole at a point on the line l as shown in Figure 1., which is referred to as local polar coordinates. With the local polar coordinates, the X axis is taken as baseline and an arbitrary point P on the line l is represented by the local polar coordinates (r, Φ), with r and Φ being radius vector and deviation, respectively. The foot of a line drawn from the pole O' perpendicular on the line l is coincident with the pole O', its local polar coordinates being (O, Φ_0). The line simple equation in such local polar coordinates is an expression determined only by the deviation Φ:

$$\Phi = \Phi_0 \tag{2}$$

Here Φ_0 is a constant representing the angle that line l makes with the X axis. As to the relation between Φ_0 and θ_0, Figure 1 shows that $\Phi_0 = \theta_0 + \pi/2$ in the case of negative deviation of l, and $\Phi_0 = \theta_0 - \pi/2$ in the case of positive deviation. In terms of correspondence between the image plane and the parameter plane, for both positive and negative deviation of a line the parameters are restricted to the following ranges:

$$0 \leq \theta_0 < \pi, 0 \leq \Phi_0 < \pi, 0 \leq \Phi < \pi$$

Relation with the Hough transform. By expanding expression (1), and substituting the relations $R \cos \Phi = X$ for polar coordinates (r, Φ) and orthogonal coordinates (X, Y) of the point P, the following formula is derived:

$$X \cos \Phi_0 + Y \sin \Phi_0 = \rho_0 \tag{3}$$

This is the widely used equation of the $\rho - \theta$ Hough transform. In other words, the equations (1) and (3) are different representations of the same line. The fact that the pole O' of local polar coordinates belongs to the line means that $\rho_0 = 0$ in equation (1). Now $\cos(\Phi - \theta_0)$, so that $\Phi = \theta_0 \pm \frac{\pi}{2}$, and replacing Φ with φ, the line equation in local polar coordinates is derived. From the above it follows that equation (2) of the line in local polar coordinates may be interpreted as a specific case of the $\rho - \theta$ Hough transform.

2.1.2 Principle of line detection using local polar coordinates

An arbitrary line in an image can be represented in local polar coordinates with the pole residing on the line as in equation (2). Normally, in computer-based image processing, scanning starts from the left-upper corner of the image (origin of orthogonal coordinates). Therefore, the

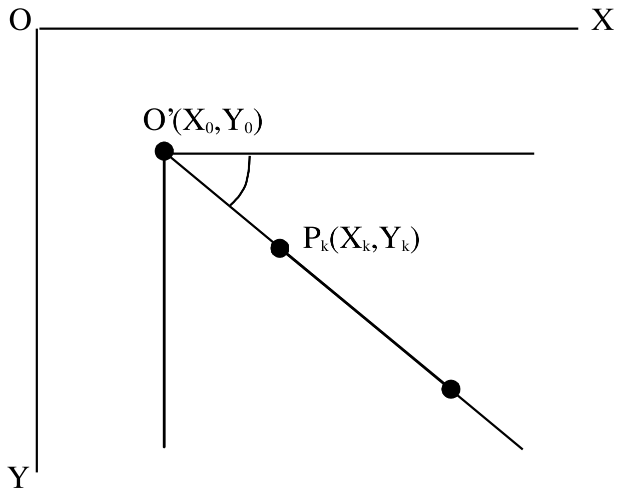


Figure 2. Line detection using the local polar coordinates

local pole is selected as the end point of a line with the smallest Y value (if multiple points have the same value, then the one with the smallest X value is selected). As shown in Figure 2, in local polar coordinates with the pole O' placed at the line end, if there are many pixels in the direction $\Phi = \Phi_k$, then those pixels can be thought to form a line. Therefore, if pixels existing along directions with deviation $\Phi = 0$ to π around the pole O' are arranged in a voting array $V[\Phi]$, then the line can be detected by increasing number of votes in the direction with deviation Φ.

2.2 Randomized Hough Transform

The Randomized Hough Transform uses a different mechanism for generating values in a histogram defined over the parameter space. Consider line detection. In the standard Hough Transform, a pixel in the image corresponds to a curve in the parameter space and this is quantified and recorded in the histogram. In the Randomized Hough Transform, a pair of pixels is randomly chosen and the parameters of the unique line passing through these pixels are computed. This line is recorded as a single entry in the histogram. This is iterated a preset number of times, where the number of iterations is much less than the number of pixel pairs in the image. In this way, entries are accumulated in the parameter space.

This algorithm is iterated to detect line segments one at a time. Thus, the global maximum of the parameter space histogram is found, and the equation of the corresponding line computed. The pixels on this line segment are then removed, leaving a simpler image to analyze. The algorithm is then repeated to find the next line. The algorithm stops when no lines are detected for a number of iterations.

To generalize the Randomized Hough Transform to circle detection, triplets of pixels are randomly chosen. The unique circle passing through each triplet is computed and recorded as an entry in the 3-D parameter space.

For each object extracted from the image, a simple post-processing step is applied. The number of pixels lying near the object (given some preset tolerance) are counted and divided by the number of pixels expected (the line length or circle/ellipse perimeter). If the proportion of pixels found is greater than some threshold value, the object is judged to exist, and is referred to as a true line, true circle or true ellipse (as appropriate). If too few pixels are counted, the object is discarded as noise.

2.3 Combined Randomized-Local Hough Transform

2.3.1 Concept

Both the Local Hough Transform and the Randomized Hough Transform came into play because of speed restriction and the high computation effort of the classical Hough Transform. The LHT tries to reduce the size of the accumulator while RHT considers that it is not necessary to fill the accumulator with all the possible data in order to extract useful information. It was natural to combine those methods to obtain an even faster algorithm.

2.3.2 Preliminary operations

The RLHT (Randomized-Local Hough Transform) requires some preprocessing operations. The steps are:

- Binarization: the transformation of the input image into a black/white image (for the situation when the input image has more then 2 colors)
- Accurate segmentation: the detection of BLOB (a BLOB is the representation of a set of connected pixels in the image, in the form of a bounding rectangle and, optionally, the shape or an approximation of it).
- BLOB filtering.

Each of these steps is treated here as a black box (no algorithm in particular is preferred over another). The only requirement for these preprocessing steps is to produce a description of BLOB shape or at least an approximation of it. It is widely accepted that Hough Transform is very robust in presence of additional structures in the image (other lines, circles, curves, or objects) as well as being insensitive to relatively small image noise. The preprocessing steps do not try to filter the image but instead to filter the input for Hough Transform, cutting down the searching space and the computational effort. The segmentation step produces a set of BLOB, which will be used to implement the concept of Local Transform.

2.3.3 Randomized - Local Hough Transform algorithm

The new algorithm (RLHT) for curve detection resulted from combining the LHT with the RHT is described below:

1. Quantize parameter space within the limits of BLOB geometric parameters. The number of parameters that describe the curve gives the dimensionality N of the parameter space.

2. Form an N-dimensional accumulator array A, with structure matching the quantization of parameter space.

3. Initialize all the cells of accumulator with 0.

4. Randomly choose N pixels (a loci vector, its size depending on the dimensionality of the parameter space) from the pixels that describe the shape of the current BLOB. If the BLOB doesn't contain an exact description of the shape, it can be used an approximation (profile information) in the same manner. Compute the curve describing the model and, for each element, increase the corresponding accumulator cells:

$$A(a) = A(a) + \Delta A$$

5. Local maxima in the accumulator array $A(a)$, exceeding a threshold, correspond to curves $f(x, y)$ that are present in the original image.

When looking for circles, the analytic expression $f(x, a)$ of the desired curve is: $(x_1 - a)^2 + (x_2 - b)^2 = r^2$ where the circle has the center in the coordinates (a, b) and the radius r. Therefore the accumulator data structure must be three-dimensional. If the accumulator value is smaller than a threshold, then the current analyzed BLOB does not respect the parameters of desired curve, so it is treated as noise.

2.3.4 Training step

Because of noise, image defects, geometrical distortions and the randomized character of the algorithm, it is necessarily an off line training step. In this step, it is extracted a p-tile threshold value that is used to discriminate curves from noise. Of course, this is possible when we can count on the uniformity of the shapes to be detected, i.e. the shapes have more or less the same size. The training step is implemented in a hierarchical manner. Basically, for each step (BLOB filtering and RLHT), values are detected for the parameters that wouldn't exclude *any* similar shapes (circles in our case), but would reject as much as possible different shapes. Practically, this was implemented using a minimum distance classifier. A relation was observed between the BLOB size and the accumulator p-tile threshold value. This can impose a direction in feature space so an Euclidean metric (where points at the same distance describe a circle in 2D) was avoided, but it was used instead a Mahalanobis metric (where the points at the same distance describe an ellipse in 2D) (Figure 3).

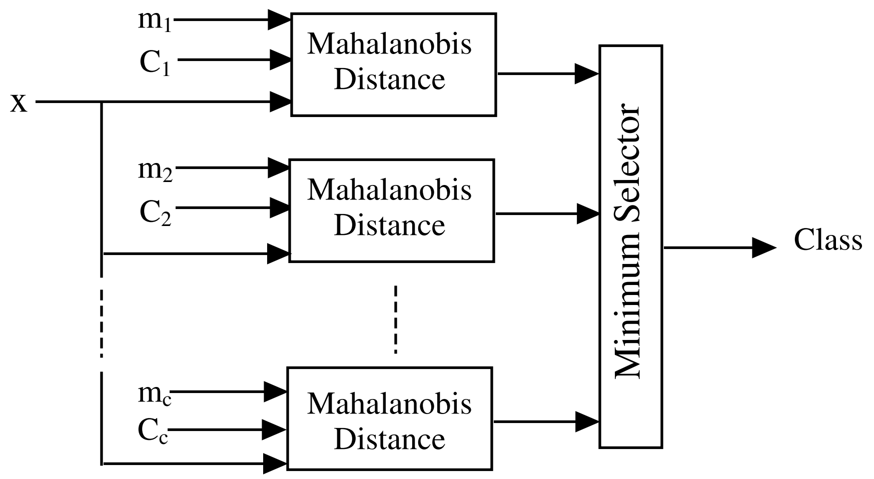


Figure 3. Minimum distance classifier using Mahalanobis distance (m is the mean, C is the covariance matrix)

3 Up Write Transform

The Up Write is another method for automatically detecting a class of geometric objects. It begins by modeling the image locally and then finding sets of these local models that it pieces to form an object. Each object is then described by computing moments, these being represented as a point in R^n for some n. This is not an explicit parameterization of the class of objects, but an embedding of the class of objects in the enclosing space R^n. The objects generally generate points lying on a manifold embedded in R^n, or on a union of manifolds. During recognition, an object is judged to be of a particular class if it corresponds to a point on the manifold. The Up Write comprises three stages. First, local models are computed of small regions of the image. Second, these local models are pieced together to form objects. Finally, each object is characterized by computing moments. These are used to decide whether an object belongs to the desired class.

A. Local Models

The first part, constructing local models, takes place with the following algorithm. This algorithm has been termed The Spot Algorithm.

1. *A resolution radius* is selected.
2. A pixel in the image is chosen and the mean m and covariance matrix C are computed for those pixels within a circular neighborhood of the radius r. C and m constitute a local model of the pixels within the neighborhood.
3. The pixels that are within the neighborhood are tagged.
4. The process is repeated from step 2, starting with an untagged pixel.
5. The process stops when no untagged pixels remained. This generates a set of local models that model the image at a resolution r.

Finally, each pixel in the image is assigned as belonging to the local model whose mean is nearest to the pixel. This information is later used in the recognition.

There are two initialization conditions that the Spot Algorithm is subject to, the first being the choice of resolution radius. Slight variations of resolution radius alter the set of local models generated. In section B (Objects), it is explained how this is used to improve the robustness of the Up Write algorithm. Secondly, in step 2 of the Spot Algorithm, a pixel is chosen as the center of a circular neighborhood. The local models generated are largely invariant of the actual pixels chosen. Thus, the decision of whether pixels are chosen randomly or sequentially is considered an implementation detail.

B. Objects

In the second stage of the Up Write, the local models that constitute an object are pieced together. This may be thought of as segmenting the set of local models into subsets, such that each subset contains the local models of a particular object. To perform this segmentation, each local model is regarded as a predictor of other local models. Predictions are based on estimates of local curvature. Consider the mean m of each local model as being a point on the object, and the dominant eigenvector of C as an estimate of the tangent to the object at the mean. This dominant eigenvector can be represented as a vector $[\cos(\theta); \sin(\theta)]$. By measuring the rate of change of angle θ as we progress along the object, an estimate of the curvature is obtained. The algorithm shown below in pseudo-code is used to group the local models that lie on an object:

```
Variables
integer $i$;
boolean success

Initialization
Choose two ungrouped local models which are close to each other
and group them. This group will be called $G$.
Computation
do
{
Let $n$ be the number of local models currently in the group $G$.
    if $(n \geq 4)$ then
            $i = 4$
    else
            $i = n$
    do
```

```
        {
            Estimate curvature using the last $i$ models that were added
            to the group $G$. Predict that the next local model will lie
            a little further along, with a mean and dominant eigen-
            vector as expected from the curvature.
            **if** such a local model exists **then**
            {
                remove it from the collection of ungrouped local mo-
                dels and add it to the group $G$.
                success = **TRUE**
            }
            **else**
            {
                $i = i - 1$
                success = **FALSE**
            }
        }
        **while** (success = FALSE) and ($i > 0$)
    }
    **while** ($i > 0$)
```

Having finished identifying the local models on an object, the process is repeated with the remaining ungrouped local models, identifying other objects. This process continues until no ungrouped local models remain. At completion, there will be several groups of local models, each corresponding to a different object in the image.

Above, the principles of perceptual organization are used to group the local models. The algorithm has been found to sometimes fail under either of two common conditions. The first is when two objects intersect. At such a point, the dominant eigenvector of a local model does not give a sensible estimate of tangent, thus estimates of curvature are inaccurate and the algorithm incorrectly predicts the next local model. This situation is evidenced by the algorithm being unable to find a local model corresponding to prediction. Reasoning that the most recently grouped local model is inaccurate, this is discarded and the process of prediction is repeated using the previous n local models, but now looking further along the curve for the predicted local model (with the intention of passing the intersection). If still no local model is found, it is considered that indeed the extreme of an object was reached, and grouping for this object stops.

The second condition under which the algorithm may fail is when noise in the curvature of the object (as in "wobbly" hand-drawn images) prompts inaccurate predictions. Such failures are unstable. If the resolution radius used to generate the local models in Section A (Local Models) is varied slightly, the failure will not occur. Therefore the first two stages of the Up Write are repeated, generating local models and grouping these with a range of resolution radii.

This produces a number of objects, some found several times. The task is to identify those objects which are stable under perturbation of the resolution radius, and discard those which were found only once, reasoning that these are merely artifacts of a particular resolution radius.

This is somehow complicated, as slightly different subsets of pixels are assigned to each object at different resolution radii. To gain a representation of each object which is largely invariant to this, first and second order central moments of each object (corresponding to the mean and covariance matrix respectively) are computed. The mean is described by two numbers and the covariance matrix (being a 2×2 symmetric matrix) by three numbers. These are concatenated to form a single point in $\Re^5$. Thus each object is mapped to a point in $\Re^5$, and stable objects manifest as clusters of such points.

The task now reduces to identifying clusters in $\Re^5$. These clusters will not be circular, but will vary notably in some directions whilst little in others. To extract the clusters, first a point in $\Re^5$ is randomly chosen and the mean m and covariance matrix C are computed for all the points within a small circular neighborhood. The covariance matrix gives a measure of in which directions the data varies. A basis B is defined for $\Re^5$, with the origin at the mean m and basis vectors that are the eigenvectors of C, each scaled by the square-root of the corresponding eigenvalue. Reasoning that the data does in fact vary evenly in all directions if one were to use the appropriate metric, the basis B is used to define an estimate of the local metric. This will just be the Euclidean metric relative to B.

Next the number of data points within a circular neighborhood (circular relative to the local metric defined from B) is counted. If this number is above some threshold value, a cluster corresponding to an object in the image has been identified. These points are removed from the data set in $\Re^5$, and the object which generated the point closest to the mean m is taken as representative. If the number of points is below the threshold value, there is no cluster, and the process is repeated with a different point.

C. Final Representation

At the completion of the previous stage, the algorithm will have produced a collection of objects. For each, the corresponding pixels are identified. Each object is finally represented by computing Zernike moments of these pixels. Using this particular class of moments is not a prescription by the Up Write algorithm, rather it is an implementation decision. Zernike moments were chosen as the most appropriate here as they allow translation, scale and rotation invariance to be built in. It was sufficient to compute second to fourth order Zernike moments. These produce a list of seven numbers, which are interpreted as a point in $\Re^7$. Thus each object has been mapped to a single point in $\Re^7$.

4 Stamp detection. Comparison and experiments

The detection of stamps on scanned images of envelopes is a very challenging problem, due to the usual bad quality of the stamp, which is almost never a clear image with solid edges. This is also a rewarding problem, since it has a very high practical value contributing at the automation of the process of mail sorting and mail delivery.

Even though the Hough Transform is a very powerful technique for curve detection, exponential growth of the accumulator data structure with the increase of the number of curve parameters restricts its practical usability. Since the amount of envelopes to be examined is huge and time is a very pressing issue, faster alternatives were considered.

If prior information about edge direction is used, computational demands can be decreased significantly. In the case of searching the circular boundary of a stamp, the circle is considered to have a constant radius $r = R$ for simplicity. Without using edge direction information, all accumulator cells $A\,(a, b)$ are incremented in the parameter space if the corresponding point (a, b) is on the circle with the center x. With knowledge of direction, only a small number of the accumulator cells need to be incremented. For example, if edge directions are quantized into eight possible values, only one-eight of the circle need to take part in incrementing the accumulator cells. Of course, estimates of edge direction are unlikely to be precise - if we anticipate edge direction errors of $\pi/4$, three eighths of the circle will require accumulator cell incrementing. Using edge directions, candidates for the parameters a, b can be identified from following formulae:

$$a = x_1 - R\cos(\psi(x))$$

$$b = x_2 - R\sin(\psi(x))$$

$$\psi(x) \in [\Phi(x) - \Delta\Phi, \Phi(x) + \Delta\Phi]$$

where $\Phi(x)$ refers to the edge directions in pixel x and $\Delta\Phi$ is the maximum anticipated error. Accumulator cells are then incremented only if the (a, b) satisfy the previous equation.

Other optimizations do not refer at reducing the computational complexity of the Hough Transform, but at filtering the actual input of the Hough Transform (classical Hough Transform, Local Hough Transform, Randomized Hough Transform). From the very first step of the algorithm (binarization) a great loss of information happens (see Figure 4 and Figure 5), resulting in small noise and circles broken in lots of small pieces.

Consequently, the preprocessing steps gain a significant importance, as it is required to differentiate between noise or small, useless pieces of

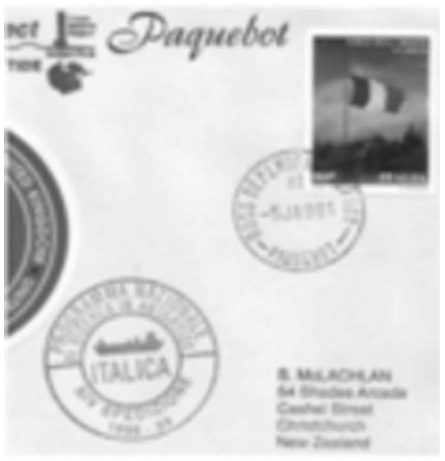


Figure 4. The gray image of a piece of envelope

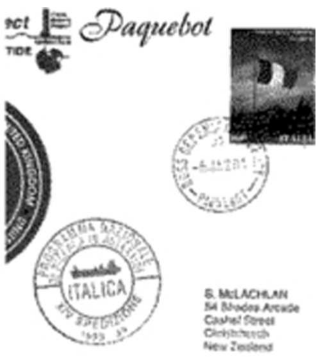


Figure 5. The black/white image

the stamp and the significant pieces (the ones big enough to estimate the position of the stamp).

One approach is to analyze the profiles of a *blob* (set of connecting black pixels) and conditioning a rough deformation model. After detecting the BLOB, it is considered for the HT processing only if its size is bigger then a threshold (this works when the size of the shapes to be detected is already known, i.e. for stamps - as in this case - or for coins). Also having information about the shape of the BLOB, the current BLOB is considered for HT computation only if at least one of its profiles is convex. In this way, we do not exclude broken circles or very noisy ones (see Fig. 6)

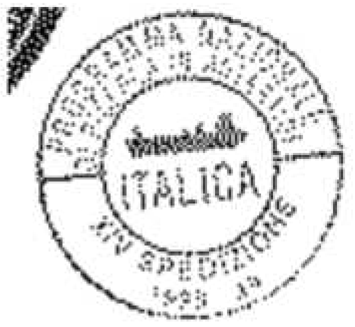
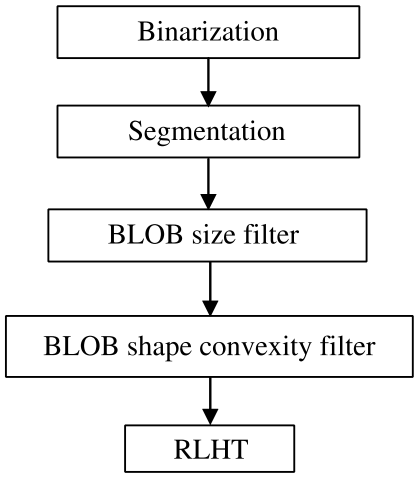


Figure 6. The poor quality of the stamp due to binarization.

For this filtering step, it was very useful the off-line training step, that yield a value of $28 - 30\%$ for the p-tile threshold value, used here to filter.

In the implementation of the RLHT algorithm the steps were followed as described in section 2.3.3. Experiments yield as optimum value for the randomized part of the algorithm a percentage of $75 - 80\%$ of pixels situated on the profile of the stamp to be considered for the computation. The major steps of the stamp detection are as below:



The Up Write transform was also considered for the reduced speed of computing maintaining reasonable detection results. A standard implementation of the algorithm described in section 3 was used in order to evaluate the efficiency of the RLHT algorithm, for both results and speed.

The second approach, using Hough transform, turned out to be equal in accuracy with the Up Write transform, but faster with about 35%. The test was done on a set of 1282 black/white images with envelopes. One very important quality of this approach is that no "good" circle was missed, yielding a value of success of 95%. The remaining 5% were due to the very poor quality of the image (circles split in very small segments, below the threshold value, with large gaps between them), resulted after binarization.

One weakness of this method is its sensitivity to the off line training step. We observed that off-line detected parameters slightly depend on the used data set so a relaxed policy is recommended.

5 Conclusions

Generally the main flaw of the Hough Transform is considered to be the complexity and the big amount of time required by the computation, so on tasks that demand speed other transforms are usually considered. In the search for a good and fast algorithm that would detect stamps, it was found that the Hough Transform approach still has many aspects to explore in order to achieve a better algorithm, especially when useful assumptions can be made on the shapes to be detected.

This paper proposes a fast method for curve detection, and in particular stamp detection, using a hybrid from Randomized Hough Transform and Local Hough Transform as well as some heuristic techniques in a hierarchical manner. Experiments made on 1282 envelope images showed that RLHT performs a good detection in terms of accuracy (95%) and speed.

As a future development, a more general theoretical framework can be purposed for curve detection. Future work will include the some residual distortion modeling in the parameter space.

References

[1] Ichikawa, K.; Aibara, T.; Muranaka, M.; Izumida, M.; Murakami, K. - *Automated assessment of the appearance of seam pucker on textiles using Hough transform*, Visual Communications and Image Processing '96, 17 – 20 March 1996, Orlando, Florida.

[2] Ichikawa, Koji; Izumida, Masanori; Murakami, Kenji - *A Method of Detecting Lines Using Local Polar Coordinates*, Systems and Computers in Japan, Vol. 28, No. 13, 1997.

[3] Kiryati, Nahum; Kalviainen, Heikki; Alaoutinen, Satu - *Randomized or Probabilistic Hough Transform: Unified Performance Evaluation*, http://citeseer.nj.nec.com, 1999.

[4] McLaughlin, Robert A. Technical Report - *Randomized Hough Transform: Improved Ellipse Detection with Comparison*, 1999.

[5] McLaughlin - *Randomized Hough Transform: Improved Ellipse Detection with Comparison*, http://citeseer.nj.nec.com, 1998.

[6] McLaughlin, Robert A.; Alder, Michael D. - Technical Report - *The Hough Tranform versus UpWrite*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997.

[7] Parker, J. R. - *Algorithms for Image Processing and Computer Vision*, Wiley Computer Publishing, 1997.

[8] M. Sonka, V. Hlavac, R. Boyle, *Image Processing Analysis and Machine Vision*, Second Edition, Brooks/Cole Publishing, 1999

Daniela Tondini,
MET Department,
University of Teramo, Italy,
E–mail : *dtondini@yahoo.it*

Received May 27, 2003