Pergamon

Pattern Recognition, Vol. 30, No. 12, pp. 2031–2042, 1997

0031-3203/97 $17.00+.00

PII: S0031-3203(97)00025-3

FUZZY CELL HOUGH TRANSFORM FOR CURVE DETECTION

VASSILIOS CHATZIS* and IOANNIS PITAS
Department of Informatics, University of Thessaloniki, Thessaloniki 540 06, Greece

(*Received* 1 *August* 1996; *in revised form* 4 *February* 1997)

Abstract—In this paper a new variation of Hough Transform is proposed. It can be used to detect shapes or contours in an image, with better accuracy, especially in noisy images. The parameter space of Hough Transform is split into fuzzy cells which are defined as fuzzy numbers. This fuzzy split provides the advantage to use the uncertainty of the contour point location which is increased when noisy images are used. By using fuzzy cells, each contour point in the spatial domain contributes in more than one fuzzy cell in the parameter space. The array that is created after the fuzzy voting process is smoother than in the crisp case and the effect of noise is reduced. The curves can now be detected with better accuracy. The computation time that is slightly increased by this method, can be minimized in comparison with classical Hough Transform, by using recursively the fuzzy voting process in a roughly split parameter space, to create a multiresolution fuzzily split parameter space.

Fuzzy theory Hough transform Curve detection Fuzzy cell Fuzzy voting

1. INTRODUCTION

In this paper, we propose a novel method called Fuzzy Cell Hough Transform, based on a fuzzy split of the Hough Transform parameter space. The Hough Transform[1] is of fundamental importance for many applications in image processing and computer vision. It can be used to detect any curve described by a number of parameters. The main disadvantage of the conventional Hough Transform is its large storage and computational time requirements, depending on the number of parameters and the split of the parameter space.[2,3] Up to now, a lot of variations have been proposed in the literature, mainly regarding the reduction of its computational complexity.[4–9] Most of them can be modified to incorporate the proposed use of fuzzy cells.

Another problem of Hough Transform is its performance in noisy images. Frequently, an image is noisy due to acquisition noise and degradation. Thus, contour points may be moved far enough from the actual curve and they may not contribute to the right Hough Transform cell. Fuzzy Hough Transform[10] tries to use the uncertainty of the contour point location, by substituting the contour points with fuzzy contour points. Furthermore, when an edge image is used as an input, cells are voted by edge points that do not correspond to specific curves and the corresponding values are often as large as the values of cells voted by actual curves.[11] In such cases Hough Transform leads to false detections. The use of fuzzy cells help us to reduce false detections and estimate the contours with better accuracy, especially when the images are corrupted by noise. Accuracy can be traded with computational time, if needed.

* Author to whom correspondence should be addressed. Tel., Fax: +30-31-996304; e-mails: [chatzis,pitas]@zeus.csd.auth.gr

Fuzzy Cell Hough Transform uses a fuzzy split of the Hough Transform parameter space which leads to a fuzzy voting process. The parameter space is split into fuzzy cells, which are considered as fuzzy sets with membership functions $\mu(a_1, a_2, \ldots, a_p)$ of p parameters. The values of the membership functions are limited in the range [0,1]. The domain of definition of each membership function, which is called interval of confidence, can intersect with neighbouring domains. Each contour point (x_j, y_j) in the spatial domain, contributes to more than one fuzzy cells in the parameter space (accumulator array). Moreover, the contribution of each point is not constant, but depends on the value of the cell membership function at the specific transformed point $(a_1, a_2, \ldots, a_p)$. Then, the local maxima in the array are found and the corresponding contours are detected.

Furthermore, we propose Select and Split Fuzzy Cell Hough Transform, a recursive method to use fuzzy cells. In this method, the parameter space is split in a small number of fuzzy cells and Fuzzy Cell Hough Transform is applied. After the voting process, the local maxima of the accumulator array are detected. The fuzzy cells that correspond to the maxima are split into a small number of fuzzy cells. Fuzzy Cell Hough Transform is again applied only on the regions of the parameter space that correspond to the maxima. The regions of the parameter space which do not contain contours are rejected during the iterations. As a result the computation time is significantly decreased.

In the following, the definitions of the fuzzy cells are introduced in Section 2. In Section 3, the fuzzy voting process of Fuzzy Cell Hough Transform is described. In Section 4, the Select and Split Fuzzy Cell Hough Transform is described and in Section 5 experimental results of the use of the proposed methods

are presented and compared with classical Hough Transform and Fuzzy Hough Transform. Conclusions are drawn in Section 6.

2. DEFINITIONS OF FUZZY CELLS

2.1. Definitions of fuzzy cells in a two-dimensional parameter space

The polar parameter space of Hough Transform is commonly used to detect straight lines in an image. Two parameters are needed and the transform equation is given by

$$\rho = x\cos\theta + y\sin\theta, \tag{1}$$

where the parameter $\theta \in [0, \pi)$ is an angle, $\rho \in [-\sqrt{2}N/2, \sqrt{2}N/2]$ is the algebraic distance to the origin and $N \times N$ is the image size. Then, the straight line given by equation (1) is transformed to a point (ρ, θ) in the parameter space. The parameters θ, ρ are split in N_θ and N_ρ sets (intervals) respectively, symbolized as Θ_i, P_i. The crisp cell C_{ij} where classical Hough Transform is applied, can now be defined as:

$$C_{ij} = \{(\rho, \theta),\ \rho \in P_i,\ \theta \in \Theta_j\}. \tag{2}$$

Thus, a couple (ρ, θ) belongs to a cell C_{ij} if $\rho \in P_i$ and $\theta \in \Theta_j$.

In order to define fuzzy cells, we split each parameter in fuzzy, instead of crisp, sets. By using the same number of partitions for each parameter, the fuzzy sets in θ coordinate are defined by the following equation:

$$\Theta_j^f = \{(\theta, \mu_{\Theta_j^f}(\theta)),\ \theta \in R\}. \tag{3}$$

where $\mu_{\Theta_j^f}(\theta)$ is a membership function. The interval of confidence of each fuzzy set Θ_j^f is assumed as the crisp set $(l_{\Theta_j^f}, r_{\Theta_j^f})$. This fuzzy split of the parameter θ provides the ability to use overlapped sets with different membership values for every θ. Then by using this fuzzy split in θ coordinate, a fuzzy-θ cell can be defined as a fuzzy number with two variables:

$$C_{ij}^\theta = \{((\rho, \theta), \mu_{C_{ij}^\theta}(\rho, \theta)),\ (\rho, \theta) \in R^2\}, \tag{4}$$

where

$$\mu_{C_{ij}^\theta}(\rho, \theta) = \begin{cases} \mu_{\Theta_j^f}(\theta) & \text{if } \rho \in P_i, \\ 0 & \text{elsewhere.} \end{cases} \tag{5}$$

The same technique can be used to split ρ coordinate and define a fuzzy-ρ cell C_{ij}^ρ.

If the fuzzy split in two coordinates is combined then a fuzzy-$\rho\theta$ cell is defined as

$$C_{ij}^{\rho\theta} = \{((\rho, \theta), \mu_{C_{ij}^{\rho\theta}}(\rho, \theta)),\ (\rho, \theta) \in R^2\}, \tag{6}$$

where

$$\mu_{C_{ij}^{\rho\theta}}(\rho, \theta) = \begin{cases} \min(\mu_{P_i^f}(\rho), \mu_{\Theta_j^f}(\theta)) & \text{if } \rho \in P_i \text{ and } \theta \in \Theta_j, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

2.2. Definitions of fuzzy cells in a three-dimensional parameter space

Let us assume that a circle in a $N \times N$ image is to be detected by using the conventional Hough Transform. The parameters used are the coordinates of the circle centre a, b and the radius r. When Hough Transform is applied, the three-dimensional parameter space is split in $N_a \times N_b \times N_r$ cells. The crisp cell C_{ijk} is defined by

$$C_{ijk} = \{(a, b, r),\ a \in A_i, b \in B_j, r \in R_k\}, \tag{8}$$

where A_i, B_j and R_k are crisp sets.

In order to define fuzzy cells, each point which belongs to the interval of confidence of the fuzzy cell, corresponds to a value in the range [0,1] through its membership function. The fuzzy sets in coordinate a can be defined as $A_i^f = \{(a, \mu_{A_i^f}(a)), a \in R\}$, where $\mu_{A_i^f}(a)$ is a membership function. Then, by using this fuzzy split, a fuzzy-a cell C_{ijk}^a can be defined as a fuzzy number with three variables:

$$C_{ijk}^a = \{((a, b, r), \mu_{C_{ijk}^a}(a, b, r)),\ (a, b, r) \in R^3\}, \tag{9}$$

where

$$\mu_{C_{ijk}^a}(a, b, r) = \begin{cases} \mu_{A_i^f}(a) & \text{if } b \in B_j \text{ and } r \in R_k, \\ 0 & \text{elsewhere.} \end{cases} \tag{10}$$

The same technique can be used to split b and r coordinate in fuzzy sets and define the corresponding fuzzy-b C_{ijk}^b and fuzzy-r C_{ijk}^r cells as fuzzy numbers with three variables.

If the fuzzy split in two coordinates, for example a and b coordinates, are combined then a fuzzy-ab cell can be defined as the following fuzzy number:

$$C_{ijk}^{ab} = \{((a, b, r), \mu_{C_{ijk}^{ab}}(a, b, r)),\ (a, b, r) \in R^3\}, \tag{11}$$

where

$$\mu_{C_{ijk}^{ab}}(a, b, r) = \begin{cases} \min(\mu_{A_i^f}(a), \mu_{B_j^f}(b)) & \text{if } r \in R_k, \\ 0 & \text{elsewhere.} \end{cases} \tag{12}$$

Similarly fuzzy-ar and fuzzy-br cells can be defined.

Finally, if the fuzzy split in three coordinates are combined, then a fuzzy-abr cell can be defined as the following fuzzy number:

$$C_{ijk}^{abr} = \{((a, b, r), \mu_{C_{ijk}^{abr}}(a, b, r)),\ (a, b, r) \in R^3\}, \tag{13}$$

where

$$\mu_{C_{ijk}^{abr}}(a, b, r) = \min(\mu_{A_i^f}(a),\ \mu_{B_j^f}(b),\ \mu_{R_k^f}(r)). \tag{14}$$

The concept of a fuzzy cell can be easily generalized to p dimensions.

3. DESCRIPTION OF THE FUZZY VOTING PROCESSES

3.1. Detection of straight lines by using fuzzy cells

Let us assume that a pixel x, y is a contour point in an image and that the parameter space is split into $N_\theta \times N_\rho$ fuzzy-ρ cells C_{kl}^ρ. By following the same process as in

conventional Hough Transform, the centres θ_i of the crisp sets Θ_i are used to compute the distances ρ_i by using the following equation:

$$\rho_i = x\cos\theta_i + y\sin\theta_i. \tag{15}$$

Each couple (θ_i, ρ_i) belongs to more than one fuzzy cells C^{ρ}_{kl}. The corresponding elements of the accumulator array $M(k,l)$ are increased by the values $V_{kl}(x,y) = \mu_{C^{\rho}_{kl}}(\theta_i, \rho_i)$. Finally, the local maxima in the array M must be detected. The array that is created after the fuzzy voting process is smoother than in the crisp case. This means that local maxima which correspond to the effect of noise in an image disappear. The straight lines can now be detected with better accuracy. However, this method slightly increases the computation time.

When fuzzy-$\rho\theta$ cells are used, the parameter θ is also split into fuzzy sets Θ^f_i as in equation (3). The parameter space is split into fuzzy-$\rho\theta$ cells $C^{\rho\theta}_{kl}$ defined in equations (6 and 7). The fuzzy numbers Θ^f_i are used to compute the fuzzy distances P_i by using the following fuzzy equation:

$$P_i = x\,\mathrm{Cos}\,\Theta^f_i + y\,\mathrm{Sin}\,\Theta^f_i, \tag{16}$$

where Cos, Sin are the fuzzy extensions of the crisp functions cos, sin and the addition and multiplications are supposed to be fuzzy operators.[12–14] Then the inclusions of each fuzzy couple (Θ^f_i, P^f_i) in every fuzzy cell $C^{\theta\rho}_{kl}$ have to be computed. Since Θ^f_i is equal to Θ^f_k, the inclusion of each fuzzy couple to a fuzzy cell is simplified to the inclusion of the computed fuzzy number P^f_i in the fuzzy set P^f_l given by the following equation:[14]

$$\mu_{T_{il}}(u) = \sup_{\rho: u = \mu_{P^f_i}(\rho)} \mu_{P^f_i}(\rho), \quad u \in [0,1]. \tag{17}$$

The compatibilities T_{il} of each fuzzy couple (Θ^f_i, P^f_i) in every fuzzy cell $C^{\theta\rho}_{kl}$ are computed and added to form the corresponding accumulator array $M^f(k,l)$. Note that, since the compatibilities are fuzzy numbers with intervals of confidence in [0,1], the array M^f is a two-dimensional array of fuzzy numbers and the additions of compatibilities are additions of fuzzy numbers. When the voting process is completed, the fuzzy values are defuzzified and the local maxima are detected.

3.2. *Detection of circles by using fuzzy cells*

Let us assume that a pixel (x,y) is a contour point in an image and that the parameter space is split into fuzzy-r cells C^r_{lmn}. By following a similar procedure to the two parameter case, the centres a_i of the crisp sets A_i and the centres b_j of the crisp sets B_j are used to compute the distances r_{ij} by solving the equation:

$$(x - a_i)^2 + (y - b_j)^2 = r^2_{ij}. \tag{18}$$

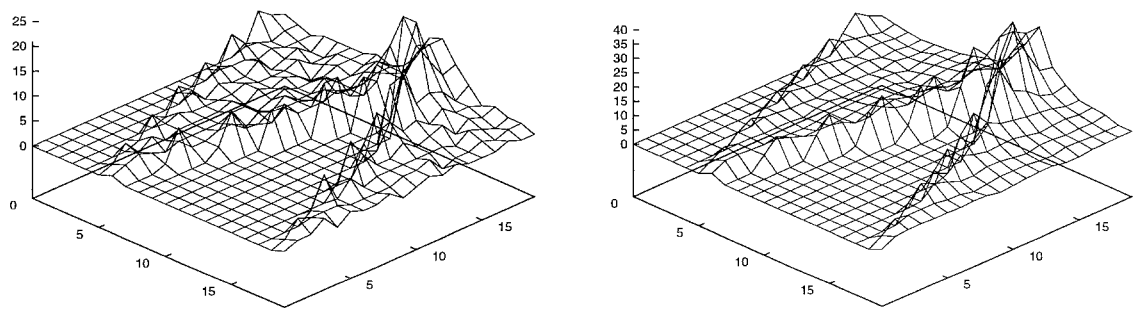
Each point (a_i, b_j, r_{ij}) belongs to more than one fuzzy cells C^r_{lmn}. The corresponding elements of the accumulator array $M(l,m,n)$ are increased by the values $V_{lmn}(x,y) = \mu_{C^r_{lmn}}(a_i, b_j, r_{ij})$. Finally, the local maxima in the array M have to be detected. The voting process is similar when fuzzy-a or fuzzy-b cells are used. By using fuzzy instead of crisp cells, the accumulator array is smoother and the number of local maxima is reduced. An example of the two coordinates a, r of an accumulator array that is created after a classical voting process is shown in Fig. 1(a), and after a fuzzy voting process in Fig. 1(b), when the methods are applied on an artificially generated image with one circle in it.

When fuzzy-ab cells C^{ab}_{lmn} are used defined as in equations (11 and 12), the voting process is more complicated but still similar to the one used on fuzzy-$\rho\theta$ cells for the detection of straight lines. The fuzzy numbers B^f_j and the crisp centres r_k of the crisp sets R_k are used to compute the fuzzy distances A^f_{jk} by solving the following fuzzy equation:

$$(x - A^f_{jk})^2 + (y - B^f_j)^2 = r^2_k, \tag{19}$$

where all the operators are the extended fuzzy operators, through the extension principle. The first root of equation (19) is the fuzzy number A^f_{jk} which is symbolized as

$$A^f_{jk} = \bigcup_{\alpha} \alpha \cdot [a^{(\alpha)}_{jk_l}, a^{(\alpha)}_{jk_r}], \quad \alpha \in [0,1], \tag{20}$$



(a) (b)

Fig. 1. Two coordinates of a three-dimensional accumulator array after a classical voting process (a) and after a fuzzy voting process (b).

and calculated by the equations:

$$a_{jk_l}^{(\alpha)} = x - \sqrt{r_k^2 - \min((y - b_{j_r}^{(\alpha)})^2, (y - b_{j_r}^{(\alpha)})(y - b_{j_l}^{(\alpha)}), (y - b_{j_l}^{(\alpha)})^2)}, \tag{21}$$

$$a_{jk_r}^{(\alpha)} = \begin{cases} x - \sqrt{r_k^2 - \max((y - b_{j_r}^{(\alpha)})^2, (y - b_{j_l}^{(\alpha)})^2)} & \text{if } r_k^2 \geq \max((y - b_{j_r}^{(\alpha)})^2, (y - b_{j_l}^{(\alpha)})^2), \\ x - \dfrac{r_k^2 - \max((y-b_{j_r}^{(\alpha)})^2, (y-b_{j_l}^{(\alpha)})^2)}{\sqrt{r_k^2 - \min((y-b_{j_r}^{(\alpha)})^2, (y-b_{j_r}^{(\alpha)})(y-b_{j_l}^{(\alpha)}), (y-b_{j_l}^{(\alpha)})^2)}} & \text{if } r_k^2 < \max((y - b_{j_r}^{(\alpha)})^2, (y - b_{j_l}^{(\alpha)})^2). \end{cases} \tag{22}$$

The second root can be calculated by similar equations. Then, the inclusions of each fuzzy triple (A_{jk}^f, B_j^f, r_k) in every fuzzy cell C_{lmn}^{ab} have to be computed. They can be simplified to the inclusion of the computed fuzzy number A_{jk}^f in the fuzzy set A_l^f given by the following equation:[14]

$$\mu_{T_{jkl}}(u) = \sup_{a:u=\mu_{A_l^f}(a)} \mu_{A_{jk}^f}(a), \quad u \in [0, 1]. \tag{23}$$

The compatibilities T_{jkl}, which are fuzzy numbers, are added to form the corresponding accumulator array $M^f(l, m, n)$ of fuzzy numbers. Fuzzy-*abr* cells can also be used to insert fuzziness to the detection of the centres and the radius of the circles. The voting process is similar and equations similar to equations (21 and 22) can be used.

4. SELECT AND SPLIT FUZZY CELL HOUGH TRANSFORM

4.1. Description of the algorithm

In order to decrease storage and time requirements of Fuzzy Cell Hough Transform, we propose the Select and Split Fuzzy Cell Hough Transform. This algorithm is an iterative procedure that uses Fuzzy Cell Hough Transform in a progressively refined split of the parameter space. Let us first assume that only one contour is to be detected in an image. Each coordinate of the parameter space is split in a small number of fuzzy sets. Note that the selection of the number of partitions affects the storage requirements of the accumulator array and the computational complexity of the algorithm. Fuzzy Cell Hough Transform is performed by using this roughly fuzzily split parameter space. After the voting process, the maximum in the accumulator array is detected. This maximum corresponds to a fuzzy cell (the winner), which is a first estimation of the detected contour parameters. Then the winner fuzzy cell is split into fuzzy cells by using a relatively small number of partitions for each coordinate. The union of their intervals of confidence is chosen to be the interval of confidence of the winner cell. Fuzzy Cell Hough Transform is again performed by using only the winner split in smaller fuzzy cells. After the voting process, a new winner is detected and the procedure continues until two successive detections are close enough. The algorithm can be summarized to the following steps:

1. Select the number of partitions for each coordinate of the parameter space.
2. Define fuzzy cells.
3. Apply Fuzzy Cell Hough Transform.
4. Detect the winner fuzzy cell.
5. If the winner cell parameters are close enough to the previous winner cell parameters, the algorithm ends. Else go to step 1 (to use new number of partitions) or step 2.

By using this algorithm we succeed to minimize storage requirements since the accumulator array that is created after a voting process can be used again during next iteration. Moreover, domains of the parameter space, that do not correspond to contours, are rejected during the first iterations. As a result, the necessary computational time is significantly reduced. In a subsequent section, the reduction in computational complexity and storage requirements will be discussed. The main disadvantage of this method is that the winner cell may not contain the actual contour. This may happen when the location of the actual contour is such that contour points vote almost equally to neighbouring cells. When such a false detection happens the algorithm cannot recover and detect the actual contour. False detections are more frequent when noisy images are used.

When more than one contours have to be detected, the accumulator array that is created after the fuzzy voting process have to be normalized before the detection of the local maxima, by dividing each item by an estimation of the number of the contour points which could vote to this cell. For example, when circles are detected in an image, the cells that correspond to circles with large radius are voted by more contour points than cells that correspond to circles with small radius. In this case the accumulator array has to be normalized by dividing each item by, e.g. the perimeter of the corresponding circle.

The parameter space can also be split to rough but crisp (not overlapped) cells. This variation known as Fast Hough Transform[7] can be considered as a special case of Select and Split Fuzzy Cell Hough Transform, since crisp sets can be considered as special cases of fuzzy sets. By using fuzzy instead of crisp cells, the accumulator array that is created after the voting process is smoother and the number of local maxima is reduced. Thus, the possibility of false detections during the iterations is reduced.

4.2. Storage and computational time requirements

Let us assume that in a $N \times N$ image one contour, described by p parameters $a_1, a_2, \ldots a_p$, is to be detected. Each parameter can be restricted to take values in a finite

interval (set) having size $A_1, A_2, \ldots A_p$. Let us also assume that the detection is satisfactory when the estimation error for each parameter is less than $e_i/2$, $i = 1, 2, \ldots, p$. This means that conventional Hough Transform can guarantee such an accuracy when the edges of the crisp cell in each coordinate is less than e_i. Thus, the necessary numbers of partitions N_i for each coordinate is given by

$$N_i = \frac{A_i}{e_i}, \quad i = 1, 2, \ldots, p, \tag{24}$$

and the size of the accumulator array is greater than $N_1 \times N_2 \times \cdots \times N_p$. On the contrary, when Select and Split Fuzzy Cell Hough Transform algorithm is used, the parameter space is roughly split. The size of the array depends on the maximum number of partitions M_i for every coordinate i for all iterations and it is given by $M_1 \times M_2 \times \cdots \times M_p$. When more than one contours are to be detected, the storage requirement is multiplied by the number of the local maxima which are detected in every iteration. Note that the storage requirement in the proposed algorithm does not depend on the necessary accuracy. Since M_i can be selected to be small enough it is obvious that storage requirements are much smaller when using Select and Split Fuzzy Cell Hough Transform in comparison to the conventional Hough Transform.

The computational time requirements of conventional Hough Transform depend on the number of contour points K that belong to an image, and the number of partitions for every coordinate of the parameter space. During the voting process, the transform equation has to be solved for every contour point (x, y) and for every parameter $a_1, a_2, \ldots, a_{p-1}$, which values are equal to the interval centres, to determine parameter a_p and vote in the corresponding cell. This means that the computational complexity T_h is proportional to the product:

$$T_h = K \cdot (N_1 \cdot N_2 \cdot \cdots \cdot N_{p-1}). \tag{25}$$

If the same number of partitions N is selected for every coordinate, the above product is simplified to

$$T_h = K \cdot N^{p-1}. \tag{26}$$

The computational time requirements of Select and Split Fuzzy Cell Hough Transform depend again on the number of contour points K, the number of partitions for every coordinate of the parameter space, as well as on the number of iterations needed to reach the necessary accuracy. In order to compare the algorithms, let us first assume that the parameter space is split during the iterations in rough crisp cells and that the number of partitions n remains the same for all iterations. We can also assume, without loss of generality, that $N = n^m$. This means that, the same accuracy that N partitions can guarantee, can also be reached in mth iteration by using n partitions in each iteration. The computational time T_c is now proportional to

$$T_c = m \cdot K \cdot n^{p-1}. \tag{27}$$

Since $p \geq 2$, $n \geq 2$ and $m \geq 1$, it can be easily proven that $T_h \geq T_c$ since $(n^{p-1})^m \geq mn^{p-1}$ and equality holds only when $m = 1$ or $m = n = p = 2$.

When the coordinate of the parameter space are fuzzily split, the fuzzy cells which are created, have overlapping intervals of confidence and are larger than the corresponding crisp ones. Thus, a fuzzy split of the parameter space cannot guarantee the accuracy of a crisp split with equal number of partitions. Let us assume that f_i is a factor that controls fuzziness for every coordinate of the parameter space $i = 1, 2, \ldots, p$. The length l_i of the interval of confidence for every fuzzy set that will be used during first iteration is given by

$$l_i^{(1)} = 2f_i \frac{A_i}{n_i}, \tag{28}$$

where A_i is the interval to be split and n_i is the number of partitions. After k iterations, by using the same number of partitions n_i, the length of the interval of confidence will be given by

$$l_i^{(k)} = \left(\frac{2f_i}{n_i}\right)^k A_i. \tag{29}$$

The accuracy of conventional Hough Transform is given by A_i/N_i. By using equation (29), it is proven that the proposed method can guarantee accuracy equal to the conventional Hough Transform after k iterations:

$$k = \max\{k_i\}, \quad i = 1, 2, \ldots, p, \tag{30}$$

where

$$k_i \geq \frac{\log N_i}{\log n_i - \log(2f_i)}. \tag{31}$$

Note that in case that one coordinate is selected to be split in crisp sets the corresponding factor f_i is equal to 0.5. The corresponding computational time T_f is now given by

$$T_f = k \cdot K \cdot (n_1 \cdot n_2 \cdot \cdots \cdot n_p) \tag{32}$$

or

$$T_f = k \cdot K \cdot n^{p-1}, \tag{33}$$

when equal number of partitions n is selected for every coordinate of the parameter space. It is obvious from equation (31) that f_i and n_i should be selected such that $n_i > 2f_i$. In any other case, the intervals of confidence lengthen after each iteration and accuracy cannot be reached. The computational time increases in comparison to crisp case. However, the time saving is significant in comparison to the conventional Hough Transform and increases when more parameters are needed to describe the contour or when it has to be detected with better accuracy.

5. EXPERIMENTAL RESULTS

5.1. Detection of straight lines by using Fuzzy Cell Hough Transform

We have performed extensive simulations to study the performance of the proposed approach. In order to

compare Fuzzy Cell Hough Transform (FCHT) with classical Hough Transform (HT) and Fuzzy Hough Transform (FHT) the number of correct detections of straight lines was used as a criterion. First, we assumed 100 artificially generated 256×256 images with only one straight line in each image. The line parameters ρ, θ were randomly selected. The images were corrupted by uniform noise having range $\pm d$ pixels added to the y-axis of a contour point (x, y). A detection was considered correct if the estimation error of each parameter was less than a certain error, depending on the interval where the parameters were restricted and the number of partitions. We used 30 partitions to split the intervals $\theta \in [0, 2\pi)$ and $\rho \in [10, 70]$ where the parameters were restricted. In the FCHT case, only the ρ parameter was fuzzily split. Fuzzy sets were chosen to have triangular slope. Their interval of confidence were controlled by a fuzziness factor f. When $f = 1$ the upper and lower limits were chosen to be in equal distances from the centre and equal to the distance of the centres of two neighbouring fuzzy sets. In the FHT case the fuzziness of contour points were supposed to have the same fuzziness as in FCHT case. The effect of fuzziness of the chosen triangular fuzzy sets was investigated as well. The intervals of confidence of the fuzzy sets were enlarged by using various values for the factor f. The results are presented in Table 1 and shown graphically in Figs 2 and 4(a). The performance of FCHT is better than HT. An improvement of 7%–31% on the number of HT correct detections was succeeded for different values of noise range. The computational time needed for FCHT is about 2 times more than the time needed for HT. The performances of FCHT and FHT are more or less similar, but the computational time needed for FHT is about 5 times more than FCHT and 10 times more than HT when $f = 1$ and it increases to about 70 times more than HT when $f = 3$.

Then, we investigated the case of multiple detections. We assumed 100 artificially generated images with five straight lines at random places. The images were corrupted by uniform noise having range $\pm d$ pixels as in the previous case. An example of such an image corrupted by uniform noise with range $d = \pm 10$ pixels is shown in Fig. 5(a). Five straight lines were detected in each image. The detections were considered correct if the estimation

Table 1. The number of successfully detected straight lines (out of 100) by using 100 images with one straight line at a random place, and the average time needed per image, by using HT, FHT and FCHT for different values of fuzziness factor f and noise range d

Method	Fuzziness	Detections				Time (s)			
	(f)	d=0	d=2	d=5	d=10	d=0	d=2	d=5	d=10
HT		62	68	74	72	0.03	0.03	0.03	0.03
FHT	1	64	71	75	75	0.26	0.25	0.25	0.25
	1.5	66	70	76	76	0.58	0.54	0.55	0.55
	2	69	74	77	76	0.95	0.90	0.94	0.92
	2.5	70	76	78	76	1.55	1.48	1.53	1.52
	3	75	75	78	74	2.17	2.05	2.11	2.11
FCHT	1	68	69	73	81	0.05	0.05	0.05	0.05
	1.5	66	70	76	78	0.05	0.05	0.05	0.05
	2	68	75	76	77	0.05	0.05	0.05	0.05
	2.5	75	78	79	79	0.05	0.05	0.05	0.05
	3	**81**	**82**	**79**	**82**	0.05	0.05	0.05	0.05

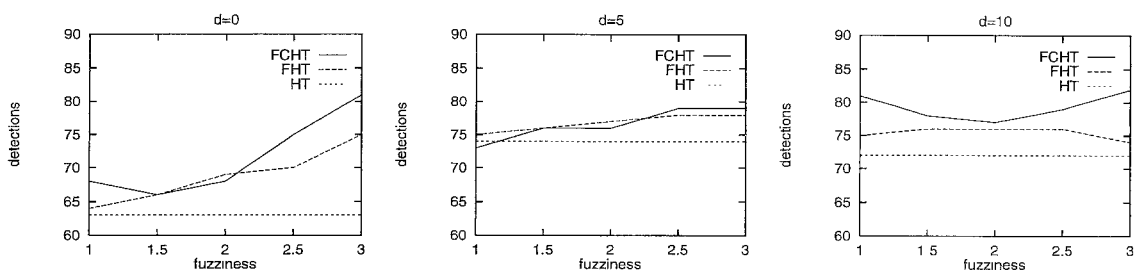


Fig. 2. The number of successfully detected straight lines (out of 100) for different values of fuzziness factor, by using 100 images with 1 straight line at random place, not corrupted by noise $d = 0$ (a), corrupted by uniform noise with range $d = 5$ (b) and $d = 10$ pixels (c).

Table 2. The number of successfully detected straight lines (out of 500) by using 100 images with five straight lines at random places, and the average time needed per image, by using HT, FHT and FCHT for different values of fuzziness factor f and noise range d

Method	Fuzziness	Detections				Time (s)			
	(f)	d=0	d=2	d=5	d=10	d=0	d=2	d=5	d=10
HT		297	286	300	271	0.1	0.2	0.2	0.2
FHT	1	310	309	308	284	1.4	2.7	3.3	3.6
	1.5	310	**319**	315	287	3.2	5.9	7.3	7.9
	2	322	311	314	296	5.3	9.9	12.3	13.3
	2.5	334	315	317	300	8.8	16.3	20.1	23.0
	3	337	312	309	307	12.3	22.7	28.1	30.8
FCHT	1	311	300	310	278	0.2	0.4	0.5	0.6
	1.5	317	315	320	297	0.2	0.4	0.5	0.6
	2	331	313	319	311	0.2	0.4	0.5	0.6
	2.5	**340**	318	318	310	0.2	0.4	0.5	0.6
	3	338	314	**323**	**319**	0.2	0.4	0.5	0.6

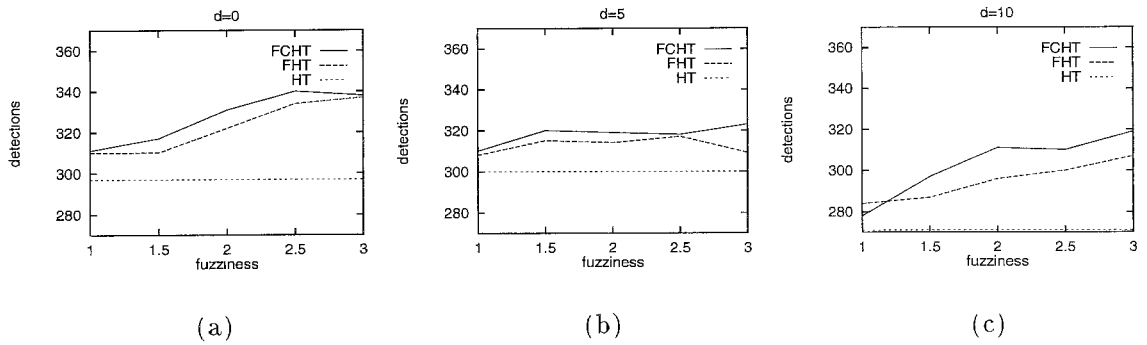


Fig. 3. The number of successfully detected straight lines (out of 500) for different values of fuzziness factor, by using 100 images with 5 straight lines at random places, not corrupted by noise $d = 0$ (a), corrupted by uniform noise with range $d = 5$ (b) and $d = 10$ pixels (c).

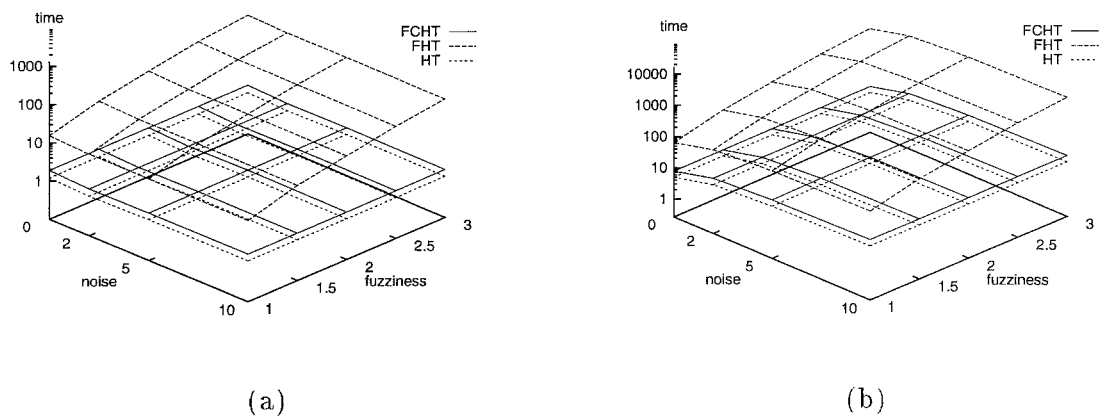
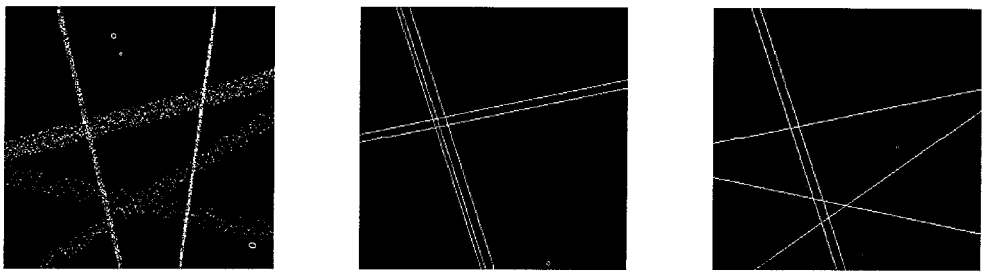


Fig. 4. The average time in seconds, needed for the detection of one straight line (a) and five straight lines (b) in an image, for different values of fuzziness factor and noise range.

error of each parameter for every line was less than a certain error defined as in the previous case. The effect of fuzziness was also investigated. The results are presented in Table 2 and shown graphically in Figs 3 and 4(b). By using FCHT an improvement of 8%–18% on the number of HT correct detections was succeeded for different values of noise range. The performances of FCHT and FHT are more or less similar but the computational time



(a) (b) (c)

Fig. 5. An example of an artificially generated 256×256 image with five straight lines, corrupted by uniform noise with range $d = \pm 10$ pixels (a), the detected lines by using classical HT (b) and FCHT with fuzziness factor $f = 2.0$ (c).

Table 3. The number of successfully detected circles (out of 100) by using 100 images with one circle at a random place, and the average time needed per image, by using HT, FHT and FCHT for different values of fuzziness factor f and noise range d

Method	Fuzziness	Detections				Time (s)			
	(f)	d=0	d=2	d=5	d=10	d=0	d=2	d=5	d=10
HT		61	65	64	54	1.2	2.2	2.5	2.7
FHT	1	70	71	71	57	16	32	43	49
	1.5	75	77	76	61	36	71	95	109
	2	76	77	75	67	60	119	160	184
	2.5	77	78	75	73	99	197	264	304
	3	78	77	76	72	139	275	369	424
FCHT	1	68	72	69	52	1.9	3.4	3.6	3.7
	1.5	76	75	**76**	60	1.9	3.4	3.6	3.7
	2	78	79	75	**73**	1.9	3.4	3.6	3.7
	2.5	76	78	75	72	1.9	3.4	3.6	3.7
	3	**81**	**80**	75	72	1.9	3.4	3.6	3.7

needed for FHT increases from 7 to 60 times more than the time needed for FCHT as the fuzziness increases from 1 to 3. Fig. 5(b) shows the straight lines that were detected by classical HT, when Fig. 5(a) was used as an input image. Fig. 5(c) shows the corresponding straight lines that were detected by FCHT with fuzziness factor $f = 2.0$. The proposed algorithm detects correct the four of the five straight lines whereas the classical HT detects only the two straight lines.

5.2. *Detection of circles by using Fuzzy Cell Hough Transform*

The performed simulations for the detection of circles were similar to the simulations for the detection of straight lines which were described in the previous section. First, we assumed 100 artificially generated 256×256 images with only one circle at a random place. The images were corrupted by uniform noise having range $\pm d$ pixels added to the ρ-coordinate of a contour point (x, y). A detection was considered correct if the estimation error of each parameter was less than a certain error, depending on the interval where the parameters were restricted and the number of partitions. We used 30 partitions to split the intervals $a \in [-128, 128]$, $b \in [-128, 128]$ and $r \in [10, 70]$ where the parameters were restricted. In the FCHT case, only the r parameter was fuzzily split. Fuzzy sets were chosen to have triangular slope and their interval of confidence were controlled by the fuzziness factor f. In the FHT case the fuzziness of contour points were supposed to have the same fuzziness as in FCHT case. The effect of fuzziness of the chosen triangular fuzzy sets was investigated as well. The results are presented in Table 3 and shown graphically in Figs 6 and 8(a). The performance of FCHT is always better than HT. An improvement of 19%–35% on the number of HT correct detections was succeeded for different values of noise range. The computational time needed is about 1.5 times more than the time needed for HT. The performances of FCHT and FHT are more or less similar, but the computational time needed for FHT is about 10 times more than FCHT and 15 times more than HT when $f = 1$ and it increases to about 115 times more than HT when $f = 3$.

Then, we investigated the case of multiple detections. We assumed 100 artificially generated images with five

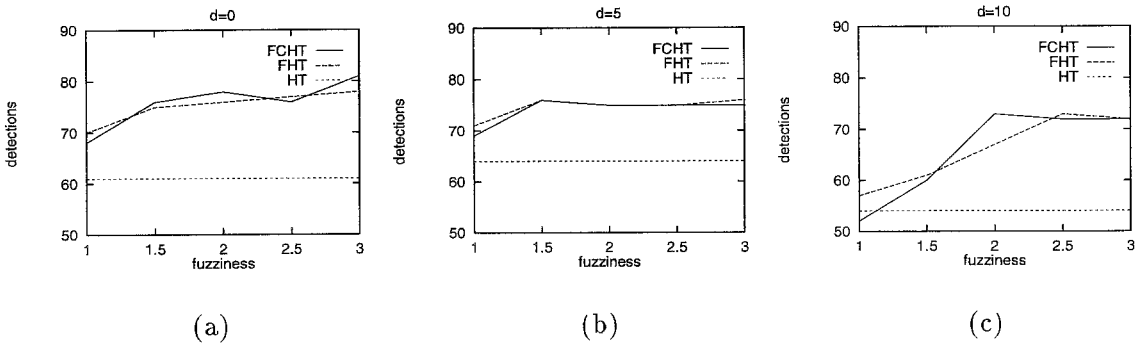


Fig. 6. The number of successfully detected circles (out of 100) for different values of fuzziness factor, by using 100 images with 1 circle at random place, not corrupted by noise $d = 0$ (a), corrupted by uniform noise with range $d = 5$ (b) and $d = 10$ pixels (c).

Table 4. The number of successfully detected circles (out of 500) by using 100 images with five circles at random places, and the average time needed per image, by using HT, FHT and FCHT for different values of fuzziness factor f and noise range d

Method	Fuzziness	Detections				Time (s)			
	(f)	d=0	d=2	d=5	d=10	d=0	d=2	d=5	d=10
HT		241	226	236	197	5.0	9.8	13.1	15.0
FHT	1	259	234	234	205	66	131	175	202
	1.5	281	238	248	220	147	292	390	448
	2	294	244	248	234	249	491	658	758
	2.5	299	255	255	236	410	811	1088	1251
	3	311	260	253	239	572	1133	1521	1744
FCHT	1	251	230	235	207	7.8	15.8	20.6	23.9
	1.5	287	234	249	224	7.8	15.8	20.6	23.9
	2	308	253	253	235	7.8	15.8	20.6	23.9
	2.5	313	261	258	245	7.8	15.8	20.6	23.9
	3	**319**	**266**	**270**	**249**	7.8	15.8	20.6	23.9

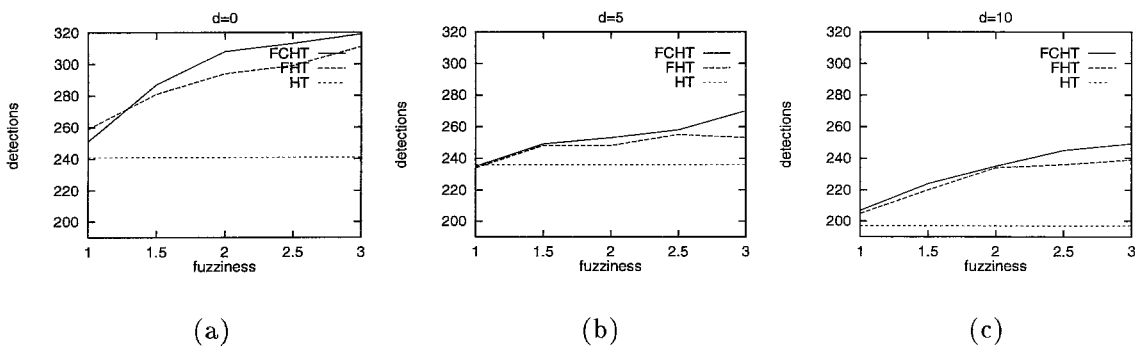
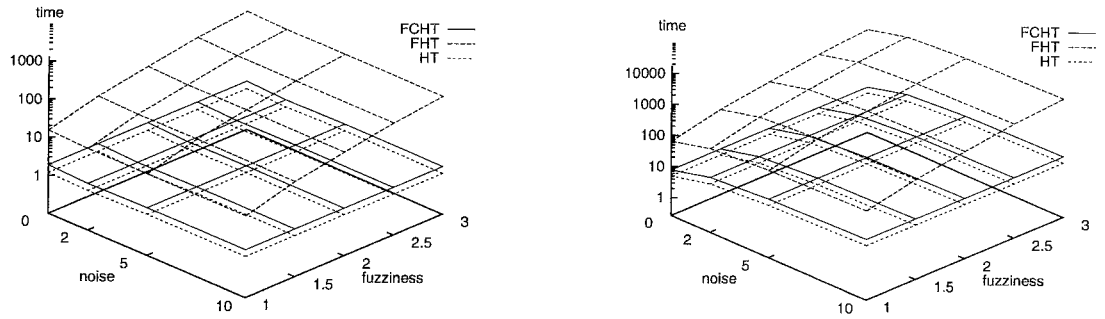


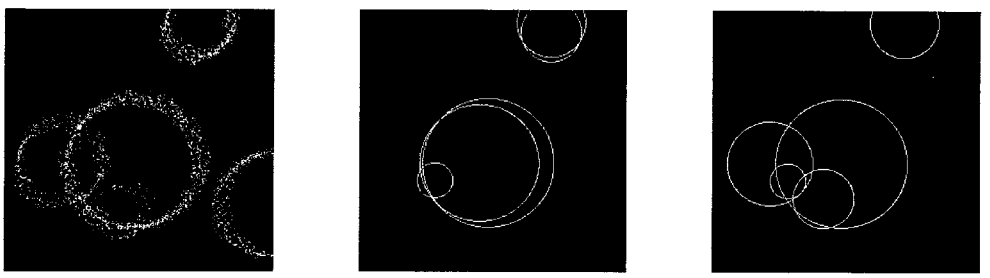
Fig. 7. The number of successfully detected circles (out of 500) for different values of fuzziness factor, by using 100 images with 5 circles at random places, not corrupted by noise $d = 0$ (a), corrupted by uniform noise with range $d = 5$ (b) and $d = 10$ pixels (c).

circles at random places. The images were corrupted by uniform noise having range $\pm d$ pixels as in the previous case. An example of such an image corrupted by uniform noise with range $d = \pm 10$ pixels is shown in Fig. 9(a). Five circles were detected in each image. The detections were considered correct if the estimation error of each parameter for every circle was less than a certain error defined as in the previous case. The effect of fuzziness was also investigated. The results are presented in Table 4 and shown graphically in Figs 7 and 8(b). By



(a) (b)

Fig. 8. The average time in seconds, needed for the detection of one circle (a) and five circles (b) in an image, for different values of fuzziness factor and noise range.



(a) (b) (c)

Fig. 9. An example of an artificially generated 256×256 image with five circles, corrupted by uniform noise with range $d = \pm 10$ pixels (a), the detected lines by using classical HT (b) and FCHT with fuzziness factor $f = 2.0$ (c).

using FCHT an improvement of 14%–32% on the number of HT correct detections was succeeded for different values of noise range. The performances of FCHT and FHT are still more or less similar but the time needed for FHT is significantly more. Fig. 9(b) shows the circles that were detected by classical HT, when Fig. 9(a) was used as an input image. Fig. 9(c) shows the corresponding circles that were detected by FCHT with fuzziness factor $f = 2.0$. The proposed algorithm detects correct the four of the five circles whereas the classical HT detects only the two circles.

5.3. Detection of circles by using Select and Split Fuzzy Cell Hough Transform

We have performed simulations on artificially generated 64×64 images that contain a circle to be detected. Parameters a, b, r were restricted in the range $a \in [-32, 32]$, $b \in [-32, 32]$ and $r \in [1, 21]$. The image was corrupted by uniform noise in the range $\pm d$ pixels added to the radial dimension of a contour point (x, y). First classical Hough Transform was used to detect the circle. The parameter space was split in 32 sets in each coordinate. By using this split the actual circle was detected with less than $e_i/2$ error in each parameter. The experiment was repeated for $N_c = 10,571$ different circles and four different values of noise range. Then HT was applied in a parameter space split in 16 sets in each parameter. Detections were supposed to be successful when the parameter estimation errors in each coordinate were less than or equal to the previously defined error threshold.

The same circles were detected by using Select and Split Fuzzy Cell Hough Transform (SSFCHT). First, coordinates a, b and r were split in $n = 4$ crisp cells in each iteration. Then, coordinate r was split in $n = 4$ fuzzy sets. Fuzzy sets were selected to have triangular membership function and the upper and lower limits were adjusted by the fuzziness factor f. When $f = 1$ the upper and lower limits were selected in equal distances from the centre and equal to the distance of the centres of two neighbouring fuzzy sets. The experiment was repeated for $f = 1$, 1.5 and 2. The detection was considered correct when the errors for every coordinate were less than or equal to the corresponding errors by using conventional HT.

The results are presented in Tables 5–7 for three different values of noise range. The number of successfully detected circles, the rate of success and the average time needed for the detection of one circle for every

Table 5. The number of successfully detected circles (out of 10,571), the rate of success and the average time needed by using HT ($N = 32$), HT ($N = 16$) and SSFCHT with crisp and fuzzy ($f = 1.0, 1.5, 2.0$) cells in an image not corrupted by noise

Method	Detected	Rate of success %	Average time (s)
HT (N=32)	10571	**100**	**1.12**
HT (N=16)	8358	79	0.46
SSFCHT (crisp)	6116	58	0.06
SSFCHT (f=1.0)	8111	77	0.07
SSFCHT (f=1.5)	10515	**99**	**0.07**
SSFCHT (f=2.0)	48	0.5	0.60

Table 6. The number of successfully detected circles (out of 10,571), the rate of success and the average time needed by using HT ($N = 32$), HT ($N = 16$) and SSFCHT with crisp and fuzzy ($f = 1.0, 1.5, 2.0$) cells in an image corrupted by uniform noise in the range ± 2 pixels

Method	Detected	Rate of success %	Average time (s)
HT (N=32)	10571	**100**	**1.15**
HT (N=16)	8358	79	0.48
SSFCHT (crisp)	5538	52	0.10
SSFCHT (f=1.0)	736	70	0.12
SSFCHT (f=1.5)	10438	**99**	**0.13**
SSFCHT (f=2.0)	178	2	0.65

Table 7. The number of successfully detected circles (out of 10,571), the rate of success and the average time needed by using HT ($N = 32$), HT ($N = 16$) and SSFCHT with crisp and fuzzy ($f = 1.0, 1.5, 2.0$) cells in an image corrupted by uniform noise in the range ± 5 pixels

Method	Detected	Rate of success %	Average time (s)
HT (N=32)	10571	**100**	**1.49**
HT (N=16)	3443	33	0.51
SSFCHT (crisp)	3302	31	0.11
SSFCHT (f=1.0)	4731	45	0.14
SSFCHT (f=1.5)	9012	**85**	**0.14**
SSFCHT (f=2.0)	3409	32	0.43

method are presented. A success rate of 99% is reached by using a factor $f = 1.5$ and the corresponding Fuzzy Cell Hough Transform is 16 times faster over the conventional HT algorithm. Note that accuracy cannot be reached in case of $f = 2$ since $n = 2f$.

6. CONCLUSIONS

We introduced the use of Fuzzy Cells in Hough Transform as a method to detect contours in an image. We proposed a fuzzy split of the Hough Transform parameter space which led to a fuzzy voting algorithm. The array that was created after the fuzzy voting process was smoother than in classical case. Local maxima that correspond to the effect of noise or any kind of uncertainty were disappeared. The number of correct detections of curves was increased. Moreover, they were detected with better accuracy in comparison to classical Hough Transform and, in most cases, in comparison to Fuzzy Hough Transform. We also proposed Select and Split Fuzzy Cell Hough Transform, the use of Fuzzy Cells in a recursively roughly split parameter space. This method decreases significantly the storage requirements and the computational time requirements in comparison to classical Hough Transform and does not practically decrease the performance of Hough Transform.

REFERENCES

1. P. V. C. Hough, Method and means for recognizing complex pattern, U.S. Patent 3069654 (December 1962).
2. H. Kalviainen, P. Hirvonen, L. Xu and E. Oja, Probabilistic and non-probabilistic hough transforms: Overview and comparisons, *Image Vision Comput.* **13**(4), 239–252 (May 1995).
3. V. F. Leavers, *Shape Detection in Computer Vision Using the Hough Transform.* Springer, Berlin (1992).
4. M. Atiquzzaman, Multiresolution hough transform—An efficient method of detecting patterns in images, *IEEE Trans. Pattern Analysis Mach. Intell.* **14**(11), 1090–1095 (1992).
5. D. Chang and S. Hashimoto, An inverse voting algorithm for Hough transform, *Proc. ICIP'94*, **1**, 223–227 (1994).
6. J. Illingworth and J. Kittler, The adaptive Hough transform, *IEEE Trans. Pattern Analysis Mach. Intell.* **9**(5), 690–698 (September 1987).

7. H. Li, M. A. Lavin and R. J. L. Master, Fast Hough transform: A hierarchical approach, *Computer Vision Graphics Image Process.* **36**, 139–161 (1986).
8. S. Tsuji and F. Matsumoto, Detection of ellipses by a modified Hough transform, *IEEE Trans. Comput.* **27**(8), 777–781 (August 1978).
9. L. Xu and E. Oja, Randomized Hough transform (RHT): Basic mechanisms, algorithms, and computational complexities, *CVGIP: Image Understanding* **57**(2), 131–154 (1993).
10. J. H. Han, L. Koczy and T. Poston, Fuzzy Hough transform, *Pattern Recognition Lett.* **15**, 649–658 (July 1994).
11. W. Eric, L. Grimson and D. P. Huttenlocher, On the sensitivity of the Hough transform for object recognition, *IEEE Trans. Pattern Analysis Mach. Intell.* **12**(3), 255–274 (March 1990).
12. A. Kaufmann and M. M. Gupta, *Introduction to Fuzzy Arithmetic: Theory and Applications*. Van Nostrand Reinhold, New York (1985).
13. L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering*. John Wiley, New York (1996).
14. D. Dubois and H. Prade, *Fuzzy Sets and Systems, Theory and Applications*. Academic Press, New York (1980).

About the Author — VASSILIOS CHATZIS was born in Kavala, Greece, in 1968. He received the Diploma in Electrical Engineering from the Aristotle University of Thessaloniki, Greece, in 1991. He is currently a research and teaching assistant and studies towards the Ph.D. degree at the Department of Informatics, Aristotle University of Thessaloniki. His research interests include nonlinear image and signal processing and analysis, fuzzy logic and pattern recognition. Mr Chatzis is a member of the *Institute of Electrical and Electronics Engineers* (IEEE) and of the *Technical Chamber of Greece*.

About the Author — IOANNIS PITAS received the Diploma of Electrical Engineering in 1980 and the Ph.D. degree in Electrical Engineering in 1985, both from the University of Thessaloniki, Greece. Since 1994, he has been a Professor at the Department of Informatics, University of Thessaloniki. From 1980 to 1993 he served as Scientific Assistant, Lecturer, Assistant Professor, and Associate Professor in the Department of Electrical and Computer Engineering at the same University. He served as a Visiting Research Associate at the University of Toronto, Canada, University of Erlangen-Nuernberg, Germany, Tampere University of Technology, Finland, and as Visiting Assistant Professor at the University of Toronto. He was a lecturer in short courses for continuing education. His current interests are in the areas of digital image processing, multidimensional signal processing, and computer vision. He has published over 190 papers and contributed in eight books in his area of interest. He is the co-author of the book, "Nonlinear Digital Filters: Principles and Applications" (Kluwer, 1990) and author of "Digital Image Processing Algorithms" (Prentice-Hall, 1993). He is the editor of the book "Parallel Algorithms and Architectures for Digital Image Processing, Computer Vision and Neural Networks" (Wiley, 1993). Dr Pitas has been a member of the *European Community ESPRIT Parallel Action Committee*. He has also been an invited speaker and/or member of the program committee of several scientific conferences and workshops. He is an Associate Editor of the *IEEE Transactions on Circuits and Systems* and co-editor of *Multidimensional Systems and Signal Processing*. He was chair of the 1995 IEEE Workshop on Nonlinear Signal and Image Processing (NSIP95).