A New Approach to Sketch Recognition using Heuristic

G. Sahoo[†] and Bhupesh Kumar Singh[††]

[†]Department of Computer Science & Engineering
Birla Institute of Technology, Mesra, Ranchi, India
[††]Department of Computer Science & Engineering
Lingaya's Institute of Mgt. & Technology, Faridabad, India

Summary
The people of various domains, mechanical engineering for machine drawing, electronics engineering for circuit drawing, computer engineering for architectural design etc. generally use sketches. This paper presents an efficient approach for Sketch Recognition using Heuristic. Many papers have been presented in the literature to recognize sketches using various approaches. We discuss here multi-domain Sketch Recognition that provides flexibility to the user and increases the capabilities of a system. The Heuristic framework proposed in this paper offers the user more liberty for free-style sketching as well as for grouping the strokes, reducing the complexity for recognizing the sketches. It also solves the purpose of online as well as offline input sketch recognition. An attempt has been made here to remove the local search strategy and introduce a global search strategy to improve the recognition of the structure.

Key words:
Sketch recognition, Bayesian network, Heuristic, A Algorithm, Canny Algorithm, Gaussian Filter, Fuzzy-Membership Rule*

1. Introduction

This paper presents an efficient approach to deal with the problems of recognition of free-hand drawing or free-style sketch input given to the system using keyboard, mouse, digitizing tablet etc. Many papers have been presented in this regard, but the basic problem of efficiency, effectiveness and accuracy still persists. Sketch Recognition provides intelligence to the system in understanding the inputs given by the user so that system can recognize the input and respond as per the user's requirement. **Sketch understanding (SU)** [1] is a part of intelligence system which aimed at deriving the semantic knowledge from the sketch that helps user to convey ideas and guide over thinking process to make recognized problem more concrete. User may draw informal input, which produces inconsistent and ambiguous input to the system. So, a recognition engine should automatically adopt particular user styles. Recognition can be performed on the basis of two things: **Stroke based where** each stroke is recognized separately and **Feature based where we** make use of geometrical properties of the sketch [2].

We follow a recognition style based on both – strokes and geometrical features. The recognition can be performed over a single domain or on multiple domains.

i. **Single domain Recognition:** It employs the knowledge in only one domain. Yet it is simplistic but an inefficient approach because the person of different domain cannot use it.
ii. **Multi-Domain Recognition:** This recognition system incorporates the knowledge from multiple domains, so a single system can be used for various domains. Even, the system would find the result without changing the mode of the system [3], [4]. Input to this type of recognition system can be provided in the following two ways.
1. Online input (dynamic input): It means that system can be provided the input at run time while the system recognizes the strokes at the same time. As a natural tendency user draws the sketches with some pauses. The method to be used, without interfering the user, takes the input provided at the previous time instant and recognizes it at the next instant. After the user has given a long pause it shows the recognized figure to the user.
2. Offline input (Static input): User can draw a sketch on a paper and then feed it as input to the system. Our recognition engine will draw the output for this static input as well. The motive of our recognition system is to provide flexibility to the user as well as to represent the most efficient and accurate output. The paper is organized by introducing the concept of heuristic that removes the statistical approach used in Bayesian networks. Then, we explain the architecture of our system. Next, we explain the hierarchy of our recognition process that includes stroke filtering, sketch segmentation, elementary shape recognition, structure matching and semantic checking using heuristic and finally, the output, followed by future work and conclusion.

2. Recognition Framework

Bayesian Network uses local search strategy that makes an incremental changes aimed at improving the score of the structure [5]. A global search algorithm like heuristic

can avoid getting trapped in local minima. We select A* algorithm from a number of heuristic algorithms. Using heuristic approach we introduce how much resources like time and energy have been spent in recognizing a particular node from the start. This algorithm proves to be admissible i.e. the algorithm secures to find the most optimal way to recognize a stroke (node) [6]. Note that A* Algorithm evaluates nodes by combining the cost to reach the node and the cost to get from the node to the goal. If $g(x)$ denotes the cost of the path from START node to GOAL node x, and $h(x)$ defined to be the estimated cost of the cheapest path from x to the goal then the estimated cost of the cheapest solution through x can be given as

$$f(x) = g(x) + h(x)$$

To find the cheapest solution, we need to calculate the smallest $f(x)$ value. A* strategy is more than just reasonable provided that the heuristic function $h(x)$ satisfies certain conditions. A* Algorithm is both complete and optimal.

For tree-search, A* is optimal if $h(x)$ gives an admissible heuristic – that is, $h(x)$ must never overestimate the cost to reach the goal. Admissible heuristics is by nature optimistic. Since $g(x)$ provides the exact cost to reach x, so its immediate consequence is that $f(x)$ never overestimates the true cost of a solution through x.

Straight line proves to be admissible because the shortest path between any two points is a straight line, so the straight line can never be an overestimate.

To make a graph based searching optimal A* Algorithm imposes an extra requirement of consistency or monotonic. A heuristic function $h(x)$ is consistent if, for every node x and every successor s of x generated by an action a, the estimated cost of reaching the goal from x never greater than the step cost of getting s plus the estimated cost of reaching the goal from s, that is,

$$h(x) \leq c(x, a, s) + h(s).$$

This is a form of general triangle inequality, which stipulates that each side of a triangle cannot be longer than the sum of the other two sides. Here the triangle is formed by x, s, the goal closest to x. So, it represents that every consistent heuristic is also admissible.

If $h(x)$ is consistent, then the values of $f(x)$ along any path do not decrease. Suppose s is successor of x, then

$$g(s) = g(x) + c(x, a, s) \text{ for some } a$$

and

$$\begin{aligned} f(s) = g(s) + h(s) &= g(x) + c(x, a, s) + h(s) \\ &\geq g(x) + h(x) \\ &= f(x) \end{aligned}$$

It follows that sequence of nodes expanded by A* using graph-search is in non-decreasing order of $f(x)$. Hence, the first goal node selected for expansion must be an optimal solution, since all later nodes become at least as expensive. If C* be the cost of optimal solution path, then A* expands all nodes with $f(x) < C^*$ and no nodes with $f(x) > C^*$.

Further, A* algorithm ignores a sub tree that gives the solution while it still guarantees optimality. All above explanation shows that A* is **optimally efficient** for any heuristic function because any algorithm that does not expand all nodes with $f(x) < C^*$ run the risk of missing the optimal solution [6].

A* Inference Algorithm

A* maintains a set of partial solutions, i.e. paths through the graph starting at the start node, stored in a priority list. The priority assigned to a path x can, of course, be determined by the function

$$f(x) = g(x) + h(x)$$

where, $g(x)$ is the cost(weight of the edge) of the path so far and $h(x)$ is the heuristic estimate of the minimal cost to reach the goal from x. For example, if "cost" is taken to mean distance traveled, the straight-line distance between two points on a map is a heuristic estimate of the distance to be traveled. The lower $f(x)$ the higher the priority. We define the algorithm as

```
A*_function (START, GOAL)
{
if (START ==NULL || START==GOAL)
return;
a = remove_first(START);
if (a == GOAL)
return success;
else
{
if(successors_a(a != NULL))
s = successors_a(a);
fitness_no_s = cost_s(s) + evaluation_s(s);
LIST1 = sort_s(fitness_no_s);
}
START = LIST1;
Continue;
}
return;
}
```

2.2 Complexity

The time complexity of A* algorithm depends on the heuristic. In worst case, the number of nodes expanded is exponential w.r.t. the length of the solution (the shortest

path), but it is polynomial when the heuristic function *h* meets the following condition

$$| h(x) – h^*(x) | \leq O(\log h^*(x)).$$

where, $h^*(x)$ is optimal heuristic, i.e. the true cost of getting the required distance from x to the goal. In other words, the error of *h* should not grow faster than the logarithm of the "perfect heuristic" h^* that returns the true distance from *x* to the goal.

2.3 Computation efficiency

Note that the computation time imposes no drawback on A*. As A* keeps all generated nodes in memory, it may run out of space long before it runs out of time. To overcome this problem we use memory bound heuristic search strategy of A* algorithm. To reduce the memory requirements for A* we adapt the iterative –deepening to the heuristic search context. Now, cutoff is selected as the $f – cost\ (g + h)$ rather than depth, at each iteration, the cutoff value is the smallest *f-cost* of any node that exceeded the cutoff in the previous node. It avoids the substantial overhead associated with keeping a sorted list of nodes.

3.System Framework

The Figure1 shows the framework of our recognized system. It basically contains three blocks: Domain Class block, input refining block and recognition engine block [3], [7]. Domain classes contain the description of a particular domain. So, a programmer needs only to interface his code with the desired domain. Thus, it simplifies the developer's job. Interface shows the translation scheme used between domain classes and recognition engine. It bridges the gap between the two.

Input Refining block takes raw input or a roughly designed figure and finally gives a noise-free, smooth and properly drawn figure. The main thing about this system is that it also displays the name of the recognized object.'

To explain the working of the recognition system, an example of a hut is taken. The user draws a rough sketch of a hut and then system refines it. The input provided by the input is filtered using Gaussian filter. Then this filtered image is segmented in order to classify the elementary shapes from it.

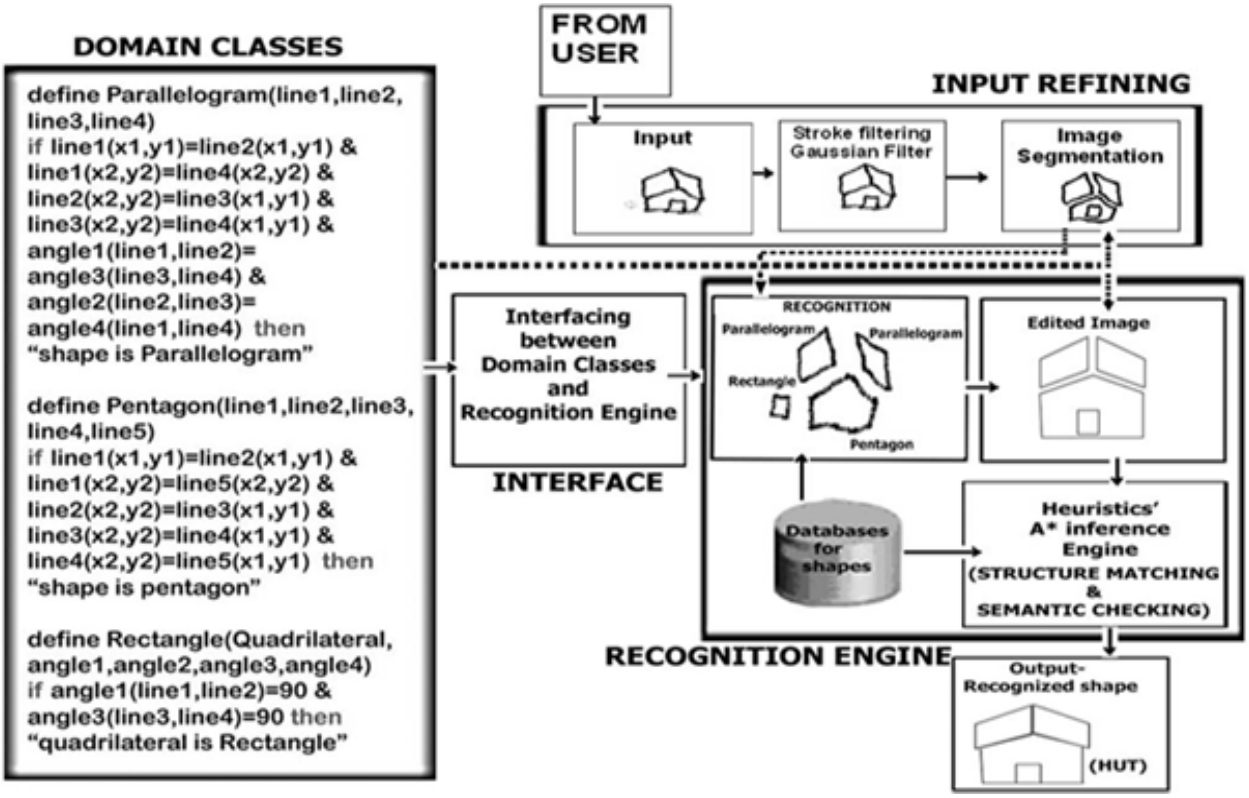


Figure1. System Framework

The system recognizes all segmented shapes using the domain description and databases of shapes. Database contains the sample images with recognized output. The system edits all the recognized shapes to display a smoothened figure [8].
The next most important task, the system performs to make a structure out of the given basic shapes. The Heuristics inference engine uses the Database set and takes the edited image as input and makes the complete structure and also performs semantic checking to clarify whether the structure shows some meaningful object.

4. Proposed Recognition Engine

The entire process of our proposed recognition engine is described by a flow chart (Figure2) given below and the modules involved are explained below.

4.1 Stroke filter Module

The input is fed to the system in the form of sketches and the sketches are represented in terms of single- stroke or multi-strokes. A stroke is a set of points from pen-down to pen-up. At this stage, we need to collect all such points and then use a filtering mechanism for removing the noise and smoothening the image (stroke). For this, we use Binary Division Algorithm to collect only relevant pixels (a smallest element of an image) from the whole captured screen.

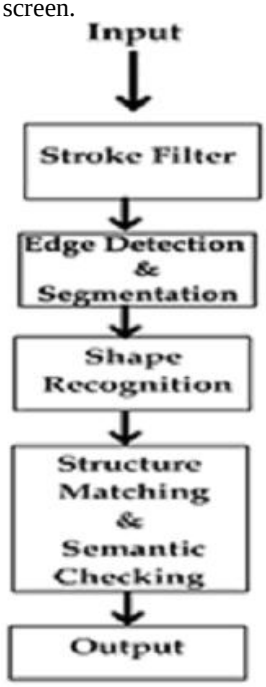


Figure2. Flow Chart of Recognition Engine

4.1.1 Binary Division Algorithm:

An image can be very small, it may take only 1/25th area of the screen or it may be too large to be displayed on the screen. In case, if the image of the sketch is too smaller, the processing of the whole area of the screen may reduce the efficiency of the system, so we need a method to determine only sketched area-pixels. The system uses an efficient algorithm namely, Binary Division Algorithm[9], which divides the screen area into two equal sub-areas. If one of the sub areas does not contain image it is simply rejected. Now area of consideration is only the half area of the screen. Now each sub area is divided in to two equal parts and this process continues until each smallest area contains single pixel information only. The whole methodology incorporated in this algorithm is depicted in Figure3,which is self explanatory.

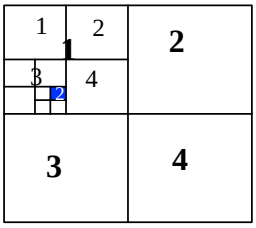


Figure3. Process of Division Algorithm

Next we use Gaussian filter as it rotationally symmetric [10].

4.1.2 Gaussian Filter

The Filter block filters the input signal using a Gaussian FIR filter. The filter requires the input signal to be up sampled, so that the input samples per symbol parameter, N, is at least 2. It requires the input bandwidth to be same as the bandwidth of the filter. Care should be taken while determining the frequency of the filter; it should be twice of the highest frequency of the input signal.

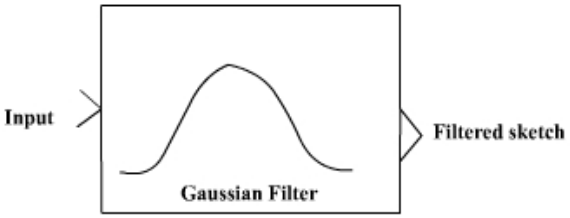


Figure4. Filtering Process

The block's icon shows the “filter's impulse response." The input signal must be a scalar or a frame-based column vector.

The input given to the filter is represented as *x(t)*, where *t* represents a particular time instant and *y(t)* denotes the output of the filter at time instant *t*. The impulse response of the Gaussian filter is given as

$$h(t) = \exp(-t^2 / 2\delta^2) / (\sqrt{(2\pi)} * \delta)$$

where, $\delta = (\sqrt{\ln(2)}) / 2\pi BT$ with B as the filter's 3-dB bandwidth. The group delay parameter *g* which is the number of symbol periods between the start of the filter's response and the peak of the filter's response and N together determine the length of the filter's impulse response, which is 2 * N * *g* + 1.
The output of the filter is calculated using following equation

$$y(t) = h(t) * x(t)$$

Apart from this, we use the concept of filter coefficient normalization, filter energy and peak amplitude in designing our system. It can be mentioned here that after the filter normalizes the set of filter coefficients, it multiplies all coefficients by the linear amplitude filter gain parameter.

4.2 Edge detection & Segmentation

The output from the Gaussian filter is taken and converted from RGB to grayscale thus; it reduces the memory requirements of the input image. The Canny edge detector takes single-valued images, also called monotone images, constant-value images, or flat images. Instead of issuing an error when an input image is single-valued, the edge function used with canny edge detection returns an output image containing all zeros, indicating that it found no edges. The Canny method finds edges by looking for local maxima of the gradient of Image. The gradient is calculated using the derivative of a Gaussian filter.
The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges(Figure5). This method is more robust to noise, and more likely to detect true weak edges.

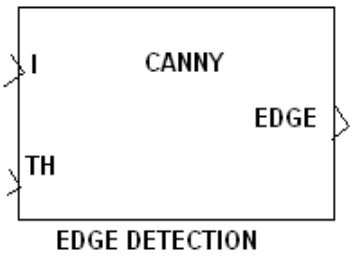


Figure5.

We define the following for the above purpose.

(i) BW = edge(I, 'canny') specifies the Canny method. where, *BW* represents the binary output.

(ii) BW = edge(I, 'canny', thresh) specifies sensitivity thresholds for the Canny method. *thresh* is a two-element vector in which the first element is the low threshold, and the second element is the high threshold. If we specify a scalar for thresh, this value can be used for the high threshold and 0.4*thresh may be used for the low threshold. If thresh is not specified, or if thresh is empty, edge function chooses low and high values automatically.

(iii)[BW, thresh] = edge(I, 'canny',...) returns the threshold values as a two-element vector.
Edge starts with the low sensitivity result and then grows it to include connected edge pixels from the high sensitivity result. This helps fill in gaps in the detected edges [10]. In all cases, the default threshold is chosen heuristically in a way that depends on the input data.
The edges are to be confirmed in to basic shapes. For this, the edges need to be segmented so that these shapes can be recognized into their well known geometric classes. Here, the system breaks the continuity of the edges while still maintaining the basic edge continuity so that it can be classified into the recognized shape. Before segmentation the system stores the continuity of the edges into the database.

4.3 Shape Recognition

After getting the edges, some mechanism is required to detect the elementary shapes like straight line, curve, circle, rectangle etc. For this, the recognition engine extracts the geometric features of the shapes to recognize the basic shapes.
A shape possesses many characteristics that can be used for matching the shapes, reorganizing the objects and for measuring the shapes. Geometric features define the geometric attributes of the shape. These are

a) Perimeter (T): It includes the total length of the shape; it can be circle, ellipse, quadrilateral etc.

$$T = \int \sqrt{\{ x^2(t) + y^2(t) \}}\, dt$$

where, *t* denotes the boundary parameter and *x* and *y* are axes coordinates.

b) Area(A): Denoting R as the object boundary region, we can find area as

$$A = \iint_R dx\, dy$$

c) Radius(r): This is the maximum distance from the centre of mass to the curve.

d) Height(H): Defines the vertical length of the shape

e) Width(W): Defines the horizontal distance of the shape.

f) Corners: These are the locations on the boundary where the curvature *k(t)* becomes unbounded and t represents the distance along the boundary

$$|k(t)|^2 = (d^2y/dt^2)^2 + (d^2x/dt^2)^2$$

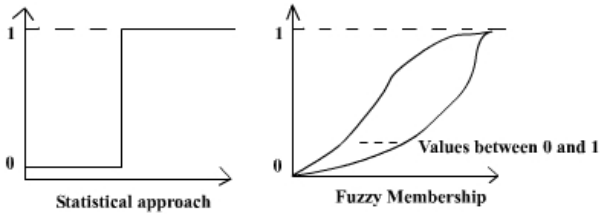
g) Roundness or compactness: This is defined as

$$\gamma = (perimeter)^2 / (4\pi\ (Area))$$

Heuristic is used in recognizing ambiguous shapes. User may draw overlapped shapes, intersected shapes or one shape within another shape (enclosed figures), so it requires some sort of treatment for discriminating such shapes. Here, we use a fuzzy-membership rule to classify the shapes drawn by the user.

4.3.1 Fuzzy Rule:

The sets for whom the boundary is ill defined are called as fuzzy-sets. To represent a fuzzy set a grade membership graph is used. All the members of a fuzzy set comprise of a membership value between 0 {complete non-membership} and 1 {complete membership}. Following diagrams illustrate the required concept on this matter.



If a rectangle is in question, its percentile graph represents the highest and smallest possibility, using it the system may answer if the shape is a rectangle or a circle.

A fuzzy set allows us to represent the set membership as a possibility distribution [11].

To determine a straight line, we need to calculate its two parameters: *height* and *width.* Then we make a percentile graph based on height / width ratio. For straight lines, it results zero. For curved line it gives a value smaller than one. Similarly, to discriminate the circle, we use its perimeter (P) and area (A), then calculate P^2/A ratio. Thus, the inference is based on these percentile graphs. The shape with maximum value is identified as the recognized shape.

4.4 Structure Matching and Semantic Checking

This stage uses a number of sample states to determine the best possible output structure. The system is trained in such a way that it can be used to detect unknown input with higher efficiency. For example, if a user has drawn a hut using various straight lines oriented in different directions, the system is trained to recognize a line, based on various sample states, the system recognizes that finally, the connected structure gives a recognized figure - hut.

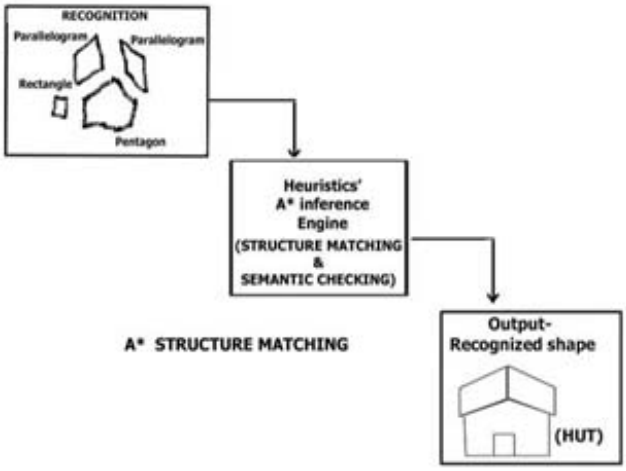


Figure6. A* Structure Matching

A set of rules or a grammar is used to make a structure using the detected shapes in the sketch recognition phase [12], [13]. This uses the tree structure and uses A* algorithm to determine the semantics of the structure (Figure6). The root of the tree is undetected structure and then, we progress in the downward direction to determine the possibility of the next node (the basic geometric shape) using fitness function (heuristic function) and cost function. The node that costs minimum value is used and further expanded.

Then, the next least expensive node is selected. This way, the structure is matched to a complete set of pattern and recognizes the figure.

Even after the sketch is recognized as a complete structure still it requires whether the interpretation done by the system is correct or not. So, the system also performs semantic checking to check if the recognized object makes a real meaning.

5. Conclusion and Future Work

In this paper we have presented a solution for the problem of local maxima incurred in Bayesian Network. The main concept is to use heuristic engine to detect the structures imposed by the user [14], [15]. The system first recognizes the basic shapes and then integrates them to form a complete figure. The structure matching is done using Heuristic approach, which gives the best result if it is found; otherwise it is unable to determine

the true structure. So, care should be taken while designing the Heuristic engine. The basic techniques discussed here can be used in the design of a recognition system with a robust architecture. We provide here the complete information for the implementation of a highly efficient, accurate and robust system. It definitely improves the flexibility to the user, efficiency of the system and flexibility to the developer. This paper promises never to be failed at any stage of the recognition.
Based on the information provided in this paper, an efficient recognition system can further be developed in future that may solve the purpose of recognition of complex shapes.

Acknowledgments

The author would like to express their cordial thanks to anonymous referees for their valuable suggestions

6. References:

[1] Davis. R., "Sketch understanding in design: Overview of work at the MIT AI lab.". Sketch Understanding, Papers from the 2002 AAAI Spring Symposium, pages 24{31, March 25-27 2002.

[2] Fonseca. M., Pimetal. C., "CALI: an online scribble recognizer for calligraphic interface". Proceedings of AAAI Spring Symposium on sketch Understanding, Stanford University, PP.51-58, March 2002.

[3] Alvarado. C. and Oltmans. M., "A Framework for multi-domain sketch recognition". Proceedings of AAAI Spring Symposium on sketch Understandings, Stanford University, pp.1-8, March 2002.

[4] Alvarado. C., "Multi-domian sketch understanding"' PhD thesis, Department of Electrical Engineering and computer science, Massachusetts Institute of Technology, September 2004.

[5] Liao. S.Z., Wang X.J., Liang. J. "An incremental approach to sketch recognition". Proceedings of the 4th International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.

[6] Russell, S.J. and Norvig, P. 1995, "Artificial Intelligence: A Modern Approach". Englewood Cliffs, NJ Pretice Hall.

[7] Alvarado C, Davis R., "Sketchread: a multi-domain sketch recognition engine." In: Proceedings of UIST '04, 2004. p. 23–32.

[8] Bimber. O., Encarnao. L.M., and Stork. A., "A multi-layered architecture for sketch-based interaction within virtual environments." Computer and Graphics, 2000.

[9] Foley, J.D., Dam, A.V., Feiner S.K. and Hughes, J.F., 1999. "Computer Graphics- Principles and Practice",, Pearson Education.

[10] Jain, A.K. 1989, "Fundamental of Digital Image Processing", Prentice Hall.

[11] Fonseca. M., Jorge J., "Using Fuzzy logic to recognize geometric shapes interactively" Proceedings of the 9th IEEE conference on Fuzzy systems, pp. 291-296, 2000.

[12] Hammond T., "A Domain Description Language for Sketch Recognition". MIT Artificial Intelligence Laboratory 200 Technology Square, NE43-809, Cambridge, MA 02141 March 19, 2003.

[13] Hammond. T.M., Davis. R., "LADDER, a sketching language for user interface developers", Computers & Graphics 29 (2005) 518–532.

[14] Chien C.F., "Modifying the inconsistency of Bayesian networks and a comparison study for fault location on electricity distribution feeders", Department of Industrial Engineering and Engineering Management", Int. J. Operational Research, Vol. 1, Nos. ½, pp. 188-202, 2005.

[15] Rish. I., "An empirical study of the naïve Bayes classifier", IBM Technical Report RC22230, 2001.

G. Sahoo received his P.G degree from Utkal University in the year 1980 and Ph.D degree in the area of Computational Mathematics from Indian Institute of Technology, Kharagpur in the year 1987. He is associated with Birla Institute of Technology, Mesra, Ranchi, India since 1988. He is currently working as a professor and heading the Department of Computer Science and Engineering. His research interest includes theoretical computer science, parallel and distributed computing, image processing and pattern recognition.

Bhupesh Kumar Singh received B.E. in (Computer Science & Engineering) from Government College of Engineering. Tirunelveli, Anna University, India in the year of 1999, and doing Ph. D in Sketch Recognition from BIT Mesra, Ranchi. Currently working as Senior (Selection Grade) in LIMAT, Faridabad, India, also life membership of International Association of Engineers (IAENG)