ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Paper / Case Study
Available online at: www.ijarcsms.com

Mahi Translator for Sketch Description

Dr. Bhupesh Kumar[1]
Professor
Faridabad
Haryana - India

Parveen Mor[2]
Assistant Professor
Faridabad
Haryana - India

Abstract: ***Diagrams are essential means of capturing and communicating information in many different domains. They are also a valuable part of the early design process, helping us to explore ideas and solutions in an informal environment. There is increasing interest in building systems that can automatically interpret freehand drawings (sketch). Sketch recognition systems are currently being developed for many domains, but can be time-consuming to build if they are to handle the intricacies of each domain. This paper presents an efficient that takes shape descriptions from MAHI domain description for sketching language and automatically interprets them into shape recognizers, shape editor, and shape exhibitors for use in conjunction with a domain independent sketch recognition system. This transformation allows us to build a single domain independent recognition system that can be adapted for multiple domains. Our framework has been tested and implemented by writing multiple domain descriptions and created a domain specific sketch recognition system for each domain.***

Keywords: Control Flow Technique, MAHI, MAHII, Heuristic, Indexing, Smart-board.

I. INTRODUCTION

People often sketch when solving problems. Some sketches are personal, others are collaborative. Some sketches help people make quick calculations and are quickly forgotten, others serve longer-term purposes. For professional designers, sketching serves as a means for thinking about problems as much as it does for communicating proposed solutions. For people who are not designers, sketching is a natural means for quick recording of spatial information such as directions to a point of interest. Design can be seen as an iterative process of problem-framing and exploring possible solutions within the current conception of the problem. Sketching allows people to visually represent ideas quickly, without prematurely committing to decisions. A sketch is a proposal that can be modified, erased, built upon. The rough look of hand-made sketches suggests their provisional nature. We are interested in enabling a generic sketch recognition system that would allow more natural interaction with design tools in various domains. Sketch recognition is an important part of sketch-based system and it needs domain information to classify and recognize the sketch.

Over the years, sketch recognition systems on pen-based input devices and hand-drawn diagrammatic domains are discussed in details. As pen-based input devices have become more popular, sketch recognition systems are being developed for many domains. Alvarado (2000) has discussed on domain description for mechanical engineering, Hammond and Davis (2002) for UML class diagrams, Lin et al. (2000) for webpage design, Gross et al. (1994) for architectural design, Caetano et al. (2002) for GUI design, Do (2001) for virtual reality, Mahoney and Fromherz (2002) for stick figures, Pittman et al. (1996) for course of action diagrams , and many others. These systems allow users to sketch a design, which is a more naturally interaction than a traditional mouse and palette tool has been discussed by Hse et al. (1999). But key issue in sketch recognition systems can be quite time consuming to build if they are to handle the intricacies of each domain. We propose that rather than building a separate recognition system for each domain, instead build a single domain independent recognition system that can be customized for multiple domain. To build a sketch recognition system for a new domain, the developer would need to write only a domain description, describing how sketch is drawn, displayed and edited. This description would then be interpreted for

use in the domain independent system. The inspiration for such a framework stems from work in speech recognition discussed by Zue and Glass (2000) and sketch recognition (LADDER) by Hammond and Davis (2005).

Our goal is to transform a grammar into a domain recognizer of hand-drawn shapes. The translation process is analogous to work done on complier compilers, visual language complier compilers developed by Costagliola et al. (1995). A visual language compiler allows a user to satisfy a grammar for a visual language, and then compiles it into a recognizer that can indicate whether an arrangement of icons is syntactically valid. Hammond and Davis have developed the system for hand drawn images and geometric shapes, and their primitives are geometric. In continuation to this Hammond and Davis developed automatically transforming symbolic shapes for sketch recognition. Our work differs from Costagliola et al. related to iconic shapes and is in the same vein as Hammond and Davis based on geometric based primitive. However our system is related to drawing hand-drawn sketches supplemented with mathematical rule associated to geometry as an agent.

In this paper we present the translator that takes shape descriptions of how sketch is drawn, displayed, and edited in a domain and automatically interpret them into shape recognizers, shape editor, and shape exhibitors for use in a domain independent sketch recognition system. To achieve our goal, we introduced a new approach to sketch recognition using Heuristic. It compares very well with the statistical approach used in Bayesian networks as discussed by Chien (2005). Then we introduced its interface MAHII (Machine and Human Interactive Interface) followed by a sketching language based on geometrical rules associated to the recognition of basic elements in its domain. The recognition engine interprets the sketches based on the information obtained from the description of domains associated with the input from the user. The implementation of this translator and domain independent sketch recognition system serves to show that such a framework is feasible and that MAHI is an acceptable language for describing domain.

We did in same vein as Hammond and Davis (2004) discussed a symbolic sketching language based on how shapes look rather than on features such as drawing speed, size of the bounding box, etc., as in systems discussed by Rubine (1991),and Long (2001). To ensure that symbols would be recognized if they looked the same, even if they weren't drawn in the same way (e.g. using different number of strokes). However our system not only number of strokes but also order of stroke and allowing users to draw the shapes as they would naturally. As Discussed by Hammond and Davis (2004) a shape definition consists of how shapes look alongwith other information helpful to the recognition process, such as stroke order or stroke direction. Because different domains have different ways of displaying and editing the shapes in their domain, whereas recognition systems need to know how to edit and display the shapes recognized.

Shape description languages have been created for use in architecture, diagram parsing, as well as within the field of sketch recognition itself. However, current shape description languages lack ways for describing editing behavior discussed by Stiny and Gips (1972), Futrelle and Nikolakis (1995), Bimber et al. (2000), Mahoney and Frommerz (2002), Caetano et al. (2002b) and Gross and Do (1996), display discussed by Futrelle and Nikolakis 1995, Bimber et al. (2000), Mahoney and Frommerz (2002), Caetano et al. (2002b), and Gross and Do (1996), or non-graphical information, such as stroke order or direction discussed by Stiny and Gips (1972), Futrelle and Nikolakis (1995) and Bimber et al. (2000).

This paper is organized as follows. We discuss Framework of system in section 2 and Translator in Section 3. Section 4 deals with testing and finally section 5 gives conclusion and further work.

II. FRAMEWORK OVERVIEW

We are able to draw any of the sketches based on entity BCE (Basic Core Entity), CE (Core Entity), and DE (Derived Entity) as defined by point, line, arc, rectangle, square, circle, ellipse, polygon, curves, surfaces, volume etc. Here a sketch is a geometrical structure with basic input as geometrical rules associated to recognition process such as the stroke. Using the sketch properties, an arbitrary sketch is extended or reduced to the desired sketch. An application designer can redefine the properties as and when he needs. The language has proven to be very powerful and highly interactive for multi domains. The language

enables more accurate and fast sketch recognition by using bottom-up as well as top-down recognition in view of mathematical definition of line, which in domain description as 'a set of adjacent pixels, such that lines drawn by joining any 2 or more pixels in a set have same slope' analogously, corresponding to MAHI language, it can be stated as follows as

Components

Pixel p1

Pixel p2

….

….

Pixel p(n)

Clause//constraints

equalSlope p1p2

equalSlope p2p3

….

equalSlope p(n-1) p(n)

Thus, a point is recognized and is used to identify and be identified as a line or arc since the line or arc is extended to form any other extended sketch(s) or extended sketch(s) merged to a line or curve. We did this to ensure that sketch would be recognized if they look the same, even if they are not drawn in the same way (like with a different number of strokes), allowing users to draw the sketches as they draw naturally. Sketch definitions primarily concern how sketches look, but may include other information helpful to the recognition process, such as stroke sequence or stroke orientation. Because different domains have different ways of displaying and editing the sketches in their domain, sketch recognition systems need to know how to edit and display the sketches. We developed a sketch recognition system easier by enabling domain experts rather than programmers to describe the shapes to be recognized. An overview of the framework for our overall research effort: (1) a sketching language (MAHI), (2) a translator that converts a domain description into components for use in conjunction with a domain independent sketch recognition system, and (3) a domain independent sketch recognition system that uses the newly generated components to recognize, edit, and display shapes in the domain. The domain description is transformed into shape recognizers, exhibitors, and editors which are used in conjunction with a domain independent recognition system to create a domain specific recognition system.

MAHI follows a very unique and robust structure for sketch recognition (Figure 1), which enables users to go beyond the limited predefined database and recognizes their sketches even if they are new to database.

Basically MAHI is composed of three modules namely a) Debugger, b) Database cum Translator and c) Recognizer.

The sketch is broken into small segments, basically small lines or arcs, using control flow segmentation technique. With the help of recognizer each segment is identified as line or arc, and is refined by the debugger, by auto-editing in same proportions. The combination of refined lines and arcs are recognized using Indexing & Heuristic Algorithm followed by top-down approach.

The sketches drawn are identified and redrawn properly, then with the help of CRO module (i.e. combination & relative orientation module), it's checked if any super-sketch exists, if yes then the super-sketch ID becomes the drawn sketch ID, else user is asked to name a super-sketch or accept it as it is. A small module, which is part of recognizer, is there to solve any domain ambiguity at the end of recognition, by linearly analyzing the dominant domain, and taking decision as per user desire.

To create the domain specific recognition system, the developer writes a MAHI domain description consisting of multiple sketch definitions. For example, refer to quadrilateral sketch definition. The components and the constraints define what the sketch looks like and are transformed into shape recognizers. The display section specifies how the sketch is to be displayed when recognized and is transformed into shape exhibitors. The editing section specifies the editing behaviors that can be performed on the recognized sketch and are transformed into shape editor.

A shape is defined if it is associated to a domain belonging to domain description. In particular, geometrical shapes are defined as the blocks belonging to the domain shapes of multi-domain. Now using the above definition alongwith the basic entity as defined by BCE, CE, DE , constraints, display methods, and editing behaviors in MAHI, These predefined elements are hand-coded into the domain independent system, allowing it to recognize, display, and edit these derived shapes. The combination of refined lines and arcs are recognized using Indexing & Heuristic Algorithm following a top-down approach, whereas Hammond and Davis (2004) discussed bottom-up approach.

III. TRANSLATOR

The domain description MAHI is translated to produce recognition code, generating one Jess rule (recognition information), and one java file (shape description), for each shape description. The system uses the Jess rules to recognize sketches. Following is the automatically generated Jess rule for quadrilateral description of Figure 1.

```
(defrule QuadilateralCheck
;; get four lines
?f0 $<$- (Subshapes Line ?side1 \$?side1_list)
?f1 $<$- (Subshapes Line ?side2 \$?side2_list)
?f2 $<$- (Subshapes Line ?side3 \$?side3_list)
?f3 $<$- (Subshapes Line ?side4 \$?side4_list)
;; make sure lines are unique
(test (uniquefields \$?side1_list \$?side1_list))
(test (uniquefields \$?side2_list \$?side2_list))
(test (uniquefields \$?side3_list \$?side3_list))
(test (uniquefields \$?side4_list \$?side4_list))
;; get accessible components of each line
(Line ?side1 ?side1_p1 ?side1_p2 ?side1_midpoint ?side1_length)
(Line ?side2 ?side2_p1 ?side2_p2 ?side2_midpoint ?side2_length)
(Line ?side3 ?side3_p1 ?side3_p2 ?side3_midpoint ?side3_length)
(Line ?side4 ?side4_p1 ?side4_p2 ?side4_midpoint ?side4_length)
;; test constraints
(test (coincident ?side1_p1 ?side2_p1))
(test (coincident ?side2_p2 ?side3_p2))
(test (coincident ?side3_p3 ?side4_p3))
(test (coincident ?side4_p3 ?side1_p3))
;;deleted code: get line with endpoints swapped
=> ;; FOUND QUADILATERAL (ACTION TO BE PERFORMED)
;; add quadilateral to sketch recognition system to be displayed properly
(bind ?nextnum (addshape Quadilateral ?side1 ?side2 ?side3 ?side4))
;; add quadilateral to Jess fact database
(assert (Quadilateral ?nextnum ?side1 ?side2 ?side3 ?side4))
(assert (Subshapes Quadilateral ?nextnum (union\$ \$?side1_list
\$?side2_list \$?side3_list \$?side4_list)))
(assert (DomainShape Quadilateral ?nextnum (time)))
;; remove Lines from Jess fact database for efficiency
(retract ?f0) (assert (CompleteSubshapes Line ?side1 \$?side1_list))
(retract ?f1) (assert (CompleteSubshapes Line ?side2 \$?side2_list))
(retract ?f2) (assert (CompleteSubshapes Line ?side3 \$?side3_list))
(retract ?f3) (assert (CompleteSubshapes Line ?side4 \$?side4_list))
;;deleted code: retract line with endpoints swapped
)
```

Translation process parses the description and generates code specifying how to recognize shapes and editing as well as how to display the shapes once they are recognized and what action to perform once an editing trigger occurs. We describe the translation process in detail for each part of the shape definition

1.1 Shape Recognizers

The work of the shape translator is to transform a shape definition into set of rules that recognize that shape. The shape definition specifies the components that make up the shape as well as the constraints on these components, including any requirements about stroke order or direction. We use the Jess rule engine for recognition as discussed by Friedman (2001). Jess is a forward-chaining pattern/action rule engine for Java. In our system, Jess facts represent recognized drawn shapes. Each stroke is segmented into a BCE, CE, and DE like point, line, arc, curve, or some combination using techniques discussed by Sezgin (2001), and the primitive shapes are added to the Jess fact database. For instance, if a quadrilateral is drawn using four lines, a fact is added of the form (Line 246 11 12 13 5.6), where 246 indicates the line's ID, 11, 12, and 13 are the IDs of the endpoints and midpoint, and 5.6 indicates the length of the line. Because we do not want to place any unspecified drawing order constraints, each line, arc, and curve is actually added twice to the Jess rule based system to take into account the fact that the endpoints may be assigned in either direction. An additional fact, (Line 246 246 246), is also sent to indicate the primitive shapes that make up the drawn shape. Our domain shape recognizers are implemented as Jess rules. In the transformation process we create a rule whose pattern specifies the components of the shape and the constraints that must hold between the components. The pattern also specifies that all components be distinct, to prevent the rule engine from returning four copies of the same line when trying to find a quadrilateral. An example of the rule generated for the quadrilateral given in Figure1 is shown above. The translation process is straightforward because of the foundation of primitives built into the domain independent system. The rule engine searches for all possible subsets of facts for the collection specified in its premise. In case of Figure 1, the rule engine searches for four lines to make up the side1, side2, side3, and side4. The rule engine then tests whether the constraints hold for each subset.

Then, for each collection of four lines marked side1, side2, side3, and side 4, the Jess engine will check that side1.p1 and side2.p1 are coincident. Every MAHI constraint is defined as a Jess function to simplify transformation of a shape definition into a Jess rule. If a shape is recognized (i.e., the rule pattern is satisfied), the rule action is fired. In the transformation process the rule action is specified to add the new shape to the database of recognized shapes so it can be displayed correctly and edited, add the new fact to the rule based system so more complicated shapes can be formed from it and remove recognized components from the database. Removing recognized components from database improves efficiency at the expense of possibly missing some shape interpretations.

Jess then confirms that this newly created shape does not share any subcomponents with any other domain shape. If it does, then only one of the domain shapes will remain, either the shape containing more primitive components (Ockham's razor), or (in the case of a tie) the shape that was drawn first. While the quadrilateral example used throughout this paper is composed of a fixed number of components (4 lines), our architecture can also support sketches composed of a variable number of components, such as a polyline or polygon. A sketch with a variable number of components is transformed into two Jess rules, the first recognizes the base case, and the second rule handles the recursive case. For example, the base case rule of a polyline recognizes a polyline consisting of two lines, while the recursive case rule recognizes a polyline consisting of an existing polyline and an additional line.


```
Edit ( <number of edges or segments 'n'> and <information about each segment>)
        ( for i = 1 to n )
        (
                trigger DoubleClickHoldDrag side 'i'
                action translate this setCursor DRAG
                showHandle MOVE side 'i'          // i  is the number of sides.
                trigger holdDrag side 'i'
                action translate this setCursor DRAG
                showHandle MOVE side'i' … side'n'
        )
```

Use of Editing (curser) in MAHI is dependent on its mode that, one can use pen based editing that is when the curser is in pen mode (sketching) the user can draw sketches and

When it is in curser mode it means the editing mode. Thus, sketching and editing use distinct pen motions. One of the editing behaviors in MAHI is if one clicks and holds the pen on the surface of the quadrilateral and drags the pen, the entire quadrilateral will translate along with the movement of the sketch. In this way the quadrilateral along with the vector is translated, scaled, and rotated as one whole shape. All of editing behaviors also change the pen cursor as display to the sketcher to know of his performing an editing command. MAHI editing behavior of triggers include click, double click, hold, hold drag, encircle, etc. along with their subsequent application. Editing gestures permit us to recognize the system to differentiate between sketching and editing. MAHII language includes a number of predefined editing behaviors using the algebraic constraints associated to the geometry of the figures such as a point, line, arc, etc. The possible editing action include wait, select, deselect, delete, translate, scale, IsRotate etc.

Editing behaviors specifies the gesture of dragging the component in any direction. In this case, the actions of these editing commands specify that the object should follow the pen at the description of the user including translating and rotating. Any arbitrary quadrilateral can be changed to a rectangle as per the action of the user, resulting in the change of coefficient of x and coefficient of y in the domain.

1.2 Shape Editor

MAHI knows when and how to recognize, and edit the shapes. Editing is the important components of sketch interface that vary as per different domains. The advantage in MAHI is that a user is encouraged to standardize different domains by including some defined editing behavior. In fact, one can define one's own editing behaviors for each domain. For example; using the same gesture such as drawing a point inside a circle may be intended as a check in one domain(check box) and a full stop in the other domain (text box) or as an editing command.

The sketch definition includes information on how a sketch can be edited. Each editing behavior consists of a trigger and an action. For instance; the quadrilateral definition specifies editing behaviors like dragging the side1, dragging the side2, dragging the side3, dragging the side4 or dragging the entire quadrilateral. Each of the four defined editing commands are triggered when the user places and holds their pen on the side1, side2, side3, side4 and then begins to drag the pen. The actions for these editing commands specify that the object should follow the pen either in a rubber band fashion for side1, side2, side3 and side4 of the quadrilateral or by translating the entire quadrilateral.

The drawing panel watches for all of the possible editing triggers predefined in MAHII for MAHI. When one occurs it calls the appropriate method to check if an editing behavior should occur. Each of the editing actions (such as translate, rotate, resize, rubber-band, delete, cut, or paste) is predefined for all shapes. During the translation process, we transform all of the editing specifications of the sketch definitions into one Java class. After a trigger first occurs (such as click or holdDrag), the Java class examines all of the viewable sketches with that trigger defined to see if an editing behavior has begun. For instance, holdDrag is defined as the pen initially resting on the screen for .6 seconds, and then dragging across the screen. After holdDrag is detected, the system looks to see if the pen is located over the side1, side2, side3 and side4 of a quadrilateral. If not, the system treats the

stroke as a drawing gesture. If there is a sketch underneath the stylus that has that trigger specified, the editing action occurs. In our example, if the side1 of a quadrilateral is underneath the pen, the quadrilateral will rubber-band (Rubber-banding allows users to simultaneously rotate and scale and object, assuming a fixed rotation point is defined. This action has proved useful for editing quadrilateral and other linking shapes) with the side1, following the path of the pen and other sides remaining fixed.

1.3 Shape Exhibitors

Controlling is a vital part of sketching interface when the user recognizes the shape after he has drawn that sketch. MAHI defines recognition layer that has the power of the recognition of the sketch. It gets the input from editing layer as defined above and uses heuristic and indexing algorithm to perform semantic checking for the relevant details from the system database. It also includes the concept of domain independence with reference to multi-domain. It recognizes the output and displays the results to the input/output layer of MHAII interface.

Display methods indicate what to display when the object is recognized. The sketching shape follows the original component followed by the constraints associated to the other related component to give rise to the extended shape such as line, circle, rectangle, etc. to give the final shape.

MAHI has a number of predefined display methods, including color, original-strokes, and cleaned-strokes, which are hand-coded into the domain independent recognition system for use with all shapes. The domain independent recognition system also defines how to handle hierarchical display of shape, and provides generalized methods for altering the display of a shape, including translates, scale, and rotate. The sketch definition includes information on how a sketch should be displayed once it is recognized. A recognized sketch or its components may be displayed in any predefined display methods of MAHI language. For instance; the following function

Display (<number of edges or segments ‘n’> and <information about each segment>)

cleaned-strokes side1, side2, side3

original-stroke side4

color blue

Specifies that the quadrilateral should be displayed in blue in color, that side1, side2, side3 should be drawn using cleaned-strokes (a straight line in this case), and that the side4 should be drawn using the original strokes in Figure2.

During the transformation process we create a shape exhibitor (a Java class file) to display the shape for each shape. This shape exhibitor specifies how the shape is to be drawn with original or cleaned. If the original strokes are to be drawn, the translator creates a method that displays the original strokes by attempting to solve the constraints of the shape (e.g., ensuring that points are coincident if the description indicates so) and then redraws the shape with the constraints solved. Likewise, if a cleaned stroke display is defined, the translator creates a function that displays the cleaned stroke. The shape exhibitor controls the displaying of the newly created shape and ensures that the components (e.g., the original strokes) are not drawn, but only the abstract shape (e.g., quadrilateral) itself. The shape exhibitor keeps track of the location of the accessible components and aliases of a shape, which 1) can be used by the editing module to determine if an editing gesture is occurring, and 2) ensures that when a shape is moved or edited its components are moved or deleted with it. The shape exhibitor also keeps an original copy of each of the accessible components and aliases for use when scaling an object to ensure that we don’t lose any precision after a number of scaling.

IV. TESTING

To test our approach we have built and transformed domain descriptions for multiple of domains like finite state machine, mechanical engineering, UML class diagrams. Figure 3, a finite state machine of the domain independent recognition system described in this paper, was created using the finite state machine domain of sketch recognition system. The domain description specified that states displayed in blue and the transaction in red, both using cleaned-strokes. It supports both on-line and offline recognition mode and can be applied to a variety of domains. We have also performed experimental evaluation of the system on the use mathematics and flowchart domain, and the Figure 4 shows results that have significantly reduced recognition error over a baseline system that did not consider contextual information.

V. CONCLUSION AND FURTHER WORK

We suggest a framework for sketch recognition that uses a single, customizable domain independent recognition system for multi domain. This paper presents the translator which takes descriptions of how sketches are drawn, displayed, and edited in a domain and automatically convert them into shape recognizers, shape editor, and shape exhibitors for use in a domain independent sketch recognition system. To achieve our goal, we created 1) MAHI (Machine and Human Interface, a language for describing how shapes are drawn, displayed and edited in a domain, 2) the translator described above, and 3) a simple domain independent recognition system that uses the newly translated components to recognize, display, and allow editing of the domain shapes. The implementation of this and domain independent sketch recognition system serves to show both that such a framework is feasible and that MAHI is an acceptable language for describing domain information. Further we would like to improve our and domain independent recognition system to include the handling of soft constraints and continue to test it on additional domains. While we are attempting to make MAHI as intuitive as possible, shape definitions can be difficult to describe textually. However, even automatically generated shapes will need to be checked and modified. Thus we would like to create a Graphical User Interface to debug a domain description, providing interfaces to test whether the shape is under-constrained or over constrained.

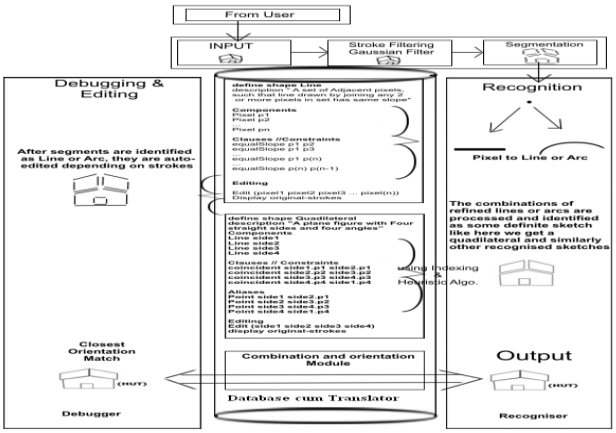

Figure 1: Framework Overview showing MAHI Domain Description and Domain Independent Sketch Recognition System.

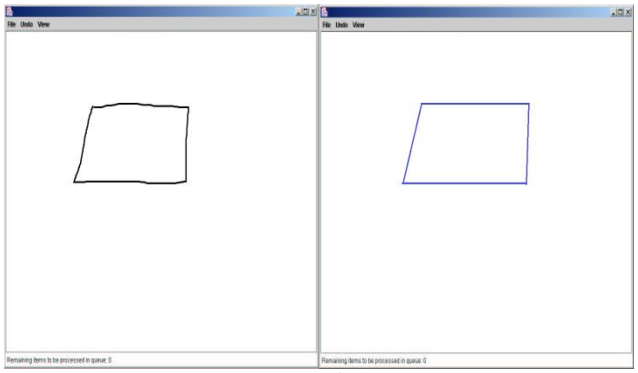

Figure 2: Shape generated from MAHI domain description using translator. Left side represents the original strokes. Right side represents the cleaned strokes.

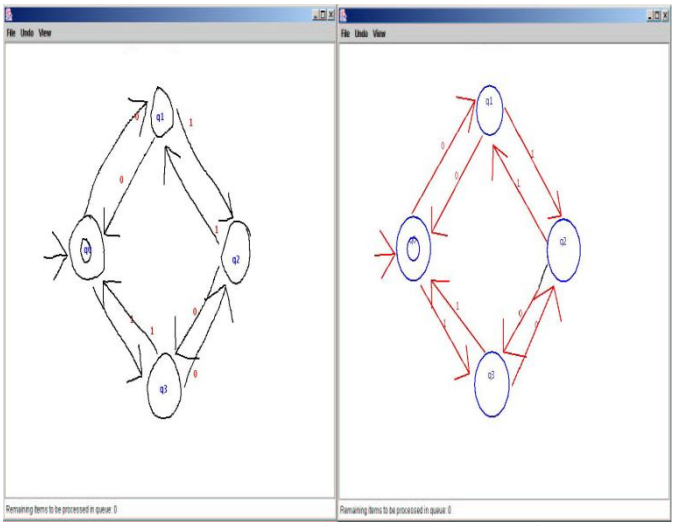

Figure 3: Auto-generated finite state machine interface.

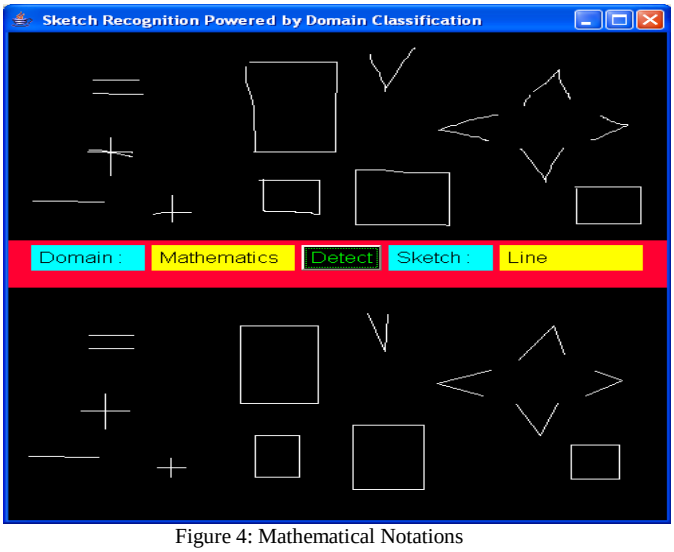

Figure 4: Mathematical Notations

References

1. Alvarado, C. (2000) 'A natural sketching environment: Bringing the computer into early stages of mechanical design' Master's thesis, MIT
2. Bimber, O.; Encarnacao, L. M.; and Stork, A. (2000) 'A multilayered architecture for sketch-based interaction within virtual Environments' Computer and Graphics: Special Issue on Calligraphic Interfaces: Towards a New Generation of Interactive Systems 24(6), PP. 851–867.
3. Caetano, A.; Goulart, N.; Fonseca, M.; and Jorge, J. (2002b) 'Sketching user interfaces with visual patterns' Proceedings of the 1st Ibero-American Symposium in Computer Graphics (SIACG02) PP. 271–279.
4. Chien C.F.,(2005) 'Modifying the inconsistency of Bayesian networks and a comparison study for fault location on electricity distribution feeders' International Journal Operational Research, Vol. 1, Nos. ½, pp. 188-202, 2005.
5. Costagliola, G.; Tortora, G.; Orefice, S.; and Lucia, D. (1995) 'Automatic generation of visual programming environments' In IEEE Computer, 56–65.
6. Do, E. Y.-L. (2001) 'Vr sketchpad - create instant 3d worlds by sketching on a transparent window' CAAD Futures 2001,Bauke de Vries, Jos P. van Leeuwen, Henri H. Achten (eds) 161–172.
7. Gross, M.; Zimring, C.; and Do, E. (1994) 'Using diagrams to access a case library of architectural designs' In Gero, J., and Sudweeks,F., eds., Artificial Intelligence in Design '94. Netherlands: Kluwer Academic Publishers. 129–144.
8. Hammond, T., and Davis, R. 2002. Tahuti: A geometrical sketch recognition system for uml class diagrams. AAAI Spring Symposium on Sketch Understanding 59–68.
9. Hammond, T., and Davis, R.,(2005) 'LADDER, a sketching language for user interface developers' In Computer &Graphics-UK
10. Hammond, T., and Davis, R., (2004) 'Automatically Transforming symbolic shape for use in sketch recognition' Proceeding of the19th National Conference on Artificial Intelligence(AAAI-04)
11. Hse, H.; Shilman, M.; Newton, A. R.; and Landay, J. (1999) 'Sketch-based user interfaces for collaborative object-oriented modeling' Berkley CS260 Class Project.
12. Jacob, R. J. K.; Deligiannidis, L.; and Morrison, S. (1999) 'A software model and specification language for non-WIMP= user interfaces' ACM Transactions on Computer-Human Interaction 6(1):1–46.

13. Lin, J.; Newman, M.W.; Hong, J. I.; and Landay, J. (2000) 'Denim: Finding a tighter fit between tools and practice for web site design' In CHI Letters: Human Factors in Computing Systems, CHI2000, 510–517.
14. Long, A. C. (2001) 'Quill: a Gesture Design Tool for Pen-based User Interfaces' Eecs department, computer science division, U.C. Berkeley, Berkeley, California.
15. Mahoney, J. V., and Fromherz, M. P. J. (2002) 'Handling ambiguity in constraint-based recognition of stick Fig sketches' SPIE Document Recognition and Retrieval IX Conf., San Jose, CA.
16. Pittman, J.; Smith, I.; Cohen, P.; Oviatt, S.; and Yang, T. (1996) 'Quickset: A multimodal interface for military simulations' Proceedings of the 6th Conference on Computer-Generated Forces and Behavioral Representation 217–224.
17. Rubine, D. (1991) 'Specifying gestures by example' In Computer Graphics, volume 25(4), 329–337.
18. Sezgin, T. M. (2001) 'Feature point detection and curve approximation for early processing in sketch recognition' Master's thesis, Massachusetts Institute of Technology.
19. Sahoo. G., and Singh, B.K, (2008) 'A New Approach to Sketch Recognition using Heuristic' International Journal of Computer Science and Network Security, Volume 8, Issue 2, PP. 102-108.
20. Sahoo. G., Singh, B.K., and Raina, B. L.,(2009) 'MAHI: Machine And Human Interface' International Journal of Image Processing (IJIP), Volume 3, Issue 2, PP. 80-91.
21. Sahoo. G., and Singh, B.K.,(2008) 'MAHII: Machine And Human Interactive Interface' International Journal of Image Processing, Volume 2, Issue 3, PP. 1-11.
22. Sahoo. G., and Singh, B.K., (2008) 'A Human Detector and Identification System' International Journal of Computer Science, Systems Engineering and Information Technology, volume 1, Issue 1, PP. 39-44.
23. Veselova, O. (2003) 'Perceptually based learning of shape descriptions .Master's thesis' Massachusetts Institute of Technology, Cambridge, MA. Zue, and Glass. (2000) 'Conversational interfaces: Advances and challenges' Proc IEEE 1166–1180.
24. Kenneth Forbus, Jeffrey Usher,Andrew Lovett,Kate Lockwood,Jon Wetzel,' Sketch Understanding for Cognitive Science Research and for Education' Volume 3, Issue 4, pages 648–666, October 2011.

