A Data Collection Tool for Sketched Diagrams

Rachel Blagojevic[1], Beryl Plimmer[1], John Grundy[1, 2], Yong Wang[3]

[1]Department of Computer Science
[2]Department of Electrical and Computer Engineering
[3]Department of Statistics
University of Auckland
Private Bag 92019, Auckland, New Zealand

ABSTRACT

Repositories of digital ink sketches would be invaluable for testing and evaluation of sketch recognition software. However, there is no existing tool for flexible data collection and management of digital ink data for building repositories of hand drawn diagrams. We present a tool for the efficient collection, management and analysis of ink data. A resultant dataset records each ink stroke accompanied by participant and diagram information, stroke labels and measurements of various stroke features. This tool enables the effective construction of a large database of sketches to aid the development of recognition techniques.

Categories and Subject Descriptors (according to ACM CCS): I.7.5 [Document Capture]: Graphics recognition and interpretation.

1. Introduction

Stylus input hardware has spurred research of sketching tools. By imitating the pen and paper environment, use of sketch tools allows for ambiguity and quick construction of diagrams [PA03, BK03]. This is advantageous for early phase design due to its unconstrained nature which minimises cognitive load and decreases interruptions to the flow of creativity [Bla90, Goe95]. This flexibility is a stark contrast to conventional widget-based environments. With sketch tools, in contrast to traditional whiteboards and pen-and-paper sketching, there is the ease of digital storage, transmission and replication gained from computerisation. Potential uses include office automation, software design, electronics design, architecture and civil engineering, and education.

However diagramming-based sketch tools are yet to gain general acceptance. One of the reasons for this is the need for far more accurate recognition than is currently available. Recognition is important as it allows these sketch tools to support more intelligent tasks such as editing, execution and conversion of these diagrams. However the ambiguity of hand drawn diagrams makes recognition problems hard to solve.

Typically recognisers are comprised of capturing stroke features and using algorithms to combine these features to identify the meaning of the ink. While many recognition algorithms have been developed to date [AD04, FPJ02, Gro96, LM96, Rub91a, SSD01, You05,], most have been informed by ad-hoc, heuristic-based assumptions about sketch properties. There is a critical need for more rigorous analysis of sketch recognition performance and tuning. The development of high precision recognition techniques requires large amounts of digital ink data to aid the training and evaluation stages. In addition to quantity, the quality of this data is paramount to the success of their development and therefore must be un-biased and representative of a wide range of diagram types. However, to enable this we require a corpus of well-authored sketches, sketch components and categorisation of data elements to be assembled.

There is little ink data available that meet these criteria and little support for obtaining such data. A tool that provides ease of data collection and management would allow us to construct a data repository more efficiently and therefore aid the development of recognition techniques.

This paper describes such a tool. The next section gives an overview of past work related to the collection of ink data. Sections three and four describe the requirements of a sketch data authoring tool and the implementation details of the prototype we developed according to these requirements. Section five describes our evaluation

relating to the usability of our software. We then proceed to discuss the wider potential of our tool in section six and conclude with a summary.

2. Background

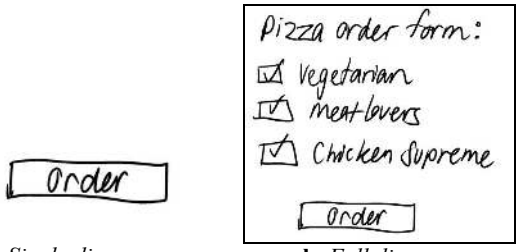
Sketch tools generally include some form of recognition. Early sketch tools include the user interface design software Silk [LM96] and The Electronic Cocktail Napkin [DG01, Gro96] for architecture design. Both of these tools provide some form of recognition of hand drawn diagrams. Rubine's work [Rub91, Rub91a] in gesture recognition has been used by many other sketch recognition systems. It involves using a linear classifier for single stroke ink recognition. Rubine reported a 96.8% success rate. However, further experiments that re-implement Rubine's algorithm have been lower 86% [Pli04] and 84% [You05]. Despite this his algorithm has been widely adopted [CGH03, CMP05, DHT00, FP07, LM95, LNHL00, PA03a, Pli04, You05] with various alterations to the feature set reported. Recognition for many diagram domains have been explored including CALI [FPJ02] for general shape recognition, a mechanical engineering design tool [SSD01], Tahuti [HD02] for UML class diagrams and SketchREAD [AD04] a multi domain recognition tool.

However, little rigorous analysis has been applied when identifying the features and algorithms to be used in each recognition technique. Typically feature and algorithm selection is made heuristically [Rub91a, SSD01, YC03]. Fonseca et al [FJ01, FJ00, FPJ02, JF99] report using percentile graphics for each possible feature which show the statistical distribution of feature values for different shape classes. This is one of the few ink feature sets that is scientifically-based.

Our previous work [Pat07, PPGI07] looked at using formal techniques to identify a feature set for dividing text and shape strokes in diagrams. We built a dataset of 1519 strokes from various types of diagrams. This dataset was then analysed using a statistical partitioning technique which constructed a decision tree. The resulting tree contained the eight most significant features chosen from a set of 46 candidate features.

However, there were limitations of this work stemming from problems with collecting unbiased, high quality data. Participants were asked to copy diagrams from pre-drawn figures on paper. This may have caused some bias in the timing data obtained as we would expect participants to copy diagrams much faster than when constructing their own from scratch (and timing was identified as one of the important features). Also many of the diagrams that the data was obtained from were not complete, but were composed of one single diagram component as shown in Figure 1a as opposed to a full diagram such as Figure 1b. This would have influenced some of the information obtained regarding stroke relationships. It was also clear that a more efficient method of collecting, labelling and managing large amounts of data was required.

Wolin et al [WSA07] designed a tool for more efficient labelling of ink data using a stylus. Their tool is able to complete three main tasks; stroke fragmentation (automatic and manual), stroke grouping and symbol labelling. They claim that fragmenting strokes is important before labelling as users may draw more than one symbol using a single stroke. Fragmenting can also help divide strokes into primitives i.e. lines and arcs. Stroke grouping is for the opposite problem of labelling components made of more than one stroke. Once these tasks have been performed labelling the symbol in the sketch can be carried out efficiently. Their tool also allows for multiple labels to be applied to a stroke. Their usability study showed that overall the user interface was intuitive and easy to use. Possible improvements that were discussed were that better support is required for using multiple labels and that undo/redo is helpful in such an interface. Although this tool has very useful features for labelling sketches it only covers one stage of the overall data collection and management process.



a. *Single diagram component* **b.** *Full diagram*

Figure 1

There are very few databases of hand drawn sketches available. Oltmans et al [OAD04] describe their experiences in collecting sketch data while building an Experimental Test Corpus of Hand Annotated Sketches (ETCHA Sketches). The process of constructing this database included collecting sketches and then labelling the primitive shapes within the sketches. Their data covers four domains including circuit diagrams, family trees, floor plans and geometric configurations; however there is no text included in these sketches. Participants were asked to label their diagrams themselves. As different recognition problems require slightly different data from each sketch four possible types of labels were identified: (a) "Best in isolation" labels for a single stroke classifier, second, (b) context based labels, and (c) "Is a" and (d) "Can be a" labels where a group of labels are assigned to a stroke for example a slightly curved line is a line and an arc.

Hse and Newton [HN04] have also compiled a test corpus of sketches. They asked their participants to sketch examples of 13 different symbols, which are predominately basic shapes such as rectangles and circles. This dataset has similar problems to our previous work

[Pat07, PPGI07] as only one single diagram component is drawn for each sketch. In this case data concerning stroke relationships in a full diagram are lost. In addition there is no writing included in any of these examples.

These databases provide a good start to building a repository of data for sketch recognition research however there are many other domains to consider and limitations to overcome. In addition there is still the lack of a tool that can support all aspects of data collection and management.

3. Requirements

The fundamental requirement of our tool is that it minimises the time and effort taken when carrying out ink data collection and management tasks. In order to meet this requirement our tool must provide support for data collection, labelling and dataset generation, as well as meeting common user interface requirements.

The tool must support the collection of data in an unbiased manner to ensure its quality. By unbiased we mean the method of collection used follows as closely as possible to the natural practice of drawing diagrams so that the data obtained provides a true representation of typical diagrams. In terms of quantity, it must have the ability to manage large amounts of data and ensure that this data collection is fully extensible, for example when adding new features to measure. In addition the data must be easily extractable in a variety of formats for third party analysis tool purposes.

The user interface requires basic functions of (sketch) draw, erase and select. Also functions to open and save a project (using xml files) are necessary. A project can contain many participants who can sketch many diagrams (see Figure 2). There should be pre-defined templates for each diagram type which contain a diagram name and instructions to participants on what to sketch. This way each participant sees the same information before sketching their diagrams which helps to keep these variables constant. Also pre-defined labels for each diagram template are required. These are defined by the user based on the type of diagram that is being collected.

A representative usage scenario for creating a new project is as follows. The researcher opens the application and is prompted to either open an existing project or create a new one. They choose to create a new project and give it a name. They are then prompted to define templates for the types of diagrams that they wish to collect. For example they want to collect organisation diagrams and user interface designs so they create two templates where they define a name and instructions on how to construct each diagram. They also define labels for each diagram type, for example for organisation diagrams labels may include “person”, “connectors” and “text”. The researcher is then free to begin collecting sketches.

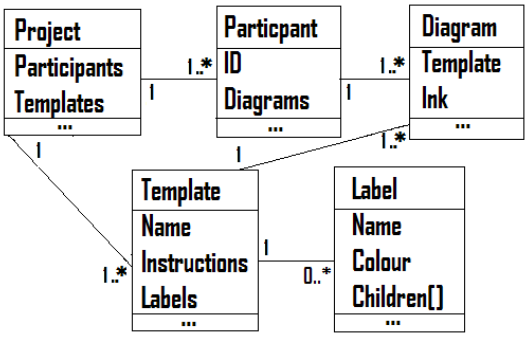


Figure 2: *Class diagram. A Project can have many Participants. A Project is defined by one or many Templates where each Template describes the type of diagram to be collected (this includes pre-defined Labels). Participants can draw many Diagrams. Each Diagram is based on a pre-defined Template.*

As mentioned earlier our tool must support the collection of multiple sketches from many participants. A tab view with a drawing area for each diagram defined by the project templates and written instructions on how to construct these diagrams would be an ideal way to display what participants are required to sketch. Editing facilities such as select and erase are available when drawing these diagrams. When a participant is finished the sketches are viewable but not editable.

A representative usage scenario for collecting sketches is as follows. The participant reads the instructions and draws the diagrams defined by the instructions in the drawing area. If there is more than one diagram required each one will have a separate tab. When the participant is ready to complete a new diagram they can switch tabs to complete the remaining diagrams in the same manner.

Once sketches have been collected each component of the sketch must be labelled. Automatic and manual labelling is available. We begin by supporting automatic labelling of shape and text strokes using our “sketch divider” [Pat07, PPGI07], which categorises ink as text or a shape. This will be extended later with further recognition and categorisation algorithms. Manual labelling can be used to correct the automatic parser and add further information. A hierarchy of labels should be pre-defined in the diagram template. A hierarchy is used so that enough information is available for different recognition problems. For example one stroke in a diagram maybe labelled as a circle which will automatically imply that it can also be labelled as a shape stroke for more general recognition problems. Strokes should also be numbered for unique identification at a later stage.

When enough sketches are collected and labelled they can be turned into a dataset. This involves calculating a number of features for each stroke in each sketch and outputting a dataset file. The interface should make it easy to select which participants/diagrams/features should

be included in the dataset and the format of the output file. The dataset can then be imported into data mining tools such as R [RDC06] and Weka [WF05] to be analysed for the development of new recognisers.

4. Prototype Usage Example

We illustrate a sample usage scenario of our sketch data capture prototype. A user wants to create a corpus of sketches for the domain of family tree/organisation charts.

When the application starts a dialog is shown giving the researcher the option of either creating a new project or opening an existing project. If they choose to begin a new project they first specify template diagrams on which this project is based.

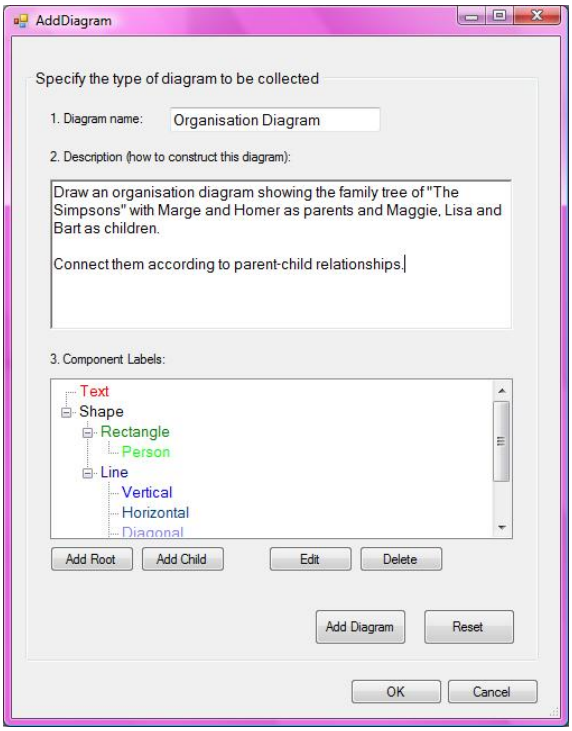


Figure 3: *Add template form*

A template provides information on a diagram type that the researcher wishes to collect. It consists of a name, instructions on how to draw a diagram and a set of labels to describe the components of that diagram. A dialog box (Figure 3) is displayed asking them to define a template by specifying this information.

4.1 Data Collection

Once the templates have been specified the researcher may begin collecting sketches from participants. Using the tools menu they can click on Data Collector which will take them to a screen similar to that shown in Figure 4. Figure 4 shows a list box (a) which lists the ID numbers of the participants who have contributed to the project. Clicking on each ID number will show the diagrams that the corresponding participant has drawn.

In the middle of the screen (b) is the drawing area as shown in Figure 4. There is one tab for each diagram. Clicking on each tab will also change the text area (c) to display the correct instructions for drawing that diagram (as specified by the researcher when creating the diagram templates shown in Figure 3).

All data collected is saved to xml files. This includes project information such as the diagram templates, all the raw stroke data for each participant and the corresponding labels applied to these strokes as discussed in the next section.

4.2 Labelling Data

Once a diagram has been drawn the strokes can be labelled. Using the tools menu the user (researcher or participant) can select the Labeller which will take them to a screen similar to that shown in Figure 5.

The user interface for the Labeller has the same list box (a) showing the participant ID's and tabs for each diagram. The drawing area (b) on each tab is un-editable except for changing the colour of the stroke. Pressing the auto parse button (c) will automatically parse the current

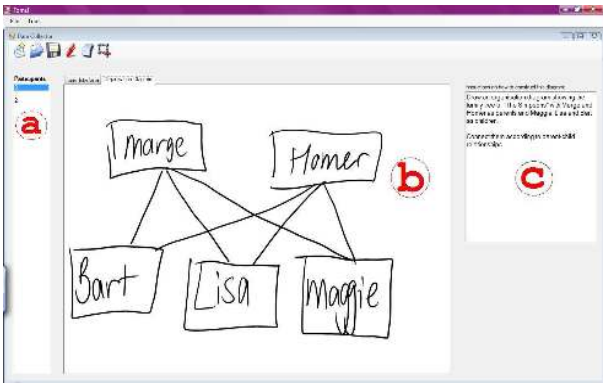


Figure 4: *Data collector form*

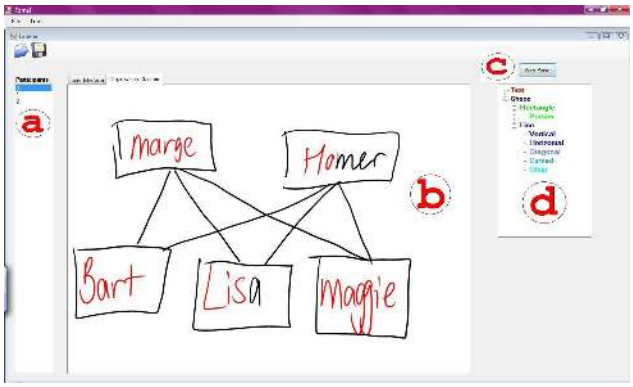


Figure 5: *Labeller form showing a diagram labelled using the automatic parser.*

diagram using our divider [Pat07, PPGI07] into shape and text strokes. It colours the strokes according to the colour map shown in the tree view box (d) e.g. text strokes are red and shape strokes are black.

The user can also manually label strokes by selecting the correct label from the tree view component and clicking on the stroke/strokes that require this label. These labels are those specified when defining the template for that diagram type as seen in Figure 3. The stroke is then coloured to match the deepest label in the tree as shown in Figure 6. We have chosen this hierarchical labelling structure to allow more than one label to be applied to a stroke without manually specifying each one.

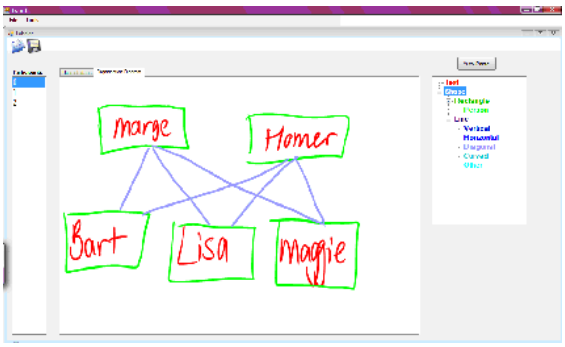


Figure 6: *Labeller form showing a diagram labelled manually.*

4.3 Dataset Generation

The final step to this data collection process is to generate a dataset. To generate a dataset the researcher selects Dataset Generator from the tools menu. A screen similar to that shown in Figure 7 will appear.

There are three steps to generating a dataset; first we must choose which stroke features we want to measure, then which diagrams we are interested in measuring and finally the format for the output file.

A list of possible features is displayed in a list box (a). This list is dynamically generated to ensure that the feature set is fully extensible. There is a check box (b) to enable the user to select or deselect all features with ease. Only those features selected are calculated in the dataset.

All the diagrams that are part of the current project are displayed in another list box (c). It has a tree structure showing which diagrams each participant has drawn. A quick select list (d) is available to enable the user to select or deselect all the diagrams or easily choose only certain diagram types. Only the diagrams that have been selected are included in the dataset.

The final combo box (e) shows the file format options for the dataset. This currently includes .xls, .csv and .arff

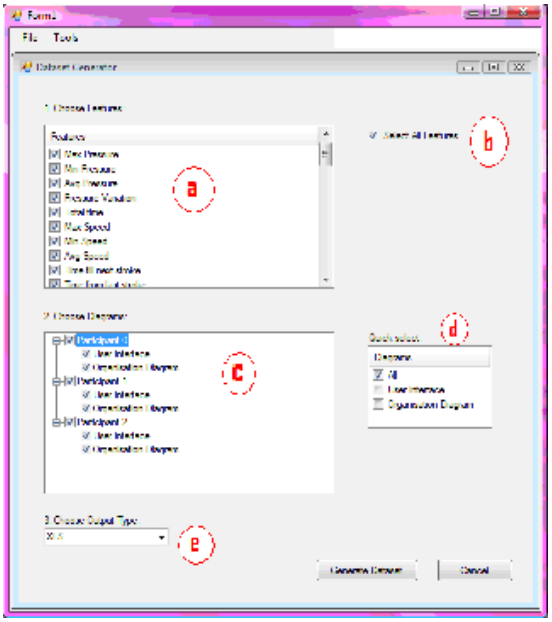


Figure 7: *Dataset generator form*

(Weka format [WF05]). However, more output options can be added.

Once all the desired selections have been made the user clicks on the generate dataset button. Each selected feature will be calculated for each stroke in the selected diagrams with the results written to the chosen output file. Figure 8 shows an example dataset (using the .xls format) for the organisation diagram in Figure 6. There are extra pieces of information added to each stroke including the participant ID of the person who drew the diagram, the diagram name, the stroke ID and the labels applied to that stroke.

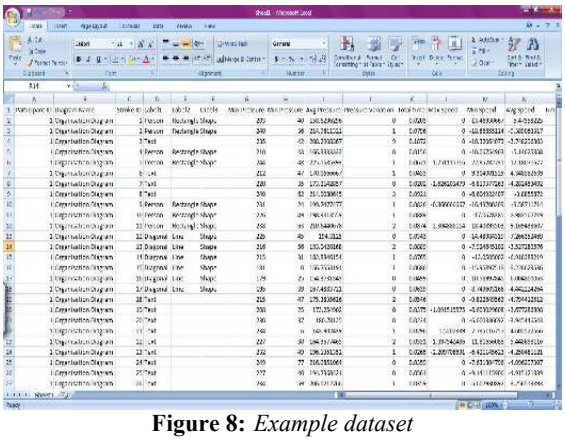


Figure 8: *Example dataset*

The resulting dataset can then be analysed by data mining tools to determine the most significant features and algorithms for any given sketch recognition problem. For example, Figure 9 shows the data from Figure 8 being analysed by Weka [WF05] to construct a decision tree for recognising basic shapes.

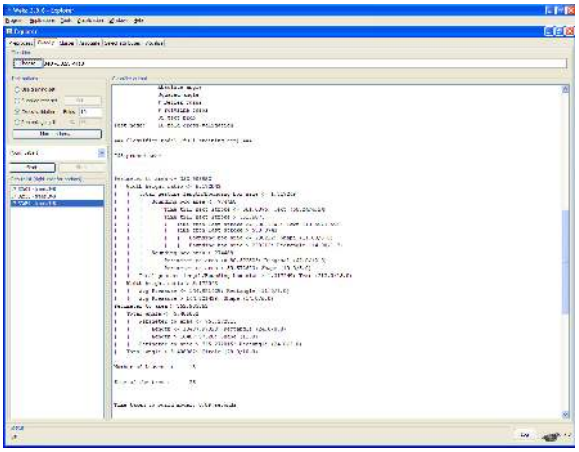


Figure 9: *Decision tree analysis in Weka [WF05]*

5. Evaluation

A usability study was used to test how intuitive our tool is to use, in particular the data collection and labelling interfaces. For the data collection interface we wanted to determine how easy it is for participants to sketch diagrams using the provided instructions. Then for the labeller we are interested in how efficiently we can label the collected diagrams with the existing interface.

5.1 Data collection

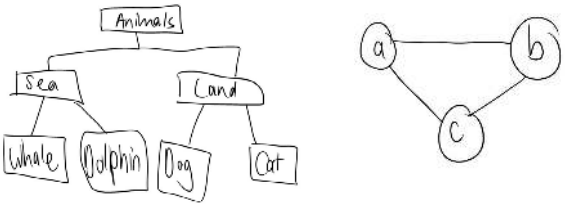
Six students from a computer science and software engineering background participated in the study. Half of the participants use pen input on a computer frequently and half had used pen input only occasionally or once before.

The participants were asked to draw two types of diagrams, an organisation chart and a graph as shown in figure 10. They were given very specific instructions on how to construct these diagrams. However, at this stage we were not interested in evaluating the way we present problems to participants to sketch, we were only testing the usability of the interface.

We observed the participants as they completed each task and then asked them to complete a short questionnaire. The questionnaire focused on learning how easy it was for participants to complete the tasks with our software on a Tablet PC.

All participants strongly agreed that creating the diagrams was easy given the environment and also agreed that the interaction tools (hardware and software) helped them to complete each task.

All of the participants agreed that they understood the tasks they were to perform. Although we were not evaluating the way that the tasks were presented this feedback gives us a positive indication that the style used to display instructions to the user on how to complete each task is effective. We will evaluate this aspect of data collection further in the future.



a. *Organisation diagram* **b.** *Graph*

Figure 10: *Diagrams collected for the usability study*

Five of the six participants, were neutral when asked if editing and checking the diagrams was easy. This is because most completed the task without a need to edit the diagram as the tasks were easy to understand and presented with clear instructions as discussed previously. The sixth participant strongly agreed that editing and checking the diagrams was easy.

Three participants, after completing the first task, almost used the participant list box by accident to navigate to the next task. However before clicking in the wrong place they quickly realised that they needed to use the tabs to switch tasks. The names of the tabs could include the label “Task n” before the diagram name and have a larger font to make this selection obvious.

Also one participant was unsure where the instructions for the second task were as they did not realise that the text area would change to display the instructions corresponding to the selected tab. We intend to simply include the text area within the tab to make it clear which instructions belong to which task.

5.2 Labelling

We were also interested in evaluating how efficient our labeller is to use. After collecting all the diagrams from the participants we labelled each sketch and measured how long this process took.

To label all 12 diagrams (two diagrams per participant) it took approximately seven minutes. An extra minute was used to double check all the diagrams and another 2.5 minutes for the dataset to be generated using all 45 features in our current feature set. This is a total of approximately 10.5 minutes to label all diagrams and generate a dataset of 476 strokes. In comparison, manually labelling the data for our previous work with 26 participants and 1519 strokes took approximately a 1-2 days work.

When labelling the diagrams we found that the automatic parser using our text-shape divider [Pat07,

PPGI07] to give preliminary labels to the diagram was especially helpful given the amount of text that was in the diagrams.

One possible improvement that could be made is to allow all the diagrams of the same type to be labelled together. For example label all the organisation diagrams first, and then label all the graph diagrams, rather than labelling all the diagrams for each participant. This may make the labelling process more efficient as it minimises the cognitive load required when switching tasks and allows for familiarity with labelling one type of diagram. Modifying the participant list box, shown in Figure 5(a), to display by diagram types as an alternative to participant would allow the user to navigate through each sketch as required.

6. Discussion

The tool we have presented provides a framework for the processes involved in data collection and management of sketches. The modularity allows for existing and future tools to be easily included to provide more functionality or to build on the current functions. For example the sketch labelling tool presented by Wolin et al [WSA07] could easily replace the labeller that exists in our tool.

Our objective is to develop this tool into a framework for building recognisers. In addition to the data collection and management support that exists, a framework would involve building a library of common recognition and feature selection algorithms and an automated evaluator for the recogniser.

The feature selection algorithms would first be used to determine the best feature set to use for the required recognition problem. The chosen feature set could then be applied to various recognition algorithms, resulting in a collection of recognisers.

These recognisers could be ranked using an automated evaluator. This would involve using each recogniser on the sketches collected and determining their accuracy by comparing the recognition results with the labels previously applied to the diagram. Using this evaluator we could identify the most accurate recogniser as the one with the best recognition rate in comparison to the other recognisers.

These recognisers could also be added to the labeller for automatically parsing diagrams to apply stroke labels and continue the cycle.

7. Conclusion

We have described the key requirements for a sketched diagram digital ink capture tool for assembling a corpus of quality ink data. We have developed and evaluated a prototype authoring tool enabling such a corpus to be assembled. Preliminary evaluation results indicate the tool provides a good environment for capturing and categorising ink data for further analysis. We are using this analysis to inform our development of higher precision sketch recognition algorithms for diagram-based sketching tools.

Our prototype is available for download from http://www.cs.auckland.ac.nz/~rpat088/

References

[AD04] ALVARADO C. DAVIS R.: SketchREAD: a multi-domain sketch recognition engine. Proceedings of the 17th annual ACM symposium on User interface software and technology (2004), pp. 23-32.

[BK03] BAILEY B. P. KONSTAN J. A.: Are Informal Tools Better? Comparing DEMAIS, Pencil and Paper, and Authorware for Early Multimedia Design. In CHI (2003), pp. 313-320

[Bla90] BLACK A.: Visible planning on paper and on screen: The impact of working medium on decision-making by novice graphic designers. Behaviour and information technology, 4, 9 (1990), pp. 283-296.

[CGH03] CHEN Q., GRUNDY J. HOSKING J.: An E-whiteboard application to support early design-stage sketching of UML diagrams. Human Centric Computer Languages and Environments (2003), pp. 219-226

[CMP05] CHUNG R., MIRICA P. PLIMMER B.: InkKit: A Generic Design Tool for the Tablet PC. In CHINZ (2005), pp. 29-30

[DHT00] DAMM C. H., HANSEN K. M. THOMSEN M.: Tool support for cooperative object-oriented design: Gesture based modelling on and electronic whiteboard. In CHI (2000), pp. 518-525

[DG01] DO E. Y. L. GROSS M.: Thinking with Diagrams in Architectural Design. Artificial Intelligence Review, 15, (2001), pp. 135-149

[FJ01] FONSECA M. J. JORGE J. A.: Experimental Evaluation of an on-line Scribble Recognizer. In Pattern Recognition Letters (2001), pp. 1311–1319.

[FJ00] FONSECA M. J. JORGE J. A.: Using Fuzzy Logic to Recognize Geometric Shapes Interactively. In Proceedings of the 9th International Conference on Fuzzy Systems (FUZZ-IEEE), (2000)

[FPJ02] FONSECA M. J., PIMENTEL C. E. JORGE J. A.: CALI: An Online Scribble Recogniser for Calligraphic Interfaces. In AAAI Spring Symposium on Sketch Understanding (2002)

[FP07] FREEMAN I. PLIMMER B.: Connector Semantics for Sketched Diagram Recognition. In AUIC (2007), pp, 71-78

[Goe95] GOEL V. Sketches of thought. In The MIT Press, (1995)

[Gro96] GROSS M.: The Electronic Cocktail Napkin-a computational environment for working with design diagrams. Design Studies, 1, 17 (1996), pp. 53-69

[HD02] HAMMOND T. DAVIS R.: Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams. In AAAI Spring Symposium on Sketch Understanding (2002)

[HN04] HSE H. NEWTON A. R.: Sketched Symbol Recognition using Zernike Moments. International Conference on Pattern Recognition (2004), pp. 367-370

[JF99] JORGE J. A. FONSECA M. J.: A Simple Approach to Recognise Geometric Shapes Interactively. In Proceedings of the Third Int. Workshop on Graphics Recognition (GREC), (1999)

[LM95] LANDAY J. MYERS B.: Interactive sketching for the early stages of user interface design. In CHI'95 Mosaic of Creativity (1995), pp. 43-50

[LM96] LANDAY J. MYERS B.: Sketching storyboards to illustrate interface behaviors. In CHI '96 (1996), pp. 193-194

[LNHL00] LIN J., NEWMAN M. W., HONG J. I. LANDAY J. A.: Denim: Finding a tighter fit between tools and practice for web design. In CHI 2000 (2000), pp. 510-517

[OAD04] OLTMANS M., ALVARADO C. DAVIS R.: ETCHA Sketches: Lessons learned from collecting sketch data. In AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural. (2004)

[Pat07] PATEL R.: Exploring better techniques for diagram recognition. University of Auckland, (2007), MSc

[PPGI07] PATEL R., PLIMMER B., GRUNDY J. IHAKA R.: Ink Features for Diagram Recognition. In 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling (2007)

[Pli04] PLIMMER B. Using Shared Displays to Support Group Designs; A Study of the Use of Informal User Interface Designs when Learning to Program. University of Waikato, (2004), PhD

[PA03] PLIMMER B. E. APPERLEY M.: Evaluating a Sketch Environment for Novice Programmers. In SIGCHI (2003), pp. 1018-1019

[PA03a] PLIMMER B. E. APPERLEY M.: Freeform: A Tool for Sketching Form Designs. In BHCI (2003), 2, pp. 183-186

[RDC06] R DEVELOPMENT CORE TEAM. R: A language and environment for statistical computing. R Foundation for Statistical Computing, (2006)

[Rub91] RUBINE D. H. The automatic recognition of gestures. Carnegie Mellon University, (1991), PhD

[Rub91a] RUBINE D. H.: Specifying gestures by example. In Proceedings of Siggraph '91 (1991), pp 329-337

[SSD01] SEZGIN T. M., STAHOVICH T. DAVIS R.: Sketch based interfaces: early processing for sketch understanding. In Proceedings of the 2001 workshop on Perceptive user interfaces (2001), pp. 1-8

[WF05] WITTEN I. H. FRANK E. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, (2005).

[WSA07] WOLIN A., SMITH D. ALVARADO C.: A Pen-based Tool for Efficient Labeling of 2D Sketches. In 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling (2007)

[You05] YOUNG M. InkKit: The Back End of the Generic Design Transformation Tool. In University of Auckland, (2005), BEng

[YC03] YU B. CAI S.: A domain-independent system for sketch recognition. Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia (2003), pp. 141-146.