1 Introduction

Diagrams are an important, if not pivotal, part in both education and design. In an effort to understand abstract concepts, students play an active role in their education by specifying visual and abstract concepts in hand-sketched diagrams. While students are understanding abstract concepts through hand-drawn diagrams, designers are creating those abstract concepts. Just as the act of hand-drawing a diagram (as opposed to using a mouse-and-palette CAD tool) better engages the student in the learning process, the act of hand-drawing a diagram also improves the design process by freeing the designer of constraints that may otherwise impede creativity and innovation.

Diagrams, and thus hand-sketched diagrams, are fundamental to many domains including circuit diagrams, mechanical engineering, concept maps, flow charts, finite state machines, civil engineering, and foreign language learning. Hand-sketched diagrams offer many benefits not yet present in a traditional CAD tool, such as intuitive input methods. Hand-sketched diagrams are pervasive, but currently most computer programs are unable to understand this natural and informal form of input. Diagrams are commonly sketched on paper, and then re-entered into a computer-understood program using a mouse and palette tool for processing, wasting time and effort, as well as interrupting the design and learning flow.

Tablet-PCs have attempted to bridge that gap, allowing people to use a digital pen to draw directly on a computer screen and store the information for later examination. The tablet-PC technology on its own leaves the sketch mostly unrecognizable by a CAD tool. Sketch recognition is a subfield in computer science that develops algorithms using artificial intelligence to recognize and understand the hand-drawn strokes on a screen just as a human would; these understood sketches can then be sent to a CAD or other computer program for simulation or other intelligent interaction.

Sketch recognition technologies exist for a limited set of domains but would benefit a myriad of currently unsupported domains. Two tools, InkKit and LADDER, currently exist to allow users to create their own sketch recognition systems without expertise in sketch recognition. The goal of this tutorial is to provide each participant with an overview of the two currently existing sketch tools with which they can design domain specific sketch recognition technology.
Each toolkit provides 1) recognition engines to recognize hand-drawn sketches, 2) innovative ways to define new shapes and provide details that can help recognition of those shapes, and 3) methods for interaction, including display and editing settings, to help make the created user interface as natural and intuitive as possible. Interaction settings can have a profound effect on the usability of a system. Changes in display can act as an important method of recognition feedback, whereas self-modifying objects (such as instant object beautification) on the screen may be distracting for certain domains.

2 InkKit

InkKit is a fully-featured sketch toolkit. The fundamental goal is to minimize the effort required to support recognition of a specific type of diagram (for example entity-relationship diagrams). There are three main parts of InkKit: the user interface, the core recognition engine, and extensibility via plug-ins. It has been developed for the Microsoft Vista OS in Visual Studio .Net using C#. This has the advantages of providing a good hardware platform and the Ink SDK for basic ink data support and character recognition. The InkKit user interface contains two main interaction spaces, sketch pages and a portfolio manager (Figure 1). Sketch pages belong to a portfolio and pages can be linked together the meaning of the links is dependent on the plug-in. There are supplementary interfaces for defining recognition components and other options.

The recognition engine is able to recognize diagrams containing shapes and text. The core InkKit recognizer consists of a divider, which separates writing and drawing; a basic shape recognizer for rectangles, circles, etc.; and a component recognizer, which combines basic shapes and text into diagram components.

To add another diagram type to InkKit the plug-in developer writes a small interpreter, provides some example components, and writes appropriate output

![Fig. 1. InkKit User Interface Showing ER Diagrams](image)
modules. The developer can exclude unused basic shapes and add new basic shapes from the user interface. Also from the UI the developer provides a few examples of each component. The core recognition engine analyses these examples to derive the recognition rules. The plug-in must define the component names and can be used to supplement the recognizer with additional rules. InkKit does not support any beautification of the digital ink. Formalization is achieved by writing an output module that converts the recognized diagrams into a format for other tools. We have written plug-ins for a range of diagrams and output formats including UI designs to Java and html; directed and undirected graphs and organization charts to text, picture formats or MS Word diagrams; and ER diagrams to Java. A typical plug-in consists of less than 500 lines of simple code.

3 LADDER/GUILD

LADDER (initially an abbreviation for a Language for Describing Drawing, Display, and Editing for use in sketch Recognition) has grown to be a multi-tier recognition system which generates a user interface based on a LADDER description. A developer can use LADDER to recognize shapes and gestures in a particular domain. In order to recognize a shape, a user first draws an example of the shape or gesture. If the user specifies that he or she is drawing a shape, the system will automatically generate a geometric description of the shape composed of geometric primitives: lines, arcs, curves, ellipses, circles, spirals, or helixes. The user is then shown the shape’s generated geometric constraint list and provided the option (if the description needs improvement) to either modify the list by hand, draw more examples, or choose from a number of automatically generated near-miss examples. Contrarily, if the user specifies that he or she is drawing a gesture, a list of features specific to the gesture are recognized and the user is asked to draw several more examples. Since context plays a significant part in recognition the user can also specify contextual information by hand that may help to better recognize the object, preventing ambiguity.

As editing and display are important in creating a usable sketch recognition system, in LADDER, a developer has the option to specify how each shape should
be displayed and edited either by using some simple preset selections (such as
translating an object by pressing and holding the pen briefly over the object
to ‘grab’ it before moving it and/or displaying the beautified, icon-replaced, or
original strokes) or by specifying it more specifically using the LADDER sketch
language. LADDER supplies a Java API to connect to an external program.
Thus, the developer can create his or her own display panel, monitor and con-
trol recognized objects, as well as connect to an existing program to provide
simulation or other intelligent interaction.

4 Conclusion

The two sketch recognition toolkits described above provide complimentary ad-
vantages. InkKit is available for download at [https://www.cs.auckland.ac.nz/
~beryl/inkkit.html](https://www.cs.auckland.ac.nz/~beryl/inkkit.html), whereas LADDER is available for download at [http://srl.cs.tamu.edu](http://srl.cs.tamu.edu). We are always looking for collaborators who would like to
be trial users for our software, and we encourage you to try our toolkits in un-
precedented domains. These tools are still in beta versions, and we look forward
to your comments as to how we can make them better.

5 Contributors

Dr. Beryl Plimmer, Ph.D.: Dr Plimmer is a Senior Lecturer in Computer Science
at the University of Auckland, New Zealand. She has many years experience in
industry and education. She has a BBus, MSc in Science Maths Education and
a PhD in Computer Science. Her passion is to make computers more usable and
useful sketch tools is one part of this endeavour. Her other research interests
include digital ink annotation, using multi-modal interaction to support visually
impaired students and user centred design.

Dr. Tracy Hammond, Ph.D.: Dr. Hammond is the Director of the Sketch
Recognition Lab and Assistant Professor in the Computer Science department at
Texas A&M University. Previously, she taught computer science courses for many
years at Columbia University. Additionally, she worked for four years full time
at Goldman Sachs as a telecom analyst developing speech recognition, and other
intelligent user interface applications. She earned her B.A. in Mathematics, B.S.
in Applied Mathematics, M.S. in Computer Science, and M.A. in Anthropology
all from Columbia University in the City of New York. She earned an F.T.O
in Finance and a Ph.D. in Computer Science from the Design Rationale Group
(DRG) in the Computer Science and Artificial Intelligence Laboratory (CSAIL)
at the Massachusetts Institute of Technology.

Acknowledgements

The authors would like to thank NSF IIS grants 0757557 & 0744150 as well as
members of the Sketch Recognition Lab.