EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2008)
C. Alvarado and M.- P. Cani (Editors)

SOUSA: Sketch-based Online User Study Applet

B. Paulson, A. Wolin, J. Johnston, and T. Hammond

Sketch Recognition Lab
Department of Computer Science
Texas A&M University
College Station, Texas

Abstract

Although existing domain-specific datasets are readily available, most sketch recognition researchers are forced to collect new data for their particular domain. Creating tools to collect and label sketched data can take time, and, if every researcher creates their own toolset, much time is wasted that could be better suited toward advanced research. Additionally, it is often the case that other researchers have performed collection studies and collected the same types of sketch data, resulting in large duplications of effort. We propose, and have built, a general-purpose sketch collection and verification tool that allows researchers to design custom user studies through an online applet residing on our group's web page. By hosting such a tool through our site, we hope to provide researchers with a quick and easy way of collecting data. Additionally, our tool serves to create a universal repository of sketch data that can be made readily available to other sketch recognition researchers.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [User Interfaces]: Interaction styles

1. Introduction

Sketch recognition works toward computer understanding of hand-drawn symbols and diagrams. Allowing users to interact with programs through sketches increases visual communication with a computer, and some domains and ideas are easier to express through drawing. For instance, architects describe buildings through floorplans, and having a computer understand sketched floorplans can assist in the design process [Gro96]. Other domains that can utilize sketch data include circuit diagram recognition [KS04,AD04], family trees [AD04], and physical system schematics [KS02, HD05].

Researchers in the sketch and visual fields of computer science have been working toward large, standardized datasets that can be used in many domains. A standardized dataset has benefits including removing the burden of data collection from future researchers, as well as common data available to all researchers resulting in uniform comparisons between systems.

One large corpus of sketch recognition data is the ETCHA Sketches data referred to in [OAD04]. This dataset contains hand-sketched drawings of family trees, circuit diagrams, floor plans, and geometric objects and can be found at http://rationale.csail.mit.edu/ETCHASketches/. Two image processing datasets are Caltech's 256 Google and Picsearch collection [GHP07] and the CBCL StreetScenes image database [Bil06]. The Caltech dataset contains over 30,000 object images in 256 categories, each category with a different number of images. The CBCL StreetScenes

Figure 1: *Participant using the user study applet.*

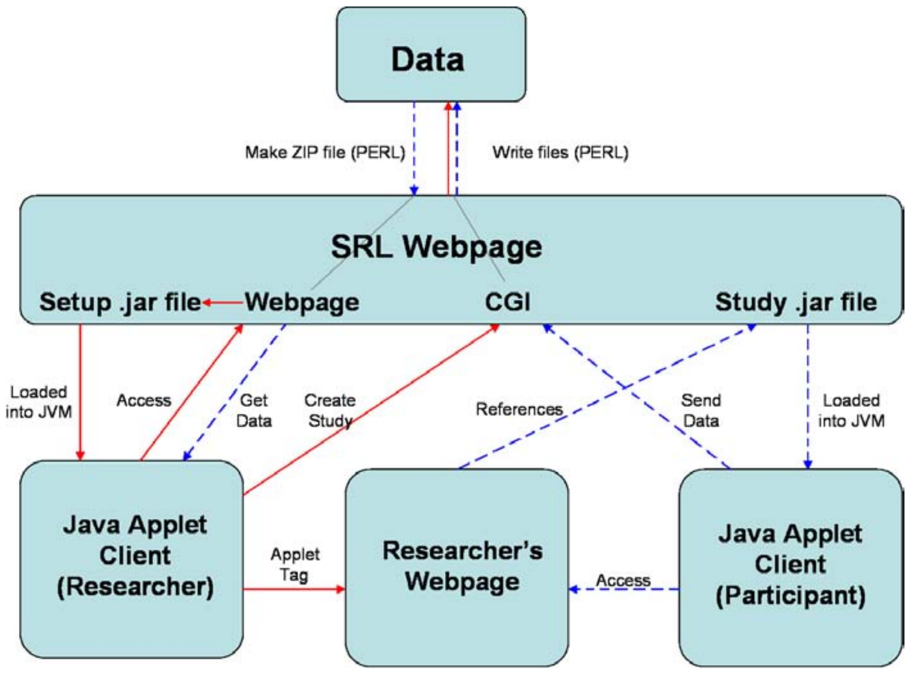


Figure 2: *Framework for our user study system. The red (solid) path denotes the flow of information for creating a study, while the blue (dashed) path denotes the flow of information for performing a study.*

database has labeled components of images taken on streets (e.g. cars, people, and roads).

Unfortunately, an abundance of general data is not always applicable to specialized domains. Creating a recognition system to recognize sketched Kanji symbols, for example, forces the developer to perform his or her own data collection and labeling as no standardized Kanji dataset exists. The two main ways to collect this necessary data are by gathering and labeling real-world data or by having users interact with a specialized program that handles the data collection and labeling process.

Labeling real-world data has been an issue in visual processing. For static images, labeling programs have attempted to make the labeling process fun and entertaining in order to have the public label large datasets for the researchers [RTMF05, vAD04, vALB06]. Sketch recognition labeling programs have not yet embraced these techniques, but the labeling program of Wolin et al. [WSA07] attempts to make labeling quick and efficient.

In some cases, real-world data is not easy to gather or is unnecessary during the prototyping phase. To gather data for a small user study, researchers often have users manually draw a given stroke or shape, and the sketch is saved according to the drawing's label name. We propose a sketch-recognition user study program that improves this collection technique by allowing researchers to interact with a web applet to:

1. Create and save user study domain descriptions
2. Associate multiple example images with each sketch or symbol in the domain
3. Allow researchers to collect data from multiple users through a web-based Java applet
4. Allow researchers to perform verification studies to get classification opinions based on numerous human interpreters
5. Automatically create a ZIP archive of data for easy download

Our user study program alleviates the issues involved with creating a program for a particular study while embracing the idea that data can be easier collected through web-based applications (e.g. [vALB06]). Through our applet, sketch recognition researchers can quickly gather data from multiple users across the globe, allowing for a large collection of diverse, automatically-labeled datasets (Figure 1). Furthermore, datasets are automatically packaged into ZIP files which can be easily downloaded online. This allows datasets to be shared amongst researchers.

2. Framework & Implementation

SOUSA, our user study system, allows sketch recognition researchers to create studies for both collection and verification purposes. Collection studies are those in which researchers can collect stroke data from users for a particular domain. Verification studies allow users to verify the sketches of other users; this can help determine if there is ambiguity among human interpretation.

The studies created by SOUSA can be used to not only collect individual shapes and symbols drawn with one or more strokes, but also to collect data for entire diagrams drawn with any number of strokes. Although our tool does not provide facilities for the labeling of data, since we use common data storage formats (see the discussion below about MIT SketchML), it would be easy to use a labeler after sketch data has been collected (for the cases where a sketch contains an entire diagram or complex shape and it is useful know which strokes, or sub-strokes, belong to which component).

The overall framework of our user study system can be seen in Figure 2. All user studies are created and performed online through Java applets hosted on our group's website. Although the applets themselves are hosted on our web page, sketch recognition researchers can create their own custom web pages which can contain an external link to the applet JAR (Java ARchive) files.

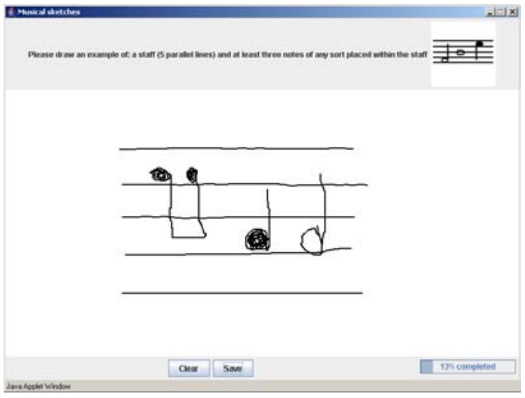
2.1. Setting up a collection user study

Sketch recognition researchers can create a user study by accessing a setup applet contained on our group's web page. Once the applet loads, the researcher will be prompted to enter information about the study. The researcher will first specify a save directory which will be created on our group's server. This will designate the study's location and where the researcher can later go to access the collected data.

Researchers can change options, such as the study title or the data format to save sketch information to (basic XML or MIT SketchML [OAD04], http://rationale.csail.mit.edu/ETCHASketches/format/), and can also specify information to gather from the user before the study begins (contextual information about the user; e.g. age, occupation, and the method of input—mouse or digital pen). Once these options have been selected, researchers can then define the types of sketch shapes they want to collect. A shape is specified by giving:

1. A textual description of the shape which will be displayed to the user
2. A short directory name specifying the folder in which the data will be saved
3. The number of examples to be drawn by each user
4. The maximum number of strokes the user is allowed to use to draw the shape (-1 is used to designate no limit)

There is no limit as to what types of things a researcher can ask users to draw. He can create a user study that collects shapes drawn with single strokes, or he can create a study that has users draw entire diagrams or large-scale sketches. For example, one user-study created with our tool is used to collect sketches pertaining to musical symbols. One may



Please draw an example of: a staff (5 parallel lines) and at least three notes of any sort placed within the staff

Figure 3: *Large sketch collected with the user study tool. Researchers are not limited to single strokes or simple shapes.*

be asked to draw an individual note, or a musical staff with many notes, as in Figure 3.

The researcher can also optionally specify (any number of) images that should be shown along with a shape description. Although researchers can specify the image names that are associated with each shape, they are unable to actually upload images through the applet. This is due to Java's security, which does not allow an applet to access the local user's file system without first signing the applet with an RSA key [Sun08]. To avoid this issue, we have the applet create a custom upload page on our server where the researcher can then go to upload the images. This page is created by having the applet send a POST command, along with all of the necessary data, to a CGI script written in PERL on our server, which then writes the upload page.

There are advantages and disadvantages, both, to showing the user an image of the sketch you are trying to collect. On one hand, the image will help to clarify your meaning, possibly disambiguating the textual description. On the other hand, images can contaminate the natural and unconstrained sketching process, biasing the user's sketch toward the predefined strokes illustrated in the image. A way to alleviate this is to create several different example images for a shape (which are chosen and shown at random in the study), or to leave out images altogether, which is up to the discretion of the creator of the user study.

A researcher may also wish to test the influences of context on how a user draws a symbol. For example, a plus sign (+) drawn in isolation might appear very differently than a plus sign drawn inside the middle of an equation. If this is the case, the researcher can request the user to sketch/write a mathematical equation with a plus sign in it (such as "2 + 2 = 4") instead of a solitary plus sign. Later, a separate labeling program can be used to extract the strokes/sub-strokes that make up the plus sign.

Once the researcher finishes with the study setup, he or

```
<applet
code="edu.tamu.bpaulson.userstudy.UserStudy"
archive="userstudy.jar"
userstudy="http://srl.csdl.tamu.edu/userstudy/mathsigns/
mathsigns_collect.xml" >
</applet>
```

Figure 4: *Example applet tag*

she simply clicks the "Save" button and the user study file is saved to our web server in XML format. The process of writing the file to the server is again done through a CGI script written in PERL. The file will be written to the directory specified in the "Save Directory" textbox and will be named with the following convention: `<save directory>_collect.xml`. If the researcher chooses to later modify the study, he or she can reload the study simply by pressing the "Load" button in the setup applet and entering in the URL of the saved study file.

The researcher can interface with the applet through his/her personal web page by simply adding an applet tag to existing HTML code (Figure 4). The tag need only specify the URL of the study, which lies on our group's web server. All other parameters in the applet tag will be constant. By default, studies will ask users to draw shapes in random order. If the researcher prefers users to draw shapes in a non-random order, then a `random="false"` parameter can also be specified in the applet tag.

2.2. Setting up a verification user study

Once data has been collected, a researcher may decide to perform a verification study to determine if human interpretation agrees with the sketches drawn by other users. A verification study is set up in the same manner as a collection study.

A researcher goes to our group's webpage, accesses the setup applet, and enters in the requested information:

1. The location (URL) of the saved data on our group's server
2. The study title
3. The user information to collect
4. The shapes which need to be verified
5. The number of times each shape should be verified by a single user

If the researcher wants additional classification options (such as a "None of these" option), then an additional shape can be specified which has its "Number to Verify" value set to zero and its description set to "None of these" (example Figure 11). Once this information is saved, a new study file is saved to the server in XML format with the naming convention: `<save directory>_verify.xml`. By default, like the collection study, all examples shown to the user appear in random order.

2.3. Performing a collection study

A user can perform a study by accessing the web page created by the researcher which contains the applet tag for the user study. The applet is loaded into the local user's Java Virtual Machine and begins by first warning the user that they are about to begin a study. From this point a user can either agree to participate, or can cancel and exit the applet. If the user agrees to participate, he or she will first be prompted to answer the pre-study questions that were designated by the researcher. This information is saved in a text file which is written to the main save directory. During this time, the user is assigned a unique ID number.

After entering in the requested information, users are given a description of a shape to draw. If images were specified, the user will be shown an example image, selected at random from those provided by the study creator, to go along with the description. When images are used, a subdirectory is created which contains a text file for each image, specifying which users were shown the image, along with the data files collected when that particular image was shown. This information is repeated, but on a per-user basis, in a text file saved to the main save directory.

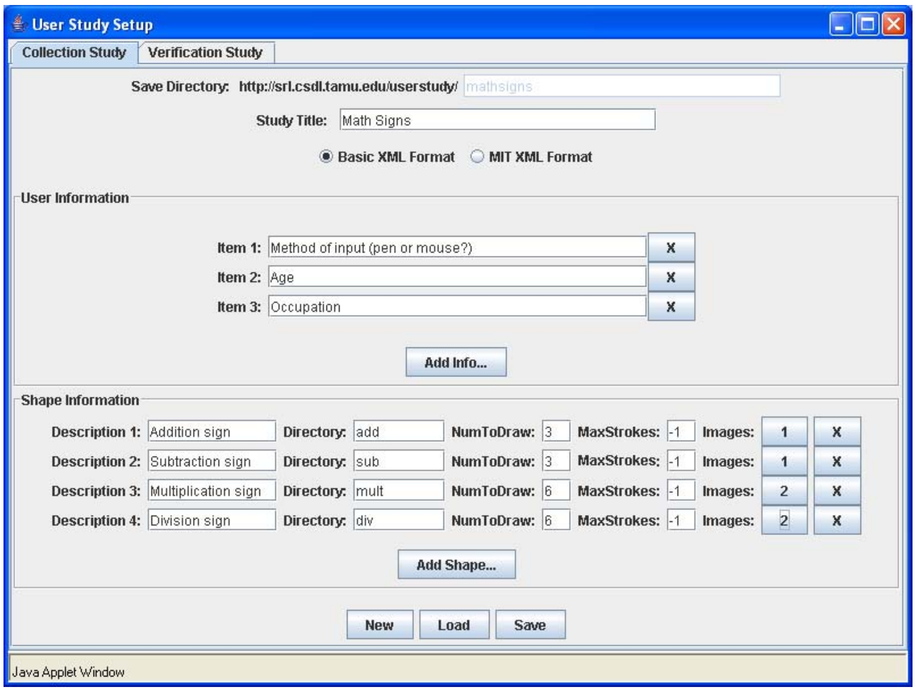
The user can sketch the example in the allotted white space and simply press the "Save" button to go to the next shape. If the user makes a mistake, he or she can press the "Cancel" button which will clear the screen so that the shape can be redrawn.

As the user presses the save button, data is saved to our group's server using the same CGI scripts as before. The filename of the data file encodes information about the example—`<short description>_<user number>_<example number>.xml`. Currently, the data collected about the strokes is limited to `<x, y, timestamp>` tuples per point within the stroke. This is a limitation of Java which we hope will be eliminated in the future to allow us to collect information such as pen pressure and tilt (if applicable), as well. The temporal data, as well as possible future inclusions of pressure and tilt data, provide additional context about the strokes that can be used by sketch researchers analyzing the collected sketches. This context includes stroke ordering and the amount of time taken between strokes. Once the user ends the study, all of the data for the study (including data from previous users) is written to a ZIP file which can be easily downloaded by the researcher at any time.

2.4. Performing a verification study

As with collection studies, verification studies can be interfaced with through the researcher's own web page. When users choose to perform a verification study, they will again be first prompted to enter in the user information specified by the researcher.

After the data is entered and written to a text file in the



Save Directory: http://srl.csdl.tamu.edu/userstudy/ mathsigns

Study Title: Math Signs

[✓ Basic XML Format] MIT XML Format

User Information

Item 1: Method of input (pen or mouse?)
Item 2: Age
Item 3: Occupation

Shape Information

Description 1:	Addition sign	Directory:	add	NumToDraw:	3	MaxStrokes:	-1	Images:	1
Description 2:	Subtraction sign	Directory:	sub	NumToDraw:	3	MaxStrokes:	-1	Images:	1
Description 3:	Multiplication sign	Directory:	mult	NumToDraw:	6	MaxStrokes:	-1	Images:	2
Description 4:	Division sign	Directory:	div	NumToDraw:	6	MaxStrokes:	-1	Images:	2

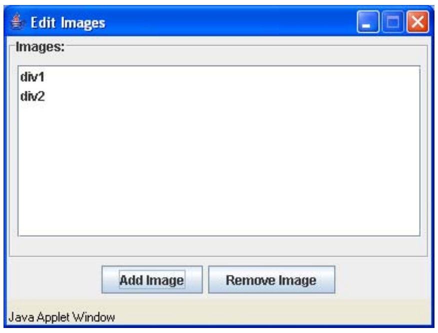
Figure 5: *Screenshot of the setup applet for creating a collection study, filled with the parameters for our example study to collect mathematical symbols.*

main save directory, the user will then be shown random examples of shapes sketched by previous users. They will then be prompted to select the button which best corresponds to the shape that is displayed on the screen. As the user selects a button, data is written to the server. Verification data and agreement percentages (if the label shown to the user that drew the example matches the label selected by the verification user) are written both on a per-user and per-shape basis.

3. Sample Study Generation: Math Signs

As a simple example, imagine a researcher wants to collect sketch data for a math recognition program. The researcher wishes to collect operator symbol data for addition, subtraction, division, and multiplication. She allows a plus sign (+) for addition, the minus sign (-) for subtraction, the asterisk ($*$) or cross ($\times$) for multiplication, and the slash (/) or obelus ($\div$) for division.

To create her study, the researcher begins by running the setup applet on our group's website. Once the applet loads, the researcher enters in the requested information, such as "mathsigns" as the name of the save directory. Figure 5 shows a screenshot of the setup applet after she has entered in all of the information. Note that for the multiplication and division signs, the researcher collects twice as many examples and specifies two images for these signs (Figure 6); this is to account for the different ways in which the signs can be



Images:

div1
div2

Figure 6: *Form used to specify image names. One of the two images, chosen at random, is shown when requesting the user draw a division sign.*

drawn and to try to collect an equal number of the different variations. Figure 7 shows two examples of the user being prompted to draw a division sign, each with a different example image. The researcher also sets the maximum number of strokes to "-1" for all shapes, meaning that these shapes can be drawn with any number of strokes. In some domains (such as collecting shapes for a single-stroke, Rubine recognizer [Rub91]) it may be beneficial to limit the number of strokes a user can draw. Once the study is created and the researcher presses "Save", the study is now located at:

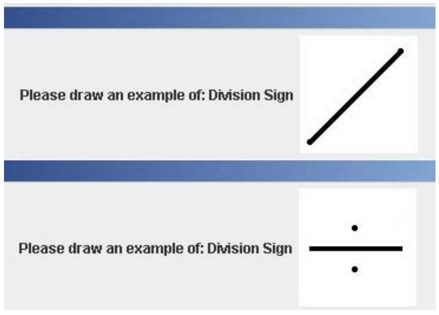

Please draw an example of: Division Sign
Please draw an example of: Division Sign

Figure 7: *One of the two different example images for division are shown at random when asking the user to sketch a division symbol.*

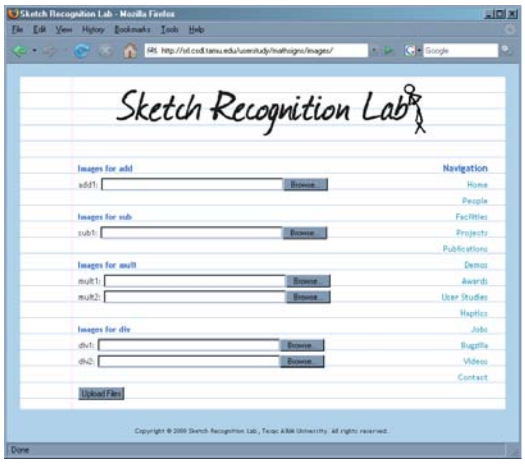


Figure 8: *Example upload form that is automatically generated by the setup applet.*

http://srl.csdl.tamu.edu/userstudy/mathsigns/mathsigns_collect.xml

Since images were specified for each sign to collect, the researcher then needs to upload the images that should be shown to people taking the study. Images are uploaded by simply visiting the upload page (Figure 8) that was automatically generated by the setup applet in the following location:

http://srl.csdl.tamu.edu/userstudy/mathsigns/images

After uploading any images, the researcher can create a personal webpage with the applet tag referencing her study's URL from above (Figure 9).

When users access this page, the applet will load and will first prompt for the user information specified in the study (Figure 10). This information is saved in a text file at the following URL:

http://srl.csdl.tamu.edu/userstudy/mathsigns/userinfo_collect.txt

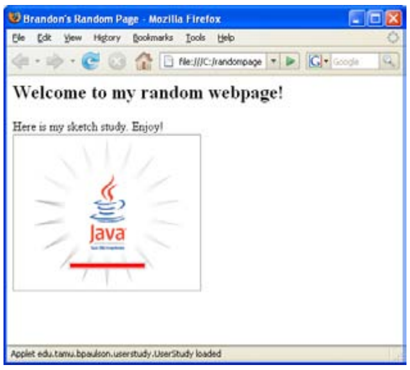

Brandon's Random Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/randompage
Welcome to my random webpage!
Here is my sketch study. Enjoy!
Java
Applet edu.tamu.bpaulson.userstudy.UserStudy loaded

Figure 9: *Example web page hosting an applet that launches a user study.*

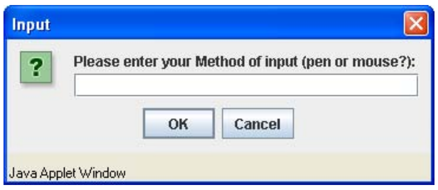


Figure 10: *Example prompt requesting user information.*

Next, users will be asked to draw the defined math signs in random order. The data will be saved within the subfolders specified when setting up the study. For example, the first example of the addition sign drawn by the first user (user number 0) will be located at:

http://srl.csdl.tamu.edu/userstudy/mathsigns/add/add_0_0.xml

Because the shapes have associated images, additional information will be written for this example. First, a folder named "collect" will be created in the "add" directory. In it, there will be a text file called "add1.txt" which corresponds to the name of the image that was shown to the user. If the researcher had specified more images for the addition sign (as she did for multiplication and division) then this folder would contain one text file per image. In this text file, the user number and name of the data file is saved. This file will continue to be appended to by this user and others as the "add1" image continues to be shown.

In addition, the system also creates a file in the main folder ("mathsigns") called "user0collect.txt." This file lists every image shown to the user during the session, along with the corresponding data file that was drawn. Although this data may be redundant, it eventually saves researchers time since they are able to quickly analyze the data both from a per-image view, as well as a per-user view.

After the user finishes the study, a command is sent to the server to create a ZIP file containing all the data. Essentially, this script packages the entire contents of the main folder, “mathsigns”, into a single ZIP file located at:

http://srl.csdl.tamu.edu/userstudy/mathsigns.zip

Although our toy example does not contain many ambiguous examples, we could have optionally chosen to perfom a verification study as well. Figure 11 shows the setup options we would have chosen for the study.

4. Evaluation

The primary beneficiaries of our system are sketch recognition researchers. To evaluate our system, we had 10 students from a sketch recognition class, as well as 11 students from a computer-human interaction class, use our system to collect data and evaluate perceptual thresholds with verification studies. Data was collected for various domains including Kanji, a physics simulator, and memory games, just to name a few. Verification studies were also created by some students to help determine perceptual thresholds for sketched strokes, such as whether two lines are parallel, touching, intersecting, near, far, etc. Our system has also been used by various researchers from our group to collect data for a number of projects [PH08, HEP*].

We asked users to give us comments and feedback afterwards to help evaluate the usefulness of our system. Overall, students and researchers felt the system was very helpful and quick and easy to use. According to one user, “after you use it once and get it working, it is very easy and quick to create new user studies to get data for different sketching domains.” The most time consuming part of creating studies for most users was creating the images to be shown for each shape.

One suggestion for the system was the ability to ask questions of the user after a study in addition to before the study in a pre-study/post-study questionnaire form. Users also noted occasional lags and delays during peak times when performing studies. Most of these lag delays were due to other resource-intensive processes being run on the web server, unbeknownst to us at the time. As of today, most of the lag problems have been remedied. While delays are always a concern with networked applications, we believe our application has a small enough network footprint (just transmitting small image files and stroke data in text format) that it will not cause problems.

5. Future Work

The biggest downside of our system is lack of security. Anybody can visit our website and create user studies without the need of a username, password, or any other form of verification. Furthermore, any user can load any study and make changes to it if they know the study’s URL. In the near future, we hope to remedy this by requiring username/password combinations for creating and modifying a study. In addition, we would also like to create a way to allow researchers to remove studies from our website after they have collected their data. A possible solution is to eventually abandon the creation applet and create an entirely web-based system for logging in and creating and managing studies. With this would come an interface overhaul as well.

Other, smaller areas of future work include creating additional features (such as the post-study questionnaire suggested by one user), providing more interactive feedback during study creation rather than relying simply on documentation, allowing verification buttons to be displayed in a random order, and potentially animating sketches during a verification study rather than showing the final static versions of the sketch (stroke replay, as is done in ETCHA Sketches [OAD04]).

6. Conclusion

We have created an online user study system that allows researchers to easily collect sketch data for a variety of domains. Furthermore, researchers can use the system to perform verification studies to determine if the intention of a drawer is the same as the perception of another human. SOUSA not only saves sketch researchers the time and effort needed to create their own data collection tools, but we also allow for the creation of a large repository of sketch data open to all researchers, saving time needed for collection and unifying data used to test different systems.

In order to evaluate our system, over 20 different students over the course of two separate classes and semesters have created and performed user studies with our system. The general consensus is that the system provides an easy and efficient way of collecting sketch data. By hosting the system on our own server, we hope to use the data from a multitude of domains to create a universal repository of sketch data that can be used by other sketch recognition researchers.

7. Acknowledgments

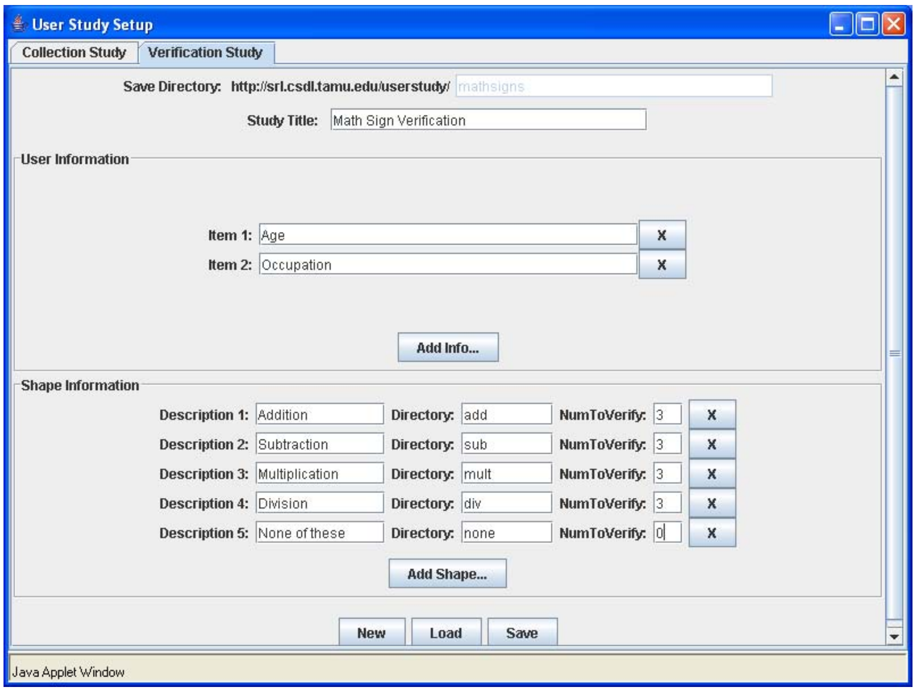
This work is supported in part by the National Science Foundation (Grants numbers 0744150 and 0757557).

References

[AD04] ALVARADO C., DAVIS R.: SketchREAD: a multi-domain sketch recognition engine. In *UIST ’04: Proceedings of the 17th annual ACM symposium on user interface software and technology* (New York, NY, USA, 2004), ACM, pp. 23–32.

[Bil06] BILESCHI S. M.: *StreetScenes: Towards Scene Understanding in Still Images*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2006.

[GHP07] GRIFFIN G., HOLUB A., PERONA P.: Caltech-256 Object Category Dataset.



Save Directory: http://srl.csdl.tamu.edu/userstudy/ mathsigns

Study Title: Math Sign Verification

User Information

Item 1: Age

Item 2: Occupation

Shape Information

Description 1:	Addition	Directory:	add	NumToVerify:	3
Description 2:	Subtraction	Directory:	sub	NumToVerify:	3
Description 3:	Multiplication	Directory:	mult	NumToVerify:	3
Description 4:	Division	Directory:	div	NumToVerify:	3
Description 5:	None of these	Directory:	none	NumToVerify:	0

Figure 11: *Screenshot showing the options we would choose for our example verification study.*

[Gro96] GROSS M.: The electronic cocktail napkin: A computational environment for working with design diagrams. *Design Studies 17*, 1 (1996), 53–69.

[HD05] HAMMOND T., DAVIS R.: LADDER, a sketching language for user interface developers. *Computers & Graphics 29*, 4 (2005), 518–532.

[HEP*] HAMMOND T., EOFF B., PAULSON B., WOLIN A., DAHMEN K., JOHNSTON J., RAJAN P.: Free-sketch recognition: Putting the CHI in sketching. In *CHI '08: Proceedings of the SIGCHI conference on human factors in computing systems*.

[KS02] KURTOGLU T., STAHOVICH T.: Interpreting schematic sketches using physical reasoning, 2002.

[KS04] KARA L. B., STAHOVICH T. F.: Hierarchical parsing and recognition of hand-sketched diagrams. In *UIST '04: Proceedings of the 17th annual ACM symposium on user interface software and technology* (New York, NY, USA, 2004), ACM, pp. 13–22.

[OAD04] OLTMANS M., ALVARADO C., DAVIS R.: ETCHA Sketches: Lessons Learned from Collecting Sketch Data. *Making Pen-Based Interaction Intelligent and Natural* (2004).

[PH08] PAULSON B., HAMMOND T.: Paleosketch: Accurate primitive sketch recognition and beautification. In *IUI '08: Proceedings of the 13th international conference on intelligent user interfaces* (New York, NY, USA, 2008), ACM, pp. 1–8.

[RTMF05] RUSSELL B., TORRALBA A., MURPHY K., FREEMAN W.: LabelMe: A Database and Web-based Tool for Image Annotation. *MIT AI Lab Memo AIM-2005-025 1* (2005), 1–10.

[Rub91] RUBINE D.: Specifying gestures by example. *SIGGRAPH Comput. Graph. 25*, 4 (1991), 329–337.

[Sun08] SUN: FAQ applet security. website. http://java.sun.com/sfaq/, 2008.

[vAD04] VON AHN L., DABBISH L.: Labeling Images With a Computer Game. *Proceedings of the SIGCHI conference on human factors in computing systems* (2004), 319–326.

[vALB06] VON AHN L., LIU R., BLUM M.: Peekaboom: A game for locating objects in images. In *CHI '06: Proceedings of the SIGCHI conference on human factors in computing systems* (New York, NY, USA, 2006), ACM, pp. 55–64.

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