EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2009)
C. Grimm and J. J. LaViola Jr. (Editors)

The Effect of Task on Classification Accuracy: Using Gesture Recognition Techniques in Free-Sketch Recognition

Martin Field[1], Sam Gordon[1], Eric Peterson[2], Raquel Robinson[1], Thomas Stahovich[2], Christine Alvarado[1]

[1]Harvey Mudd College, Claremont, CA
[2]University of California, Riverside

Abstract

Generating, grouping, and labeling free-sketch data is a difficult and time-consuming task for both user study participants and researchers. To simplify this process for both parties, we would like to have users draw isolated shapes instead of complete sketches that must be hand-labeled and grouped, and then use this data to train our free-sketch symbol recognizer. However, it is an open question whether shapes draw in isolation accurately reflect the way users draw shapes in a complete diagram. Furthermore, many of the simplest shape recognition algorithms were designed to recognize gestures, and it is not clear that they will generalize to freely-drawn shapes. To answer these questions, we perform experiments using three different recognizers to measure the effect of the data collection task on recognition accuracy. We find that recognizers trained only on isolated shapes can classify freely-sketched shapes as well as the same recognizers trained on free-sketches. We also show that user-specific training examples significantly improve recognition rates. Finally, we introduce a variant of a popular and simple gesture recognition algorithm that recognizes freely-drawn shapes as well as a highly-accurate but more complex recognizer designed explicitly for free-sketch recognition.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information interfaces and presentation]: User Interfaces—Interaction styles I.5.2 [Pattern recognition]: Design methodology—Classifier design and evaluation I.5.5 [Pattern recognition]: Implementation—Interactive systems

1. Introduction

Having realistic training data is essential to developing robust recognition algorithms. Obtaining such data for sketch recognition can be an arduous process. Public data sets are emerging [OAD04,HN05,WSA07,PWJH08,BPGW08], but these necessarily represent a limited number of domains and often include only shapes drawn in isolation, rather than complete sketches.

Many of the most robust and complete sketch recognition systems today (e.g. [ZBLF08] and [LdSP*08]) focus on gesture or isolated symbol recognition rather than the more difficult problem of free-sketch recognition. Such systems constrain users to draw shapes one at a time, avoiding the considerable difficulties of finding the boundaries between them. Gesture recognition additionally constrains the user to draw each shape with a specific style (e.g., users might be required to draw rectangles with a single clockwise pen stroke), minimizing drawing style variation.

Because of these simplifications, collecting training data for gesture recognizers is (relatively) straightforward. Users can be prompted to draw each gesture individually, without need for any manual processing or labeling. Because of the limited variation in drawing style, a recognizer trained on data from one set of users is likely to work well for other users. Additionally, many gesture recognition algorithms (e.g. [WWL07]) can be fine-tuned for an individual user by simply having them draw a few examples.

Obtaining training-data for free-sketch recognition systems is considerably more difficult than for gesture-based systems. In recent years, researchers have developed several tools to facilitate the collection and labeling of data [WSA07, PWJH08, BPGW08]. However, even with these tools, processing the data is still a labor and time-intensive task because the pen strokes must be manually grouped into individual symbols and labeled. Furthermore, most of these tools are designed for researchers. It would be impractical for an end-user to manually group and label the symbols in a free-sketch to improve a system's recognition performance.

For these reasons, researchers frequently obtain recognition training data for free-sketch recognition systems using the same simplified data collection process suitable for gesture-based systems (e.g., [KS05, HN04]). In the current work, we seek to evaluate the soundness of this approach, explicitly examining the usefulness of symbols drawn in isolation for training shape recognizers for free-sketch recognition. The answer to this question is not obvious as drawing isolated symbols is not the same as drawing them in the context of a larger task. For instance, drawing a logic gate by itself is substantially different from designing a digital circuit. In the former case, the user's focus is on the drawing task, while in the latter, the user's focus is likely to be the *design* task.

We conducted a study in which participants were asked to perform sketching tasks of varying complexity. Specifically, participants were asked to draw logic gates in isolation, copy digital circuits, and synthesize and sketch new circuit designs. We then examined how recognition accuracy varied with the task used to obtain training data.

A second piece of this work explores how well simple gesture recognition algorithms can be applied to more complex recognition problems. We address the question: can existing gesture recognition algorithms easily be modified to perform well on shapes drawn freely?

We show that recognition accuracy is not highly dependent on the task users performed when generating training data. Simple isolated symbol data works just as well for training symbol recognizers as symbols extracted from complex sketches. We also show that including user-specific examples in the training process gives a significant boost to accuracy, but highly accurate recognizers can be produced even without user-specific examples. We also present a variant of the recently proposed and very popular $1 recognizer [WWL07] that works on freely-drawn shapes comprised of any number of strokes drawn in any order. While necessarily more complicated than the original, our variant maintains much of the simplicity of the original algorithm, can be implemented quickly and easily, and performs with comparable accuracy to a more complex algorithm designed specifically for recognizing freely-drawn shapes.

2. Background

The central focus of the present work is to determine if variations in drawing task may lead to variations in drawing style, thus impacting symbol recognition accuracy. Relatively few studies have examined how people draw, and no study that we are aware of has compared drawing styles across tasks. Oltmans et al. present some preliminary observations from sketches from multiple domains [OAD04]. Although collected in a laboratory, these sketches simulate natural drawing style as users were asked to produce an original design rather than copy an existing diagram or draw specific shapes. Alvarado and Lazzareschi analyze drawings made by students in a circuit design class as part of their routine course work [AL07]. They find that drawing styles vary across students and even across the sketches of a single student. On the other hand, Sezgin and Davis find that the order in which users draw strokes is relatively consistent across users when they are asked to repeatedly draw military course of action symbols in isolation [SD07]. Shilman et al. examine free-from handwritten notes in detail, specifically focusing on identifying structure and handwriting within these notes [SWR*03], but they do not examine the diagrams themselves. Finally Anderson et al. examine diagrams created by professors while lecturing [AAH*05], but again their work does not focus on how people draw diagrams.

Researchers have developed countless shape and gesture recognition algorithms over the last few decades. Some focus on shape (e.g. [HD04, AD04, LM05, KS05, WWL07] to name only a few) while others extract features from the shapes (e.g. [BPGW08, HN04, Rub91, FPJ02, LKS06], again to name only a few).

In the field of gesture recognition, two algorithms stand out as the most widely-used: the Rubine classifier [Rub91] and the recently proposed $1 recognizer [WWL07]. The Rubine classifier, the first widely used gesture recognizer, is a feature-based linear classifier, while the $1 recognizer is a nearest-neighbor classifier that compares points in the sampled pen-strokes for each gesture. The $1 recognizer has the advantage that users can easily train it to match their drawing styles by drawing a few examples of each gesture.

In this paper we compare extensions of these two recognizers to a robust image-based recognition algorithm for sketch recognition presented in [KS05]. This recognizer, also a nearest neighbor classifier, has been shown to be one of the best at recognizing symbols in the face of drawing style variations and also can be easily adapted to an individual user by having the user draw a few examples of each symbol.

3. Approach

Our goal is to determine how much a user's drawing style varies when performing different tasks. In particular, we want to know if data collected under one drawing task can be used to train a recognizer to be used under a different task.

In the work surveyed above, we find three general tasks researchers use to collect sketch data, each involving a different level of cognitive load: isolated shape drawing, diagram copying, and diagram synthesis. In the isolated shape drawing task, subjects are instructed to repeatedly draw isolated shapes, often while looking at examples of them. For this task, subjects likely focus primarily on how they are drawing, rather than on the meaning of the shapes. Copying complete diagrams, by contrast, requires additional thought about how the pieces of the drawing fit together. However, users can still perform this task without thinking about the semantics of the diagram. Finally, in the diagram synthesis task, users are asked to draw a complete diagram that satisfies specific requirements, such as drawing their family tree or designing a floor plan for their house. In this task, users must devote thought to the design task, and likely think less about the actual process of drawing shapes.

Collecting training data in the form of isolated shapes is relatively inexpensive, as this can be done without need for manual processing or labeling. Extracting training data from copied or synthesized sketches is considerably more difficult. Tools such as [WSA07, PWJH08, BPGW08] help, but the process of manually grouping strokes into individual symbols and labeling them is laborious (see [WSA07] for a complete discussion of the labeling process). Furthermore, ambiguities in a sketch can lead to labeling errors.

While collecting data with the isolated shape drawing task is clearly easier than with the other two tasks, it is unclear if such data represents the way users actually draw when performing complex tasks. To investigate this question, we conducted a user study in which participants were asked to provide data using all three tasks. We then trained and tested three different recognizers to determine the effect of the data-collection task on classification accuracy.

3.1. Data Collection

We collected digital circuit diagram sketch data from 24 subjects, all college students. We focus on digital logic diagrams because previous work found that students tend to vary their drawing style in this domain [AL07], and recognizing the symbols in this domain is particularly challenging because many symbols look similar. The participants sketched freely on a Tablet PC; the data collection program performed no recognition.

We asked subjects to perform the three tasks described above: isolated shape drawing (ISO), diagram copying (COPY), and diagram synthesis (SYNTH). In the ISO task we asked participants to draw ten isolated instances of each single gate: AND, OR, and NOT, the most common gates in digital circuit design. In the COPY task, participants were asked to copy two entire circuit diagrams (Figure 1). In the SYNTH task, participants were shown the following logical equations (one at a time)

$$Y1 = (\overline{(AB+C)} \oplus \overline{AC}) + B + C$$

$$Y2 = (\overline{A} + BC)(A\overline{B} + AC + B)$$

and told to create a circuit that satisfies each. This task simulates a real-world activity, as it is analogous to homework problems students typically complete in a digital circuit design class. For the copied and synthesized diagrams, we analyze only the AND, OR and NOT gates because there were

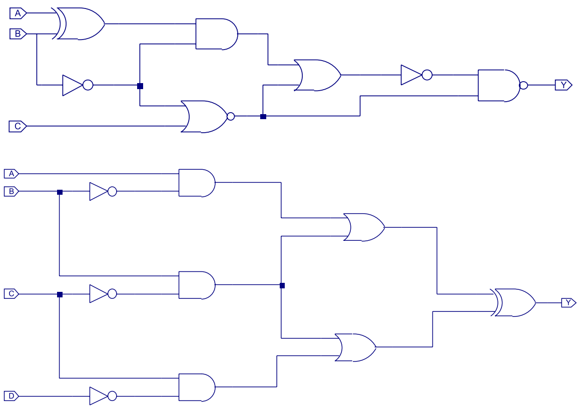

Figure 1: *The circuits participants copied in our study*

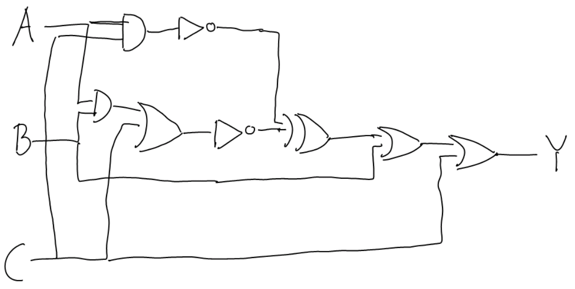

Figure 2: *One subject's sketch for the first synthesis equation*

not enough of the other gates for meaningful analysis. Figure 2 shows an example of one subject's sketch for the first equation above. The tasks were not necessarily performed in this order—four students completed the tasks in each of the six order permutations.

3.2. Recognizers

In our tests, we use three distinct recognizers: two modified gesture recognizers [Rub91, WWL07] and one standard image-based symbol recognizer [KS05]. We chose to include the modified gesture recognizers because we wanted to test how sensitive simple gesture recognition techniques are to drawing style.

The Rubine classifier [Rub91] is a standard and widely-used gesture classifier that works by extracting features from a single-stroke gesture and then training a linear classifier. We used a simple approach to adapt the Rubine classifier to multi-stroke symbols: we simply join the strokes together in temporal order. We then use the same features and the classification algorithm as in the original approach. We expect that this adaptation will be highly sensitive to stroke order, and thus we expect the modified Rubine classifier to be the least robust of our three recognizers.

We used a slightly more complex method to adapt the \$1 recognizer to handle multi-stroke symbols. The original single-stroke \$1 recognizer matches a stroke to a template in four stages. First, it resamples the stroke to have the same number of equally-spaced points as the template. Second, it performs an initial rotation, using the starting point of the stroke and its centroid to align the stroke with the template. Third, it centers and scales the stroke to match the center and scale of the template. Finally, optimization is used to refine the angular alignment between the stroke and template. This process finds the angular alignment that results in the smallest path distance between the stroke and the template.

Our first modification was to the resampling step. Rather than resampling each stroke to contain the same number of points, we resample each *shape* to contain the same number of points as the template, regardless of how many strokes it contains (we use 48 points in our templates). We achieve this goal by resampling each stroke to a variable number of points based on the ratio of its stroke length to the total length of the strokes in the shape, such that the total number of points from all the strokes totals 48.

The initial rotation, centering, and rescaling steps remain largely unchanged. However, instead of centering on the centroid of one stroke, we use the centroid of the whole symbol. Similarly, we resize the whole symbol to a 64×64 pixel bounding-box.

In the final step, the rotation search remains the same, but the distance function used to guide the search is heavily modified. As in the original algorithm, we require a one-to-one match between the points in the symbol to be classified and the points in the template, but we can no longer use drawing order to determine this match, as we want our algorithm to be robust to variations in drawing order. We instead search for the match that minimizes the sum of the distances between each pair of matched points. Calculating the global minimum would be prohibitively time-consuming, so we approximate a solution using simulated annealing. We begin the search using the simple greedy approach of matching each point to the nearest available point. We search nearby solutions by picking two points and swapping their matches.

More flexible methods of point-matching, such as Dynamic Time Warping (DTW), have been applied to gesture recognition [WWL07, KZ04]. The main difference between these flexible methods and our method is that they allow multiple points in one shape to be matched to a single point in the other. As noted in [KZ04], a flexible matching of points introduces additional ambiguity between similar templates and degrades the quality of the recognizer.

We also used the Image-Based Trainable Symbol Recognizer in [KS05]. Because this algorithm was originally designed to recognize multiple-stroke shapes, it provides a good contrast to the \$1 and Rubine classifiers. The image-based recognizer is based on template matching, similar to the \$1 recognizer, so it is very easy to add user-specific training examples. Hence, it is especially well suited to both aspects of our testing.

3.3. Comparison of Recognizers

Since we completed this work, the authors of the original \$1 recognizer have released (but not yet published) a version suitable for multi-stroke recognition (called the \$N recognizer)[†]. Their approach is similar to our own, with two differences. First, instead of using simulated annealing to match points in the test symbol to points in the template, they order the strokes and then use the same temporal matching techniques as above. To guard against drawing style variation, they search over the stroke order permutations. It is not yet clear which approach is superior (if either). A second important difference between their method and ours is that when they link strokes together, they also consider the ink that was drawn "in the air" from the end of one stroke to the beginning of the next. These virtual points then take away from the total number of points sampled from the actual ink drawn (because the total number of points remains fixed), giving their approach lower resolution and higher sensitivity to variation in shape.

Our modified \$1 recognizer is actually quite similar to the image-based recognizer. Both recognizers preprocess shapes by rescaling and centering them, but where \$1 resamples points, the image-based recognizer rasterizes points into a bitmap image. The next step for both is a rotation search. However, because the image-based recognizer performs its rotation search in polar coordinates, it must also perform a separate distance calculation in screen coordinates once the ideal angle has been located. To perform classification, both recognizers compare the unknown symbol to each of the training examples.

Although the general process is very similar for both recognizers, the small differences have implications for how each performs. The image-based recognizer is extremely robust to noise like overstroking, touch-up strokes, or differing numbers of strokes in the unknown symbol and the template. The modifications we made to the \$1 recognizer attempt to address touch-up strokes or differing numbers of strokes in each shape. However, the resampling process includes the endpoints of all strokes, so with more strokes, the points will be poorly distributed, leading to an increased distance between the unknown shape and template.

The advantage that our modified \$1 has over the image-based recognizer is its simplicity. We tried to stay as much in the spirit of the original \$1 recognizer as possible. The simulated annealing step requires only a small amount of extra

[†] http://depts.washington.edu/aimgroup/proj/dollar/ndollar.html

Activity (train-test)	Rubine	Dollar	Image
Synth-Synth	21	93	90
Copy-Synth	39	88	84
Iso-Synth	53	88	87
Synth-Copy	39	89	90
Copy-Copy	54	91	96
Iso-Copy	48	90	93
Synth-Iso	32	88	88
Copy-Iso	43	90	90
Iso-Iso	53	95	94

Table 1: *Recognition results for all three recognizers, in percent.*

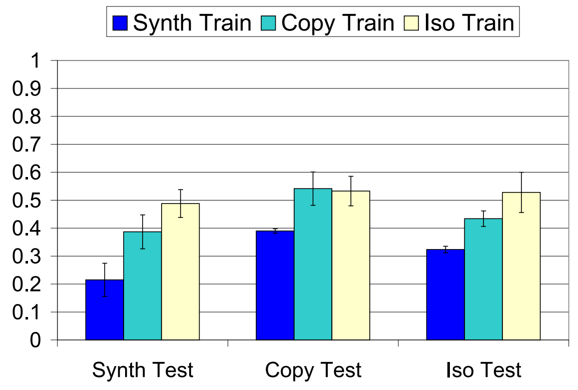

Figure 3: *Recognition results for the Rubine recognizer. Error bars show one standard deviation.*

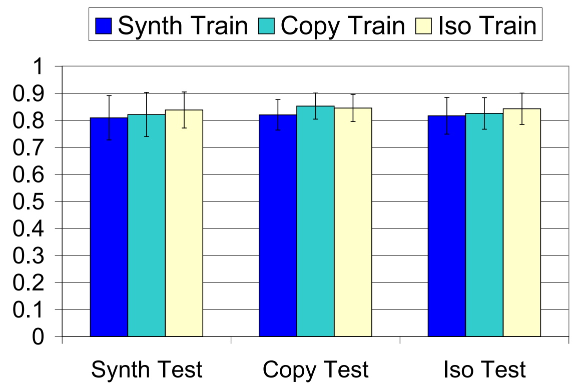

Figure 4: *Recognition results for the $1 recognizer. Error bars show one standard deviation.*

code, but adds a large amount of complexity. However, this complexity is still much simpler than the image-based recognizer, especially the rotation search in polar coordinates and the fact that our recognizer requires only one distance metric rather than the four the image-based recognizer requires.

4. Experiments and Results

Our goal with these experiments was threefold. First, we wanted to examine how task affects recognition accuracy. Second, we wanted to know how much impact user-specific training data has on recognition accuracy. Finally, we wanted to compare the modified versions of the simple gesture recognition algorithms with an image-based recognition algorithm for shapes drawn in the synthesis task (i.e. a task that gesture recognition was not designed to support).

4.1. The Effect of Task

To examine the first question, we separated the data into three categories based on task (ISO, COPY, SYNTH). We set aside data from 80% of the users (randomly selected) for training and used the data from the remaining 20% of the users for testing. Note that training and testing our recognizers on different user sets simulates the experience of a "factory trained" recognizer that cannot be customized to a particular user, which is the normal situation for most sketch recognition systems. We trained three separate versions of each recognizer on the training data, separated by task. That is, we trained one Rubine classifier on the ISO training set, a separate Rubine classifier on the COPY set, etc. The $1 and image-based recognizers are nearest neighbor classifiers, hence their running time is dependent on the number of training examples. To achieve reasonable performance, we randomly selected a small subset of the training set to use as templates for these recognizers.

After training, we tested each recognizer on the remaining 20% of the data in every category (i.e. three test runs for each recognizer). For example, we ran the ISO Rubine recognizer separately on the test data from the ISO, COPY and SYNTH tasks, for a total of three test runs with this recognizer. This process resulted in 9 distinct training-testing combinations (iso-iso, iso-copy, iso-synth, copy-iso, etc.).

Table 1 presents the average classification accuracy for all combinations over 25 trials in which we randomly selected a new 80/20 user split for training/testing. Figures 3, 4, and 5 show these same results graphically, including their standard deviations. For the two recognizers that perform at a reasonable level of accuracy ($1 and image), the task used for training has little if any effect on recognition accuracy (none of these differences are statistically significant using repeated measures ANOVA). In particular, training on SYNTH data does not appear to improve the accuracy, even when recognizing SYNTH data.

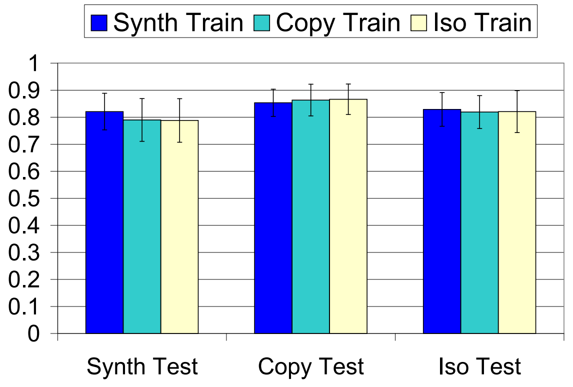

Figure 5: *Recognition results for the image-based recognizer. Error bars show one standard deviation.*

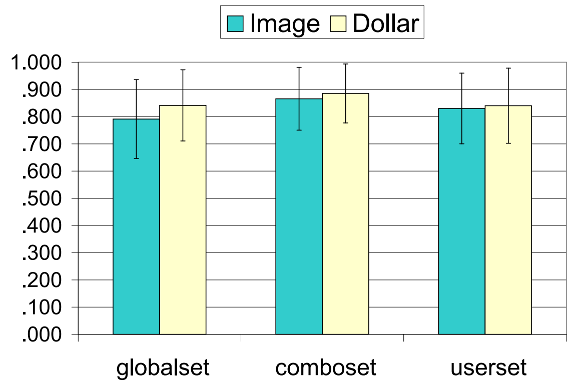

Figure 6: *Recognition results for the $1 and image-based recognizers with different training sets. Error bars show one standard deviation.*

4.2. The Effect of User-Specific Data

To examine the second question, we measured recognition performance as we increased the amount of user-specific training data used to train the recognizer.

In our first user-data experiment, our goal was to measure the impact of easily collected user-specific data on recognition accuracy in real-world tasks (i.e. the SYNTH task). In other words, we imagine that the user has acquired a factory-trained recognition system (such as the system produced with the training above), and now has the option of training the system by drawing isolated shapes. We want to know how much (if at all) this user-specific data will improve classification accuracy. We focus on only the $1 and image-based recognizers because of the Rubine classifier's poor performance in the first tests.

We again split the users into a training set and a testing set as above; however, this time we use only the SYNTH data for testing and only the ISO data for training. Again, this choice simulates the situation where the base recognizer was trained in the factory using ISO data from other users, and then trained by the user's individually-drawn symbols. Note that we use ISO data for the "factory trained" recognizer for consistency, but we showed above that any of the datasets produce an equally robust recognizer.

We then generated three of each recognizer ($1 and image-based) for each user by training the recognizers on training sets containing different amounts of user-specific data. All of our trials use a constant 5 training examples. The first training set, called the GlobalSet, contains 5 training examples randomly selected from the 80% of users in the training set. (Recall that we use only ISO data for training.) Because the GlobalSet does not contain user-specific training examples, we used the same GlobalSet for each testing user. Next, we created a ComboSet recognizer for each user in the testing set by randomly selecting 2 of the GlobalSet training examples and replacing them with 2 randomly selected examples from the testing user's data. We removed the same 2 GlobalSet examples for each user, so the ComboSets differed per-user only in the 2 user-specific examples. Finally, we created a UserSet recognizer for each user composed solely of 5 examples from the user in question. Hence, the UserSet recognizers are unique to the users they were created for.

Recognizer	Comparison	P - Bonf.
$1	GlobalSet and ComboSet	**0.001**
	UserSet and ComboSet	**0.001**
	GlobalSet and UserSet	1.000
image-based	GlobalSet and ComboSet	**<0.001**
	UserSet and ComboSet	0.074
	GlobalSet and UserSet	0.144

Table 2: *Repeated measures ANOVA Post Hoc results comparing user-independent and user-dependent recognizers*

Figure 6 shows the results of this experiment, averaged over all runs and all users. While the ComboSet appears to produce the best recognizer, the graph obscures the variations for each individual user and each individual trial. When we focus on the individual trials, we find that differences in the training sets lead to significantly different classification accuracies. Table 2 gives the results of our repeated measures ANOVA. We tested our data to verify that it satisfies the assumptions made by ANOVA and used the Bonferroni correction to reduce the chance of type 1 error.

With the knowledge that user-specific ISO data does in fact improve recognition accuracy, we then investigated whether user-specific SYNTH data might improve recognition accuracy even more for SYNTH recognition. For this experiment, we again focused only on the SYNTH task for recognition and we focused on training sets that have a mix of user-specific and generic data, as in the ComboSet above.

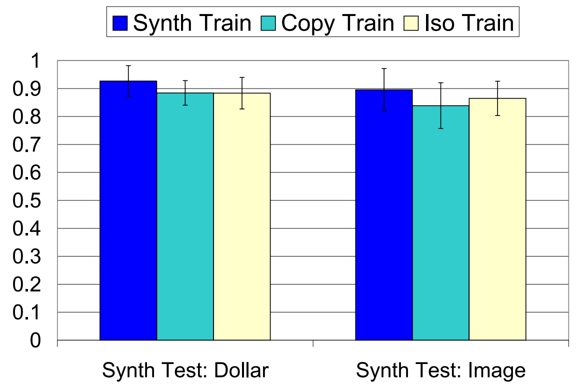

Figure 7: *Recognition results from the $1 and image recognizers, trained on different tasks using some user-specific data. Error bars show one standard deviation.*

Recognizer	Comparison	P - Bonf.
$1	Iso train and Copy train	1.000
	Iso train and Synth train	**0.011**
	Copy train and Synth train	**0.019**
image-based	Iso train and Copy train	0.573
	Iso train and Synth train	0.369
	Copy train and Synth train	**0.033**

Table 3: *Repeated measures ANOVA Post Hoc results comparing task for user-dependent recognizers*

We generated three different training sets for each user, one for each task. All three sets contain three generic shape examples and two user-specific examples, as above in the ComboSet. The sets differ only in which tasks subjects performed when generating this data: in the first set we use only SYNTH data, in the second we use only COPY data, and in the third we use only ISO data. Note that the results from the ISO-trained data will be identical to the results from the ComboSet in Figure 6.

Figure 7 shows the results of this experiment for both recognizers. This graph shows that, unlike in the "factory-trained" case, training on the SYNTH data does in fact give a slight but statistically significant boost to recognition performance over training on the other tasks. For the $1 recognizer, it improves performance over the ISO or the COPY data. For the image-based recognizer, training on the SYNTH data improves performance over training on the COPY data (but not the ISO data). Table 3 gives the results of our repeated measures ANOVA Post Hoc tests using Bonferroni correction.

4.3. The Effect of the Recognizer

Finally, in all cases, the $1 and image-based recognizers performed comparably and with reasonably high accuracy. Unfortunately, we ran the image-based and $1 tests over different random samples of data, so we cannot directly compare performance using a repeated-measures test (which would be more sensitive to differences in accuracies). Using aggregate statistics, there is no significant difference between the results given by the two classifiers. This result illustrates the promise of our new modified $1 recognizer for non-gesture recognition tasks.

5. Discussion

In this section we consider the implications of our findings on the general task of free-sketch recognition.

First, it seems clear that not all gesture recognizers will generalize easily to non-gesture recognition tasks, as evidenced by the poor results of the Rubine recognizer. The simple methods we used to generalize the Rubine classifier to multiple-stroke shapes did not produce an effective free-sketch recognizer likely because of variations in drawing order and sensitivity of some of the Rubine features to stroke end-points. Because the Rubine accuracy is so low, we focus on the $1 and image-based results for the rest of our discussion.

The $1 recognizer performed very well across the board. The accuracies of the modified $1 even begin to approach the accuracies originally presented by Wobbrock et al. [WWL07]. The $1 recognizer was presented originally as an easy-to-implement, highly accurate recognizer for gesture recognition. Our modification has produced an easy-to-implement, highly accurate recognizer for symbol recognition in freely-drawn sketches.

Our result that training task does not significantly affect recognition performance of "factory-trained" recognizers when they are used for other tasks is very good news for sketch recognition researchers. As discussed throughout this paper, isolated shapes are much easier to collect and label than truly freely-drawn sketches. Researchers can use this type of data to train their shape recognizers and these shape recognizers can be adapted to individual users by having them draw a few examples of each shape in isolation.

Our second experiment indicates that it is important to include user-specific training examples whenever possible. The ComboSet performs significantly better than the GlobalSet for both recognizers. Additionally, for the $1 recognizer, the ComboSet also significantly outperformed the UserSet. This result can be explained by variation within individual users' drawing styles. The ComboSet contains a wider breadth of examples (from the GlobalSet contribution) than the UserSet, and is hence able to better accommodate this variation. On the other hand, for the image-based recognizer, we cannot safely say that the ComboSet outperformed the UserSet. This result is likely because the image-based recognizer is more robust than our modified $1, so there is less benefit from the breadth of the GlobalSet.

The results of our third test indicate that, unlike in the "factory-trained" case, collecting task-specific user training data can aid recognition accuracy even more than simply collecting user-specific isolated shape data. While this type of data seems difficult to collect (no user will group and label their complete sketches), we might be able to leverage the error correction process to obtain this data. For example, every time the user corrects a classification error in their freely-drawn sketch, the system can record this example and use it to train the recognizer.

The high performance of the modified $1 and image-based classifiers shows that while drawing *style* may vary between tasks, drawing *shape* remains similar. We expect that our task-specific results would generalize to any other shape-based recognition algorithm, but not to algorithms that depend on stroke drawing order.

Finally, our results show the importance of using easily retrainable recognizers for free-sketch recognition. The $1 and image-based recognizers support online learning with very little effort or time required. Furthermore, these recognizers only need a small number of examples for training, requiring little extra work from the user to benefit from increased recognition accuracy. However, there are many recognizers for which it is not feasible to incorporate user-dependence in a user-friendly way. For example, some recognizers might need to rerun the entire training process, requiring a long computation. Other recognizers might need to collect many examples from a user before any effect on recognition would be noticed.

6. Conclusion

We have shown that it is possible to train a highly accurate recognizer using only isolated shapes as training data. This is important because isolated shape data is much easier to collect than naturally drawn shapes. It is also much easier to group and label isolated shapes than complicated diagrams. We have also shown that whenever possible, user-specific examples should be incorporated into training. For classifiers like the image-based and $1 recognizers that support online learning, user-specific examples are an easy way to improve accuracy.

References

[AAH*05] ANDERSON R., ANDERSON R., HOYER C., PRINCE C., SU J., VIDEON F., WOLFMAN S.: A study of diagrammatic Ink in lecture. *Computers and Graphics: Special Issues on Pen-based Computing* (2005).

[AD04] ALVARADO C., DAVIS R.: Sketchread: A multi-domain sketch recognition engine. In *Proc. UIST* (2004).

[AL07] ALVARADO C., LAZZARESCHI M.: Properties of real-world digital logic diagrams. In *Proc. of the 1st International Workshop on Pen-Based Learning Technologies (PLT-07)* (2007).

[BPGW08] BLAGOJEVIC R., PLIMMER B., GRUNDY J., WANG Y.: A data collection tool for sketched diagrams. In *5th Eurographics Conference on Sketch Based Interfaces and Modeling (SBIM Š08)* (2008).

[FPJ02] FONSECA M. J., PIMENTEL C., JORGE J. A.: Cali: An online scribble recognizer for calligraphic interfaces. In *Sketch Understanding, Papers from the 2002 AAAI Spring Symposium* (2002), pp. 51–58.

[HD04] HAMMOND T., DAVIS R.: Automatically transforming symbolic shape descriptions for use in sketch recognition. *Proceedings AAAI* (2004).

[HN04] HSE H., NEWTON A.: Sketched symbol recognition using zernike moments. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* (2004), vol. 1.

[HN05] HSE H., NEWTON A. R.: Recognition and beautification of multi-stroke symbols in digital ink. *Computers and Graphics 29*, 4 (2005).

[KS05] KARA L. B., STAHOVICH T. F.: An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers and Graphics: Special Issue on Pen-Based User Interfaces* (2005).

[KZ04] KRISTENSSON P.-O., ZHAI S.: Shark2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of UIST* (2004), pp. 43–52.

[LdSP*08] LEE W., DE SLIVA R., PETERSON E. J., CALFEE R. C., STAHOVICH T. F.: Newton's pen—a pen-based tutoring system for statics. *Computers and Graphics Special Issue on Sketch-Based Interfaces and Modeling 32*, 5 (October 2008), 511–524.

[LKS06] LEE W., KARA L. B., STAHOVICH T. F.: An efficient graph-based symbol recognizer. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2006).

[LM05] LEARNED-MILLER E.: Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 3 (2005), 236–250.

[OAD04] OLTMANS M., ALVARADO C., DAVIS R.: Etcha sketches: Lessons learned from collecting sketch data. In *Making Pen-Based Interaction Intelligent and Natural* (2004).

[PWJH08] PAULSON B., WOLIN A., JOHNSTON J., HAMMOND T.: Sousa: Sketch-based online user study applet. In *Proceedings SBIM* (2008).

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

[SD07] SEZGIN T. M., DAVIS R.: Sketch interpretation using multiscale models of temporal patterns. *IEEE Computer Graphics and Applications 27*, 1 (2007), 28–37.

[SWR*03] SHILMAN M., WEI Z., RAGHUPATHY S., SIMARD P., JONES D.: Discerning structure from freeform handwritten notes. In *Proc. ICDAR* (2003).

[WSA07] WOLIN A., SMITH D., ALVARADO C.: A pen-based tool for efficient labeling of 2d sketches. In *Proceedings of SBIM* (New York, NY, USA, 2007), ACM, pp. 67–74.

[WWL07] WOBBROCK J. O., WILSON A. D., LI Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2007), ACM, pp. 159–168.

[ZBLF08] ZELEZNIK R. C., BRAGDON A., LIU C.-C., FORSBERG A.: Lineogrammer: creating diagrams by drawing. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology* (New York, NY, USA, 2008), ACM, pp. 161–170.