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

From Paper to Machine: Extracting Strokes from Images for use in Sketch Recognition

Pankaj Rajan† and T. Hammond‡
Texas A&M University
Dept. of Computer Science
College Station, TX 77843-3112

ABSTRACT

Sketching is a way of conveying ideas to people of diverse backgrounds and culture without any linguistic medium. With the advent of inexpensive tablet PCs, online sketches have become more common, allowing for stroke-based sketch recognition techniques, more powerful editing techniques, and automatic simulation of recognized diagrams. Online sketches provide significantly more information than paper sketches, but they still do not provide the flexibility, naturalness, and simplicity of a simple piece of paper. Recognition methods exist for paper sketches, but they tend to be domain specific and don't benefit from the advances of stroke-based sketch recognition. Our goal is to combine the power of stroke-based sketch recognition with the flexibility and ease of use of a piece of paper. In this paper we will present a stroke-tracing algorithm that can be used to extract stroke data from the pixilated image of the sketch drawn on paper. The presented method handles overlapping strokes and also attempts to capture sequencing information, which is helpful in many sketch recognition techniques. We present preliminary results of our algorithm on several paper-drawn, hand-sketched, scanned-in pixilated images.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Image processing and Computer Vision]: Edge and Feature Detection).

1. Introduction

Sketching is a natural way of devising and communicating ideas. Sketching is a universal language understood across varied cultures. Sketches can convey ideas that would be difficult to describe with only text. Designers in a variety of fields rely on sketches early in their idea development due to the ease in which ideas can be quickly conveyed.

Tablet PCs, once the domain of the professional graphic artist, have dropped in price and thus are more attainable. With their increase in availability, researchers have begun to experiment with different uses of the input medium. Sketch recognition is a research field whose goal is to identify and understand the sketches of a user so that the sketch can be used as part of a computational process, for instance by simulating the drawn diagram. A simple example is of the mechanical engineer who designs gear-train systems. During the design phase, the engineer may make quick, rough sketches on a piece of paper until he decides how the part should be designed. In order for a simulation system to understand his diagram, current technology requires the designer to manually enter his designs into a CAD system using a mouse and menu based system.

† e-mail: pankaj@cs.tamu.edu

‡e-mail: hammond@cs.tamu.edu

Many current sketch systems available use sketches to build the foundation for the complex CAD designs [ML06, JLA04, LM05, LS96].These sketch systems, and others [AD04, HD05, KS04, GD96], require users to draw their sketches directly on the computer screen. Such systems use stroke-based sketch recognition techniques that utilize the stroke point information (*x*, *y*, and *time* of each point) for the recognition process.

Sketch recognition researchers argue that they are combining the "freedom of a hand-drawn sketch with the power of a computer" [Ham07], but we will make the bold claim that not a single sketch recognition researcher out there will deny that a piece of paper is still more intuitive, more flexible, and more accessible than any existing Tablet PC. Paper is still the predominant medium for creating sketches.

Serendipity can occur anywhere and at anytime. Good ideas do not simply come to users when they are sitting in front of a computer. Ideas can come to a person while traveling in a plane, hiking, at a restaurant (let us not forget the inspiration provided to us by the *electronic cocktail napkin* [GD96]), and many other unexpected places. We may not always have a Tablet PC at our fingertips, but a sheet of paper and a pen is almost always readily available (and much lighter to carry). Eventually, though, we may get back to a computer, and wish to take advantage of the capabilities of a recognized sketch. Additionally, there exist millions of legacy design sketches that could benefit from stroke-based sketch recognition. Apart from this,

© The Eurographics Association 2008.

© The Eurographics Association 2008.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

studies have also shown that people prefer to use paper and pen during the early design stage [FBC05].

Paper-based capture systems, such as the Anoto pen [Ano], provide a pen and paper notebook interface for inputting stroke data. We have tried this interface, and the users we have tested it with have complained that they did not like being tied to the particular pen and paper supplied with the system, and that they preferred to have their drawings left unrecognized and have to reenter them into the computer when necessary than to have to carry around the "bulky" notebook and "cheap-feeling" pen. One user pulled out a folded piece of paper and a pen from his pocket and said, "This is all I need."

The aim of the presented work is to extract stroke information from a pixilated image so that the strokes may be recognized using currently available stroke-based techniques. It is important to mention here that our current method does not gather pen speed information, which is used in a wide number of current sketch recognition techniques [SSD06, SD05, PH08], but only labels the time values for each point sequentially using a counter. Nonetheless, several low-level [KK06, WWL07, WEH08, HEPW08] and higher-level [HD05, AD04] sketch recognition techniques exist that do not take advantage of speed information. In this paper, we will deal only with the problem of stroke extraction from a pixilated image, and leave the integration into an existing sketch recognition system for future work.

2. Related works

Any system that requires the extraction of stroke data from an image has to understand which pixels are the parts of the stroke and which are not. The system has to understand the natural sketching mode of users, which can be utilized to extract the perceptually meaningful shape. In [BCFBO06], authors describe a co-occurrence-matrix based approach to simplify the drawings and obtain smaller number of vectors to describe the drawing more meaningfully. Most of the literature in the field describes a term called saliency measure, which selects the pixel that favors long, smooth and continuous strokes in the image.

The use of global and local saliency measures has been well debated in the literature, though some [GM93] have argued that global features best represent the actual perception of the stroke. As such, they have used probabilistic measurements based on machine learning techniques to calculate the contribution of the single unit edge to the global perception of the stroke. In [SU88], the authors describe a saliency measure based on the curvature and curvature variation of the network of closely connected components. Using an iterative approach, similar to dynamic programming, they were able to extract out the components of the shape that got the maximum saliency measure. Likewise, in [GT99], the authors describe saliency measures as a gain of energy because of addition of the new edge to the curve being traced. In [NM04], a pen-paper based system UDSI is described which takes as an input a scanned UML diagrams drawn on paper and presents the user with the interpretation of the scanned diagram for incorporating it into technical digital documents. Though their goal may appear similar to ours, their aim and implementation differs from ours in that they have in place domain specific rules to recognize specific domain shapes. Our goal is to recognize the original stroke path for any hand-drawn shape to enable existing stroke-based recognizers to recognize paper-sketched shapes**.**

In [AT89], a method based on grouping and clustering the set of points underlying perceptual patterns is described. A saliency measure based on figural closure was proposed in [Sau03]. Their algorithm traces the path to find the closed contours based on the smoothness and continuation constraints. Other work dealing with similar problems is [BCFB07] which talks about converting the paper scribble to single vector lines. Our method differs from these techniques in that they are trying to recognize primitives (such as lines) in the latter and/or merge distinctly-drawn strokes in the former, whereas our goal is to match, as best as we can, the user's original stroke path. Thus, our method is not limited to a small set of chosen primitives, but can take advantage of existing online stroke-based techniques. In this paper, we have presented a method that utilizes both the global and local characteristics of the stroke being traced to arrive at the perceptually closest match to the stroke that the user would have intended to draw.

3. Approach

Our approach consists of three basic steps. First, as in most image processing techniques, we have to preprocess the image to remove noise. Second, we thin the pen strokes to a thickness of only a single pixel. Three, we trace the line using both the local and global characteristics of the previously identified stroke points to determine the future points of the stroke.

3.1 Noise Removal

In order to process images, we currently assume that the sketch is made on good quality white paper using a thick black marker. These images are scanned into a computer to be later recognized as vectorized strokes. As scanned images often contain noise, we first remove the noise. We remove the noise from the image using global thresh-holding and median filtering. Global thresh-holding converts the image to a binary black-and-white image. Median filtering "replaces the value of the pixel by the median of the grey-values in the neighborhood of that pixel" [GW02]. Median filtering works very well on impulse noise, a.k.a. salt-and-pepper noise. All of the hand-drawn figures in this paper except Figure-1 and Figure-2 were hand-drawn and scanned in.

Our scanner produced relatively clean drawings. In order to show our ability to handle a greater amount of noise, Figure-1 (left) shows a hand drawing artificially corrupted by a Gaussian noise with a standard deviation of 0.02 and a mean 0. Figure-1 (right) shows the same drawing artificially corrupted by salt-and-pepper noise with a noise density of 0.02. Figure-2 (left) shows the image obtained

© The Eurographics Association 2008.

© The Eurographics Association 2008.

after global thresholding and median filtering to remove both types of noise.

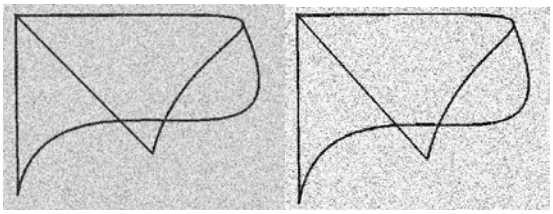

Figure 1: ***LEFT:*** *A hand-drawn image artificially corrupted by a Gaussian noise with a standard deviation of 0.02 and a mean of 0.* ***RIGHT:*** *Same image corrupted by salt-and-pepper noise with a noise density of 0.02.*

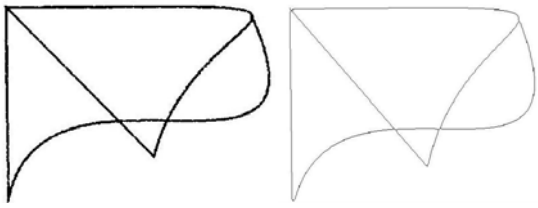

Figure 2: ***LEFT:*** *The image obtained after performing noise removal using (3x3) median filtering and thresholding of the hand-drawn image shown in the left and right of Figure-1.* ***RIGHT:*** *The image obtained after performing morphological thinning.*

3.2 Stroke thinning

Given the varying pressure of a pen, the strokes of the hand-drawn images have a varying thickness. Stroke thickness can cause difficulties when trying to determine the intended direction of stroke. Thus, we thin the pen strokes to a thickness of a single pixel. In order to thin our noise-free images, we use morphological thinning as described in [HS92] and [Pra91] to obtain a single pixel connected linkages as shown in Figure-2 (right). The image obtained after morphological operations is binary (black and white).

3.3 Stroke extraction

After the strokes are thinned into a single pixel thickness, we attempt to extract and distinguish each separate stroke. Step 1 in stroke extraction is to identify a starting point from which to begin stroke extraction. As our starting point, we choose the black pixel closest to the top left corner of the image. That first point is selected to belong in the first stroke and stored in an array called *stroke history*, which maintains the record of the pixels designated as part of the stroke under consideration. Initially, all drawn (black) points are labeled as '1', and all white points are labeled are '0'. A flag for each pixel added to the *stroke history* is changed to '0' so that it becomes invisible to the algorithm. Step 2: We trace the stroke using that pixel as the starting point; continuously selecting pixels to add to the *stroke history* is based on following rules:

1. If the 8-pixel neighborhood around the current pixel is empty except for one neighboring pixel, we move to that pixel and add that pixel to the *stroke history*. Figure-3 shows the transition when there is only one choice available.

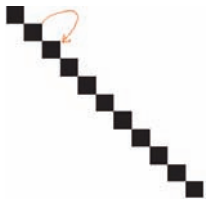

Figure 3: *The transition when just one pixel is available*

2. If there are multiple pixels to choose from in the 8-pixel neighborhood, we identify this pixel as a *point of ambiguity*. Figure-4 shows an example *point of ambiguity*. If the number of elements (pixels) in the stroke history is less than a certain threshold (12 for our case), the next pixel is selected by method called *local solver* that selects the pixel with minimum angular deviation from the direction given by the last two pixels of the *stroke history*, as shown in Figure-5.

Figure 4: *A point of ambiguity is shown by arrow.*

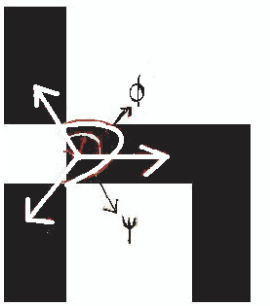

Figure 5: *Showing the approach to solve local ambiguity. Since, inner angle* Φ *is less than the outer angle* Ψ *, the pixel pointed by the lower angle* Φ *would be selected over pixel pointed by* Ψ *.*

© The Eurographics Association 2008.

© The Eurographics Association 2008.

3. If there are multiple pixels in the 8-pixel neighborhood of the last pixel of the *stroke history* and the number of elements in the stroke history is greater than the threshold (12 in our case), we declare the current pixel to be a *global point of ambiguity*. An example of a *global point of ambiguity* is shown in Figure-6. We then send the both the pixel that is the *point of ambiguity* and the last 12 points of the stroke history to the process called *global pixel selector*, which selects the next pixel based on the global direction of the previous points. The structure of the pixel selector is described in the next section.

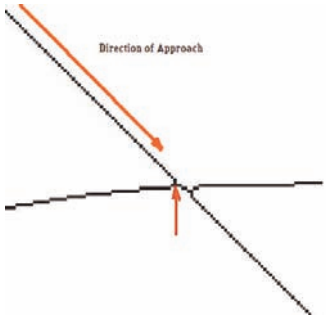

Figure 6: *The arrow labels a global point of ambiguity.*

3.4 Pixel Selection based on Global Characteristics

The pixel selector is the process that selects the next pixel based on the global characteristics of the stroke being extracted. Rather than only look at the local direction (single pixel neighborhood), the global direction is determined by looking at a string of 12 previous pixels. The input to the process is the *point of ambiguity* and the previous 12 pixel points in the *stroke history*. The pixel selector then completes the stroke at that point and starts multiple new strokes (called the *stroke futures*), one beginning from each of the possible directions in the 8-pixel neighborhood. If the maximum length of a *stroke future* is less than a certain threshold (8 pixels) after tracing the line as far as possible, then that stroke future is discarded and considered to be a tail. In Figure-7, two *stroke futures* exist, with the figure labeling the *stroke feature* that will be identified as a tail and discarded.

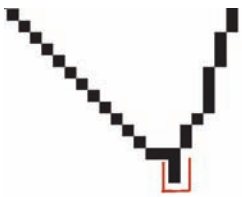

Figure 7: *A point of ambiguity with two possible stroke futures. The stroke future that will be labeled as a tail and discarded is circled in red pen.*

It is well understood that the direction corresponding to the actual direction of the stroke will have the minimum direction deviation from the previous 12-pixels in the *stroke history*. In order to determine the general direction of the previous 12-pixels in the *stroke history* and how they correspond to the *stroke future*s, we use a well-known method in the statistical pattern recognition called Principal Component Analysis (PCA) [Bis96].

The principle component is defined as the eigenvector corresponding to the maximum eigenvalue of the covariance matrix. The principle component thus captures the direction of maximum variance, which in our case is the direction of the stroke. Mathematically we can write covariance matrix as:

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$, where X and Y are the coordinates and n=12. Given the principle component of the s*troke history* ($\vec{P}_h$) and the principal component of the *stroke future* ($_i\vec{P}_f$) where *i* corresponds to the *stroke future* direction considered. We calculate the dot product of the two as:

$$DP_i = (\vec{P}_h \bullet_i \vec{P}_f).$$

The *stroke future* which corresponds to the

$$\max_{1 \le i \le n}(\|DP_i\|),$$

where *n* is the number of *stroke futures*, is chosen, and the initial pixel of that *stroke future* becomes the next pixel in the original (previous) stroke. That pixel is added to the *stroke history*, the other *stroke futures* are discarded, and we continue with the original stroke in the direction of the chosen *stroke future.*

It is quite possible that while selecting of the *stroke future* pixels, we may encounter additional *local points of ambiguity* .The local points of ambiguity are the points in the stroke futures which lead to multiple directions. An example of local ambiguity, relative to a global ambiguity is shown in figure 8.

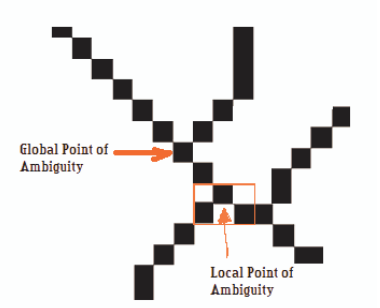

Figure 8: *A point of local ambiguity shown relative to the global ambiguity. Note that in the stroke future, there is again an ambiguity, which is local in nature.*

© The Eurographics Association 2008.

© The Eurographics Association 2008.

These *local points of ambiguity* are resolved using our *local solver* method that treats the *stroke future* as a completely new stroke, which necessarily contains fewer than 12 pixels (in range of 2-6 pixels). Thus, the continued direction is chosen using only local direction information, examining only the previous two pixels of the *stroke future* to determine the next point. If the previous 2 pixels are not yet available in that *stroke future* (i.e., the *local point of ambiguity* is very close to the *global point of ambiguity*), then we will chose two pixels from the global *stroke history* array. The process is iterative and continues till all the strokes in the image are extracted.

3.5 Stroke Merging

Since we have overlapping strokes, it is common that a stroke may lose a pixel to another stroke, and hence, during extraction process we may prematurely halt the traversal of a stroke. In order to deal with it, we use a process called *stroke merging*.

The *stroke-merging* algorithm computed the distance between the endpoints of distinct strokes. After this step, the following distance metrics are computs:

1. *Start-Start Distance:* It computes the distance between all of the starting points of the extracted strokes.
2. *End-Start Distance:* It computes the distance between the endpoint of each stroke with the starting point of the other extracted strokes.
3. *Start-End Distance*: It computes the distance between the start point of the first stroke and the end point of the second stroke being considered.
4. *End-End Distance:* It computes the distance between the endpoints of the extracted strokes.

Since we intend to obtain strokes that are perceptually close to what the user actually drew, we examine both the distance as well as the angle between the two strokes when merging them.

Users tend to draw long straight continuous strokes rather than short angular strokes; as such, when there are several nearby strokes, the strokes formed by merging strokes with similar angles are favored. In order to deal with this issue, we have defined a confidence function that gives a confidence value between 0-1 that contains both the distance and the angle difference. A higher confidence value means that the two strokes are the most probable candidates for merging among all other possibilities. The confidence function is defined as:

$$\mathrm{E}(s_1, s_2) = abs(\cos(\Omega)) \times \frac{(\tau - \Delta)}{\tau}$$

$$\text{where } \Delta < \tau$$

Where s_1, s_2 are the two strokes to be merged, Ω is the angle between the Eigenvectors obtained by using 4 pixels each from the end of the strokes s_1, s_2 to be merged and τ is the global threshold obtained empirically (3-10 pixels), which is dependent upon the pixel-length of the strokes being extracted from the scanned image.

© The Eurographics Association 2008.

After obtaining the most-likely strokes to be merged, the missing pixels between them are obtained by linearly interpolating between the strokes ends having the minimum distance.

Figure 9: *The strokes obtained by the stroke extraction process that is part of the bottom curve of Figure-2.*

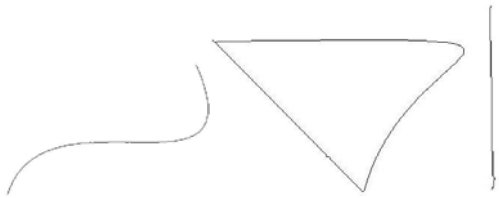

Figure 10: The left figure shows result after merging the three strokes obtained in Figure-9. The middle and left figure show the other strokes obtained.

4. Results

Thus far, the preliminary results of our algorithm appear quite promising. We have tested our algorithm on ten different hand-drawn images, of which one (the example above in Figure-1) contains artificially-added noise, nine contain natural scanning noise (of these seven were drawn on sheet of white piece of paper with a black marker and scanned-in, one was drawn on the computer using paint, then printed out and then scanned in, and in the spirit of the electronic cocktail napkin [GD96] the ninth (Figure-22) was drawn on an actual cocktail napkin and then scanned in). Note that in all of the images below, strokes are resized for space.

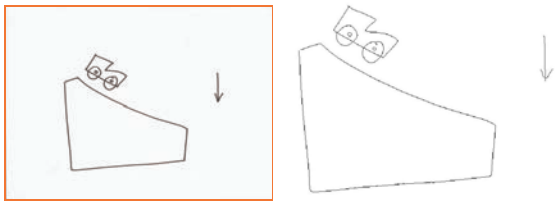

Figure 11: *The left show the original hand-drawn paper sketched image, and the right image shows the image after morphological thinning.*

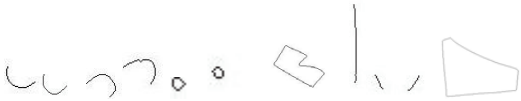

Figure 12: *The results of the images in Figure-11 after stroke extraction but before stroke merging.*

© The Eurographics Association 2008.

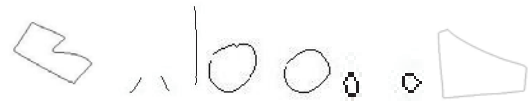

Figure 13: *Extracted strokes after merging.*

Figure 14: *The single computer-drawn scanned image in our study. The left image in the red border shows the original hand-sketched image, the right images show the extracted strokes after merging.*

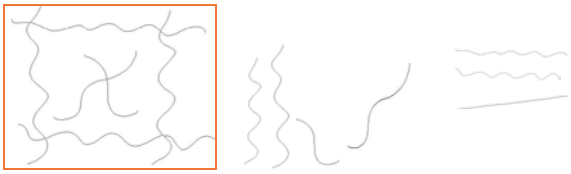

Figure 15: *The left image shows the original paper-sketched image; the right images show the extracted strokes after merging.*

Figure 16: *The left image shows the original paper-sketched image, the right images show the extracted strokes after merging.*

Figure 17: *The left image shows the original paper-sketched image, the right images show the extracted strokes after merging.*

Figure 18: *The left image shows the original paper-sketched image; the right images show the extracted strokes after merging.*

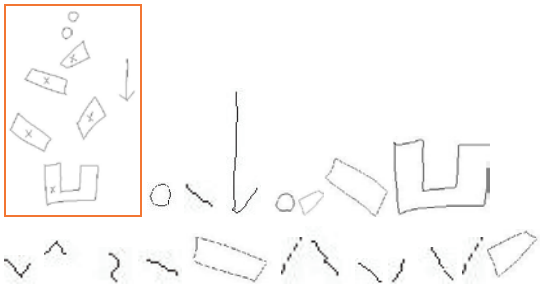

Figure 19: *The top left image in red shows the original paper-sketched image, the right images show the extracted strokes after merging. The gaps that appear in the images above do not actually exist in the stroke but only appear because this image was particularly large.*

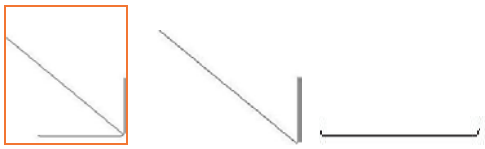

Figure 20: *The top left image in red shows the original paper-sketched image, the right images show the extracted strokes after merging.*

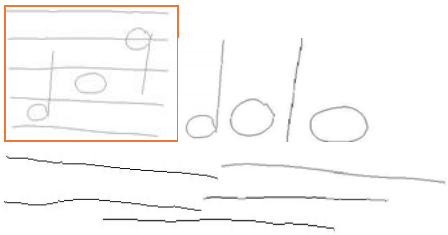

Figure 21: *The top left image in red shows the original paper-sketched image, the right images show the extracted strokes after merging.*

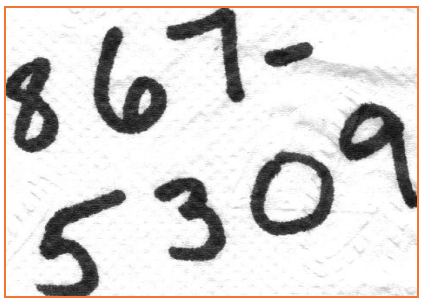

Figure 22: *A scanned in phone number drawn on a paper napkin.*

Figure 23: *The extracted strokes after merging.*

© The Eurographics Association 2008.

© The Eurographics Association 2008.

5. Future Work

Although we have made significant progress toward our goal, several advancements still remain for future work such as those drawn with a variety of pen tips and different colors as well as paper sketches that may have other artifacts and non sketched objects on the page. Additionally, we would like to integrate our method with existing online sketch recognition techniques, in particular those of [PH08, WEH08, HD05]. Figures 24 and 25 show promising initial results. Figure 24 shows a scanned in hand-drawing of a spiral and the start of the generated XML file produced using the techniques described in this paper. This XML file was read into LADDER, producing the image in the left of Figure 25.The shape was then automatically recognized in LADDER using the PaleoSketch low-level recognizer [HD05, PH08] producing the results in the right of Figure 25.

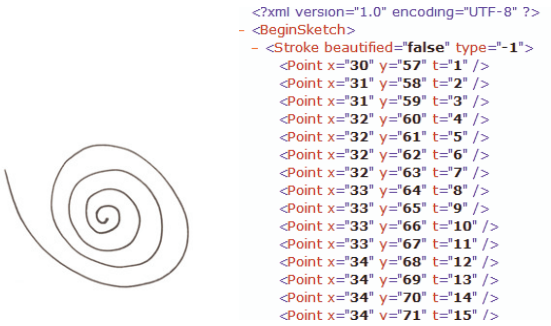

Figure 24: ***LEFT:*** *The hand-drawn scanned-in image of a spiral.* ***RIGHT:*** *The start of the XML filed generated using the extracted stroke points and order based generated time stamps. We use these generated time stamps because speed information cannot be extracted from static images. However, points are still ordered consecutively, such that the continuous stroke is preserved.*

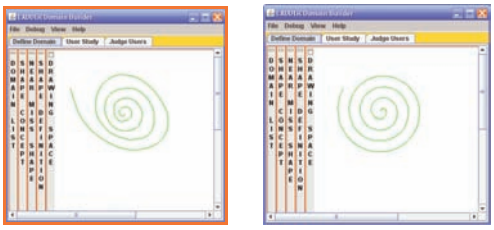

Figure 25: ***LEFT:*** *The XML file from Figure 24 RIGHT loaded into LADDER [HD05] before recognition.* ***RIGHT:*** *The XML file after being recognized in LADDER by Paleo-Sketch [PH08].*

6. Broader Impact

Our system currently requires the sketcher to draw on a white sheet of paper using a black marker. Future work aims at eliminating this constraint, but even with this constraint we argue that a sharpie and a white piece of paper is more natural, convenient, practical and economical than a tablet PC. It also enables the use of sketch recognition techniques in places where a tablet PC may not be practical (such as in a large classroom setting).

7. Contributions

In this paper, we present a method to extract vectorized stroke data from digital pixelized hand-sketched images using a black sharpie on a white piece of paper. Our method has a wide variety of possible applications including recognition of legacy sketches and the automatic correction of classroom-drawn diagrams. Our algorithm utilizes both the local and global characteristics of the stroke being extracted to address *direction ambiguities* and discovers the intended strokes of the user. To accurately assess and follow the correct direction of the stroke, the system utilizes Principal Component Analysis (PCA) to measure the global direction of the stroke and also utilizes a deviation angle-based approach to clarify local ambiguities. In order to evaluate our algorithm, we have tested our technique on a number of example figures, which provide promising results.

8. Acknowledgements

This work was funded in part by the National Science Foundation through IIS grants #0744150: *Developing Perception-based Geometric Primitive-shape and Constraint Recognizers to Empower Instructors to Build Sketch Systems in the Classroom* and #0757557: *Pilot: Let Your Notes Come Alive: The SkRUI Classroom Sketchbook.*

The authors would also like to thank the members of the Sketch Recognition Lab, in particular Brian Eoff, Brandon Paulson, Josh Johnston, Aaron Wolin, Josh Peschel,and Katie Dahmen for their valuable feedback and editing help.

9. References

[AT89] AHUJA N., TUCERYAN M.: Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component Gestalt, *Computer Vision and Graphics Image Processing (CVGIP)*, Vol. 48, (1989), pp. 304-356.

[AD04] ALVARADO C., DAVIS R.: Sketch READ: A multi-domain sketch recognition engine. In *Proceedings of 17th Annual ACM Symposium on User Interface Software and Technology (UIST'04)*. (2004).

[Ano] http://www.anoto.com/?id=906 (2008).

[Bis96] BISHOP C.: *Neural Networks for Pattern Recognition.* Clarendon, Oxford, UK. (1996).

[BCFB07] BARTOLO A., CAMEILLERI K.P., FABRI S.G., BORG J.C.: Scribbles to vectors: Preparation of scribble drawings for CAD interpretations. In *the 4h Annual* EUROGRAPHICS *Workshop on Sketch Based Interfaces and Modeling (SBIM'07),* (2007).

[BCFB06] BARTOLO A., CAMILLERI K.P., FABRI S.G., BORG J.C.: A new sketch based interface using the gray level

© The Eurographics Association 2008.

© The Eurographics Association 2008.

co-occurance matrix for perceptual simplification of paper based scribbles. In *the 3rd Annual ACM EUROGRAPHICS Workshop on Sketch Based Interfaces and Modeling (SBIM'06),* (2006).

[FBC05] FARRUGIA P.J., BORG J., CAMILLERI K.P., SPITERI C.: Experiments with the camera phone-aided design (cpad) system. *In the 15th International Conference on Engineering Design (ICED05),* (2005), pp. 130-131.

[GW02] GONZALES R.C., WOODS R.E.: *Digital Image Processing:* Second Edition, (2002).

[GM93] GUY G., MEDONI G.: Inferring global perceptual contours from the local features. *Image Understanding Workshop,* (1992), pp. 881-892.

[GT99] GUICHARD F., TAREL J.: Curve finder combining perceptual grouping and Kalman-like Fitting. In *Proceedings on the IEEE Conference on Computer Vision,* (1999), pp. 1003-1008.

[GD96] GROSS M., DO E.: The electronic cocktail napkin - computer support for working with diagrams. *Design Studies,* Volume 17, No. 1, (1996), pp. 53-69.

[HS092] HARALICK R.M., SHAPIRO L.G.: *Computer and Robot Vision,* Volume 1, Addison-Wesley, (1992).

[HD05] HAMMOND T., DAVIS R.: LADDER, a sketching language for user interface developers. *Computers & Graphics,* vol. 29, no. 4, (2005), pp. 518–532.

[Ham07] HAMMOND T.: Enabling instructors to develop sketch recognition applications for the classroom. In *Proceedings of the IEEE Conference on Frontiers of Education (FIE'07),* Milwaukee, WI, October 10-13, (2007).

[HEPW08] HAMMOND T., EOFF B., PAULSON, B., WOLIN A., DAHMEN, K., JOHNSTON, J., and RAJAN, P. Free-sketch recognition: Putting the CHI in sketching. In *Proceedings of the 26th Annual Conference on Computer Human Interaction (CHI'08),* (2008).

[JLA04] JUCHMES R., LECLERCQ P., AZAR S.: A multi agents system for the interpretation of architectural sketches. In *Proceedings of the 1st Annual EUROGRAPHICS Workshop on Sketch Based Interfaces and Modeling (SBIM'04),* (2004).

[KK06] KIM D.H., KIM M.J.: A curvature estimation for pen input segmentation in sketch-based modeling. *Computer-Aided Design* 38, 3 (2006), pp. 238-248.

[KS04] KARA L.B., STAHOVICH T.: Hierarchical parsing and recognition of hand-sketched diagrams. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST'04),* ACM Press (2004), pp. 13-22.

[LS96] LIPSON H., SHPITLANI M.: Optimization based reconstruction of 3D object from as ingle freehand line drawing. *Journal of Computer Aided Design* 28, 8 (1996), pp. 651-663.

[LM05] LIPSON H., MSARY M.: A sketch based interface for the iterative design and analysis of the 3D objects. In *Proceedings of the 2nd Annual EUROGRAPHICS Workshop on Sketch Based Interfaces and Modeling (SBIM),* (2005).

[ML06] KAPLAN M., COHEN E.: Producing models from Drawing of the curved surfaces. In *Proceedings of the 3rd Annual ACM EUROGRAPHICS Workshop on Sketch Based Interfaces and Modeling (SBIM'06),* (2006).

[NM04] NOTOWIDIGO M., MILLER R.: Off-line sketch interpretation. In *Proceedings of AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural,* October (2004).

[Pra091] PRATT W.K.: Digital Image Processing, *John Wiley & Sons, Inc.,* (1991).

[PH08] PAULSON B., HAMMOND T.: PaleoSketch: accurate primitive sketch recognition and beautification. In *Proceedings of Intelligent User Interfaces (IUI'08),* (2008).

[SDS98] STAHOVICH T., DAVIS R., SHROBE J.: Generating multiple new designs from a sketch, *Artificial Intelligence,* Vol. 104, (1998), pp. 211-264.

[SSD06] SEZGIN T.M., STAHOVICH T., DAVIS R.: Sketch based interfaces: early processing for sketch understanding. In *Proceedings of the Workshop on Perceptive User Interfaces (PUI'01),* Orlando, FL, (2001).

[SD05] SEZGIN T.M., DAVIS R.: HMM-based efficient Sketch Recognition**.** In *Proceedings of the International Conference on Intelligent User Interfaces (IUI'05).* New York, New York, January 9-12 (2005).

[Sau03] SAUND E.: Finding perceptually closed paths in sketches and drawings. *IEEE Transition on Pattern Analysis and Machine Intelligence (PAMI),* 25,4, April, (2003), pp. 475-491.

[SU88] SHA'ASHUA A., ULLMAN S.: Structural saliency: The detection of globally salient structures using locally connected network. In *Second International Conference on Computer Vision,* December (1988), pp. 321-327.

[WEH08] WOLIN A., EOFF B., HAMMOND T.: ShortStraw: A simple and effective corner finder for polylines. In *Proceedings of the 5th Annual EUROGRAPHICS Workshop on Sketch Based Interfaces and Modeling (SBIM'08),* (2008).

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

© The Eurographics Association 2008.

© The Eurographics Association 2008.