DataDriven Interactive Drawings

Abstract

DrawAFriend abstract goes here...

Keywords: Interactive Drawings, Crowdsourcing,

1 Introduction

Drawing as a means of communication dates well before other forms of recorded history. Even as photography has come to dominate visual communication, drawing still holds an important role in artistic expression. Unfortunately, two factors make drawing difficult. Although drawing skills are very learnable, few people take the time to master this medium. In addition, although electronic devices can help this by providing, for example, tracing methods, the *fat finger* phenomenon makes drawing on phones and tablets very error prone. We address these problems in two ways. First we construct a game, *DrawAFriend*, that encourages people to draw (in our initial case celebrity portraits), and second we use the corpus of drawings from previous players to aid new drawers overcome the fat finger problem by attracting their strokes to the consensus of previous drawings while leaving their individual styles intact. This application represents only a first use of a new dataset of hand drawings collected by the game.

The big data revolution is profoundly transforming computer science and society. Problems which frustrated generations of researchers, such as machine translation and object recognition, suddenly are on the brink of becoming "solved" when run on large datasets. We now see a schism between domains for which large datasets are available such as translation corpuses and those for which it is not, with much more rapid progress in the former.

One domain suffering from such data scarcity is hand-drawn images. An ideal drawing corpus would have precise stroke-level data, including timing and order. Such stroke information could allow us to rigorously answer questions about the order in which artists draw lines, and how quickly. We might even glean semantic knowledge about the subject by understanding the temporal relationships among strokes. For statistical purposes, we would like a *large* dataset, with many drawing by the same artist and many drawings of the same subject by different artists.

To address these issues, we recently developed an iPhone game called *DrawAFriend*, which is intended to generate a corpus of hand-drawn images containing all of the information described above. We currently focus on face portraits. Such drawings are exceedingly difficult to draw by hand, and even more so using a touch interface. To aid users and to collect multiple drawings of the same subject, we allow players to trace over existing photographs.

Our use of a game to generate data brings the added challenge that the game must be rewarding and encourage repeated play. We address this challenge by allowing player to draw mutual friends on the social network as well as celebrities. We phrase the challenge as a guessing game, with the important corollary that we can learn *when* a particular face was recognized, i.e. which strokes were necessary for recognition. The success DrawAFriend is beginning to appear: within DrawAFriend's first 3 days on the market, we generated over 5,000 images, all with stroke-level information.

In addition to describing DrawAFriend, we also demonstrate a first application of the initial drawing data to provide a self-correcting touch-based drawing interface on mobile devices. We observe that drawing with a touch device often suffers from the "fat finger" problem. We factor this issue into two elements: (1) the "intent" of the artist in drawing a stroke, and (2) an additional random noise component caused by inaccuracy in the touch interface. We therefore hypothesize that if strokes are "average" (in an appropriate sense) over a sufficiently large database of drawings, then we can cancel out the noise and recover the original intent. Furthermore, this data allows us to develop a "self-correcting" touch interface: in essence, we clean up the artists drawing in real time by using data from hundreds of previous drawings of the same subject. To solve this problem, we present a new metric for "local stroke coherence," that is, regions of the images where many artists agree. We further present a surprisingly simple method to correct strokes based on this metric making it significantly easier to draw faces. The interface appears "invisible" as the result feels more like the intent of the user than the originally stroke.

AT:Note we got over 5,000 images in the first three days of the game.

2 Related Work

The graphics community is witnessing a recent spike of interest in "big data" approaches to understanding drawing, with the prototypical examples being *ShadowDraw* [Yong Jae Lee 2003] which help guide freeform sketching and *WhatsMySketch* [Eitz et al. 2012] which identifies iPhone sketches as one of 250 possible object classes. A important distinction between these efforts and our work is the data-collection approach. Lee et al. used 30,000 sketches downloaded from existing web databases in raster format which Eitz et al. collected 20,000 sketches on Amazon Mechanical Turk (https://www.mturk.com). By contrast created a publicly available iPhone game *DrawAFriend* which uses mechanics similar to *DrawSomething* (http://omgpop.com/drawsomething) to intrinsically motivate players to contribute drawings. Therefore, rather than sequestering data collection into an initial phase of our research, we collect data continuously with zero marginal cost per user, and can re-instrument the game to change data collection on the fly. Our game approach also gives us more information per user, including knowledge of their social graph, but requires us to design a compelling game to collect data. In this sense, *DrawAFriend* is closer to the games of von Ahn and colleagues [2004; 2006] which uses games to label objects in images. By contract, we ask players to complete the much more complex (and creative) task of actually drawing new images. Another important distinction is that *ShadowDraw* and *WhatsMySketch* study freeform sketching, while our celebrity database consists of many registered drawings of the same image.

There has also been considerable work in the Human Computer In-

teraction community on solving the problem of inaccurate touch interactions, often called the *fat finger problem*. Wigdor et al. [2009] taxonomized touch-based interaction errors, and presented novel visual cues to help the user understand their intent. Other work has addressed the fat finger problem by designing new interaction patterns [Albinsson and Zhai 2003; Benko et al. 2006; Forlines et al. 2006; Vogel and Baudisch 2007], or adding additional interaction hardware [Scott et al. 2010; Wigdor et al. 2006; Wigdor et al. 2007]. The book by Benko and Wigdor [2010] presents a good overview of work in the field.

By contrast, we add no additional hardware, visual cues or interaction paradigms. Instead we use data from lots of previous drawings of the same image to seamlessly correct user strokes as they draw. Our stroke correction approach draws inspiration from the work of Cole et al. [2008], which studied collected statistics on a small number of hand-collected, registered sketches of the same object. Our stroke correction method was inspired by the finding of Cole et al. (confirmed by our data) that artists frequently draw similar lines. We further develop this this idea with our hypothesis that the average of these strokes represents the fundamental user "intent" towards which we snap strokes. Our approach minimizes an energy function on the stroke similar to the intelligent scissors method [Mortensen and Barrett 1995]. However, rather than snap to image contours AT:Note: it would be interesting to have this as a comparison., our energy function is based on average strokes from many registered sketches. We further preserve stroke geometry using a Laplacian method which is often used in geometry processing [Sorkine et al. 2004]. Another system called *iCanDraw* [Dixon et al. 2010] helps users draw faces, although using a tutorial approach rather than looking at automatic correction based on a large dataset.

3 DrawAFriend: The Game

DrawAFriend is a Facebook integrated game, which incentivizes players to take two actions: draw pictures of their friends and indirectly evaluate their friends drawings. Through the Facebook API, players can draw their friends profile pictures. This converts one large dataset of quality photos, Facebook profile pictures, into a large dataset of user created drawings. This collection includes much more than drawings, but drawing strokes as well as an understanding of the drawing content. Furthermore this dataset will include drawings from artists around the world with different artistic and cultural backgrounds.

AT:We should indicate that we advertised to attract users.

DrawAFriend is an asynchronous turn based guessing game. It works as follows. Players have an option to start a gmae with either a facebook friend or a random player on the internet. The player is then given four pictures which he can draw. These will either be a mutual facebook friends profile pictures or celebrity photos.

After choosing one of their mutual friends to draw, the player is brought to the drawing screen. There he/she can trace the image. At any point the user can press the eye button to hide the profile picture and just see their drawing. Since the iPhone screen is quite small, and the touch screen is not that accurate, players can zoom using the pinch zoom gesture. Once finished, the player can send his/her drawing to the friend whom he is playing the game with. The friend will receive a notification that they have a drawing to guess. Similar to hangman, the player can guess which letters are in the mutual friends name. In order to guess a vowel, players must spend coins. Once all consonants have been guessed players can then buy vowels for free.

This final process gives us a weak classifier of how good the images are. In general, a good drawing is much more likely to be guessed correctly than a bad drawing.

4 Game Decisions

While DrawAFriend is considerably easier on a larger screen (particularly the iPad vs iPhone). Considerably more people have an iPhone than an iPad.

4.1 The Dataset

The DrawAFriend dataset is made up of two categories:

1. Drawings of facebook profile pictures
2. Drawings of celebrity photos

These two sub-datasets have very different structure. The facebook profile picture was a shallow dataset, where we had many pictures (800 pictures (made up)) but a very low drawing per pixel ratio 1.2 drawings (made up) per pixture.

On the other hand the celebrity sub-dataset is a depth dataset. At the time of writing this paper, there were only 6 celebrity pictures in the game. However since anyone could draw them, there were on average 100 drawings per celebrity photo.

4.2 Cleaning the Dataset

All the drawings are not created equal. While we did not want to pass judgment on different styles, certain drawings are made by people simply trying out the game or actively trying to cheat.

We found a statistically relevant way to weed out drawings that were drawn hapazaardly or just involved spelling out the name of the model.

Potentially include those histograms that show that all the good drawings have significantly more short lines, and have ever decreasing amount of long lines (would have to re-run these experiments).

4.3 Average Drawings

Should average all the drawings into a mean drawing. 1) One of all the good drawings 2) One of all the drawings (of one celebrity)

5 Datadriven Drawing Helpers

5.1 Averaging

There are a lot of just ok drawings. Drawings where people did not even both to zoom in on the eyes to try to get anything done. Basically subcumming to the fat finger problem.

However since there is so many of those drawings, I wonder if we could average those drawings in order to get a good drawings.

Certainly doing a rasterized averaging of all the drawings would, at the very least give a really cool result. However I think the final result would actually be a great drawing.

5.2 Auto-magically Correcting Lines

Our line correction strategy has two phases.

- Using all of the training drawings available for our image, we create a vector field, which tells us for each pixel on the

image, the delta toward the nearest apparent line (section #). This phase can occur offline.

- The vector field is transmitted to each mobile device. When a user draws a stroke, the field is sampled along the path of the stroke, and the stroke is moved smoothly (section #).

This two-phase approach is advantageous, because the bulk of learning can be done a-priori, off of the underpowered mobile device.

5.2.1 Computing the Vector Field

The goal of the vector field is to move our point towards the nearest *apparent stroke*, which defined simply as a stroke that appears in many drawings.

Given a point p in the image, we first find the nearest point from every drawing in our training set. Call this set of nearest-neighbors N. Members of N fall into one of two categories: those from an apparent line, and noise. We assume that there can only be one apparent stroke represented in N. As shown in figure #, an apparent stroke will manifest in N as a line through the origin, orthogonal to the strokes.

NF:Old unincorporated stuff below here.

Have a vector field that calculate the nearest neighbor location of the the current stroke. Use least squares to preserve curvature but still map to current stroke. We have a weighting for how to balance location and curvature. Wonder if we should expose this to the user in some way?

Using any medium, but particularly the iPhone, humans are far less percise than they are with a pen and paper. iPhone is particularly plagued with the fat finger problem. The exact location of a stroke is rarely exactly where one want it to be. ¡maybe site the fat finger papers that michael sent out a while back¿

One thing that people do want preserved is curvature. At least the impression of curvature. Location is difficult, while curvature is quite a bit easier. This may differ with mouse drawings, quick curvature isn't always easy to do but exact location is possible with the mouse (however you don't really have anything to back that up, mostly conjecture).

-Stylistic Agrrement -Curvature is similar, and should not be corrected -Location is off and should be corrected. -Stylistic Disagreement -Curvature is not similar, and should not be corrected -Location should not be corrected. However maybe it should be, it would be difficult to know where though. However the assumption is that when there isn't an agreement it's most likely location doesn't matter that much.

5.3 Calculating Agreement

When analyzing the dataset, it became clear that there are moments of stylistic interpreation. While other seems purely conveyed information. It was those stylistic interpretations where people seemed to disagree the most in terms of where lines would go, while other lines were consistant across all good drawings. Hair often fell under the style category, where each artist had their own different type of hair. While the contour sillhouette, as the Where Do People Draw Lines paper put it, was almost always drawn by everyone.

The difference may not necessarily be one based on style. But none the less based on the intent of the stroke. The most common stroke (though you are totally guessing) are ones meant to represent contours. It is the most novice way to draw. Perhaps the most telling is players who draw the outline of a celebrities hair. This is a simple but not the best way to reveal hair.

Conveying shape without contour, usually done through shading, tends to lend itself to more stylistic iterpretation and less percise brush strokes.

It was important to figure out when stylistic interpretation was occurring, in order to create our automagic line correction.

To calculate which strokes should be corrected and warped and which should be left alone, an interesting discovery was made. At any point of a picture, we grabbed a single nearest point from each one of the drawings. When we plot those points we found that when lines were "in aggreement", those nearest points formed a striaght line (usually perpendicular to the direction of all the strokes that they were pulled from). When people draw a line the error tends to be in the perpendicular to the direction that they are drawings.

Using this fact we were able to calculate to what extent lines in a certain area were in agreement. By using the following formula: ¡formula¿ We were able to...

The linear correlation of nearest neighbors when strokes in an area are in agreement.

5.4 Agreement compared to location

I wonder if there's any correlation to be found between "agreement" and location in the face.

Or that people tend to be more in aggrememtn when we are dealing with things with eyes, where you need to zoom in quite a bit. While a chin, people don't need to be as exact to have about the correct drawing.

I guess this is more talkign about how tight the nearest neighbors are vs how linear they are. Does the scoring function deal with that?

5.5 Auto-Extending Lines

We don't have anything doing this.

5.6 Preserving Style

Adaboosting but for strokes. We haven't really done this yet.

6 Results

6.1 Before and After

I suppose this is where we show some before and afters of people's drawings before hand, and people's drawings afterwards.

This could either be User Studies, or we could take existing drawings and run them through the algorithm, playing them back.This would probably work. However users don't get the feedback of their strokes changing, which would be really useful.

6.2 Unscientific User Studies

This could be one and the same with the Before and After. But there may be a benefit from walking around and giving people the auto correcting app to play with.

7 Conclusion

One idea for bigger picture: Is that we have developed a format of dealing with the "fat finger" problem. A data-driven method to deal with that issue. (Though this might be more appropriate for an HCI project)

Looking further into the future, we hope that DrawAFriend will establish a template for Facebook applications that incentivizes the creation of large complex datasets. This will create a virtuous cycle between user interfaces, social experiences, and research to promote our understanding of our humanity in this ever evolving interconnected world.

References

ALBINSSON, P.-A., AND ZHAI, S. 2003. High precision touch screen interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '03, 105–112.

BENKO, H., WILSON, A. D., AND BAUDISCH, P. 2006. Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '06, 1263–1272.

COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2008. Where do people draw lines? *ACM Transactions on Graphics 27*, 3 (Aug.), 88:1–88:11.

DIXON, D., PRASAD, M., AND HAMMOND, T. 2010. icandraw: using sketch recognition and corrective feedback to assist a user in drawing human faces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '10, 897–906.

EITZ, M., HAYS, J., AND ALEXA, M. 2012. How do humans sketch objects? *ACM Trans. Graph. 31*, 4 (July), 44:1–44:10.

FORLINES, C., FORLINES, C., VOGEL, D., VOGEL, D., BALAKRISHNAN, R., AND BALAKRISHNAN, R. 2006. Hybrid-pointing: Fluid switching between absolute and relative pointing with a direct input device. In *In UIST*, ACM Press, 211.

MORTENSEN, E. N., AND BARRETT, W. A. 1995. Intelligent scissors for image composition. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 191–198.

SCOTT, J., IZADI, S., REZAI, L. S., RUSZKOWSKI, D., BI, X., AND BALAKRISHNAN, R. 2010. Reartype: text entry using keys on the back of a device. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, ACM, New York, NY, USA, MobileHCI '10, 171–180.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Eurographics / ACM SIGGRAPH Symposium on Geometry Processing*, 175–184.

VOGEL, D., AND BAUDISCH, P. 2007. Shift: a technique for operating pen-based interfaces using touch. In *PROC. CHI '07*, ACM Press, 657–666.

VON AHN, L., AND DABBISH, L. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '04, 319–326.

VON AHN, L., LIU, R., AND BLUM, M. 2006. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '06, 55–64.

WIGDOR, D., LEIGH, D., FORLINES, C., SHIPMAN, S., BARNWELL, J., BALAKRISHNAN, R., AND SHEN, C. 2006. Under the table interaction. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '06, 259–268.

WIGDOR, D., FORLINES, C., BAUDISCH, P., BARNWELL, J., AND SHEN, C. 2007. Lucid touch: a see-through mobile device. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '07, 269–278.

WIGDOR, D., WILLIAMS, S., CRONIN, M., LEVY, R., WHITE, K., MAZEEV, M., AND BENKO, H. 2009. Ripples: utilizing per-contact visualizations to improve user interaction with touch displays. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '09, 3–12.

WITH DANIEL WIGDOR, H. B. 2010. *Imprecision, Inaccuracy, and Frustration: The Tale of Touch Input*. in *Tabletops - Horizontal Interactive Displays*. Springer HCI Series. Springer-Verlag London Ltd.

YONG JAE LEE, LARRY ZITNICK, M. C. 2003. Shadowdraw: Real-time user guidance for freehand drawing. *ACM Trans. Graph. 22*, 3 (July), 879–887.