Real-time Drawing Assistance through Crowdsourcing

Alex Limpaecher†, Nicolas Feltman†, Adrien Treuille†, and Michael Cohen*

†Carnegie Mellon University *Microsoft Research

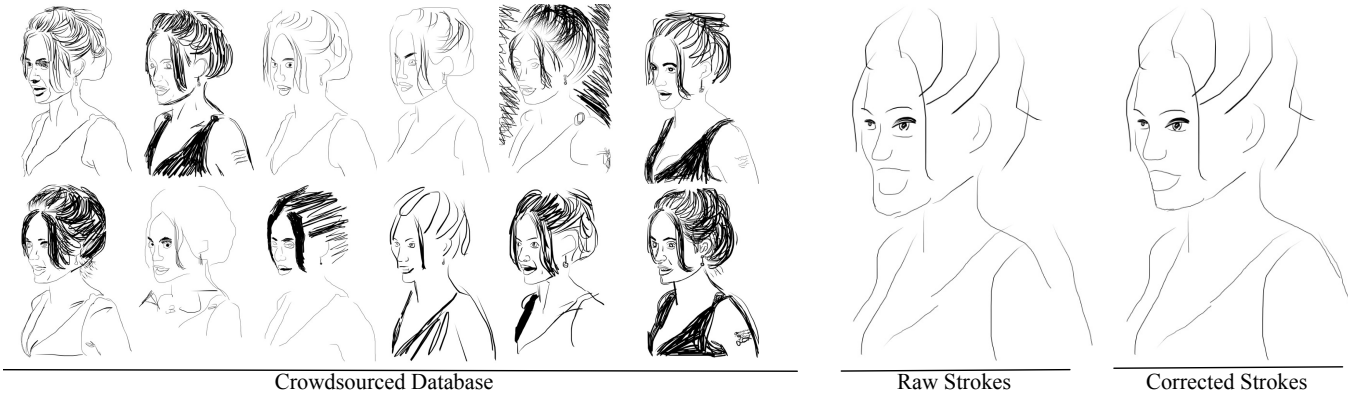


Figure 1: *Left: twelve examples of drawings of Angelina Jolie, created by DrawAFriend players. Right: our algorithm automatically corrects new drawings in real-time based on a consensus of the drawings by previous players.*

Abstract

We propose a new method for the large-scale collection and analysis of drawings by using a mobile game specifically designed to collect such data. Analyzing this crowdsourced drawing database, we build a spatially varying model of artistic consensus at the stroke level. We then present a surprisingly simple stroke-correction method which uses our artistic consensus model to improve strokes in real-time. Importantly, our auto-corrections run interactively and appear nearly invisible to the user while seamlessly preserving artistic intent. Closing the loop, the game itself serves as a platform for large-scale evaluation of the effectiveness of our stroke correction algorithm.

Keywords: Interactive Drawings, Crowdsourcing,

CR Categories: I.3.3 [Computer Graphics]: Line and curve generation—; J.5 [Arts and Humanities]: Fine arts—.

Links: DL PDF VIDEO WEB

1 Introduction

Drawing as a means of communication dates to well before other forms of recorded history. Today, drawing remains a vital form of artistic expression and an important window into human perception. However, the central challenge to further scientific analysis of drawing is data scarcity. Although search engines index a huge collection of line drawings, these images are stored in raster format with little or no useful metadata. Ideally, a drawing corpus would contain precise stroke-level data for each image, including timing information. We would also like semantic metadata identifying artists and subjects. Even more ambitiously, we would like to glean *perceptual* information, such as which strokes contributed most to image recognition. Finally, for statistical purposes, we would like a *large* dataset, with many drawings by the same artist and many drawings of the same subject by different artists.

To address this challenge, we developed *DrawAFriend*, an iPhone game specifically designed to collect drawing data, including all of the information described above. We currently focus on face portraits. Faces are exceedingly difficult to draw by hand, and even more so using a touch interface on a small mobile device. To aid users and to collect multiple drawings of the same subject, we allow players to trace over existing photographs. In its first week of release *DrawAFriend* generated over 1,500 images per day.

We believe that this large and continuously growing drawing database will enable a rich stream of future research in graphics. As a first application, we demonstrate how the *DrawAFriend* corpus can be mined 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 conceptually 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 we can determine a consensus of strokes (in an appropriate sense) over a sufficiently large database of drawings, then we can cancel out the noise and recover the artist's original intent. We analyze the drawing corpus to compute a *correction vector field* that for any location, points towards a nearby consensus of strokes. This allows us to develop a real-time *self-correcting* touch interface: as players draw, we essentially clean up their drawings by using data from previous drawings of the same subject. We further introduce a surprisingly simple method to correct strokes based on this consensus while maintaining the *stylistic* choices of the artist. The interface requires no new user interaction paradigms; in other

words, it appears "invisible" to the user. The resulting strokes feel more like the intent of the user than the raw original strokes.

To validate the effectiveness of our auto-correction algorithm, we ran a large-scale user study within the game. Each time a user draws a celebrity, we randomly turn the stroke correction on or off. Our results validate the effectiveness of our stroke correction algorithm: with autocorrect on, artists do not need to draw as accurately, and they undo their strokes less. The ability of *DrawAFriend* to serve simultaneously as a large-scale visual data collection platform and as a statistically relevant user study underscores the generality of our crowdsourcing approach.

2 Related Work

Perhaps the best known games designed to collect data are those by von Ahn and colleagues [2004; 2006] which uses games to label objects in images. By contrast, we ask players to complete the much more complex (and creative) task of actually drawing new images. The graphics community is witnessing a recent spike of interest in "big data" approaches to understand drawing, with the prototypical examples being *ShadowDraw* [Lee et al. 2011] which helps guide freeform sketching and *WhatsMySketch* [Eitz et al. 2012] which identifies iPhone sketches as one of 250 possible object classes. An important distinction between these efforts and our work is the data-collection approach. Lee et al. uses 30,000 images downloaded from existing web databases and extracted edges as proxies for possible sketches. Eitz et al. collected 20,000 sketches on Amazon Mechanical Turk (https://www.mturk.com). By contrast, we 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. Another 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 Interaction community on solving the problem of inaccurate touch interactions, often called the *fat finger problem*. Wigdor et al. [2009] taxonomizes touch-based interaction errors, and presents novel visual cues to help the user understand their intent. Other work addresses 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.

There has been little work in helping users draw. One system, *iCanDraw* [Dixon et al. 2010] helps users draw faces with a tutorial approach. A sketching beautification system [Orbay and Kara 2011] infers smooth lines from a sketch consisting of short, overlapping strokes. *Elasticurves* [Thiel et al. 2011] neatens sketches by smoothing strokes dynamically based on stroke speed. Similar to our method, *ShadowDraw* [Lee et al. 2011] uses collected images to aid users in drawing by providing *shadows* of similar drawings. In contrast to these methods and to the *fat finger problem* corrections, we add no hardware, visual cues or interaction paradigms. The user thinks they are simply drawing with no help. We use data from a corpus of previous drawings of the same image to seamlessly correct user strokes as they draw. Gingold's work [2012] uses image averaging to beautify images. However their method, which involves averaging euclidean points, is only used to improve simple drawings such as smiley faces. Our stroke correction method was inspired by the finding of Cole et al. [2008] that artists frequently draw similar lines. Our results confirm Cole's finding, and further develop this idea with our hypothesis that the consensus of these strokes represents the fundamental user "intent." Our method detects and leverages this artistic consensus in order to interactively adjust strokes.

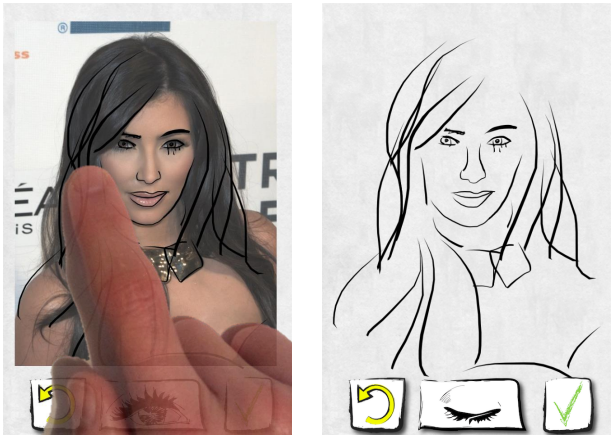


Figure 2: *DrawAFriend: tracing a photo (left), the drawing alone (right).*

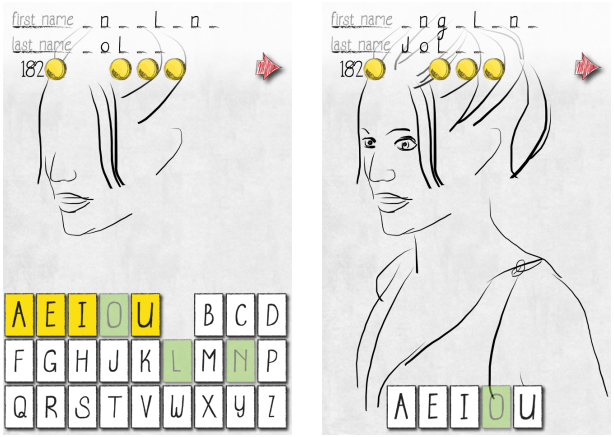


Figure 3: *DrawAFriend: guessing identity (left). Once a player guesses all consonants in a name, the consonant keys animate away and vowels no longer cost coins. (right).*

The concurrent work of [Zitnick 2013] improves handwriting and simple drawing primitives by moving strokes towards an average of similar strokes. The high level notion of leveraging the consensus is similar to our work; however they use a very different definition of a stroke to enable fast matching since they do not have the luxury of having all strokes being aligned a priori.

Our stroke-correction technique minimizes an energy function similar in spirit to the intelligent scissors method [Mortensen and Barrett 1995]. However, rather than snap to image contours, our energy function is based on a consensus of strokes from many registered drawings. We further preserve the *style* of the users' stroke geometry using an approximately gradient-preserving deformation method often used in geometry processing [Botsch and Sorkine 2008].

3 DrawAFriend: The Game

In order to capture and analyze a large-scale drawing dataset, we have developed *DrawAFriend*, a Facebook-integrated turn-based drawing and guessing game for mobile devices. The game is designed to intrinsically motivate players to contribute drawing through a hangman-like guessing mechanism. This approach enables us to gather a large number of drawings with zero marginal cost per drawing, and to modify and instrument the game to capture specific types of data.

The game works as follows. Players have an option to start a game with either a Facebook friend or an anonymous stranger. The player is then given four pictures which she can draw. These will either be mutual friends' profile pictures or celebrity photos. When playing with a stranger, the game offers only celebrity photos.

After choosing a photo to draw, the player is brought to the drawing screen. There she can trace the image (see Figure 2 left). At any point, the user can press the *eye* button to hide the photo and see their drawing on its own (Figure 2 right). To overcome the limitations of the phone's size and touch screen inaccuracies, players can pan and zoom using the pinch zoom and two fingered pan gestures.

Once finished, the player sends her drawing to the friend or anonymous player with whom she is playing. The friend receives a notification that they have a drawing to guess. The user is prompted to guess the identity of the other player's drawing (Figure 3 left). The drawing is replayed stroke by stroke, and similar to *Hangman*, the player can guess which letters are in the mutual friend or celebrity's name. Vowels originally cost coins, however once all the consonants are guessed, vowels become free (Figure 3 right).

The tracing paradigm results in a set of pre-aligned drawings. Whereas other papers begin with rasterized versions of drawings, we collect individual strokes represented as polylines along with timing information. Furthermore, by observing the guesses we can indirectly evaluate the quality of the drawings. We hypothesize that a good drawing is much more likely to be guessed correctly than a bad drawing. *DrawAFriend* thus leverages a dataset of quality photos (Faceboook profile pictures and celebrity images) and via the efforts of players, results in a large dataset of user created drawings. This dataset includes drawings from artists around the world with different artistic and cultural backgrounds.

For the purposes of tackling the fat finger problem, we focus for the remainder of the paper on the corpus of celebrity drawings. These represent sets of drawings of the same photographs by many different artists.

4 Data Driven Drawing

Our goal is to provide a drawing interface on mobile devices that provides the feeling that the user is in full control, while simultaneously providing assistance, in particular, to overcome the inherent fat finger problem. As described earlier, the problem is more constrained than a general drawing system through the use of a tracing paradigm. The simplest idea would be to attract drawn strokes to edges found in the image being traced. Unfortunately, this idea fails in two respects. First edge detectors such as Canny methods (see Figure 9, e) have no notion of semantics and thus appear noisy and inconsistent. Second, unlike automated edge detectors, humans tend to select only the most meaningful edges to draw.

To provide assistance, we take advantage of previous drawings of the same face and then pull the users strokes towards a *consensus* of strokes from previous drawings. The simplest idea for forming a consensus would be to use an *average* drawing (see Figure 9,b).

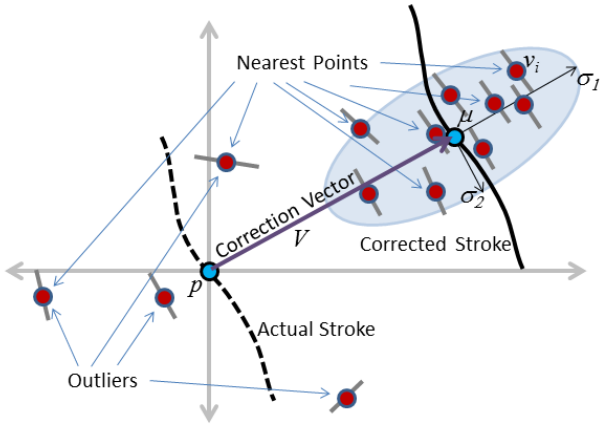


Figure 4: *A didactic blow-up of the graphs in Figure 6. The point at the origin represents a point, p on a stroke being drawn. The red points are the set of nearest stroke neighbors from each database drawing. Note that the orientation of the strokes are always orthogonal to a vector from the origin. We determine a correction vector based on a learned anisotropic Gaussian with mean μ and sigmas σ_1 and σ_2 surrounding a mode of nearby points from strokes in the database drawings.*

This idea is also very problematic as the averages tend to create very fat lines of varying darkness, as well as broad regions such as in the hair. Although some kind of *skeletonization* may aid the first problem, it would fail on the second.

Instead, we develop a stroke correction strategy with two phases.

(i) Consensus Finding: Using the training drawings available for an image, we create a correction vector field which indicates, for each pixel on the image, the delta toward the nearest *consensus stroke*. This phase is run off-line.

(ii) Interactive Correction: The correction vector field is transmitted to the mobile device along with the image to be traced. When a user draws a stroke, the field is sampled, and the stroke is moved in real-time in a way that maintains the original style.

4.1 The Correction Vector Field

A correction vector field, $V(p)$, (see Figure 9(c)and (d)) is constructed to point, for each location p in the image, towards the nearest *consensus*. Intuitively , the consensus represents a location where many almost parallel strokes pass nearby in the training drawings. Interestingly, we do not need to construct an explicit model of consensus strokes. Instead, we can directly compute V using a modified version of the *mean shift* algorithm [Comaniciu and Meer 1999].

Given a point p on a stroke in the image being drawn, we first find the nearest stroke point, v_i in each drawing, D_i, in our training set. We place the nearest points, v, in a coordinate system using p as the origin (see the red points in Figure 4 and the points in Figure 5). One key observation is that the position of the point v_i also implicitly encodes the orientation of the nearest stroke in D_i. In particular, the stroke orientation will be orthogonal to the vector v_i (unless the nearest point is a stroke endpoint). Otherwise there would be some closer point along the stroke. Thus, if the nearest strokes from different users are closely aligned in position

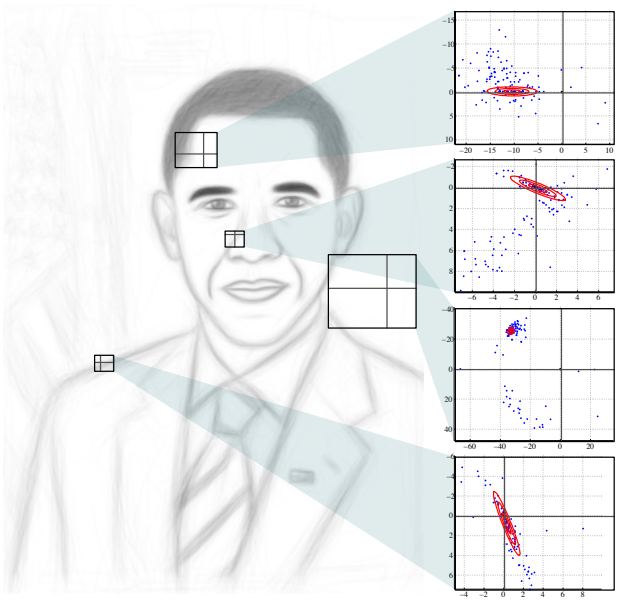


Figure 5: *Nearest neighbors and learned Gaussians for four points in the Obama dataset. The Gaussian distribution is shown with ellipses at the first, second, and third standard deviations.*

and orientation, they will produce a set of the nearest points, v, that lie approximately along a line. Furthermore, this line will intersect the origin (i.e., the point p). We use this observation to build the consensus of nearby strokes.

Our goal is find a consensus of nearby strokes that avoids undue influence from outliers. For the points, v, we seek a *mode* in which the points lie in close proximity to each other and lie roughly along a line pointing back towards the origin. We employ an iterative mean shift style algorithm for the mode finding.

Our algorithm works by iteratively updating a vector of weights, $\mathbf{w}$, which represent the belief that a particular nearest point, v_i, is a member of the consensus stroke. The final correction vector, $V(p)$, is determined by the weighted mean of the points, v. We initialize all weights using a symmetric Gaussian centered at the origin, with σ equal to the mean distance to all nearest points, v.

The iterations proceed by determining an anisotropic Gaussian with weighted mean, $\mu = \left(\sum_j w_j \mathbf{v}_j\right) / \sum_j w_j$, with σ_1 in the direction back toward the origin (Eq. 1), and (σ_2) in the orthogonal direction (Eq. 2). We define the normalized mean vector as, $\nu = \mu / ||\mu||$, and finally determine the anisotropic standard deviations of the projected distances to the mean, given by

$$\sigma_1 = \sqrt{\left(\sum_j w_j ((\mathbf{v}_j - \mu)^{\mathbf{T}} \nu)^2\right) / \sum_j w_j} \quad (1)$$

$$\sigma_2 = \sqrt{\left(\sum_j w_j (\mathbf{v}_j^{\mathbf{T}} \nu_\perp)^2\right) / \sum_j w_j} \quad (2)$$

We add a simple regularization $w_i \leftarrow w_i/(w_i + 0.05)$ to the weights. The regularization serves to reinforce points near the mean, and further discount points away from the mean. The term 0.05, which intuitively says that any points more than two standard deviations away from the mean are mostly noise, was chosen experimentally. We then reweight all points according to the new Gaussian distribution, and iterate until the mean stops moving.

The system typically converges quickly (all of our experiments use a hard-coded 10 iterations), and $V(p)$ is set to μ. Examples of nearest neighbor sets and their resulting consensus strokes are shown in Figure 5.

The vector field, V, defined above indicates a best guess of how any individual vertex, p, on a stroke polyline should move to match the consensus of all drawings. V maps the discretized vector field onto the points p using bilinear interpolation.

The naive approach is to take the points $(p_1, \ldots, p_k)$ comprising a stroke, along with their correction field samples $(V(p_1), \ldots, V(p_k))$, and directly move each point to arrive at $(p_1 + V(p_1), \ldots, p_k + V(p_k))$. Unfortunately, noise and discontinuities in the vector field cause undesirable end results. More importantly, any *stylistic* choices such as intentional wiggles inherent in the original stroke would also be lost. We hypothesize that the fat finger problem is likely to cause the input stroke to be off by a roughly constant displacement, while the original stroke shape is likely to represent the user's *intent*.

Our goal is thus to use the guidance of the vector field on where to move, but still maintain most of the shape of the original stroke. We therefore create and solve an over-constrained linear system that represents both of those objectives.

With p_i representing input stroke sample locations for $i = 1 \ldots k$, V_i a shorthand for $V(p_i)$ representing the correction vector field at p_i, and p'_i representing the corrected samples, we construct a quadratic error function as:

$$E = \sum_{i=1}^{k} (p'_i - (p_i + V_i))^2 + \alpha \sum_{i=2}^{k} ((p'_i - p'_{i-1}) - (p_i - p_{i-1}))^2 \quad (3)$$

The first term represents the faithfulness to the correction vector field. The second term tries to maintain the shape of the stroke by enforcing that neighboring points move in sync. The α term weights the relative importance of the two terms. The error is minimized with respect to p' using a standard least squares solver.

4.1.1 Setting α - Closely Spaced Consensus Strokes

A higher α value results in *stiffer* strokes in the sense that they maintain their shape instead of following variations in the vector field precisely. Aside from maintaining the *style* of the strokes, a high stiffness also helps avoid having part of a stroke pulled in one direction and another part pulled in another. This can happen especially when there are two closely spaced parallel lines in the consensus, such as the two sides of the nose, or the bottom and top of the lips. Unfortunately, such a very high α does not allow the strokes to adapt at all to the subtleties of the consensus drawing represented in the vector field.

To achieve the dual goals of avoiding the two nearby stroke problem while balancing stiffness and style, we use a *continuation method*, performing three iterations of the solver described above. We begin with $\alpha = 10$. This high stiffness value leads to an almost rigid transformation of the stroke to the most dominant consensus region. We then lower α to 7, and feed the previous result, $\hat{p}$, into the first error term, $(p'_i - (\hat{p}_i + V_i))$, while keeping the original terms for the latter half of the error measure, and solve the new system. We repeat this one final time with $\alpha = 4$ to get our final result. The α values (10, 7, and 4) were chosen experimentally.

	Avg. # Strokes	Avg. Stroke Length	Avg. Draw Time (Secs)	% Guessed Correctly
Angelina Jolie	155	84	421	54%
Robert Downey Junior	300	45	572	43%
Kim Kardashian	161	110	506	61%
Kristen Stewart	162	66	434	60%
Brad Pitt	102	50	281	55%
Barack Obama	121	65	607	81%

Figure 6: *Celebrity drawing statistics for 611 hand-picked drawings from the first four days after launch.*

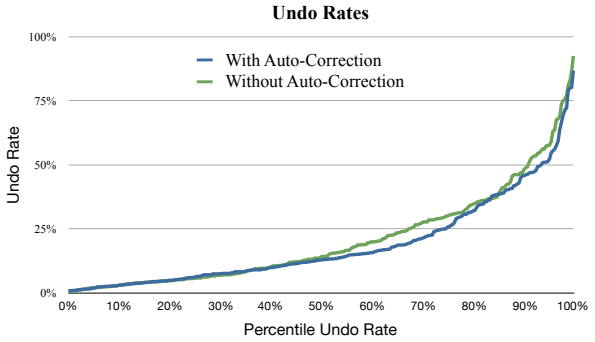


Figure 7: *This chart plots the undo rates (percentage of all strokes that are undone, i.e., undoes / undoes + not-undoes) for each drawing sorted by undo rates. The X axis indicates the percentile undo rate, for example at the 50% mark half of the drawings had less undoes and half more. As can be seen the two sets behave similarly for the first half in which the undo rate is below about 15%. However for higher percentiles the two sets diverge, with the auto-corrected drawings generally showing lower undo rates.*

4.1.2 Purely Stylistic Strokes

We also found that some strokes are purely stylistic in nature. For example, someone writes "Obama" above the president's head. We do not want such strokes to have parts pulled towards the consensus. We make a binary decision about whether the stroke is purely stylistic, in which case, we leave it alone. A stroke is defined to be purely stylistic if either of two conditions is met. Either the maximum vector length, $\max_i ||V_i||$, is greater than $5mm$ (approximately $1/10^{th}$ the screen width), or a stroke's direction is mostly orthogonal to the vector field. The latter term is measured with the average of absolute cosines $\frac{1}{N}\sum_i ||N(p_i - p_{i-1}) \cdot N(V(p_i))||$, where $N(\cdot)$ normalizes vectors. If the average falls below 1/2, the stroke is considered purely stylistic. Examples of purely stylistic strokes can be seen in our video.

5 Results

After smaller-scale trials, *DrawAFriend* was released publicly on January 8th, 2013. In 88 days there were 14,270 drawings. After the launch of *DrawAFriend*, our project proceeded in three phases. First we collected drawings for six celebrities to prime our initial correction vector field. Second we use the correction field to add an automatic stroke correction helper into the game. Lastly we crowd sourced the evaluation of our stroke corrector, by AB testing it within the game.

5.1 *DrawAFriend* Correction Vector Field

In order to integrate the correction vector field into the game, we needed to prime it with initial drawings. In four days players had downloaded the game over 2000 times and created 6373 drawings. In that time, players had already spent approximately 10 full 24 hour days drawing. We used these drawings to create the correction vector field.

Players were given the option of drawing Facebook friends or one of an initial set of six celebrities: Robert Downey Jr.[1], Angelina Jolie[2], Kim Kardashian[3], Barack Obama[4], Brad Pitt[5], or Kristen Stewart[6]. From the drawings generated in the first four days, we manually chose 611 celebrity drawings. We filtered by picking drawings in which players had made an attempt to accurately draw the eyes (the most intricate feature to draw). These 611 drawings were slightly less than 10% of the dataset. The rest of the dataset mostly consisted of drawings that players had drawn quickly, resembling a person like scribble. While this portion of the dataset could be used for analysis, we did not train our correction vector field on it. Figure 6 references statistics for these 611 drawings.

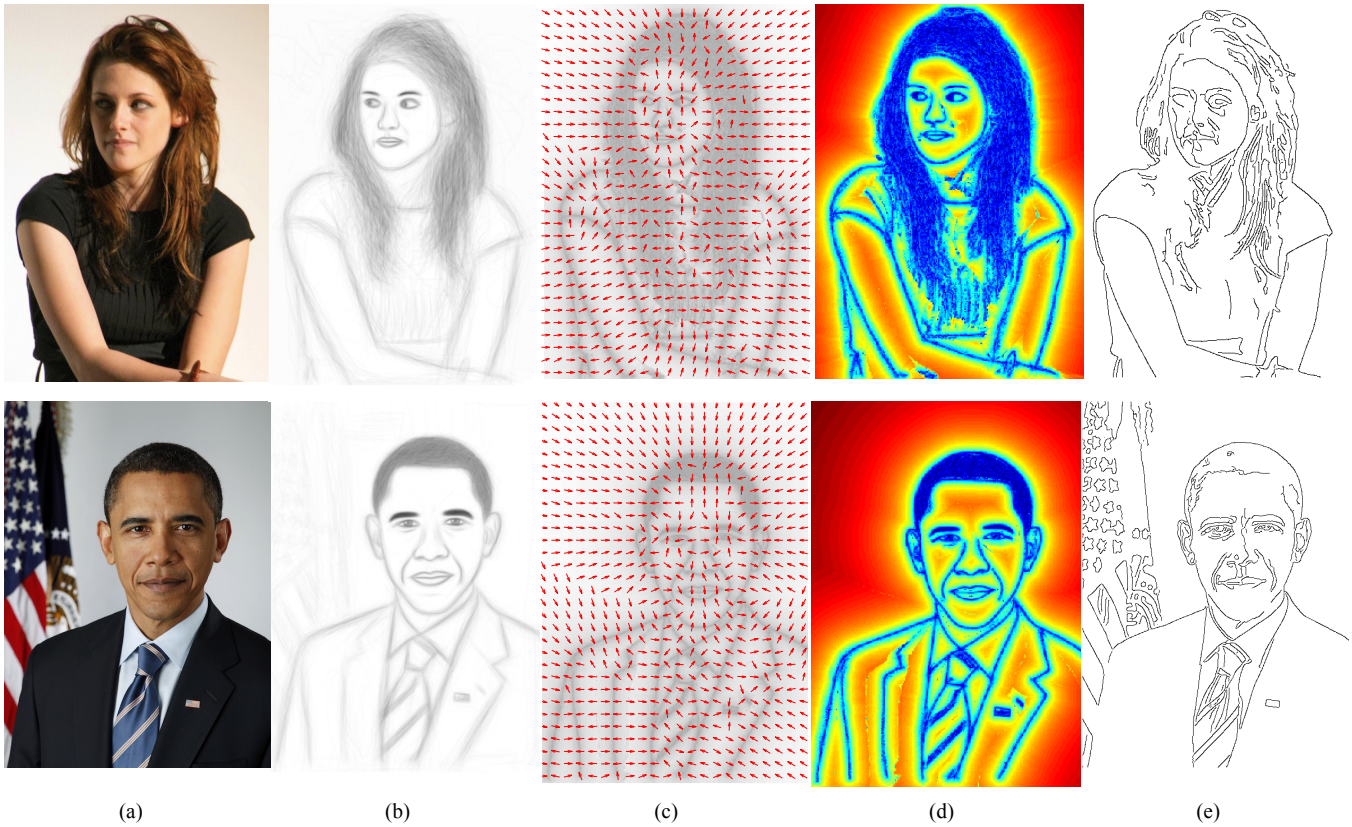
We ran our modified mean shift algorithm on this initial dataset to create the correction vector fields shown in Figure 9. Our MATLAB implementation took under 5 minutes for each celebrity. The great majority of the time was spent in un-optimized nearest neighbor search. The dimensions of the correction vector field are 460x320. In Figure 8 (top) we plot how our correction vector field converges for Brad Pitt. We do this by calculating the relative error between the vector field for a subset of the drawings and the vector field for all 124 drawings. There appears to be an inflection point around 25 drawings, where the correction vector field would probably work well. However every additional drawing still improves the correction field, implying that after 124 drawings it has not completely converged. We can also use the correction vector field to filter our drawings. Drawings whose points were on average 10 pixels away from the consensus were never actually portraits (most often they were the celebrity's name spelt out). We use this filtering method to exclude non-portraits in the user study described below.

5.2 Drawing Enhancements

We apply the correction vector field to interactively modify strokes during the drawing process. For every new stroke point added, a background thread is spun off to correct the new stroke. Once the thread completes if newer points have accumulated another thread is immediately spun off. Without adding user interface elements to the existing *DrawAFriend* UI, we seamlessly integrate stroke autocorrection. As the user draws, strokes are subtly corrected at interactive rates on an iPhone 4. In general, the fact that corrections are being applied is almost invisible to the user. Instead, strokes appear where the user *intended* to draw.

We also applied the correction vector field retrospectively to improve the existing database of user drawings. The drawings are already quite good, making the vector field corrections all the more impressive. While the algorithm improves the images, it does so without sacrificing style. For comparisons of the raw drawings and

[1]Meritano, E. "Robert Downey Jr." Photo. wikimedia.org. Apr 2008.
[2]Natt, C. "Angelina Jolie." Photo. wikimedia.org. Jun 2007.
[3]Shankbone, D. "Kim Kardashian." Photo. wikimedia.org. May 2005.
[4]Souza, P. "Barack Obama." Photo. wikimedia.org. Jan 2009.
[5]Boyd, B. "Brad Pitt." Photo. wikimedia.org. Mar 2008.
[6]Dispara, A. "Kristen Stewart." Photo. wikimedia.org. Oct 2008.



(a) (b) (c) (d) (e)

Figure 9: *DrawAFriend users draw one of six celebrities (a). We use our database of hundreds of drawings per subject – shown averaged in (b) – to precompute a* correction vector field *(c) enabling real-time drawing assistance on the iPhone. The magnitude of our vector field (d) reveals a consensus of artistic renderings strikingly different than what we could compute with automated methods, such as a canny edge detector (e).*

the corrected images please see Figure 10, our video, and the supplementary files.

5.3 Crowdsourced User Study

One key advantage of developing *DrawAFriend* as an online game is the ability to quickly deploy a study of users at scale. To test the effectiveness of the stroke auto-correction, we instrumented the game to enable a simple AB study. Every time a user started a drawing they were unknowingly placed in one of two groups. One group drew the celebrities as before. The other group (unbeknownst to them) drew with the stroke auto-correction on. By comparing these two groups, we can assess the effectiveness of the stroke auto-correction helper. Users drew one of the six celebrities. We use the consensus vector field to filter out drawings that could not be portraits. We record the geometry and timing of each stroke, all *undo*s, and the recognition rates of the players receiving the drawings.

5.3.1 User Study Results

After approximately one week, 500 players had "contributed" over 1,300 drawings to the user study. Using this data we assess recognition rates (measuring how good the drawings are from the perspective of others). We also analyze undo rates (providing an insight into how artists like their own drawings). Finally we measure average "distance" from the artistic consensus (measuring how carefully artists are drawing).

First we investigate *undoes* as an indication of how artists like their own drawing. Since players used a different number of strokes, we focus on undo rates, which is the number of strokes undone over the total number of strokes. For example if a user drew 100 strokes and pressed undo for 40 of them, this would have an undo rate of 40%. For our analysis we removed all drawings that had 0 undoes (assuming that this implied players did not know undo existed). This left 327 drawings auto-corrected and 295 that were not. We sorted each set of drawings by the undo rates from lowest to highest, and in Figure 7, plotted the undo rate for each percentile of sorted drawings (this normalizes for the different number of drawings in each set). The distribution of undo rates for drawings with fewer undoes track together for both sets with and without autocorrection. However the undo rates diverge further along the sorted lists. This seems to imply that our auto-corrector lowers the undo rate for more careful drawers (players that undo a larger percent of their strokes).

With auto-correct on, we observe a significant increase in the average distance between the actual uncorrected strokes and the consensus drawings (as measured by the *correction vector field*). We perform a two-sample Wilcoxon test of whether or not there is a difference in the mean of the correction vector field under the stroke points with auto correct on versus off. This shows a significant difference in the means, with a p-value of 1.653×10^{-6}. This suggests that artists do not need to draw as precisely. Figure 8 (bottom) shows the median distance to the consensus for each celebrity with a sample size of 570 auto-corrected and 533 not auto-corrected.

Interestingly, statistical analysis reveals that the auto-corrector does

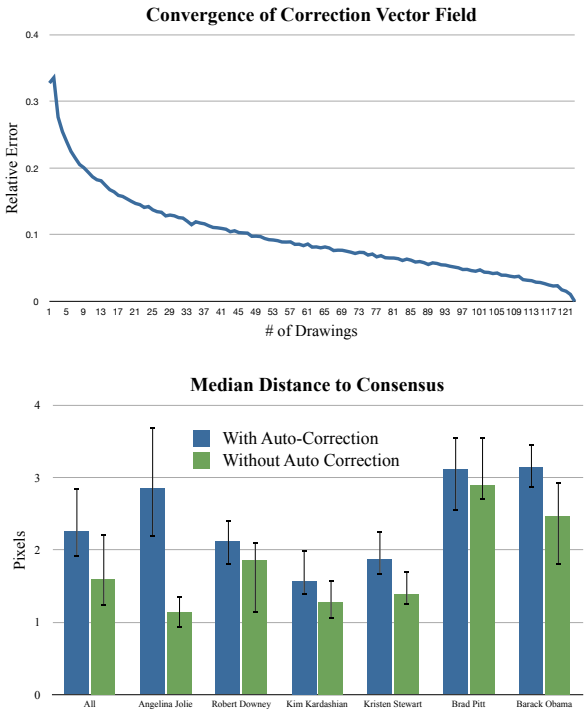


Figure 8: *Top: Convergence of the correction vector field as more images are added. Relative error is measured between the correction vector field built using X drawings and the correction vector field using all 124 drawings. (The final dip is an artifact caused by measuring convergence to a specific vector field computed with 124 drawings rather than to the "true" limit correction field. This is also why the error drops down to zero exactly.) Bottom: The median across drawings of the mean distance of stroke points to the consensus for each celebrity with and without auto- correction. Distance is measured by the value of the correction vector field under the strokes. The error bars mark the 60th and 40th percentile for each celebrity. Auto-correction universally leads to a greater distance, implying that users can be sloppier when auto-correction is activated.*

not significantly alter recognizability. Corrected drawings had a recognition rate of $37.5\% \pm 2.02\%$ while uncorrected drawings had an average recognition rate of $38.6\% \pm 2.11\%$ (again 570 and 533 drawings respectively). Running a permutation we get a p-value = .87, which implies that the two populations are uncharacteristically similar. We believe that our auto-corrector does not change players final drawing quality. Rather it makes reaching this level of quality easier by requiring less undoes and less accuracy.

6 Conclusion

We have presented a unique crowdsourcing approach of using social game mechanics to grow and use a dataset of drawings. We developed an iPhone game, *DrawAFriend*, specifically for the purpose of collecting drawing data. We then introduced a method to extract stroke-level artistic consensus from a large drawing corpus. The resulting *correction vector field* improves strokes in real-time without new interactions interfaces, while preserving artistic intent. We presented stroke correction on over 80 images which illustrate our method's ability to improve both aesthetics and recognizability. Lastly we evaluated our stroke correction algorithm through crowd sourcing, by observing how it influenced drawings "in the wild" by instrumenting the game itself. In general, we believe that *DrawAFriend* presents an unprecedented platform to perform quantitative drawing analysis at the Internet scale.

Our stroke consensus method has several drawbacks. While it allows us to improve strokes and filter out non-portrait drawings, it currently requires the initial drawings to be filtered by hand. Automated filtering is an area of future research. Even with stroke correction, many drawings are still not beautiful. We would like to study automatic aesthetic enhancement of strokes, including weight and texture. We also see several avenues to improve and generalize our correction vector field model. For example, the field depends on 2D position only. An anisotropic field, however, could correct strokes differently based on their orientation. We plan to *lift* the correction field to 3D (position plus orientation). This could be especially beneficial at stroke intersections which we do not explicitly model at present. Nevertheless, we have found that our simple isotropic field corrects strokes well in practice, especially around dominant lines for which there is great artistic consensus. Moving beyond the single consensus model, we envision automatically identifying and clustering by artistic "style". This would allow new drawings to be enhanced based on which style they most closely mimic to provide more nuanced stroke correction. Lastly we would like to extend our stroke correction algorithm to make it more general, perhaps by using an image-analogy-like approach to transfer our correction vector-field from one image to another.

Stroke correction represents just the tip of the iceberg for applications of the large (and growing) *DrawAFriend* image corpus. Thus far, we have explored only a small portion of the trove of data we are collecting. For example, measuring how quickly drawings are identified could shed light on which strokes are most salient for recognition. Studying when the user undoes strokes could provide clues as to which strokes are good or bad. While the stroke correction algorithm could have been implemented using just a rasterized image of a drawing, the vector representation will be useful for other applications. Analyzing stroke order could enable us to predict what the artist will draw next, potentially enabling completely new drawing interactions. Making this data available to the community, we hope to explore both these exciting ideas, and discover as-yet unknown applications of this rich dataset.

Acknowledgements

This work was supported by an NSF Graduate Research Fellowship, an NSF Career Award (IIS0953985), and by generous gifts from Google, Qualcomm, Adobe, Intel, and the Okawa Foundation. Thank you to Larry Zitnick for help with the initial game design, and to Martin Azizyan for helping with the statistical analysis. Also special thanks to Erdmuthe Limpaecher, Katherine Miller, Jackie Bello, Paul Kompfner, and Sam Grossberg for playtesting.

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

Figure 10: *Celebrity drawings: original strokes (bottom rows) and after automatic adjustment to consensus (top rows). (Please see the supplementary files for more examples and also for an interface so you can flip between them.)*

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.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (Jan.), 213–230.

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.

COMANICIU, D., AND MEER, P. 1999. Mean shift analysis and applications. *Computer Vision, IEEE International Conference on 2*, 1197.

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.

GINGOLD, Y., VOUGA, E., GRINSPUN, E., AND HIRSH, H. 2012. Diamonds from the rough: Improving drawing, painting, and singing via crowdsourcing. In *Proceedings of the AAAI Workshop on Human Computation (HCOMP)*.

LEE, Y. J., ZITNICK, L., AND COHEN, M. 2011. Shadowdraw: Real-time user guidance for freehand drawing. *ACM Trans. Graph. 22*, 3 (July), 879–887.

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.

ORBAY, G., AND KARA, L. B. 2011. Beautification of Design Sketches Using Trainable Stroke Clustering and Curve Fitting. *IEEE Transactions on Visualization and Computer Graphics 17*, 694–708.

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.

THIEL, Y., SINGH, K., AND BALAKRISHNAN, R. 2011. Elasticurves: exploiting stroke dynamics and inertia for the real-time neatening of sketched 2d curves. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '11, 383–392.

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.

ZITNICK, L. 2013. Handwriting beautification using tokens means. *ACM Trans. Graph.*.