EZ-sketching
Three-level optimization for error-tolerant image tracing
Su, Qingkun; Andy Li, Wing Ho; Wang, Jue; Fu, Hongbo

Published in:
ACM Transactions on Graphics

Published: 01/07/2014

Document Version:
Post-print, also known as Accepted Author Manuscript, Peer-reviewed or Author Final version

License:
Unspecified

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1145/2601097.2601202

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher's copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.
Abstract

We present a new image-guided drawing interface called EZ-Sketching, which uses a tracing paradigm and automatically corrects sketch lines roughly traced over an image by analyzing and utilizing the image features being traced. While previous edge snapping methods aim at optimizing individual strokes, we show that a co-analysis of multiple roughly placed nearby strokes better captures the user’s intent. We formulate automatic sketch improvement as a three-level optimization problem and present an efficient solution to it. EZ-Sketching can tolerate errors from various sources such as indirect control and inherently inaccurate input, and works well for sketching on touch devices with small screens using fingers. Our user study confirms that the drawings our approach helped generate show closer resemblance to the traced images, and are often aesthetically more pleasing.

1 Introduction

Freeform drawing gives artists complete freedom but is rather difficult to master. In contrast, the technique commonly known as “under painting” provides a target image as visual guidance and allows users to paint or draw over it. This method has been widely adopted by novice users as a means to improve their drawing skills, and to generate a plausible drawing that otherwise they cannot achieve. It has also been extensively used by professionals to improve the composition and layout accuracy of their art work derived from the image.

One popular drawing style is line drawing, i.e., using abstract strokes to depict the shapes of objects in a scene. When drawing over an image, the user usually wants input strokes to be aligned with object boundaries for drawing accuracy. However, accurate tracing is easy to achieve only when it is operated under direct manipulation and an input device is accurate, e.g., tracing a printed photo with a pencil. Due to the gap between the control space and the display space, indirect interaction makes it difficult for the user to accurately express his/her intention when drawing a stroke, even if the input device (e.g., a computer mouse or even a graphic tablet like Wacom Intuos) itself is of high precision. On the other hand, when the input device is inherently inaccurate (e.g., existing touch devices due to the well-known fat-finger problem), it is extremely difficult for the user to specify strokes that are accurate enough for a high quality sketch.

This work introduces a novel stroke refinement technique that automatically refines the location, orientation, and shape of input strokes by exploiting gradient features from the underlying image being traced (Figure 1). Automatically adjusting user strokes with respect to image gradients, i.e., edge snapping is not an easy task, since a desired image edge that the user intends to trace is often mixed with other, possibly much stronger ones in complex image regions, and is difficult to extract even with the state-of-the-art...
edge detectors (e.g., [Arbelaez et al. 2011]). Although various approaches have been proposed for tackling this problem, they all try to use local image features (i.e., edges near the user stroke) to find a locally-optimal solution for each user stroke independently, thus always face the danger of snapping to a wrong edge.

Our key idea is that while identifying a desired edge for a single stroke might be ambiguous, properly modeling the correlation among multiple input strokes in a larger image region can greatly resolve ambiguities. We thus propose a bottom-up optimization approach, which optimizes across three levels: local, semi-global, and global, to improve user strokes and at the same time preserve the original drawing style.

Our experiments show that the proposed technique is robust against various sources of error, e.g., errors caused by indirect control or inaccurate touch devices. It is flexible and applicable to many types of images, as shown in Figure 1. Moreover, our approach is simple and efficient, and can run at an interactive rate (see the accompanying video). Our user study confirms that the user drawings refined by our algorithm show closer resemblance to the traced images than the original ones, and are often visually more pleasing.

2 Related Work

Several systems have been developed in recent years to help users draw better. The iCanDraw system [Dixon et al. 2010] assists users in drawing human faces by providing corrective feedback. Jiarussi et al. [2013] extend this idea and provide more general guidance to draw arbitrary models. ShadowDraw [Lee et al. 2011] provides guidance for freeform drawing of objects by dynamically retrieving and blending relevant images from a large database with respect to the user’s current sketch. Such a computer-assisted drawing concept has also been explored for drawing, painting or even sculpture with traditional media and tools [Flagg and Rehg 2006; Laviole and Hachet 2012; Rivers et al. 2012]. Unlike these works, our system adopts the widely used “under painting” technique and utilizes an underlying image as visual guidance for drawing.

Gingold et al. [2012] show how to aggregate a crowdsourced set of low-quality inputs to produce better drawings, paintings, and songs. Limpacheur et al. [2013] take a similar crowdsourcing idea to correct sketches traced over a picture. That is, their work shares the same goal as ours but adopts a completely different approach. For a given picture, their approach first collects a large database of drawings by different artists through crowdsourcing and then uses a consensus of the collected drawings to correct new drawings. While their results are very positive, their technique is limited to pictures accompanied with a rich set of registered drawings and has been demonstrated for face portraits only. In contrast our technique demands the information from the picture being traced only and is thus more general.

Our local optimization step is largely inspired by the existing efforts on edge detection or contour extraction. Automatic edge detectors, even with the state-of-the-art techniques [Arbelaez et al. 2011], tend to produce noisy and/or inconsistent edges. The traditional interactive contour extraction methods such as active contour model [Kass et al. 1988] and intelligent scissor [Mortensen and Barrett 1995] typically require very precise user input for accurate selection. The recent work by Yang et al. [2013] allows the user to roughly scribble over the desired region to extract semantically dominant edges in it. We instead do not explicitly extract image edges given our goal of a drawing interface, where different input drawing styles should be retained. Kang et al. [2005] present an interactive system for generating artistic sketches from images, based on livewire algorithms [Mortensen and Barrett 1995]. However, like other contour tracing techniques, it does not solve the inherent ambiguity problem during local snapping. Instead, we show that a co-analysis of multiple roughly specified strokes often lead to results that better capture the user’s intent. This co-analysis procedure bears some resemblance to the sketch-based co-retrieval idea in [Xu et al. 2013].

Our work is also related to the previous interactive or automatic beautification systems [Igarashi et al. 1997; Orbay and Kara 2011; Zitnick 2013], none of which, however, uses a picture to guide the beautification process. We do not regard our technique as a beautification system, since drawing is a creative process and we try to preserve the original drawing style of the user. The main purpose of snapping input sketches to image edges is to improve the drawing accuracy, and it does not necessarily lead to aesthetic enhancement of strokes. However in many cases our algorithm indeed improves aesthetics, especially for users who are not good at freeform drawing.

Our semi-global optimization step essentially leads to correspondences between user-specified strokes and underlying image features. From this point of view, it is related to non-rigid image registration [Sýkora et al. 2009; Levi and Gotsman 2013] on a concept level. Unlike image registration, which often employs one-to-one feature point matching and computes a globally optimal transformation, our approach is essentially a candidate selection process, by first proposing multiple matching candidates through local optimization, and then selecting from them in the semi-global level. In addition, we have to tackle the unique issues such as temporal information (determined by the drawing order) and user interaction.

Another relevant topic is the vectorization of line drawings, where ambiguities also exist when performing vectorization locally. For clean drawings, the difficulty often lies near junctions, for which automatic non-local solutions [Bao and Fu 2012; Noris et al. 2013] often suffice. For drawings with over-traced scribbles, a possible solution is to first group over-traced line strokes into line groups, for example using Gabor filtering techniques. The Gabor filter used in [Bartolo et al. 2007] bears some resemblance to our direction-aware FDoG filter, with a key difference that our filter is driven by a user-specified stroke. There exist various techniques (e.g., [Kang et al. 2005; Baran et al. 2010]) for approximating non-smooth vector lines with high-quality curves. Such techniques might be incorporated into our method to achieve curves with high-order continuity.

3 Three-Level Sketch Stroke Optimization

As mentioned earlier, at the core of our system lies a three-level optimization method that automatically improves the user strokes based on image features. It consists of three components:

Local Optimization. Given a new user stroke, we first analyze its nearby image features to propose several snapping candidates (Section 3.1). Note that we do not simply choose one candidate as the final result as traditional edge snapping approaches do, due to the inherent ambiguity in local snapping.

Semi-global Optimization. Once the snapping candidates of a new stroke are determined, we jointly optimize the current stroke and its spatial-temporal neighboring ones, to determine their optimal snapping results together (Section 3.2). The nearby strokes will influence each other’s decision on candidate selection to produce a locally consistent result.

Global Optimization. Finally, for strokes that do not have good snapping candidates, we adjust their positions using a global optimization procedure (Section 3.3) based on an existing mesh deformation technique, to preserve the overall stroke layout.
3.1 Local Optimization

Given a parameterized user stroke \( s = \{p_1, p_2, ..., p_N\} \), where \( p_i \)'s are stroke vertex coordinates in the image space, the goal of the local optimization step is to find several good snapping candidates, each candidate being a new stroke, which satisfy two constraints: (1) the candidates should have similar shapes to the original stroke; and (2) each candidate should align well with some structural image edges. We denote these candidates as \( s'_j = \{p'_j,1, p'_j,2, ..., p'_j,N\} \), where \( j \in \{1, ..., M\} \), and \( p'_j,i \) is the new location of \( p_i \) in the \( j \)th candidate. The number of candidates for each stroke \( M \) is at most 4 in our experiments.

For each stroke point \( p_i \), we look in a local neighborhood window centered at it with radius \( r_s \) for finding its snapping candidates. We first apply a small threshold (0.2 after normalization) on the image gradient magnitudes in the window to remove all pixels that have small gradient magnitudes. In the remaining pixels, we identify those that have the locally maximal gradient magnitudes (i.e., pixels whose gradient magnitudes are the greatest in \( 3 \times 3 \) neighborhoods), and randomly sample \( K \leq 100 \) in our implementation) positions \( Q_i = \{q_{i,1}, q_{i,2}, ..., q_{i,K}\} \), as the snapping candidates for \( p_i \). We then construct a chain shaped graph \( G = (\mathcal{V}, \mathcal{E}) \), where \( \mathcal{V} \) contains all candidate points \( Q_i \). Unlike previous livewire techniques [Mortensen and Barrett 1995; Kang et al. 2005], we do not pin the stroke ends in place, since their initial positions might not be accurate. We thus add a virtual starting point \( Q_0 = \{q_{0,0}\} \), so that \( \mathcal{V} = \bigcup_{i=0}^{N} Q_i \), as shown in Figure 2. A complete bipartite graph is constructed between two neighboring node sets \( Q_i \) and \( Q_{i+1} \).

The weight of each edge \( e = (q_{i,k}, q_{i+1,k+1}) \in \mathcal{E} \) plays the most important role for this optimization process. We first want to encourage the input stroke to snap to nearby strong image edges that are parallel to the stroke direction. To identify such edges, inspired by the flow-based Difference-of-Gaussians (FDoG) filter proposed in [Kang et al. 2007], we define a DoG filter \( H \) along the stroke direction as:

\[
H(m,v) = \int_{-\infty}^{\infty} G_{\sigma_m}(y) \int_{-\infty}^{\infty} I((x,y)) f(x) dx dy,
\]

where \( I \) is the grayscale version of the input image. \( m \) and \( v \) are the midpoint and the unit direction vector of the graph edge \( e \) (see Figure 3), i.e., \( m = (q_{i,k} + q_{i+1,k+1})/2 \), \( v = c \cdot (q_{i+1,k+1} - q_{i,k}) \), where \( c \) is the normalization factor, \( I((x,y)) = m + xu + yv \), \( u \) is the unit vector that is perpendicular to \( v \), \( f(x) = G_{\sigma_v}(x) - pG_{\sigma_v}(x) \) is the difference of two Gaussian functions, as shown in Figure 3. \( H(m,v) \) thus can effectively find image edges that are parallel to the local stroke direction \( v \), thus better capturing the user’s intention. Compared to Kang et al.’s work [2007], the local filter direction is no longer determined by the image features, but provided by the user stroke. Figure 3 (bottom) shows the benefit of our direction-aware FDoG filter.

We then convert the filtering response \( H(m,v) \) into an edge weight term as:

\[
\tilde{H}(m,v) = \begin{cases} 1 + \tanh(H(m,v)) & \text{if } H(m,v) < 0, \\ 1 & \text{otherwise.} \end{cases}
\]

Specifically, if the middle point \( m \) is on an image edge whose direction is close to \( v \), \( H(m,v) \) will be a large negative value, thus \( \tilde{H}(m,v) \) in Equation 2 will be close to zero, leading to a small edge weight. On the contrary, an edge that is perpendicular to \( v \) will lead to a large edge weight, thus would not be preferred in this energy minimization framework.

We also expect that after local snapping, the shape and position of the input stroke would not change dramatically. This can be achieved by adding another term in the edge weight that penalizes the shifts of vertices. Combining these two terms, the total edge weight \( w_e \) is defined as:

\[
w_e = \frac{1}{r_s^2} \left\| (p_{i+1} - p_i) - (q_{i+1,k+1} - q_{i,k}) \right\|^2 + \alpha \tilde{H}(m,v),
\]

where \( \alpha \) is a balancing weight.

We use an iterative approach to search for snapping candidates. We aim at a small number of candidates but with large variations. Firstly, with the constructed graph \( G = (\mathcal{V}, \mathcal{E}) \), we compute the best snapping position \( s'_1 \) of the user stroke \( s \) by finding the shortest path \((q_{0,0}, q_{1,k}, q_{2,k}, ..., q_{N,k})\) from \( Q_0 \) to \( Q_N \), with the total energy of a path defined as the sum of edge weights along the path. This problem can be effectively solved using dynamic programming. Secondly, we check two conditions: a) whether the average edge weight of \( s'_1 \) is small enough \((\leq 0.07 \text{ in our implementation})\); b) whether \( s'_1 \) is significantly different from the existing ones\(^1\). \( s'_1 \) can be added to the final set of candidates only if both conditions are true. The first condition helps filter out snapping candidates.

\(^1\)This is achieved by computing the average distance between the corresponding vertices of the new candidate and existing ones. The condition is true only if the distance is large enough \((> r_s/4)\).
Figure 4: Input strokes (opaque) and their snapping candidates (semi-transparent). For clean visualization, not all candidates are visualized. Strokes with single candidates are shown in black. The size of the local neighborhood window (with radius $r_s$) is visualized by the gray circle in the top-left corner. Image credits: David Dennis; scarletgreen.

with low confidence. Note that it is possible that no snapping candidates satisfy this condition, which will leave the input stroke $s$ not to be adjusted until the global optimization step (Section 3.3). The second condition encourages the selected candidates to have large variations. Thirdly, no matter whether $s_1$ is selected or not, we remove all edges in $s_1$ from the graph. The removal of almost identical candidates lowers the computational cost for future steps. The above three steps are repeated until at most $M$ candidates are identified.

Figure 4 shows two examples of the final snapping candidate sets. When an input stroke has a high ambiguity, we keep it at its original position instead of taking a simple guess, as shown in the accompanying video. This conservative behavior is under the consideration that if simple guesses are taken early on, they will inevitably introduce frequent errors, which could easily annoy the user.

3.2 Semi-global Optimization

The local optimization step yields multiple snapping candidates for some user strokes. As discussed earlier, determining the right candidate from them is sometimes ambiguous if we only consider a single stroke. This is the major limitation of previous edge snapping approaches. In our system we propose a semi-global optimization method to jointly consider multiple strokes to determine their snapping positions.

While considering multiple strokes together is beneficial, we would like to point out that optimizing too many strokes together is also not ideal, as it brings stability and convergence problems to the system. Consider the other extreme where we always optimize all user strokes. The system then would become uncontrollable, as any new stroke the user adds would alter the existing results, even in faraway regions that have nothing to do with the local area that the user is focusing on at this moment. To avoid this problem our system only considers spatio-temporal neighboring strokes of the current user stroke. In other words, every new user stroke will invoke a semi-global optimization procedure, and only those strokes that are both spatial and temporal neighbors of the new one are considered in it.

The temporal neighboring relationship is determined by the drawing order, the most recently 3 strokes are considered as the temporal neighbors of the new stroke. To determine if they are spatially close to the new stroke, for each vertex on the new stroke, we find the closest vertex on a neighboring stroke and record their distance. If more than 30% of such distances are smaller than $1.5 \cdot r_s$, we then treat them as spatial neighbors. We call the current user stroke and its valid spatio-temporal neighbors as an active stroke set, denoted as $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$, and their candidate set $\mathcal{C} = \{c(i, j)\}$, where $c(i, j)$ means the $j$th candidate of the $i$th stroke $s_i$.

Our goal is to select the best candidate combination $\mathcal{T} = \{c(1, j_1), c(2, j_2), \ldots, c(m, j_m)\} \subset \mathcal{C}$ which satisfies the following three constraints: (1) each selected candidate has a small snapping cost in local optimization; (2) their spatial layout is similar to $\mathcal{S}$; and (3) $\mathcal{T}$ should not introduce new conflicts with other strokes that are not involved in the current optimization step. We thus define an energy term to describe each of the three constraints.

We first define $E_g(\mathcal{T})$ as the sum of the snapping costs computed in local optimization (i.e., edge weights $w_e$ defined in Equation 3) for snapping candidates in $\mathcal{T}$. By minimizing this term, the first constraint is satisfied.

To encourage layout consistency between $\mathcal{S}$ and $\mathcal{T}$, we define a 2D mesh $M$ by connecting each vertex $p_i$ in $\mathcal{S}$ to its spatially neighboring vertices, as shown in Figure 5. Note that the resulting mesh is not necessarily a triangular mesh. Similarly, we define a corresponding mesh $M'$, by connecting each vertex $c_i$ in $\mathcal{T}$ to its spatial neighboring vertices. We then define two energy terms as:

$$E_r(\mathcal{T}) = \sum_{c_{ij} \in \mathcal{M}} \eta \| (p_i - p_j) - (c_i - c_j) \|^2_2,$$

$$E_b(\mathcal{T}) = \sum_{i \in \mathcal{S}} \eta \| p_i - c_i \|^2_2,$$

where $\eta = 1/r_s^2$. Intuitively, $E_r(\mathcal{T})$ corresponds to the second constraint and measures the mesh deformation that is covered by $\mathcal{S}$. $E_b(\mathcal{T})$ measures the mesh deformation along the boundary of $\mathcal{S}$, where $\mathcal{S}$ contains all boundary vertices of $\mathcal{S}$, i.e., vertices that are connected with strokes that are not involved in the current optimization step in the mesh $M$. Minimizing this term will encourage the boundary vertices of $\mathcal{S}$ to stay at the same positions, thus avoiding introducing conflicts with other nearby strokes.

Combining the three terms together, the final energy minimization problem is defined as:

$$\min_{\mathcal{T} \subset \mathcal{C}} E_g(\mathcal{T}) + \beta(E_r(\mathcal{T}) + E_b(\mathcal{T})),$$

where $\beta$ is a balancing weight.
In our system, given that the size of $T$ is limited (the active stroke set contains at most 3 strokes in our implementation) and all energy terms can be computed efficiently, we simply use an exhaustive search to find the globally optimal solution to Equation 5. As we will demonstrate later, the exhaustive search can be done effectively in real-time in our test application. Increasing the size of $T$ could potentially slow down the computation. However as discussed earlier we intentionally limit the size of $T$ to avoid the convergence problem and provide a better user experience.

Figure 5 shows an example of how semi-global optimization can simultaneously adjust nearby strokes to achieve a semantically-compatible result (top right). In contrast, without semi-global optimization, local optimization alone cannot find the optimal snapping positions for all strokes, leading to a less satisfactory result shown in the bottom right corner of Figure 5.

3.3 Global Optimization

As discussed previously, some strokes might remain untouched during the semi-global optimization step, since their corresponding snapping energy is too high and thus they have no snapping candidate. We often have such strokes in the scenario where the user intentionally does not want to follow any image edges at all, e.g., when making hatching-like effects or applying decorative strokes.

However, since we adjust other strokes with snapping candidates, there is a potential topology issue: an adjusted stroke may cross some nearby untouched strokes that have been drawn earlier, which is undesirable (see Figure 6). To avoid this problem, we employ a global optimization step to adjust the positions of strokes with no snapping candidate so that the overall stroke topology remains unchanged, and the user’s drawing style is better preserved. In other words, we intend to preserve the original global layout of untouched strokes while responding to the change of strokes with snapping candidates.

The global optimization is achieved using a 2D deformation approach that is similar to the one used in the semi-global optimization step. However, it is worth emphasizing that the semi-global optimization essentially performs selection rather than deformation. Denote all the original strokes with snapping candidates as $S$ and their refined positions as $T$, we triangulate the image plane using all stroke vertices, and compute a mesh deformation according to the mapping from $S$ to $T$, using the method proposed by Zhou et al. [2005], which is originally designed for feature-preserving 3D mesh deformation. We then use the computed deformation field to move strokes with no snapping candidate to their adjusted positions, as shown in Figure 6.

4 Experimental Results and Evaluation

We implemented our method in C++ as a real-time drawing application running on the Windows platform. A touch-screen notebook (Aspire R7-572G with Intel(R) Core(TM) i5-4200U @1.6GHz 2.30GHz and 16GB RAM) running Windows 8.1 was used as the testing device. Our experiments show that EZ-Sketching is able to achieve real-time performance on this device: for a typical input stroke, it takes 0.03 seconds for local optimization, 0.01 seconds for semi-global optimization, and 0.5 seconds for global optimization. Note that the global optimization is more like a post-processing step and is not needed for every input stroke. The delay caused by stroke refinement is thus almost unnoticeable in a real drawing session.

We have tested the system on many images that contain a wide variety of scenes and objects, as shown in Figure 9. In Figure 7, we show some comparisons between our approach and previous edge snapping methods. The top and bottom examples are sketched using a computer mouse. Image credits: Andrea Campi.

To demonstrate the necessity and importance of the proposed semi-global optimization step, in Figure 8 we compare the snapping results of several examples (top two are from Figure 4) with and without semi-global optimization. The results show that if we only use local optimization, then dense strokes that are close to each other are often undesirably snapped to the same strong edges nearby. In
Figure 8: Comparing results generated by our system with and without using the semi-global optimization step. Left: user’s input. Middle: with local optimization only. Right: with both local and semi-global optimization. Note that without semi-global optimization, nearby dense strokes often collapse into the same position. Image credits: scarletgreen; David Dennis; Aubrey; Andrea Campi.

contrast, with the help of the semi-global optimization step, ambiguous strokes can snap to the correct positions and maintain their original spatial layout properties.

We used fixed parameter setting for generating all results. We set \( \alpha = 0.1 \) in Equation 3, \( \beta = 0.5 \) in Equation 5, and \( \sigma_s = 1.0, \sigma_m = 3.0, \rho = 0.99, X = 4, \) and \( Y = 7 \) in Equation 1. The radius of the local window introduced in Section 3.1 is set to be \( r_s = 7.5 \text{mm} \). For this parameter we purposely use a unit of physical measurement instead of pixels in order to cover the errors produced by “Fat Fingers”, a well-known problem in touch interfaces. Since a “fat finger” has a fixed physical size which is independent of the display resolution, defining this size using pixels would make the system depend more on a specific display and thus lose its generality. Defining it as a physical distance has shown to work well on multiple types of displays in our experiments.

We conducted a formal user study to evaluate the effectiveness of EZ-Sketching. The user study was divided into two parts. In User Study I, a number of participants were invited to create drawings with our system. It was designed to collect user feedback on the usability of our system. The drawings created before and after auto-refinement were then evaluated by a different group of participants in User Study II.

Figure 9: Images used in User Study I. Image credits: David Saddler; gynti46; Steve Cadman; Jeremy G.; Moyan Brenn; DeusXFlorida; Tim Pearce; John Ragai; Herv; Kevin Gibbons; Ian Barbour.

4.1 User Study I

Participants. We invited 12 volunteers (a1 to a12) to participate in the study: 7 men and 5 women, from 21 to 27 years old, 10 right-handed and 2 left-handed. The first 6 participants (a1 to a6) were with good drawing skills. They had either received professional drawing training, self-learned some drawing skills or had been practicing drawing for more than 5 times per week. The other 6 participants (a7-a12) had little drawing experience or knowledge.

Design and Procedure. We prepared 12 images (A to L) to be traced by the participants, as listed in Figure 9. The image set covers different types of objects or scenes, including man-made objects, architectures, plants, animals, human portraits etc. Each of the participants was assigned to trace 6 of the 12 images twice, once with naïve image tracing (i.e., without auto-refinement) and the other with EZ-Sketching. All the drawings were sketched by touching fingers on images displayed in a fixed window size of 5-inch (diagonal), which was used to simulate a mobile touch screen device.

Each participant was first given a short tutorial of the drawing interface and its two modes: with and without auto-refinement, followed by a short practice on an image that was out of the 6 assigned images not included in the formal test dataset. Next, each participant sketched the assigned 6 images with one of the drawing modes: with and without auto-refinement. The same set of images were then traced with the other mode. Finally, the participants were asked to complete a questionnaire.

We applied a Latin square design on image assignment and tracing order: 1) each participant was assigned to a different image set; 2) every image was drawn by 6 different participants: 3 with good drawing skills, 3 with little drawing experience; 3) half of the participants with and without good drawing skills started with auto-refinement first; 4) ordering and first-order carryover effects were balanced.

Results. After completing the sketching tasks, in the questionnaire
Each participant was asked to rate the drawing interface, with and without auto-refinement, in terms of ease of use, accuracy, and aesthetics of the results on a discrete scale from 1 (poorest) to 5 (best). Figure 11 plots the average rating scores. In terms of ease-of-use, only one participant (a10) preferred the interface without auto-refinement and another participant (a4) had no preference. The rest of the participants gave a higher rating to “with auto-refinement”. Paired t-test confirmed that the ratings with auto-refinement were significantly higher (t = 2.86, p < 0.02). Similar conclusions were applicable to the evaluation in terms of accuracy, the question of “how accurately your drawings capture the image structure”. Only two participants (a6 and a10) felt that the drawings made without auto-refinement were aesthetically more pleasing. However, no statistically significant difference was found in terms of aesthetics (t = 1.54, p = 0.15).

During the study our system also recorded the number of undoes and the time for drawing a stroke (from touch-down to touch-up). Our hypothesis was that EZ-Sketching could reduce the drawing time and the number of undoes. Wilcoxon rank-sum test found a statistically significant difference (z = 2.49, p = 0.024) in the drawing time per stroke between EZ-Sketching (0.431s) and naive tracing (0.511s). However, surprisingly, the average number of undoes for each drawing with auto-refinement was slightly higher, though the difference was not statistically significant (z = 0.87, p = 0.38). After looking into the drawings created by the participants, we found that since each participant was asked to trace an image twice, the two drawings often consisted of two different sets of lines. In this case, the comparison between with and without auto-refinement in the drawing time or the number of undoes would not be conclusive.

4.2 User Study II

Participants. In the second part of the user study, we invited a different group of participants to evaluate the drawings before and after auto-refinement, i.e., between the raw strokes as input to our algorithm and the refined strokes by our algorithm. Some of such drawing pairs are shown in Figure 12. In total 48 individuals participated, with the ages from 20 to 50 years old, and 33 men and 15 women. 15 of them reported that they had good drawing skills.

Tasks. Our initial attempt was to evaluate the results created by naive tracing and those created by our system. However we found this was quite a difficult task, because these two types of drawings usually contained completely different sets of strokes. We thus only asked the participants to evaluate the difference between a pair of drawings before and after auto-refinement, from User Study I. For such pairs of drawings, there naturally exists one-to-one stroke correspondence, thus their difference is only produced by the stroke refinement technique.

An online questionnaire was designed. Each pair of drawings before and after auto-refinement were placed at the same position, periodically switching between each of them for every 1.5 seconds. Each source image and the corresponding overlaid pair of drawings were placed side-by-side. Each participant was asked to pick which drawing better resembled the input image and which was aesthetically more pleasing.

To avoid fatigue, each participant was asked to evaluate only 12 out of the 72 pairs created in User Study I. The image pairs were randomly selected subject to the following constraints: 1) The 12 drawing pairs rated by individual participants were different; 2) the creators of the 12 drawing pairs were different. The presentation order of drawings was randomized.

Results. Each of the 72 drawing pairs was evaluated by at least 4 participants. The overall opinion favored the drawings refined by our system in both of the criteria, as summarized in Table 1. Specifically, 69.97% of votes were given to the auto-refinement results as more accurate. Paired t-test confirmed a statistically significant difference (t = 10.44, p < 0.01). 66.84% of the participants agreed that the auto-refinement results were aesthetically more pleasing (t = 5.87, p < 0.01). Around 33% of the original sketches were still considered to be more aesthetic. We speculate that this is because some sketches may appear to be more artistically appealing, or be considered as artistic exaggeration when they do not exactly follow the original image features. For example, people may think the curved building outlines in the forth column of Figure 12 to be artistically more appealing even if they obviously do not obey the image. Our system is not designed to capture such high-level aesthetic properties of sketches.

We further analyzed the opinions on the drawings created by the two groups of participants, with (a1-a6) and without (a7-a12) good drawing skill. As shown in Table 1, there were higher percentages of refined drawings from the participants with little drawing skill.

<table>
<thead>
<tr>
<th></th>
<th>accuracy</th>
<th>aesthetics</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1 - a6</td>
<td>64.93%</td>
<td>60.41%</td>
</tr>
<tr>
<td>a7 - a12</td>
<td>75.00%</td>
<td>73.26%</td>
</tr>
<tr>
<td>overall</td>
<td>69.97%</td>
<td>66.84%</td>
</tr>
</tbody>
</table>
voted as both more accurate and aesthetically more pleasing. Independent \( t \)-test confirmed that both differences were significant \((t = 2.65, p < 0.01)\) and \((t = 3.30, p < 0.01)\). We speculate that it is because users with less drawing skill produced more errors on the strokes, which gave more room for our method to improve, leading to more obvious difference before and after refinement. It suggests that our method would be more useful to users with less drawing skill. Nevertheless, for both participant groups, either with (a1-a6) or without (a7-a12) good drawing skill, the refined drawings received significantly more votes \((p < 0.01)\) in both accuracy and aesthetics, suggesting that our method would be helpful to users at different skill levels.

For each individual image (Figure 13), paired \( t \)-test confirmed that there were significantly more votes given to the refined drawings as more accurate \((p < 0.05)\), except for images B \((t = 0.57, p > 0.56)\), K \((t = 1.46, p > 0.15)\) and L \((t = 1.46, p > 0.15)\). Paired \( t \)-test also confirmed there were significantly more votes given to the refined drawings as aesthetically more pleasing for images A \((t = 2.77, p < 0.01)\), C \((t = 3.54, p < 0.01)\), D \((t = 5.49, p < 0.01)\), E \((t = 2.77, p < 0.01)\), G \((t = 4.42, p < 0.01)\) and H \((t = 3.96, p < 0.01)\). However, no significant preference was found in terms of aesthetics for the other images (B, F, I, J, K, L).

4.3 Limitations

To better understand the limitations of the system, we examine the examples that did not perform well in the user study. In particular, our system only achieved minimal success on test image L. As shown in Figure 14 (left), this is largely because the image contains a lot of weak edges, and the object boundary is quite weak compared with the background clutter. Thus, our system may snap some boundary strokes to wrong edges in this difficult case. In contrast, human perception can easily tell the boundary by using higher level semantics. This suggests that our system could potentially incorporate high-level object recognition methods for more semantic edge snapping.

Our system is designed to be conservative to preserve the original drawing style. Hence if the original sketch contains large errors, our refined result will not be accurate as well. Such an example is shown in Figure 14 (right). However this might be a desired feature in practice, as the user still gets correct feedback for improving his/her drawing skills.

5 Conclusion and Future Work

We presented EZ-Sketching, a new drawing interface with the power of automatic stroke refinement, using a novel three-level optimization framework. Our system automatically improves the accuracy of the user’s drawing while maintaining the original style. We believe that our technique will encourage novice users with limited drawing skills to create drawings under an under-painting paradigm, especially on touch devices.

As future work, we plan to port our implementation to popular mobile platforms such as Android and iOS and release the app in the app stores, so that EZ-Sketching can reach a larger group of users, enabling a larger-scale evaluation. Our idea of modeling the interaction of multiple strokes might inspire other applications such as scribble-based image segmentation. How to provide snapping suggestions in the scenario where multiple reference images are available [Lee et al. 2011], is also an interesting direction to explore.


References


