Interactive Vectorization for Graphic Design

HARRY POTTER, HERMIONE GRANGER, and RON WEASLEY, Hogwarts School of Witchcraft and Wizardry

Designers and artists have longed for intelligent authoring tools which could capture and predict user intents and automatically complete desired operations. In practice the largest barrier to create digital content is not technological limitations, but the tedious effort required to create even the most prosaic objects.

In this paper, we propose a general framework for interactive design that offer high-level powerful control than simple parametric curve editing. More specifically, our system takes 2-dimensional parametric graphical content such as font-faces and logos which current delicate design relies heavily on. By capturing and understanding users' ambiguous and heursitic intents from either users' imprecise sketch or direct low-level manipulation, our system can automatically refine current content by offer aesthetically better hints and suggestions.

Through our experiments ... we observe that with our system both novice users and experienced designers produced better design by evaluation.

CCS Concepts: • **Computing methodologies → Computer graphics**;

Additional Key Words and Phrases: research

ACM Reference format:

Harry Potter, Hermione Granger, and Ron Weasley. 2017. Interactive Vectorization for Graphic Design. *ACM Trans. Graph.* 36, 4, Article 1 (July 2017), 6 pages.

https://doi.org/

1 INTRODUCTION

Design requires heavy manual payload, especially today when delicate graphical design work such as fonts and logos are defined by carefully tuned parametric vector images. Weeks of both mental and physical efforts could have been spent for creating even the most prosaic digital objects. Given the advanced technology of computer software in modelling, rendering and simulating, creating digital content still remains huge amount of tedious work. Machines today are getting so computationaly powerful, yet little progress has been made in helping human users on easier and faster design and artistic creation than ten years ago.

Modern software with mouse-menu interface or stylus still require designers to do tedious work such as alignment and repetition. On the other hand, modern computers now have strong power for extremely advanced computation, which allows machine today to do quite clever things like play chess [Silver et al. 2016] and video games [Mnih et al. 2013]. Computers have the potential to offer smarter working experience for design process. While human have intents of design prototypes and great aesthetic tastes, and modern computers can provide efficient computation, in this paper, we leverage the machine learning techniques and artifical interligence to combine both human and machine's strength.

In this paper, we propose a general framework for constructing a automatic system to aid the design process. For a deep and thoroughly test on our framework, we applied our framework into graphic design of vector image, especially typography and logo design.

Making computer capable of doing creative design is not a novel idea. Both HCI and Graphics community have been working on this subject for users to create digital content easily with high quality, such as autocomplete repetitions [Xing et al. 2014, 2015] and automatic layout [O'Donovan et al. 2014, 2015] and pattern [Gieseke et al. 2017]. But these systems all seem been hard coded for some determined usages while ignoring that in practice users may have various behaviors and objectives in the design process. However, few works focused on improving the design process, taking the process and interaction between machine and users seriously to provide more creative design workflows.

We provided a framework of design aiding system for designers to efficiently prototype and complete typography and logo design by understanding intents from user's actions. Our system can provide novel suggestions on potentially good design ideas and also refine what has already been created by the user by analyzing user's behaviors, such as alignment and smoothing. Our system allows users to quickly create digital content from scratch, and largely improve the quality of design through our suggestions and refinement.

Our framework consists of two major component: a capturer for understanding user's intent and a generator which refines, beautifies and brainstorms current content. Our system leverages the potential of the recurrent neural network in understanding user's ambiguous design intent by encoding user's actions into internal representations. Given user's action, our model offers novel design solutions by state-of-the-art generating model. A user can accept the new idea and continue to work on it in a recursive setting, as design work is non-deterministic and often requires frequent and repeated revisions.

We propose a general framework combines the recurrent neural network to model users' intents and a powerful generative model to offer diverse and novel design solution. By encoding users' action as a sequence of actions, our system starts to understand user's intent and provide crucial suggestions.

Our system differs from other work mostly because it is not specifically designed for a single purpose, such as autocomplete or element alignment [O'Donovan et al. 2014, 2015; Xing et al. 2014, 2015].

Also, previous works have all focus on pixel bitmaps or stroke-based data, while current designers heavily based their delicate work on vector-based images using vector design software like Adobe Illustrator. Our algorithms take this form as our task, which makes it more important in practice.

We evaluate our system on novice and experienced designers showing that the quality and efficiency of design are improved by our system. Through the experiments, the machine starts to understand user's vague intent from even scratch and automatically providing suggestions and refinement on the current work. By understanding users' intent and providing suggestions and refinements, we could make better design solutions via better human and machine collaboration.

2 RELATED WORK

Font [Campbell and Kautz 2014; O'Donovan et al. 2014]

Handwriting [Haines et al. 2016]

RNN for vector workflows [Graves 2013; Ha and Eck 2017]

Layout design papers by Aaron Hertzmmann (plus others).

Element alignment papers by Hongbo Fu (plus others).

intro this is a survey of related work on graphic design. The following parts are some specigic problems and methods.

My thoughts on it Mainly the graphic content is edited in these forms

- pixel-based image.
- stroke-based image (there are points along the track of strokes and can be easliy rendered to pixel data.)
- outline image(or glygh in typography), consists of control points.

I want to focus on the last kind of graphic content. To make the design process of outline glygh better, automatically complete some user action.

interacive design on contents Interactive means one user sees result given user input in real time. And certain content is also modified. This change can be mainly categorized in two kind: refine current content or change it thoroughly.

work like [Peng et al. 2017; Xing et al. 2014, 2015] Autocomplete and [Thiel et al. 2011] Neating the strokes, focus on refine content which has been already done, by analyzing user's intent and other hand-designed principles to refine content.

[Suveeranont and Igarashi 2010] takes the outline of a single character drawn by the user and computes blending weights from a database to reproduce the given outline. The weights are then applied to all characters to synthesize the new font. This system was built to accelerate the font design process with minimal user design cost while preserving the overall consistency style of font.

Another direction for interactive design is to explore and find alternatives

Exploring alternatives is very important in the design process. [Gross 1996] present a prototyping interface which allows users to sketch drawings and store alternatives. [Terry et al. 2004] present an interaction technique allowing users to manipulate alternative variants in the same window during the design process (together or not together). [Dow et al. 2010] finds out that forcing users to create multiple design variants, instead of refining a single design, leads to improved results.

[Lee et al. 2010] present a web-design interface exposing user in a enviroment where one could browse related good design examples in order to learn from examples to help user produce better designs. [Merrell et al. 2011] demonstrate interactive suggestions for furniture layouts.

Many works [O'Donovan et al. 2014, 2015] propose automatic ways to generate and adjust graphic layout. Details in layout section.

convert stroke or bitmap to vector image [Xie et al. 2017] discusses how to enable users to use rough brush for detailed and interactive vectorization of bitmap image. Mainly it first identify potential candidate edges, and then construct a hierarchical edge map. [Xie et al. 2017] seems of little help in our setting because it focuses on natrual image, while our job is to take users' stroke as indications and transform it into some sort of parametric curves.

[Richardt et al. 2014] discuss how to extract the parameters for semi-transparent gradient layers of a vector image. Image segmentation and semantic annotation is completely manual. It also seems of little help to us.

[Favreau et al. 2016] talks about the trade-off of fidelity to the input bitmap and the simplicity to the output vector image. It seems that many vectorization methods tends to produce over-complex vector curves and control points.

[Noris et al. 2013] uses clustering for stroke disambiguation, yielding cleaner images and then extract topology from the drawing.

[Zhang et al. 2009] talks about extracting vector image for carton animations.

I read this papers and find that they seem not so relevent to our project. For fonts, the outline curve could be extracted directly from the fontface format. And for user stroke-like input, we can also directly track the stroke.

method of learning representation of graphic content [Shamir and Rappoport 1998] views outlined font as more higher-levels than points and lines, but serifs, bars, arcs and joints. It ([Shamir and Rappoport 1998]) develops a visually GUI to directly manipulate the font glyph.

While [Hu and Hersch 2001] follows the approach and extends further on how to composite a shape, like building blocks of parameterized template shapes. Note that in this two paper [Hu and Hersch 2001; Shamir and Rappoport 1998] the font templated is defined by some special shape or area in the font, like bars, top serif, foot serif, etc. which is manually designed for English Font, which is the major constraints to them/

This is quite different from work [Xu et al. 2004] and its followers [Li and Zhou 2013; Liu et al. 2012; Xu et al. 2009] which focus on stroke based content.

[Jakubiak et al. 2006] also focuses on stroke-based content. This paper actually focuses on how to define a data format to use, like Chinese characters.

There are industry standards for specifying fonts exactly used for more advanced interpolation between different glyph shapes, including 'OpenType GX' and 'Adobe Multiple Masters'[Systems 1997]. In the case of the latter, these are a set of outlines defined by bezier curve control point which is in exact correspondence. Thus it makes it easy to have a weighted interplolation over fonts

[Campbell and Kautz 2014] decribes a way to get a full manifold of fonts so we can browse in a latent variable space to generate interpolations of fonts. [O'Donovan et al. 2014] proposed approaches to browse fonts in a more reasonable way by high-level descrptions, hierarchy and similarity and made a good example on how to build a software for users to select fonts.

There are also works [Xu et al. 2013] done to generate 3d model from 2d sketch.

method of content shape-preserving deformation Many methods were proposed to preserve some characteristic while being edited [Hsu et al. 1993; Igarashi et al. 2005; Nealen et al. 2006; Shamir and Rappoport 1999; Sorkine et al. 2004].

[Igarashi et al. 2005] develops a interactive application allowing users can directly manipulate 2-d image while preserving the rigidity, without using a skeleton . A elegant two-setp closed form algorithm is used to minize the distortion.

[Nealen et al. 2006; Sorkine et al. 2004] are general models using Laplacian and positional constraints. [Nealen et al. 2006] propose a framework for 3-d shape (triangle mesh) optimization. Smoothing the mesh while preserving the details [Sorkine et al. 2004] performs interactive editing of a surface (or specificly, free-form deformation)

They developed the Laplacian method in order to edit while preserve some very import details for 2-d object [Suveeranont and Igarashi 2010].

There also works on just strokes, curves [Hertzmann et al. 2002] instead of mesh.

In short, shape-preserved deformation is general treated as a optimization problem. Many method (like mentioned above) can be reduced to a energy optimization problem, the trick seems to be the definition of energy after all.

calligraphy and font synthesis [Xu et al. 2004] is the earliest work I found in this subject for calligraphy, a automatic system is built to generate Chinese calligraphy.

Followed are [Li and Zhou 2013; Liu et al. 2012; Xu et al. 2009] which focus on how to create stroke-based calligraphy using different approaches.

[Miyazaki et al. 2017] focus on how to generate typographic font using a small subset, which is more general than works above.

[Zitnick 2013] automatically generate synthesis of letters from a user stylus by averaging multilple instances of the same strokes. For this method, an essential part is how to identify the match of strokes.

[Haines et al. 2016] also argues the tablet stylus for that even experienced user do not write well or electric devices and propose a system to generate hand-writing letters from a subset

For more or less, I found that these papers concentrate on three different problem settings.

- generate font using some parameters from handwritten or typographic font.
- blending through different fonts
- from a small subset, extract and find strokes and then a compositing mechanism (with blending and deformation)

Layout design Many works [O'Donovan et al. 2014, 2015] propose automatic ways to generate and adjust graphic layout. These approaches seems to have a lot of hand-designed rules involved, like the aligh principle, color, importance, etc.

While [O'Donovan et al. 2014] focus on the arrangements of location or relocate graphic elements. [O'Donovan et al. 2015] proposed a application system which can provide layout suggestions from two aspects: one kind of suggestion is to slightly refine the current layout, another is to brainstorm a little with a huge change of layouts. It [O'Donovan et al. 2015] use an energy-based model to generate designs that en- code design principles such as symmetry, alignment, and overlap. User constraints are used to infer the designer's in- tent, and to make refinement suggestions on the current layout.

[Xu et al. 2015] explore the correlation of graphic elements as a group to enhance layout design.

others [Baluja 2017] talks about discriminating font and generate font givn just 4 letters as a small subset.

[Wang et al. 2015], discrimiate the font type from a image

Generative Models [Liu et al. 2017; Zhu et al. 2016] use GANs to generate contents using user's input as contraints.

[Ha and Eck 2017] use a dataset of vector images, and also a generative model to make images, freely. Can also generate given conditions like a certain category. But do not support interacive editing.

[Liu et al. 2017] uses generative model to do interacive 3-d model editing. Very impressive.

state-of-the-art methods on learning representations Here lists some state-of-the-art method on learning represenatations of something I find interesting while I am reading. Hope that some of these method would be useful

[Larsen et al. 2015] use GAN and VAE to encode data into latebt variable. Proved to be a very good similarity metric. Might be useful in learning unsupervised representations of a object.

Some Good user interface [Zhang et al. 2017] use user input on some pixel to do automatic and still controllable colorization of gray-scale image.

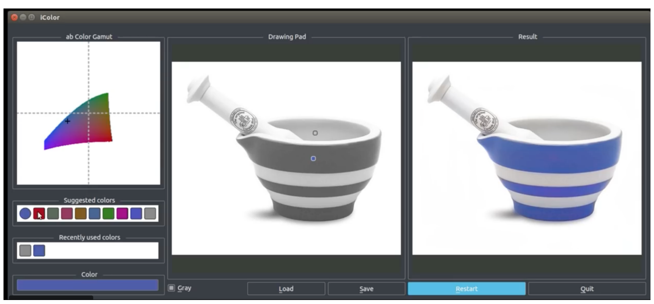


Fig. 1. *Example figure.* User interafce for [Zhang et al. 2017], Real-Time User-Guided Image Colorization with Learned Deep Priors. The left are color platte and suggestions. In the right is real-time result.

[Zhu et al. 2016] use user input stroke to draw images.

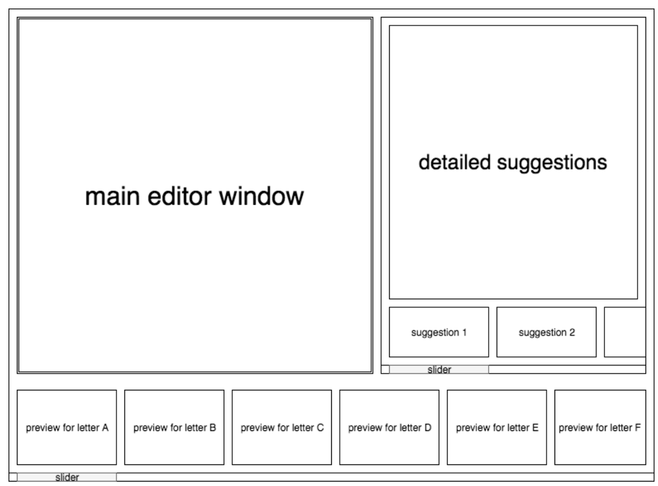
3 USER INTERFACE

Input and output. Data being manipulated should be a set of control points. These points have correlations, which means that each other could be connected to one another or not (professionally these points are called anchor points, and the connections are called paths). Each point has several types: **corner**, **smooth** and **change of direction**.

There is a good video illustrating what is anchor points on youtube <Understanding anchor points in Illustrator>.

Our machine should be okay to get the set of anchor points, as well as user's action history as input to generate a somehow-better set of anchor points. (There is one advanced usage: user can indicate the area of anchor points to show that this kind of design is good, as an input to the machine to generate a new set of anchor points.)

Fig. 2. *Example figure.* User interafce for [Zhu et al. 2016], Generative Visual Manipulation on the Natural Image Manifold. It provides several suggenstions in the right and you can select on it.



(a) simple diagram

Fig. 3. *simple illustration for interface.* (a) is a simple diagram of user interface. There are mainly four parts: main window, suggestions, details of suggestions, and below is the meta-view of the whole alphabet. (If we are doing logo design, then we should ignore the last part.)

Interactions. (It is worth noticing that, despite what we actually see is the path, the final output is still anchor points because the anchor points define the parameters of the path.) So the result of our system is anchor points.

these are actions user could make:

- Main usage: A user could draw on the pad like using a pen, our system could recognize and interpret the stroke to generate suggestions fitting the stroke like [Zhu et al. 2016], except that we are not generating pixel data (our output is a set of anchor points).

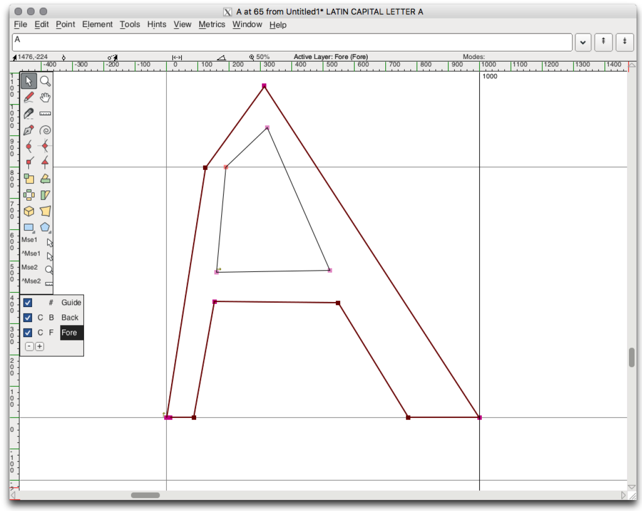
- In order to pick the things he likes, he can use a brush to indicate some area in the suggestions that he likes the best, as some sort of guidance for the machine to generate more appealing suggestions.
- As basic low-level manipulations, users can create new anchor points, or drag and move anchor points. The type of anchor point could also be changed.
- He can pick, view and select our provided suggestions.

These actions will all recorded as history. The editing history along with the current set of anchor points will be fed as input to the black-box machine to generate a new set of anchor points. The type and location of output anchor points may be modified, or a point itself may even be eliminated.

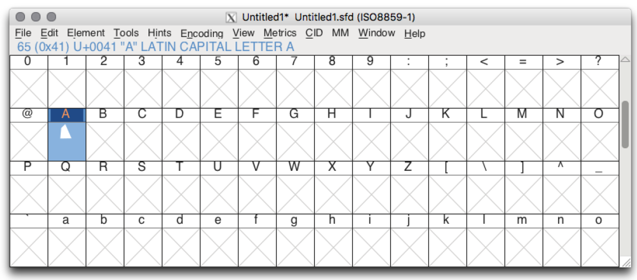
***interface*.**

***UI*.** The interface should like a combination of a font editor and a suggestion sidebar on the right which should display a set of suggestions. Users can pick and view the details of suggested changes (in real-time on the main editor window) and decide to accept or not. See Figure 3

For details of the main editor, there should be a toolbox. See Figure 4



(a) working window of a single character



(b) meta-view of whole alphabet

Fig. 4. *simple illustration for main editor interface of fontforge.* (a) is the editing window of a single letter A. (b) is the meta window of editor, from which you can select a single character to enter into (a)

4 METHOD

5 RESULTS

6 LIMITATIONS AND FUTURE WORK

REFERENCES

Shumeet Baluja. 2017. Learning typographic style: from discrimination to synthesis. *Machine Vision and Applications* (2017), 1–18.

Neill D. F. Campbell and Jan Kautz. 2014. Learning a Manifold of Fonts. *ACM Trans. Graph.* 33, 4, Article 91 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601212

Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L. Schwartz, and Scott R. Klemmer. 2010. Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-efficacy. *ACM Trans. Comput.-Hum. Interact.* 17, 4, Article 18 (Dec. 2010), 24 pages. https://doi.org/10.1145/1879831.1879836

Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2016. Fidelity vs. Simplicity: A Global Approach to Line Drawing Vectorization. *ACM Trans. Graph.* 35, 4, Article 120 (July 2016), 10 pages. https://doi.org/10.1145/2897824.2925946

Lena Gieseke, Paul Asente, Jingwan Lu, and Martin Fuchs. 2017. Organized Order in Ornamentation. In *CAE '17*. Article 4, 9 pages. https://doi.org/10.1145/3092912.3092913

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850 (2013). http://arxiv.org/abs/1308.0850

Mark D Gross. 1996. Ambiguous Intentions: a Paper-like Interface for Creative Design. ACM Press, 183–192.

David Ha and Douglas Eck. 2017. A Neural Representation of Sketch Drawings. *ArXiv e-prints* (April 2017). arXiv:1704.03477

Tom S. F. Haines, Oisin Mac Aodha, and Gabriel J. Brostow. 2016. My Text in Your Handwriting. *ACM Trans. Graph.* 35, 3, Article 26 (May 2016), 18 pages. https://doi.org/10.1145/2886099

Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M Seitz. 2002. Curve analogies. In *Proceedings of the 13th Eurographics workshop on Rendering*. Eurographics Association, 233–246.

Siu Chi Hsu, Irene HH Lee, and Neil E Wiseman. 1993. Skeletal strokes. In *Proceedings of the 6th annual ACM symposium on User interface software and technology*. ACM, 197–206.

Changyuan Hu and Roger D Hersch. 2001. Parameterizable fonts based on shape components. *IEEE Computer Graphics and Applications* 21, 3 (2001), 70–85.

Takeo Igarashi, Tomer Moscovich, and John F Hughes. 2005. As-rigid-as-possible shape manipulation. In *ACM transactions on Graphics (TOG)*, Vol. 24. ACM, 1134–1141.

Elena J Jakubiak, Ronald N Perry, and Sarah F Frisken. 2006. An improved representation for stroke-based fonts. In *Proceedings of ACM SIGGRAPH*, Vol. 4.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. 2015. Autoencoding beyond pixels using a learned similarity metric. *CoRR* abs/1512.09300 (2015). http://arxiv.org/abs/1512.09300

Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with Interactive Example Galleries. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 2257–2266. https://doi.org/10.1145/1753326.1753667

Wei Li and Changle Zhou. 2013. Automatic Creation of Artistic Chinese Calligraphy. *JSW* 8 (2013), 3048–3054.

Jerry Liu, Fisher Yu, and Thomas A. Funkhouser. 2017. Interactive 3D Modeling with a Generative Adversarial Network. *CoRR* abs/1706.05170 (2017). http://arxiv.org/abs/1706.05170

Peng Liu, Songhua Xu, and Shujin Lin. 2012. Automatic Generation of Personalized Chinese Handwriting Characters. *2012 Fourth International Conference on Digital Home* (2012), 109–116.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive Furniture Layout Using Interior Design Guidelines. *ACM Trans. Graph.* 30, 4, Article 87 (July 2011), 10 pages. https://doi.org/10.1145/2010324.1964982

Tomo Miyazaki, Tatsunori Tsuchiya, Yoshihiro Sugaya, Shinichiro Omachi, Masakazu Iwamura, Seiichi Uchida, and Koichi Kise. 2017. Automatic Generation of Typographic Font from a Small Font Subset. *CoRR* abs/1701.05703 (2017).

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013).

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2006. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*. ACM, 381–389.

Gioacchino Noris, Alexander Hornung, Robert W. Sumner, Maryann Simmons, and Markus Gross. 2013. Topology-driven Vectorization of Clean Line Drawings. *ACM Trans. Graph.* 32, 1, Article 4 (Feb. 2013), 11 pages. https://doi.org/10.1145/2421636.2421640

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2014. Learning Layouts for Single-Page Graphic Designs. *Visualization and Computer Graphics, IEEE Transactions on* 20, 8 (2014), 1200–1213.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. Designscape: Design with interactive layout suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1221–1224.

Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. 2014. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Trans. Graph.* 33, 4, Article 92 (July 2014), 9 pages. https://doi.org/10.1145/2601097.2601110

Mengqi Peng, Jun Xing, and Li-Yi Wei. 2017. Autocomplete 3D Sculpting. *CoRR* abs/1703.10405 (2017). http://arxiv.org/abs/1703.10405

C. Richardt, J. Lopez-Moreno, A. Bousseau, M. Agrawala, and G. Drettakis. 2014. Vectorising Bitmaps into Semi-Transparent Gradient Layers. *Comput. Graph. Forum* 33, 4 (July 2014), 11–19. https://doi.org/10.1111/cgf.12408

Ariel Shamir and Ari Rappoport. 1998. Feature-based design of fonts using constraints. In *Electronic Publishing, Artistic Imaging, and Digital Typography*. Springer, 93–108.

Ariel Shamir and Ari Rappoport. 1999. Compacting oriental fonts by optimizing parametric elements. *The Visual Computer* 15, 6 (1999), 302–318.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 7587 (2016), 484–9.

Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM, 175–184.

Rapee Suveeranont and Takeo Igarashi. 2010. Example-based Automatic Font Generation. In *Proceedings of the 10th International Conference on Smart Graphics (SG'10)*. Springer-Verlag, Berlin, Heidelberg, 127–138. http://dl.acm.org/citation.cfm?id=1894345.1894361

Adobe Systems. 1997. Designing multiple master typefaces. (1997).

Michael Terry, Elizabeth D. Mynatt, Kumiyo Nakakoji, and Yasuhiro Yamamoto. 2004. Variation in Element and Action: Supporting Simultaneous Development of Alternative Solutions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 711–718. https://doi.org/10.1145/985692.985782

Yannick Thiel, Karan Singh, and Ravin Balakrishnan. 2011. Elasticurves: Exploiting Stroke Dynamics and Inertia for the Real-time Neatening of Sketched 2D Curves. In *UIST '11*. 383–392. https://doi.org/10.1145/2047196.2047246

Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt, and Thomas S. Huang. 2015. DeepFont: Identify Your Font from An Image. *CoRR* abs/1507.03196 (2015). http://arxiv.org/abs/1507.03196

Jun Xie, Holger Winnemöller, Wilmot Li, and Stephen Schiller. 2017. Interactive Vectorization. In *CHI '17*. 6695–6705. https://doi.org/10.1145/3025453.3025872

Jun Xing, Hsiang-Ting Chen, and Li-Yi Wei. 2014. Autocomplete Painting Repetitions. *ACM Trans. Graph.* 33, 6, Article 172 (Nov. 2014), 11 pages. https://doi.org/10.1145/2661229.2661247

Jun Xing, Li-Yi Wei, Takaaki Shiratori, and Koji Yatani. 2015. Autocomplete hand-drawn animations. *ACM Trans. Graph.* 34 (2015), 169:1–169:11.

Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. 2013. Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 123.

Pengfei Xu, Hongbo Fu, Chiew-Lan Tai, and Takeo Igarashi. 2015. GACA: Group-aware command-based arrangement of graphic elements. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2787–2795.

Songhua Xu, Tao Jin, Hao Jiang, and Francis Chi-Moon Lau. 2009. Automatic Generation of Personal Chinese Handwriting by Capturing the Characteristics of Personal Handwriting. In *IAAI*.

Songhua Xu, Francis Chi-Moon Lau, William Kwok-Wai Cheung, and Yunhe Pan. 2004. Automatic generation of artistic chinese calligraphy. *IEEE Intelligent Systems* 20 (2004), 32–39.

Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. 2017. Real-Time User-Guided Image Colorization with Learned Deep Priors. *CoRR* abs/1705.02999 (2017). http://arxiv.org/abs/1705.02999

Song-Hai Zhang, Tao Chen, Yi-Fei Zhang, Shi-Min Hu, and Ralph R. Martin. 2009. Vectorizing Cartoon Animations. *IEEE Transactions on Visualization and Computer Graphics* 15, 4 (July 2009), 618–629. https://doi.org/10.1109/TVCG.2009.9

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative Visual Manipulation on the Natural Image Manifold. *CoRR* abs/1609.03552 (2016). http://arxiv.org/abs/1609.03552

C. Lawrence Zitnick. 2013. Handwriting Beautification Using Token Means. *ACM Trans. Graph.* 32, 4, Article 53 (July 2013), 8 pages. https://doi.org/10.1145/2461912.2461985