EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

Semantic Segmentation for Line Drawing Vectorization Using Neural Networks

Byungsoo Kim[1], Oliver Wang[2], A. Cengiz Öztireli[1] and Markus Gross[1]

[1]ETH Zürich
[2]Adobe Research

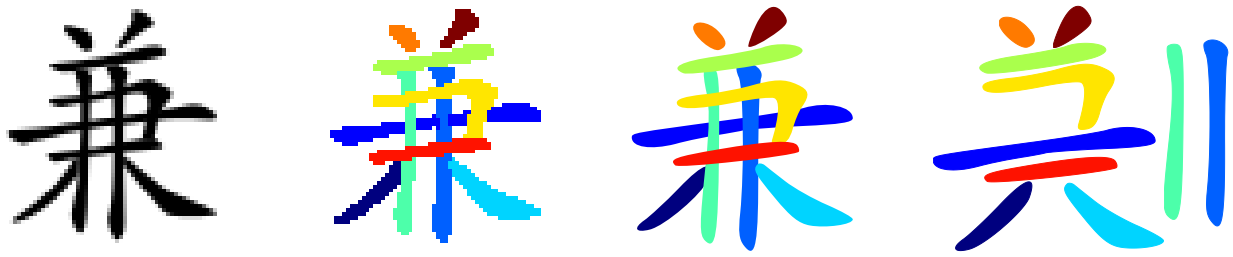


Figure 1: *Our method takes a raster image (left) as input, and produces a semantic vectorization of this image by first segmenting it into a likely configuration of paths given the global context of the image, and vectorizing each path separately. To do this, it leverages a neural network trained to predict inter-pixel path similarities and overlap regions, which allows us to extract the set of vector paths shown on the right.*

Abstract

In this work, we present a method to vectorize raster images of line art. Inverting the rasterization procedure is inherently ill-conditioned, as there exist many possible vector images that could yield the same raster image. However, not all of these vector images are equally useful to the user, especially if performing further edits is desired. We therefore define the problem of computing an instance segmentation of the most likely set of paths that could have created the raster image. Once the segmentation is computed, we use existing vectorization approaches to vectorize each path, and then combine all paths into the final output vector image. To determine which set of paths is most likely, we train a pair of neural networks to provide semantic clues that help resolve ambiguities at intersection and overlap regions. These predictions are made considering the full context of the image, and are then globally combined by solving a Markov Random Field (MRF). We demonstrate the flexibility of our method by generating results on character datasets, a synthetic random line dataset, and a dataset composed of human drawn sketches. For all cases, our system accurately recovers paths that adhere to the semantics of the drawings.

CCS Concepts

•***Computing methodologies*** → ***Image manipulation;*** *Computational photography;*

1. Introduction

Vector images have many benefits over raster images, including resolution independence and support for higher-level editing operations. Nonetheless, most images available "in the wild" are stored in raster format. As a result, vectorizing images has been a long standing, important problem in computer graphics.

However, the process of converting raster images into vector images is challenging, as many possible vector configurations can lead to the same raster image. While state-of-the-art vectorization work is able to reconstruct *visually* coherent vector representations of raster images, they often do not capture the global context of the image. This typically produces what is known as a "vector soup", a collection of small vector paths that make performing any edits on the resulting vectors very difficult. Vectorization ambiguities cannot be solved simply by looking at low level details in the image pixels themselves, instead they require domain-specific knowledge about the content that is being drawn. Consider the simple example of a cross:

The black cross on the left could be composed of four short segments, two corners, two longer segments that cross, or a few other possible configurations. Only by understanding how people tend to draw crosses, can we know that the last option (two longer segments) is probably the right one. One specific challenge is that at intersection points, a single pixel can actually belong to multiple paths, which must be explicitly handled.

In this work, we present a method that attempts to reconstruct a *semantically plausible* vectorization by learning the context of a drawing from a database of vector images. These vector images tell us *how* people create different shapes, which then act as a prior for the rest of our method. Our approach consists of three steps. We first train two neural networks to recognize common patterns and global relationships between paths during vectorization. The first predicts a spatially varying probability map that describes which pixels are likely to belong to the same path, while the second identifies regions where multiple paths overlap. We train these networks on databases of vector images, which provide all the supervision we need, using their corresponding rasterized versions as input. Next, we combine all the local predictions by defining a global energy using an MRF, on an *intersection-augmented* pixel graph, which we solve efficiently using graph cuts. This procedure yields a set of segmented paths which we then vectorize independently using existing tools, and combine into a final vectorized image.

While our key technical approach is applicable to any application domain for which we can acquire a database of examples of the underlying parts that compose the final raster image, we focus this work specifically on line drawings, including characters and sketches. These domains are difficult for existing segmentation and vectorization approaches due to the lack of useful appearance information in the pen strokes.

In summary, our work makes the following contributions: We propose the concept of *semantic* vectorization, where the goal is to reconstruct a set of paths that corresponds to the most likely set of original paths that could be used to create the raster image, yielding a more easily editable vector representation. To do this, we present a novel graph cut energy that globally aggregates local predictions of neural networks to compute a per-path segmentation. Unlike most work on image segmentation, our approach can explicitly handle intersections with multiple overlapping paths by augmenting the pixel grid with overlap regions. Finally, we demonstrate our method on a number of datasets, including Chinese and Japanese fonts, where the network learns the specific stroke order, random synthetic lines, and human drawn sketches of objects. Our method is robust and works with fixed parameters on all examples shown.

2. Related Work

Vectorization techniques. Existing approaches for line drawing vectorization often roughly follow a three-step procedure. First, an input image is simplified by removing needless strokes (or adding meaningful strokes) [BTS05, BCF*07, OK11, LWH15, SSISI16, SISSI17]. Second, the topological structure of the drawing is identified, which includes fitting path segment and finding junction points at which the paths intersect [Che09, NSS*12, JF14]. Finally, the initial topological structure is refined by locally merging or splitting vector primitives [NHS*13], or by means of a global optimization [BLP10]. Commercial vectorization tools include Image Trace in Adobe Illustrator, or Potrace (also used in Inkscape), or the Vectorizer website. As opposed to these approaches, which focus on visual fidelity, our method focuses on extracting semantically meaningful paths learned from data by segmenting strokes prior to vectorization.

Recent vectorization work addresses this as well, phrasing the topological refinement step as a global optimization. This goal attempts to balancing simplicity of the reconstruction with fidelity to the input [FLB16], under the assumption that the simplest possible vectorization is the most likely. As opposed to this, we derive an entirely data-driven set of prior constraints, which are learned by a neural network. This allows us to capture higher-level information about object that is being represented, rather than operating on low level cues.

Segmentation techniques. The crux of our method is an instance segmentation technique for vector paths in rasterized line drawings. Image segmentation is a classic problem in computer graphics and vision, and we refer the reader to a recent survey [YM12] for a full review. One common approach is to define the problem as an energy on a MRF that includes data and smoothness terms, where the data term enforces some appearance model while the smoothness term enforces coherent segments. This can be solved efficiently using graph cuts [BJ01], graph partitioning [SM00], or message-passing with high dimensional filters [KK11]. We phrase our segmentation problem using an MRF, but rather than deriving similarity terms from local pixel values directly, we learn them using a neural network.

CNNs have proven to yield state-of-the-art accuracy in a number of applications, including semantic object segmentation [CPK*14, LSD15, CPK*16]. The success of these methods is driven by large scale, manually annotated databases, such as MS CoCo [LMB*14], or PASCAL VOC [EEVG*15], which consist of collections of photographs with segmented objects, such as people and cars. These approaches often jointly predict a segmentation mask and an objectness score based on some appearance features that are specific to individual classes [PCD15]. Instead, we rely on using *every* query point on the input domain to compute neighborhood similarities based on the context of other similar looking paths.

Recently, a few methods have shown promise in the task of end-to-end trained *instance* segmentation [DHS15, DHL*16, LQD*16]. These approaches consist of local, spatially varying "objectness" estimates, with a simple global aggregation step at the end. While these approaches have shown promising results on photos, our problem is fundamentally different, as unlike prior work, we are considering a challenging case with a single class (path) whose appearance defined largely by its shape and position relative to other paths. This means that the global relationship is crucial when understanding path structure, and our proposed MRF formulation is key to obtaining a consistent final result. In fact, it has been shown

that the forward pass through a CNN can be though of as an advanced type of data term [WFU16], and while these approaches achieve very high quality results due to the learned data terms, the lack of a principled way to introduce global smoothness constraints means that their performance on many optimization tasks is still below that of traditional optimization-based techniques.

An additional motivating factor for our proposed graph-based solution is that we actually operation on an *irregular* domain. We do this by augmenting the problem space, and solving not for a single label at each pixel, but by first identifying overlap regions, and solving for *multiple labels* at these locations by adding additional nodes to our graph. CNN's are not well suited to this kind of modification of the problem domain, which maps to an irregular graph structure, however MRF solutions, in particular using graph-cut optimizations extend trivially to such structures.

Combining global refinement with the output of neural networks has been used for other tasks in segmentation, it has been shown that by integrating a dense CRF approximation [KK11] into a semantic segmentation network, higher fidelity results can be achieved [CPK*14, ZJRP*15], especially around edge regions. Alternately, an interactive approach generates object segments from user clicks which are further refined using a traditional image segmentation, with the neural network prediction serving as the data term [XPC*16]. Instead, our method uses the neural network to predict the *smoothness term* of an atypical MRF computed over an augmented pixel domain.

Non-photo based segmentation. Document segmentation is an important problem, and prior work has shown high accuracy in classifying for example different chart types and inferring the underlying data which enables the format to be quickly changed [SKC*11]. We show results segmenting Chinese characters, which have been investigated in a somewhat different application for recognition purposes, in which the authors propose a MRF-based solution specific to Chinese characters, where the goal is to model the topology of the character for recognition [ZL08]. Instead, we present a general purpose vectorization algorithm that reconstructs a per-pixel segmentation.

There have been prior works that address the semantic segmentation task for sketches, the main challenge being that appearance is sparse. The work of Huang et al. [HFL14] uses a learning based approaches, while Schneider and Tuytelaars [ST16] performs a segmentation of sketch regions by looking at features such as curvature point, T junction and X junctions, and then refines the result with a CRF to compute a segmentation. These approaches generate a high quality sketch segmentation, however their task is fundamentally different from ours, in that we are attempting an instance-segmentation on the stroke level, rather than semantic classification of a stroke or groups of strokes. Our method is designed to prevent the vector soup effect that results from naive vectorization of the entire sketch, or even semantically related parts.

3. Method

The main task in our approach is to compute a semantically meaningful segmentation of paths, so that each path can then be vectorized and combined into a final image.

Our approach combines the strengths of neural networks to understand the context of images, with those of optimization techniques to combine this information in a globally optimal way. We phrase our instance segmentation as a labeling problem where given a set of pixels $\mathcal{P}$ and a finite set of labels $\mathcal{L}$, the goal is to assign a label $l_p \in \mathcal{L}$ to a pixel $p \in \mathcal{P}$ such that a set of assigned labels $l = (l_p)_{p\in\mathcal{P}}$ minimizes an objective energy function $E(l)$.

We define our energy function using a common image segmentation formulation [BFL06, DOIB12]:

$$E(l) = \sum_{p\in P} D_p(l_p) + \lambda_s \sum_{pq\in\mathcal{N}} V_{pq}(l_p, l_q) \tag{1}$$

The goal is to find the optimal label set $\hat{l}$ for segmenting line drawings that minimizes the energy function $E(l)$:

$$\hat{l} = \arg\min_l E(l) \tag{2}$$

In this formulation, $D_p(l_p)$ is the data term, which measures how likely the label assignment l_p is, while $V_{pq}(l_p, l_q)$ is the smoothness term, which measures the penalty of assigning l_p and l_q, $\mathcal{N}$ is the set of all neighbors, and λ_s is a weight that balances the terms.

While this energy appears typical, our application differs significantly from traditional image segmentation. First, we have no priors on label assignments, meaning that $D_p(l_p) = 0$, i.e., each pixel is attributed an equal score independent of its label assignment. Second, the smoothness term $V_{pq}(l_p, l_q)$ is usually driven by image appearance, i.e. it incurs a large cost when two pixels with similar appearance are assigned different labels, and vice versa. In our case, all pixels have similar appearance, so instead we need to learn to predict how likely two pixels are to occur on the same path, which is a function of their *context* in the image. To achieve this, we propose to train a neural network to compute V_{pq} by observing numerous examples of plausible path configurations (Sec. 3.1).

Finally, traditional pixel labeling methods assume that a pixel can have only one unique label. However, in our application, paths can overlap, and in order to correctly vectorize paths that may have parts *under* other paths, we may need to assign multiple labels to a single image pixel. Therefore, we formulate the problem such that the set of pixels $\mathcal{P}$ to be labeled consists of an augmented set with additional pixels added at the intersection regions, which we elaborate in Sec. 3.2. As determining where overlaps exist from a rasterized image is again a complex function of the global configuration of the drawing (overlapping vs. abutting segments are locally ambiguous), we propose a second neural network to predict these intersection regions (Sec. 3.2). We discuss the details of how to construct and minimize the final energy as described Eq. 1 (Sec. 3.3).

3.1. PathNet: Learning to Predict Path Similarity

In order to define the neighborhood similarity term, $V_{pq}(l_p, l_q)$, the first step is to compute a path-similarity measure for each path pixel to all other pixels. We consider a set of pixels $\mathcal{P}'$ corresponding to all path pixels in the drawing (note that $\mathcal{P}'$ does not contain duplicated overlap pixels, unlike $\mathcal{P}$ as we define in the last section). We train a neural network, which we call PathNet that at test time takes

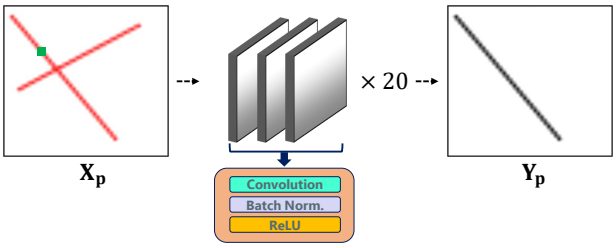
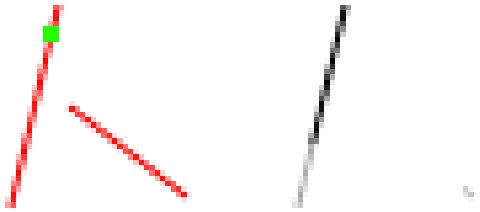


Figure 2: *PathNet Architecture. The input to the network $\mathbf{X}_p$ is shown with the rasterized drawing in red, and the highlighted pixel p drawn as a green square for visualization. The network predicts a path similarity score $\mathbf{Y}_p$ between p and all other pixels.*



(a) (b)

Figure 3: *An example of a prediction becoming less accurate as the distance between p and q increases. Left: $\mathbf{X}_p$, the input image with the query point p (green square). Right: $\mathbf{Y}_p$, the output prediction. Black represents high confidence for being on the same path. You can see that the network erroneously classifies with low confidence, a small region in the bottom part of the right line, and loses confidence at the end of the left line as well.*

a specific $p \in \mathcal{P}'$, and the entire image I as input and produces a matrix $\mathbf{N}_p$ of fixed size $h \times w$, which is the same size of input image, with values in the range of 0 to 1 corresponding to how likely it is that the pixels p and q lie on the same path.

To do this, we use a network architecture adapted from one proposed for image super-resolution [KKLML16]. This network is inspired by residual learning, which performs better in this case than an auto-encoder type network such as [SSISI16]. As shown in Fig. 2, this network consists simply of 20 sequential filter blocks, each with 64 3×3 convolution kernels, followed by a batch normalization layer, and a rectified linear unit layer (ReLU). In our case, the last layer has only a single channel output and the response is capped to be at most 1.

Unlike the previous work [KKLML16], we provide a two-channel input $\mathbf{X}_p$, consisting of the input raster line drawing $\mathbf{I}$, and a second *mask image* that is black everywhere, except at a single indicator pixel p (Fig. 2). The network is trained to output the single channel image $\mathbf{Y}_p$, equivalent to the matrix $\mathbf{Y}_p = \mathbf{N}_p$, which is the path similarity measure from all pixels to the indicator pixel p. We train the network using L2 loss to the known path $\mathbf{Y}$, i.e., the loss is $\|\mathbf{Y} - \mathbf{Y}_p\|_2$. We use the Adam optimizer [KB14] for training.

We can now use feed forward passes through this network to predict $\mathbf{N}_p(q)$ for all p. Note that there is no guarantee that $\mathbf{N}_p(q)$ and $\mathbf{N}_q(p)$ are equal, thus we define N'_{pq}, how likely two pixels are to be on the same path by averaging $\mathbf{N}_p(q)$ and $\mathbf{N}_q(p)$:

$$N'_{pq} = \frac{\mathbf{N}_p(q) + \mathbf{N}_q(p)}{2}, \tag{3}$$

and then we compute a final per-pixel neighbor path similarity score as:

$$K_{pq} = e^{-\frac{(1-N'_{pq})^2}{2\sigma_k^2}}. \tag{4}$$

Using this approach, the more likely two pixels p and q lie on the same path, the closer the K_{pq} is to 1. Intuitively, σ_k controls the falloff of the neural network output which plays an important role in segmentation; with higher σ_k leading to less segments. All weights will be defined in Sec. 4.

We now have all the pieces to define our novel *smoothness term*:

$$V(l_p, l_q) \stackrel{def}{=} \begin{cases} 1 - K_{pq} & if \quad l_p = l_q \\ K_{pq} & otherwise. \end{cases} \tag{5}$$

Intuitively, if two different labels are assigned to a given pair of pixels p and q, *i.e.* $l_p \neq l_q$, the penalty $V(l_p, l_q)$ for mislabeling the two pixels p and q is K_{pq}, the similarity of the two pixels. However, if $l_p = l_q$, then the penalty is *the inverse* of the predicted similarity K_{pq}.

Finally, we observe that the prediction of the neural network becomes less accurate farther from the query point p as shown in Fig. 3, so we add a spatial weighting term onto the prediction which decreases the confidence in the predicted similarity K_{pq} far from the query point.

$$V_{pq}(l_p, l_q) = e^{-\frac{\|p-q\|_2^2}{2\sigma_w^2}} V(l_p, l_q), \tag{6}$$

where σ_w controls the extent of the prediction at p. This also helps with efficiency during inference, as we can reduce the connectivity of the graph by pruning 0-weight edges.

3.2. OverlapNet: Handling Overlapping Paths

As elaborated above, another key difference of our setting from classical image segmentation formulations is the definition of the augmented pixel set $\mathcal{P}$ to handle pixels at the intersections of paths, which belong to more than one segment. Determining such regions is a hard problem that requires context, as intersections can be ambiguous (e.g., abutting vs intersecting lines). This can be seen in the blended image of Fig. 4, (top right), the closed square-shape symbol has three, shallow overlaps, which are unlikely to be simply classified as t-junctions or x-junctions.

We thus train a second network which we call OverlapNet to infer not only the location of overlaps, but also data-dependent overlap shapes as well. Its architecture is identical to PathNet, but the input is only the rasterized line drawing, and the output is a sigmoid layer, thresholded by 0.5 to get a binary overlap map as shown in

Input **X** Ground Truth **Y** Blended Image

Figure 4: *An example of training data pair for OverlapNet.*

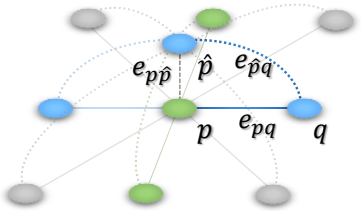


Figure 5: *Overlap augmented graph. Note that the actual graph we use is densely connected, but we show only a reduced set of neighbor edges for clarity. Here the blue pixels are part of one path and the green pixels are part of another path. Pixel p in the center is where the two paths overlap, and $\hat{p}$ is a newly added node.*

Fig. 4. We train this on the same datasets as PathNet, with L2 loss to the known ground truth overlap regions as determined from the vector images.

Once the pixels in overlaps are obtained (Fig. 4), we augment the simple per-pixel graph $\mathcal{P}'$ with new nodes corresponding to these overlapping pixels. Let us assume that there are pixels p and q, and an edge e_{pq} between them. If the pixel p has been determined to be a pixel in an overlap region by the network, we add a new node $\hat{p}$, and connect $\hat{p}$ and q based on the edge e_{pq} with weight K_{pq} (*i.e.* $K_{\hat{p}q} = K_{pq}$). Figure 5 shows an example of a simple augmented graph. In case pixel q is also an overlap pixel, we also add $\hat{q}$ and link $\hat{p}$ and $\hat{q}$ with an edge $e_{\hat{p}\hat{q}}$ with weight $K_{\hat{p}\hat{q}} = K_{pq}$. We also want to ensure that the augmented pixel $\hat{p}$ is labeled *differently* from the original pixel p, so we create high cost for assigning them the same label, e.g.,

$$V(l_p, l_{\hat{p}}) \stackrel{def}{=} \begin{cases} \kappa & if \quad l_p = l_{\hat{p}} \\ 0 & otherwise. \end{cases} \tag{7}$$

in our experiments, we use κ=1000.

While our approach can handle only two paths overlapping on a single pixel, the same idea could be trivially extended to predict the specific *number* of overlaps at each pixel, and the graph could be similarity augmented, however we did not find that this was necessary in our experiments as there were very few regions where more than two paths overlapped at the same region.

3.3. Global aggregation

We solve Eq. 1 using graph cuts [BVZ01]. This problem is highly non-submodular, so we are not guaranteed convergence, however, similar to other work in the area we found that we often achieve good convergence nonetheless. We use the $\alpha\beta$-swap algorithm [BVZ01] to handle multiple labels, which we found performed significantly better than the commonly used alpha-expansion approach for this task. See Sec. 4.1 for more details. Finally, as we found the labeling can occasionally create a small number of isolated single pixels with a unique label, we run a simple post-processing step where isolated pixels are merged with their largest neighboring path.

4. Experiments

We have tested our method on three different datasets: characters, synthetic random lines and hand drawn sketches.

Characters. We have first used two kinds of open-source character datasets which include vector graphics. The first is *Make Me a Hanzi* (https://github.com/skishore/makemeahanzi) provides vector graphics for 9507 of the most common simplified and traditional Chinese characters. Second, we used *KanjiVG* (http://kanjivg.tagaini.net/index.html), which consists of 11456 vector graphics for Japanese Kanji characters in different fonts. An example of one character in both *Make Me a Hanzi* and *KanjiVG* is shown in Fig. 6.

Synthetic random lines. As the second line drawing dataset, we have trained our PathNet and OvelapNet on a dataset consisting of a randomly generated mix of four straight lines and/or cubic Bézier curves with four control points, as shown in Fig. 6.

Sketches. The last dataset we use is the *The Quick, Draw! Dataset* (https://github.com/googlecreativelab/quickdraw-dataset). It includes 50 million sketches of 345 categories drawn by different people with various drawing styles, as shown in Fig. 6. We perform two experiments. We first train and test our method on individual classes (BASEBALL, CAT). Second, we train a single network on a set of classes (BASEBALL, CAT, CHANDELIER and ELEPHANT), which we call the MULTI-CLASS dataset. We then compute the testing error on a set of unseen classes (BACKPACK and BICYCLE). We chose a subset of classes that have challenging, overlapping lines and distinct structure, which makes correct vectorization more challenging.

4.1. Implementation

We have conducted several experiments in order to understand how our networks are trained and how they work with different design parameters and types of data. We implemented our networks using *Tensorflow*, and CairoSVG for rasterizing and converting SVG files. In each dataset, we trained both networks for 50,000 iterations with a batch size of 8 and the image size of 64×64 (16 and 128×128 for sketch dataset) using a PC with Intel Core i7-4790K at 4.00GHz with 4 cores and NVIDIA GeForce GTX 1080. Training lasts approximately 2 hours. All experiments are carried out with fixed training parameters using Adam [KB14], with a gradient clipping value of 0.1, initial learning rate of 0.01, decaying factor of 0.1, 30,000 iterations per decay, and momentum of 0.9999 for training.

For segmentation, we used the GCO libary for multi-label optimization [DOIB12]. In addition, we tested α-expansion, $\alpha\beta$-swap

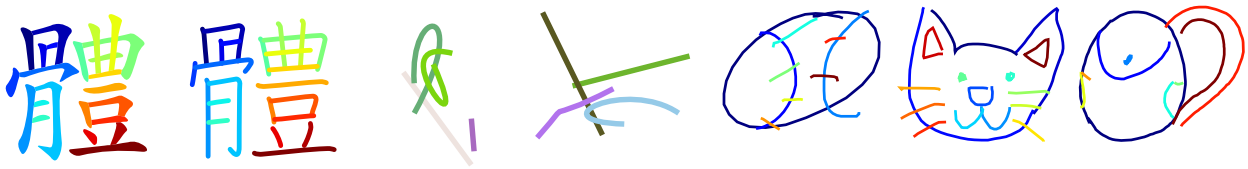


Figure 6: *Examples of images from the different datasets: characters (left), random lines (middle), sketches (right). Individual strokes are color coded.*

Accuracy on Test (Train) Splits						
	CHINESE	KANJI	RANDOM	MULTI-CLASS	BASEBALL	CAT
PathNet	0.947 (0.948)	0.957 (0.959)	0.922 (-)	0.919 (0.920)	0.909 (0.909)	0.934 (0.935)
OverlapNet	0.949 (0.961)	0.944 (0.956)	0.871 (-)	0.607 (0.651)	0.677 (0.676)	0.705 (0.713)

Table 1: *Accuracy of individual networks from training and testing splits. The similar accuracy scores in training and testing indicate good generalization capabilities of our networks*

Testing Accuracy (IoU)						
	CHINESE	KANJI	RANDOM	MULTI-CLASS	BASEBALL	CAT
Potrace	0.577	0.534	0.523	0.570	0.608	0.638
Fidelity vs. Simplicity	0.161	0.284	0.128	0.319	0.350	0.322
Our method	**0.958**	**0.917**	**0.872**	**0.753**	**0.827**	**0.811**

Table 2: *Accuracy for the experiments. Please see the text for the definition of IoU.*

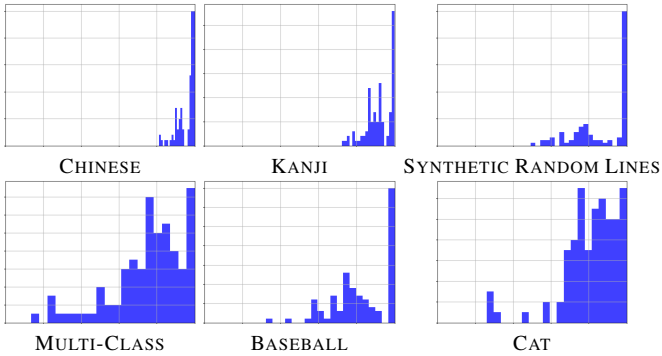


Figure 7: *Histograms of the segmentation accuracy (IoU) for different datasets. We can see that most strokes are well segmented in each dataset, showing rightmost peak.*

and Fusion move algorithms based on Quadratic Pseudo-Boolean Optimization (QPBO) minimization but found that there was no significant difference in terms of accuracy, and much longer execution time. Each pixel label is randomly initialized, the maximum number of labels is fixed as 128, and we find that it is sufficient to use 3 iterations of the $\alpha\beta$-swap algorithm with fixed parameters $\sigma_k = 0.7$ and $\sigma_w = 8$. In the final stage, the per-path vectorization is computed using Potrace [Sel15].

We found our approach to be robust to parameter selection, and all results shown here were computed with the same *fixed* set of parameters.

4.2. Results and Analysis

Label count. We investigated the number of unique labels in the ground truth data compared to our segmented results. Most datasets have similar statistics, from a minimum of 4-5 paths per image to a maximum of 20-25, and an average of 9 paths. The average difference in the label count of our segmented images is roughly 0.7 labels, across all datasets. This is somewhat surprising, and implies that our PathNet is trained well enough to follow the semantics of its dataset. Therefore, while 128 labels is sufficiently higher than the maximum of number of paths in any specific dataset, we use a high number as it does not degrade performance, and choosing a high maximum number of labels gives flexibility to support datasets with more strokes.

Individual network accuracy. We next evaluate the training of the individual networks by splitting our datasets into training and test sets. In each case we use 90% of the data for training, and 10% for testing. Our OverlapNet converges to roughly 0.961 training and 0.949 testing accuracy measured by commonly used Intersection-Over-Union criteria. PathNet converges to a loss of roughly 45, which indicates on average 0.947 prediction accuracy per pixel, where prediction accuracy is computed as 1 - L2 loss to the ground truth prediction, normalized by the size of prediction window. For example, the prediction accuracy of Fig. 3 is 0.968. The accuracy of both networks is shown over different datasets in Table 1.

Final segmentation accuracy. Finally, we evaluate our full pipeline, including segmentation. For a quantitative evaluation, we use IoU as the measure of Overlap accuracy. For each segmented pixel path, we find the closest path in the ground truth (*i.e.* the intersection area between them is the largest among all paths), and compute the ratio of the number of pixels in the intersection divided by the number in the union. We show quantitative results with this metric in Table 2 and Fig. 7. The two character datasets follow strict stroke rules, and we see our method scores much better as a result. On sketch datasets, our IoU scores drop, but this is expected given the inherent variance in how different people sketch common objects. We note that this quantitative evaluation is one way to gauge quality, but it is not the final goal as judging a "good" segmentation is somewhat subjective. Instead, we desire a *plausible* vectorization which is more editable than a pure vector soup. For this, we present a subset of results for qualitative evaluation

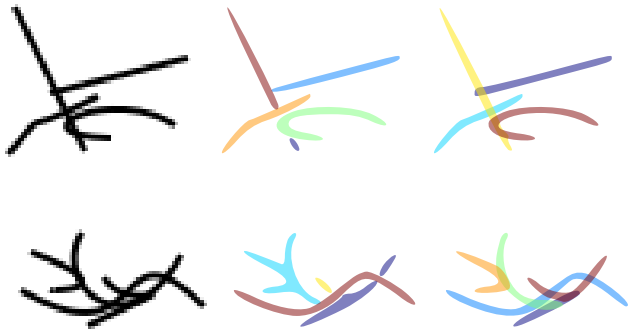


Figure 8: *The effect of the OverlapNet. We can see examples where a curve brushes against a straight line or abutts it.*

in Fig 9 and Fig. 12, and in the supplementary material, where we observe that these results are often vectorized in a meaningful and useful way.

One question is how well do networks trained on one dataset work on a different dataset. To test this, we trained using the BASEBALL, CAT, CHANDELIER and ELEPHANT classes. The testing accuracy on the multi-class dataset is 0.753, and on the unseen classes, BACKPACK and BICYCLE, 0.729 and 0.716 respectively. In addition, PathNet trained on CHINESE shows 0.836 on KANJI compared to 0.917. This is encouraging as neither of these classes were seeing during training, so we observe reasonable generalization to different domains. However, as expected, training and testing on the same class shows a slight improvement over cross-class testing.

Secondly, by using OverlapNet to handle overlaps between strokes, we see an improvement in accuracy by 2-5% approximately, and much qualitatively better results in Fig 8.

We compare our method to the state-of-the-art method of Favreau et al. [FLB16], as well as a commercial technique Potrace [Sel15]. While Favreau et al. can take drawings with varying stroke width as input, it skeletonizes the strokes and outputs constant-width vector curves. In Fig. 9, we show an example correct segmentations and vectorizations generated with our method, compared to other approaches. In all comparisons, we used the software provided by the authors. We searched across all stroke thicknesses, and chose the one that performs best with their method. In Fig. 10, we show that our method can be combined with a sketch clean-up method such as [SSISI16] as a preprocessing step to handle rough line drawings. In general, we can see the importance of semantic information in the vectorization process, in that our strokes better correspond to individual instances.

Editing. Finally, we show some edged vector images, where a novice user was easily able to edit the output of our approach, moving around parts as desired (Fig. 11). In a traditional vector soup result, the user would have to pay more attention to the context of the paths to maintain consistency across intersections and t-junction ambiguities. With our result, these edits were made in just one

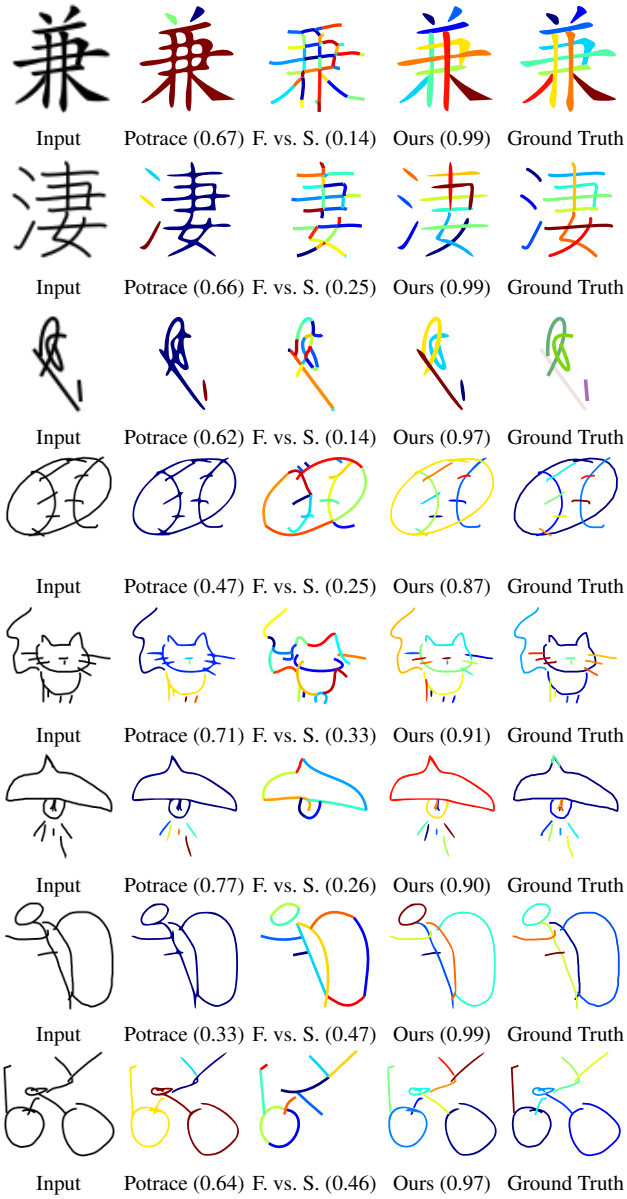
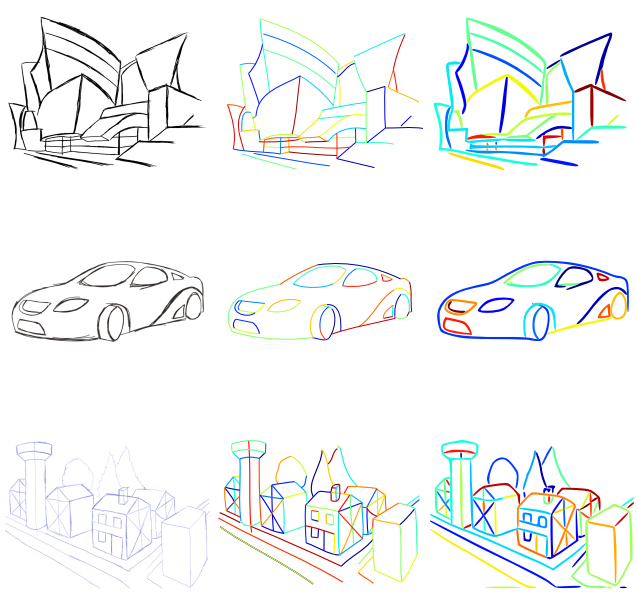


Figure 9: *Vectorization results showing the input image, the result from Potrace [Sel15](2nd), the result from a recent global optimization technique [FLB16], the result from our method, and the ground truth. Without being able to learn the stroke rules and drawing conventions, previous approaches miss, split, or merge strokes incorrectly.*

or two minutes, using *Vector Paint* (http://vectorpaint.yaks.co.nz/).

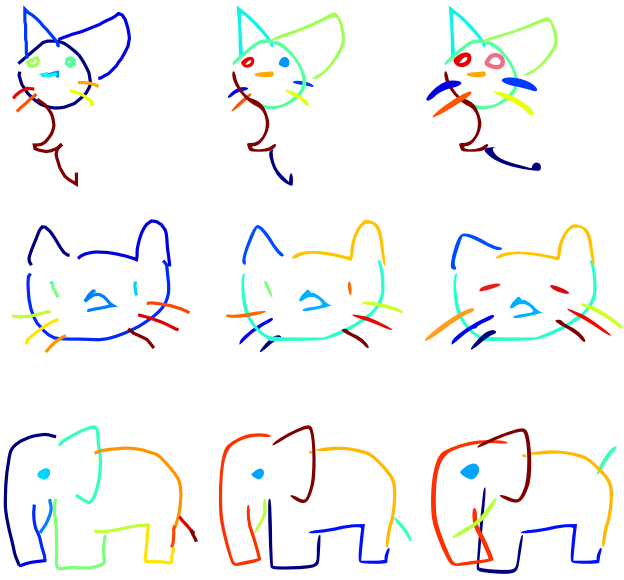
Timing. The runtime depends on the number of non-zero pixels in an image and the number of iterations used for the graph-cut. As seen in Table 3, the bottleneck is the final CRF in our (unoptimized) implementation.

Limitations. While our solution produces good results a lot of



Input F. vs. S. Ours

Figure 10: *From left to right: vectorization results showing the rough input image (courtesy of [FLB16]), the color coded result from [FLB16] and the result first cleaned up by [SSISI16] then processed by our method.*



Ground Truth Strokes Our Result Edited Image

Figure 11: *User-editing on individual strokes of our result.*

Timing Information (sec.)						
	CHINESE	KANJI	RANDOM	MULTI-CLASS	BASEBALL	CAT
PathNet	1.51	1.21	0.88	9.59	10.3	9.81
OverlapNet	0.01	0.01	0.01	0.07	0.04	0.03
Optimization	11.3	6.76	7.57	399	415	223

Table 3: *Here we show the runtime of different parts of our method.*

the time, there are still times when it fails. Fig. 13 shows an incorrect segmentation where the vertical stroke is split into 3 separate paths. As a purely data-driven approach, its accuracy significantly depends on training dataset, and to achieve semantically plausible results, it is important to select a training dataset that fits the application domain. It would be interesting to train our approach in other domains, such as handwriting, or architectural drawings, if such datasets were available. In particular, what is defined as a "path" in the original dataset can vary, for example in the ground truth data from the sketches, it tends to be in places where people are likely to lift the pen, which might not be the ideal vectorization target.

Secondly, as we currently support only overlaps between two paths, occasional cases where more than two paths overlap can cause problems as shown in Fig. 13. As we mentioned in Sec. 3.2, our idea could be extended to handle more than two overlapping paths. Also, the complexity of our method depends on the number of pixels, so it is still a challenge to scale the approach up to high resolution images. In addition, we have found that our method works best when trained in a similar domain that it is applied to. Hopefully with larger datasets, this restriction will be reduced in the future. Despite these limitations, our method is able to generate more useful vectorization than prior segmentation-based techniques.

5. Conclusions

We have presented a method for context-aware vectorization of rasterized graphics. This addresses the fundamental challenge of extracting semantically meaningful paths from line art drawings. As this is a highly domain specific problem, we utilized data to learn the non-trivial rules people apply to generate paths in such drawings, in contrast to previous model-based approaches in the literature.

Future Work. Although we have focused on line drawings in the scope of this work, our method is a general pixel-level segmentation approach trained with vector art files. In addition to brush-stroke style paths, we believe that a similar approach could be used to support the full range of SVG path objects, such as patches, and color fills.

Introducing context to segmentation when appearance information is not relevent can be useful for many other problems. An immediate application could be parsing more complex objects such as PDF files. We are planning to investigate how our technique can be utilized for such challenging cases.

References

[BCF*07] BARTOLO A., CAMILLERI K. P., FABRI S. G., BORG J. C., FARRUGIA P. J.: Scribbles to vectors: preparation of scribble drawings

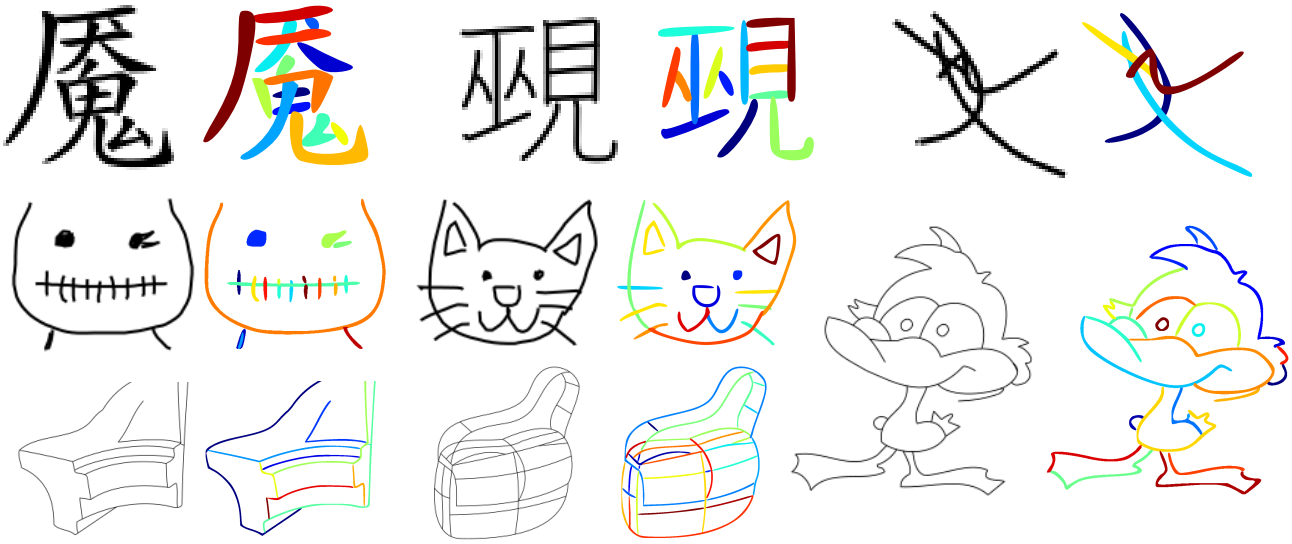


Figure 12: *A collection of results generated by our technique. Left to right, top to bottom:* CHINESE, KANJI, SYNTHETIC RANDOM LINES, STITCHES, CAT *and Line drawings courtesy of [FLB16].*

Figure 13: *Incorrect vectorization examples. The overlaps are missed on the figure on the left, which means that the vertical stroke is split into several small ones. On the right, the center of the pizza is incorrectly vectorized because many paths cross in the middle.*

for cad interpretation. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling* (2007), ACM, pp. 123–130. 2

[BFL06] BOYKOV Y., FUNKA-LEA G.: Graph cuts and efficient nd image segmentation. *International journal of computer vision 70*, 2 (2006), 109–131. 3

[BJ01] BOYKOV Y. Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001), vol. 1, IEEE, pp. 105–112. 2

[BLP10] BARAN I., LEHTINEN J., POPOVIĆ J.: Sketching clothoid splines using shortest paths. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 655–664. 2

[BTS05] BARLA P., THOLLOT J., SILLION F. X.: Geometric clustering for line drawing simplification. In *ACM SIGGRAPH 2005 Sketches* (2005), ACM, p. 96. 2

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (Nov. 2001), 1222–1239. URL: http://dx.doi.org/10.1109/34.969114, doi:10.1109/34.969114. 5

[Che09] CHENG M.-M.: Curve structure extraction for cartoon images. In *Proceedings of The 5th Joint Conference on Harmonious Human Machine Environment* (2009), pp. 13–25. 2

[CPK*14] CHEN L.-C., PAPANDREOU G., KOKKINOS I., MURPHY K., YUILLE A. L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062* (2014). 2, 3

[CPK*16] CHEN L.-C., PAPANDREOU G., KOKKINOS I., MURPHY K., YUILLE A. L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915* (2016). 2

[DHL*16] DAI J., HE K., LI Y., REN S., SUN J.: Instance-sensitive fully convolutional networks. *arXiv preprint arXiv:1603.08678* (2016). 2

[DHS15] DAI J., HE K., SUN J.: Instance-aware semantic segmentation via multi-task network cascades. *arXiv preprint arXiv:1512.04412* (2015). 2

[DOIB12] DELONG A., OSOKIN A., ISACK H. N., BOYKOV Y.: Fast approximate energy minimization with label costs. *International journal of computer vision 96*, 1 (2012), 1–27. 3, 6

[EEVG*15] EVERINGHAM M., ESLAMI S. M. A., VAN GOOL L., WILLIAMS C. K. I., WINN J., ZISSERMAN A.: The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision 111*, 1 (Jan. 2015), 98–136. 2

[FLB16] FAVREAU J.-D., LAFARGE F., BOUSSEAU A.: Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics* (2016). 2, 7, 8, 9

[HFL14] HUANG Z., FU H., LAU R. W.: Data-driven segmentation and labeling of freehand sketches. *ACM Transactions on Graphics (TOG) 33*, 6 (2014), 175. 3

[JF14] JAYARAMAN P. K., FU C.-W.: Interactive line drawing recognition and vectorization with commodity camera. In *Proceedings of the 22nd ACM international conference on Multimedia* (2014), ACM, pp. 447–456. 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). URL: http://arxiv.org/abs/1412.6980. 4, 5

[KK11] KRÄHENBÜHL P., KOLTUN V.: Efficient inference in fully connected crfs with gaussian edge potentials. NIPS. 2, 3

[KKLML16] KIM J., KWON LEE J., MU LEE K.: Accurate image super-resolution using very deep convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). 4

[LMB*14] LIN T.-Y., MAIRE M., BELONGIE S., HAYS J., PERONA P., RAMANAN D., DOLLÁR P., ZITNICK C. L.: Microsoft coco: Common objects in context. In *European Conference on Computer Vision* (2014), Springer, pp. 740–755. 2

[LQD*16] LI Y., QI H., DAI J., JI X., WEI Y.: Fully convolutional instance-aware semantic segmentation. *arXiv preprint arXiv:1611.07709* (2016). 2

[LSD15] LONG J., SHELHAMER E., DARRELL T.: Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3431–3440. 2

[LWH15] LIU X., WONG T.-T., HENG P.-A.: Closure-aware sketch simplification. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 168. 2

[NHS*13] NORIS G., HORNUNG A., SUMNER R. W., SIMMONS M., GROSS M.: Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics (TOG) 32*, 1 (2013), 4. 2

[NSS*12] NORIS G., SỲKORA D., SHAMIR A., COROS S., WHITED B., SIMMONS M., HORNUNG A., GROSS M., SUMNER R.: Smart scribbles for sketch segmentation. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 2516–2527. 2

[OK11] ORBAY G., KARA L. B.: Beautification of design sketches using trainable stroke clustering and curve fitting. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (2011), 694–708. 2

[PCD15] PINHEIRO P. O., COLLOBERT R., DOLLAR P.: Learning to segment object candidates. In *Advances in Neural Information Processing Systems* (2015), pp. 1990–1998. 2

[Sel15] SELINGER P.: Portrace, 2015. URL: http://potrace.sourceforge.net/. 6, 7

[SISSI17] SASAKI K., IIZUKA S., SIMO-SERRA E., ISHIKAWA H.: Joint Gap Detection and Inpainting of Line Drawings. 2

[SKC*11] SAVVA M., KONG N., CHHAJTA A., FEI-FEI L., AGRAWALA M., HEER J.: Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (2011), ACM, pp. 393–402. 3

[SM00] SHI J., MALIK J.: Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence 22*, 8 (2000), 888–905. 2

[SSISI16] SIMO-SERRA E., IIZUKA S., SASAKI K., ISHIKAWA H.: Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 121. 2, 4, 7, 8

[ST16] SCHNEIDER R. G., TUYTELAARS T.: Example-based sketch segmentation and labeling using crfs. *ACM Transactions on Graphics (TOG) 35*, 5 (2016), 151. 3

[WFU16] WANG S., FIDLER S., URTASUN R.: Proximal deep structured models. In *Advances In Neural Information Processing Systems* (2016), pp. 865–873. 3

[XPC*16] XU N., PRICE B., COHEN S., YANG J., HUANG T.: Deep interactive object selection. *arXiv preprint arXiv:1603.04042* (2016). 3

[YM12] YI F., MOON I.: Image segmentation: A survey of graph-cut methods. In *Systems and Informatics (ICSAI), 2012 International Conference on* (2012), IEEE, pp. 1936–1941. 2

[ZJRP*15] ZHENG S., JAYASUMANA S., ROMERA-PAREDES B., VINEET V., SU Z., DU D., HUANG C., TORR P. H.: Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1529–1537. 3

[ZL08] ZENG J., LIU Z.-Q.: Markov random field-based statistical character structure modeling for handwritten chinese character recognition. *IEEE transactions on pattern analysis and machine intelligence 30*, 5 (2008), 767–780. 3