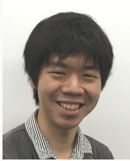
Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup

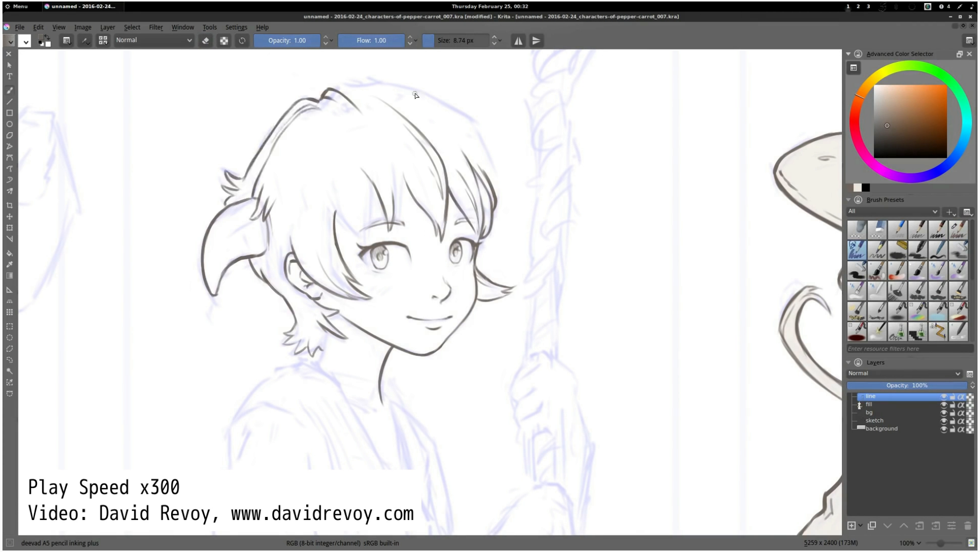
Edgar Simo-Serra*, Satoshi Iizuka*, Kazuma Sasaki, Hiroshi Ishikawa (*equal contribution)

July 27th, 2016

Waseda University



Sketch Simplification



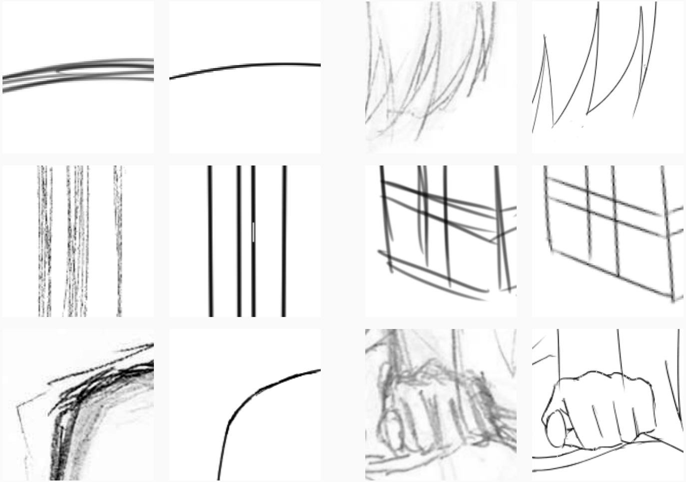
Sketch Simplification

Input: Rough Sketch

Output: Line Art

Sketch Simplification

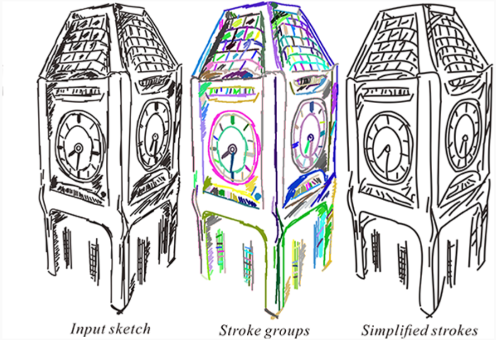
Rough Target Rough Target



Related Work

Related Work

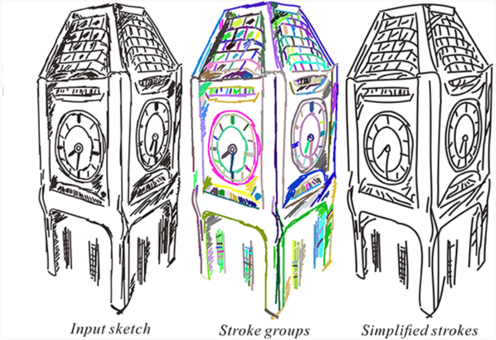
1. Sketch Simplification
   1.1 Progressive Online Modification
   1.2 Stroke Reduction
   1.3 Stroke Grouping



Liu et al. 2015

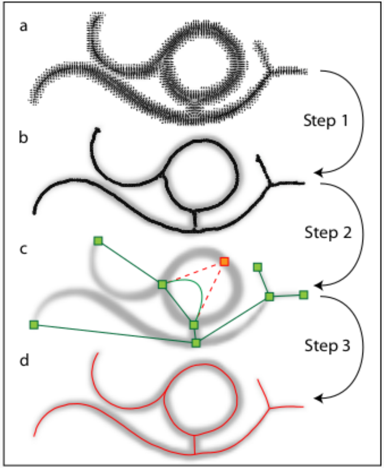
Related Work

1. Sketch Simplification
   1.1 Progressive Online Modification
   1.2 Stroke Reduction
   1.3 Stroke Grouping
   1.4 Vector input



Liu et al. 2015

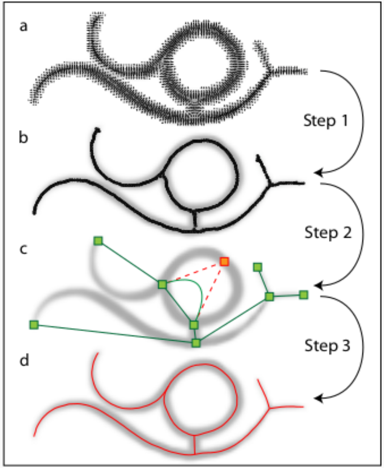
1. Sketch Simplification
   1.1 Progressive Online Modification
   1.2 Stroke Reduction
   1.3 Stroke Grouping
   1.4 Vector input
2. Vectorization
   2.1 Model Fitting (Bezier, ...)
   2.2 Gradient-based approaches



Noris et al. 2013

Related Work

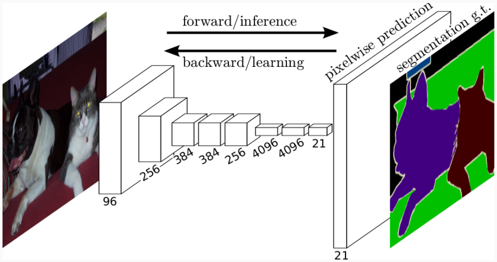
1. Sketch Simplification
   1.1 Progressive Online Modification
   1.2 Stroke Reduction
   1.3 Stroke Grouping
   1.4 Vector input
2. Vectorization
   2.1 Model Fitting (Bezier, ...)
   2.2 Gradient-based approaches
   2.3 Require fairly clean input sketches



Noris et al. 2013

Related Work

1. Sketch Simplification
    1.1 Progressive Online Modification
    1.2 Stroke Reduction
    1.3 Stroke Grouping
    1.4 Vector input
2. Vectorization
    2.1 Model Fitting (Bezier, ...)
    2.2 Gradient-based approaches
    2.3 Require fairly clean input sketches
3. Deep Learning
    3.1 Fully Convolutional Network

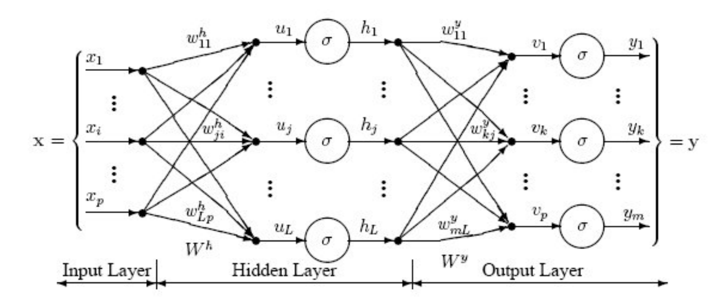


Long et al. 2015

Proposed Approach

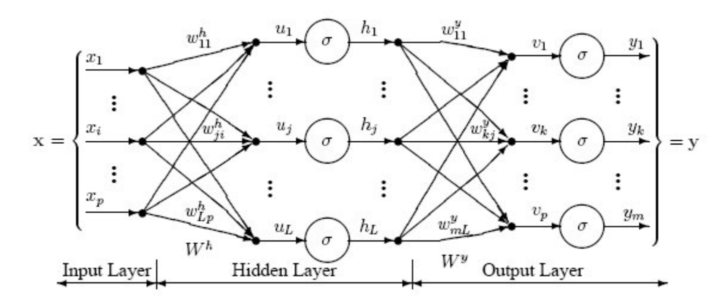
Deep Learning

- Modern Neural Networks
  - Computational efficiency with GPU
  - Large scale datasets



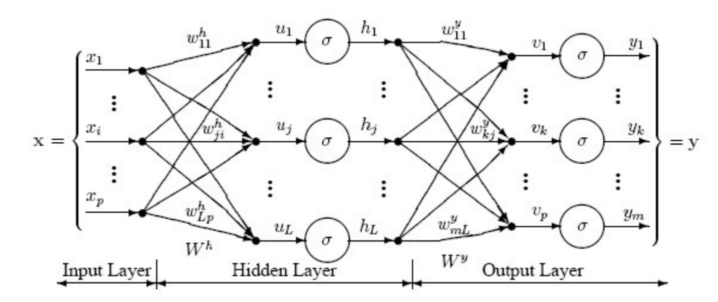
Deep Learning

- Modern Neural Networks
  - Computational efficiency with GPU
  - Large scale datasets
- Learns input to output mapping
- Basic building block layer: $f(x) = \sigma(Wx)$



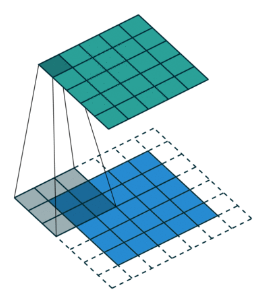
Deep Learning

- Modern Neural Networks
  - Computational efficiency with GPU
  - Large scale datasets
- Learns input to output mapping
- Basic building block layer: $f(x) = \sigma(Wx)$
- Parameters $(W \ldots)$ are learnt
- Hyper-parameters are set by hand



Fully Convolutional Network

- Uses only convolutional layers
- Each layer convolves many filters
- Layer hyperparameters: kernel, padding, and stride
  - Weights expressed with kernels
  - Padding conserves the image size
  - Stride can change the output resolution



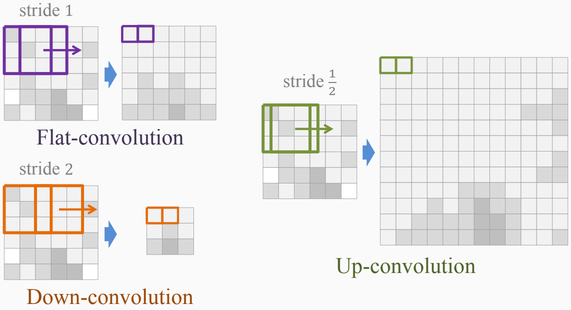
Output

Input

Fully Convolutional Network

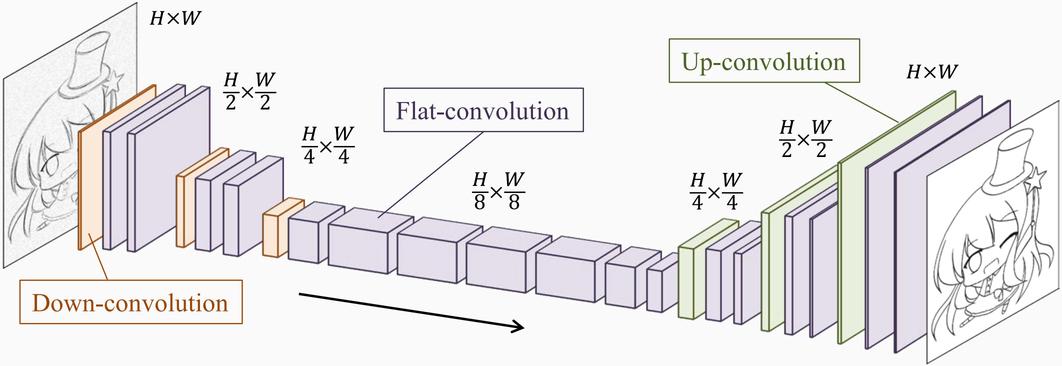
We create three building blocks by modifying the stride:

1. Flat-convolution
    1.1 3×3px kernel, 1×1px padding, 1px stride
2. Down-convolution
    2.1 3×3px kernel, 1×1px padding, 2px stride
3. Up-convolution
    3.1 4×4px kernel, 1×1px padding, $^1/_2$px stride



Model

- 23 convolutional layers
- Output has the same resolution as the input
- Encoder-Decoder architecture
    - Reduces memory usage
    - Increases spatial resolution

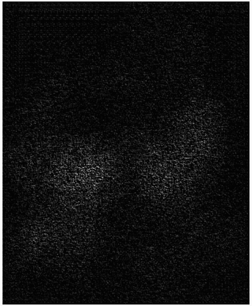


Learning

- Trained from scratch
- Using 424×424px patches
- Weighted Mean Square Error loss
- Batch Normalization [Ioffe and Szegedy 2015] is critical
- Optimized with ADADELTA [Zeiler 2012]

Input

Output

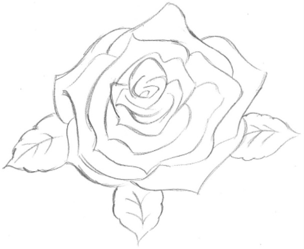


Target

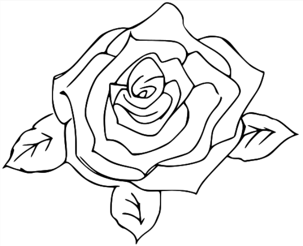
Vectorization and Simplification

- Vectorization with potrace
  - Open source software
  - High pass filter and binarization

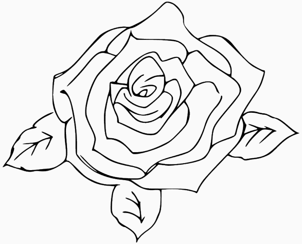
Input



Output



Vector



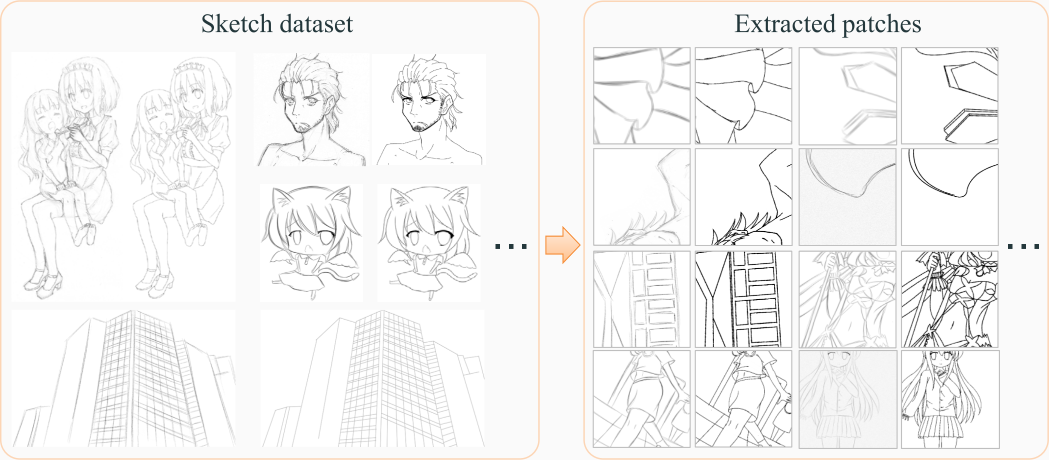
Vectorization and Simplification

- Vectorization with potrace
  - Open source software
  - High pass filter and binarization
- Scaling input changes simplification degree

Sketch Dataset

Sketch dataset

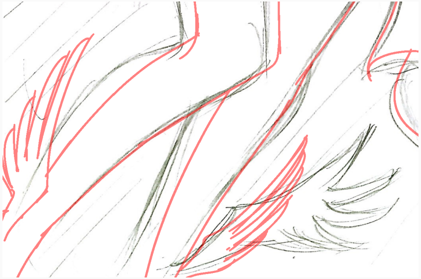
- 68 pairs of rough and target sketches
- 5 illustrators



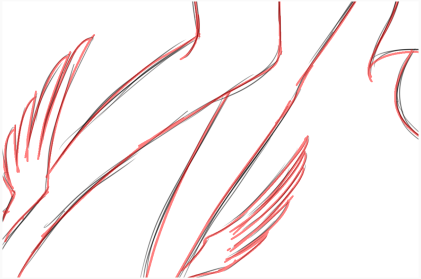
Inverse Dataset Creation

- Data quality is critical
- Creating target sketches from rough sketches has misalignments
- Creating rough sketches from target sketches properly aligns

Standard

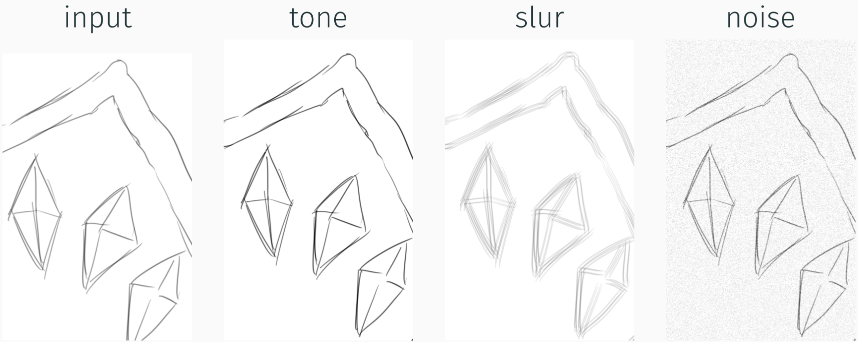


Inverse Creation



Data Augmentation

- 68 pairs is insufficient
- Scaling training data
- Random cropping, flipping and rotation
- Additional augmentation: tone, slur, and noise



Results and Comparisons

Results

- Intel Core i7-5960X CPU (3.00GHz)
- NVIDIA GeForce TITAN X GPU
- 3 weeks training time

Image Size	Pixels	CPU (s)	GPU (s)	Speedup
320×320	102,400	2.014	0.047	$42.9\times$
640×640	409,600	7.533	0.159	$47.4\times$
1024×1024	1,048,576	19.463	0.397	$49.0\times$

Comparison

Input

Potrace

Adobe Live Trace

Ours

Comparison

Input

Potrace

Adobe Live Trace

Ours

User Study

- Comparison with 15 images
- 19 users participated (10 with illustration experience)
- Absolute rating (1 to 5 scale)
- Relative evaluation (best of two)

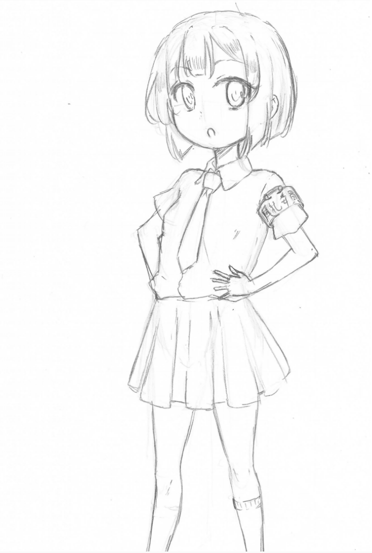
	Ours	Live Trace	Potrace
Score	4.53	2.94	2.80
vs Ours	-	2.5%	2.8%
vs Live Trace	97.5%	-	30.3%
vs Potrace	97.2%	69.7%	-

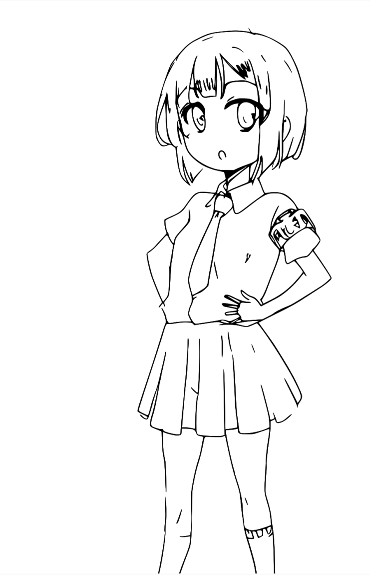
Comparison

Results

Results

Results





Results

Conclusions

- Automatic Sketch Simplification Approach
- Convolutional networks are suited to image processing
- Proper data is crucial for training

Conclusions

- Automatic Sketch Simplification Approach
- Convolutional networks are suited to image processing
- Proper data is crucial for training

Try Online: http://hi.cs.waseda.ac.jp:8081/

Model

type	kernel size	stride	output size
input	-	-	$1 \times H \times W$
down-convolution	5×5	2×2	$48 \times H/2 \times W/2$
flat-convolution	3×3	1×1	$128 \times H/2 \times W/2$
flat-convolution	3×3	1×1	$128 \times H/2 \times W/2$
down-convolution	3×3	2×2	$256 \times H/4 \times W/4$
flat-convolution	3×3	1×1	$256 \times H/4 \times W/4$
flat-convolution	3×3	1×1	$256 \times H/4 \times W/4$
down-convolution	3×3	2×2	$256 \times H/8 \times W/8$
flat-convolution	3×3	1×1	$512 \times H/8 \times W/8$
flat-convolution	3×3	1×1	$1024 \times H/8 \times W/8$
flat-convolution	3×3	1×1	$1024 \times H/8 \times W/8$
flat-convolution	3×3	1×1	$1024 \times H/8 \times W/8$
flat-convolution	3×3	1×1	$1024 \times H/8 \times W/8$
flat-convolution	3×3	1×1	$512 \times H/8 \times W/8$
flat-convolution	3×3	1×1	$256 \times H/8 \times W/8$
up-convolution	4×4	$1/2 \times 1/2$	$256 \times H/4 \times W/4$
flat-convolution	3×3	1×1	$256 \times H/4 \times W/4$
flat-convolution	3×3	1×1	$128 \times H/4 \times W/4$
up-convolution	4×4	$1/2 \times 1/2$	$128 \times H/2 \times W/2$
flat-convolution	3×3	1×1	$128 \times H/2 \times W/2$
flat-convolution	3×3	1×1	$48 \times H/2 \times W/2$
up-convolution	4×4	$1/2 \times 1/2$	$48 \times H \times W$
flat-convolution	3×3	1×1	$24 \times H \times W$
flat-convolution	3×3	1×1	$1 \times H \times W$