

Combining Sketch and Tone for Pencil Drawing Production

Stefan Burnicki, Raphael Braun, Andreas Altergott

August 26, 2014

Table of Contents

1 Introduction

- Table of Contents
- Project Objectives
- Implementation Steps

2 Implementation

- Line Drawing
- Tone Drawing
- Merge of Line and Tone Drawing
- Kernel Pipeline and Benchmarks

3 Live Demonstration

4 Conclusion

Project Objectives

- GPU implementation of the paper Combining Sketch and Tone for Pencil Drawing Production
- GPU implementation must be faster than the CPU implementation
- Understanding of a larger CUDA implementation

Implementation Steps

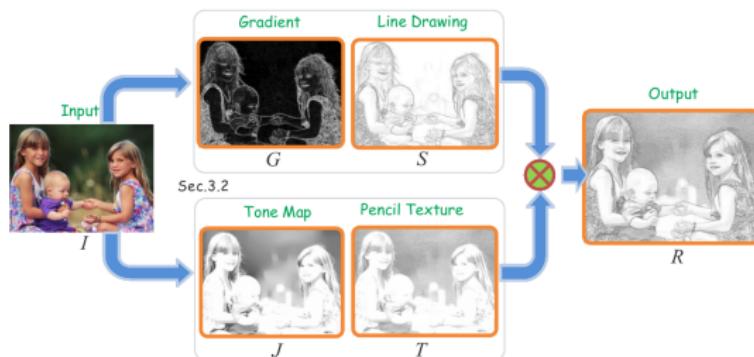


Figure : Overview of the pencil drawing framework

- Line Drawing
- Tone Drawing
- Merge of Line and Tone Drawing

Line Drawing



- Forward gradient calculation on the grayscale version of the image with $G = \left((\partial_x I)^2 + (\partial_y I)^2 \right)^{\frac{1}{2}}$

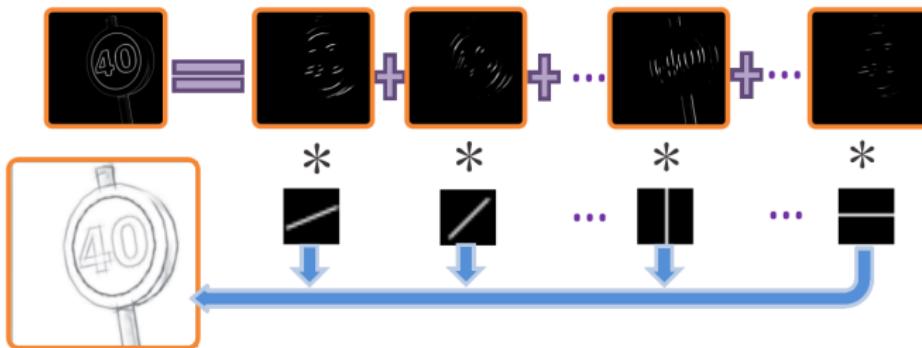
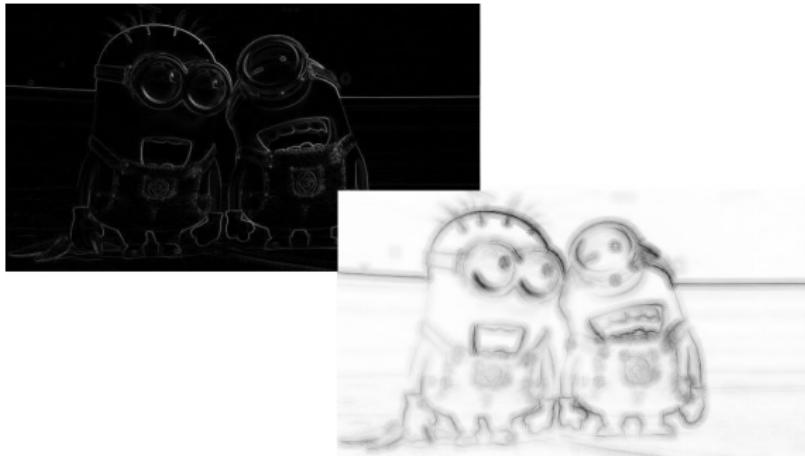


Figure : Line drawing with strokes

- Computation of the response map with $G_i = L_i * G$, where L_i is a line segment at direction i (implemented in the convolution kernel) and $*$ the convolution operator
- Selection of the response for the pixel p with the maximum value by $C_i(p) = \begin{cases} G(p) & \text{if } \operatorname{argmin}_i \{G_i(p)\} = i \\ 0 & \text{otherwise} \end{cases}$



- Line Generation at each pixel with $S' = \sum_{i=1}^8 (L_i \otimes C_i)$
- Final stroke map S obtained by inverting pixels of S' and mapping them to $[0, 1]$

Tone Drawing

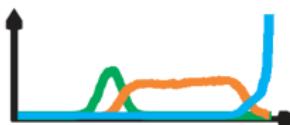


Figure : Tone distribution p_1, p_2, p_3

- Usage of constant parameters $\omega_1, \omega_2, \omega_3, \sigma_b, u_a, u_b, \mu_d, \sigma_d$
- Computation of distribution per tonal layer with
$$p(v) = \frac{1}{Z} \sum_{i=1}^3 \omega_i p_i(v)$$
- Computation of target tone distribution p_1, p_2, p_3
- Pencil Texture rendering

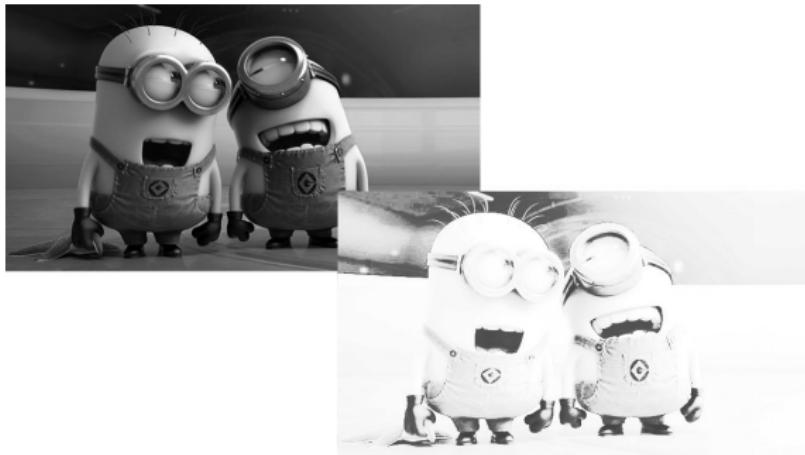
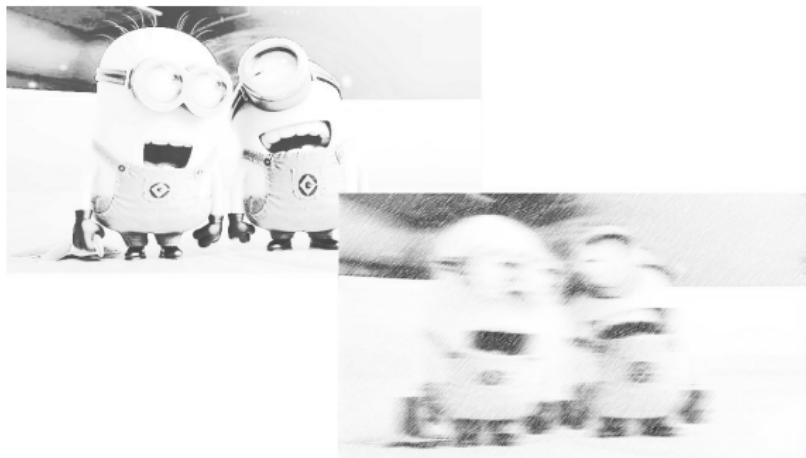
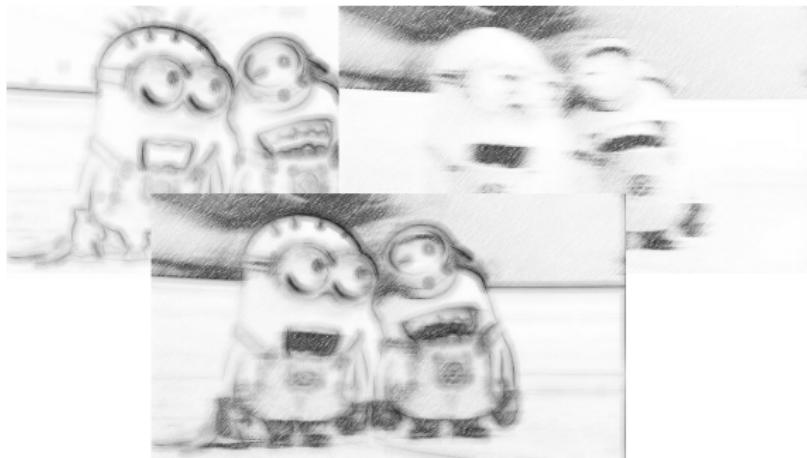


Figure : Result of tone map

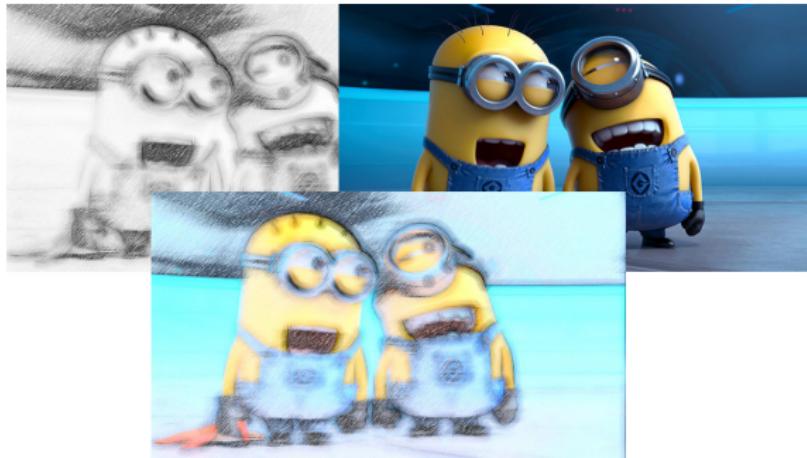


■ Pencil Texture rendering

Merge of Line and Tone Drawing

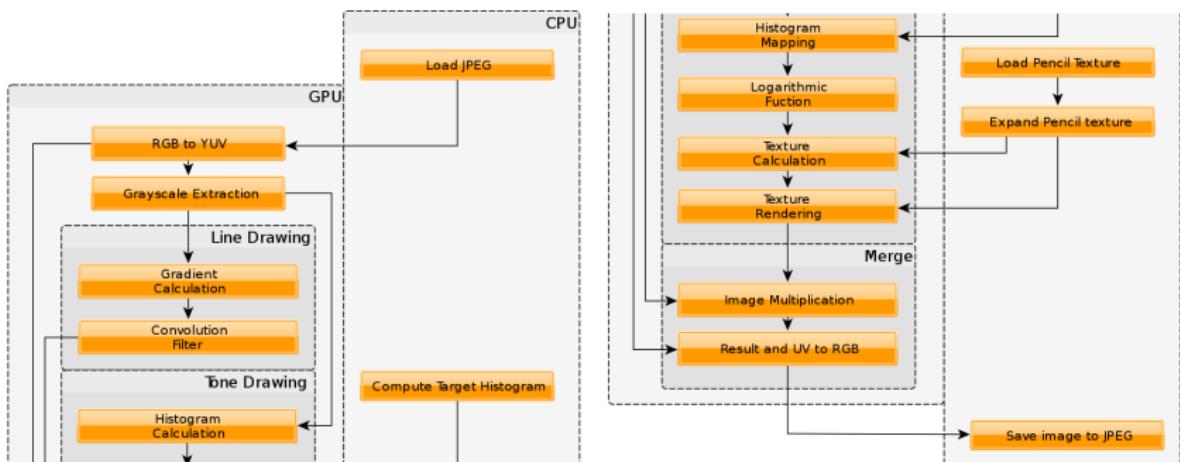


- Multiplication of S and T values per pixel in $R = S \cdot T$



- Simply use generated output R as Y in YUV colour space, keeping U and V from original image for coloured output

Kernel Pipeline



Introduction

Implementation

Live Demonstration

Conclusion

Line Drawing

Tone Drawing

Merge of Line and Tone Drawing

Kernel Pipeline and Benchmarks

Benchmarks

Live Demonstration













Conclusion

- Successful GPU implementation of the sketch filter as described in the paper
- Our GPU implementation is faster than the version of the paper
- 30 ms per frame in DVD quality
- 60 ms per frame in HD Ready
- 130 ms per frame in Full HD quality

Mastered Obstacles

- Understanding of the paper
- Newcomers to image processing
- Realization of a bigger GPU implementation

Future Improvements

- Better optimization can improve performance even more
- Pre image analyser could automatically calculate the ideal line length for the line drawing
- Pre image analyser could also calculate the parameters for the tone drawing

Thank you for your attention.

Questions?