Testing, Tuning, and Applications of Fast Physics-based Fog Removal

William Seale & Monica Thompson

CS 534 Final Project

Fall 2012

1 Abstract

Physics-based fog removal is the method by which a standard atmospheric model of haze is used to try to estimate fog density in a given region of an image. The model first estimates the skylight level of the image by performing a region-based intensity search of the image. Atmospheric veil is determined by the intensity maximum of a given pixel's normalized RGB values and a haze map is created based on the varying levels of this intensity to estimate the varying degrees of fog. The haze map is then inverted and applied to the image to remove the 'fog' and keep the surface albedo image behind the haze. The technique outlined in "Physics-based Fast Single Image Fog Removal" used standard models and applies approximations of traditional transforms to produce a fast estimate of these fog and albedo maps. In this paper we reproduce Yu, et al. techniques, and further tune their technique and reformulate the underlying mathematical estimation model to allow applications beyond simple fog removal.

2 Introduction

There are many algorithms available that remove hazing effects. However, there are very few that attempt to enhance atmospheric effects, despite many fogged images having a high aesthetic quality. In this paper we implement Yu, et al. algorithm for removing haze and then successfully reverse the algorithm to create a natural looking image with enhanced atmospherics. In addition we used these same algorithms in experimental ways such as colored fog introduction and hazing unfogged images.

3 Implementation of Defogging

We first started by searching for an algorithm to implement. We settled on a paper called "Physics-based Fast Single Image Fog Removal" by Jing Yu, Chuangbai Xiao, and Dapeng Li. The algorithm proposed by Yu, et al. is conceptually simple, relatively easy to implement, and fast. The algorithm, like many other defogging algorithms, follows this basic formula for modeling hazy images.

$$I(x) = A\rho(x)e^{-\beta d(x)} + A\left(1 - e^{-\beta d(x)}\right)$$

$I(x)$ represents a pixel x within the image I. A indicates the estimated skylight of the image. $\rho(x)$ represents scene albedo. $d(x)$ represents depth in a scene. Lastly, β is a constant that denotes extinction coefficient of the atmosphere. The product of $A\rho(x)e^{-\beta d(x)}$ represents direct attenuation and $(1 - e^{-\beta d(x)})$ represents the atmospheric veil, or haze map. To simplify this model further, we will use $t(x) = e^{-\beta d(x)}$

$$I(x) = A\rho(x)t(x) + A(1 - t(x))$$

3.1 Skylight Calculation

One problem that a few defogging algorithms tend to have is that they search for the pixel with the greatest RGB intensity and use that for the skylight. However, this intensity value isn't guaranteed to actually be part of the sky and the aforementioned algorithms can be thrown off by bright lights or objects that appear white within the image. Yu, et al. attempt to solve this problem by defining a sky region and picking the pixel with the greatest intensity within that region. The algorithm they use was implemented in our code as follows:

1. Apply a min filter to reduce any noise that may occur within an image to produce I_{min}. The min filter that we use for this step was provided by Frederico D'Almeida.
2. Run Canny edge detection on I_{min}.
3. Calculate a percentage map N_{edge}. The percentage map is used to help determine where the sky is in an image. For an edge pixel x within the Canny image, calculate the percentage of edge pixels within x's neighborhood.
4. Determine which pixels are candidates for the sky region. Yu, et al. use two thresholds to determine this.

   $$N_{edge}(x) < T_v \qquad\qquad I_{\min}(x) > T_p$$

   Where T_v is a flatness threshold set to 0.001 and T_p is a brightness threshold set to 95% of $I_{\min}(x)$ maximum brightness value.

5. Search from top to bottom from the first connected pixel to the next to determine sky region. We modified this step of the algorithm to stop after 500 pixels to prevent the algorithm from running too long.

We found that while this method does a fairly good job of defining sky regions, however, if a bright light is contained within the sky region the algorithm still tends to choose it to be the value of A over the actual sky.

3.2 White Balance and Haze Map Calculation

From this point the model is simplified further by Yu, et al.

$$V(x) = (1 - t(x))$$

$$I(x) = A\rho(x)t(x) + AV(x)$$

With the skylight calculated, we were able to determine the haze map $V(x)$.

$$\frac{I(x)}{A} = \rho(x)t(x) + V(x)$$

To ensure proper white balance, intensity was restricted between 0 and 1.

$$I'(x) = \min\{\frac{I(x)}{A}, 1\}$$

$$I'(x) = \rho(x)t(x) + V(x)$$

We then calculated a rough estimation of the haze map by performing a min operation on each individual channel in the RGB colorspace. [1]

$$\tilde{V}(x) = minRGB\ I'(x)$$

To further refine $\tilde{V}(x)$ a bilateral filter was applied. For this function, a fast bilateral filter written by Jiawen Chen based on the research of Paris and Durand was used.

Lastly, scene albedo is calculated as follows

$$\rho(x) = \frac{I'(x)\ - \alpha V(x)}{t(x)}$$

We know $t(x)$ because $V(x) = 1 - t(x)$ so it follows that $t(x) = 1 - V(x)$. α was introduced in order to ensure that we do not divide by 0. Ideally, α would be calculated based on the sky region (as the sky region grows, α shrinks) is but for our purposes we set it to a static value of 0.95 [1].

Overall, we found the algorithm provided by Yu, et al. to be satisfactory for our purposes, however, we noticed that the hazed images that we defogged would often come back with oversaturated colors. We researched this problem further and, after reviewing a comparison of algorithms provided on Jean-Philippe Tarel's personal website [2], we found that a number of algorithms seemed to suffer from this same problem. We wanted to rectify this issue before proceeding with the fog enhancement and addition.

4 Methodology and Results

4.1 Saturation Matting

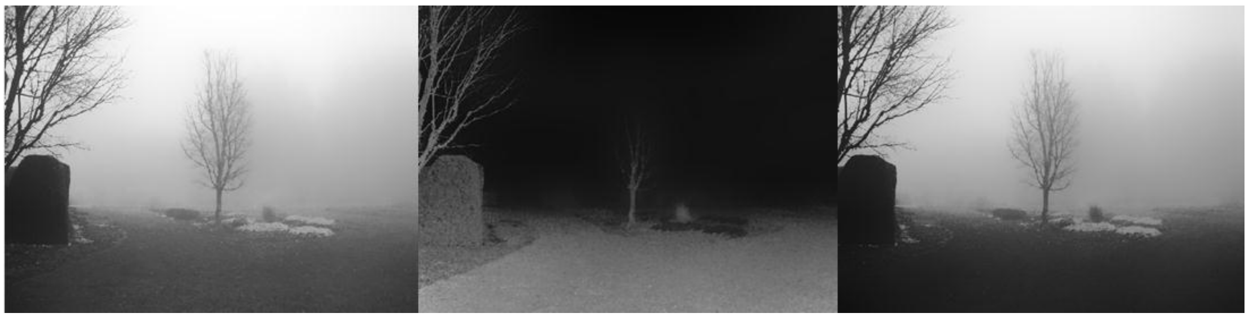

Figure 1: Original Haze Map, Saturation Matte, Resulting Haze Map

4.1.1 Techniques

As an initial attempt, we decided to take a straightforward path and matte the haze map with the Saturation channel from an HSV-converted version of the source image. This was done after the haze map was generated using the minRGB(I/A):

$$V(x) = V(x) * (1 - Sat(x))$$

This approximated the results that we were looking for, in that the saturation values from the original image were greatly preserved, but it also brought a complication in that we discovered the S channel was noisy on compressed images.

The next iteration was to attempt pre-filtering of the S channel to try to smooth out rough details, with an eye towards maintaining the linear time of the original technique. Erosion/dilation, separable box blur, and Gaussian blur were tested, but none of them yield the expected results, often affecting nearby brightness levels incorrectly.

In our final iteration we ended up coming to a similar conclusion as Yu, et al., in that a bilateral filter was the best to remove noise in a fast manner without leading to image blur and bleeding.

Figure 2a -2c: Saturation Adjustment Crops. *Box Filter, Min Filter, Bilateral Filter*

4.1.2 Results

This final version produces fast, naturally-saturated, defogged images. In the examples in **Figure 2a** the grass is subtly more nuanced green/brown, and in **Figure 2b** the pink/orange tree is particularly maintained in a much more subdued palette which seems more appropriate by human examination of the source image. Beyond this further attempts at using this saturation matte in a different manner (notably inverse matting the transmission map or injecting it after processing) did not yield superior results, so we feel this is the most appropriate usage.

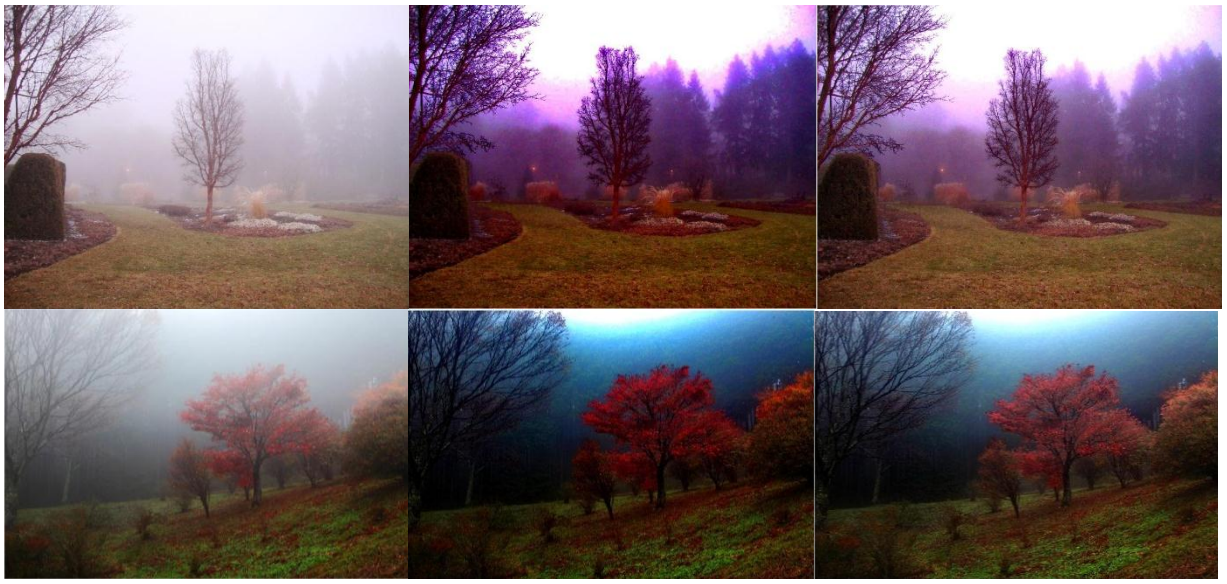
Figure 3a and 3b: Saturation Modification Results. *Left-to-Right: Original Image; Yu, Et A; Modified Version*

4.1.3 Areas of Improvement

While this technique successfully introduces more pleasing color to saturated areas, it illustrated the idea that not only was haze map affecting color but also over-pumping contrast. And while both of these were handled for saturated regions by tempering the haze map, this can leave areas of over-contrasted regions when saturation is available to overtly cue the need for matting.

For example the road in **Figure 4** is pulled to black by both the original and modified implementations, wherein a more appropriate level would be a mid-dark grey.

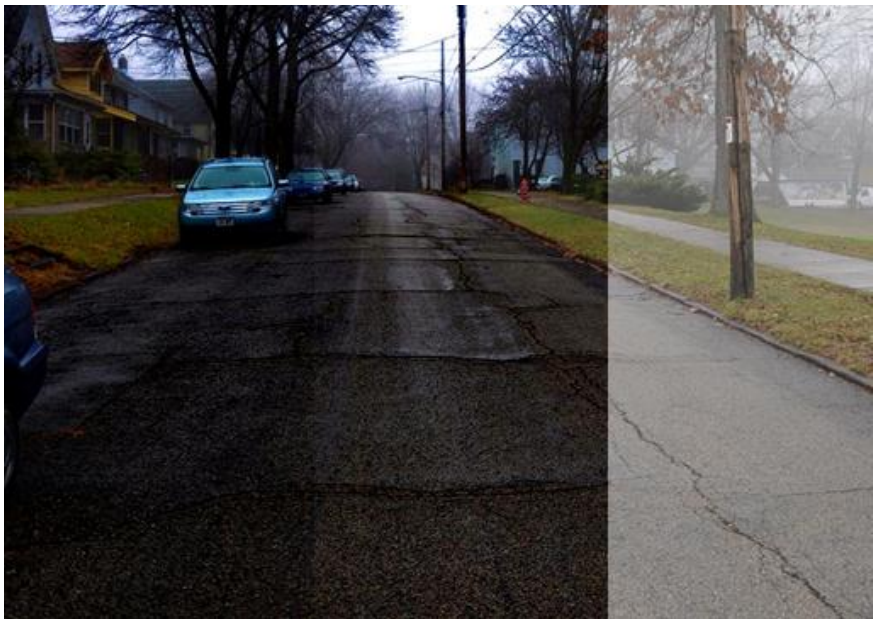
Figure 4: Saturation Modification Results. *Left-to-Right: Yu, Et A; Modified Version; Original*

4.2 Fast Re-fogging

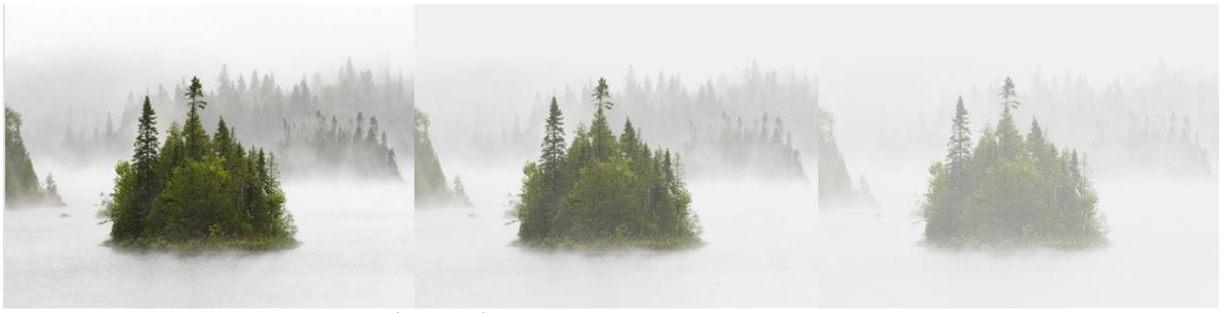
Figure 5: Re-fogging. *Left-to-Right: Original Image; Fog Doubling; Fog Quadrupling*

4.2.1 Techniques

As most of the symbols in the final equation were based on each other, we had to determine which should be considered variables and which should be solved for. In the end we fixed the haze map as an isolated item (even though in the original derivation it was based on $I'(x)$) and then used $t(x) = 1 - V(x)$ for the transmission estimate.

This led to the following reformulation, with I as the re-fogged result.

$$I(x) = A * (\rho(x) - \rho(x)V(x) - \alpha V(x))$$

A – Skylight Intensity (Double)
$\rho(x)$ – Albedo Image (RGB)
$V(x)$ – “Veil” estimate (aka Fog Map, BW)

4.2.2 Results

This final version produces very natural-looking re-fogged images. Objects deeper within the scene fade smoothly into a nonlinear grey without totally disappearing, while close areas are subtlety more fogged without a profound loss of definition.

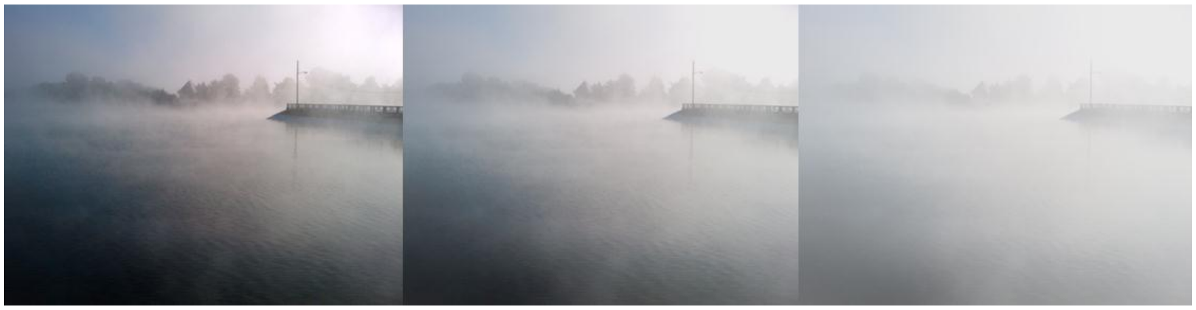
Figure 6: Re-fogging Results. *Left-to-Right: Original Image; Fog Doubling; Fog Quadrupling*

4.2.3 Areas of Improvement

While this technique works well for adding deeper fog to existing image, the fact remains that this has a narrow window of usefulness. Adding fog to existing images in a clean, mostly-automated fashion would be a big win. As can be seen in the Further Experiments section we tried extending this technique to non-fogged images with minimal success, but the fact remains that this is a useful area of future investigation.

5 Further Experiments

5.1 Fogging Non-fogged Images

Attempts to make a re-fogging algorithm that introduces fog into non-fogged images were not initially very successful. The primary technique we attempted to use was to take an arbitrary gradient, centered on a user-specified point with a specified gradient falloff rate, and then run a bilateral filter on the gradient with the edge map instead taken from the albedo image. This produced the results that can be seen in **Figure 7**, which aren't very realistic, and would fail more dramatically on more complex images. However, this could be a usable fake with more user specified parameters.

Additionally with computer vision object recognition, techniques like this method could potentially be improved.

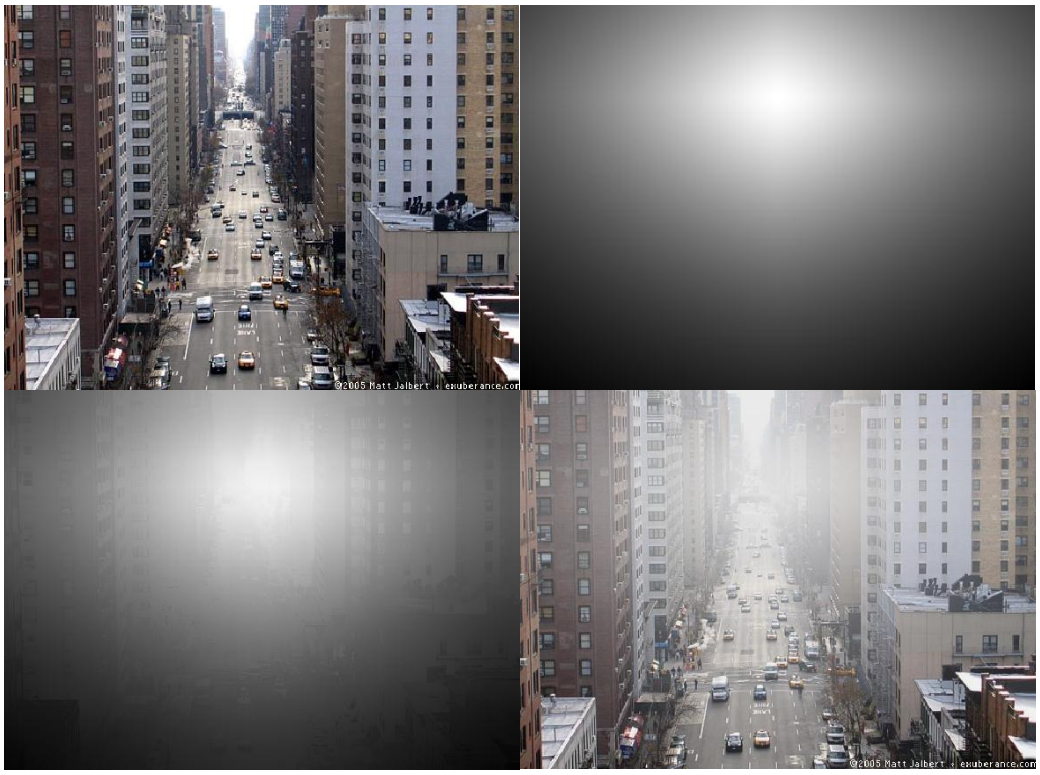
©2005 Matt Jalbert exuberance.com
©2005 Matt Jalbert exuberance.com

Figure 7: Autofogging. *Top Row: Original image; User-centered gradient*
Bottom Row: Gradient bilaterally filtered; Fogging result

5.2 Colored Fog Addition

Additionally as a further experiment we attempted to re-introduce colored fog by bringing a BW fog map into RGB grey space and then tweaking the balance of colors. As can be seen in **Figure 9,** the results had an interesting aesthetic that maintained much of the original image's definition but no deep, meaningful value beyond that. In addition, if the original albedo removal introduced hard edges then this technique would not make them more subtle.

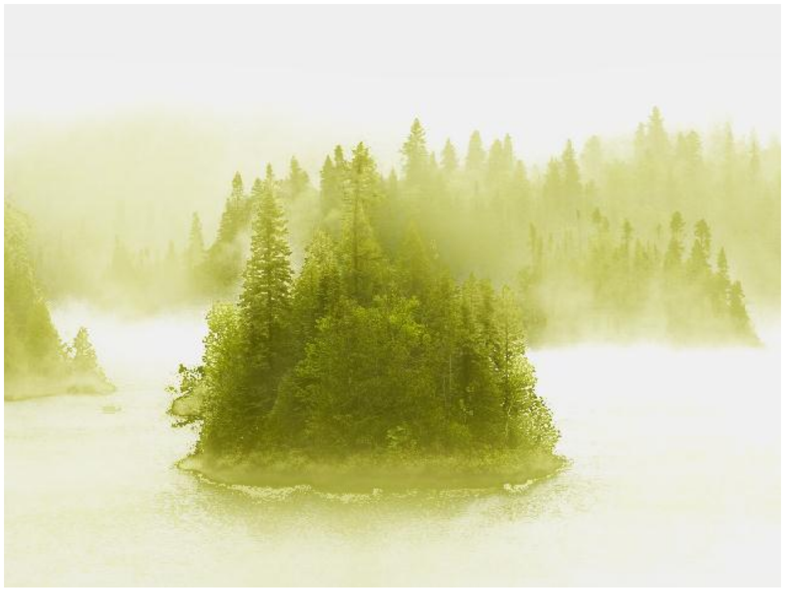

Figure 9: Colored Re-Fogging.

5.3 Other Experiments

We also experimented with two other items with minimal success. In the code we have a flag for colored fog removal, which takes the skylight color in addition to the intensity and applies this to a RGB haze map. In the end the technique was not able to stably remove color without introducing further unwanted tinting.

Additionally, we experimented with removing underwater haze from images, but many of the assumptions of the original technique were unsuitable. Notably the 'skylight' search had to be broadened to find much darker 'skylights', and even then it still found front highlights. We also tried other modifications to invert the formula to deal with the fade-to-black quality, but in the end the characteristics of underwater brightness loss/color shift made most of the results were useless. Eventually, we ended up specifying the 'skylight' manually and then manually inverting the matte, which yielded a modicum of water-haze removal, so the results were not worth including at length.

References

[1] J. Yu, C. Xiao, and D. Li, "Physics-based Fast Single Image Fog Removal," in *Proceedings of the tenth ICSP*, 2010, pp. 1048-1052.

[2] J. Tarel, "Single Image Visibility Restoration Comparison," 2012, http://perso.lcpc.fr/tarel.jean-philippe/visibility/.

Image Sources

All images other than Figure 4 were sourced from the internet and used under fair use provisions. Figure 4 was not in the public space but is granted Creative Commons Attribution by photographer William Seale.

Project Team Work

Defogging/Re-fogging Code (~300 lines, MATLAB)

Code by Seale/Thompson (core logic based on Yu, Et Al)

Details: Initial rough journal implementation framework by Thompson, with additional code and testing/reworking (Skylight recursion, integration of Bilateral) by Seale. Extensions and experiments written primarily by Seale, with conceptual assistance and testing/tweaking by Thompson

Project Report

Written by Seale/Thompson

Details: Project report tasks were evenly divided between Seale and Thompson. Each wrote approximately half and reviewed the combined document as a whole.

External Code Sources

Fast Bilinear Filter, MATLAB

Code by Jiawen Chen (slightly modified by Seale, logical based on Paris and Durand)

http://people.csail.mit.edu/jiawen/software/bilateralFilter.m

Fast 2D Max/Min Filter, MATLAB

Code by Frederico D'Almeida

http://www.mathworks.com/matlabcentral/fileexchange/1358-fast-2d-maxmin-filter