# EECS 442 Computer Vision: Final Project Progress Report

Nathan Immerman
College of Engineering, University of Michigan
Ann Arbor, Michigan
immerman@umich.edu

Alexander Chocron
College of Engineering, University of Michigan
Ann Arbor, Michigan
achocron@umich.edu

## 1. Introduction

Our project is to implement the algorithm outlined by Johnson and Farid's paper "Exposing Digital Forgeries by Detecting Inconsistencies in Lighting" [1]. This algorithm is able to detect artificially composited images by determining whether the components are consistently lit. We will present this application as a MATLAB Graphical User Interface application that will take user input, decide whether the image is a forgery, and display the computed light directions on the objects specified by the user.

## 2. Work Accomplished

We have written the code to collect occluding boundaries of objects in the image, partition these boundaries into patches, find the normal vectors of these patches, and extrapolate the intensity of the occluding pixels at these patches. We have also begun to create a GUI that will stitch all of the pieces together.

In accordance with the method described in the paper, we determined the best way to collect the occluding boundaries of objects is through user input. As of now, the GUI displays the image in question and allows a user to draw the various relevant boundaries on the image itself (Figure 1).

Once the boundaries are collected, they are partitioned into about eight patches. This partitioning function has been implemented. To estimate the normal vectors for each patch, we are going need to collect three user entered points for each patch (not implemented yet) and then fit a quadratic curve to estimate the contour of the patch which can then be used to estimate the normal vector (implemented). We can then use the normals to fit a power function to the pixels along the normal direction on the object to estimate the intensities of the occluding pixels (implemented). At a certain point in the algorithm, we need to calculate a matrix $C$ as a function of the number of partitions and the size of each partition (implemented). At this point we have coded but not tested the mechanisms to collect almost all of the information that we need to be able to run the algorithm described in the paper.

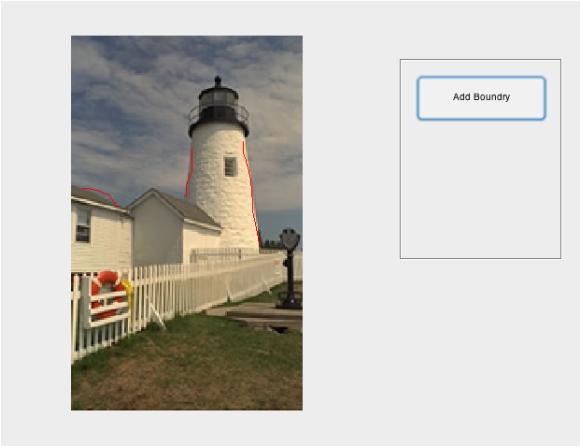


Figure 1. Current State of our GUI, user selected boundaries in red

## 3. Work to be Done

We have accomplished a significant portion of our algorithm, but we still have many tasks to complete. All of our implemented code is untested as of yet. This is because testing requires more code to be completed - we must implement another function that takes all of the outputs from our existing functions and runs the algorithm in completion. We are unable to test our existing functions without the proper input, and this means that we must complete the other input portions of our GUI. We must then test all of our code, and fix any bugs we find. The code we have already written is for detecting inconsistencies in lighting assuming an infinite light source, but we also must implement code that detects inconsistencies assuming a local light source. Though some of our functions will be applicable to this problem, we will also need to write additional code. This includes implementing or finding an implementation of conjugate gradient descent. This is because the local light source least squares error minimization has no closed form solution and must be solved iteratively. We will also need to update our GUI to accommodate this setting.

Lastly, we will need to test everything. We will create test cases by taking pictures both outdoors and indoors with

known light sources, and testing how well the algorithm is able to determine the light directions. We will also create composite images in different lighting and determine how well the algorithm is able to distinguish between the two (or more) inconsistent directions.

We would like to point out that while originally we had planned to create an iOS application to demonstrate the algorithm, we have since decided to make a MATLAB GUI instead. We were at first unsure whether MATLAB was capable of providing a robust interface for accepting various types of input that the algorithm requires. Further, making an iOS application would require extensive learning about iOS development whereas both of us are already competent in MATLAB. It would also require converting all of our MATLAB code to c++, which would take time that we would rather spend improving our algorithms and adding features to our GUI.

## 4. References

[1] M. K. Johnson and H. Farid. Exposing digital forgeries by detecting inconsistencies in lighting. *In Proceedings of the 7th workshop on Multimedia and security*, pages 1-10, 2005.

# EECS 442 Computer Vision: Final Project Proposal

Nathan Immerman
College of Engineering, University of Michigan
Ann Arbor, Michigan
immerman@umich.edu

Alexander Chocron
College of Engineering, University of Michigan
Ann Arbor, Michigan
achocron@umich.edu

## 1. Introduction

With the advent of software tools such as Photoshop and Gimp, it is becoming increasingly simple to doctor and create fake images. One method of doctoring images is to create a composite image out of existing source images. Since composite images are difficult to detect for the common person, we hope to create a tool that enables users to determine composite images.

## 2. Approach

It is often difficult to make the light direction throughout the entirety of composite images consistent. This fact can be leveraged to detect whether even well-stitched images are fake. By analyzing the light direction of different surfaces within the image, one can detect whether or not the two surfaces came from the same original image. An algorithm for detecting such inconsistencies has been outlined in Johnson and Farid's paper, *Exposing digital forgeries by detecting inconsistencies in lighting*. We shall attempt our own implementation of this algorithm and measure our implementation based on speed and accuracy.

We will attempt to implement the algorithm outlined in [1] as closely as possible. In addition, we may explore other techniques that are mentioned in the references of [1] and in other papers that we find while researching the topic, and perhaps synthesizing a method of our own.

We are going to work together to implement the algorithm and the encompassing iOS deliverable. We will then work separately to optimize different subparts of the algorithm.

## 3. Expected Outcome

We plan to package the algorithm as an iOS app demonstration. This app will be built using Apple's xCode software in Objective-C, C++, and OpenCV. Alexander has familiarity with iOS development and Nathan will be learning from scratch.

We will evaluate the success of our implementation by measuring the accuracy of fake detection on a test bench of fake and non-fake images. We will also generate our own synthetic images with a know light source direction and compare this ground truth direction with our implementation's output.

We foresee calculating the estimated normal vectors in the image as risky. We believe that is possible but if it turns out to be too much of a challenge to complete within our deadline, we will use images with known normals. Another issue we foresee is the practical efficiency of the algorithm when running on an iOS device. To overcome this, we will optimize our implementation as much as possible. Even with optimizations, if the implementation is not efficient enough on an iOS device we will run the implementation on a desktop computer which will demonstrate the primary functionality of the implementation.

## 4. References

[1] M. K. Johnson and H. Farid. Exposing digital forgeries by detecting inconsistencies in lighting. *In Proceedings of the 7th workshop on Multimedia and security*, pages 1-10, 2005.