I. Technique Overview

It is quite difficult to exactly match the lighting effects within a photograph due to directional lighting, such as the sun. If there are differences within the directional lighting, the inconsistencies would suggest tampering. There are four underlying assumptions when attempting to estimate the light source direction. Firstly, the surface of the object under observation should be Lambertian, meaning that the surface reflects light isotropically. The surface is smooth and thus, light is reflected in all directions. The second assumption is that the surface has constant reflectance value. This is to make sure that one area of the object would not display an intensity that is not expected of the area as the technique does not account for that variability. Thirdly, the surface of the object is illuminated by a point light source is infinitely far away. This is so that a vector in the direction of the light source from the surface is parallel to all other directional vectors from the surface. If the light source is local, these vectors would not be parallel and constant. Finally, the angle between the surface normal and the light direction must be between -90° and 90°. With these assumptions, the image intensity is:

$$\mathbf{I}(x, y)=\mathbf{R}(\mathbf{N}^{\mathrm{T}}(x, y) \cdot \mathbf{L})+\mathbf{A}$$

R is the constant reflectance value, which will be set to 1 in our program because it can be considered to have unit-value, acknowledging the fact that the estimation of **L** will be within an unknown scale factor. $\mathbf{N}(x, y)$ is a 3-vector representing the surface normal at the point (x, y). The z-component of this vector is to 0 in this program under the assumption of orthographic projection. **L** is a 3-vector that points in the direction of the light source, and the z-component is also set to 0 here. **A** is a constant ambient light term. The dot product of **N** and **L** contributes to the fact the amount of light that strikes a surface is proportional to the angle between the surface normal and the light direction.

Knowledge of 2-D surface normals from at least three distinct points on a surface with constant reflectance is required to solve for the **L** and **A** terms using least-squares estimation, starting with the following quadratic error function:

$$\mathbf{E(L,A)}=||M\mathbf{v}-\mathbf{b}||^2$$

The *M* refers to an nx3 matrix with the first column denoting the x-component and the second column denoting the y-component of the surface normal **N** at (x_i, y_i). Each value in the third column is 1. The term **b** denotes an nx1 matrix of image intensities at (x_i, y_i). The unknown, **v**, is a 3x1 matrix containing the x- and y- component of the directional light vector and the ambient light term, **A**. We get the estimate of **v** by differentiating the quadratic error function with respect to **v** and setting the result to zero and solving for **v** to achieve the least-squares estimate. By solving **v**, the directional light vector components and the ambient light term are determined.

II. Program Overview

The program P2 will attempt to estimate the x- and y- components of the direction to a point light source that is infinitely far away. The four assumptions mentioned in the technique overview are kept in the program. In addition, there is the assumption that the object in the image has a circular contour. The user can selected any number of points along this contour and based on the user selected points, a circle will be fitted to the contour. The program then extracts estimates of surface normals from the user-specified portion of the circle after retrieving a sampled set of points spanning the portion of the circle specified by the user selected points. With knowledge of the surface normals, P2 will calculate the image intensities at each of the points within the set. Then the 2-D light direction and ambient term can be estimated through the least-squares estimation method described in the technique overview.

III. Code and Solution

The program, P2, will initially read an image, usually in color, and grayscale the image before displaying. This is to calculate the image intensities later on in the program. P2 will then allow the user to choose the number of points he or she wishes to enter. The coordinates are recorded (note that the entire image is treated as the fourth quadrant of a Cartesian coordinate system for simplicity) and sent to an optimizing function, fitcirc(), that utilizes fminsearch() in order to fit a circle along the contour on which the user selected his or her points. The function then returns the radius and coordinates of the center of the circle of best fit.

With the circle data, the program calculates the unit vectors of each of the points that the user selected. This unit vector is used to determine the angle of the point on a unit circle with the point (-1, 0) being both 0 and 360 degrees. Thus, the program can determine which point forms the largest angle and which forms the smallest, giving us the two end points that form the boundaries of the user selected region. By bounding the region, the program can collect sample points within the user-specified region. After the sample points have been collected in the form of unit vectors to represent the surface normals, the program records the image intensity values from the grayscale image for each of the sample points on the circle before going into the least-squares estimation method described in the technique overview to solve for the **v** term, which contains the directional light estimate as well as the ambient light term, giving us our solution. Once we get both the x- and y- component of the directional light vector, we can obtain the angle at which the light is shining from, as is shown in the figures in the next section.

IV. Examples

There are several examples to display the program in action. Below are the results from the images of a circle with light from -45, 0, and 45 degrees relative to vertical. In each case 5

points were selected along the contour, paying careful attention not to pick points that would be over 180 degrees and would undermine our forth assumption. The green stars represent the borders of the user-selected region on the circular contour. The red dots are the sampled points along the contour within the user-selected region.

Figures 1, 2, and 3 represent P2 working on the images with the light source at -45, 0, and 45 degrees relative to vertical respectively. The green dots represent the user selected points. Two of the green dots become stars to specify that those points mark the boundaries of the user-selected region. A dialog box displays the result of the program.

Figure 1. -45 Degrees Light Source

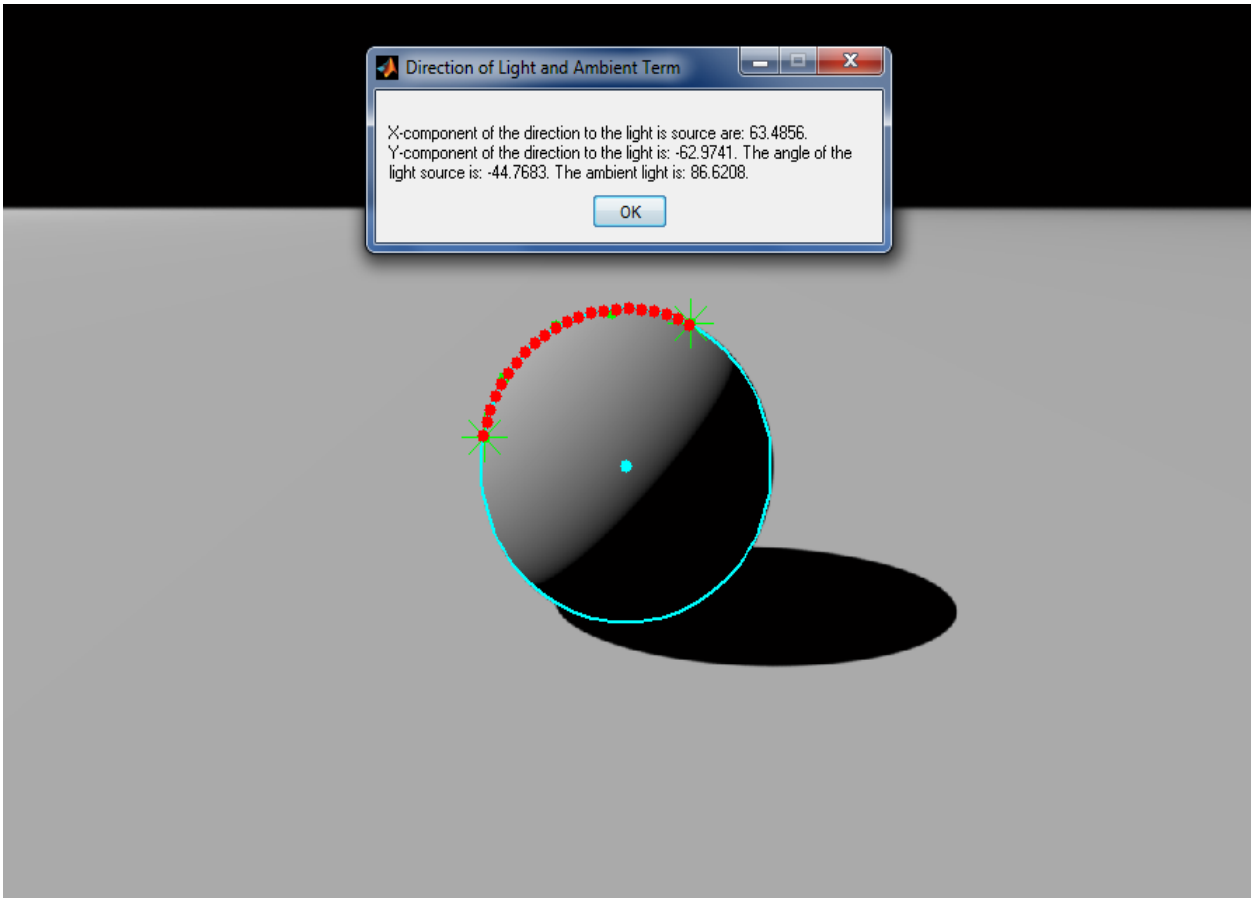


Figure 2. Vertical, in this case ~90 Degrees Light Source

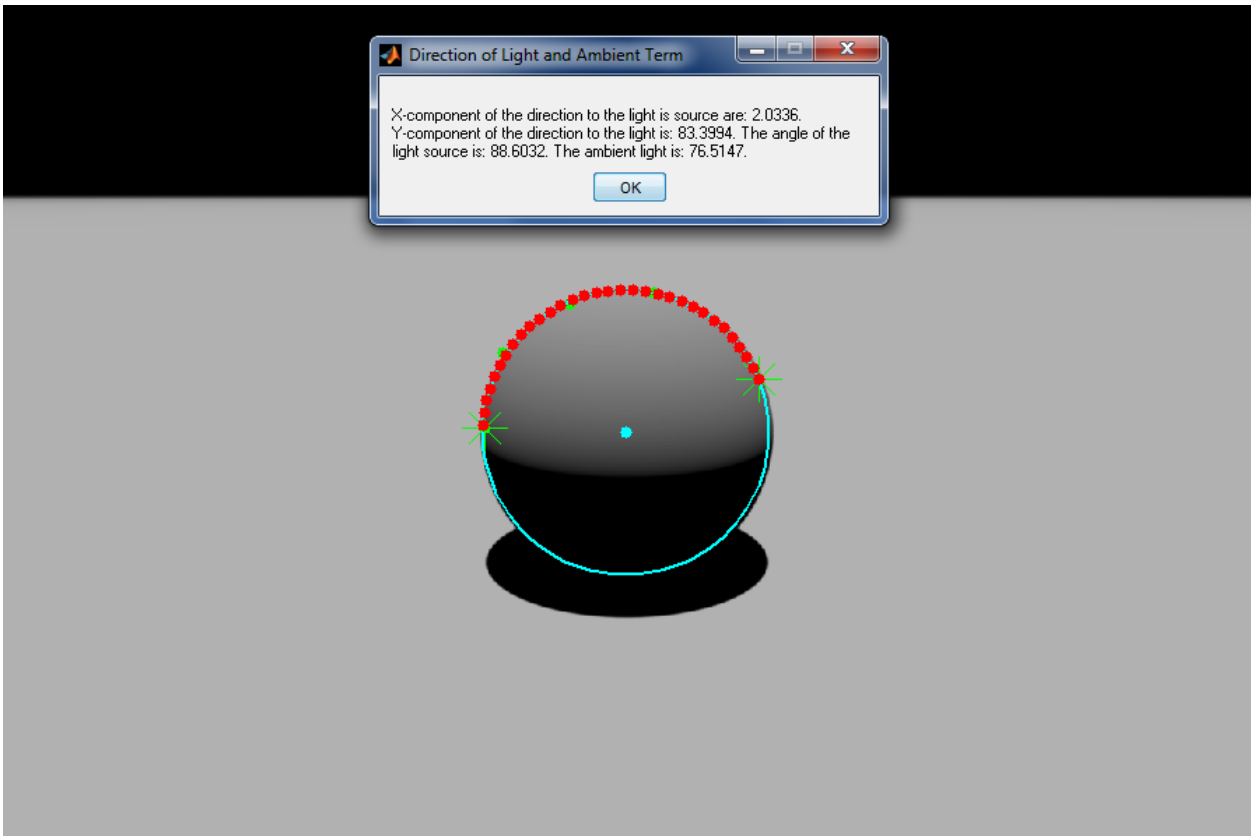
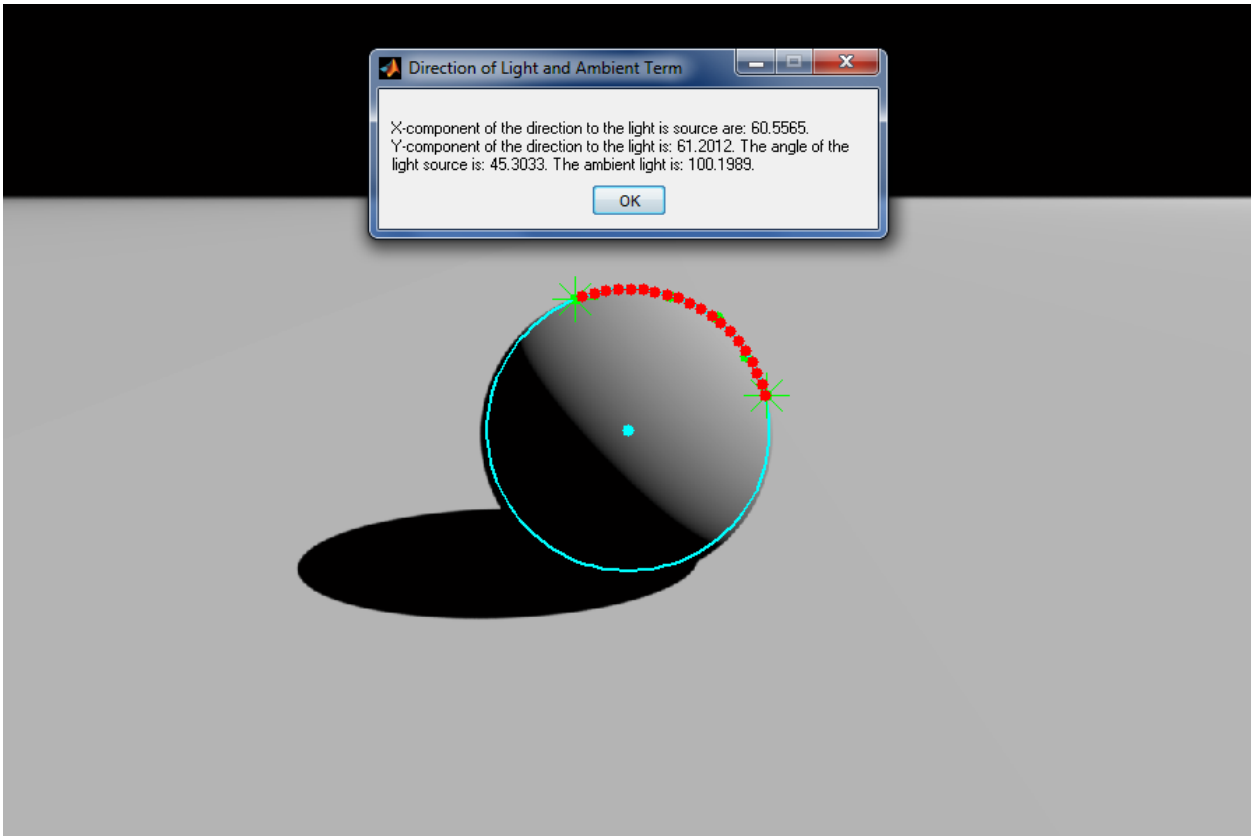


Figure 3. +45 Degrees Light Source



V. Edge Cases

There are several edge cases to consider with this program. The first deals with only 2 user input points. With such little information, fitting the circle would not be feasible. Thus, when the user inputs only 2, the program will halt and display the appropriate error message.

Using the method in my program to calculate the boundaries of the user-selected region has one drawback, which is when the unit circle goes from below the horizontal axis to above and the user selects points from both region. In that case my method would assume the two closest points (the ones near 0 and 360 degrees) to be the boundaries. My solution to this is to move back further away and collect sample points from both sides: above the horizontal axis and below, starting from the point furthest away and moving towards the horizontal axis. Refer to Figures 4 and 5 for examples. The green stars are the original boundaries while the blue and yellow are the new boundaries of the user-specified region.

Figure 4. -45 Degrees Light Source – Edge Case

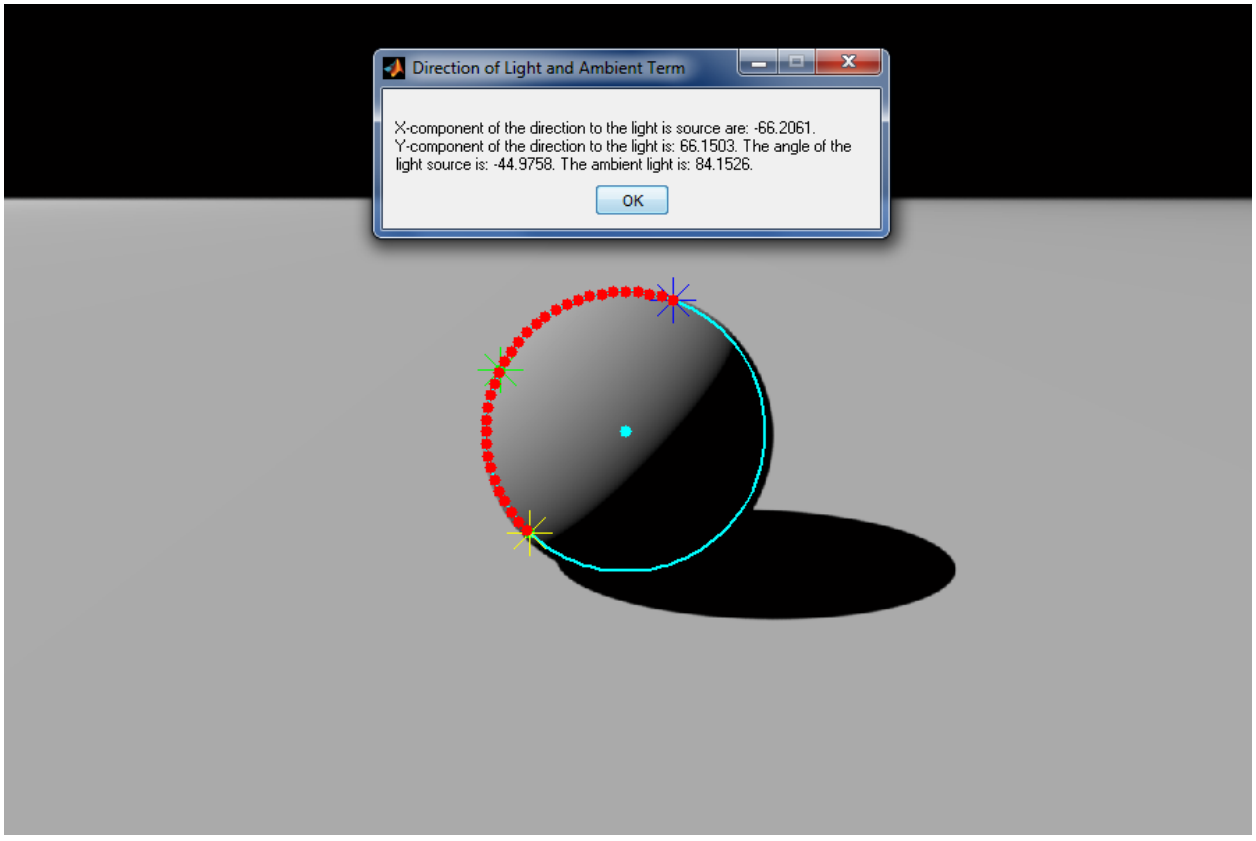


Figure 5. +45 Degrees Light Source – Edge Case