CS 477 HW #5

Questions completed: All undergrad level questions

Programming/Deliverables: MATLAB

Estimated time for assignment: 14 hours

A. Finding the camera matrix.

The camera matrix M found using homogenous least squares is reported below.

M =

-0.1539 0.0384 -0.0302 0.8477

0.0187 0.0463 -0.1351 0.4844

0.0000 0.0000 0.0000 -0.0009

To compute the matrix in MATLAB, we have to first build matrix P as described in the slides, compute $U = P^T * P$, get the column of eigenvectors corresponding to the smallest eigenvalue of U, and then reshape this column of eigvectors into our 3x4 matrix M.

Below in Figure 1, I provide a plot of the actual correct image coordinates versus the estimated image coordinates from our derived camera matrix M.

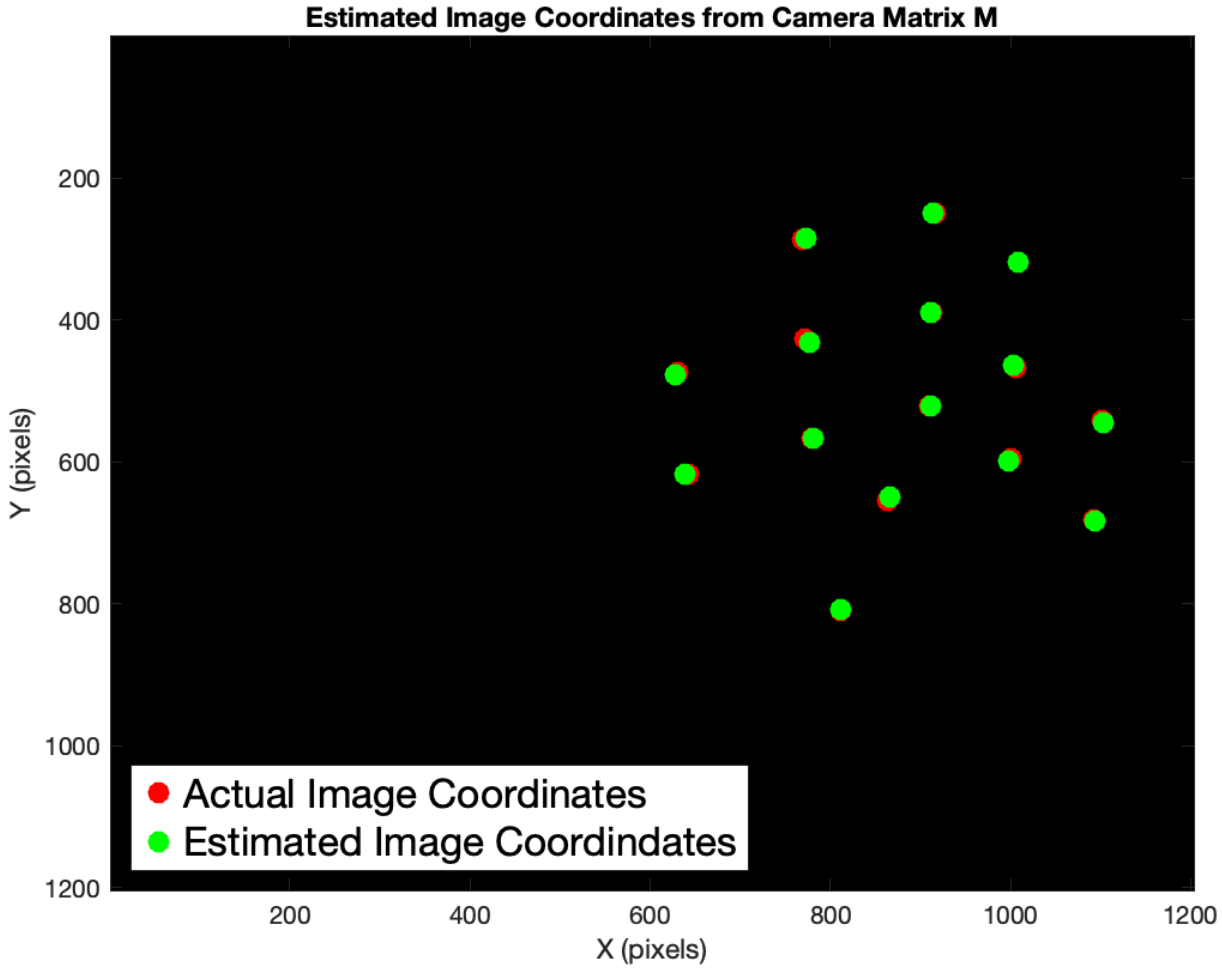

Figure 1: Actual image coordinates (red) versus the estimated image coordinates (green) from camera matrix M plotted in MATLAB. The estimated image coordinates are the projection of our world coordinates from hw04 using our camera matrix M.

Notice in Figure 1 above that the estimated image coordinates (green) are mostly overlapping (and covering up) the actual image coordinates (red) which serves as an informal visual check that our camera matrix M is fairly accurate in projecting our clicked world coordinates from hw04 (compared to the previous two camera matrices provided).

The root mean square (RMS) error for our estimated image coordinates is reported below.

Camera Matrix M RMSE = 3.68

This means that, on average, our camera matrix M's projection of our world coodinates to image coordinates is less then 4 pixels off of where they should be.

Compared to camera matrix 1's RMSE of 23.9 and camera matrix 2's RMSE of 40.7 (from hw04), we conclude that we have found a better camera matrix M.

The sum of the squared error is not the same error that the calibration process is minimizing. Although what we care about is the camera re-projection error (the distance between the actual and estimated points) which is equivalent to the sum of the squared error, our camera calibration is not finding this error because the matrix U in our calibration process is not obtained by computing something similar to $U = (x_i - \bar{x}, y_i - \bar{y}), Ux \approxeq 0$ like in homogenous least squares best-fit line which shows that we are minimizing residuals. Instead, what we have is $(U; V; W) = MP$ where M is the camera matrix and P are the 3D points in homogenous coordinates. We are then finding a matrix M that provides U, V, W that project to u, v by $u = U/W,\ v = V/W$, and this minimization is expressed in homogenous coordinates $U = m_1 * P, V = m_2 * P, W = m_3 * P \rightarrow u_i = \frac{m_1 * P}{m_3 * P}, v_i = \frac{m_2 * P}{m_3 * P} \rightarrow (m_1 - u_i m_3)P = 0, (m_2 - v_i m3)P = 0 \rightarrow Pm = 0, |m| = 1$ where P in $Pm = 0$ is our matrix P for homoegenous least squares (and not the homogenous coordinates P) and m is our camera matrix M as a vector. Thus, the minimization problem of $Pm = 0, |m| = 1$ is not the same as minimizing the camera re-projection error synonymous with the sum of the squared error

B. Rendering a sphere.

Below in Figure 2, I provide the image of a rendered sphere obeying the Lambertian reflectance model.

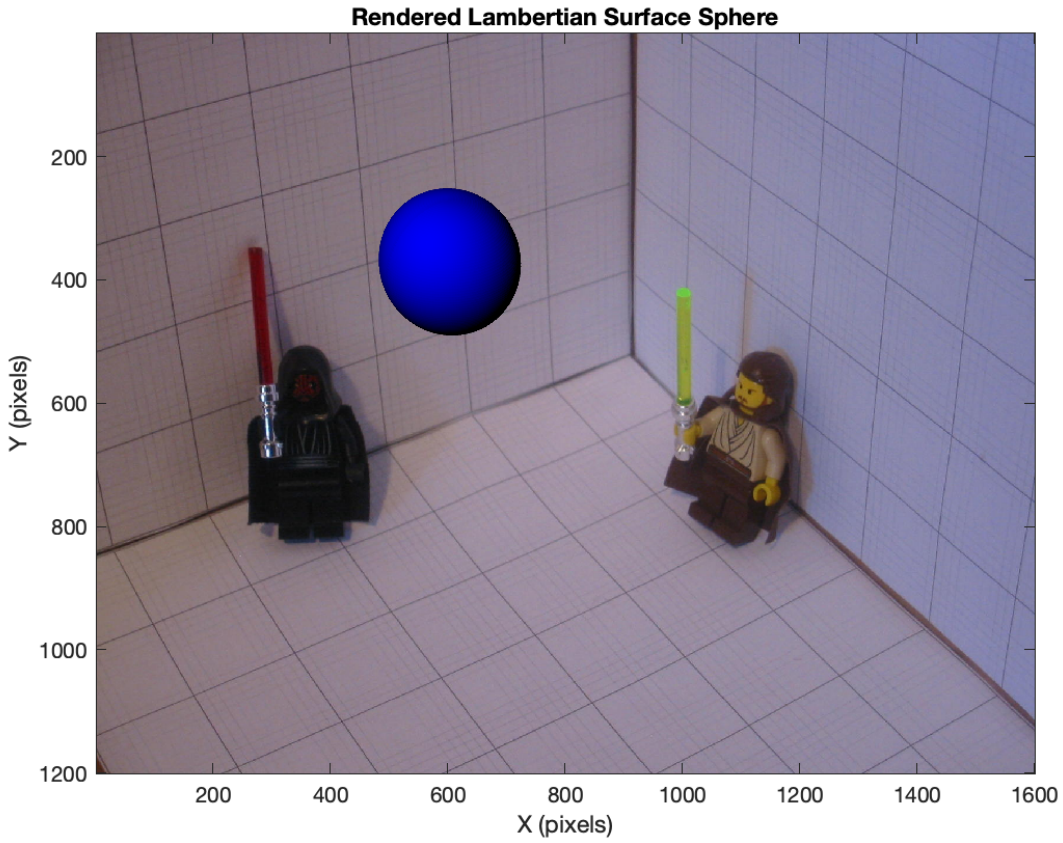

Figure 2: Lambertian surface sphere rendered in MATLAB. The camera is assumed to be at position (9, 14, 11) in world coordinates, and the light is reportedly at (33, 29, 44) in world coordinates. Darth Maul (left) and Obi-Wan Kenobi (right) are leaning against the X,Z plane and Y,Z plane respectively (the world coordinates as defined in the last assignment).

In Figure 2 above, the shading of the lambertian surface sphere seems consistent with the shading/shadows of Darth Maul and Obi-Wan. The world coordinates of the sphere were rendered using the simple sphere model implemented in a nested for loop as suggested in this assignment.

One nuance was to filter out the points of the sphere that were not visible from the camera's perspective (although in actual practice this does not impact the rendered sphere that you see, but it is more computationally efficient to have less points on

the MATLAB image plot since all points of the image are re-rendered every time you resize the image), and the visibility of the points were determined as suggested in the assignment (taking the dot product of the direction of the surface normal of the point and the distance of camera to the point.) Specifically, the direction of the surface normal was computed as

$\vec{N} = (\frac{x-x_0}{r}, \frac{y-y_0}{r}, \frac{z-z_0}{r})$

where r is the radius of the sphere. Having obtained the world coordinates of sphere, the world coordinates were projected onto our 2D image using our camera matrix M which was computed in Part A.

The last interesting thing of note was plotting each point with its corresponding lambertian surface luminance mapped to the blue color channel from 0 to about 255. An albedo value of 4 was used as it produced a max surface luminance value of 248 which then gave us the mapping to our blue color channel for free (negative surface luminance values imply that light is coming from behind the surface, so these values are just floored to 0 for our blue color channel).

In Figure 3 below, I provide the rendering of our previous lambertian surface sphere with the twist that the light point is assumed to be at world coordinate position (-30, 0, 0).

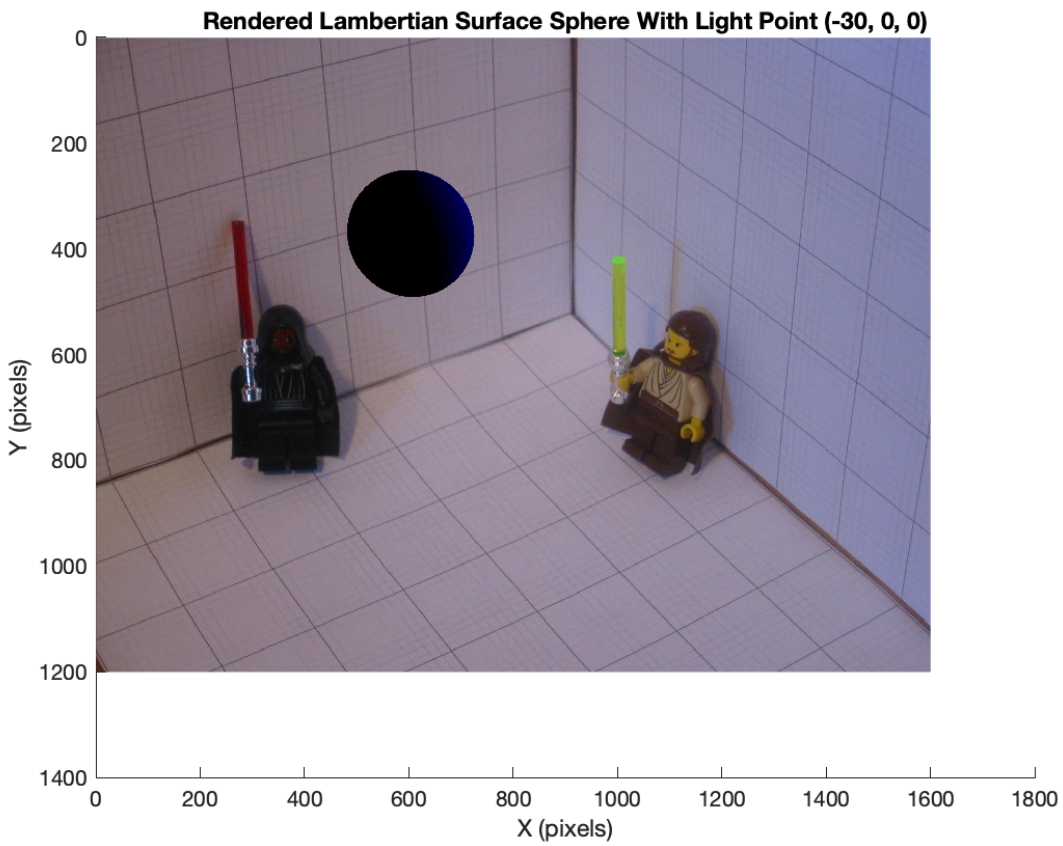

Figure 3: Lambertian surface sphere plotted with light point at world coordinate (-30, 0, 0).

In Figure 3 above, the resulting shading of the lambertian sphere makes sense since the light point is along the negative X-axis, and we see most of the surface of the lambertian sphere facing away from the negative X-axis is dark while the edge of the sphere facing the negative X-axis (the right edge of the sphere) from our view shows some reflection of the light point.

A tiny nitpick is that I would have expected more of the bottom edge of sphere to be lit up since the sphere is higher (on the Z-axis) relative to the light point which lies right on the X-axis. Perhaps since the sphere is also placed at Y = 2 and not directly along the X-axis with the light point, it makes sense that there is less surface reflection at the bottom of the sphere from our point of view, but I am not too sure on this as it really would seem that the light point should be higher up than it actually is just from scrutinizing the reflection. This could also be a rendering error on my part.