HOMEWORK 5: 3D RECONSTRUCTION & PHOTOMETRIC STEREO

16-720B Introduction to Computer Vision (Fall 2020)
Carnegie Mellon University
Homework PDF Link
BY: Feng Xiang*
DUE: Thursday, November 19, 2020 11:59 PM

Notes

- I worked on this project with Tom Xu and Gerald D'Ascoli

*Compiled on Thursday 19th November, 2020 at 19:39

Question 01

Q1.1

Background The equation of the Fundamental matrix, F, to map a point, x, on the C1 image to a line, l, on the C2 image is:

$$Fx = l' \tag{0.1}$$

Answer In the case where the coordinates of C1 and C2 are normalized about the coordinate origin (0,0) about the principal axis and 3D point, P, then the equations for x and l are equation to:

$$x = [0, 0, 1]^T$$
$$l' = [l'_x, l'_y, 0]^T$$

The elements of the x vector is based on the homogeneous coordinates of a point located at the origin. The elements of the l' vector is based on a line that passes through the origin and is based on the following line equation:

$$c + ax + by = 0$$
$$l' = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

The expansion of the fundamental matrix above will be equation to:

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} l'_x \\ l'_y \\ 0 \end{bmatrix} \tag{0.2}$$

Expanding the equation for the third element to the matrix equation shown above, the equation is given as:

$$f_{31} * (0) + f_{32} * (0) + f_{33} * (1) = 0$$

Analyzing the equation shown above, one can determine that the coefficient of the f_{33} must equal zero if the third element of the line vector is equation to zero.

References

- Homogeneous Representations of Points, Lines, and Planes
- Lecture Slides: Single View Geometry

Q1.2

Background The respective equation and vector representation of a line are given by:

$$ax + by + c = 0$$
$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Given a point correspondence between two camera images, the epipolar line on one camera image is defined as the line containing the corresponding point on on that image plane and the epipole. The epipole is defined as the shortest distance vector between the two camera centers.

Between two camera images, a corresponding point in one camera image is translated into a line that passes through that corresponding image point in the second camera image. This is given by the following matrix equation that contains the Essential Matrix:

$$Ex = l' \tag{0.3}$$

Answer Given that one camera is distanced from the first camera along the x-axis as pure translation, the epipole distance vector between the two cameras must pass through the centers of the two camera images. This means that epipole distance vectors travels purely along the x-axis past the two camera centers. In addition, this means that the epipole of the two camera images are located at infinity. The figure shown below (see Figure 0.1) details the epipole distance vectors between the two camera images.

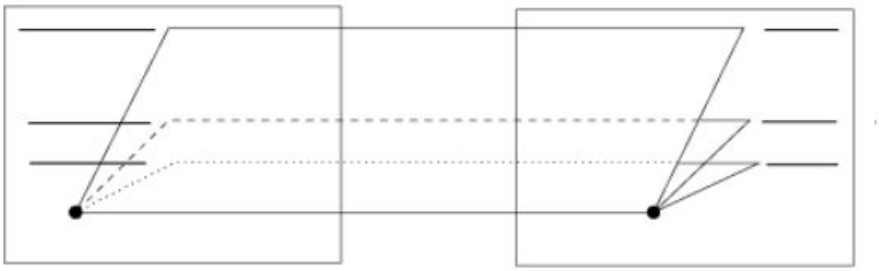


Figure 0.1: Visualization of two camera images separated by pure translation along one axis

In addition, a camera image that is shifted apart along the x-axis contain pixel values that are also shifted along the x-axis. For example, the homogeneous vector of a pixel point, $[1, 1, 1]^T$, shifting along the x-axis by 2 units has an updated pixel location of $[-1, 1, 1]^T$.

Referring to the Essential Matrix equation, the corresponding point denoted on the first camera image is translated into a line on the second camera image. The output line from the equation will be a line that passes through the corresponding image point and is parallel along the x-axis. Performing the reverse operation from the second camera image to the first, the same is true. A corresponding point on the second camera image is translated as a parallel line along the x-axis (line drawn between the corresponding point and the epipole which lies at infinity).

References

- Lecture Slides: Single View Geometry
- Lecture Slides: Epipolar Geometry
- Lecture Slides: Essential Matrix

Q1.3

Background Given a world point in 3D space, the equation to acquire the homogeneous coordinate in the camera image is derived to be:

$$x_c = KR(X_w + R^{-1}t)$$
$$C = R^{-1}t$$

Given an epipolar plane between two camera images and a 3D point being interpreted between the two points, there exists two properties. The first property is that the vectors x, t, and x' are coplanar. The second property is that the transformation between point x on the first camera image and x' in the second camera image can be interpreted as the point undergoing a translation and rotation. The equations of the two respective properties are given below:

$$(x - t)^T(t \times x) = 0$$
$$x' = R(x - t)$$

If the Essential Matrix is transforming the point x on the first camera image to line l' on the second camera image, the matrix can be decomposed to be the cross product between the translation vector at the subsequent rotation vector as shown below:

$$E = [t] \times R = R[R^T t]_x$$

Because the Fundamental Matrix can be interpreted as an Essential Matrix when the camera intrinsic properties are not known, the following equation is derived:

$$F = (K'^{-1})^T E K^{-1}$$

Answer Given separate rotation matrices R_i and translation vectors t_i for each of the two respective camera images, which translate 3D world coordinates to 2D homogeneous coordinates on the camera image, the following equations are created for camera image spaces 1 and 2:

$$x_1 = KR_1(X_w + R_1^{-1}t)$$
$$x_2 = KR_2(X_w + R_2^{-1}t)$$

Solving for X_w for both equations and substituting the two together, we get the equation:

$$R_2^T K^{-1} x_2 - R_2^T t_2 = R_1^T K^{-1} x_1 - R_1^T t_1$$
$$K^{-1} x_2 = R_2 R_1^T K^{-1} x_1 - R_2 T_1^T t_1 + t_2$$
$$x_2 = k R_2 R_1^T K^{-1} x_1 - K R_2 R_1^T t_1 + K t_2$$
$$x_2 = K R_2 R_1^T K_{-1}(x_1 - K t_1 + K R_1 R_2^T t_2)$$

With the final equation shown above taking the form of $x_2 = R_{rel}(x_1 - t_{rel})$, the relative rotation and translation terms are equation to:

$$R_{rel} = KR_2R_1^TK_{-1} \tag{0.4}$$

$$t_{rel} = Kt_1 - KR_1R_2^Tt_2 \tag{0.5}$$

The equation for the Essential Matrix is then derived to be:

$$E = t_{rel} \times R_{rel} \tag{0.6}$$

The equation for the Fundamental Matrix is then derived to be:

$$F = K(t_{rel} \times R_{rel})K^{-1} \tag{0.7}$$

References

- Lecture Slides: Camera Matrix
- Lecture Slides: Essential Matrix
- Lecture Slides: Fundamental Matrix

Q1.4

Background Similar to the behavior of an Essential Matrix, the Fundamental Matrix is used to match a point in one camera image to a line in the other camera image via the following matrix equation:

$$Fx = l'F = K'^{-1}EK^{-1}$$

Where E is the Essential Matrix and equals:

$$E = t_{rel} \times R_{rel} E = [t]_x R$$

The homogeneous vector x denotes a corresponding point on the first camera image. The line vector l' denotes the coefficients of a line that passes through the matched point in the second camera image.

A skew symmetric matrix is equation to a matrix where the transpose of the matrix is equal to the negative of the original matrix (i.e. $A^T = -A$. If the matrix is 3x3, the skew symmetric matrix is derived to be:

$$\begin{bmatrix} 0 & f_1 & f_2 \\ -f_1 & 0 & f_3 \\ -f_2 & -f_3 & 0 \end{bmatrix}$$

Answer Assuming that object is flat and all points on the object are equally distant from the camera, one can assume that object and its reflection on the camera image are equally distant and of the same size as shown in the image. The only difference is that they are shifting along one direction.

One can assume no rotation (i.e. $R_{rel} = I$) and translation being arbitrary (i.e. $t_{rel} = [t_x, t_y, t_z]$). The equation for the fundamental matrix based on the rotation and translation definitions are equal to:

$$F = K'^{-1}([t_{rel}]_x R_{rel})K^{-1} F^T = K^{-1}([t_{rel}]_x R_{rel})^T K'^{-1}$$

Where $t_{rel} \times R_{rel}$ is equal to the skew symmetric matrix of the translation times the rotation matrix. In addition, because the transpose of the Fundamental Matrix is equal to essentially equal to the reverse position of the K matrices and the negative of the skew symmetric matrix of the translation, one can deduce that the Fundamental Matrix is itself a skew symmetric matrix.

Reference

- Lecture Slides: Fundamental Matrix
- Lecture Slides: Essential Matrix

Question 2

Q2.1

Background Given a Lambertian model, the equation for the given intensity of a pixel location is denoted to be:

$$I = \boldsymbol{\rho}\vec{n} \cdot \vec{s}$$

Where I is the pixel intensity as seen through the camera, $\boldsymbol{\rho}$ is the albedo, N is the surface normal vector perpendicular to the surface of the object, and s is the light source vector into the object at the point-in-question.

As a mathematical identity, the following equation holds true:

$$cos(\boldsymbol{\theta}) = \vec{n} \cdot \vec{s}$$

Answer Given the figure shown below (See Figure 0.2), there exists a proportionality between the angle of attack of the light source on the surface normal and the reflected light off of the point on the object. This phenomena is called Lambert's Cosine Law. Geometrically, as the angle of attack of the light source increases against the surface normal vector, the larger the area that the packets light are contacting on the surface. This means that for a given point, the reflected off light at that specific point is less because the distribution of reflected light is more spread out with a larger angle of attack of the light source. The mathematical description of this geometric phenomena is described by the dot product between the light source vector and the surface normal vector.

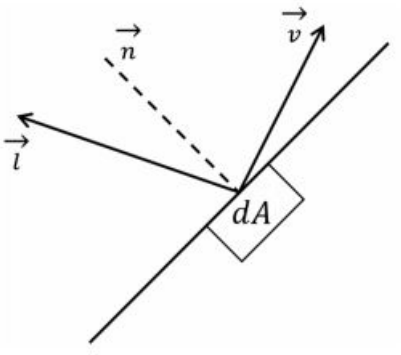


Figure 0.2: Photometric Stereo Vector Visualization

The dot product comes from the component of the light source vector that travels along the surface normal vector. This denotes to the amount of light that is reflected off the of the surface at that point on the object.

The projected area comes into the following equation shown below:

$$I = \boldsymbol{\rho}\vec{n} \cdot \vec{s} \tag{0.8}$$

Where the intensity that is coming into the camera is based on the equation of the albedo and the projected area between the light source vector and the surface normal at that point on the object.

The viewing direction would not matter because the reflected light out of the point on the object travels in all directions outward. No matter the viewing direction, the sensor will absorb the same degree of reflected light, assuming theoretical conditions.

References

- Lecture Slides: Physics Based Vision

Q2.2

Background Given a surface normal vector that is perpendicular and facing outward from a point on the object, there exists a tangential vector that travels along the surface at that point on the object and is parallel to the surface of the object at that point. This means that the dot product between the surface normal vector and the tangential vector is equal to zero.

$$N \cdot V = 0$$

Answer Given a vector V_x which is the tangential vector along the surface of the object in the x-direction at the point where the surface normal exists, the 1D equation is the following:

$$V_x = \frac{dz}{dx}$$

Computing the dot product between the surface normal vector and the tangential x-direction vector yields the a result of zero and the following equation:

$$\begin{aligned} N \cdot V_x &= (n_1, n_2, n_3) \cdot \frac{dz}{dx} = 0 \\ N \cdot V_x &= (n_1, n_2, n_3) \cdot (1, 0, \frac{dz}{dx}) = 0 \\ 0 &= n_x + n_z \frac{dz}{dx} \end{aligned}$$

Simplifying the equation, the following derivation exists:

$$\frac{dz}{dx} = -\frac{n_1}{n_3}$$

Performing a similar proof for the y-directional tangential vector V_y, the derivation is shown below:

$$\begin{aligned} V_y &= \frac{dz}{dy} \\ N \cdot V_y &= (n_1, n_2, n_3) \cdot \frac{dz}{dy} = 0 \\ N \cdot V_x &= (n_1, n_2, n_3) \cdot (0, 1, \frac{dz}{dy}) = 0 \\ 0 &= n_2 + n_3 \frac{df}{dy} \\ \frac{dz}{dy} &= -\frac{n_2}{n_3} \end{aligned}$$

References

- Lecture Slides: Physics Based Vision

Question 3

Q3.2.1

To use the 8-Point Algorithm to compute the Fundamental Matrix, the following figure (see Figure 0.3) overlays the general steps to take:

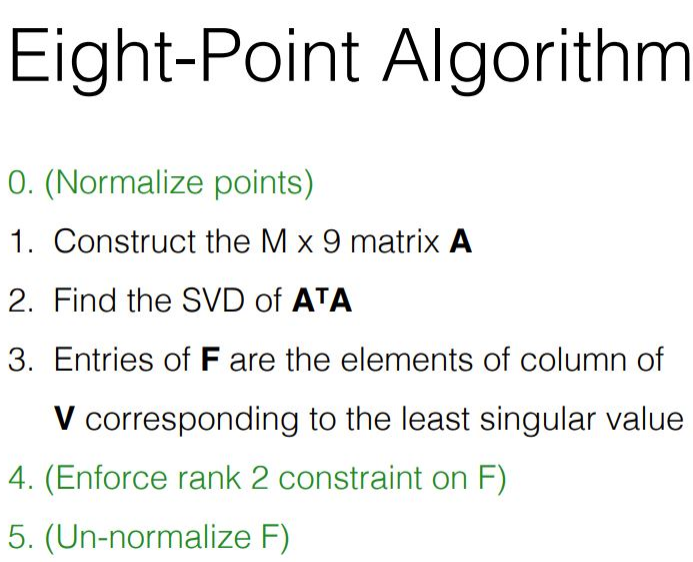


Figure 0.3: 8 Point Algorithm for Fundamental Matrix Overview

Answer The figure shown below (see Figure 0.4) displays the outputted Fundamental Matrix after running the Section 3.2.1 code of the *CheckA4Format.py* file:

	0	1	2
0	-210.328	-65.1113	-442.571
1	-22350.4	341.91	12.5802
2	460.748	-7.00898	-0.00412612

Figure 0.4: Fundamental Matrix Output

A visualization of the Fundamental Matrix using manually places key points is shown below (see Figure 0.5). The *displayEpipolarF* function was used to create the visualization:

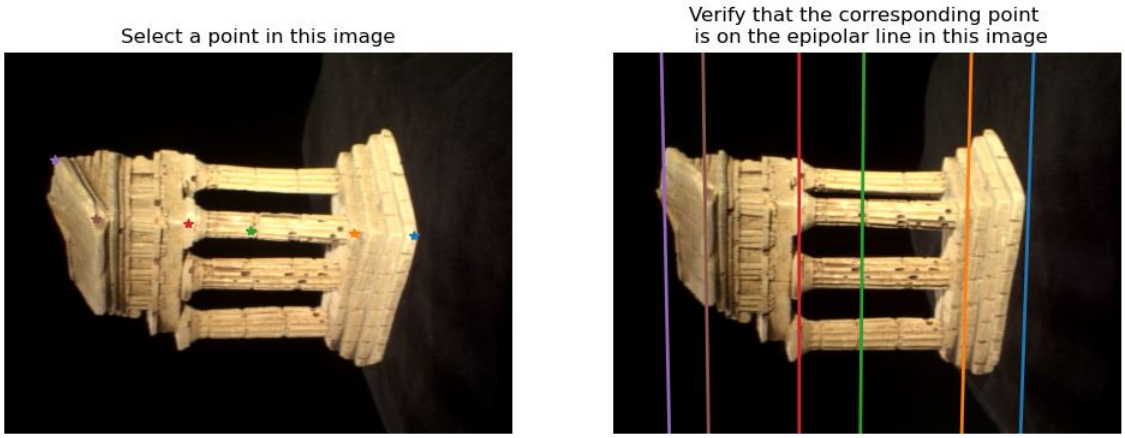


Figure 0.5: Fundamental Matrix Visualization

References

- Lecture Slides: 8-Point Algorithm

Q3.3.1

Background N/A

Answer See *essentialMatrix* function in the *submission.py* file for code implementation to the question.

Given a Fundamental Matrix F, the Essential Matrix E is determined by the following matrix equation:

$$E = K_2^T F K_1$$

References

- Lecture Slides: Fundamental Matrix

Q3.3.2

Background Given the point correspondence and camera matrices of the two camera images, the 3D point can be reconstructed using Least Squares as described in the matrix equation shown below (see Figure 0.6):

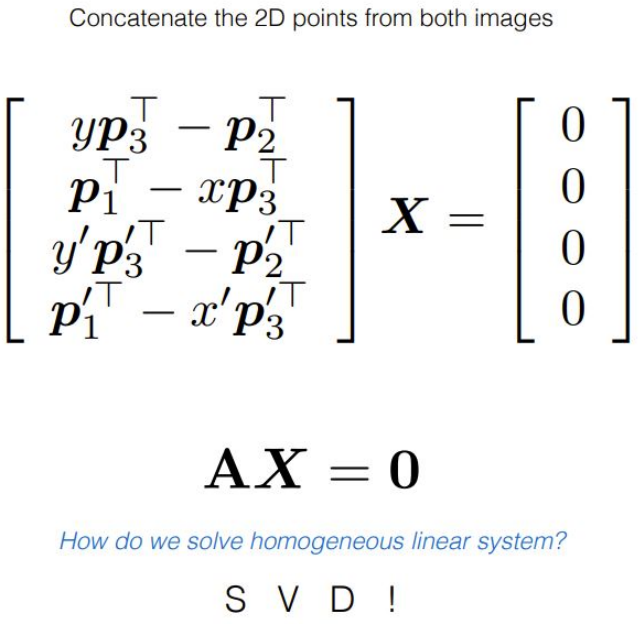


Figure 0.6: Fundamental Matrix Output

After computing the A matrix, SVD is performed on the A matrix. The last column corresponding to the V matrix is determined to be the column with the lowest reprojection error, therefore those values in the columns are in the least squares estimate of the 3D point.

Answer Given the following C1 and C2 matrices shown below:

$$C_1 = \begin{bmatrix} \vec{c_1} \\ \vec{c_2} \\ \vec{c_3} \end{bmatrix}$$

$$C_2 = \begin{bmatrix} \vec{c_1'} \\ \vec{c_2'} \\ \vec{c_3'} \end{bmatrix}$$

Where each element vector in the camera matrices denotes a row vector that $1x4$, the A_i matrix can be determined to be equal to:

$$A_i = \begin{bmatrix} y\vec{c_3} - \vec{c_2} \\ \vec{c_1} - x\vec{c_3} \\ y'\vec{c_3'} - \vec{c_2'} \\ \vec{c_1'} - x'\vec{c_3'} \end{bmatrix}$$

Because the C_1 and C_2 matrices are computed using the random number functions in the Python code, the exact value of the reprojection error can differ significantly between different runs of the program. During on run of the program, the total reprojection error was computed to be the following shown below (see Figure 0.7):

```
ipdb> err
4522.031331990691

ipdb>
```

Figure 0.7: Reprojection Error Output

References

- Lecture Slides: Triangulation

Q3.3.3

Background Given a camera intrinsic matrix and extrinsic matrix, the camera matrix P is calculated based on the following equation shown in the figure below (see Figure 0.8):

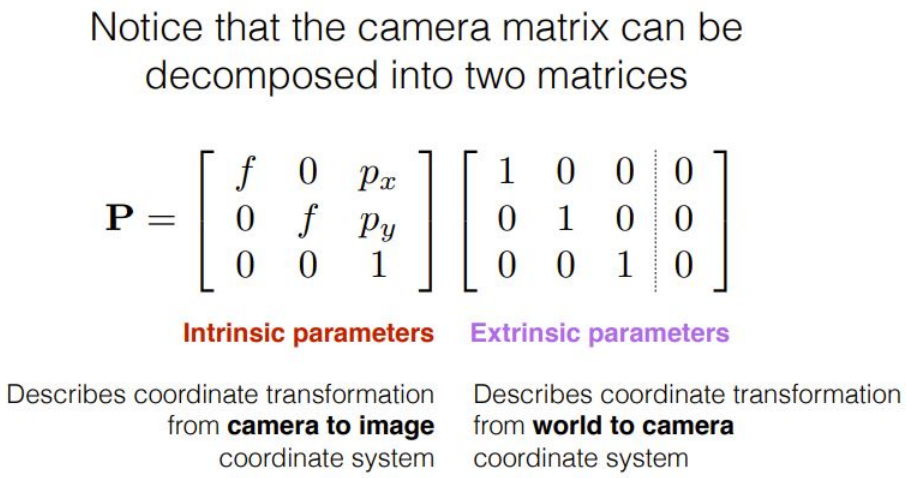


Figure 0.8: Camera Matrix Equation

Assuming that the camera matrix and extrinsic matrix for the first camera are respectively equal to:

$$M_1 = [I|0]$$
$$C_1 = K_1 M_1$$

The follow algorithm exists to compute the extrinsic matrix for the second camera as shown in the figure below (see Figure 0.9):

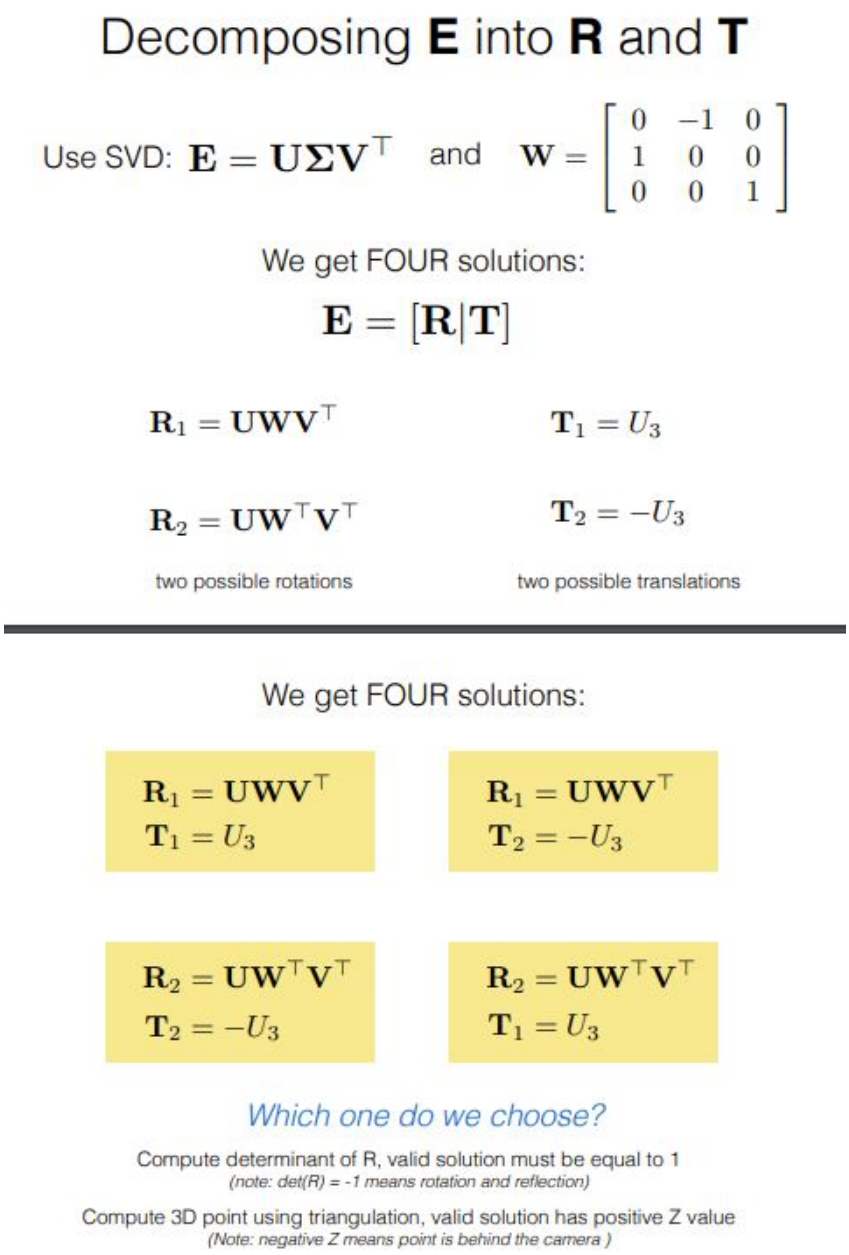


Figure 0.9: M2 Solution Algorithm

Answer See the *findM2.py* file for code implementation to the question.

References

- Lecture Slides: Fundamental Matrix
- Lecture Slides: Essential Matrix
- Lecture Slides: Camera Matrix

Q3.4.1

Background N/A

Answer The figure shown below displays a screenshot of the $epipolarMatchGUI$ function window showing selected points on the first camera image on the left hand side and the corresponding point along the epipolar line on the right hand side (see Figure 0.10):

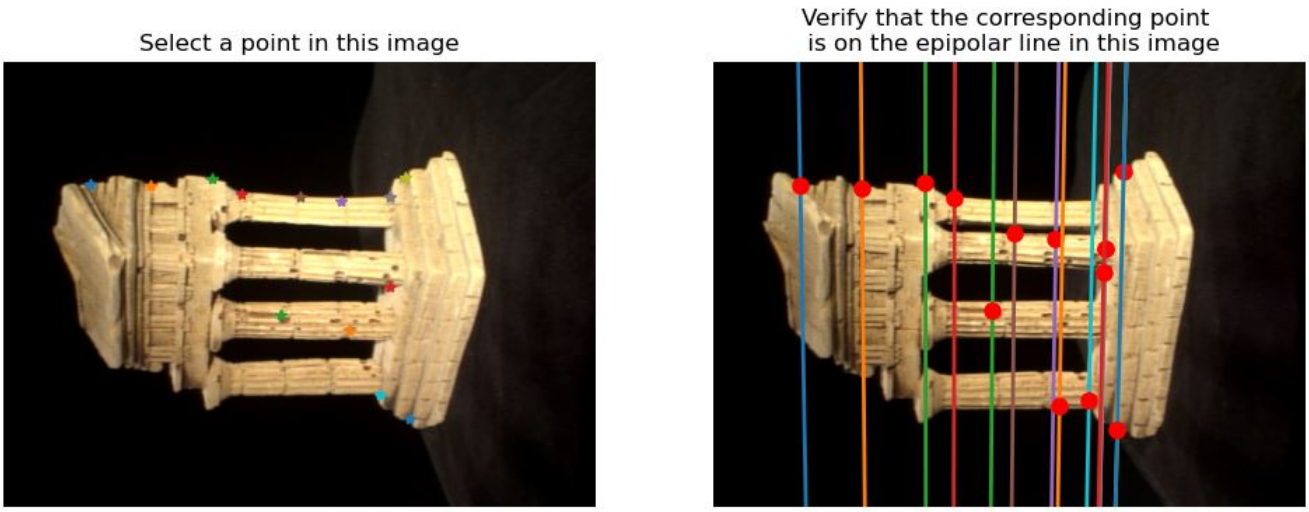


Figure 0.10: $epipolarMatchGUI$ Window Screenshot

To describe the behavior of the image above, the epipolar matching and point correspondence estimation perform well most of the time. For selected points on the first camera image that were distinct corners with intensity features neighboring those points, the model worked well to estimate a point that is close to the first image. For example, on the roof of the building where the corners are very distinct, the model worked well to estimate corresponding points that were matched.

When mid-sections of the columns were selected, however, the model was not able to perform as well. In some instances as shown in the image, the model estimated a point that was on the adjacent column. Point predictions at the bottom of the steps of the building performed the worst. Many points in the beginning top steps of the building were significantly offset. Any selected point located around the last two bottom steps of the building did not register a matched point anywhere near the building itself.

References

- Lecture Slides: Essential Matrix

Q3.4.2

Background N/A

Answer The following figure shown below displays the outputted point cloud of the object based on default views from the $scatter$ function from the $matplotlib$ python library (see Figure 0.11):

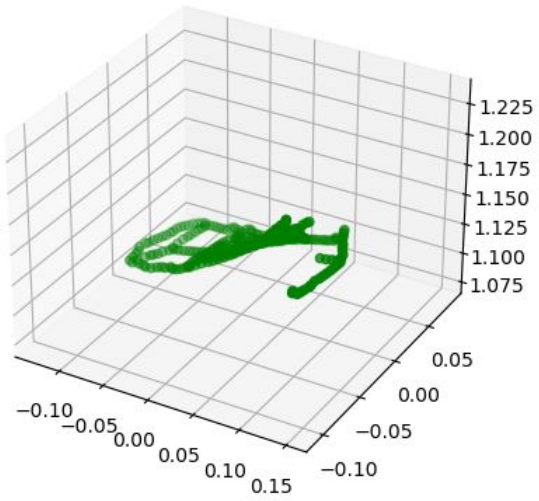


Figure 0.11: Point Cloud - Default View

Observing the figure shown above, one can see that the image is skewed from its intended viewing angle. Rotating the 3D plot helped to align the point cloud to be in the shape of the intended object.

The figure shown below (see Figure 0.12) displays a point cloud that is more inline with the expected shape of the object.

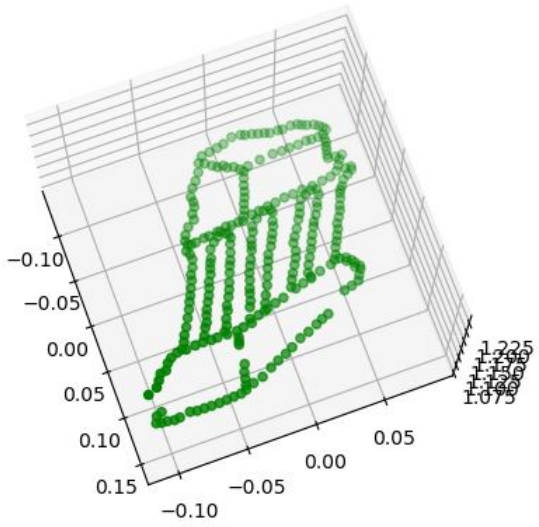


Figure 0.12: Point Cloud - View 02

The figure shown below(see Figure 0.13) displays the point cloud from another angle:

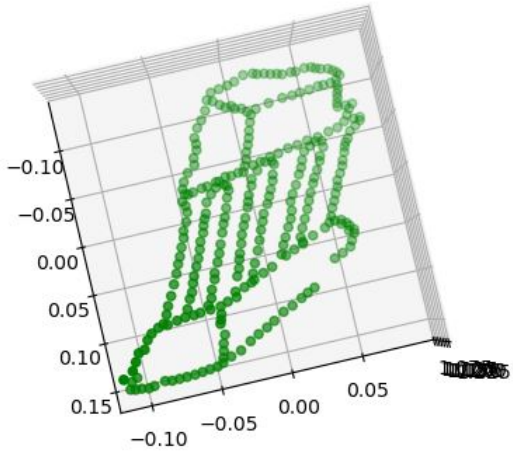


Figure 0.13: Point Cloud - View 03

References

- N/A

Q3.5.1

Background N/A

Answer The matrix shown below is the output matrix from the *ransacF* function using the *some_corresp_noisy.npz* dataset(see Figure 0.14):

	0	1	2
0	-2.55836e-08	-1.77474e-09	-0.00107745
1	5.29682e-09	-2.08348e-10	1.558e-05
2	0.00112813	-5.35815e-05	0.000333819

Figure 0.14: *ransacF* Output F Matrix

The output F matrix from the *eightpoint* function using the *some_corresp_noisy.npz* dataset is shown below for comparison (see Figure 0.15):

	0	1	2
0	-5.78497e-07	-4.07531e-09	0.000219496
1	-3.98657e-08	2.0814e-06	-0.000490397
2	0.000154911	-0.000457517	0.052593

Figure 0.15: *eightpoint* Output F Matrix

The screenshot shown below (see Figure) displays the total fraction of inliers compared to the total amount of points being inputted into the *ransacF* function. The fraction of inliers outputted aligns well with the expected amount of inliers described in the homework prompt for the question

```
ipdb> np.sum(inliers)
105

ipdb> 105/140
0.75
```

Figure 0.16: Calculated Total Amount of Inliers Compared to Total Amount of Inputs

The *ransacF* function utilized a max number of iterations equal to 2000 and a tolerance threshold of 10 in this instance. The function selected eight points from each of the variables $pts1$ and $pts2$. From there, the fundamental matrix was computed and line coefficients for all corresponding points in $pts1$ in $im2$ were outputted based on the matrix equation $l' = Fx$.

From there, a distance was calculated from the $pts2$ given point and their respected computed epipolar line. The equation used to compute the distance between point to line is given by the equation below:

$$dist = \frac{|ax_o + by_o + c|}{\sqrt{a^2 + b^2}}$$

If the calculated distance was less than the predefined tolerance (in this case the tolerance was 10), then the point correspondence is considered an inlier.

References

- Wikipedia: Distance to Line

Q3.5.2

Background To compute the Rodrigues Matrix based on a given Rodrigues vector, or epipole, the following derivation holds as shown in the figure below (see Figure 0.17):

$$\text{Let } R_{\text{rect}} = \begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \\ \boldsymbol{r}_3^\top \end{bmatrix}$$

$$\boldsymbol{r}_1 = \boldsymbol{e} = \frac{T}{||T||}$$

$$\boldsymbol{r}_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y & T_x & 0 \end{bmatrix}$$

$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$$

Figure 0.17: Rodrigues Matrix Equation

Inversely, to compute the epipole vector, or the Rodrigues vector, the following derivation holds as shown in the figure below (see Figure 0.18):

The rotation vector $\mathbf{r}$ corresponding to the rotation matrix

$$R \quad \text{such that} \quad R^T R = R R^T = I \quad \text{and} \quad \det(R) = 1$$

can be computed as follows:

$$A = \frac{R - R^T}{2} \quad , \quad \rho = \begin{bmatrix} a_{32} & a_{13} & a_{21} \end{bmatrix}^T$$

$$s = \|\rho\| \quad , \quad c = (r_{11} + r_{22} + r_{33} - 1)/2 \; .$$

If $s = 0$ and $c = 1$, then $\mathbf{r} = \mathbf{0}$. Otherwise, if $s = 0$ and $c = -1$, let $\mathbf{v} =$ a nonzero column of $R + I$. Then,

$$\mathbf{u} = \frac{\mathbf{v}}{\|\mathbf{v}\|} \quad , \quad \mathbf{r} = S_{1/2}(\mathbf{u}\pi) \; .$$

In this expression, the function $S_{1/2}(\mathbf{r})$ flips signs of the coordinates of vector $\mathbf{r}$ (assumed here to have norm π) to force it onto the half-hemisphere of Figure 1, in order to ensure uniqueness:

$$S_{1/2}(\mathbf{r}) = \begin{cases} -\mathbf{r} & \text{if } \|\mathbf{r}\| = \pi \text{ and } ((r_1 = r_2 = 0 \text{ and } r_3 < 0) \\ & \quad \text{or } (r_1 = 0 \text{ and } r_2 < 0) \text{ or } (r_1 < 0)) \\ \mathbf{r} & \text{otherwise.} \end{cases}$$

Finally, if $\sin\theta \neq 0$,

$$\mathbf{u} = \frac{\rho}{s} \quad , \quad \theta = \arctan_2(s, c) \quad \text{, and} \quad \mathbf{r} = \mathbf{u}\theta$$

where the function $\arctan_2$ is defined in equation (1).

Figure 0.18: Rodrigues Vector Equation

Answer See the *rodrigues* and *invRodrigues* functions in the *submission.py* file for code implementation to the question.

References

- Lecture Slides: Stereo Rectification
- Duke University: Vector Representation of Rotations

Q3.5.3

Background N/A

Answer

The figures shown below (see Figures 0.19 and 0.20), showcase the 3D points of the $some_corresp_noisy.npz$ points in 3D space before going through the $ransacF$ function. The points have also not gone through the $bundleAdjustment$ function yet:

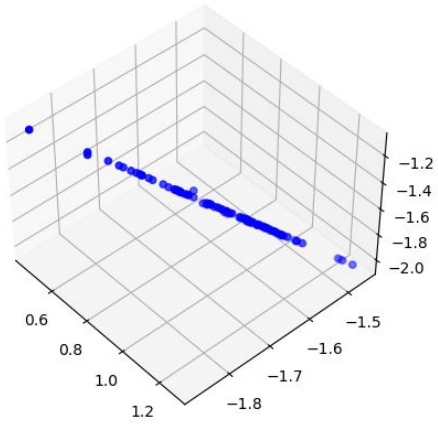


Figure 0.19: 3D Points, Pre-Ransac, View 1

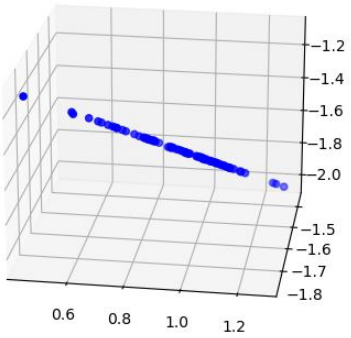


Figure 0.20: 3D Points, Pre-Ransac, View 2

The two figures shown below (see Figures 0.21 and 0.22) showcase the output 3D points of the $some_corresp_noisy.npz$ file after going through both $ransacF$ and $bundleAdjustment$:

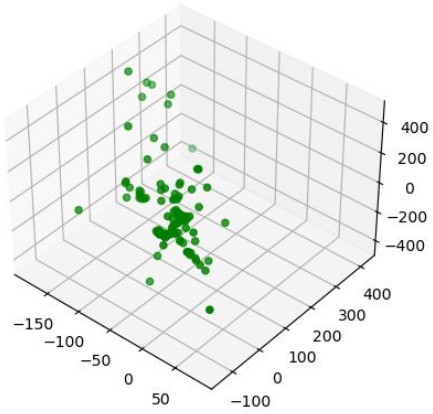


Figure 0.21: 3D Points, Post-Ransac and Bundle Adjustment, View 1

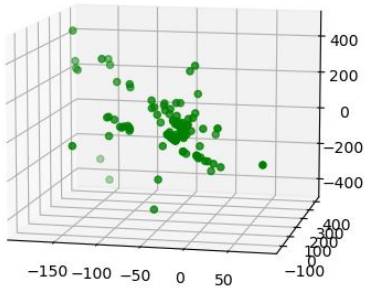
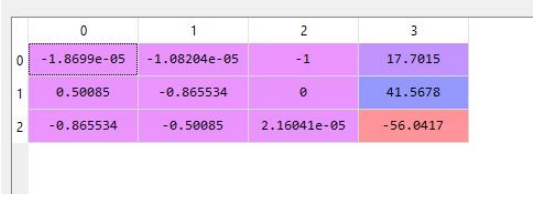


Figure 0.22: 3D Points, Post-Ransac and Bundle Adjustment, View 2

The reprojection error output was about 17556.

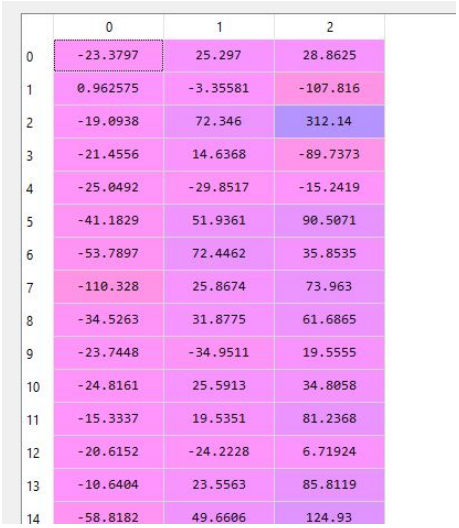
The figure shown below (see Figure 0.23) displays the output $M2$ matrix from the $bundleAdjustment$ function:



	0	1	2	3
0	-1.8699e-05	-1.08204e-05	-1	17.7015
1	0.50085	-0.865534	0	41.5678
2	-0.865534	-0.50085	2.16041e-05	-56.0417

Figure 0.23: $M2$ Output Matrix

The figure shown below (see Figure 0.24) displays a snippet of the w output array of 3D points from the $bundleAdjustment$ function:



	0	1	2
0	-23.3797	25.297	28.8625
1	0.962575	-3.35581	-107.816
2	-19.0938	72.346	312.14
3	-21.4556	14.6368	-89.7373
4	-25.0492	-29.8517	-15.2419
5	-41.1829	51.9361	90.5071
6	-53.7897	72.4462	35.8535
7	-110.328	25.8674	73.963
8	-34.5263	31.8775	61.6865
9	-23.7448	-34.9511	19.5555
10	-24.8161	25.5913	34.8058
11	-15.3337	19.5351	81.2368
12	-20.6152	-24.2228	6.71924
13	-10.6404	23.5563	85.8119
14	-58.8182	49.6606	124.93

Figure 0.24: Snippet of w array

References

- N/A

Question 4

: Testing code for Question 4 was implemented in the $submission.py$ file. The testing code only runs when the file is run as the main Python file.

Q4.1

Background N/A

Answer The figure shown below (see Figure 0.25) displays the rendering of the Lambertian sphere given the three lighting directions.

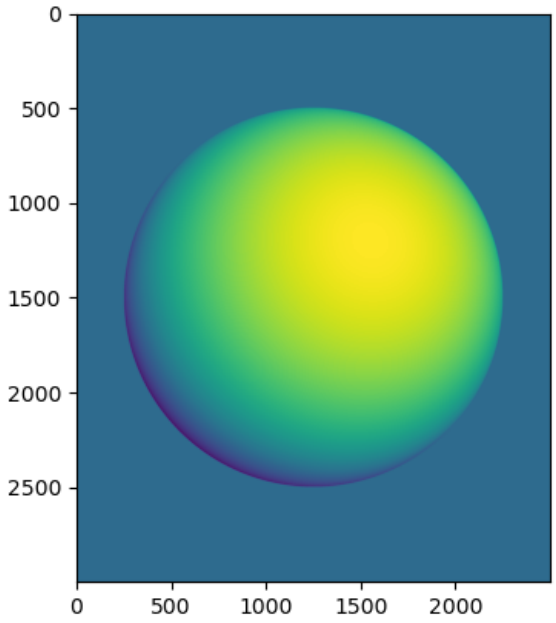


Figure 0.25: Lambertian Sphere Rendering Image

The pixels in the images were increased by a factor of 50 in order to accentuate the brightness and differences of the image.

References

- N/A

Q4.2.1

Background N/A

Answer See the $loadData$ function in the $submission.py$ file for code implementation to the question.

References

- N/A

Q4.2.2

Background N/A

Answer See the $estimatePseudonormalsCalibrated$ and $estimateAlbedosNormals$ functions in the $submission.py$ file for code implementation to the question.

The *scipy sparse* module was not used in this code implementation because the local machine used to run the program has ample resources to handle the code program.

References

- N/A

Q4.2.3

Background N/A

Answer see the $displayAlbedosNormals$ function in the $submission.py$ file for code implementation to the question.

The figure shown below showcases the grayscale image of the Albedo output at each pixel (see Figure 0.26):

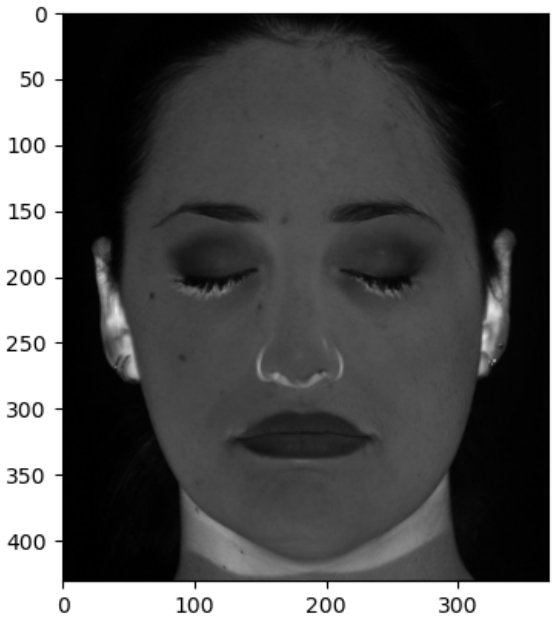


Figure 0.26: Albedo Output Image

Observing the figure above, one can point out the highlighted edges below the nose and namely at both ear canals of the subject in the image. On a minor note, one can also observe highlights around the person's eyelashes and chin too. Because the Albedo is in proportion to the Bidirection Reflectance Distribution Function, one can argue that the high values of BRDF in these images attribute to higher amount of surface radiance in these areas.

References

- N/A

Q4.2.4

Background N/A

Answer see the $displayAlbedosNormals$ function in the $submission.py$ file for code implementation to the question.

The figure shown below showcases the rainbor color map of the unit vector surface normals at each pixel in the image (see Figure 0.27):

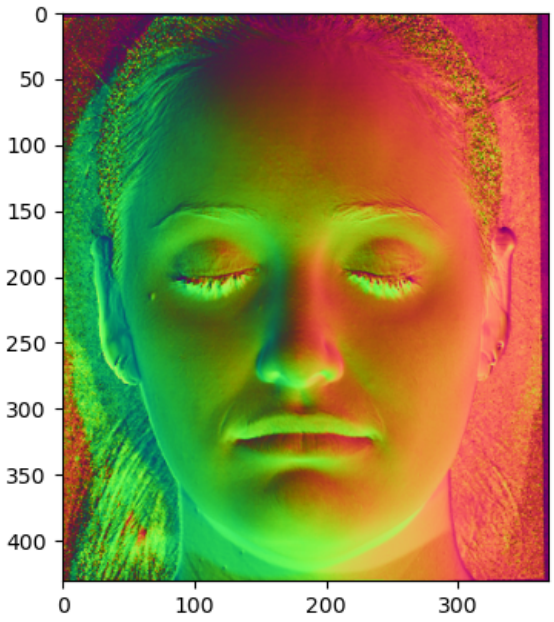


Figure 0.27: Albedo Output Image

Observing the image above, the normals do match the expectations at the curvatures of the face at the majors area such as the eyes, notes, mouth, and chin. In the hair areas of the image, the estimated curvatures were not as expected as there exists very spotty regions around the hair. Because the texture of hair is not considered a smooth surface, different lighting angles could have given varying differences in surface normal values, thus showcasing a region with many ”spots” in the image.

References

- N/A