16-720 Computer Vision: Homework 4
3D Reconstruction

Anbang Hu
anbangh@andrew.cmu.edu
Due: March 24, 2016

Part I

Theory

Q1.1 **(5 points)** *Proof.* If the image coordinates are normalized so that the coordinate origin $(0,0)$ coincides with the principal point, all points along principal axes are projected to $(0,0)$, i.e. $(0,0,z)$ in 3D coordinate system is projected to $(0,0)$ in image coordinate system for $\forall z$. Then we have

$$\begin{bmatrix} 0 & 0 & z_2 \end{bmatrix} \mathbf{F} \begin{bmatrix} 0 \\ 0 \\ z_1 \end{bmatrix} = 0, \quad \forall z_1, z_2$$

$$\Rightarrow \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} & \mathbf{F}_{13} \\ \mathbf{F}_{21} & \mathbf{F}_{22} & \mathbf{F}_{23} \\ \mathbf{F}_{31} & \mathbf{F}_{32} & \mathbf{F}_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

Solving the above equation, we find that $\mathbf{F}_{33} = 0$. □

Q1.2 **(10 points)** [1] *Proof.* Assume that two cameras have the same intrinsic matrix $\mathbf{K}$ and common camera center $\mathbf{C}$ and also focal length f in intrinsic matrix is 1. Let $\mathbf{P}_l$, $\mathbf{P}_r$ be projection matrices for left and right cameras, respectively. Then we have

$$\mathbf{P}_l = \mathbf{K}\left[\mathbf{I}|\mathbf{0}\right], \quad \mathbf{P}_r = \mathbf{K}\left[\mathbf{I}|\mathbf{T}\right], \quad \mathbf{P}_l^\star = \begin{bmatrix} \mathbf{K}^{-1} \\ \mathbf{0}^T \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} T_x \\ \mathbf{0} \end{bmatrix} \tag{1}$$

Using Eq(1), we can write fundamental matrix $\mathbf{F}$ as follows:

$$\mathbf{F} = \left[\mathbf{P}_r\mathbf{C}\right]_\times \mathbf{P}_r\mathbf{P}_l^\star = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T_x \\ 0 & T_x & 0 \end{bmatrix} \tag{2}$$

where we use $[\mathbf{A}]_\times\mathbf{B}$ to denote $\mathbf{A}\times\mathbf{B}$.

Eq(2) allows us to write out epipolar line in right image that corresponds to $\mathbf{x}_l = (x_l, y_l, 1)^T$ in left image: (the coordinate is in homogeneous form)

$$\ell_r = \mathbf{F}\mathbf{x}_l = \begin{bmatrix} 0 \\ -T_x \\ y_l T_x \end{bmatrix} \tag{3}$$

We see that Eq(3) shows that the x-dimension of the epipolar line has value 0. So the epipolar line is parallel to x-axis.

Similarly, we can show that epipolar line in left image that corresponds to point $\mathbf{x}_r$ in right image is also parallel to x-axis. □

Q1.3 **(10 points)** *Proof.* We consider the model in Figure 1. We choose the intersection of the line connecting two cameras and mirror plane (i.e. yz-plane), denoting it by O. C_1 is the real camera and C_2 is the mirror

[1] I read http://www.robots.ox.ac.uk/~vgg/hzbook/hzbook1/HZepipolar.pdf for reference.

image of camera C_1. Let the distance between C_1 and C_2 be t_x. The rotation-translation matrix is $[\mathbf{R}|\mathbf{T}]$ with

$$\mathbf{R} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} T_x \\ 0 \\ 0 \end{bmatrix}$$

Since C_2 is the mirror image of C_1, we know that they have the identical intrinsic matrix $\mathbf{K}$. So the fundamental matrix $\mathbf{F}$ can be written as

$$\mathbf{F} = \mathbf{K}^{-T}\mathbf{E}\mathbf{K}^{-1}$$

where $\mathbf{E}$ is the essential matrix

$$\mathbf{E} = [\mathbf{T}]_\times \mathbf{R} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T_x \\ 0 & T_x & 0 \end{bmatrix}$$

We know that $\mathbf{E}^T = -\mathbf{E}$. Then we have

$$\mathbf{F}^T = \mathbf{K}^{-T}\mathbf{E}^T\mathbf{K}^{-1} = -\mathbf{K}^{-T}\mathbf{E}\mathbf{K}^{-1} = -\mathbf{F}$$

Therefore, the situation is equivalent to having two images of the object which are related by a skew-symmetric fundamental matrix. □

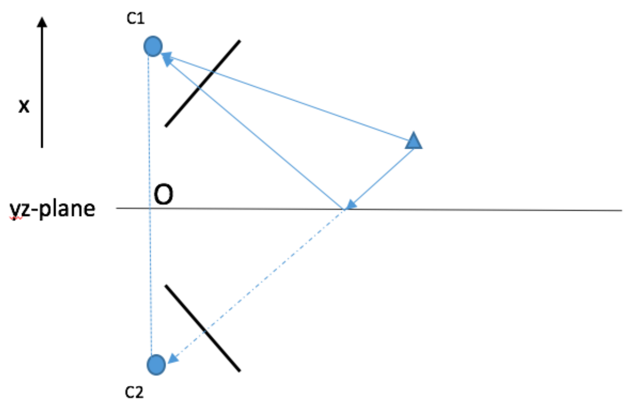

Figure 1: Mirror image of camera view

Part II

Practice

1 Overview

Nothing to be done here.

2 Fundamental matrix estimation

The Eight Point Algorithm

Q2.1 **(10 points)** [2] I use provided correspondences in `data/some_corresp.mat` in this problem. The implementation is in `matlab/eightpoint.m`. The required variables are saved in `matlab/q2_1.mat`. The recovered **F** is

$$\mathbf{F} = \begin{bmatrix} -0.000000001444392 & -0.000000078626884 & 0.001132688437909 \\ -0.000000112046048 & 0.000000001261817 & 0.000004154313196 \\ -0.001088103533379 & 0.000015377311155 & -0.004642313040742 \end{bmatrix}$$

An example output of `displayEpipolarF` is shown in Figure 2. It is generated by script `matlab/testQ2_1.m`.

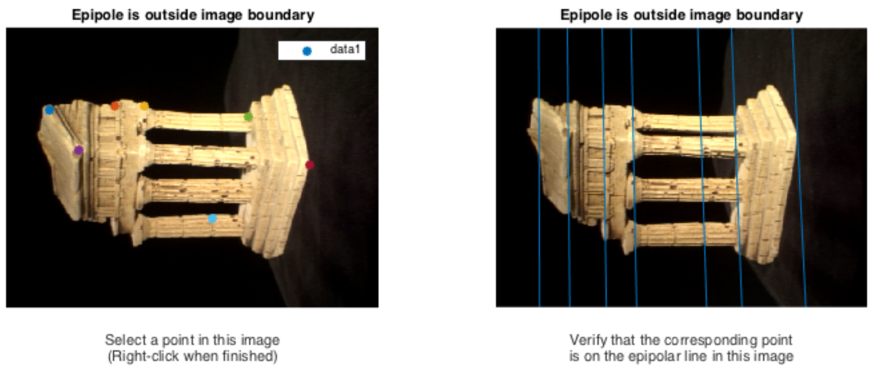
Epipole is outside image boundary
● data1
Select a point in this image
(Right-click when finished)
Epipole is outside image boundary
Verify that the corresponding point
is on the epipolar line in this image

Figure 2: Example output of `displayEpipolarF` from eight point algorithm

[2] In implementing 8-point algorithm, I consulted Richard I. Hartley's In Defence of the 8-point Algorithm. http://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/f07/proj_final/www/amichals/fundamental.pdf

The Seven Point Algorithm

Q2.2 **(20 points)** [3] In this question, I choose to use the first 7 correspondences in `pts1`, `pts2`, because I cannot get good correspondence by manual selection via `cpselect` function, and the seven correspondences of my choices are relatively good (see Figure 3,4). The recovered fundamental matrix **F** is

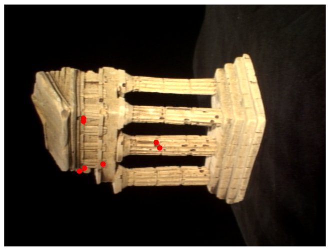

Figure 3: First seven correspondences in image 1 (red dots)

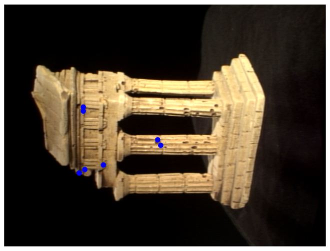

Figure 4: First seven correspondences in image 2 (blue dots)

$$\mathbf{F} = \begin{bmatrix} 0.000000000316646 & -0.000000084252617 & 0.001145769310932 \\ -0.000000109062589 & 0.000000001430837 & 0.000002927963112 \\ -0.001101566773665 & 0.000016965829185 & -0.004615894102812 \end{bmatrix}$$

An example output of `displayEpipolarF` from seven point algorithm is shown in Figure 5. It is generated by script `matlab/testQ2_2.m`. The required variables are saved in `matlab/q2_2.mat`.

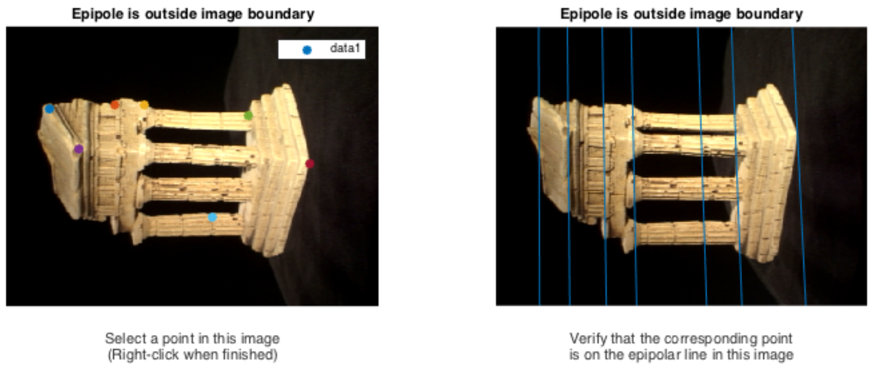
Epipole is outside image boundary
data1
Select a point in this image
(Right-click when finished)
Epipole is outside image boundary
Verify that the corresponding point
is on the epipolar line in this image

Figure 5: Example output of `displayEpipolarF` from seven point algorithm

[3]In implementing 7-point algorithm, I consulted slides https://staff.fnwi.uva.nl/l.dorst/hz/chap11_13.pdf

Extra credit: Automatic Computation of F

Q2X **(10 points)** The implementation is in `matlab/ransacF.m`. I followed the main idea of RANSAC described in slide 43, 44 of lecture notes `http://16720.courses.cs.cmu.edu/lec/two-view2.pdf`. The error metric I use is the scaled distance from a point in one image to the epipolar line defined by the correspondence point in the other image. When the error is smaller than a threshold, which I choose to be 0.001, this pair of correspondence is an inlier. Instead of accumulating the distances and minimizing it, I count the number of inliers and keep the best fundamental matrix computed using seven point algorithm so far (that has largest number of inliers over the entire set of correspondences). The number of iterations is computed using the method in slide 44 of `http://16720.courses.cs.cmu.edu/lec/two-view2.pdf`:

$$k = \lceil \frac{\log(1-p)}{\log(1-w^7)} \rceil$$

where the target p is set as $1 - 1 \times 10^{-12}$ so that the probability of finding a good model is more than 99.9999%. $w = 0.75$ since the approximate portion of inliers is 0.75. So $k = 193$.

The recovered $\mathbf{F}_{\text{RANSAC}}$ using seven point algorithm with RANSAC is

$$\mathbf{F}_{RANSAC} = \begin{bmatrix} 0.000000020099745 & -0.000000653099459 & -0.000887126578879 \\ 0.000000760940072 & 0.000000010391565 & -0.000133750836839 \\ 0.000855605287036 & 0.000124585485943 & 0.002292265617716 \end{bmatrix}$$

The recovered $\mathbf{F}_{\text{eight}}$ using eight point algorithm is

$$\mathbf{F}_{eight} = \begin{bmatrix} 0.000000493501351 & -0.000000907126844 & 0.000043710170543 \\ 0.000001417512586 & -0.000001492512888 & 0.000019784613230 \\ -0.000392342728234 & 0.000511921132772 & -0.014866033460154 \end{bmatrix}$$

An example output of `displayEpipolarF` from seven point algorithm with RANSAC is shown in Figure 6. An example output of `displayEpipolarF` from eight point algorithm is shown in Figure 7. We can see that due to noise, eight point algorithm does not perform well, whereas seven point algorithm with RANSAC gives reasonable epipolar lines. They are generated from script `matlab/testQ2_X.m`.

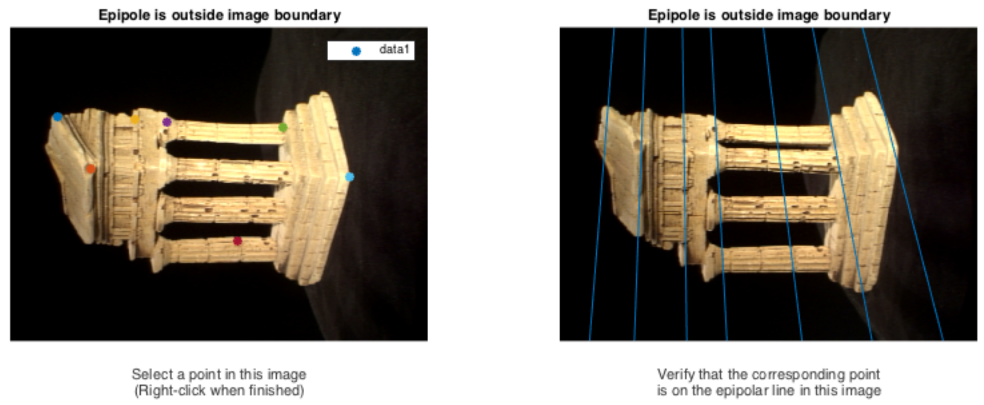
Epipole is outside image boundary
data1
Select a point in this image
(Right-click when finished)
Epipole is outside image boundary
Verify that the corresponding point
is on the epipolar line in this image

Figure 6: An example output of `displayEpipolarF` from seven point algorithm with RANSAC on noisy correspondences

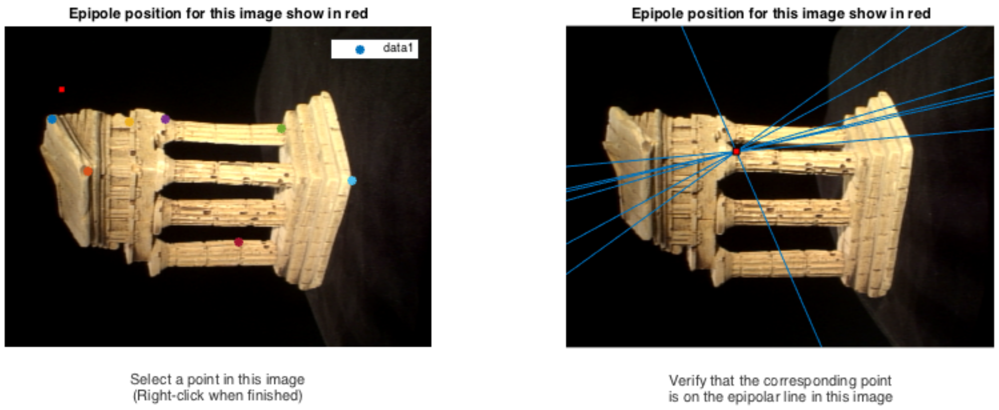
Epipole position for this image show in red
data1
Select a point in this image
(Right-click when finished)
Epipole position for this image show in red
Verify that the corresponding point
is on the epipolar line in this image

Figure 7: An example output of `displayEpipolarF` from right point algorithm on noisy correspondences

3 Metric Reconstruction

Q2.3 **(5 points)** Implemented in `matlab/essentialMatrix.m`, `matlab/testQ2_3.m`. Estimated $\mathbf{E}$ using $\mathbf{F}$ from eight point algorithm is

$$\mathbf{E} = \begin{bmatrix} -0.003338879242196 & -0.182412669998165 & 1.691963685200263 \\ -0.259944407570236 & 0.002937977877869 & -0.044873580814653 \\ -1.697072015373238 & -0.012331810687479 & -0.000627378815987 \end{bmatrix}$$

Q2.4 **(10 points)** I choose to implement linear triangulation (algebraic solution) in `http://www.ics.uci.edu/~dramanan/teaching/cs217_spring09/lec/stereo.pdf`, which is amenable to Matlab implementation. The implementation is in `matlab/triangulate.m`.

Q2.5 **(10 points)** The implementation is in `matlab/findM2.m`. I find the best M_2 by choosing the one with smallest reprojection error and all estimated 3D points having z value nonnegative. The required variables are saved in `matlab/q2_5.mat`.

4 3D Reconstruction

Q2.6 **(10 points)** Implementation is in `matlab/epipolarCorrespondence.m`. The required variables are saved in `matlab/q2_6.mat`. Figure 8 shows the corresponding points that I find.

My method for finding epipolar correspondences is as follows:

1. Define a Gaussian filter of size 25×25 and $\sigma = 4$. This filter is used to weight the pixel intensities that are in turn used to compute Euclidean distances.
2. For a given point (x_1, y_1) in image 1, I compute the epipolar line $ax_1 + by_1 + c = 0$ in image 2. Instead of searching best 25×25 patch along the entire epipolar line, I only search over a 101 pixel range centered at $(\frac{-by_1 - c}{a}, y_1)$ in image 2. The reason behind is that the two images only differ by a small amount.
3. The error between the target patch centered at (x_1, y_1) in image 1 and the proposed patch along the epipolar line in image 2 is computed as the sum of Euclidean squared weighted by aforementioned Gaussian filter.
4. The corresponding point is chosen as the one in the search range with smallest error.

Q2.7 **(10 points)** Implementation is in `matlab/q2_7.m`. The required variables are saved in `matlab/q2_7.mat`. Figure 9,10,11,12 show the outline of the temple from different angles.

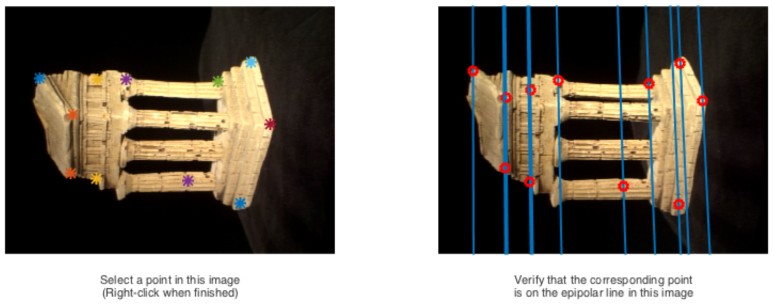
Select a point in this image
(Right-click when finished)
Verify that the corresponding point
is on the epipolar line in this image

Figure 8: `epipolarMatchGUI` shows the corresponding point found by calling `epipolarCorrespondence`

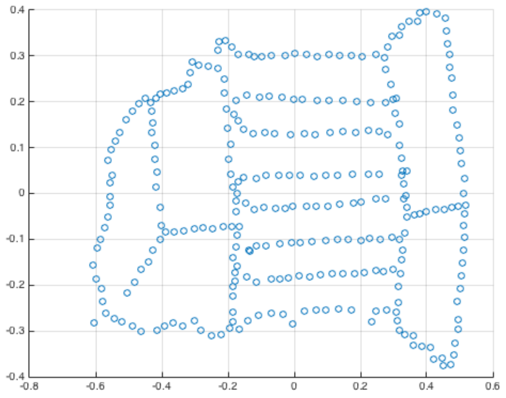

Figure 9: Outline of temple (xy view)

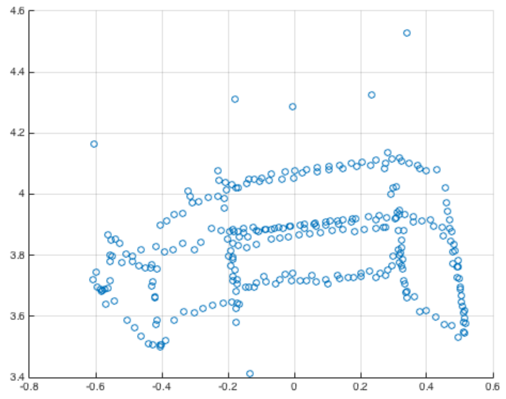

Figure 10: Outline of temple (xz view)

5 Extra Credit: Awesome Visualization

Q2.8 **(upto 10 points)** Implemented in `custom/q2_8.m`. Figure 14,15,16,17 show 3D visualization of point cloud of the temple.

My work in this extra credit problem is an extension to the work in previous problems. I change the type of images to `double` and create a `gray` version of images. Then I filter out and retain those points with value

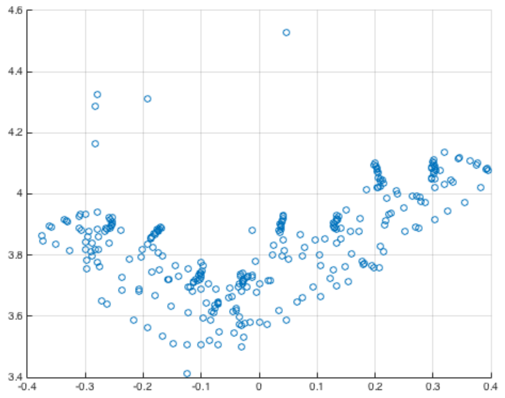

Figure 11: Outline of temple (yz view)

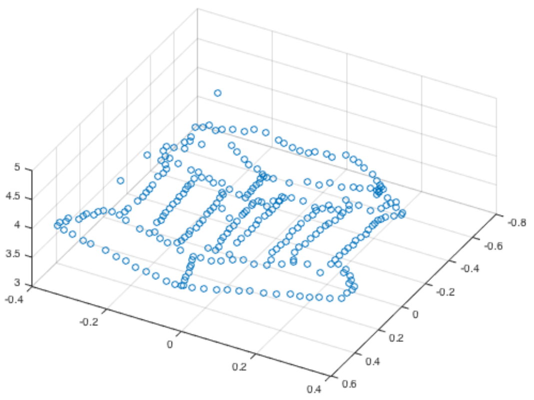

Figure 12: Outline of temple (random side view)

strictly greater than 0.21. This basically gives as temple as an object (see Figure 13). Taking these retained points as x_1, y_1 in image 1, I compute the corresponding (x_2, y_2) in image 2 using fundamental matrix $\mathbf{F}$ from Q2.1. by triangulate all these correspondences, I get a set of kind of dense 3D point cloud (actually not so dense). I visualize these 3D points with filled circles with corresponding colors from image 1 points. I tried to use the average of colors of correspondences, but it was too bright to see clearly.

Admittedly, more work can be done to achieve better 3D visualization of the temple by interpolating points to form a surface.

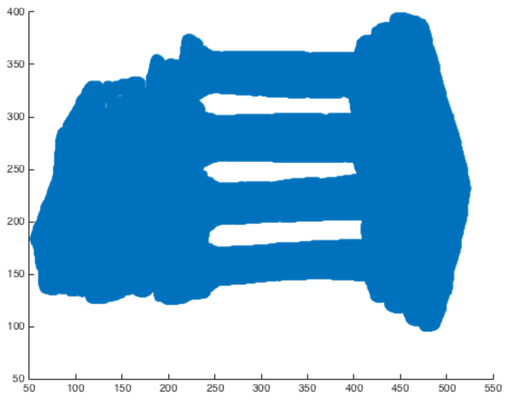

Figure 13: Temple object

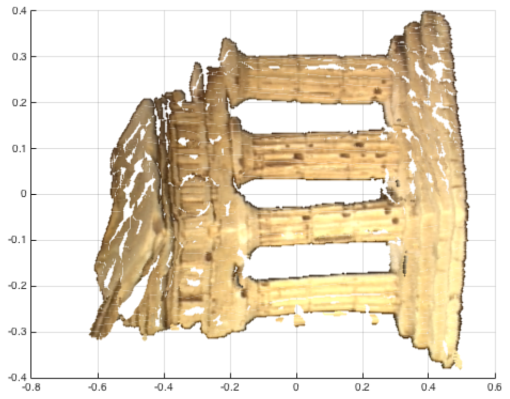

Figure 14: 3D visualization of temple point cloud (xy view)

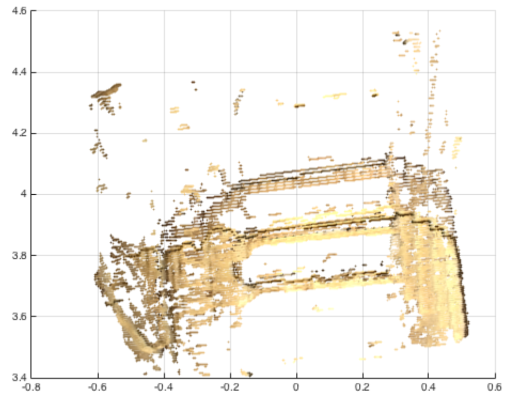

Figure 15: 3D visualization of temple point cloud (xz view)

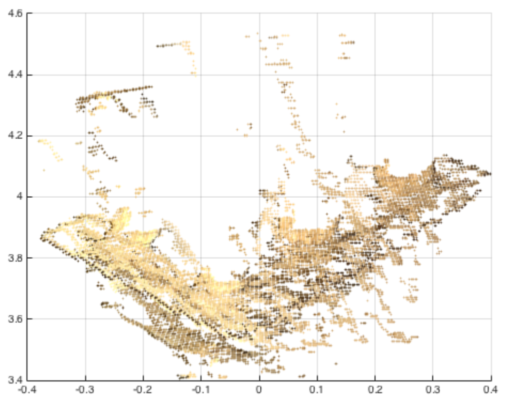

Figure 16: 3D visualization of temple point cloud (yz view)

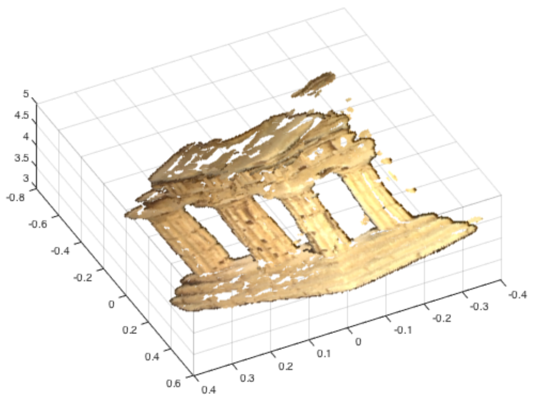

Figure 17: 3D visualization of temple point cloud (random side view)