Computational Photography and Capture Project (Part 1):

Structured Light Depth Acquisition

Section 1: Synthetic data sets

1.1. 3D Reconstruction

Light patterns decoding

In this section, I used images included in synthetic data folders to decode the light pattern for corresponded model. The decoding procedure could be summarized as below steps.

- Load colored image sequence and convert them into gray images.
- To get *(u, v)* code for each pixel based on input images, two binary vectors with 10 elements were computed for each pixel by recording frequency of white or black color on current position across 10 images for *u* element and 10 images for *v* element. Each element is determined be a pair of images which are exposed under inverse light patterns. If the pixel at certain position has higher intensity in the first image than that in the second image with inverted light patterns, the corresponded element will be recorded as 1 in binary vector, otherwise it will be set as 0. Finally, convert binary vectors to two decimal numbers as *(u, v)* code for that particular position within image.

The specific algorithm could be seen as *'reconstruction/get_pix_code.m'* and 'reconstruction/*get_code_uv.m'*, which decode *(u, v)* code for an image and a particular pixel respectively and all (u, v) codes mentioned in this report were saved in folder *'uv_code*'. An example *(u,v)* code could be seen as below.

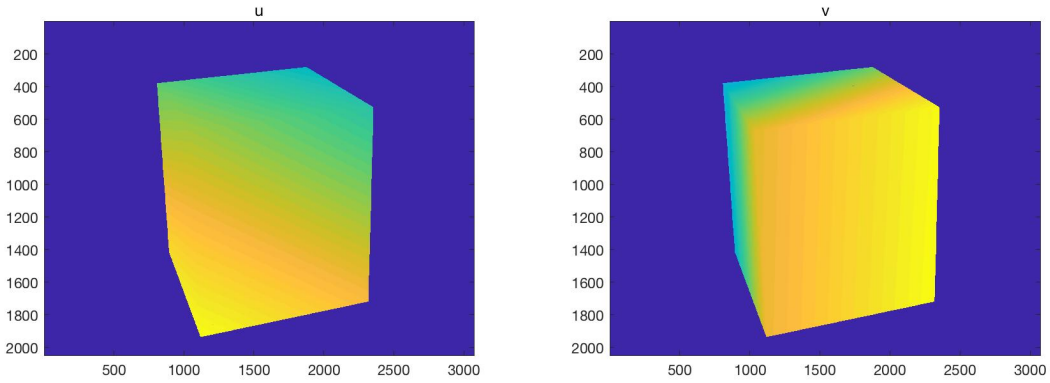


Figure 1: (u, v) code for cube_T1

Unreliable pixels elimination

In order to make sure each computed (u, v) code is relevant to the model to be reconstructed, I computed differences between two successive images with invert light patterns for each pixel. The result of summing 10 elements of both obtained binary vectors for *u* and *v* was used to evaluate if the particular code is relevant. If

the result is smaller than threshold, it means that it has less relevance to the wanted model so it will be set as zero. The particular threshold is different for different situations hence it should be adjusted based on actual situation (it was set as 32 in given paper).

The code for unreliable pixels elimination was integrated into *'reconstrucntion/get_pix_code.m'*.

Get unique depth map and point clouds

From above steps, we have already obtained the corresponded pair between the projector and camera point $\left(\binom{u}{v}, \binom{x}{y}\right)$. 3D points could be triangulated by making use of corresponded pair and intrinsic and extrinsic matrices of projector and camera to minimize the squared distance as below,

$$\hat{\omega} = arg\min_{\omega}\left(\sum_{j=1}^{J}\left(\mathrm{x_j} - pinhole\left[\omega, \Lambda_j, \Omega_j, \tau_j\right]\right)^{\mathrm{T}}\left(x_j - pinhole[\omega, \Lambda_j, \Omega_j, \tau_j]\right)\right)$$

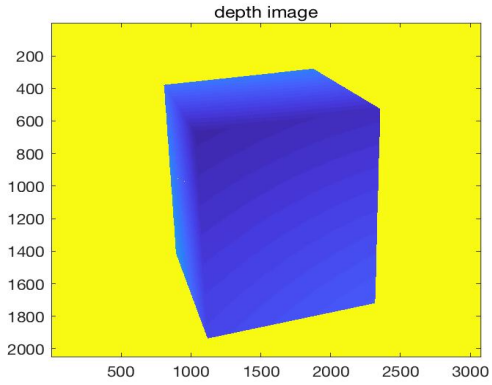
This equation could be solved by arranging it into the form like $Ax = b$ by using methods described in Simon Prince algorithm and solve the obtained linear system by applying least square estimation. Then we got the corresponded 3D point cloud and it could be projected by multiplying extrinsic matrix of camera. The depth of each point was saved as depth map and point cloud was saved as PLY file by using function *'saveas_ply.m'* to be visualized using MeshLab software.

The algorithm could be ran by using function *'reconstruction3D(folder, img_name, first, digit, format)'*, where *'folder'* represents name folder path that given image set exists, *'first'* and *'digit'* represents the first number and digit number of image names respectively. In this task, they should be *0* and *4*. The last parameter *'format'* is the file format of image set.

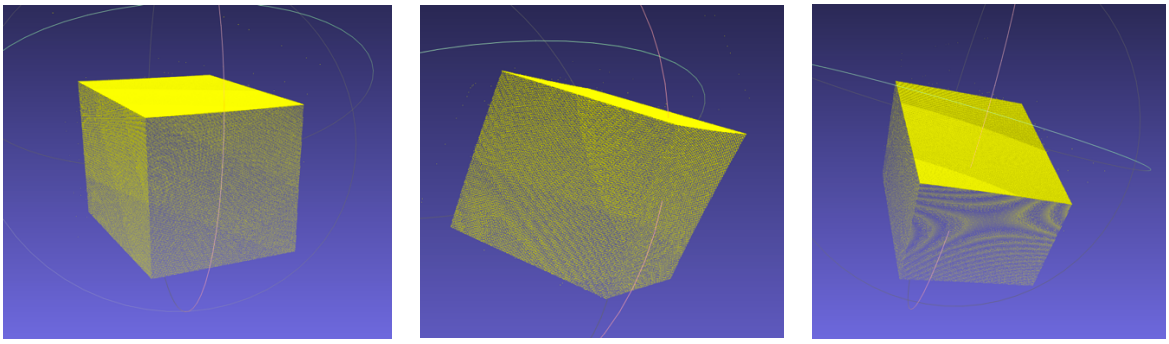
Reconstruction results for provided data set

We got some satisfying results for synthetic image data set. Points clouds are almost reconstructed with great details and less noise. To view the reconstruction model completely, different view of each point clouds will be shown below. Additionally, the corresponded depth map will be display as well.

In the aspect of depth map, the darker color represents smaller depth, i.e. closer distance to the camera. From these reconstruction scenes, we could be found that some perspective distortions seem to occur in projector and camera calibration. Most of dark surface cannot be reconstruct successfully and they are always shown as holes in the reconstruction scenes even for well reconstruction performance.

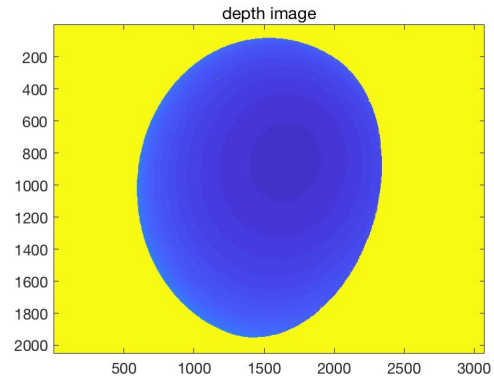


(a) Depth map

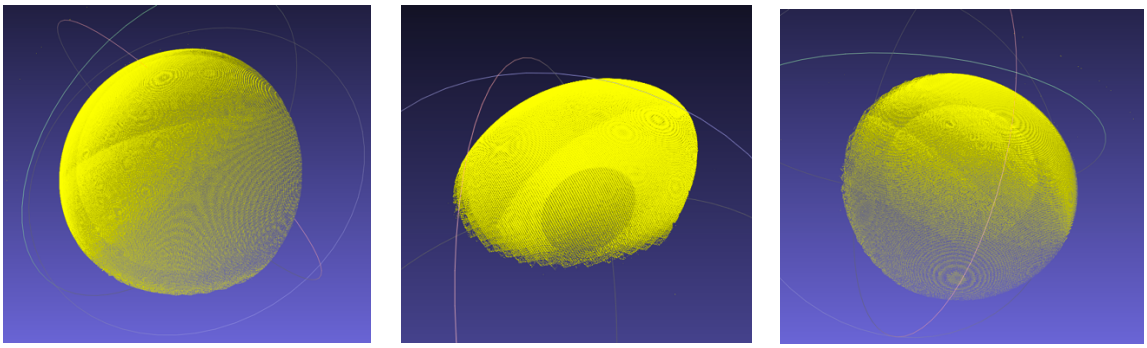


(b) Different views of scene

Figure 2: cube_T1

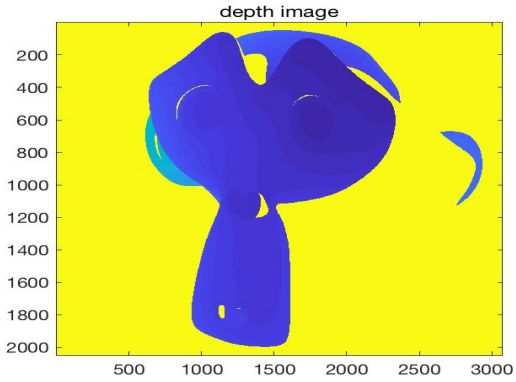


(a) Depth map

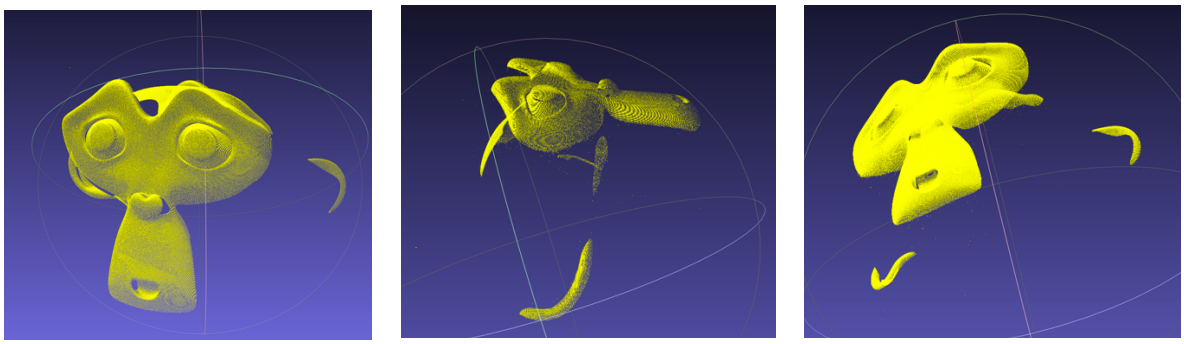


(b) Different views

Figure 3: sphere_T1

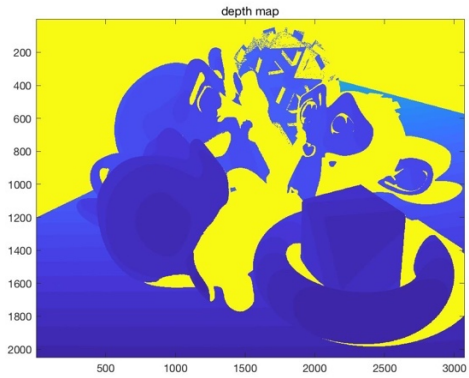


(a) Depth map

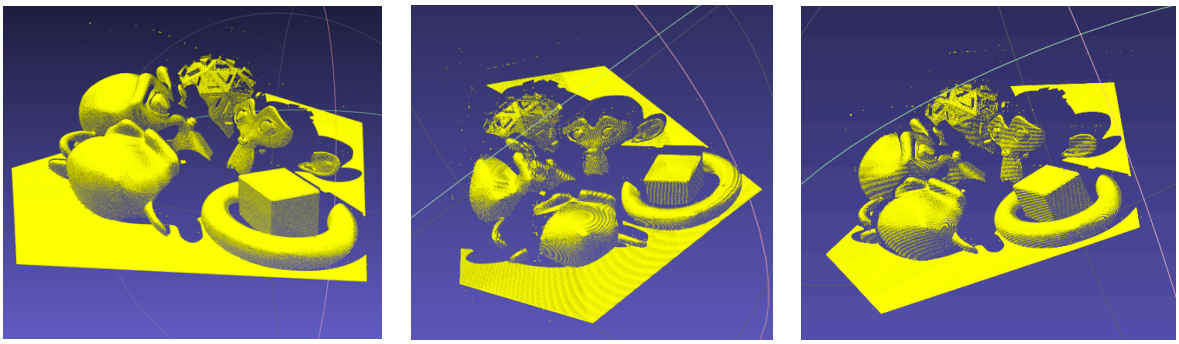


(b) Different views

Figure 4: monkey_T1

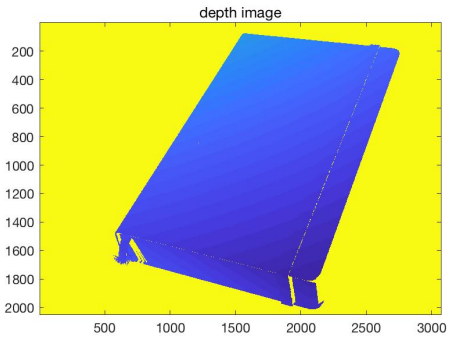


(a) Depth map

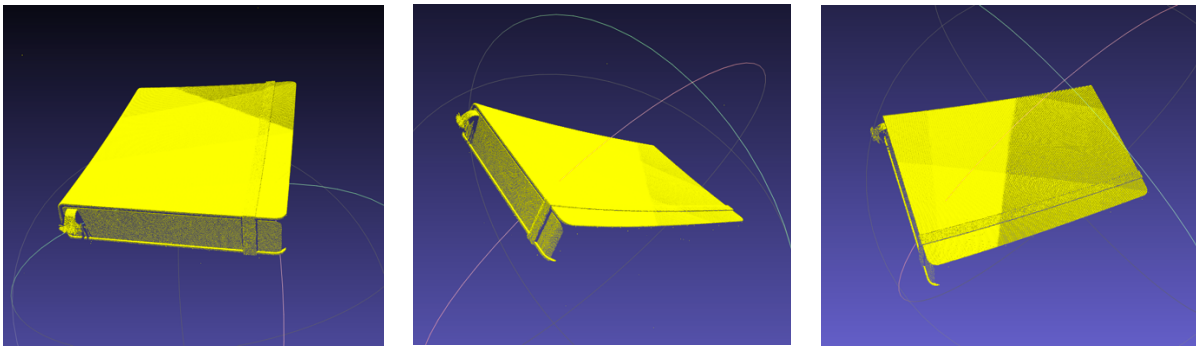


(b) Different views

Figure 5: red_T1

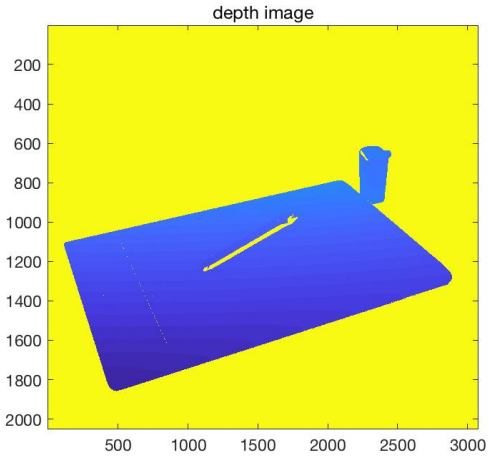


(a) Depth map

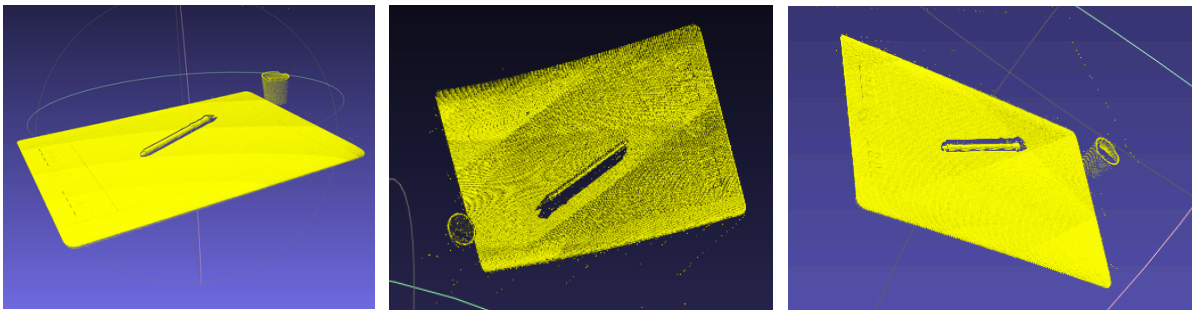


(b) Different views

Figure 6: notebook_T1



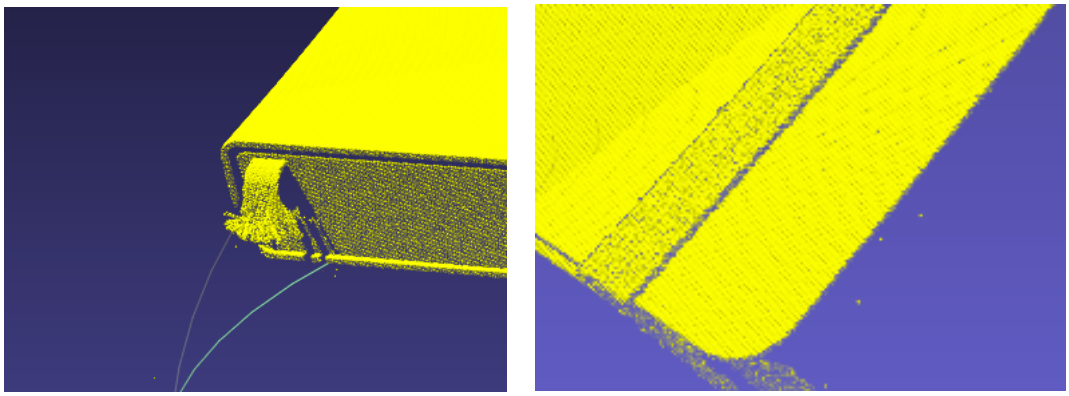
(a) Depth map



(b) Different views

Figure 7: tablet_T1

Some reconstruction works show very well performance, such as ‘notebook_T1’, we even could see the hair details and belt texture clearly.



(a) Hair details. *(b) Belt texture*

Figure 8: Reconstruction details in ‘notebook_T1’

1.2. Camera calibration

In this part, it is required to calculate the Intrinsic Λ and Extrinsic matrices Ω for both camera and projector.

Reprojecting images for projector

The approaches used to reproject provided images was introduced in *Projector calibration for 3D scanning using virtual target image* (by Hafeez Anwar, Irfanud Din and Kang Park). The provided images are ‘seen’ by camera, not projector, hence we need to reproject them to obtain the view ‘seen’ by projector. To achieve the image transformation, a homography matrix should be calculated by solving a least-square linear system based on reference coordinates, such as projector resolution, start origin and size of chequerboard. Next, apply the homography matrix on unprojected images (‘seen’ by camera) to reproject them. An example of reprojection could be seen as Figure 9. The first column shows the original images ‘seen’ by camera and the second column shows reprojected chequerboard and its rectified version that obtained by applying homography matrix on actual reference chequerboard.

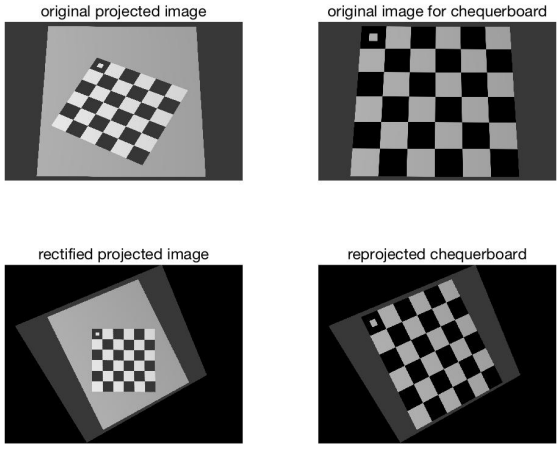


Figure 9: Reprojection

The reprojection was coded as function *'Calibration.m'*(run *Calibration(true, false, false)*) in folder *'calibration'*. Additionally, the resulted images for synthetic data sets were saved under *'calibration/synthetic_calibration'* folder.

Intrinsic and Extrinsic matrices estimation

The Intrinsic and Extrinsic matrices of camera and projector could be obtained by using *'calib_gui'* tool box we learned in lab 2. Estimated matrices were saved in *'load_synthetic_calibration.m' in 'reconstruction' folder.*

In the aspect of Intrinsic matrix estimation, he data set we used for estimation for camera and projector is shown as below respectively.

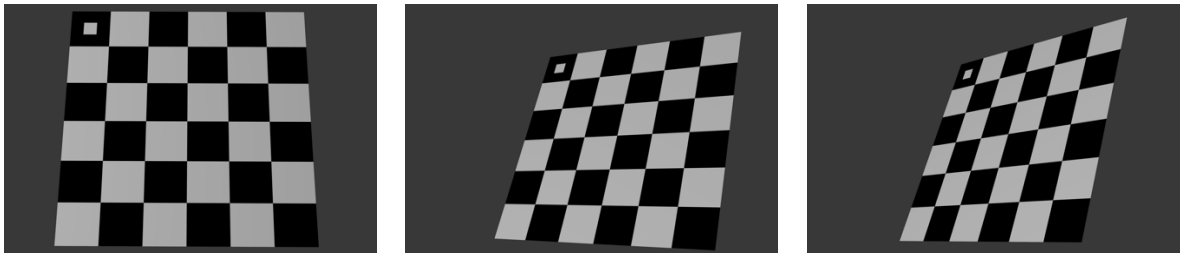


Figure 10: Data set used for camera matrices estimation

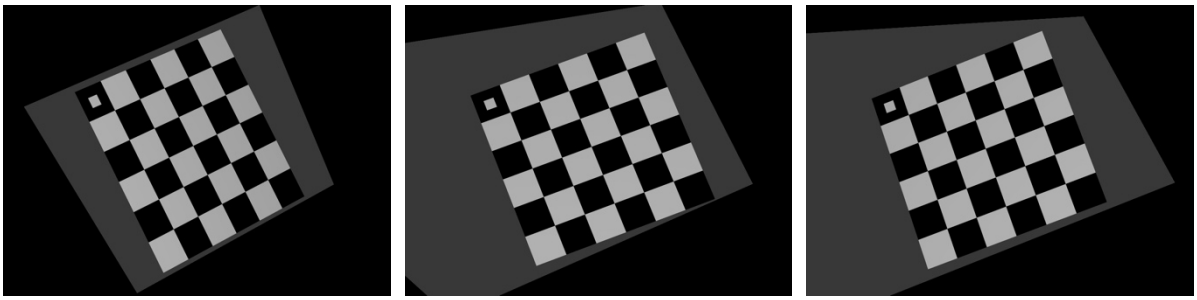


Figure 11: Data set used for projector matrices estimation

In the aspect of Extrinsic matrix estimation, we need to use pair of data set that illustrates the view to the same chequerboard from the camera and projector to make sure the same world origin. I selected the first pair of data set and use the function *'comp.ex'* included in tool box window to estimate Extrinsic matrices of both camera and projector. The corresponded 3D world coordinates for them could be seen as Figure 11.

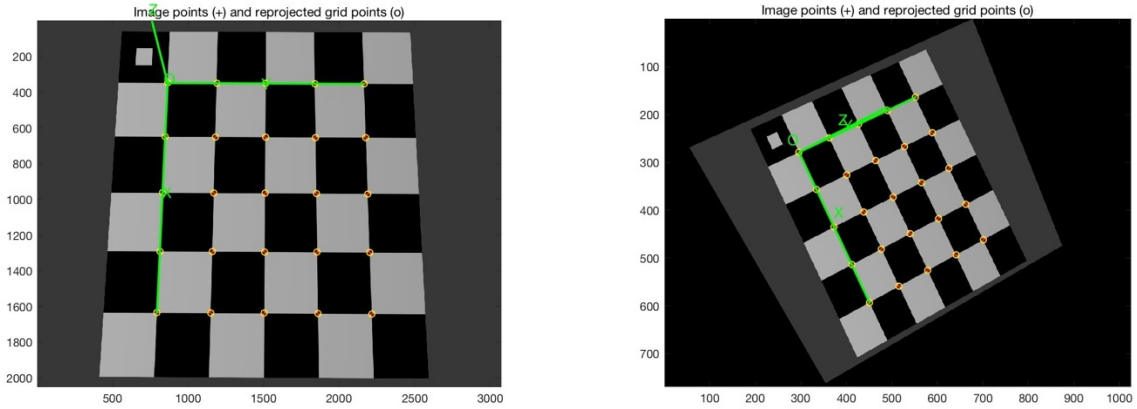


Figure 12: World coordinates for camera (left) and projector (right)

Reconstruction results

Based on estimated matrices, I obtained the correct reconstructed 3D model but with much smaller scale. After rescaling them in MeshLab software, the comparison between the reconstruction scenes that got based on given and estimated calibration could be seen as below, where the red one is based on estimated calibration and the yellow on is based on given calibration.

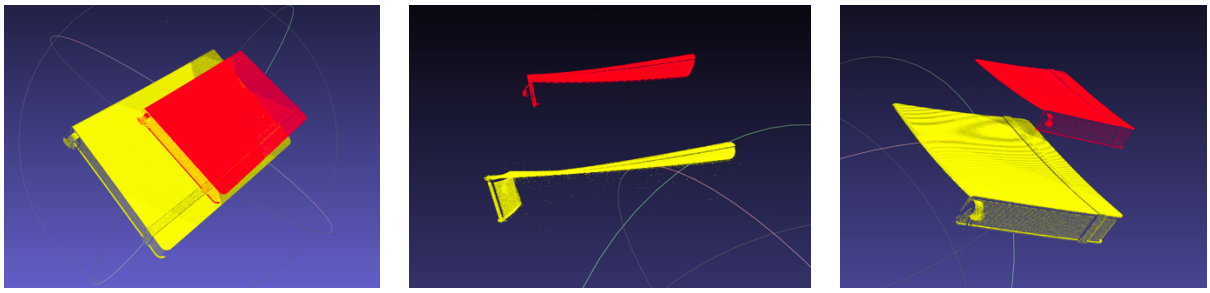


Figure 13: Comparisons between meshes obtained based on given calibration (yellow) and own estimated calibration (red)

From above image set, it could be found that two reconstructed scenes show similar structure but with some little distortions, which is very clear in the right view (middle image of Figure 13).

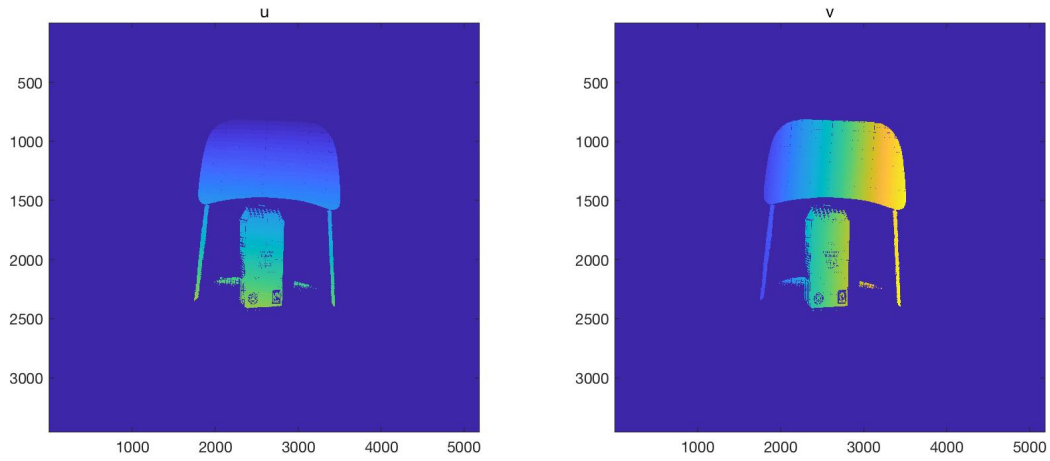
There are two possible reasons for this in my opinion. Firstly, calibration have a significantly influence on both perspective and global geometry of reconstructed point cloud. It means that if camera and projector do not calibrate with the same world origin well, some distortions will occur in the reconstruction. This kind of errors may be caused by click offsets during 'grid corner extraction' when working with '*calib_gui'* tool box.

Another possible reason is from reprojection. In the algorithm of homography estimation, we need to select 4 corners of the chequerboard. Similar to things described above, small selection offsets will affect final homography estimation. The ideal rectified version of reprojection should match the original chequerboard to be projected completely, but small offsets will result to some distortions in reprojected images.

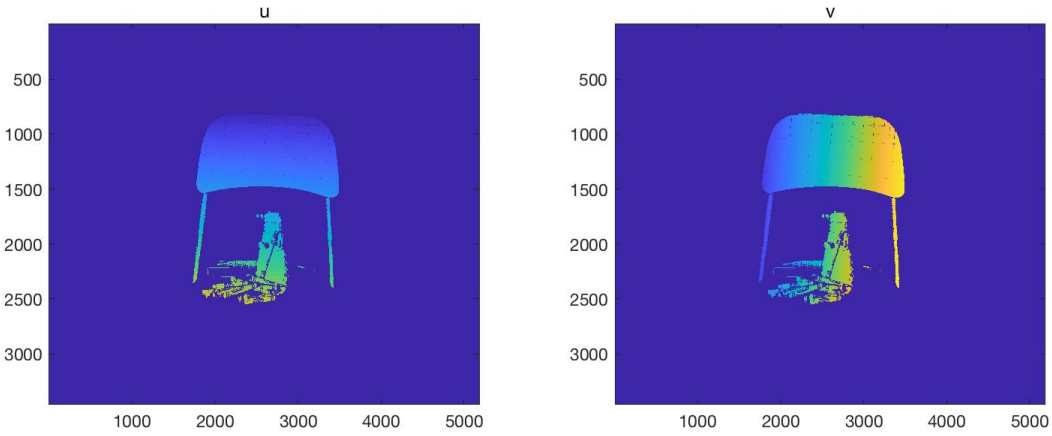
Section 2: Provided real data sets

Light patterns decoding

The decoding process was achieved by using the same function *'get_pix_code'* in *'reconstruction'* folder as for the above section. It is obvious to find that the decoding process became slow as the large size of input image sequence *(u,v)* code for provided real data set *'real_tea'* and '*real_crayon_dalek'* could be seen as below.



(a) real_tea



(b) real_cryon_delak

Figure 14: (u,v) code for two provided real data sets

Projection matrices determination

To get the view 'seen' by projector, we need to reproject given images that reflect views 'seen' by camera. Different from separate images for camera and projector in synthetic section, each image includes a constant chequerboard printed pattern and a projected pattern on the wood board, hence we could transform each image directly without process them separately. By using implemented function *'Calibration(false, true, true)'*, the reprojection results could be obtained and Figure 15 shows one of examples. However, it could be found that some printed chequerboard becomes uncomplete after the transform and we need to use the same pattern to estimate matrices. Therefore, I used 3×3 pattern whose left top grid is at (4, 3) within the complete printed

chequerboard. The reprojection data set was stored in folder *'real_calibration'*.

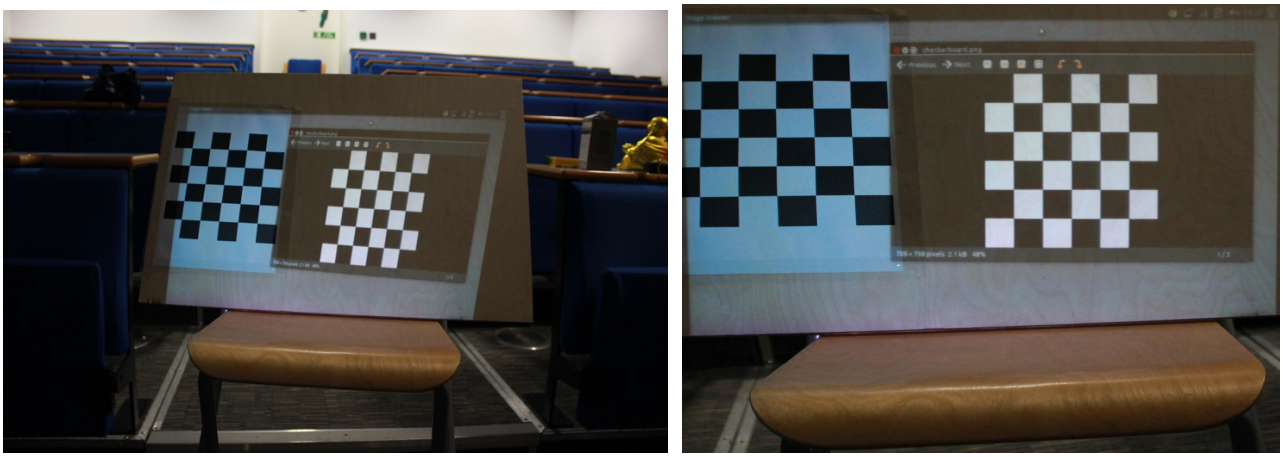


Figure 15: Views from camera (left) and projector (right)

By using function '*comp ex.'* in '*calib_gui'* tool box, the extrinsic matrix of both camera and projector could be obtained and I selected the first image '*IMG_9321.jpg'* to define the same world coordinate.

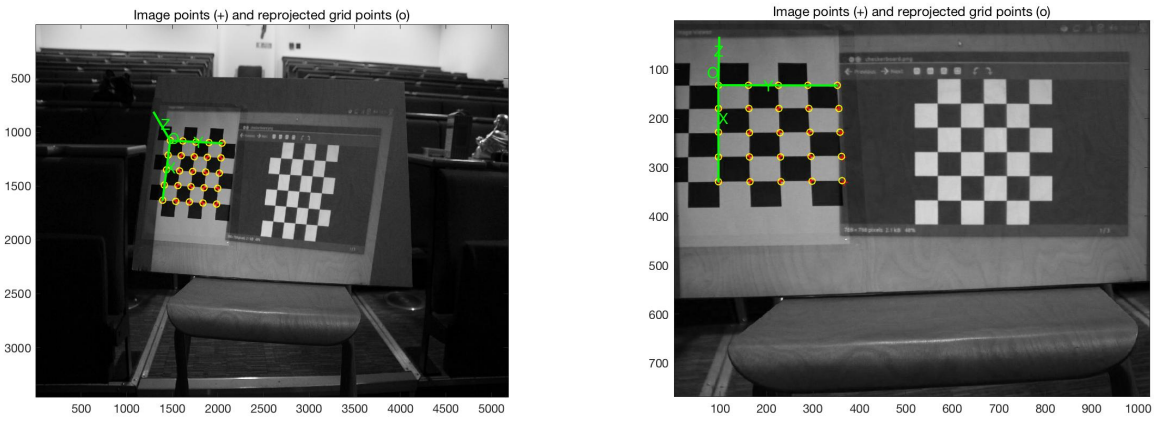


Figure 16: World coordinates for camera (left) and projector (right)

The obtained matrices were saved in *'load_real_calibration.m'* in *'reconstruction' folder.*

Unique depth map determination

In order to remove unreliable pixels, I set threshold to evaluated sum of the intensity differences between all gray patterns and their inverses. If the sum is lower than predefined threshold, it means the certain pixel might be noise and it will be eliminated; if the sum is higher than threshold, the pixel at the particular position will be remained as it is clear enough. The setup of threshold is important and they need to be high enough to eliminate most of noise, but too high number will influence wanted model structure. Therefore, I tested some different thresholds to find the one that could achieve better performance.

The final depth map for two provided data set could be seen as below.

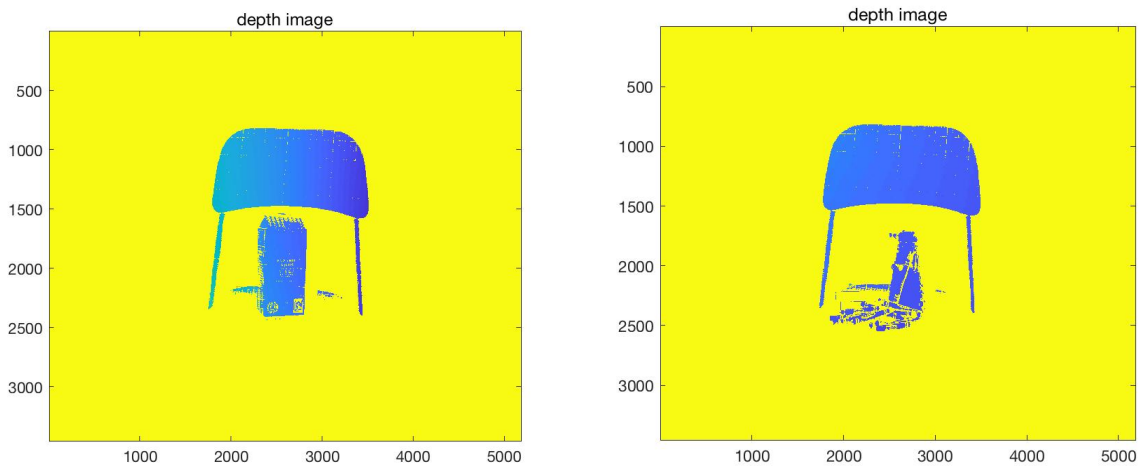


Figure 17: Depth maps for 'real_tea' (left) and 'real_crayon_delak' (right)

Reconstruction results for provided data set

The reconstruction results were stored as point clouds and visualized in MeshLab.

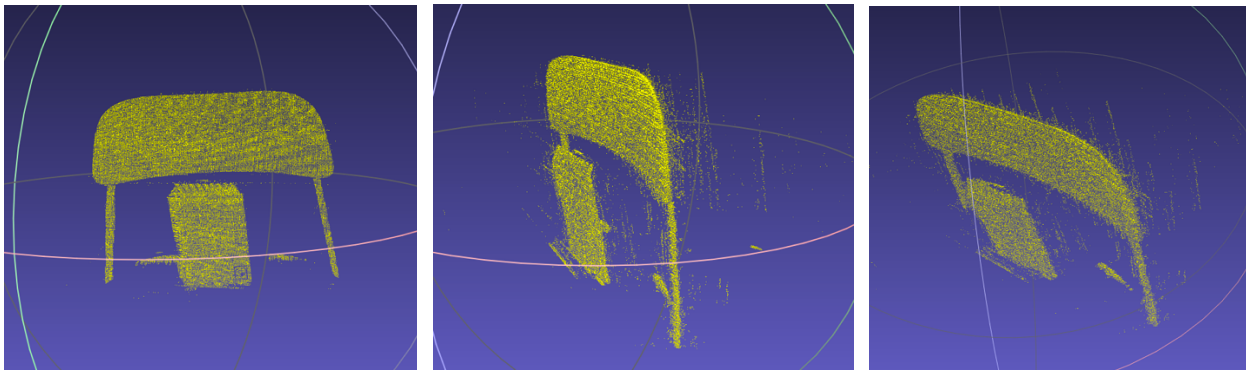


Figure 18: Different views for reconstruction of 'real_tea'

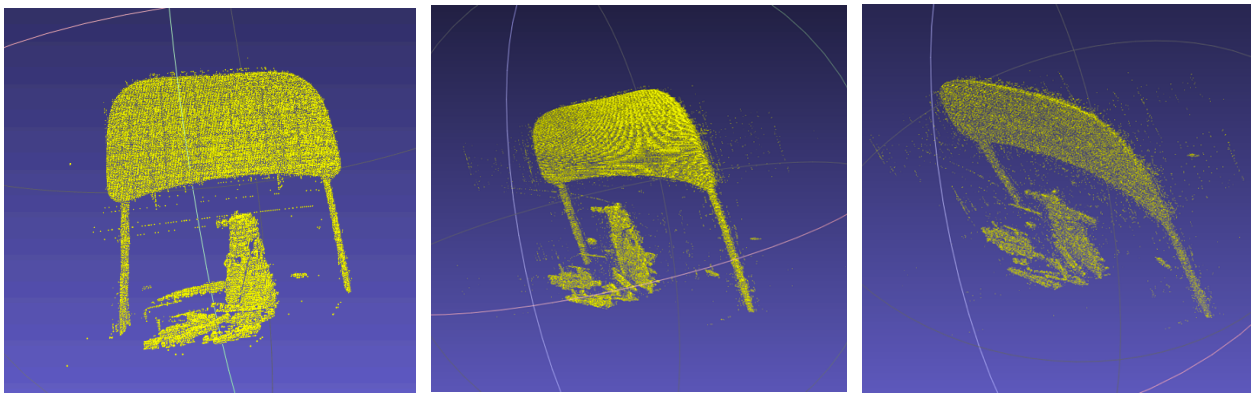


Figure 19: Different views for reconstruction of 'real_cryon_delak'

It could be found that some distortions exist though the reconstructed works shows the correct overall structure. They may result from reasons we described in last section. Additionally, some noises still exist even after eliminating unreliable pixels. It means the threshold might need to be adjusted slightly.

Section 3: Own data captured

In this section, I obtained data with Xinyi YU and Qifan SHU. We captured one scene which included a wood board and a which head model. The head model has a clear outline which is easy to recognize. We put the model in front of the wood board and overlap a part of view to see how our algorithm works on such situation.

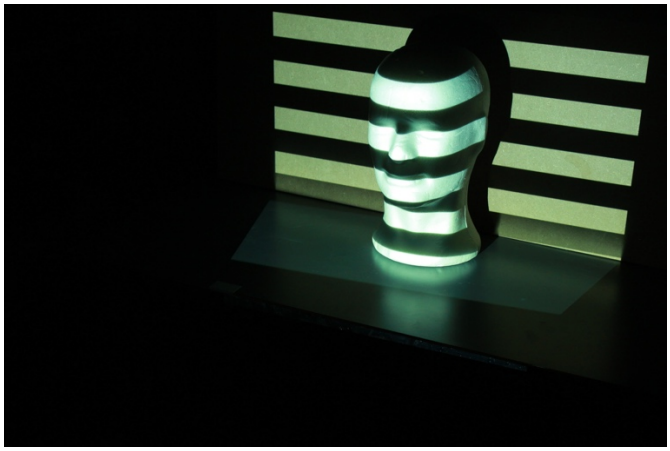


Figure 20: Captured head model

We turned off the auto-focus mode of the camera and set related parameters, such as aperture number and shutter speed, to take photos for objects. Captured images for calibration could be seen in folder *'calibration/own_calibration'* and data set of objects with light patterns could be found in folder *'calibration/own_face'*.

Determination of projection matrices

From gathered photos, I selected 6 of them to do the calibration. The process for calibration for them are similar to above sections. The reprojected images could be obtained by run '*Calibration(false, false, true)*' in Matlab. We need to do reprojection firstly and one of reprojection set could be seen as below.

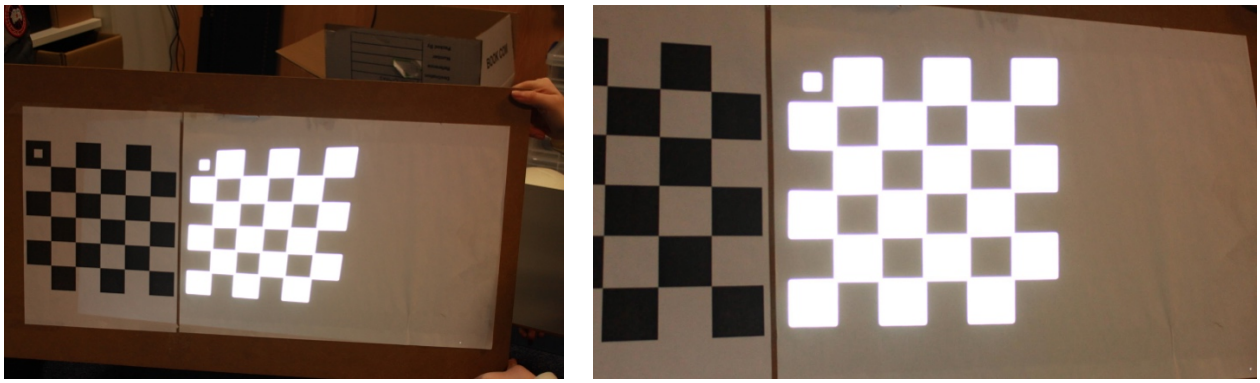
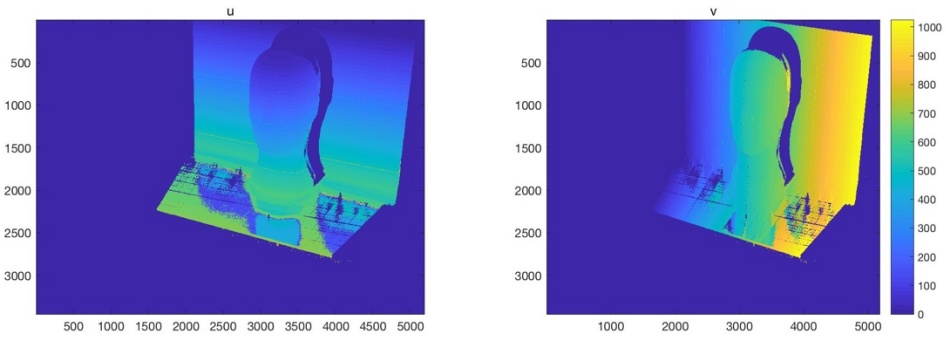


Figure 21: Views from camera (left) and projector (right)

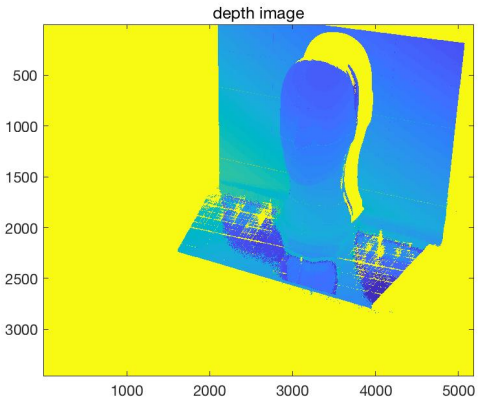
It could be found that the photo is a little overexposed, it was our mistake as we did not adjust the appropriate aperture size (F) under that lighting environment and it should be smaller. Moreover, the printed chequerboard has been cut off some parts and becomes uncompleted, hence I only used 3×3 grids for calibration to estimate matrices for both camera and projector.

Light pattern decoding, unreliable pixels elimination and depth map determination

In this part, I used the same code to finish reconstruction work. In addition, I used different thresholds to eliminate unreliable pixels in decoding. The figure shows below is the (u, v) code and depth map computed based on captured data. It could be found that the depth of table that head model stands is not continuous. After looking the gathered data set, we found that the table has quite strong reflectivity and Figure 20 illustrates this very obviously and it even reflected light from both head model and wood board. That is why we can find 'dark mirror shadow' in depth map at the connection part between head model, wood board and table.



(a) (u, v) code



(b) Depth map

Figure 24: (u, v) code and depth map with threshold = 1.5

Reconstruction results for own gathered data set

The reconstruction results of gathered data set with threshold=1.5 could be seen as below, which is quite noisy.

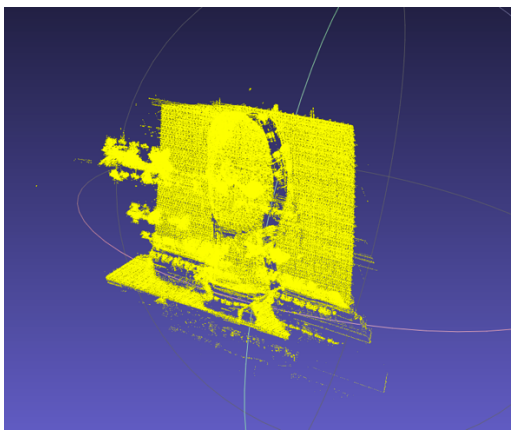


Figure 23: Reconstruction results with threshold = 1.5

It could be found that there are some reconstruction errors at ‘reflectivity’ part that we mentioned in last par as the discontinuity of depth leads the wrong position estimation and because they show closer depth in darker color, so those points appear to be shown in closer position in reconstruction. That is why we find that many points gather at the front part of the whole scene. In addition, the overall outline of head model and wood board are shown in reconstruction.

Then I increased the threshold to 4 and the reconstruction results can be seen below. Although most of points for table has been eliminated, the outline of head model become clearer because the large reduction of noises. We even could observe the outline of nose on the head.

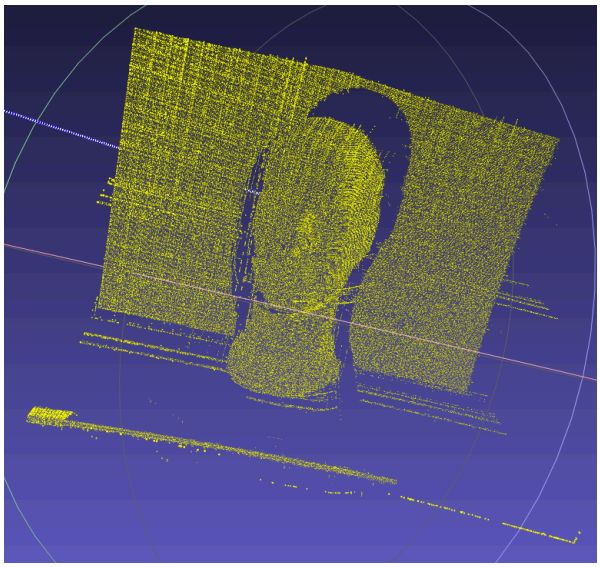


Figure 24: Reconstruction results with threshold = 4

One of differences between own captured data set and synthetic set is that the real projector displays patterns with high frequency by mapping multiple pixels to a single pixel. It means that there exists a narrow stripe to make projected image blur quickly, which it is easy for us to ignore it. This kind of errors only influence light patterns with high frequency and the final computed *(u, v)* code only have very slight offsets.

In the process of data capture, the camera still shacked slightly though it was fixed on tripod tightly when we pressed shutter or checked previous captured data. This might also lead some very few errors.

References

[1] Daniel Scharstein and Richard Szeliski. **'High Accuracy Stereo Depth Maps Using Structured Light'**.

[2] Hafeez Anwar, Irfanud Din and Kang Park. **'Projector calibration for 3D scanning using virtual target images'**.