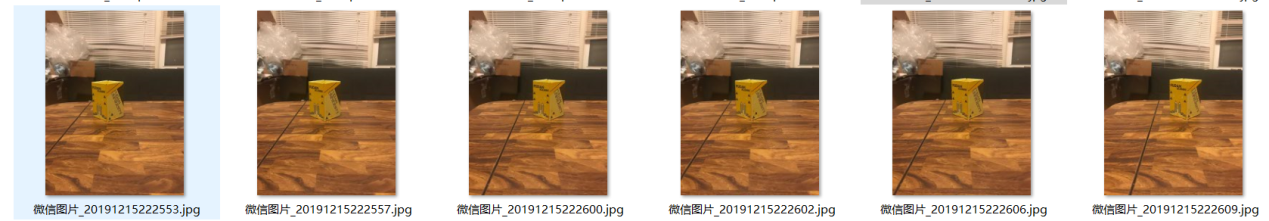
Q1:

a) In this question, I refer Matlab SFM Open-Source code to achieve the bundle adjustment.

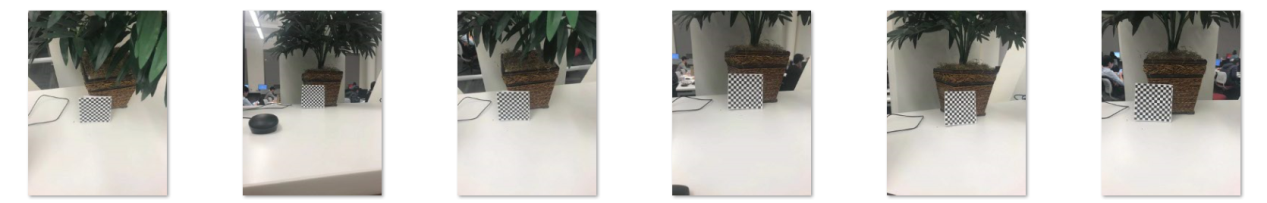
<https://www.mathworks.com/help/vision/examples/structure-from-motion-from-multiple-views.html>

The source images are shown below(9 images in total):

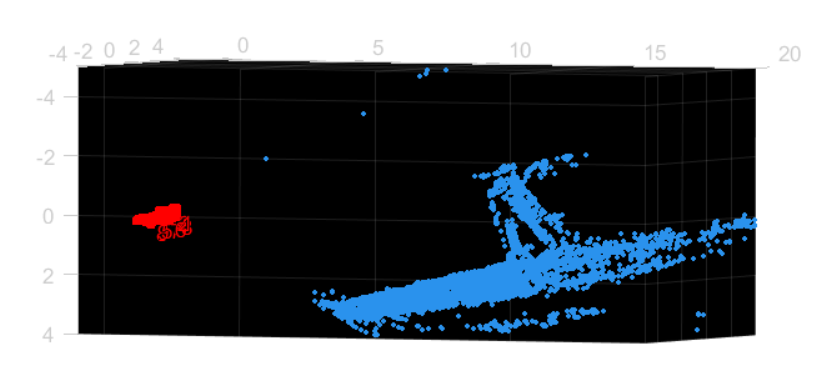


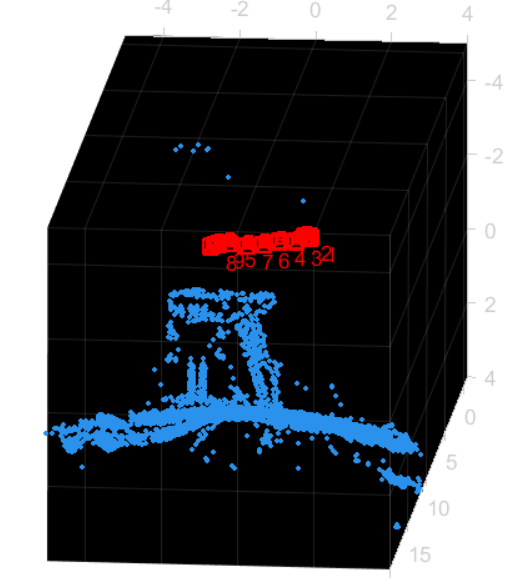


The matlab code also require checkerboard image to do camera calibration. Images (9 images in total) used for camera calibration are shown below:



Some of the results are shown here. More results can be found in the directory.





Focusing on the reconstructed object, there are two peaks (two buildings actually), which is as same as two peaks in the original images.

b) The matlab version of SFM includes several part:

**Step1:** Do camera calibration to get the intrinsic matrix. Here matlab use Camera Calibration App to finish the step.

**Step2:** Detect the matching points. We use SURF feature to finish this step including finding, extracting and matching features.

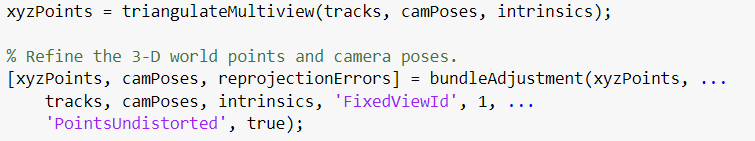
C:\Users\lenovo\AppData\Local\Temp\1576533661(1).png

C:\Users\lenovo\AppData\Local\Temp\1576533691(1).png

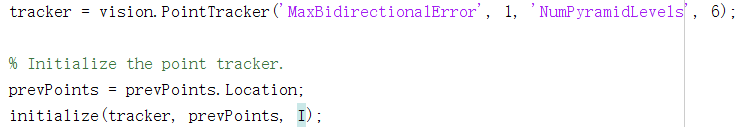
**Step3:** The next step is to locate the camera, in other words, **calculating the extrinsic matrix.** The Matlab code use a helper function called helperEstimateRelativePose to calculate the value.

C:\Users\lenovo\AppData\Local\Temp\1576533796(1).png

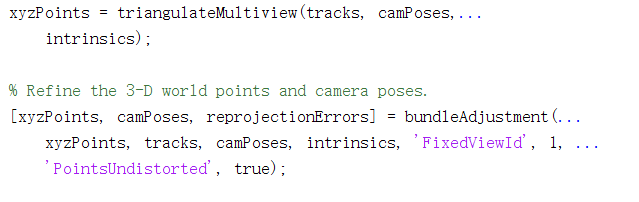
**Step4:** Refined the cameras’ poses for each one by 3D reconstruction and bundle adjustment.



**Step5:** Detect corners’ set for each camera to construct a point-track set



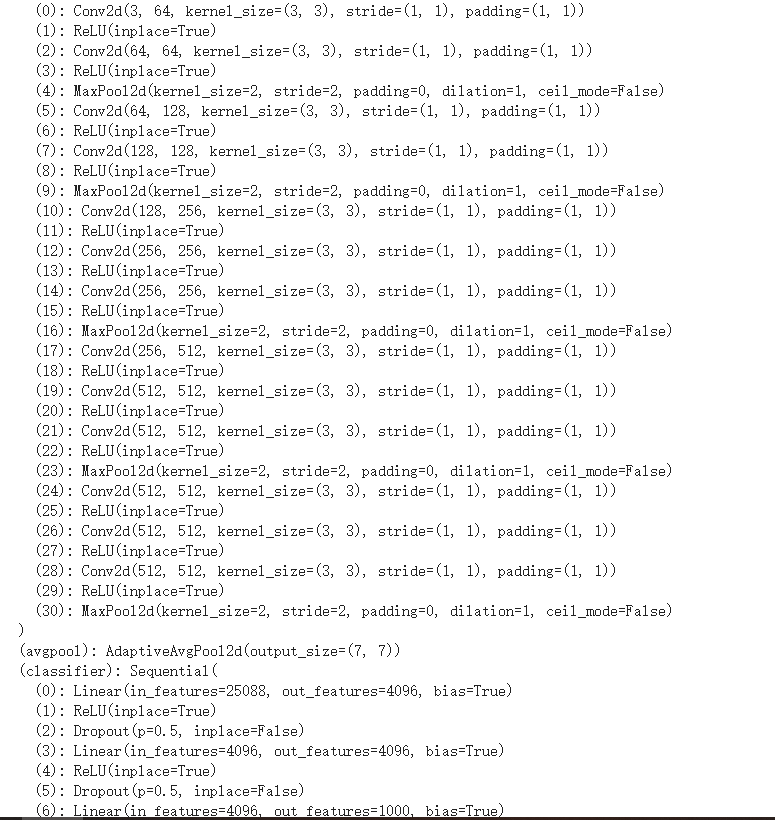
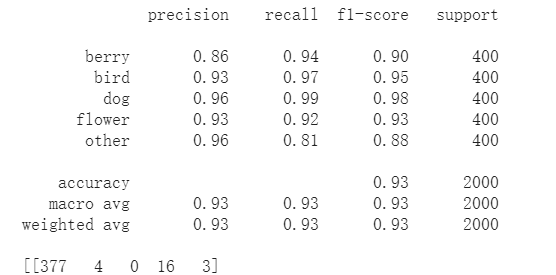
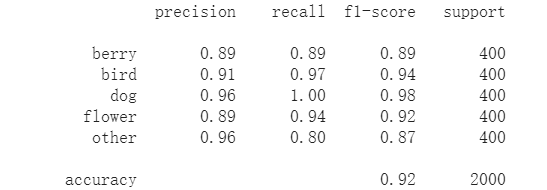
**Step6:** Reconstruct the object to get the point cloud.



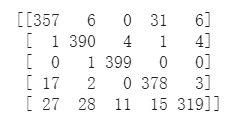
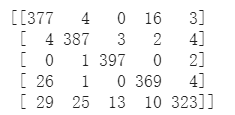
Q2:

Q2 require us to fine-tune an existing network and test the datasets. The code here I use is based on HW5 Q2 and PyTorch tutorial.

The network I use is VGG\_16 and VGG\_16\_BN for batch normalization.

1. Here is the result of VGG\_16 network without dropout and batch normalization (VGG network has a default dropout p=0.5). Here is the structure of the network:  
   
2. The VGG network is modified from AlexNet. Use three 3-by-3 convolution kernels to replace one 7-by-7 kernel. By stacking multiple convolution and max pooling layer to get a better result. The network used here contain:
3. The dataset used here is with 5 classes:  
   Here is the accuracy, precision, recall and f1-score of the result  
   
4. The VGG\_16 network has its own batch\_normalization version without any other differences. Also, the network has two original dropout layers. Here is the result with dropout and batch normalization.  
     
   The dropout is worked by randomly abandon several parameters in the network. Generally, the network would be overfitting due in the training and the dropout should produce better results.  
   The batch normalization is used to make the input of each layer obey the same distribution. With deeper layer, the input of each layer (we can call it features sometimes) would not obey the same distribution as the original input but more close to the up or down boundary of the non-linear function, like 0 or 1 in sigmoid function. The batch normalization can normalize the input data of each layer to make it close to the distribution of the input.
5. However, in my experiment, the results don’t show a clear difference. I guess the dataset may be too easy to show the effect of the improvement.

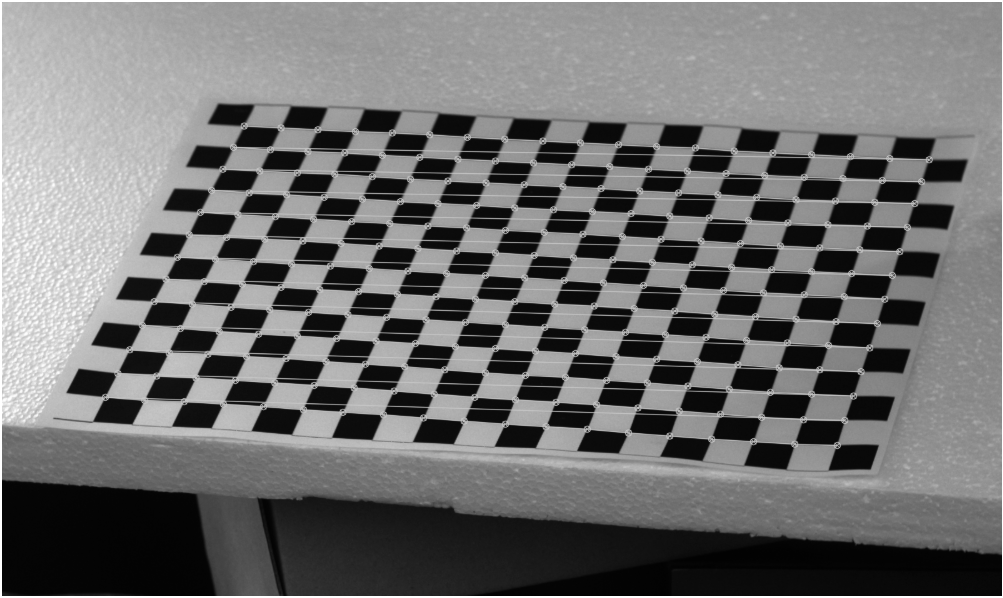
Here are confusion matrix of question c) and d)



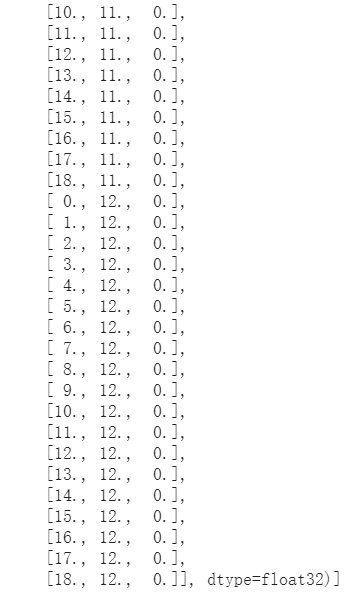
1. Obviously, most of the error is occurred by the mis-classifying of the “others” class. Considering the name of this class, it makes sense. Also, the berry and flower class also have some misleading. The most possible reason may be the same colorful scene of both classes.

Q3:

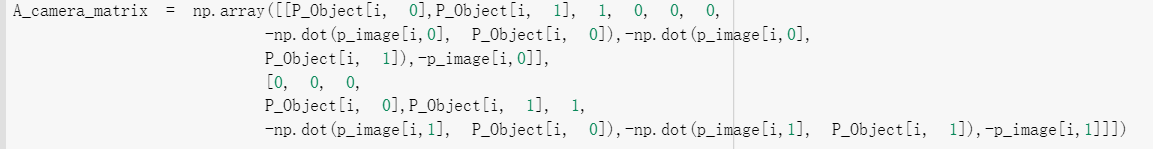
Here I use the code based on HW4 Q3. The code is

b) Here is the plane (checkerboard) in the original image.  
 

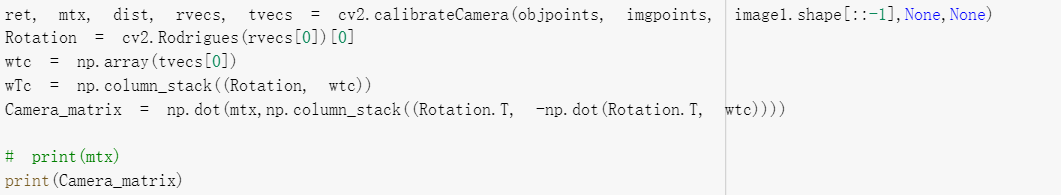
c) The new coordinates are the same as HW4 Q3, which is (0,0,0), (0,0,1)….. The



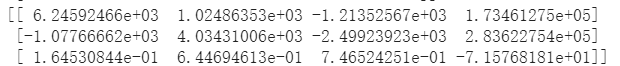
d) In this part, we need to calculate the camera matrix. To get the camera matrix, we need two parameters which are intrinsic matrix K and Homography matrix H. The K can be obtained by using cv2. CalibrationCamera. The homography matrix H, however, are actually the extrinsic matrix when the third row(Z=0) as the checkerboard is on a paper. So the here we can simply calculating the result by the same way without the third column.



This is the equation Aq=0 when q = [m11,m12,m14;m21,m22,m24;m31,m32,m34] and we can solve it by SVD. After that, we need to combine the intrinsic matrix and parameter to calculate the result. However… I can’t get the right one or I can say there is some error in my image. I double-check the result of camera matrix by OpenCV’s function but it still the same wrong result.

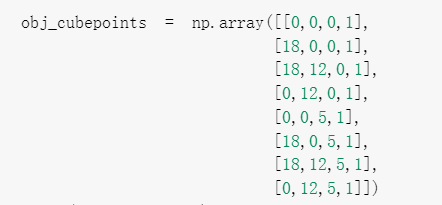


And the result is terrible, the focus length is more than 6000, maybe it is caused by the coordinates?



Anyway, I’m stuck and since I can’t get right camera matrix, there is no possibility to keep on. The answer of the following question is based on the assumption that I’ve already got the camera matrix.

e) As I’ve got the camera matrix, I can make a set of verticals like below to construct a cube:



Calculate the dot product of camera matrix and Object points and we can get the projection of the cube in the image.

f) In e) I get the projection of the cube. So we can simply correspond the corners of my own image with the verticals of one side to calculate the homograph matrix. The thing left is to do homograph transformation like the Harry Potter (I’ve done that in HW2)

Q4：

Here I use LSTM to construct a network. As the LSTM is the recurrent network and can obtain the information of the past time, I guess it can be used to do semantic segmentation or image classify in sequence images or videos.

However, I cant find a handy video dataset to do semantic segmentation. So I just use the MNIST dataset in this question to verify the network. The further working on semantic segmentation could be down after finding a suitable dataset.

Here is the result of the network with multiple metrics and the confusion matrix. The network is trained with 15 epochs without any pre-trained parameters. Considering this situation, we could say the network has a good performance. The only problem is that the training of the network seems can’t be done in the GPU. Each time I want to try training in GPU, the collar is broken down.

