Andrew Aquino
Midterm
DS681

I just wanted to give a small overall of my midterm submission and the code I used to accomplish the required task. So overall the script was broken up five main components:

1. Loading the images and detecting the checkerboard corners
2. Estimate homographies or Direct Linear Transform (DLT)
3. Solve for the intrinsic matrix K or Zhangs method
4. Compute the rotation and translation matrix for each image
5. Optimization using PyTorch and Gradient Descent

The first component mainly involved converting the dataframe image into a NumPy array and generating the real world coordinates of the checkerboard. We only needed the coordinates of X and Y while Z = 0 , since the checkerboard is a 2D plane. I attempted several times to get the corners of the checkerboard my manually writing the script was extremely unsuccessful. So for this part I used a OpenCV helper function to just detect and refine the checkerboard corner locations to apply the camera calibration.

The second portion of the code was the DLT Initialization and use the core of Zhangs method for camera calibration. Here is where we have to normalize the 2D points to improve the numerical stability and then estimate the 3x3 homography matrix H that maps 3D plane points into image pixels. Then using the Zhnags method we have to use the homographies to solver a linear system for an intrinsic matrix. From there is can be broken apart to create the rotation and translation portions of the matrix .

Lastly, I used PyTorch to optimize the initial values of the matrices. So we first have the Rodrigues formula then camera transformation then distortion to intrinsic matrix. From there using gradient descent the script can used the trained image to optimize the camera calibration parameters.

Category	Parameter / Metric	OpenCV Value	PyTorch Value	% Difference	Absolute Difference
Intrinsic Parameters	*fx* (Focal Length X)	559.04	411.30	26.43%	N/A
	fy (Focal Length Y)	560.19	420.46	24.94%	N/A
	cx (Principal Point X)	651.04	673.84	3.50%	N/A
	cy (Principal Point Y)	498.39	476.07	4.48%	N/A
	Skew (*s*)	0.00	-4.69	0.00%	N/A
Distortion Coefficients	*k1* (Radial)	-0.231619	-0.098839	N/A	0.132780
	k2 (Radial)	0.060984	0.005605	N/A	0.055379
	p1 (Tangential)	0.000101	0.003841	N/A	0.003740
	p2 (Tangential)	0.000071	-0.006602	N/A	0.006673
	k3 (Radial)	-0.007412	0.000378	N/A	0.007790
Reprojection Error	**RMSE**	**0.0942 pixels**	**3.0138 pixels**		
Images Used	**Count**	39	39		

As we can see from my implementation from the OpenCv there are some parameters that quite a large difference. From testing this is due to the initial K matrix that is predicted. I did do some hyperparemeter tuning like increasing the number of epochs (10000) and adjusting the learning rate (0.001), and found these values did get closer to the OpenCV values but not the same. This was only an issue for the training images, as I believe I was no inputing the correct checkerboard dimensions and square size.

```
Epoch 2800: Current Loss = 1.0480, Avg. Reprojection RMSE = 0.7239 pixels
Epoch 3000: Current Loss = 1.0454, Avg. Reprojection RMSE = 0.7230 pixels

--- Final Optimized Parameters ---
Optimized K:
 [[857.11909215  96.86574071 210.26580314]
 [  0.         862.33142852 111.99709262]
 [  0.           0.           1.        ]]
Distortion coefficients (k1, k2, p1, p2, k3):
 [-0.02504038  0.17674676 -0.01606734 -0.02462939 -0.24299825]

Final Average RMSE: 0.7227 pixels
```

But when I inputted the checkerboard images that I took with the known checkerboard dimensions, the Reprojection Error was around 0.72 pixels. Which is extremely good as compared to the OpenCV version. So that was just a small overall of my code implementing Zhangs method for camera calibration.