CS 477 HW #4

Questions completed: All undergrad level questions

Programming/Deliverables: MATLAB

Estimated time for assignment: 14 hours

A. Clicking and indexing points.

I provide the image coordinates for the image 'tent.jpg' from hw01 in Figure 1 below.

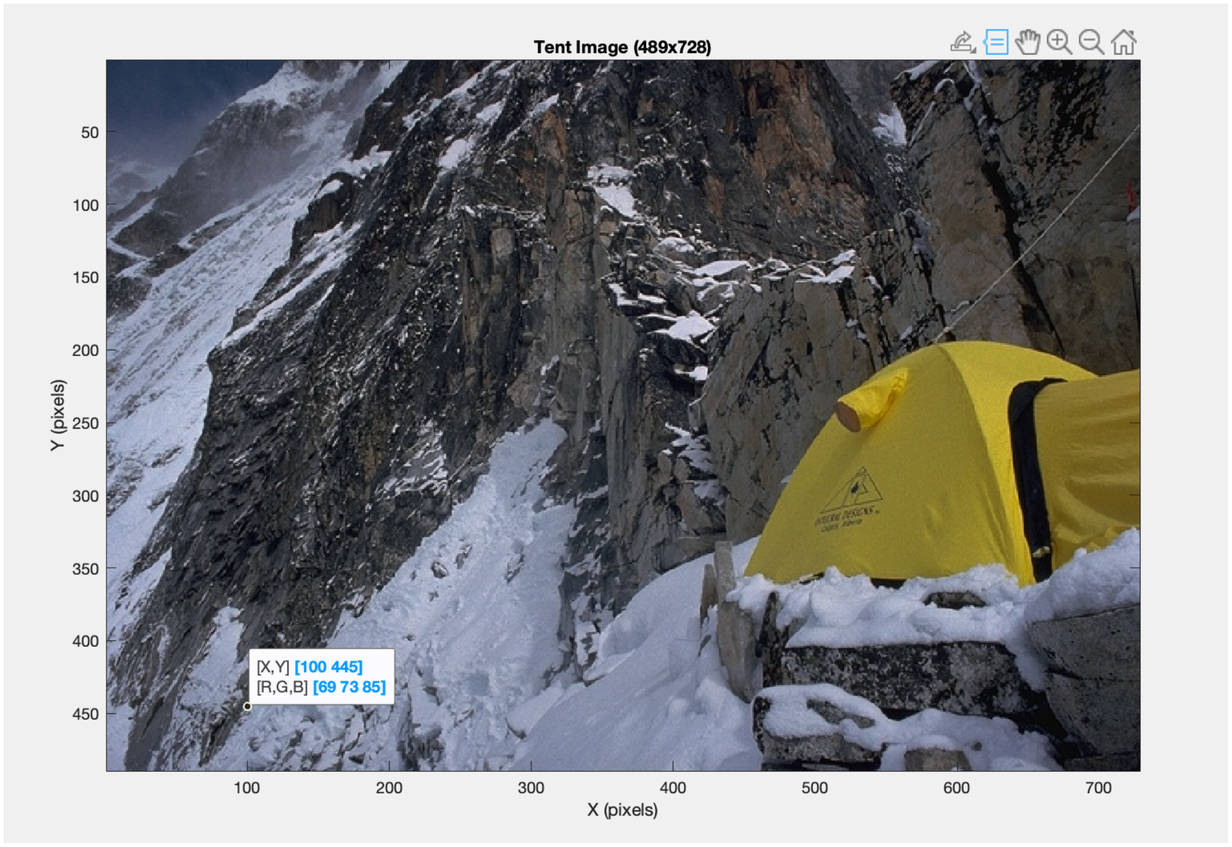
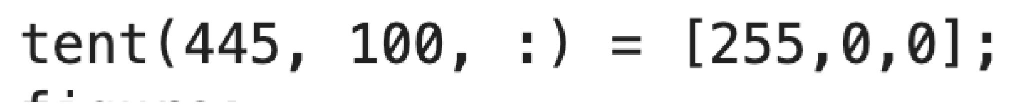


Figure 1: The image 'tent.jpg' with X and Y coordinates. This was generated in MATLAB using 'datacursormode on' and the function 'imagesc()'. There is also a clicked (X,Y) coordinate at (100,445).

Notice in Figure 1 that X is growing from left to right while Y is growing from top to bottom. There is also a clicked (X,Y) coordinate at (100,445) which you can see is consistent with the X and Y coordinate axes. Below the clicked (X,Y) coordinate are the RGB values for that clicked pixel which you can consider as additional info. From Figure 1, the reader can understand the clicked coordinate system we have setup and should be able to guess what the clicked coordinates for the top right corner of the tent image for example.

Accessing a clicked coordinate in the array in the software system, MATLAB in this case, is not the same as the (X,Y) coordinate format from Figure 1. To get access to the (X,Y) coordinate in the array, you must flip the coordinates to be (Y, X). This is because arrays are accessed row first and then by column. That is, the Y values are the rows, and the X values are the columns, so (X,Y) programmatically becomes (Y,X).

To get the clicked coordinate (100, 445) as we see in Figure 1. We need to access it in the tent matrix as (445, 100, :). We remember that tent is a 3D matrix in MATLAB so while conceptually going from (X,Y) to (Y,X) will remain the same, we need to return the RGB channels of the pixel in order to change its actual value. We see the snippet of code for this in Figure 2 below.



```
tent(445, 100, :) = [255,0,0];
```

Figure 2: Screenshot of the code to change the clicked coordinate (100, 445) in Figure 1 to a red pixel (255, 0, 0) in MATLAB.

Let's see if we can see the change of our clicked coordinate. I provide Figure 3 below with the modified tent matrix.

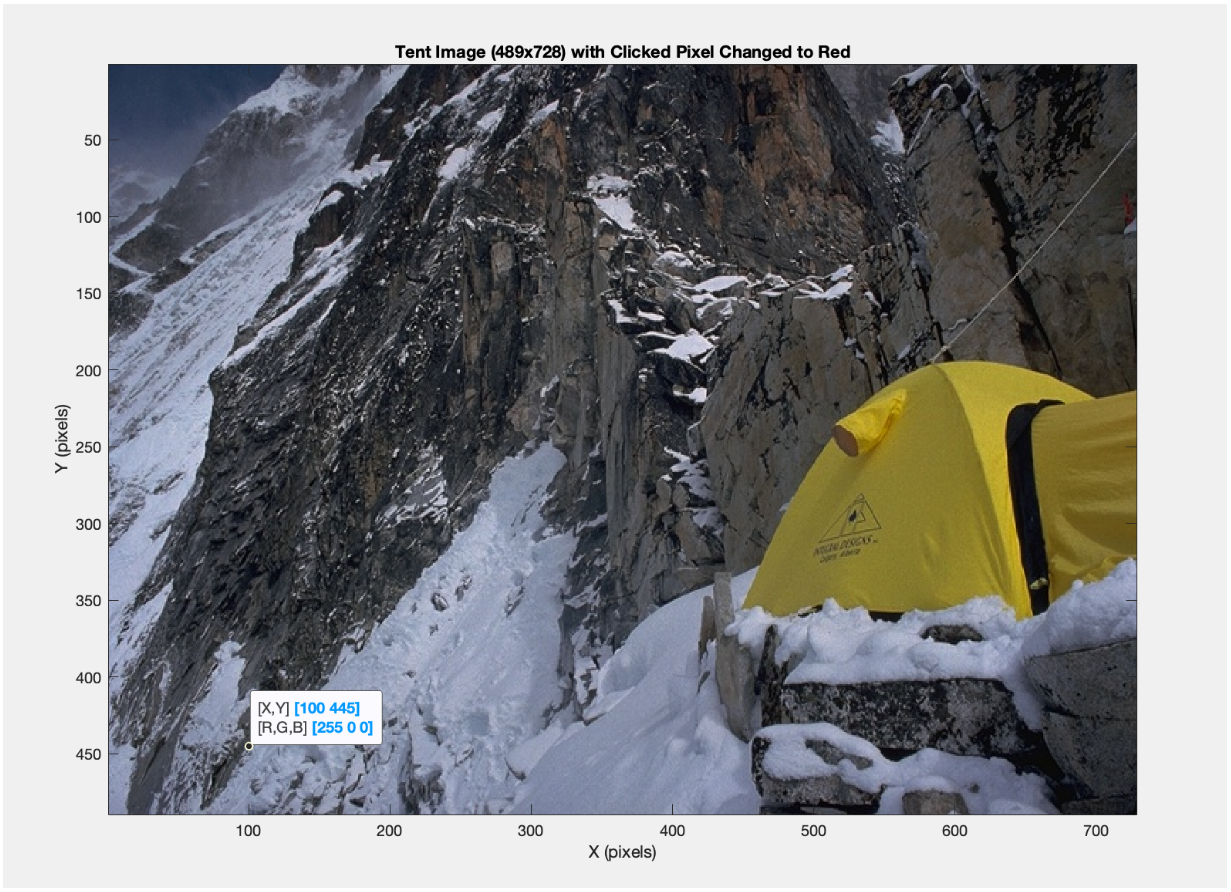


Figure 3: New tent.jpg image with our clicked coordinate (100, 445) from Figure 1 changed to a pure red pixel (255, 0, 0) in MATLAB.

We can see that our mapping of (X,Y) to (Y,X,:), specifically (100, 445) to (445, 100, :) in the tent matrix successfully changed our clicked coordinate to a red pixel as seen above in Figure 3.

B. Making a world coordinate system.

Below in Figure 4, I provide the world coordinate system for the image provided for hw03.

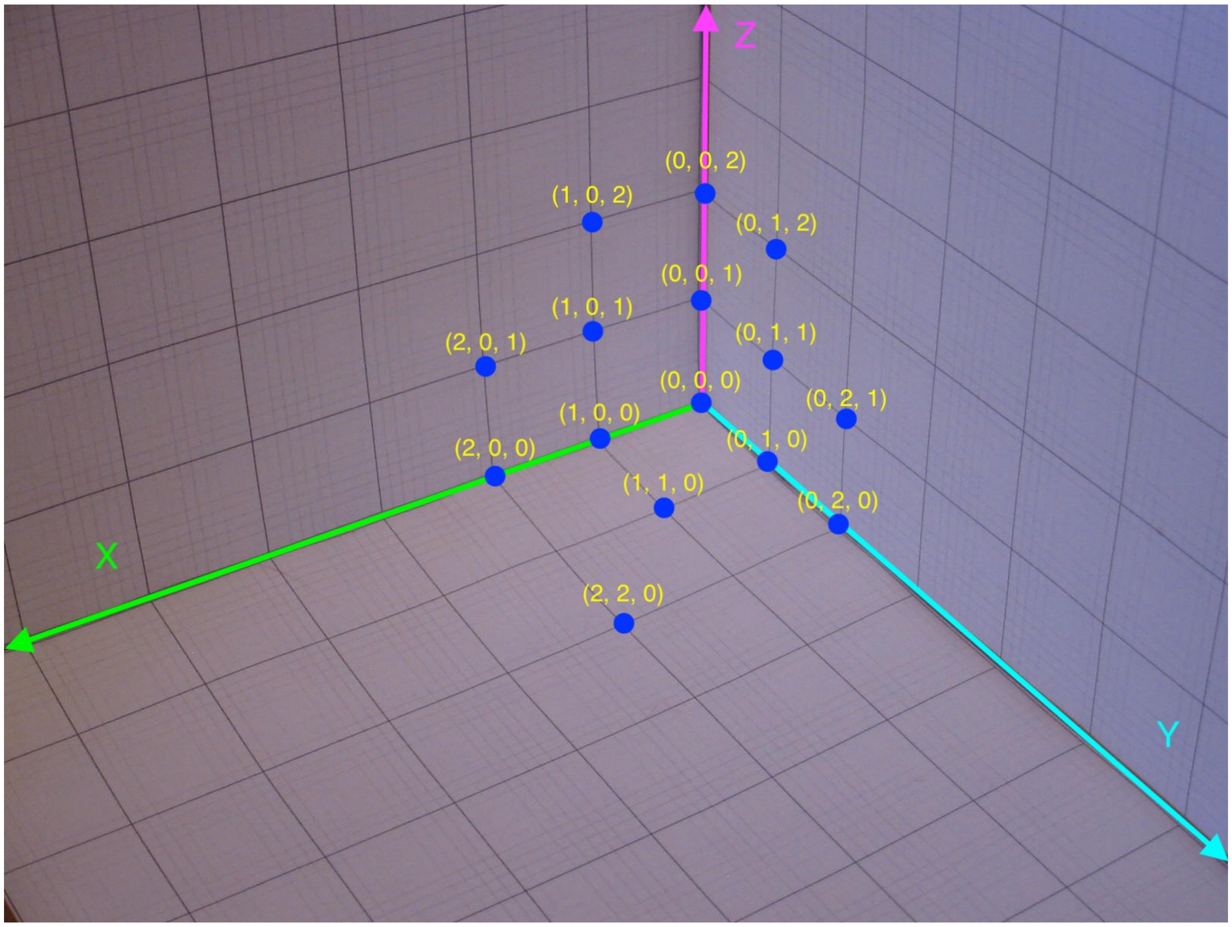


Figure 4: World coordinate system for provided image for hw03. The X, Y, and Z axes are clearly labeled, and the direction of the axes' arrows indicate in which direction they grow. 15 blue points have been selected for this experiement as shown, and their (X,Y,Z) coordinates are labeled in yellow.

These 15 blue world coordinate points in Figure 4 will be used for Part C to compare how two cameras map these 3D world coordinates to 2D. The files world_coords.txt and image_coords.txt have also been provided in this submission for Part C. The world coordinates of the points are as they are in Figure 4, and the image coordinates (X,

Y) were obtained using the 'datacursormode on' feature in MATLAB and manually clicking on the labeled blue points.

C. Analyzing how a camera matrix maps points from 3D to 2D.

Below in Figure 5, I provide the visual representation of our clicked world coordinates along with the two camera estimates of our world coordinates.

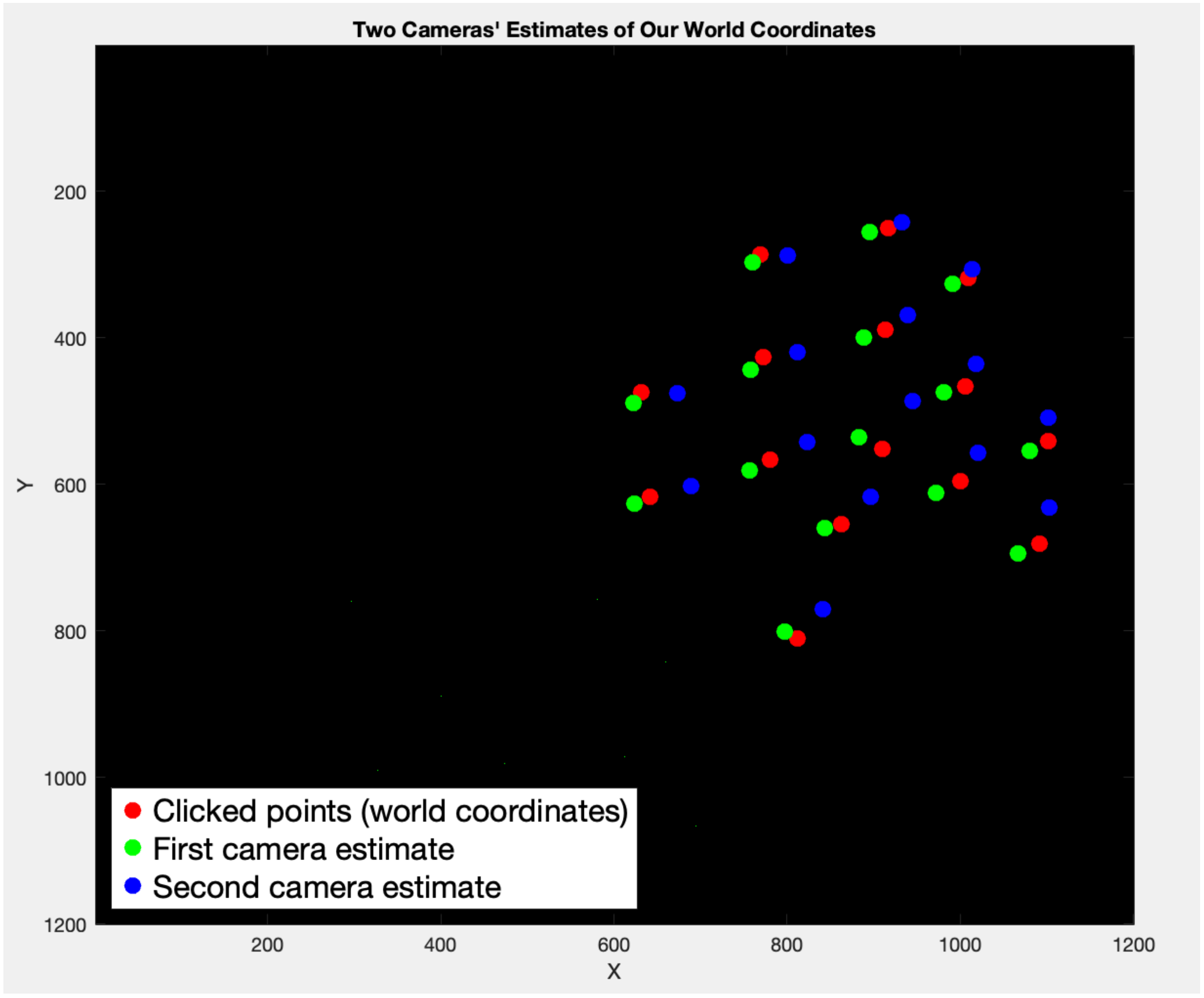


Figure 5: The red clicked points are our world coordinates from Figure 4 moved onto this black canvas to compare to our first (green) and second (blue) camera estimates. Produced in MATLAB.

In Figure 5 above, we can see that both the first and camera estimates are visually close to our red clicked points. One thing to note when plotting the estimates was to flip the X and Y coordinates generated from the given camera matrices which we explored the reason for in Part A.

The RMSE of the distance between the camera estaimte point and the actual clicked point was calculated for both cameras in MATLAB. Again, when calculating the distsances between points, we have to flip the coordinates generated from the camera matrices, or flip the image coordinates of the world coordinates. The key thing is to be consistent with our coordinate system (either work in X vs Y values or work with indices of a matrix). The RMSEs are as follows:

$RMSE_{camera1} = 23.9$

$RMSE_{camera2} = 40.7$

The RMSE of the camera estimated points and actual world coordinate points is also known as the re-projection error as stated in the homework. From the reported RMSEs of the cameras, the first camera is better quantitatively speaking, and from scrutinizing Figure 5, one can see that the first camera (green) points are closer to the clicked (red) points than the second (blue) camera points on average.

Thus, the first camera's matrix is more accurate.