Structured Light and Point Cloud Data

Alan Zhuolun Zhao, Pepper Shiqin Wang

3/26/17

ENGR027 Project #3

Professor Zucker

ENGR027 Computer Vision

Procedure:

Based on the `starter.py` given by professor Zucker and the project handout. We first implemented an iterative method to obtain the dot product of inverted camera projection matrix and every pixel coordinate, which is proportional to the actual (x,y,z) value. Then we examined the disparity value at every pixel coordinate and obtained the actual z value, and then reconstructed the original (x,y,z) coordinates. This method took 5.1 seconds on average when running with `cam1.png`

We also wrote a vectorized version of the method, which took 0.6 seconds on average when running with `cam1.png`

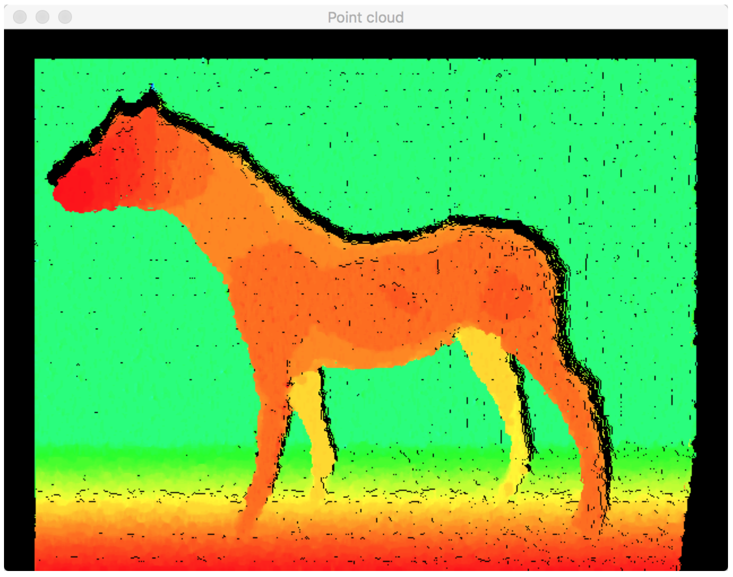
Discussion:

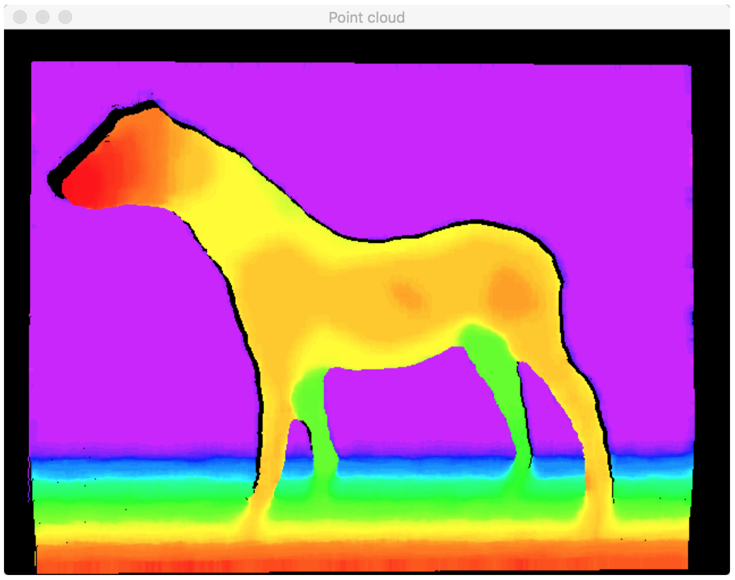
Who did what for this project?
We worked together on both the coding part and the report.

How did you approach writing the vectorized version of the program? Did you implement the iterative version first and then modify it, or did you do something else?
We implemented the iterative version first and then modified it. We used `meshgrid` and `hstack` to put all the pixel coordinates (z coordinates being all 1's) into a 3xn matrix and then performed dot product with `Kinv` (inverted camera projection matrix). Then we reorganized `disparity` to be an 1-D array and generated a mask based on a min-filter of disparity values (using Zmax = 8). We then calculated the updated Z values using the mask. Then we filtered the xyz matrix (to have the same length as the updated Z array) and performed element-wise multiplication of the xyz matrix and updated Z array.

Vary the `window_size` parameter for stereo matching. Try values of 7, 9, 15, and 21. How do the point clouds change? Are there any benefits to using smaller values? Larger ones?

As the value of window size grows larger, the point clouds have less black speckles. The difference is shown in the following pictures. The first one is the result of window size 7, and the second one is the result of window size 21.





According to the point clouds with different window size, we speculate that cv2.StereoSGBM_create would blur points on the same plane, that is points at the same distance from the camera, to remove noises, and the size of the blurring kernel is determined by the parameter window size. With a larger window size, the cloud points are smoother on each plane. However, a larger window size eliminates some points that are between the object and the background, and points whose planes don't have many other points. This can be proven by the number of points generated by the method. With a window size of 21, there are 297961 point of cam3. With a window size of 7, the number increases to 298136. With a smaller window size, the reconstruction looks more noisy but it keeps a more complete set of points.

In addition to the projector and IR camera, Kinect sensors also have an RGB camera which records color images. This can be useful for assigning colors to the points in the XYZ point cloud data (the resulting representation is sometimes referred to as XYZRGB). The starter code for this project includes both data taken from a real Kinect as well as a program load_kinect_data.py to view it. This particular data was taken from a sensor mounted on a TurtleBot 2 (http://www.turtlebot.com/). Given that knowledge, why is there a blank vertical stripe on the right hand side of each point cloud from the real Kinect? What physical feature does the stripe correspond to? (Study the robot photo on the website for hints...)

The stripe corresponds to the vertical metal supporting cylinder located at the forward-right position of the robot. This is because the projection camera is on the right side of the kinect (from the robot's perspective). The metal cylinder is likely to cover a vertical section of background in the projector image. However, for the cameras (which are located at the center of the kinect), the metal cylinder is not blocking. Therefore, the disparity would be abnormal for that region, causing the stripe on the point cloud.

Acknowledgments:

Responsibility were distributed evenly between lab partners. We would like to thank Professor Zucker for his starter code and help throughout the project.