Structured Light and Point Cloud Data

Mayank Agrawal and Owen Kephart

1. We did not create an initial iterative version of our program. In general, we found it easier to think of the code in terms of “filtering”, which was done using masks. We wanted to get a list of points where the disparity was greater than some minimum threshold, and so we filtered out all the data below this threshold. One thing that greatly simplified our process was converting our data into a format that was essentially a list of 3-vectors (the points) as soon as possible, rather than keeping it in a 2-D grid structure. This allowed us to much more easily do filtering operations, and more easily conceptualize how the data was flowing from instruction to instruction.

2. Increasing the window_size parameter generally gives a more fine-grained output depth map, and decreasing it does the reverse. However, when the window_size parameter is increased past a certain point (we tried 51), you will see wildly incorrect output data, as the algorithm is finding incorrect point correspondences as it is searching in a larger space (and thus has a higher probability of picking the wrong point). When the user is choosing the parameter, there is thus a tradeoff between smoothness and accuracy.[1] If the image is relatively smooth, it might be beneficial to have a larger value for window_size. On the other hand, if the image has sharp differences, users should use a smaller window_size value.

3. The blank vertical stripe on the TurtleBot corresponds to one of the right poles on the robot. The reason we don’t see the left one is because the sensor is not centered in the machine but rather more so on the righthand side.

[1] http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html