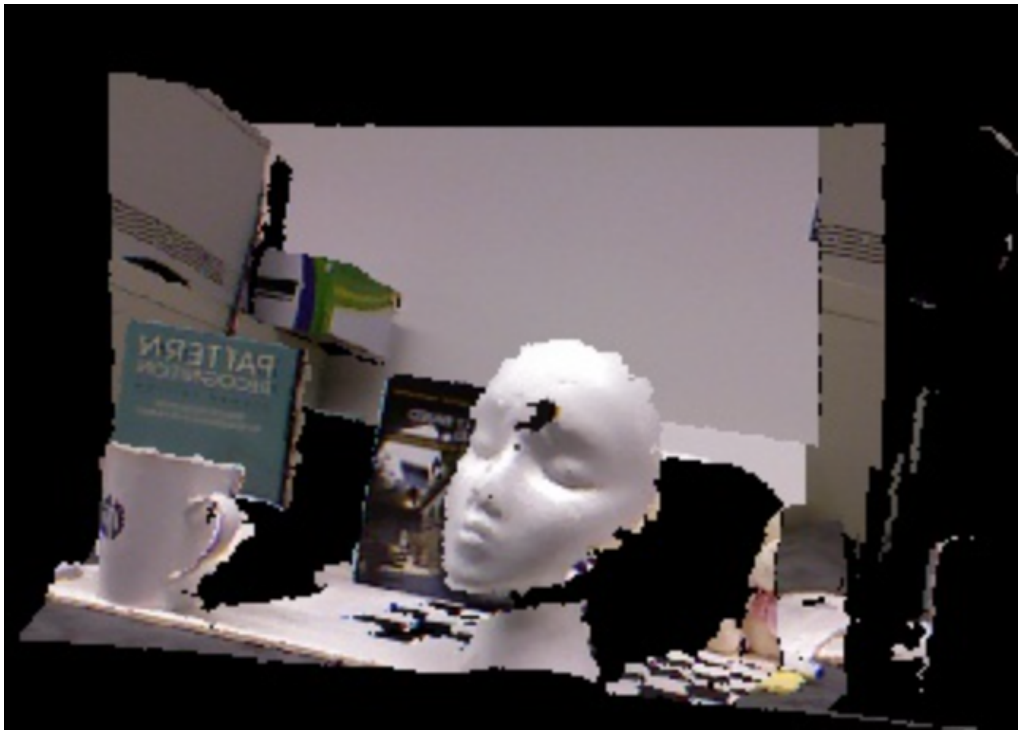
Camera Intrinsics and Extrinsics

For this project, you need to achieve 3D navigation on a static 2D image by using camera intrinsics and extrinsics. Given a pair of color and depth images captured by Kinect, your job is to reconstruct the 3D geometry with texture information using the inverse camera model. Then project the 3D scene into any arbitrary virtual camera and re-render it on a 2D image. As the example shows below:

Input Color

Input Depth



Re-constructed Virtual Image

Figure 1. Virtual 3D view demonstration using RGB-D image pair

Input Data

You are provided two image files:

(1) “color.jpg” a regular RGB image

(2) “depth.dat” a depth image that has the same size as the color image in terms of width and height. But the difference is each pixel of the depth image has the depth value instead of color value. The depth represents how far the corresponding scene point away from the camera center (measured in cm).

Both the color and depth image are captured by a Kinect sensor at the same time. **So their pixels are aligned already**, which means for a pixel at (x, y) in the color image, you know the corresponding depth value in the depth image at (x, y). You need to pay attention to is the data type for each image. For the color image, each pixel value is stored as “uchar”; however, for the depth image, each pixel value is stored as “short int”. So be aware to use the corresponding pointer to access the individual pixels accordingly.

What should you achieve?

(1) Successfully load the color and depth images into memory. (10%)

(2) Successfully reconstruct the 3D geometry by applying camera inverse model and render the virtual image. (30%)

(3) Successfully use the keyboard to update the camera's intrinsics and extrinsics to change the virtual view. (60% with the details of distribution below)

(a) Press ‘a’ key to translate the camera along x-axis by 10 units. [20% for (a)-(l)]

(b) Press ‘d’ key to translate the camera along x-axis by -10 units.

(c) Press ‘w’ key to translate the camera along y-axis by 10 units.

(d) Press ‘s’ key to translate the camera along y-axis by -10 units.

(e) Press ‘z’ key to increase the focal length, e.g. +30 pixels.

(f) Press ‘x’ key to decrease the focal length, e.g. -30 pixels.

(g) Press ‘r’ key to restart the virtual camera extrinsics.

(h) Press ‘o’ key to output the virtual image to a local file.

(i) Press ‘1’ key to rotate the image about x-axis clockwise, e.g. increasing 5 degrees.

(j) Press ‘2’ key to rotate the image about x-axis anti-clockwise, e.g. decreasing 5 degrees.

(k) Press ‘3’ key to rotate the image about y-axis clockwise, e.g. increasing 5 degrees.

(l) Press ‘4’ key to rotate the image about x-axis anti-clockwise, e.g. decreasing 5 degrees.

(m) Press ‘5’ key to rotate the camera about an arbitrary line L in clockwise, e.g. increasing 5 degrees. The Line passes the center of the 3D scene and has a direction vector v = [0.3, 1.0, 0]. You need to generate a series of matrices for this operation. [40% for (m)-(n)]

(n) Press ‘6’ key to rotate the the camera about the same line L (defined above) in counter-clockwise.

(4) Write up a “.pdf” document with several virtual images rendered from different angles or perspectives and clearly state what you have achieved or what parts you fail to accomplish. (10%)

Submission

Please turn in the code (e.g. .cpp file) and the pdf report through Isidore by the due time. Make sure to submit them on time. No email submission is accepted unless you are granted with extension.