Homework 4

Problem 1. Point Cloud Fusion. The goal is to fuse the point clouds of three different views of a scene into a single point cloud.

Part 1. Harris Corner Detection. This part aims to determine features (corners) that will be used to fuse the point clouds.

2. Compute the image derivates I_x and I_y for each pixel of the image separately. Use $\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}$ to compute I_x and its transpose to compute I_y. Use these first-order derivatives to compute I_x^2, I_y^2, and I_{xy} at each pixel.
1. Apply Gaussian smoothing to the image using the 5x5 filter shown at https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.
3. Compute the Harris operator response function for each pixel.
4. Apply non-maximum suppression on the responses of the Harris operator in 3x3 windows. This means that if a pixel does not have the maximum response in its 3x3 neighborhood, then it should not be included in the output. Also, make sure the order you process pixels does not affect the output of this step.
5. Pick the 100 corners with the strongest response.

Part 2. Corners to 3D points. This part aims to obtain the 3D points corresponding to the corners detected in part 1, using the depth map of each image.

1. For each detected corner $[x, y]^T$ in the image, obtain its corresponding 3D point $[X, Y, Z]^T$ as $\begin{bmatrix}X\\ Y\\ Z\end{bmatrix} = \frac{1}{s} d(x, y) K^{-1} \begin{bmatrix}x\\ y\\ 1\end{bmatrix}$, where $S = 5000$, $K = \begin{bmatrix}525.0 & 0 & 319.5\\ 0 & 525.0 & 239.5\\ 0 & 0 & 1\end{bmatrix}$, and d is the depth map of the image. Note that the depth map is an image with the same dimensions as the RGB image, so $d(x, y)$ is the (scaled) depth of the pixel at $[x, y]^T$. A value of $d(x, y) = 0$ indicates that the depth of the pixel is unknown.
2. Discard all corners for which $d(x, y) = 0$.

Part 3. Corner Matching. This part aims to find putative matches between the corners of images 2 and 1, and between the corners of images 2 and 3.

1. For each image, compute the rank transform in 5x5 windows.
2. Compute the distance between every corner of image 2 to every corner of image 1. Use SAD in 11x11 windows (on the rank transformed images) for computing distances.
3. For each corner of image 2, select the corner of image 1 that minimizes the distance.
4. Select the top 10 matches.
5. Repeat steps 2 through 4 but use image 3 instead of image 1.

Part 4. Pose estimation. This part aims to estimate the relative pose between images 2 and 1, and between images 2 and 3, using the 3D points from part 2 and the putative matches from part 3.

1. Apply RANSAC to estimate a rigid transformation between the 3D points of image 2 and the 3D points of image 1. The rigid transformation between a 3D point $P_{2,i} = \left[X_{2,i}, Y_{2,i}, Z_{2,i}\right]^T$ of image 2 and a 3D point $P_{1,i} = \left[X_{1,i}, Y_{1,i}, Z_{1,i}\right]^T$ of image 1 is given by $P_{2,i} = R_1 P_{1,i} + t_1$, where R_1 is a 3x3 rotation matrix and t_1 is a 3x1 translation vector. Note that $P_{2,i}$ and $P_{1,i}$ are corresponding 3D points determined from the putative matches of part 3. The values of R_1 and t_1 can be estimated from three non-collinear 3D point correspondences as follows:

$$
\begin{gathered}
v_{1,1} = P_{1,1} - P_{1,2} \\
v_{1,2} = P_{1,2} - P_{1,3} \\
v_{2,1} = P_{2,1} - P_{2,2} \\
v_{2,2} = P_{2,2} - P_{2,3} \\
R_1' = \left[v_{2,1}, v_{2,2}, v_{2,1} \times v_{2,2}\right]\left[v_{1,1}, v_{1,2}, v_{1,1} \times v_{1,2}\right]^{-1} \\
USV^T = \mathrm{svd}(R_1') \\
R_1 = UV^T \\
t_1 = P_{2,1} - R_1 P_{1,1}
\end{gathered}
$$

2. Repeat step 1 but use the 3D points of image 3 instead of image 1 to obtain the rigid transformation (R_3, t_3) between the 3D points of images 2 and 3.

Part 5. Finis Coronat Opus. This part aims to use the rigid transformations estimated in part 4 to merge the 3D points of images 1, 2, and 3 into a single point cloud.

1. For all pixels of the images, compute the corresponding 3D points as in part 2, discarding points with zero depth. Color the 3D points using the RGB images.
2. Use (R_1, t_1) and (R_3, t_3) to transform the 3D points of images 1 and 3, respectively.
3. Write the transformed 3D points of images 1 and 3, and the points of image 2 to a ply file as in homework 3.

Problem 2. Normal Mapping.

1. For all pixels of the image compute the 3D points as in problem 1 part 2. Discard points with zero depth.
2. For each 3D point compute the normal of the plane that passes through it. Use the 3D points in its 7x7 neighborhood (with respect to the image) to estimate the plane. Note that at least three non-collinear points are required to estimate the plane.
3. Store the normal $n = [a, b, c]^T$ of each 3D point in an RGB image. Use the following formula to map the elements of the normal to the $[0{,}255]$ range:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \left(\frac{n}{2\|n\|} + \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right) (255)$$

For pixels with zero depth, set the normal to $[0,0,0]$.

4. As you can observe, the resulting image is very irregular. This is because both n and $-n$ are valid normals for the plane, and which one is obtained depends on how the plane was estimated. A simple, albeit flawed, fix is to negate the normal if the coefficient of the plane equation corresponding to the distance from the origin is negative. For example, if the coefficients of the plane are $[a, b, c, -d]$, then set the normal to $n = [-a, -b, -c]$.

Acknowledgment

The input images, depth maps, and calibration data are from the TUM RGB-D dataset:

Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: Proc. of the International Conference on Intelligent Robot Systems (IROS) (Oct 2012).

Requirements and notes

Ignore pixels for which any of the filters falls even partially out of the image boundaries.