Assignment 5 Documentation

Submitted by
Soumava Paul (16EE10056)

- In this assignment, first I have computed the first and second-order gradients of a range image using the **Scharr filter** available in **OpenCV**. These arrays facilitate the calculation of mean, Gaussian and principal curvatures for each pixel. Next, I find the NPS at each pixel. These are then used for performing region growing of homogenous labels, segmenting the image into connected components. Details of those are given below.
- Different helper functions are implemented in the **utils.py** file. The broad tasks, namely, **curvature, topology, and NPS finding** and **image segmentation** using these properties are implemented in the **modules.py** file.
- All modules are sequentially called from the **main.py** file.
- For almost each range image, the **quality of segmentation** is as follows:

  Principal Curvature > Gaussian Curvature > NPS

- In some cases, though, **principal** and **Gaussian curvatures** perform almost equally. **NPS** only succeeds in finding the **outlines** of each object.

In `utils.py` file-

1. `def isvalid(i, j, r, c)` - Determines whether a pixel (i,j) is outside, or illegal for the image of dimensions r x c. Outputs 0 or 1 accordingly.
2. `def NPS_pixel(img, pix, threshold, print_flag)` - Determines the set of Digital Neighborhood Planes (DNP) belonging to a particular pixel by looking at a 3x3x3 neighborhood. A DNP is included in the Neighborhood Plane Set (NPS) if it contains more than `threshold` number of points out of a possible 9.
3. `def disp_2imgs(img1, img2, str1, str2, save_flag, save_name)` - This function is used for displaying a pair of images, one the original and the other a transformed version of it after doing some operations. The two images are stacked side by side. The save_flag also handles the saving of this image to disk.
4. `def bfs_visit(A, visited, label_img, i, j, r, c, label)` - Runs the BFS algorithm on a particular pixel (i, j) of **A** to combine connected pixels having the same value of a property into a bigger connected component. **Visited** keeps track of unvisited pixels. A single run of this function marks a single connected component in **label_img** with **label**.
5. `def generate_seg_img(label_img, r, c, smooth_thresh)` - After generating **label_img**, this function first eliminates spurious labels by removing components with smaller pixel counts and assigns them with the label of the biggest connected component. Finally, it generates a grayscale segmented image by

assignment grayscale values between **0-255** to the different components. Connected component with the highest pixel count is colored **black**, assuming it's part of the background.

In `modules.py` file-

1. `def mean_and_gaussian_curvatures(Ix, Iy, Ixx, Iyy, Ixy)` - Calculates mean and Gaussian curvatures per pixel using first and second-order gradients of Range image.
2. `def principal_curvatures(H, K)` - Calculates principal curvatures K1 and K2 per pixel using mean and Gaussian curvature arrays.
3. `def topology_HK(H, K)` - Finds topology at each pixel using Slide 24.
4. `def topology_K2K1(K2, K1)` - Finds topology at each pixel using Slide 25.
5. `def find_NPS(img, threshold, print_flag)` - Allots a decimal integer to each pixel depending on the combination of DNPs it has in its NPS. Resulting array is **NPS_img**.
6. `def seg_NPS_bfs(NPS_img)` - Labels different connected components by running BFS over NPS_img.
7. `def seg_gaussian_bfs(K)` - Labels different connected components by running BFS over Gaussian Curvature Array **K**.
8. `def seg_principal_bfs(K1K2_Top)` - Labels different connected components by running BFS over Topology Array **K1K2_Top** found using the principal curvatures K1, K2.

Running Instructions

From the terminal, run the `main.py` file as **"`python3 main.py -range range_img_name -th DNP_threshold -seg segmentation_method(0/1/2) -save save_flag(0/1) -print print_flag(0/1)`"**. The image name should also contain the proper file extension.

The 3 .py files are in the **codes** folder, all test images should in **RGBD_dataset** and all results are saved to the **sample_results** folder (if save_flag=1). While running different modules sequentially, you will need to **press 'Esc' to close** the image display windows. The **print_flag** if turned on, prints all curvatures, topology, and NPS per pixel.

Results

Segmented Images generated using NPS/Gaussian Curvature/Principal Curvatures are saved in the **sample_results** folder for each of the 5 range images.

Packages Required

Python3, NumPy, cv2