EE 645 {3D-Computer Vision}

Assignment 2: Warping Depth Images

GitHub Repo: (https://github.com/Harshp1802/3d-CV-Warping-Depth-Images)

- Name: **Harsh Patel**
- Roll No: **18110062**

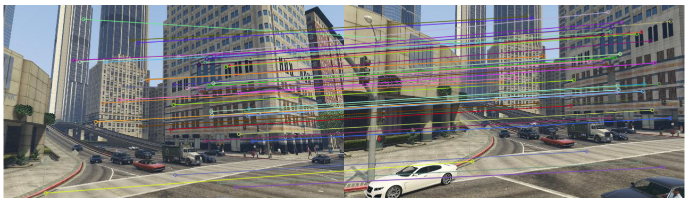
Tasks:

1. Detect, extract and match features between a pair of RGB-D images.
2. Let the pair of RGB-D images comprise a source image and a reference image along with their depth images.
3. Quantize the depth image corresponding to the reference image into m>10 levels. (For eg: if depth image has values from 0 to 100, then the image quantized to 5 depth levels will only have values 0,20,40,60,80,100)
4. Estimate homography matrix for each depth level of the quantized reference depth image. (using your function written in 1st Assignment and inbuilt function and compare the performance)
5. Warp each portion of the reference RGB image corresponding to each depth level using the corresponding homography matrix to obtain an image similar to the source image.
6. Now, estimate a single homography matrix for the image pair and warp reference image to obtain an image similar to the source image. Compare the warped image with the result obtained in (e).

Approach Used

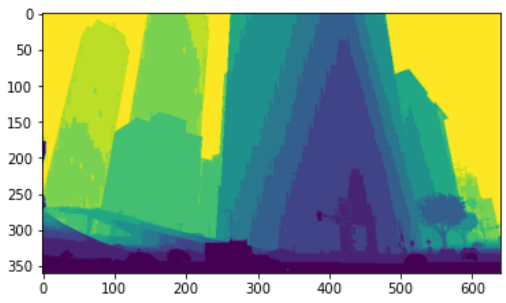
1. Quantizing the image corresponding to base image into multiple levels. (greater than 10).
   - I have tried to perform the quantization using 2 different methods:
     - Using Equal-width binning: This method deos not seem to be working well as frequency of pixels divided into different depth-levels are not uniform and result into poor homography generation.
     - Using Equal-frequency binning: This method tries to keep the no. of pixels in each depth as close as possible to keep the distribution uniform. Performs better than the equal-width binning. Used PIL quantization method (https://pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.Image.quantize). It uses Median-Cut algorithm for accurate and efficient quantization!
2. Detect and extract key features from a given image
   - **SIFT Algorithm**
3. Feature Matching [Simple Brute Force Method]

Example Feature Matching:



4. Distribute the found point correspondences into different depth-levels of the source image to calculate individual homography matrices of each depth-level.

Example Depth Quantization for m = 11:

-  |

:-------------------------😐:-------------------------:

[Original Image | [Depth quantized upto 11 levels]

5. Using RANSAC (RANdom SAmple Consensus) Algorithm that best fits the Feature matchings of two images in each depth-level
6. Homography Matrix Calculation using the Best Point Correspondences for each-depth level.
7. Performed Warping for each indivual depth-level consecutively to find the final warped image.

- Note: Each step is described with the code.

Stitching done for 4 images in all the Datasets

- Notes:
  - **I4** --> Left-most Image | I1 --> Right-most Image
  - The Second Image in all the datasets has been used as the Base Reference image for stitching, i.e, **I3**
  - **H_34** --> Homography matrices of I3 w.r.t I4 for **11 different depth-levels** [Needs to be inversed before finding the Warp as I4 warp is to be found w.r.t I3
  - **H_23** --> Homography matrices of I2 w.r.t I3 for **11 different depth-levels**
  - **H_12** --> Homography matrices of I1 w.r.t I2 for **11 different depth-levels**
    - Here we calculate H_13 for stitching the right most image by using:
    - `H_13[i] = H_12[i].dot(H_23_full)`
    - where, H_23_full is the full homography matrix withoust using depth quantization
  - Warping is done by finding the above matrices (**H**) and then transforming every image w.r.t I3
  - Inverse Warping method used to avoid creation of holes in the Output Image!

- The in-built Homography estimation is done using the Cv2 library.

How to run the Code:

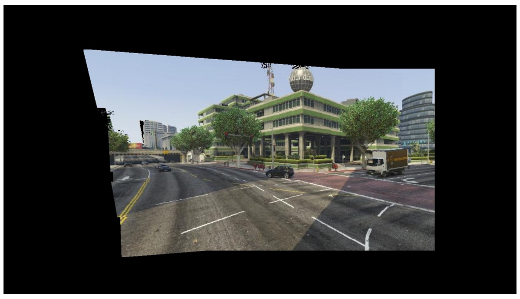
Use the Jupyter Notebook for the code:

1. 18110062_3D_CV_Assignment_2.ipynb [For with depth Warping]
2. 18110062_3D_CV_without_depth_warping

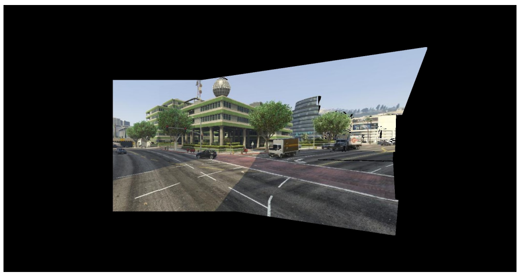
- Required Inputs:
    - **Image Dataset Number** --> (**Folder no.**) from `["000000029", "000000292", "000000705", "000001524", "000002812", "000003345"]`
    - **In_built** --> (**Bool**) True or False
        - True, if want to use the in-bulit Homography Estimation
- Tunable Parameters:
    - **No. of depth-levels for quantization : *default --> 11***
    - **No. of Iterations to run the RANSAC : *default --> 800***
    - **L1 Threshold for no. of inliers calculation : *default --> 5***
    - **No. of Iterations to run the RANSAC : *default --> 4000***
- The code automatically saves the following files in the following directory: `.\results\{Image Dataset Number}`
- Stitched Images (Without the use of in-built functions):
    - **Panaraoma_no_depth_1_2_3_4.jpg** [Final Output: 4 images stitched together]
    - stitch_no_depth_1_2.jpg [Stitching of 2 images]
    - stitch_no_depth_1_2_3.jpg [Stitching of 3 images]
    - **stitch_1_2_3_4_11.jpg** [Final Output: 4 images stitched together]
    - stitch_1_2_11.jpg [Stitching of 2 images with 11 depth_levels]
    - stitch_1_2_3_11.jpg [Stitching of 3 images with 11 depth_levels]
- Stitched Images (Using in-built Homography Estimation):
    - **Panaraoma_no_depth_in_built_1_2_3_4.jpg** [Final Output: 4 images stitched together]
    - stitch_in_built_1_2_11.jpg [Stitching of 2 images with 11 depth_levels]
    - stitch_in_built_1_2_3_11.jpg [Stitching of 3 images with 11 depth_levels]

Example Results:

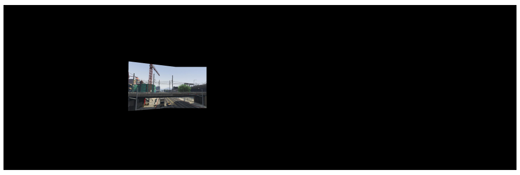
1. Warping of left image to the base image with 11 depth-levels using my homography estimation algorithm:

- 

2. Warping of right image to the base image with 11 depth-levels using my homography estimation algorithm:

- 

3. Warping of left image to the base image with 11 depth-levels using in-built homography estimation algorithm:

- 

4. Warping of right image to the base image with 11 depth-levels using in-built homography estimation algorithm:

-

Results for each Dataset:

The first 3 datasets seem to show more accurate stitching than others for the default parameters. Please look into the indiviual report files for images!

- Individual .md files [Images in these files are automatically updated on running the code]

```
        |   results_000000029.md
        |   results_000000292.md
        |   results_000000705.md
        |   results_000001524.md
        |   results_000002812.md
        |   results_000003345.md
```

```
```

- Individual pdf reports (".\results_report_pdf")

```
└───results_report_pdf
        results_000000029.pdf
        results_000000292.pdf
        results_000000705.pdf
        results_000001524.pdf
        results_000002812.pdf
        results_000003345.pdf
```

  - Please ignore the emoji in the pdf files. [Generated due to the conversion from .md to .pdf]

Observations:

- Warping the images using individual depth-level homographies seems to be a great approach for Panaroma stitching. While stitching it make sures that the object distant from the camera in the source image remains at the same depth level for the stitched image. This makes the stitching seem more realistic and effective.
- Increasing the no. of depth levels is good to perform improved stitching but upto a certain level only. If the depth is increase more than that level, the estimated homography matrices are not very accurate because the no. of accurate point correspondences decrease as we increase the levels.
- For me, the results seemed very satisfactory (better than without-depth homography) at depth levels upto 5 or 6. But in the assignment, we are asked to perform stitching for more than 10 levels, so I have included only those results.

Run the code for around 5 to 6 depth levels to get more accurate stitching!