Lab 7, 3D vision

- 3D Vision
- Disparity map
- Dense mapping
- 3D reconstruction
- Visualization and manipulation of 3D point clouds using open3D
- Registration of point clouds using ICP

7.1 Disparity map

Recover the code from the exercise on image rectification (exercise 5 in last lecture - in alternative you might use the available code in reconstruct.py). Use the class `StereoBM` and the function that implements a block matching technique to find correspondences over two rectified stereo images. Use the parameters specified as follow since we will not enter in details of these functions. Be careful to use gray level rectified images for the correspondence algorithm. You might modify the Stereo Matching parameters or even try other methods (for example `StereoSGBM_create`).

Note

The conversion to an 8 bits grey level image to display the disparity map.

```
# Call the constructor for StereoBM
stereo = cv2.StereoBM_create(numDisparities=16*5, blockSize=21)

# Calculate the disparity image
disparity = stereo.compute(remap_imgl,remap_imgr)

# -- Display  as a CV_8UC1 image
disparity = cv2.normalize(src=disparity, dst=disparity, beta=0,
    alpha=255, norm_type=cv2.NORM_MINMAX);
disparity = np.uint8(disparity)

cv2.imshow("left", left)
cv2.imshow('Disparity Map', disparity)
cv2.waitKey()
```

Listing 21: Code for Stereo Block Macthing

7.2 3D Reconstruction

Use the function `cvReprojectImageTo3D` to compute the 3D coordinates of the pixels in the disparity map. The parameters of `cvReprojectImageTo3D` are the disparity map (disp in previous exercise), and the matrix Q given by the function `cvStereoRectify`. Save the 3D coordinates in a npz file.

7.3 Visualization of point cloud in pcl

Modify the source code viewcloud.py to read the 3D points of the file you have saved in the previous section and visualize the results of the 3D reconstruction.

```
p = points_3D.reshape(-1, 3)
fp = []
for i in range(p.shape[0]):
    if np.all(~np.isinf(p[i]):
        fp.append(p[i])
pcl = o3d.geometry.PointCloud()
pcl.points = o3d.utility.Vector3dVector(fp)
```

Listing 22: Code for pointCloud creation

Visualize the 3 points and add any filtering necessary to improve the visualization of the reconstructed 3d Points . You might also use the left image to add colour information to each 3D point, for these you can specify the color of the point cloud with the `pcl.colors`, specifying the color as an rgb value between [0,1]

7.4 PCD (point cloud data) 3D format

Modify the source code `viewcloud.py` to read and visualize the two provided kinect images `filt_office1.pcd` and `filt_office2.pcd`. The Point Cloud Data file format (PCD) used is the 3D file format from PCL and can be written and read directly using the open3D functions `o3d.io.read_point_cloud` and `o3d.io.write_point_cloud`.

Note

By default, kinect sensor returns NaN values that may cause problems in the processing, to remove NaN values and at the same time, downsample (reduce the number of points in the file) using the `_down_sample`. filter with a grid size of 0.05 in each direction. The `filt_office1.pcd` and `filt_office2.pcd` files have already been treated with this filter resulting down sampled cloud of points (original Kinect images are in files `office1.pcd` and `office2.pcd`).

7.5 ICP alignment

Use the `registration_icp` function to align the given down-sampled cloud of points. See: `http://www.open3d.org/docs/release/tutorial/Basic/icp_registration.html/#point-to-point-icp` Note that the values of the threshold should be adapted for each case. Normally an initial rough registration should also be provided to avoid bad registration However in this case, given the proximity of the provided depth images, this should not be necessary. Visualize in the same window the original and the aligned cloud of points. Modify the ICP parameters to check the quality of the registration (for example use the default values and evaluate the results).

Note

The evaluated transform can be recovered with the method `transformation` of the `registration_icp` object. You can apply the transformation to a point-cloud using the transform method.

7.6 Report

Continue the report started in the last lecture. It should contain an example of the images displayed in each exercise, as well as your comments about them. You may also use the second set of stereo images as example in your report (StereoL and StereoR images). You should also provide the values of the several matrices obtained along the class. Optional/homework parts should appear in your final report.