NUS CS5477: 3D Computer February 15, 2024

Assignment 2

Lecturer: Lee Gim Hee *Student: Zhang Rongqi A0276566M*

1 Implementation

- `detect_lines()`: Followed the instructions to call `cv2.Canny` and `cv2.HoughLines` to automatically get horizontal and vertical lines. Number of Lines is $O(L)$.
- `get_lines_from_line_pts()`: Simply do the cross product of each pair of points, and then do the normalization. Time complexity: $O(1)$.
- `get_pairwise_intersections()`: Iterate each pair of lines from the list of lines and do the cross product to get their intersection and save the result in a list. If the scale factor of the intersection equals 0, meaning they are parallel lines, skip adding it to the list. Number of points is $O(P) = O(L^2)$. Time complexity: $O(L^2)$.
- `get_support_mtx()`: From the pairwise intersections and lines, compute the support matrix between the intersections and the lines if the distance between the intersection and line is within the distance threshold. Use $d = \frac{|Ax+By+C|}{\sqrt{A^2+B^2}}$ to calculate the distance between the line $l = (A, B, C)^T$ and the point $p = (x, y, 1)^T$. Time complexity: $O(PL) = O(L^3)$.
- `get_vanishing_pts()`: For each of the n loops, find the point with maximum supporting lines. And remove these lines to avoid them supporting another vanishing point. Time complexity: $O(nPL)$.
- `get_vanishing_line()`: Simply do the cross product of each pair of points, and then do the normalization. Time complexity: $O(1)$.
- `get_target_height()`: First, find the horizontal and vertical vanishing points. Then project t_2 onto t_1b_1 to get $\tilde{t}_2$. Then take the norm of vectors on t_1b_1 to get their distance ratio in the real world. Time complexity: $O(1)$.

2 Result

Intermediate results I get are the same as the TA's. Due to space limitations, the results will not be displayed. Final result: `INFO: the target height is 3.44m.`