Nicholas Sun Jun Yang, A0217609B

CS4277 Lab 2 Report

detect_lines:

1. Use OpenCV's canny edge detector to detect edges in the image. The function finds edges in the input image and marks them in the output map edges using the Canny algorithm. The smallest value between threshold1 and threshold2 is used for edge linking. The largest value is used to find initial segments of strong edges.
2. Use OpenCV's HoughLinesP to find line segments in a binary image using the probabilistic Hough transform using the Canny edge image from step 1.

get_pairwise_intersections:

1. For each part of line, we use a double for loop to get the intersection for each pair
2. Intersection is found using a cross product of 2 lines
3. Only add back points of intersections where the homoegenous coordinate is not zero as that would indicate points of infinities

get_support_mtx

1. Compute the I x L distance matrix between lines and points of intersections.
2. Use a double for loop to find the distance between each pair of point of intersections and lines
3. The distance is the perpendicular distance of the normal of the line intersecting the point of intersection: $\mathbf{d = |ax + by + c| / (a^2 + b^2)^{½}}$
4. Set the support matrix [i, j] to 1 if the distance_mtx[i,j] is below the distance threshold.

get_vanishing_pts

1. Use a for loops to loop n=num_vanishing_pts times
2. Each loop find the intersection with the most support as the vanishing point. We do this by summing up the total number of supports for each intersection using np.sum(support_matrix,axis=1) and then finding the position of the max support indice using np.argmax
3. Remove its support set (of lines) from the list of lines by removing all the lines supporting it by setting their position in the support matrix to 0
4. Essentially get the n most supported vanishing_pts at the end of the loop

get_vanishing_line

1. Simple cross product to get the vanishing line between two horizontal vanishing points

get_target_height

1. Compute the vanishing point $\mathbf{u} = (\mathbf{b}1 \times \mathbf{b}2) \times \mathbf{l}$
2. Compute the transferred point t = $(\mathbf{t}1 \times \mathbf{u}) \times \mathbf{l}2$, where $\mathbf{l}2 = \mathbf{v} \times \mathbf{b}2$.
3. Calculate scene points distance (t1,v,t2) from b2 using
   - t1 = np.sqrt((transferred_point[0] - pt_b2[0])**2 + (transferred_point[1] - pt_b2[1])**2)
   - t2 = np.sqrt((pt_t2[0] - pt_b2[0])**2+(pt_t2[1] - pt_b2[1])**2)
   - v = np.sqrt((vanishing_pt_v[0] - pt_b2[0]) ** 2 + (vanishing_pt_v[1] - pt_b2[1])**2)
4. Compute distance ratio using distance_ratio = (t1 * (v - t2)) / (t2 * (v - t1))
5. Find target height using target_height = query_d1 / distance_ratio