Student: Li Hongguang (A0233309L)

Implementation: detect_lines()

1. Call **cv2.Canny()** for edge detection.
2. Pass detected edges from step 1 into **cv2.HoughLinesP()** to extract coordinates of lines in image.

Implementation: get_pairwise_intersections()

1. Compute cross product of every unique pair of line using a double-nested for loop where the inner loop start index increases to avoid double counting.
2. Remove non-existent intersections (i.e. intersections where z coordinate is 0).

Implementation: get_support_mtx()

1. Use a double nested for loop to calculate the distance between every combination of line-intersection pair.
2. If the distance between arbitrary intersection point *i* and arbitrary line *j* is lower than given threshold, set support matrix value at index (*i, j*) to 1. Otherwise, set the value to 0.

Implementation: get_vanishing_pts()

1. With support matrix computed from **get_support_mtx()**, do a row-wise summation to calculate the total number of supporting lines for each intersection point.
2. Call **np.argmax()** to retrieve the index of the intersection point with the highest number of supporting lines.
3. Extract the homogeneous coordinates of the corresponding intersection point from step 2 and append it to the list of vanishing points.
4. Loop through the support matrix for the extracted intersection point, setting the entire column of supporting lines to 0 (i.e. if line j supports the intersection point extracted in step 3, set support_mtx[:, *j*] = 0).
5. Repeat steps 1 to 4 for any number of required vanishing points.

Implementation: get_vanishing_line()

1. Compute cross product of the pair of vanishing points.
2. Scale the homogeneous coordinates of the vanishing line such that its z-coordinate is equals to 1.

Implementation: get_target_height()

1. Compute vanishing point u by computing cross-product of b_1 and b_2, then crossing the resulting line homogeneous coordinates with the vanishing line l. Then scale u such that its z-coordinate is equals to 1.
2. Compute l_2 using cross product of b_2 and t_2, then scale it until the z-coordinate of l_2 equals to 1.
3. Compute transferred point t_1*_tilda* by taking the cross product of t_1 and u, then crossing the resultant line with l_2. Scale t_1*_tilda* such that its z-coordinate is equals to 1.
4. Compute v by taking the cross product of b_1 and t_1 to derive l_1, then taking the cross product of l_1 and l_2 (derived in step 2)
5. Calculate the distances of t_1*_tilda*, t_2 and v relative to b_2.
6. Compute distance ratio using formula derived in lecture 5.
7. Multiply the query height with the inverse of the distance ratio to derive the target height.

Student: Li Hongguang (A0233309L)

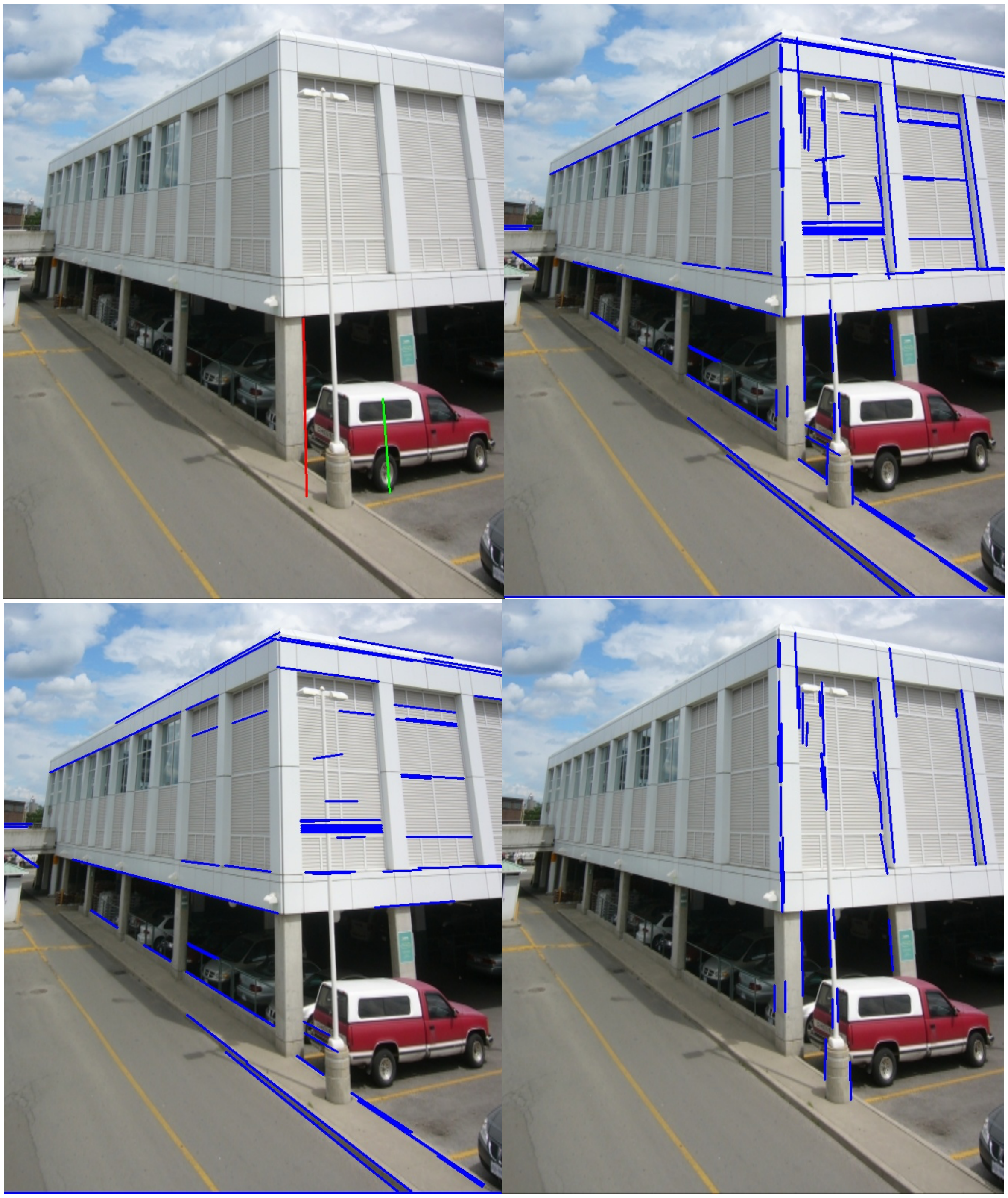
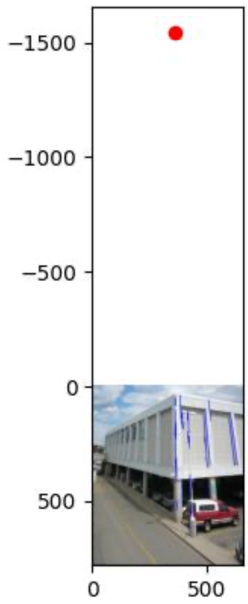


Fig 1. Results from edge & line detection

Student: Li Hongguang (A0233309L)



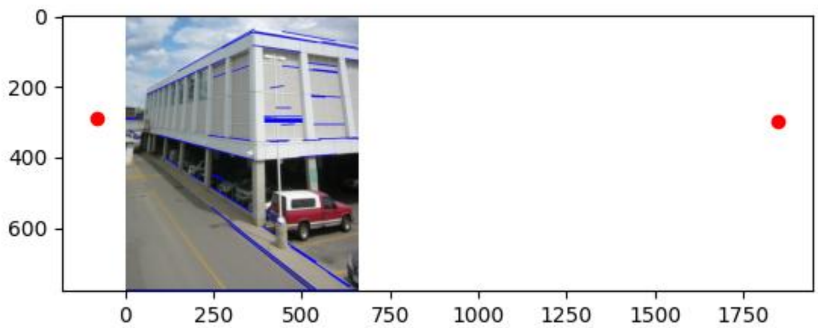


Fig 2. Computed vanishing points using RANSAC