Ltsw23

Computer Vision Assignment

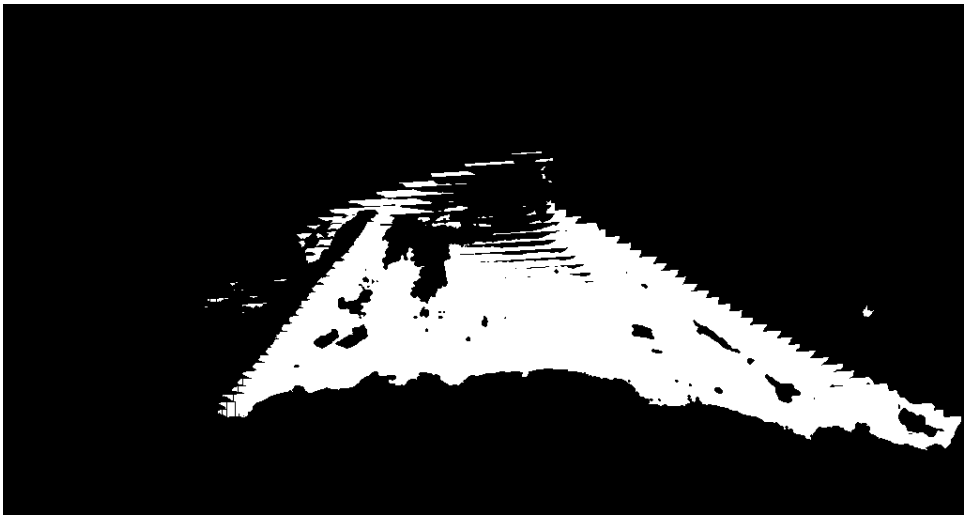
Method and justification

1. Get left, right images

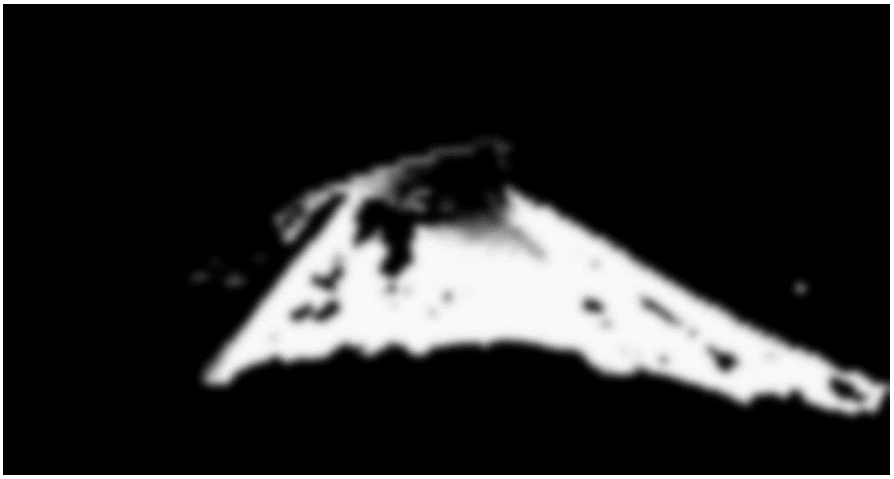
2. Generate disparity from the images

3. Also generate masked version of the disparity

4. From disparity generate the 3-dimensional point field of the image using given projection function with change of Zmax to Z as projection will change with distance (z value) from the camera.
5. Repeating n times:
    a. Randomly pick three points from the centred mask region of the image
    b. Generate plane coefficients from said points such that ax + by + cz = d and b in the normalised coefficients is tethered to within y_tolerance of 1(to ensure that the plane found is roughly upward-facing and hence cannot match with vertical structures such as houses or cars)
    c. Calculate the average distance of points in the 3d point field that are within detection_range of the plane.
    d. If minimal average, save current candidate plane as the best plane
6. Having got the best plane, gather all points in the point field that are within drawing_range of the plane and prepare points for bounding:
    a. Generate blank greyscale image of the correct size
    b. Draw white (255) on locations of selected points

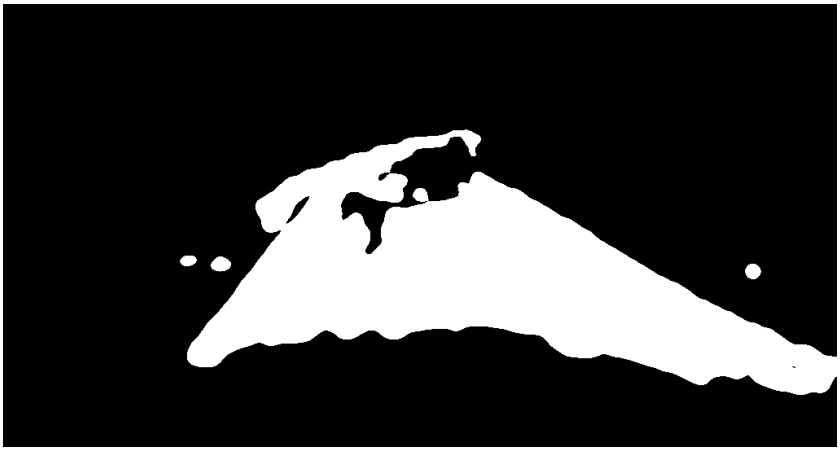


    c. Blur all points using Gaussian blur with very large dimensions

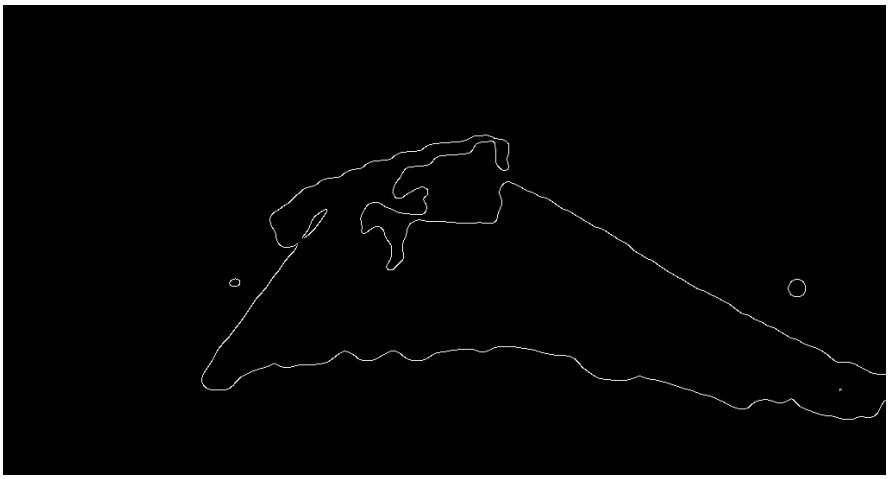


    d. Convert image back to binary (0 or 255) image with very low threshold for high values. Perform this process to ensure that reasonably nearby points are made one single object. Due to the poor disparity resolution, empty bands occur between bands of points that would otherwise form a single flat plane. This process helps to rectify this, though there will inevitably be some overspill into similar-height areas

such as pavements.



e. Perform canny edge detection on the points to aid the finding of contours.



7. Use opencv findContours to get all contours
8. Create and display a convex hull around the largest contour-the best candidate for the complete road.
9. Generate and display the normal to the plane using: $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$

10. Final image:

Development

Plane suitability metric

Multiple metrics for assessing the plane were tested: sum of distances between all points in the image and the plane, count of points within some distance of the plane and the average distance of points within some distance of the plane to the plane. The first two did not work particularly effectively, as most of the images is taken up by points that are not the road. I was not able to implement any method to detect the road by its texture or any other features and so the third metric worked the most effectively. The third metric works most effectively as the road should be the largest, flattest feature in the image and hence give the best result when matched with compared to other, smaller flat features such as houses or pavements.

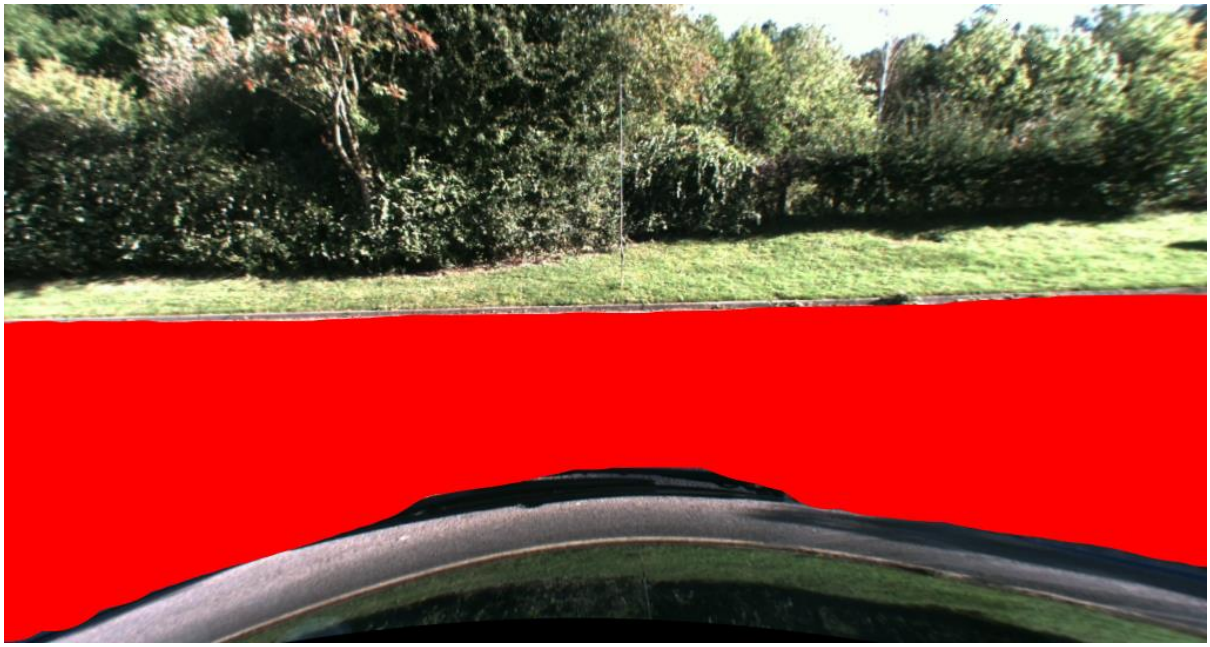
Plane Selection logic

Picking points from anywhere in the image to generate a plane from caused problems as many features exist at high elevations that they can cause very poor plane fits, yet high suitability by the chosen metrics. Picking planes from just in front of the car (assumed road) and ensuring that only roughly flat planes are tested helps to ensure that only roads are detected.

Solution Quality

In order to assess the quality of the images another Python script comparison.py was used, which will return total percentage difference between the expected result and the actual result. Far away portions are not needed as braking distance easily exceeds the distance available so comparisons

will be performed on images only up to 400px from the bottom of the picture. An example comparison result for testing would be:



When compared with the filled-in version generated by the script:

Then the percentage errors are gotten by the comparison.py script. The results from 15 randomly chosen images are shown below, illustrating how the algorithm handles different types of terrain.

Results

Name	Excess / %	Missed / %
1506942490.477398	31.2	31.5
1506942719.476793	27.5	44.3
1506942936.481877	43.6	10.0
1506943018.482395	267.5	0.0
1506943099.493660	55.3	55.5
1506943108.480176	79.7	11.7

1506943196.489851	75.9	60.1
1506943343.478088	33.3	37.4
1506943411.480338	10.7	47.9
1506943470.485110	103.1	0.9
1506943603.484243	69.8	0.5
1506943713.479255	16.2	62.2
1506943829.379867	31.0	49.1
1506943891.402059	0.4	74.3

Hence the mean image will be drawn 60.4% too large and 34.7% of the target area missing.

Code Sourcing

The provided code segments were used throughout the project.