# Vanishing Point Detection using a DSAC

Anusha Chattopadhyay

Computer Vision: 3D Reconstruction 2023

**Abstract**

An attempt in Vanishing Point Detection using a Differentiable RANSAC Model. After writing a script that is based on the paper "A new Approach to Vanishing Point Detection in Architectural Environments" by Prof Dr Carsten Rother . An attempt was made building a DSAC model by using it as a the model's sample hypothesis function.

## 1 Introduction

This Project consisted of 2 phases, building the initial Vanishing Point Detection Script using a RANSAC approach and using it as the Sample Hypothesis of a DSAC model
The first phase consisted of 3 steps

- Line detection
- The Accumulation Step
- The Search Step

This was used as the Sample Hypothesis for the DSAC model. The York Urban Line Segment Database for used for training

## 2 Previous Work

An attempt was to recreate the process as described in the paper: "A new Approach to Vanishing Point Detection in Architectural Environments" by Prof Dr Carsten Rother. However certain methods and checks may not match as will be described later on.

# 3 Repository Structure

- Demo.pynb is the main implementation, most functions and scrips are in here as well
- /Data - Stores the training data, if the .zip file is not there please download mentioned in the readme file in the /Data folder
- /Predictions - Stores predictions and /plots stores the visualizations
- If rerun, to plot the predictions, please run plot predictions.py

# 4 Vanishing Point Detection

Below shall describe the 3 steps

## 4.1 Line Detection

The paper had a brief discussion on which method to use for line detection. After trying various methods for line detection the process was continued using Image contours to detect lines. This may not be the same as the paper.

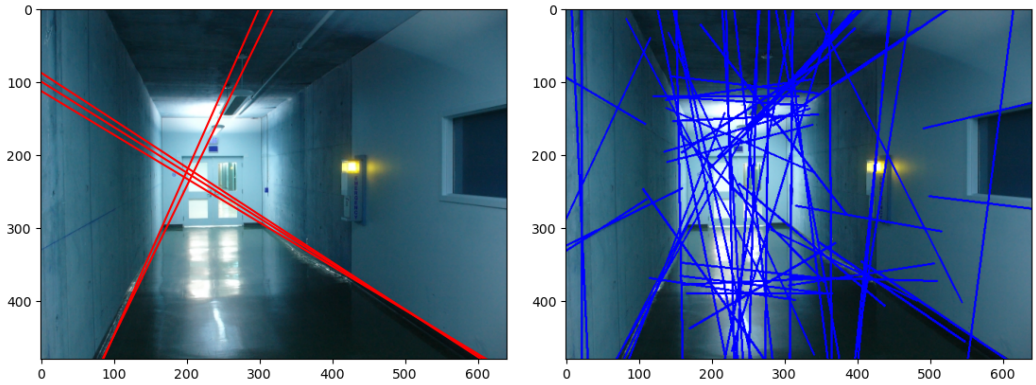


Figure 1: Line Detection: 1: Hough Transform 2.Contour Detection

### 4.1.1 Accumulator cells

The accumulator cells were calculated as the intersection of these detected lines

# 5 The Accumulation Step

The accumulator cells were calculated as the intersection of these detected lines.
Each point was then voted on using the equation as same as the paper:

$vote(a) = \sum_{allvotes for a}(w1*(1-(point_to_line_distance(x,y,x1,y1,x2,y2)/ta)))+ w2*(distance(line,a)/maxlength)$

The values were taken as w1=0.3, w2=0.7, ta=5

# 6 The Search Step

Search Step Involved 2 checks with 3 points. The points were labeled a1,ai,aj. A1 was the Cell with the max vote, and the other 2 were randomly chosen. The function runs 10 times or until total vote is greater than threshold vote. If threshold was not reached, the combination with the highest vote was chosen.
The Function does 2 criterion checks for a success:

- Vanishing Line Criterion
- Orthogonality Camera Criterion

## 6.1 Orthogonality Camera Criterion

Check if the distance between two points and the angle between them are lower than the thresholds decided. Distance was td=100 and Angle Threshold was t$\alpha$=30

### 6.2 Orthogonality Camera Criterion

We check if the 3 points form a triangle where each angle is less than 90° The paper also included a different criterion if 1 or 2 points were at infinity, but there was no storage of infinity points in the line check as all were in 2 dimensions.

Thus the RANSAC algorithim was as follows:

for tries =10:
ai,aj=random(vanishing points)
if vanishing line criterion is satisfied for (a1,ai),(a1,aj)and(aj,ai):
check if orthogonality criterion matches for a1,ai,aj:
vote = vote(a1)+ vote(aj) + vote(ai)

If vote goes above threshold we stop, or we just store the points that had the highest vote out of the 10 tries
Midpoint of the 3 points are then taken as the answer.

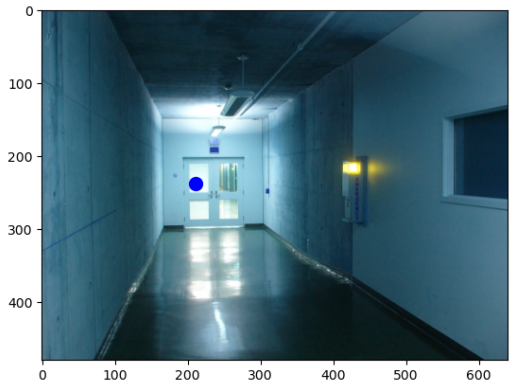


Figure 2: test on picture

## 7 The DSAC Model

Using the above method as the sample hypothesis, the model was constructed. Pytorch was used to construct most of it.

## 7.1 Dataset

The Images were trained on images from The York Urban Line Segment Database. There were around 102 images

This Dataset consisted of images and the endpoints of the lines detected and their association with 3 of the detected vanishing points labeled 1 to 3.

- 1 - First horizontal vanishing point
- 2 - Vertical vanishing point
- 3 - Second horizontal vanishing point

The lines were grouped according to their vanishing point and the intersection of the group was taken as a1,ai,aj and then their intersection was taken as the final vanishing point as the Ground Truth

## 7.2 Model Parameters and functions

## 7.3 Loss Functions

The inlier score was calculated as sum of (1 - sigmoid(inlier beta * (distance - inlier thresh))) The loss was the distance of the true value vs the predicted value squared.

## 7.4 Training Parameters

The Model ran for 5 epochs with 10 iterations over each data point(image and ground truth vanishing point)

# 8 Results and Accuracy

The accuracy of the results was calculated via how many results were at least 100 units away from the detected vanishing point. There were a few Instances of the predicted point being close to the ground truth. However that seemed to be the exception.

More commonly the results were as such

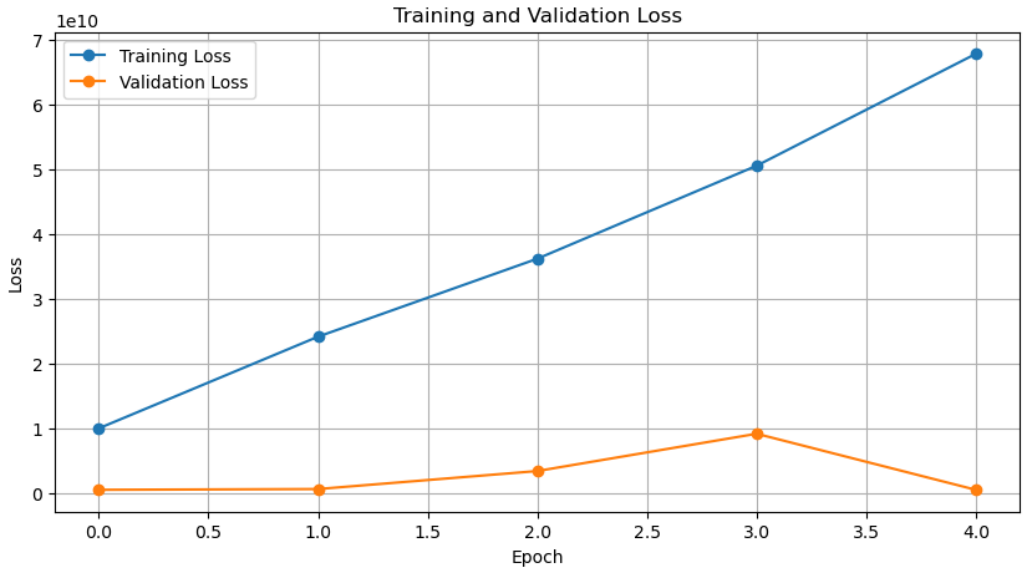


Figure 3: Loss Graph

# 9 Potential Problems and Improvements

The Ground Truth when visualized had most of it's points far outside the image. The method of deciding the Ground Truth may not be correct. Where as the sample hypothesis seemed to detect points well within the image, this was noticed even before training the model.

The sample hypothesis also did not take into account the camera parameters, which may have improved it's accuracy immensely.
A larger data set and training time may have also helped the model
[2] [1]

# References

[1] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-Differentiable RANSAC for camera localization. In *CVPR*, 2017.

Figure 4: Good:Prediction,Ground Truth

Figure 5: Bad and Common:Prediction,Ground Truth

[2] Carsten Rother. A new approach to vanishing point detection in architectural environments. *Image and Vision Computing*, 20(9):647–655, 2002.