1. **Objective**

The goal of this project was to detect the vanishing point in images by analyzing line intersections. I applied different clustering and line fitting techniques, including KMeans, DBSCAN, and RANSAC, to improve the accuracy of the vanishing point detection.

2. Methods Used

1. **KMeans Clustering:**
   I initially used KMeans clustering to group the detected line intersections into clusters. This method required specifying the number of clusters and was sensitive to the chosen number, which affected its performance.

2. **DBSCAN Clustering:**
   I then applied DBSCAN (DensityBased Spatial Clustering of Applications with Noise), which does not require specifying the number of clusters and is better suited for handling noise and varying cluster sizes.The largest cluster from DBSCAN was used to estimate the vanishing point by calculating the mean of its points.

3. **RANSAC Method:**
   To further refine the results, I used RANSAC (Random Sample Consensus) to fit a robust line to the intersection points from the largest DBSCAN cluster. This method is designed to be robust against outliers. When there were enough inliers, I computed the vanishing point based on these inliers. If there weren't enough inliers, I used the vanishing point from DBSCAN as a fallback.

3. My approach:

In my approach to vanishing point detection, I initially employed KMeans clustering, but found it inadequate due to its sensitivity to the number of clusters and its poor handling of noise. To address this, I switched to DBSCAN, which offered better performance by clustering based on density and effectively managing noise. I then refined the results further using RANSAC, which fit a robust line to the largest DBSCAN cluster, providing a more accurate vanishing point by leveraging inliers. When RANSAC did not yield sufficient inliers, I used the DBSCAN result as a fallback. By combining the results from both DBSCAN and RANSAC, I was able to achieve a more reliable and precise vanishing point detection.

4. Results

- **Initial Results with KMeans:**
- My initial attempts with KMeans clustering yielded inconsistent results. The method's sensitivity to the number of clusters and its handling of noise led to suboptimal vanishing point detection.

- **Improved Results with DBSCAN:**
  Switching to DBSCAN improved the results. DBSCAN's ability to handle noise and identify clusters based on density allowed for more accurate vanishing point detection. I calculated the vanishing point as the mean of points in the largest cluster, which was an improvement over KMeans.

- **Best Results with RANSAC:**
  Using RANSAC provided the best results. By fitting a robust line to the largest cluster of intersections, I was able to accurately determine the vanishing point. When RANSAC identified enough inliers, I used these inliers to compute the vanishing point. If not enough inliers were found, I used the DBSCAN result. Combining the DBSCAN and RANSAC results by averaging them gave the most accurate and reliable vanishing point.

5. Challenges Faced and Solutions

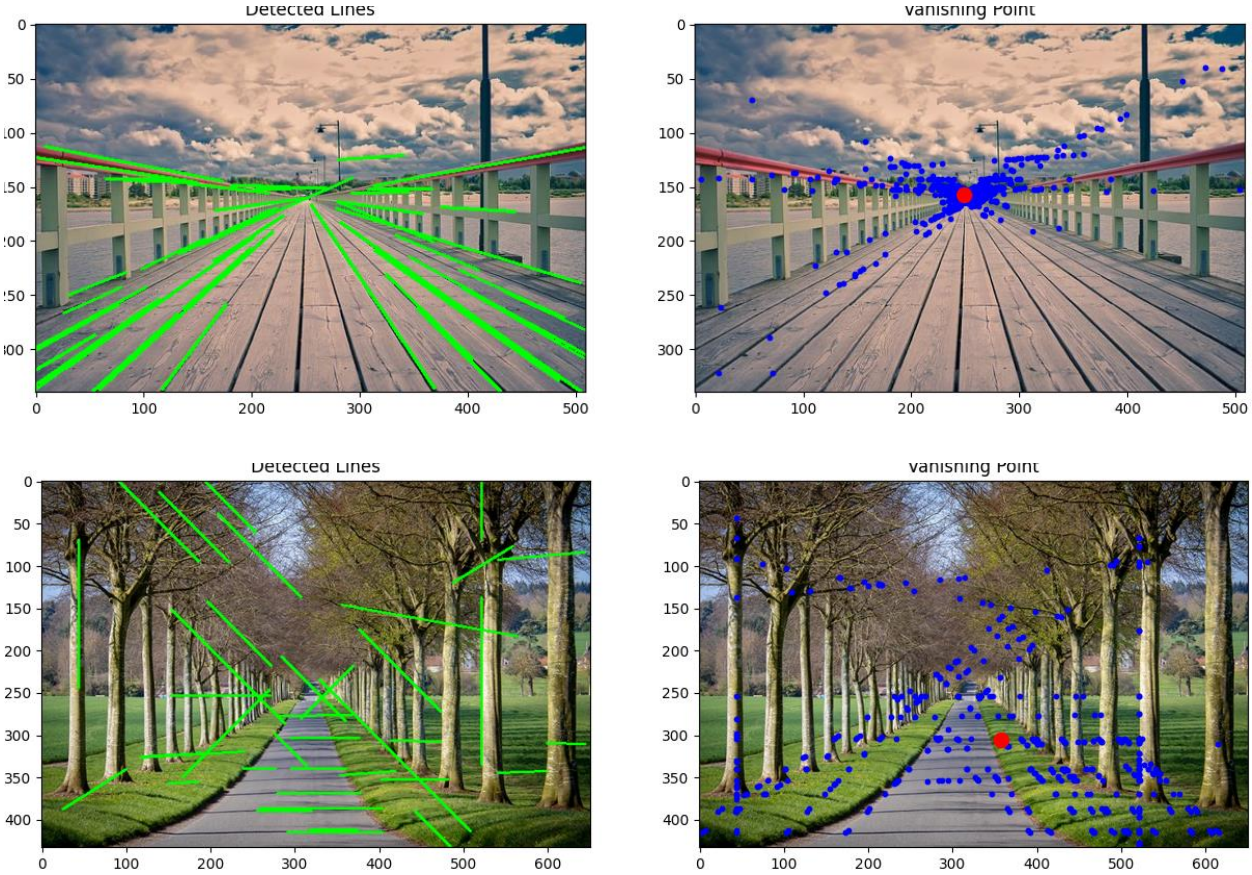
- **Challenge: Clustering Sensitivity with KMeans**
  Issue: KMeans was sensitive to the number of clusters specified and struggled with noise, which resulted in inaccurate vanishing point detection.
  Solution: I moved to DBSCAN, which does not require specifying the number of clusters and is better suited for handling noise. This change improved the accuracy of the clustering results.

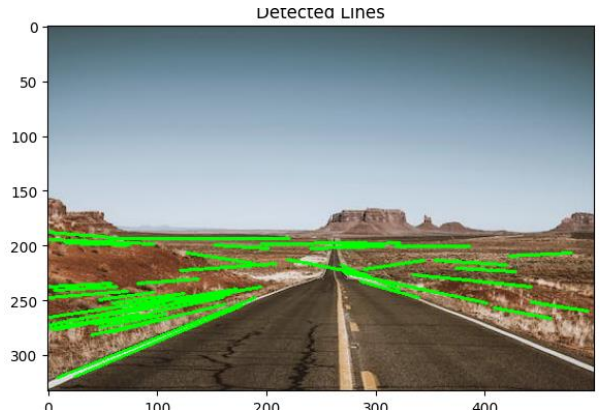
- **Challenge: Noise Handling**
  Issue: Both KMeans and initial DBSCAN attempts were affected by noise, which impacted the accuracy of line intersection detection and clustering.
  Solution: I finetuned DBSCAN parameters (such as `eps` and `min_samples`) to better handle noise and accurately cluster the intersection points. This adjustment helped in obtaining more reliable results.

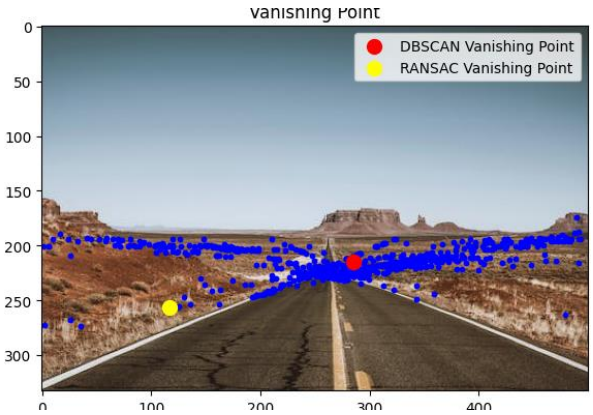
- **Challenge: Inliers Identification with RANSAC**
  Issue: RANSAC's effectiveness depended on the number of inliers. In cases where there were few inliers, the method's robustness was limited.
  Solution: I used a fallback approach where, if RANSAC did not provide sufficient inliers, I relied on the DBSCAN result. Additionally, I combined the results from both methods by averaging them to achieve a more reliable vanishing point.

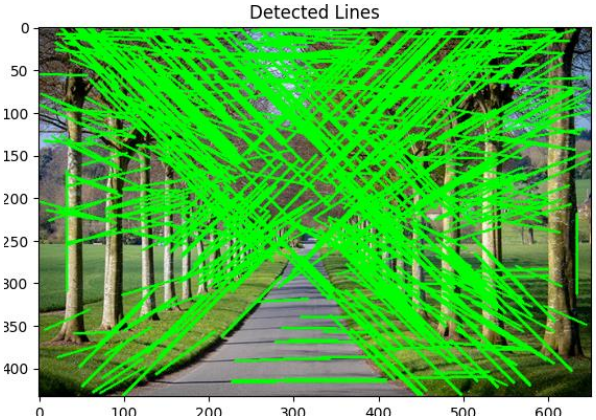
<u>Results screenshots</u>:

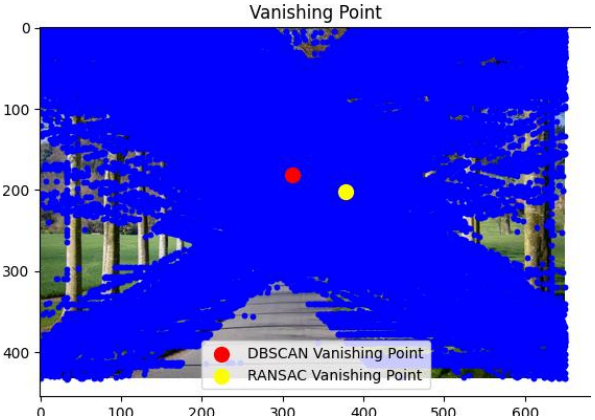
Trials:

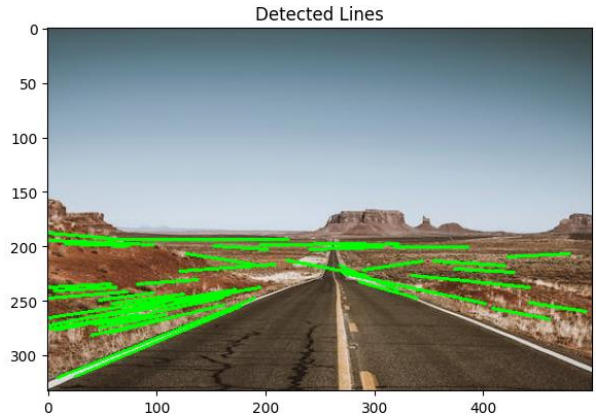


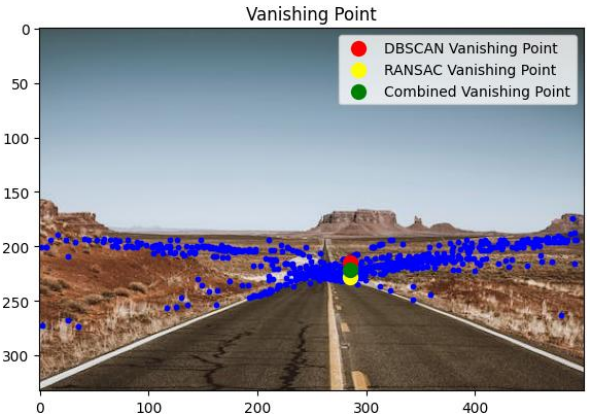

Detected Lines

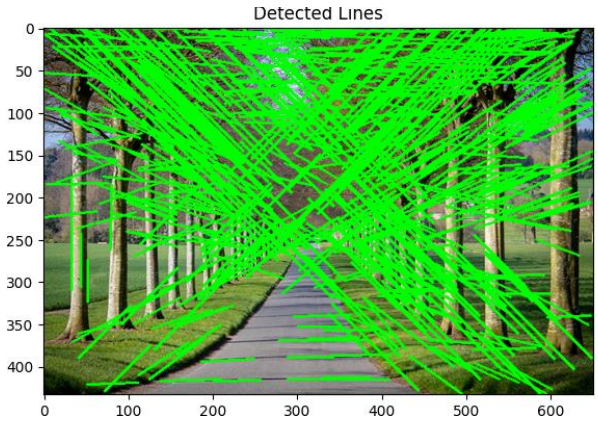

Vanishing Point
DBSCAN Vanishing Point
RANSAC Vanishing Point

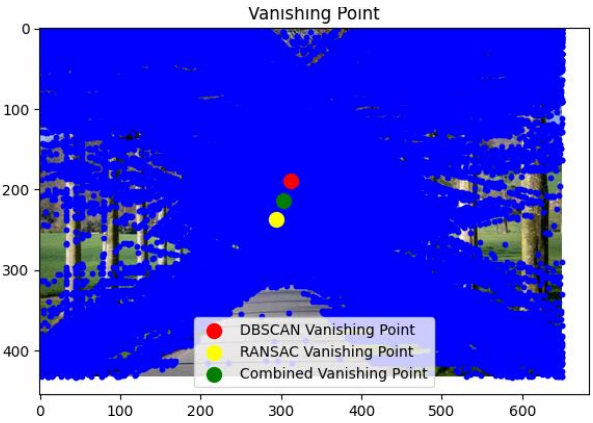

Detected Lines

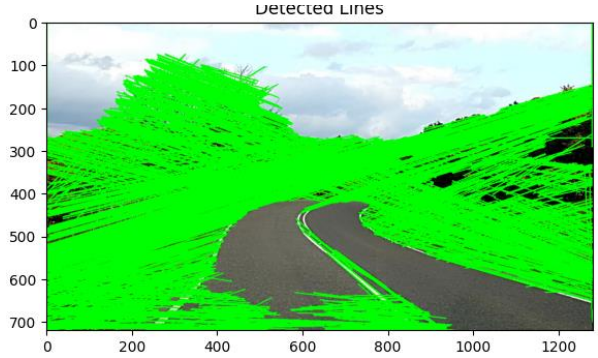

Vanishing Point
DBSCAN Vanishing Point
RANSAC Vanishing Point

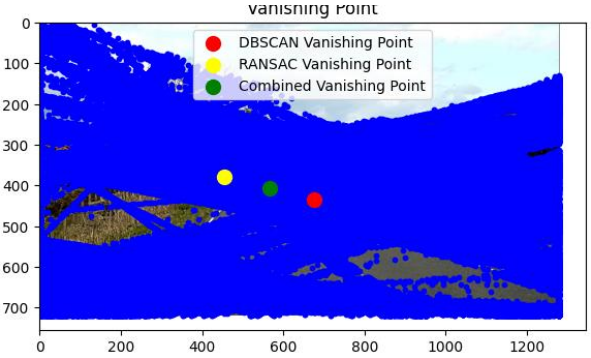

Detected Lines


Vanishing Point
DBSCAN Vanishing Point
RANSAC Vanishing Point
Combined Vanishing Point

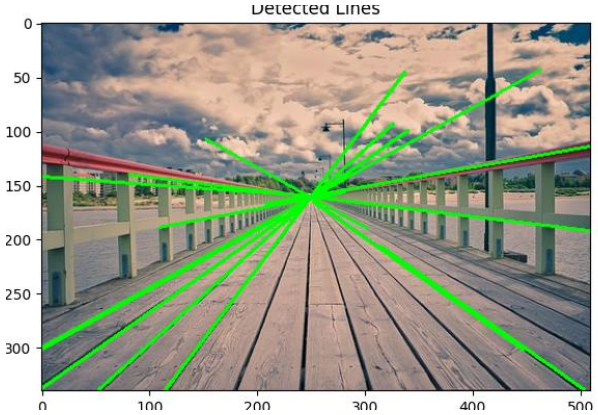


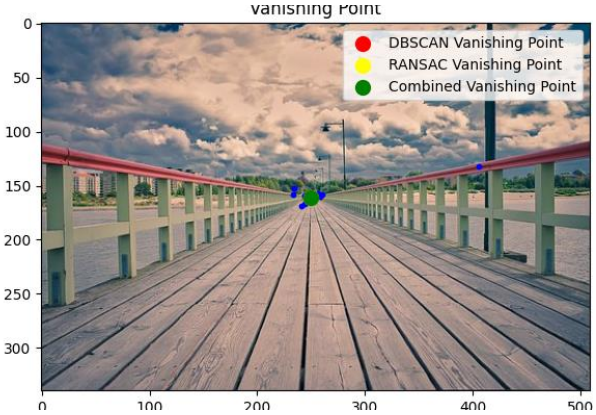


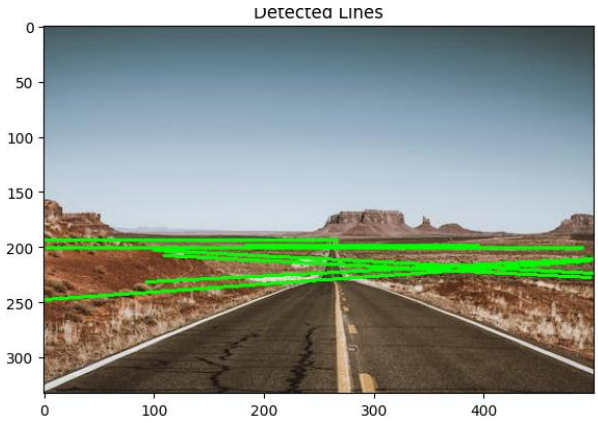


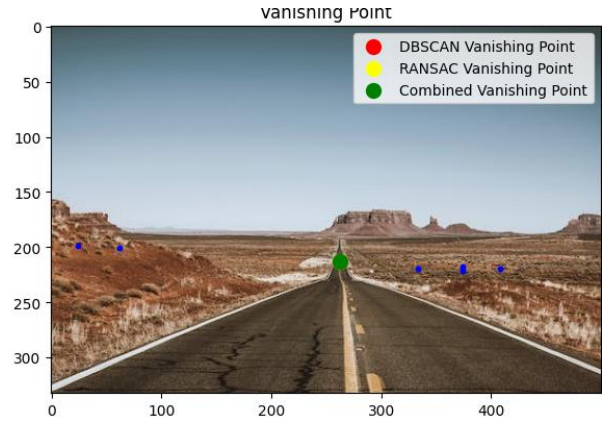


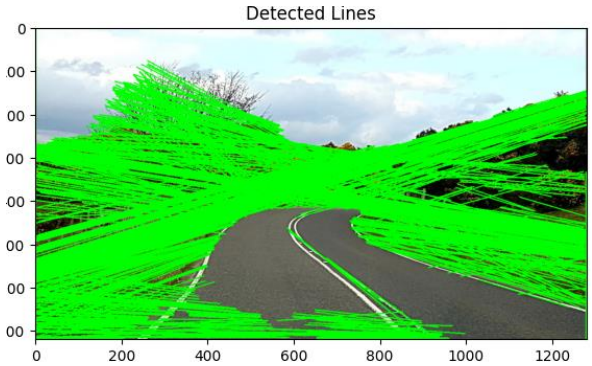
After optimization:

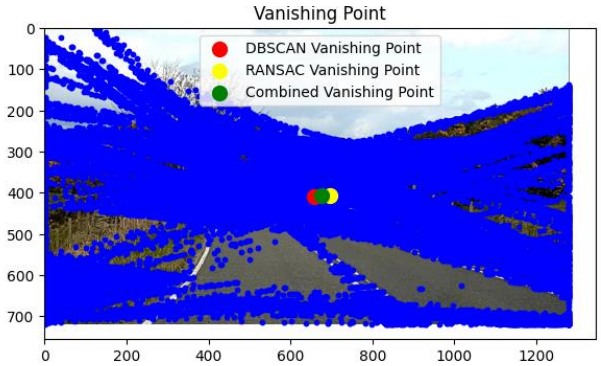





Detected Lines


Vanishing Point
DBSCAN Vanishing Point
RANSAC Vanishing Point
Combined Vanishing Point


Detected Lines


Vanishing Point
DBSCAN Vanishing Point
RANSAC Vanishing Point
Combined Vanishing Point