Analysis of Vehicle Motion Tracking

Patel Preet
School of Engineering & Applied Science
Computer Science & Engineering
Ahmedabad, India
preet.p3@ahduni.edu.in

Hingu Dhruv
School of Engineering & Applied Science
Computer Science & Engineering
Ahmedabad, India
dhruv.h@ahduni.edu.in

Patel Het
School of Engineering & Applied Science
Computer Science & Engineering
Ahmedabad, India
het.p5@ahduni.edu.in

Panchal Dhruvesh
School of Engineering & Applied Science
Computer Science & Engineering
Ahmedabad, India
dhruvesh.p@ahduni.edu.in

***Abstract*—This report presents an analysis of vehicle motion tracking using coordinate data obtained from aerial images provided by the AU Drone Dataset. Various methods were explored to interpret the movement of vehicles over time, including trajectory plotting and angle calculations. Additionally, improvements in tracking accuracy were sought through the application of trigonometric functions. The report concludes with a refined approach incorporating vector analysis for determining vehicle direction. We categorize vehicle movements into eight directions: North (N), East (E), West(W), South(S), North-East (N-E), South-East(S-E), North-West(N-W), and South-West(S-W). Python code snippets are provided to illustrate the implementation of these techniques.**

***Index Terms*—Vehicle motion tracking, trajectory plotting, angle calculations, vector analysis, Python code**

I. Introduction

The accurate tracking of vehicles' motion is crucial in numerous applications, including traffic monitoring, surveillance, and autonomous navigation systems. This report explores methodologies for analyzing vehicle movement based on coordinate data extracted from captured frames. The objective is to develop robust techniques for interpreting the trajectories and direction of vehicles over time.

II. Initial Approach

The initial approach involved plotting the trajectories of vehicles based on their coordinates extracted from frames. Python code snippets were developed to visualize the motion using the Matplotlib library. However, it was observed that this approach had limitations in accurately representing the movement, particularly in scenarios with rapid changes.

A. Plotting Individual Tracks

The first step was to plot the trajectory of individual vehicles from the captured frames. Python code was written to read coordinate data from CSV files and generate trajectory plots for specific vehicle tracks. However, it was noted that this method lacked comprehensive insight into the overall motion patterns.

B. Combining Trajectories

To gain a broader perspective, the trajectories from multiple frames were combined and plotted together. This approach aimed to provide a holistic view of vehicle movement over time. Nevertheless, the complexity of interpreting the combined trajectories highlighted the need for more sophisticated analysis techniques.

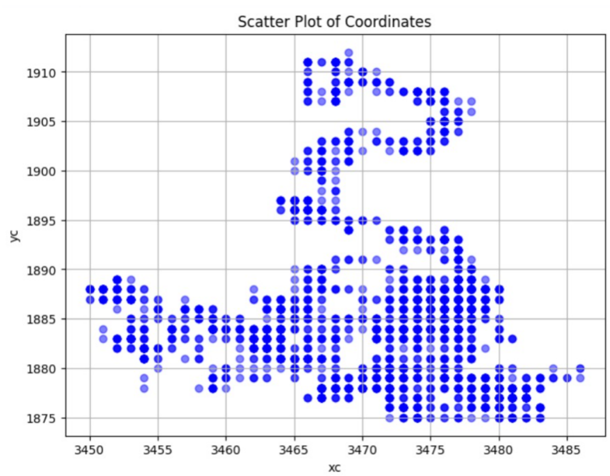

Fig. 1. Plotting the trajectory of track 12

III. Refined Approach

In response to the limitations of the initial methods, a refined approach was developed to enhance tracking accuracy and interpretational clarity. This involved incorporating vector analysis to determine the direction of vehicle motion. The subsequent sections detail the implementation of this approach using Python code.

A. Vector Analysis

Vector analysis was utilized to calculate the direction of vehicle movement based on sequential coordinate data.

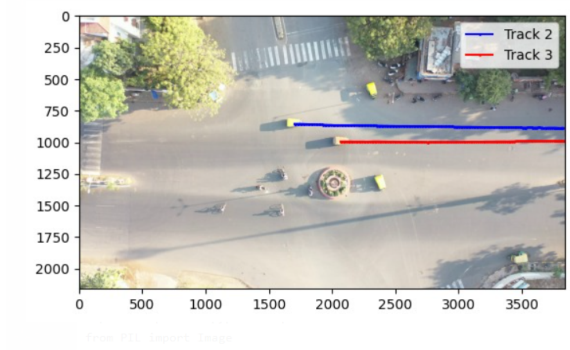

Fig. 2. Tracking objects within image

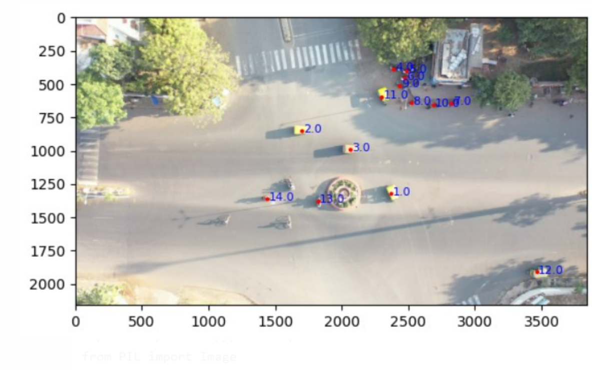

Fig. 3. Labelling all track present in image

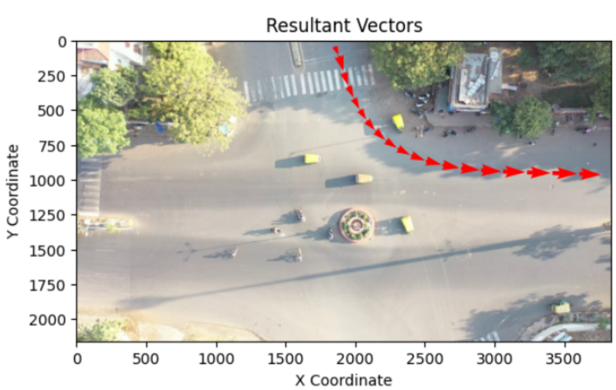

Fig. 4. Theta for track number 165

IV. Direction Analysis Implementation

article amsmath

We have used the following angles received by finding the direction vector with respect to zero degrees and then used the following division for assigning direction:

$$\begin{aligned}
&\theta = \theta\%360 \\
&\text{if } \theta < -22.5: \\
&\quad \theta+ = 360 \\
&\text{elif } \theta \geq 337.5: \\
&\quad \theta- = 360
\end{aligned}$$

The angles are decided this way:

$$\begin{aligned}
&(-22.5, 22.5) : E, \\
&(22.5, 67.5) : NE, \\
&(67.5, 112.5) : N, \\
&(112.5, 157.5) : NW, \\
&(157.5, 202.5) : W, \\
&(202.5, 247.5) : SW, \\
&(247.5, 292.5) : S, \\
&(292.5, 337.5) : SE \\
&\}
\end{aligned}$$

A. Trajectory Plotting with Vector Analysis

The following Images show the output snippet demonstrates the plotting of vehicle trajectories using vector analysis to determine direction:

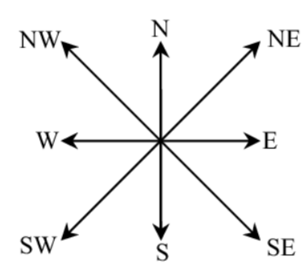

Fig. 5. 8 Directionsthat we are predicting

V. Conclusion

This report presented an analysis of vehicle motion tracking techniques based on coordinate data extracted from captured frames. Initially, trajectory plotting methods were explored, but limitations in accuracy and interpretability prompted the development of a refined approach incorporating vector analysis. The implementation of this approach using Python code demonstrated improved tracking accuracy and directional insight. Further enhancements could be pursued to optimize performance and address specific application requirements.

Track	Srno	Initial Position [x y]	Final Position[x y]	Velocity	Angle	Direction
165	1.0	[1853 42]	[1882 107]	0.0	-65.95577673063218	SE
165	4.0	[1884 112]	[1917 221]	14.5777487	-73.15619144827258	S
165	7.0	[1919 230]	[1949 322]	25.83416235	-71.93952806380082	S
165	10.0	[1951 328]	[1988 410]	20.41206673	-65.71417484434872	SE
165	13.0	[1991 416]	[2030 493]	18.33864464	-63.13808215559732	SE
165	16.0	[2034 498]	[2080 570]	17.64894614	-57.425942865427494	SE
165	19.0	[2083 575]	[2142 645]	17.92917803	-49.87389642220477	SE
165	22.0	[2147 650]	[2215 715]	19.85577417	-43.70783311375007	SE
165	25.0	[2221 719]	[2300 772]	20.39632698	-33.857110141660684	SE
165	28.0	[2306 776]	[2395 822]	20.2177977	-27.332359492167686	SE
165	31.0	[2403 824]	[2502 862]	21.62725007	-20.998719034387015	E
165	34.0	[2509 864]	[2620 892]	22.9483238	-14.157632649677245	E
165	37.0	[2628 894]	[2751 916]	24.52910275	-10.140793146112795	E
165	40.0	[2760 917]	[2894 931]	26.76643357	-5.96448710125312	E
165	43.0	[2904 931]	[3051 941]	29.44555508	-3.891676294080756	E
165	46.0	[3062 941]	[3222 946]	32.1736906	-1.7899106082460694	E
165	49.0	[3234 947]	[3404 952]	34.73206452	-1.6846843178962914	E
165	52.0	[3416 952]	[3596 957]	36.37375203	-1.5911402711945872	E
165	55.0	[3610 958]	[3765 964]	38.98503351	-2.2167943407401904	E

Fig. 6. Output Data for Track : 165

VI. References

@inproceedingszhu2018behavior, title=Behavior Recognition of Moving Objects Using Deep Neural Networks, author=Zhu, J. and Lin, W. and Sun, K. and Hou, X. and Liu, B. and Qiu, G., booktitle=2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced, year=2018