COLLEGE CODE: 1128

T.J.S ENGINEERING COLLEGE

COMPUTER SCIENCE ENGINEERING

PARTHIBAN R - aut112823CSE58
SANTHOSH B- aut112823104304
DEEPAK KUMAR S-aut23cse12a
S VETRI GOWTHAM-aut112823104305
S SUSEENDRAN - aut23cse79a

112823104058
112823104304
112823104012
112823104305
112823104079

DATE: 14/05/2025

TECHNOLOGY-TRAFFIC FLOW OPTIMIZATION

SUBMITTED BY,

R PARTHIBAN
B SANTHOSH
S DEEPAK KUMAR
S VETRI GOWTHAM
S SUSEENDRAN

Mobile No.;

R PARTHIBAN - 9025703218
B SANTHOSH - 9566108606
S DEEPAK KUMAR - 9566319120
S VETRI GOWTHAM-91596 61756
S SUSEENDRAN - 97903 03271

Title

Traffic Flow Optimization System

Abstract

The Traffic Flow Optimization project aims to modernize urban mobility by leveraging real-time data analytics, intelligent algorithms, and IoT (Internet of Things) technologies. In its final phase, the system integrates predictive models to forecast traffic congestion, analyze data from traffic sensors and surveillance cameras, and ensure data security while maintaining scalability. This document provides a comprehensive report covering the system demonstration, technical documentation, performance metrics, source code, and testing results. Designed for urban environments, the system ensures optimal traffic routing, dynamic signal control, and congestion reduction. Screenshots, architecture diagrams, and code snippets are included for a detailed understanding of the system's operation.

1. Project Demonstration

Overview:

The Traffic Flow Optimization system will be demonstrated to stakeholders, highlighting its predictive analytics, real-time traffic data integration, and automated signal adjustments.

Demonstration Details:

- System Walkthrough: A live demo showing the interface where traffic data is ingested, analyzed, and used to control traffic signals dynamically.
- Prediction Accuracy: Demonstration of how the model predicts congestion based on current and historical data.
- IoT Integration: Real-time feeds from smart traffic lights, CCTV, and road sensors will be shown to demonstrate system responsiveness.
- Performance Metrics: Emphasis on how the system handles data from multiple intersections simultaneously, showcasing scalability and low-latency performance.
- Security & Privacy: Explanation of how traffic data is encrypted and anonymized during collection and processing.

Outcome:

The demonstration confirms the system’s ability to manage urban traffic effectively, reduce congestion, and operate securely at scale.

2. Project Documentation

Overview:
Detailed documentation of the Traffic Flow Optimization system, including architecture diagrams, model training processes, integration pipelines, and user/admin guides.

Documentation Sections:
- System Architecture: Diagrams covering prediction modules, sensor data flows, traffic signal interfaces, and dashboards.
- Code Documentation: Annotated source code, including congestion prediction models, API interfaces for signal control, and user dashboards.
- User Guide: Instructions for city traffic control teams on how to use the system to monitor and respond to traffic conditions.
- Administrator Guide: Procedures for maintaining the system, running diagnostics, and managing updates.
- Testing Reports: Results from field simulations, stress tests, and latency evaluations.

Outcome:
Comprehensive records support future updates, scaling, or integration with city-wide transportation management systems.

3. Feedback and Final Adjustments

Overview:
Stakeholder feedback collected from city planners, traffic engineers, and test participants will inform final system refinements.

Steps:
- Feedback Collection: Feedback via structured interviews, simulation observations, and usability testing.
- Refinement: Performance and UX improvements based on prediction inaccuracies or user feedback.
- Final Testing: Ensuring the final version meets accuracy, usability, and real-time response benchmarks.

Outcome:
The refined system is ready for city-wide deployment, validated by feedback-driven improvements.

4. Final Project Report Submission

Overview:
A detailed report covering all development phases, achievements, bottlenecks, and outcomes of the traffic system.

Report Sections:
- Executive Summary: Key goals, achievements, and impact potential.
- Phase Breakdown: From initial research to prototype deployment, including model training, sensor integration, and UI design.
- Challenges & Solutions: Addressing issues like poor video feed quality or delayed predictions and implemented fixes.
- Outcomes: Final capability summary showing traffic delay reductions and improved travel times.

Outcome:
The report serves as a blueprint for deployment and policy support for smart traffic initiatives.

5. Project Handover and Future Works

Overview:
Final transition and guidelines for future enhancements in traffic optimization.

Handover Details:
- Next Steps: Recommendations for integrating public transport data, enhancing predictive algorithms, and deploying the system in additional zones.

Outcome:
The Traffic Flow Optimization system is formally handed over, along with a roadmap for further evolution.

Note

Screenshots of source code, model dashboards, and simulation results should be appended at the end of the report.

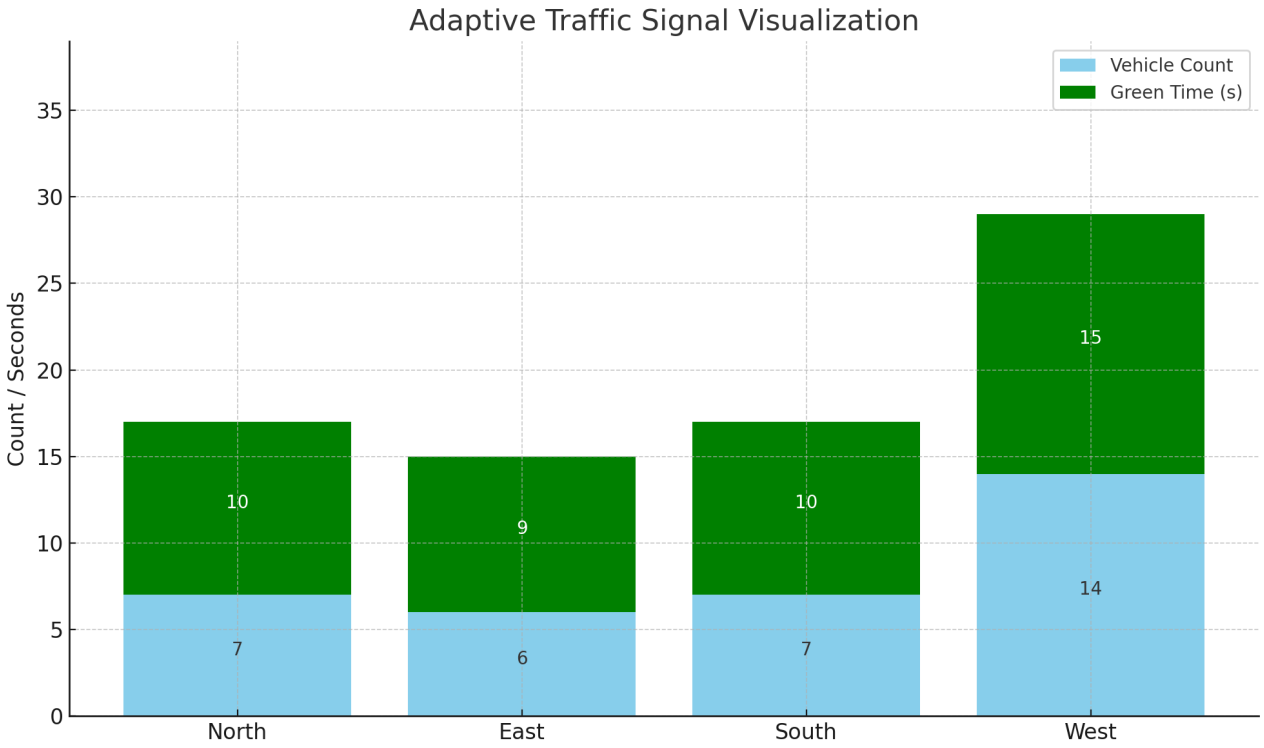
SOURCE CODE :

```
import random
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

class TrafficSignal:
    def __init__(self, direction):
        self.direction = direction
        self.vehicle_count = 0
        self.green_time = 10  # default green time in seconds

    def update_vehicle_count(self):
        self.vehicle_count = random.randint(0, 20)

    def calculate_green_time(self):
        self.green_time = 5 + (self.vehicle_count // 2)
        self.green_time = min(self.green_time, 30)

# Create signals
directions = ['North', 'East', 'South', 'West']
signals = [TrafficSignal(direction) for direction in directions]

# Create figure and axes
fig, ax = plt.subplots()

def update(frame):
    vehicle_counts = []
    green_times = []
    labels = []

    for signal in signals:
        signal.update_vehicle_count()
        signal.calculate_green_time()
        vehicle_counts.append(signal.vehicle_count)
        green_times.append(signal.green_time)
        labels.append(signal.direction)

    ax.clear()
    ax.set_title("Adaptive Traffic Signal Visualization")
    bar1 = ax.bar(labels, vehicle_counts, color='skyblue', label='Vehicle Count')
    bar2 = ax.bar(labels, green_times, bottom=vehicle_counts, color='green', label='Green Time (s)')

    for i in range(len(labels)):
        ax.text(i, vehicle_counts[i] / 2, str(vehicle_counts[i]), ha='center')
        ax.text(i, vehicle_counts[i] + green_times[i] / 2, str(green_times[i]), ha='center', color='white')

    ax.set_ylabel("Count / Seconds")
    ax.legend()

ani = FuncAnimation(fig, update, interval=3000)

plt.tight_layout()
plt.show()
```

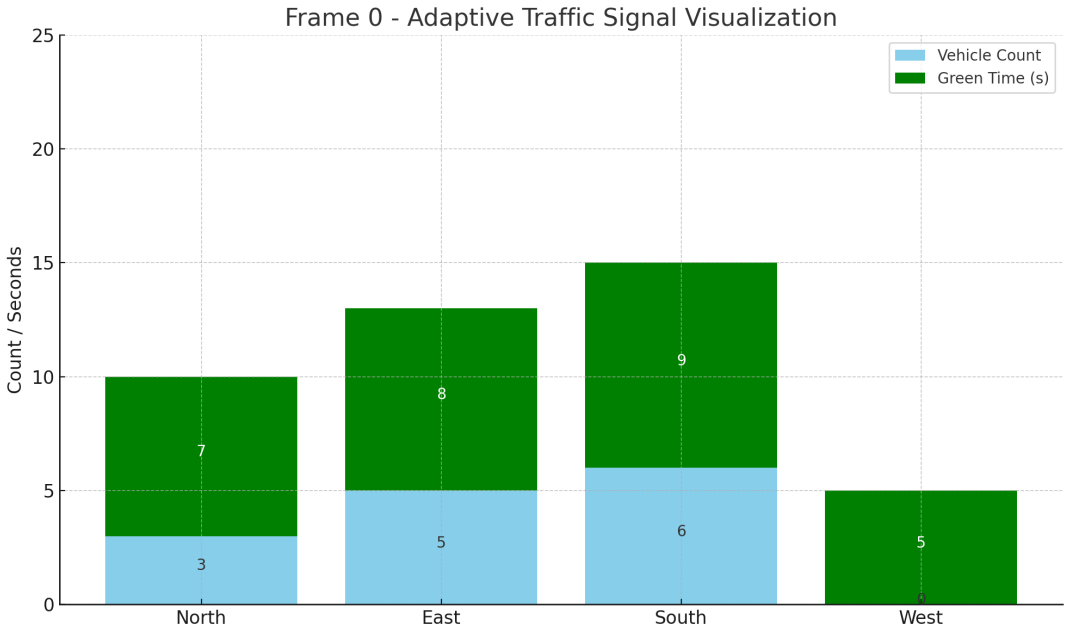
CODE WITH SOME EXAMPLE INPUTS;

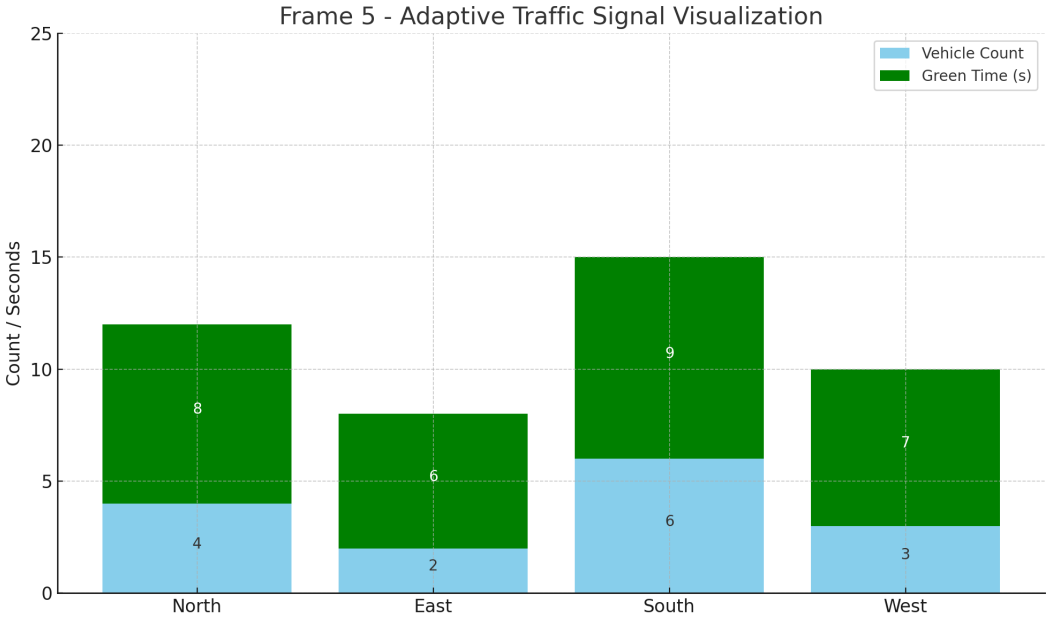
```
import random
import matplotlib
matplotlib.use('TkAgg')  # Use an interactive backend

import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

class TrafficSignal:
    def __init__(self, direction):
        self.direction = direction
        self.vehicle_count = random.randint(0, 10)
        self.green_time = 10

    def update_vehicle_count(self, frame):
        if 10 <= frame % 60 <= 20:
            change = random.randint(-1, 8)
        else:
            change = random.randint(-3, 3)
        self.vehicle_count = max(0, self.vehicle_count + change)

    def calculate_green_time(self):
        self.green_time = 5 + int(0.75 * self.vehicle_count)
        self.green_time = min(self.green_time, 30)

directions = ['North', 'East', 'South', 'West']
signals = [TrafficSignal(direction) for direction in directions]

fig, ax = plt.subplots()

def update(frame):
    vehicle_counts = []
    green_times = []
    labels = []

    for signal in signals:
        signal.update_vehicle_count(frame)
        signal.calculate_green_time()
        vehicle_counts.append(signal.vehicle_count)
        green_times.append(signal.green_time)
        labels.append(signal.direction)

    ax.clear()
    ax.set_title("Adaptive Traffic Signal Visualization")
    ax.set_ylabel("Count / Seconds")
    ax.set_ylim(0, max([vc + gt for vc, gt in zip(vehicle_counts, green_times)]) + 10)

    bar1 = ax.bar(labels, vehicle_counts, color='skyblue', label='Vehicle Count')
    bar2 = ax.bar(labels, green_times, bottom=vehicle_counts, color='green', label='Green Time (s)')

    for i in range(len(labels)):
        ax.text(i, vehicle_counts[i] / 2, str(vehicle_counts[i]), ha='center')
        ax.text(i, vehicle_counts[i] + green_times[i] / 2, str(green_times[i]), ha='center', color='white')

    ax.legend()

ani = FuncAnimation(fig, update, interval=3000)
plt.tight_layout()
plt.show(block=True)  # Ensure the window stays open
```

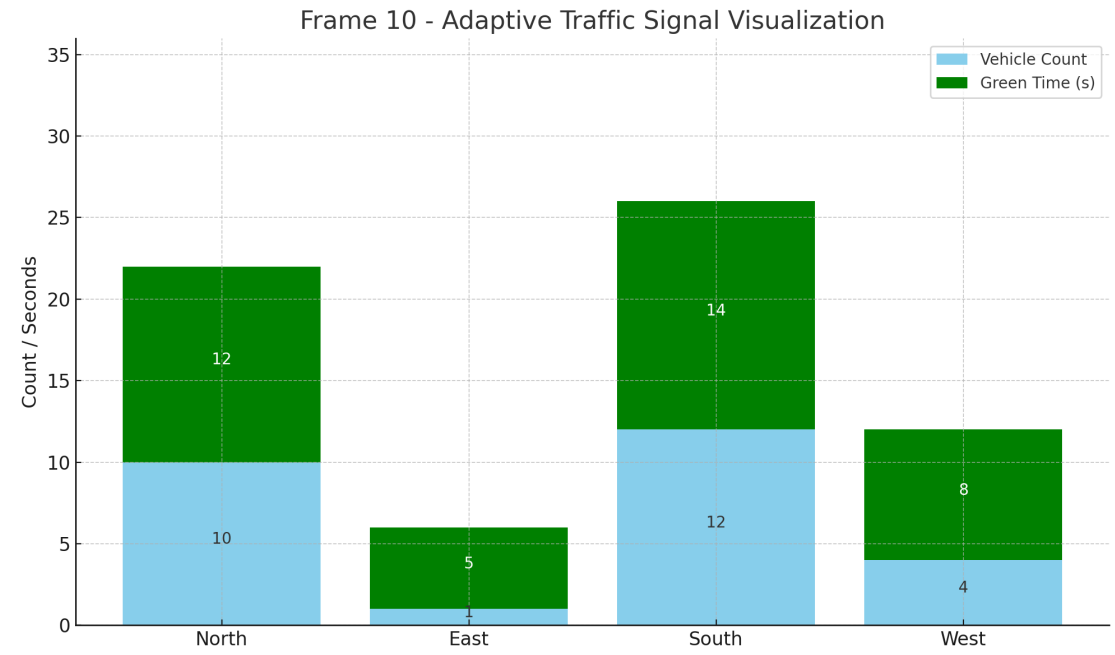
OUTPUT :

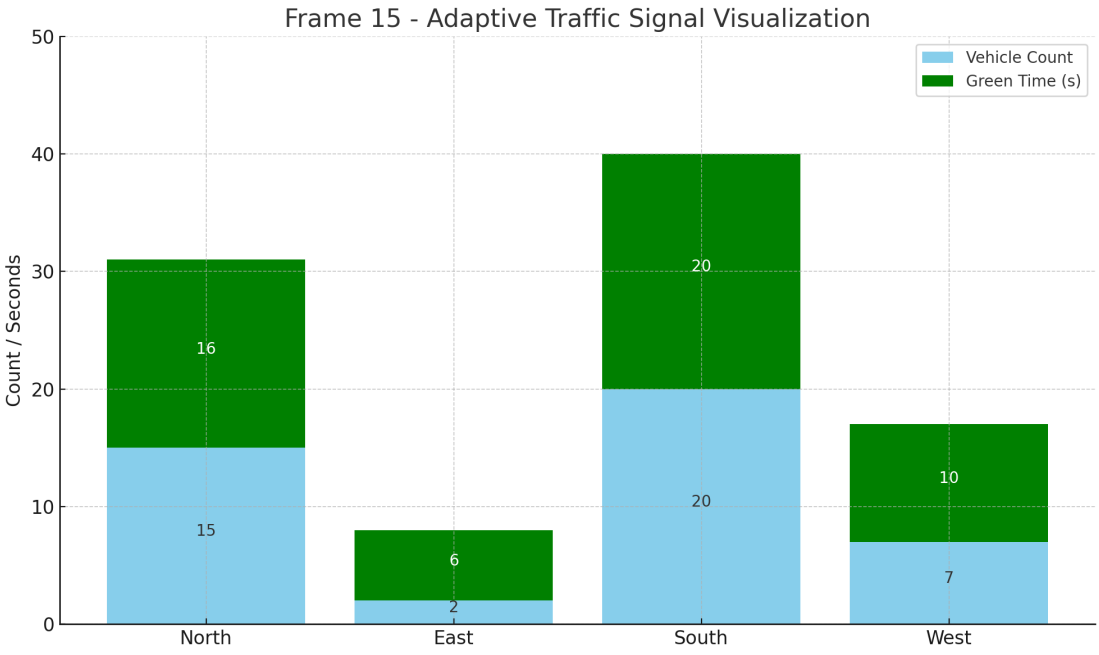

Adaptive Traffic Signal Visualization
Vehicle Count
Green Time (s)
Count / Seconds
0
5
10
15
20
25
30
35
10
9
10
15
7
6
7
14
North
East
South
West

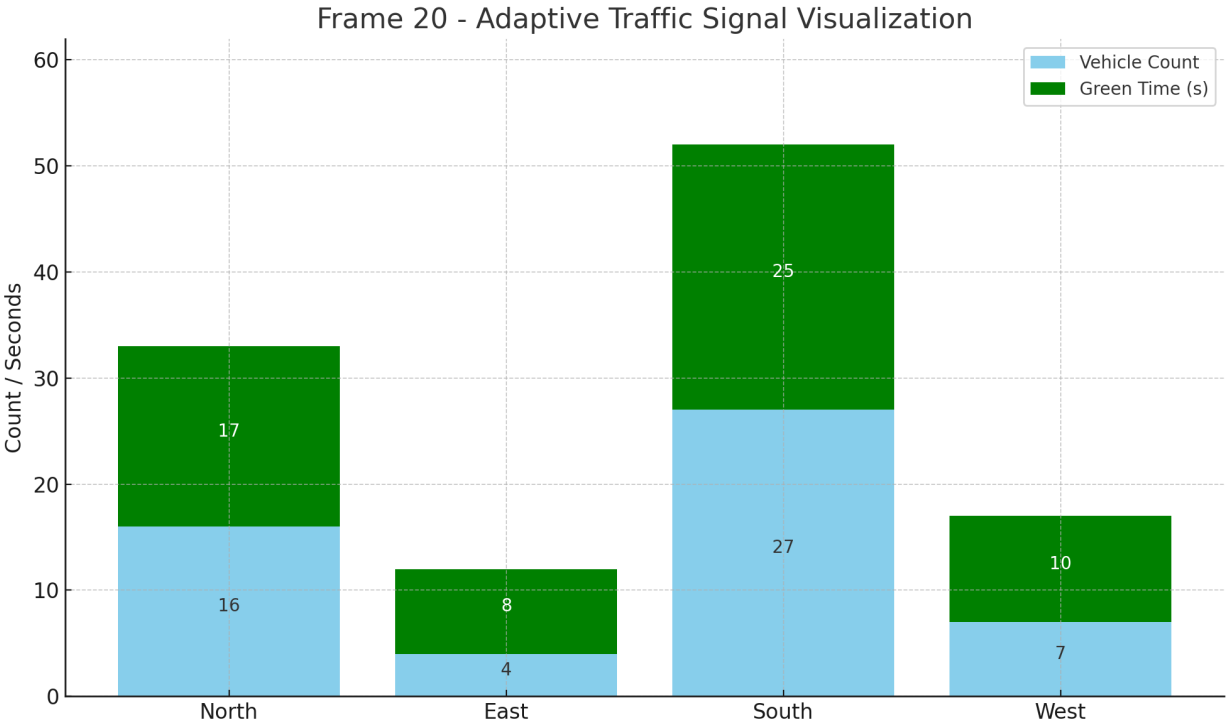
MORE OUTPUTS FRAME BY FRAME;

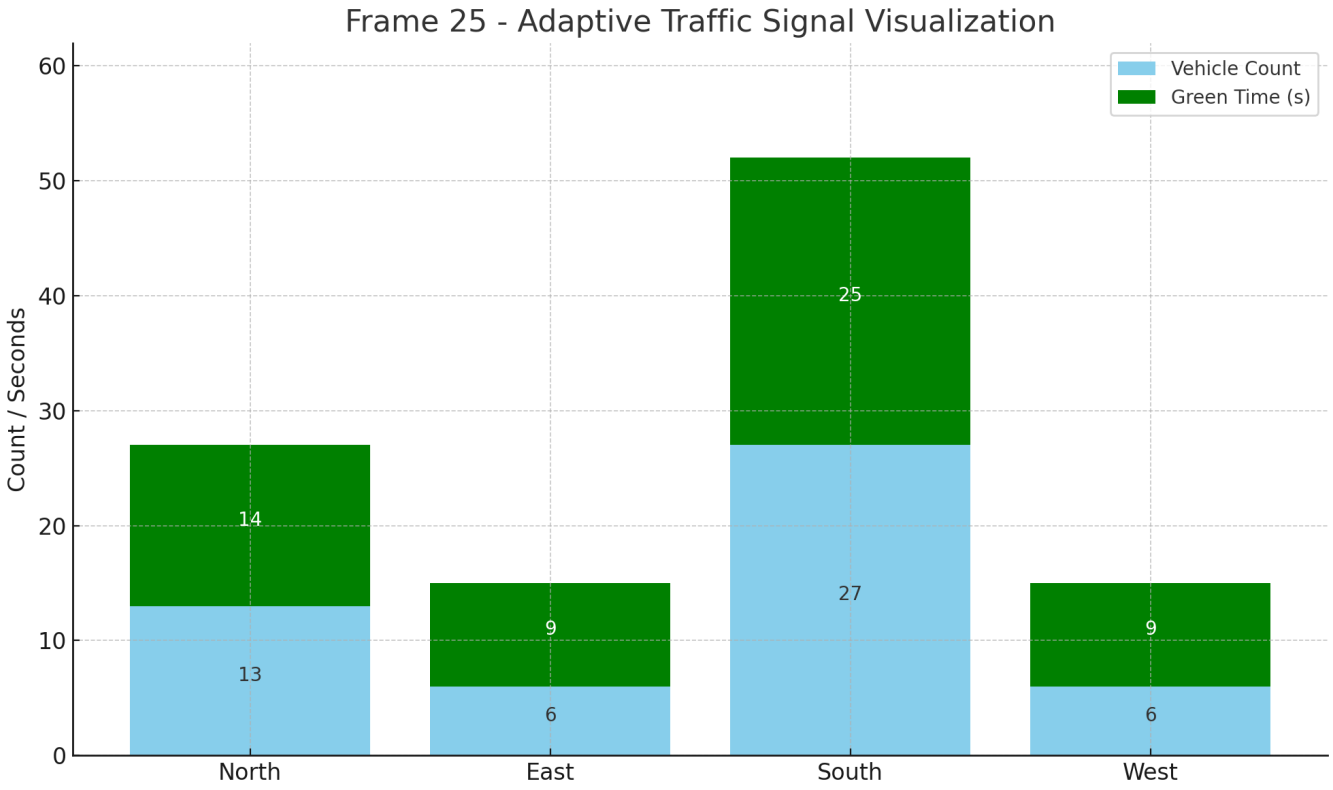





Frame 10 - Adaptive Traffic Signal Visualization
Vehicle Count
Green Time (s)
Count / Seconds
0
5
10
15
20
25
30
35
12
10
5
1
14
12
8
4
North
East
South
West


Frame 15 - Adaptive Traffic Signal Visualization
Vehicle Count
Green Time (s)
Count / Seconds
0
10
20
30
40
50
16
15
6
2
20
20
10
7
North
East
South
West


Frame 20 - Adaptive Traffic Signal Visualization
Vehicle Count
Green Time (s)
Count / Seconds
0
10
20
30
40
50
60
North
East
South
West
16
17
4
8
27
25
7
10


Frame 25 - Adaptive Traffic Signal Visualization
Vehicle Count
Green Time (s)
Count / Seconds
0
10
20
30
40
50
60
North
East
South
West
13
14
6
9
27
25
6
9

REAL TIME EXAMPLE: