Ujjwal Mishra, 2025, 13:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

# Smart Traffic Management Syetem Using Simulation

**Ujjwal Mishra, Mukesh Maurya, Krishna Yadav, Prateek Tiwari, Dr.B.K. Sharma, Assistant professor Nitin Sharma**

Department of Computer Science and Engineering,Nitra Technical Campus Ghaziabad, 201002,India

**Abstract- This study has been undertaken to build a Smart Traffic Management System using two modern technology and tools. Smart Traffic Management System (STMS) is a live Python/Pygame simulation of an urban intersection that models adaptive traffic control. It provides a graphical user interface (GUI) where users can adjust the vehicle spawn rate and the probability of emergency vehicles to create light or heavy traffic scenarios. Unlike fixed-schedule signals, the STMS dynamically alters signal durations in response to simulated real-time traffic conditions. Our simulation similarly uses virtual detectors (vehicle counts) to extend green lights when queues grow and shorten them when approaches clear, mimicking how modern controllers reduce delays and improve travel times tracking system at intersections of roads. The system detects the arrival of emergency vehicles such as ambulance, fire truck, etc. and adjusts traffic lights to speed up their passage, shortening response time. The goal of this STMS simulation is to demonstrate how intelligent signal control can reduce congestion and improve emergency response in a city setting. This conceptual model captures key aspects of smart traffic management in a simulated environment. (Note: this is a simulation– it does not use live camera or sensor feeds, but is designed to mimic their effect.)**

**Keywords- Emergency Priority,Simulation,Car detection, Traffic Density.**

## I. INTRODUCTION

The project works on the real-time adaptive traffic signal management system using simulation, specifically this system mimic the real traffic in virtual environment to solve the problems of real world. The system is designed to adjust green signal durations based on live vehicle counts and traffic patterns and emergency vehicles priorities in emergency, rather than relying on pre-set timers. It works in a simulated environment to demonstrate the real-world feasibility and impact of adaptive signal timing. Its ability to adapt to real-time traffic density in each direction of an intersection. By integrating simulation input, the system can determine the number of vehicles approaching a junction from different directions, evaluate the traffic intensity, and accordingly modify the duration of green signals. Emergency vehicles such as ambulances and fire trucks require immediate right of way to save lives and respond to emergencies. Emergency prioritize vehicle detection adds another layer of responsiveness, ensuring ambulances or fire trucks receive immediate clearance without delay.

### Literature Survey

Traffic is THE headache for any city, especially where people are multiplying faster than rabbits. More cars every year, roads packed to the gills, and average speeds sometimes drop below 10 km/h during rush hour (which, let's face it, feels like walking is faster). Managing this mess right now? It's mostly people staring at data and deciding how long lights should stay green, then telling the local cops what's up. Not exactly cutting edge.

On the smarter side, folks like Thorpe played with reinforcement learning for traffic lights— using neural networks to guess how long cars would wait. He went the RL route with SARSA and all that jazz. Roozemond took the "agents everywhere" approach with intelligent signaling agents and road segment agents trying to outsmart each other for better flow. Stuff gets a bit sci-fi, but hey, no real results to show off.

Some other researchers, like G. Sathya and friends, focused on ambulances—using things like GPS and GPRS to clear a path when seconds matter. And then there's the image processing crowd, who (like us) say cameras can spot cars way better than old pavement sensors.

**Problem Definition**

In this modern era rapid urbanization and AI based technology is every where on the other hand traffic congestion has become a significant concern across cities worldwide. every Field is using the artificial intelligence technology for their productivity however no one is focusing on the roads and Traffic management system, Traditional traffic management systems including fixed-timer signals, manual controls, and basic electronic systems. These outdated approaches lack adaptability resulting in: longer waiting times, inefficient fuel consumption, increased congestion, Inefficiency during traffic volumes, Vehicle idling and fuel consumption, Longer travel time, Higher risk of accidents, Poor handling of Emergency scenarios, No optimization for non-vehicular traffic.

## II. PROPOSED SYSTEM

Here's our fix: real-time video checks how many cars are waiting at every part of an intersection. Four cameras—one for each direction. The system compares the crowds, picks the worst jam, and gives that lane the green light until things even out. Then it does the whole thing again. Basically, it's traffic control that actually notices what's happening, not just running on autopilot. And hey, if you've ever been stuck at a red light on an empty road, you know why this matters.Alright, so here's the gist:

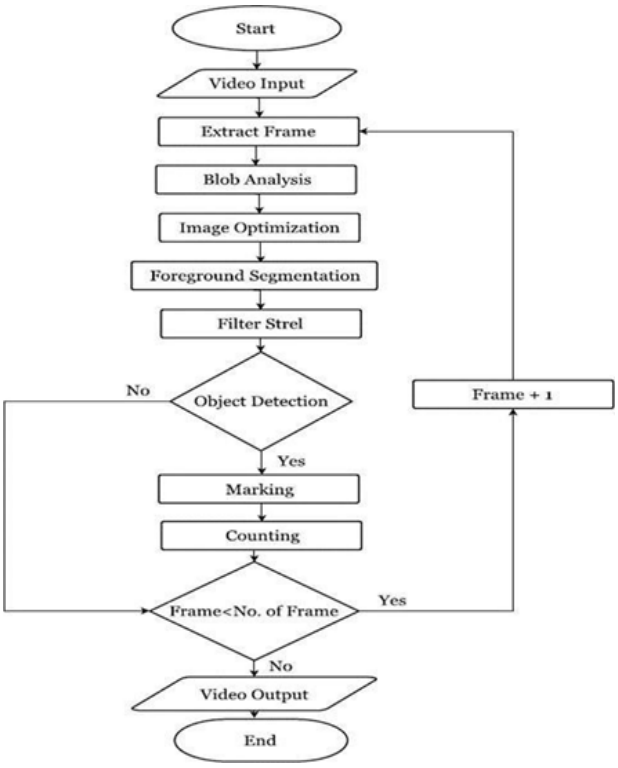
We're gonna slap some HD cameras on poles—like, way up there, about 20 feet up—right by the traffic lights. Those cameras just keep watch, 24/7, tracking all the cars rolling by. Now, the cool bit is, you can throw as many cameras as you want into the mix. Got more roads? No problem, add more cameras. Boom, congestion solution for days.So, the camera feeds live footage straight to a computer (the "CPU," but let's be real, it's just a beefy PC doing the heavy lifting). It chews through the video, frame by frame, running our custom algorithm—think of it like a digital bouncer, counting every car that tries to sneak in. Here's where it gets spicy: Based on how packed each road is, the system fires up a smart traffic light timer. The microcontroller (or, you know, an Arduino if you're DIY-ing) gets a signal from the CPU and flips the lights. When green's about to turn red, another ping goes out. It's all coordinated, like a weirdly nerdy dance.

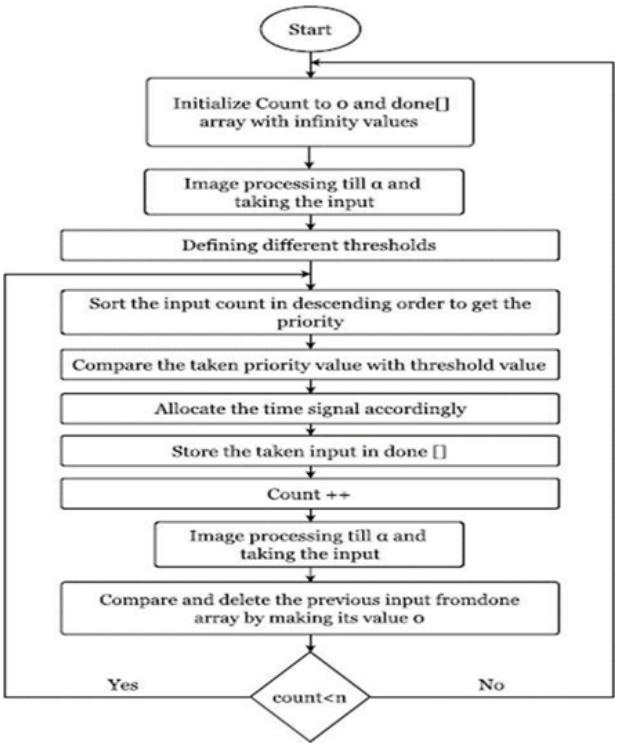
At a gnarly 4-way intersection, each road gets its own camera. The computer counts up cars on every street. Whichever road's the most jammed? That one gets the green light first.

Everyone else waits. It's like survival of the fittest, but for commuters.Now, about the brainy part—the algorithm. We're using a Gaussian mixture model. That's a fancy way of saying we're stacking a bunch of bell curves on top of each other, each one representing a different group in the data (like, clusters of cars). The model doesn't care which car belongs to which group; it just figures out the overall mix.

Some curves matter more than others, based on how many cars are in each group. In the end, you get this blended curve that helps the system spot and count all the vehicles, even when they're all piled together like sardines at rush hour.

And yeah, that's pretty much the magic behind the scenes.




Alright, here's the lowdown on Shortest Job First (SJF) — or, if you wanna get fancy, Shortest Job Next. Basically, it's this way of picking which process to run next: just grab the one that'll finish the fastest. No rocket science, right? The catch is, it's non-preemptive, so once something starts, it ain't stoppin' until it's done.

Here's the thing: SJF is kind of the superstar if you're looking for the lowest average wait time. Seriously, it's greedy in a good way. But—and there's always a but—if little quick jobs keep showing up, longer tasks might just sit there twiddling their thumbs forever (yeah, that's called starvation, and it sucks). Some folks fix that with "aging," which is just a fancy way of saying, "Hey, let's bump up the priority for stuff that's been waiting too long." But honestly, SJF is kind of a unicorn in real life. Why? Because the OS usually has no clue how long something's gonna take. It's not psychic, you know? Sometimes people try to guesstimate with tricks like looking at how long similar jobs took before, maybe weighting recent runs more—so, a little math magic. SJF works if you've got some special setup where you can actually tell how much time each job needs, but out in the wild? Not so much.

## III. IMPLEMENTATION DETAIL

The First objective of this project is to design and implement a Real-Time Adaptive Traffic Signal Management System using artificial intelligence techniques in simulated environment to improve traffic flow efficiency at intersections of roads. The system focuses to dynamically adjust traffic signal timings based on real-time vehicle density, vehicle types with their speed, and the presence of emergency vehicles, ensuring minimize wait times, congestion reduction, and quick emergency response. This project has a simple Graphical User Interface (GUI). The window is titled "Traffic Management System" and is designed to make it look clean and pleasant. The application opens in a centered window with a neat and professional look, Below the title Five buttons are presented vertically last button is use to exit the project along with this each four button represents a different traffic simulation scenario:

- Very Busy Roads
- Many Emergency Vehicles
- Two Busy Roads
- Empty Roads

## V. EXIT

Traffic simulation is a critical component of testing adaptive traffic management systems. Before deploying the real-world infrastructure, it is necessary to simulate traffic behaviour, vehicle movement, signal operations, and emergency vehicle handling in a controlled virtual environment. Simulation allows observing, analysing, and refining the system under various traffic conditions without real-world risks or costs.

**The Traffic simulation System are looks like the same as the image bellow:**

- **Creation/Initialization Process:**

There are various steps are invovled to completely buid this system to visualing the flow of traffic with the help of advance technologies like AI and ML to efficiently manage the traffic and utilize the road which lead to various factors which we are already discussed in benefits and planning section.

- **System Initialization:**

Upon execution, the simulation initializes four traffic signals representing a four-way intersection (North, East, South, West). Each signal is assigned default green, yellow, and red durations. The simulation begins with one green signal while the rest remain red.

**Vehicle Generation**

Vehicles are generated randomly at each approach of the intersection. The type of vehicle (car, bus,truck, or bike) and its direction (right, down, left, or up) are determined probabilistically. Each vehicle is assigned to a lane and begins its movement toward the intersection.

```
def generateVehicles():
    while True:
        temp2 = random.randint(0, 99)
        if temp2 < 3:
            vehicle_type = 4
        else:
            vehicle_type = random.randint(0, 3)

        lane_number = random.randint(1, 2)
        temp = random.randint(0, 99)
        direction_number = 0
        dist = [25, 50, 75, 100]
        if temp < dist[0]:
            direction_number = 0
        elif temp < dist[1]:
            direction_number = 1
        elif temp < dist[2]:
            direction_number = 2
        elif temp < dist[3]:
            direction_number = 3

        Vehicle(lane_number, vehicleTypes[vehicle_type], direction_number,
                directionNumbers[direction_number])
```

**Initializing Vehicle Movement and Positioning**

Each vehicle moves according to its defined speed, which varies by type. Movement is constrained by:

- The stop line of the signal,
- Presence of vehicles ahead (to maintain realistic spacing),
- Current signal state (green, yellow, or red).

Vehicles stop at the red light and only proceed when the signal turns green. Once they cross the stop line, they are marked as "crossed" and no longer affect green time calculations. Vehicle movement in the simulation is handled frame by frame to realistically model how vehicles behave at an intersection. Each vehicle is assigned a path and continuously updates its position based on its speed, direction, and the current state of the traffic signal. If the traffic light is green for a vehicle's lane, it moves forward; if the light is red, the vehicle stops near the stop line, maintaining a safe distance from the vehicle ahead to avoid collisions.

**Vehicle Movement**

The simulation also handles turning movements where necessary, allowing vehicles to smoothly transition left or right based on predefined rules. Furthermore, the system intelligently manages acceleration and deceleration to mimic real-world driving behaviour, ensuring that vehicle motion looks natural and responds appropriately to dynamic changes in traffic conditions, such as the appearance of an emergency vehicle or sudden signal changes.

The entire environment is visually represented: roads, vehicles, signals, and movements are displayed graphically, with signals changing colors and vehicles moving accordingly. Vehicles slow down when they approach a yellow or red light and speed up once they pass the intersection on a green light. Emergency vehicles are visually distinguishable and seen being given priority. The display updates smoothly and constantly to provide a real-time dynamic feel, allowing observers to clearly see how traffic is being managed effectively under different conditions.

**Step 2 - Spot the Cars in Your First Video Frame**

Alright, so foreground segmentation isn't magic—it'll pick up a ton of random junk along with the actual cars. You end up with these annoying specks and holes everywhere. Here's where morphological opening comes in clutch: it sweeps away most of that noise and patches up the gaps, so your cars look like, well, cars. Check out the "Noise Removed" image—way cleaner, right?

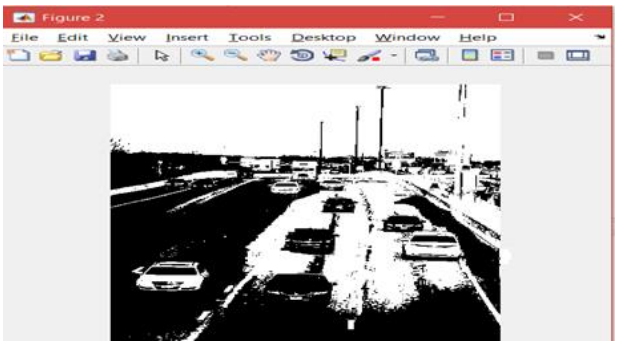
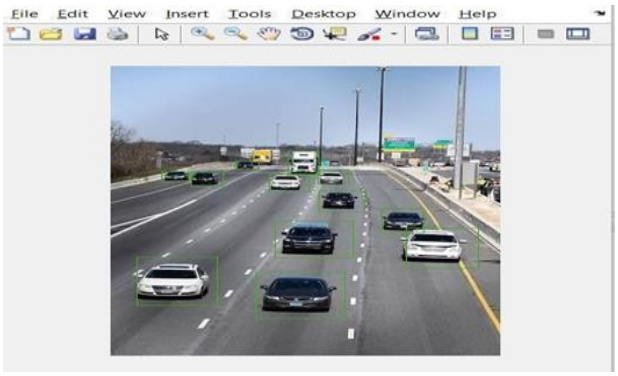



Fig -3: Clean Foreground

Alright, here's how it goes. We basically use this thing called Blob Analysis (sounds fancy, but it's just a way for the computer to spot blobs—aka, moving cars—in the video). Once the system sniffs out each moving car, it slaps a green box around it, like it's tagging them, "Yup, found you!"

And yeah, the total number of green boxes? That's literally how many cars the system picked up in that frame. Simple as that.




**Step 3 - Crunch Through the Rest of the Video & Shoot the Count to the Controller:**

Alright, time to let the algorithm loose on the rest of those frames. Basically, we're just chewing through the video, counting cars like some caffeinated highway cop, and then firing that number over to the controller. Not rocket science, but you gotta make sure the count's accurate or someone's gonna be yelling about "Why is the traffic light stuck on red again?!" You get the idea.

| Car count | Thresholds Value | Time allotted |
|---|---|---|
| 0 | 0 | Skip the signal |
| 5 | <10 and >1 | 20 sec |
| 20 | <30 and >10 | 30 sec |
| 40 | 30> | 60 sec |

Fig -4: Different Thresholds

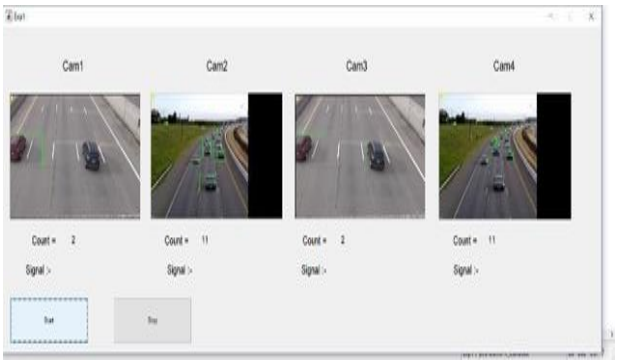



Fig -5: GUI of the project

So, here's what we did in MATLAB for the project—basically slapped together a GUI with four windows, each showing live video from a different camera. Every camera's eyeing a chunk of road, right? The code's busy spotting every car that rolls by, ticking up a vehicle count for each feed. After a bit (like, every few seconds), it sorts the roads by how jammed they are, biggest traffic gets bumped to the top. Then—bam!—the traffic signal for the road with the most cars turns green. Rinse and repeat, and yeah, traffic actually chills out. No magic, just some decent logic and a whole lotta MATLAB elbow grease.

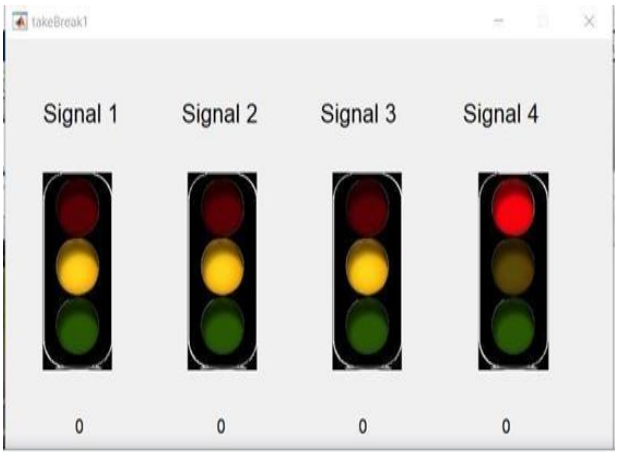
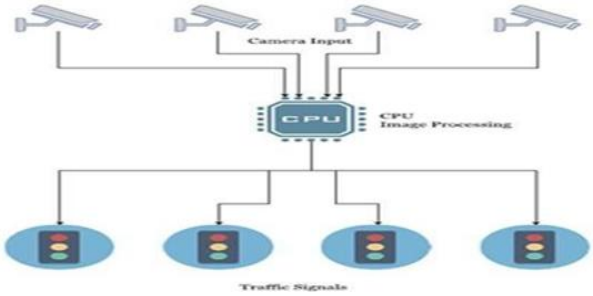



Fig -6: Signal allocated on different roads




## IV. SYSTEM ANTIQUATION

**Hardware:**

So, here's the lowdown—there's a PC at the heart of this whole thing. Old-school, sure, but it gets the job done for image processing and all that jazz. Cameras? Yeah, those are plugged in too, grabbing live video so the system can actually work. Nothing too space-age, but hey, it works.

**Software:**

MATLAB's running the show on the software side—handles all the signal stuff, crunches the images, the works. Basically, if it needs processing, MATLAB's probably doing it.

## V. CONCLUSION

In this paper the project AI based traffic management system developing by using simulation the adaptive system more cars can cross the intersection in a single cycle thanks to this system, which also enables real-time signal timing adjustments based on traffic conditions. This system makes efficient use of the road by reducing waiting times, Minimizing traffic jams, and efficiently using the road spaces by adjusting the traffic Lights according to the situation. It dynamically modifying the green light timing in response to the actual traffic load on each lane. It guarantees that traffic are flowing efficiently by optimizing varies points like busy roads are given more clearing time, empty roads are not wasted with lengthy green lights. Overall, this system makes road use more efficient and smoother by intelligently adjusting green signal timing based on

actual traffic load, clearing heavy lanes more quickly, and avoiding time wasted on empty roads.

**ACKNOWLEDGMENT**

It gives us a great sense of pleasure to present the report of the B. Tech Project under taken during B. Tech Final Year. We owe special debt of gratitude to Professor Dr. B.K Sharma (Director) and Mr. Nitin Kumar Sharma (Placement Head) of NITRA Technical Campus, Ghaziabad for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Professor Dr. B.K Sharma (Director) and Mr. Nitin Kumar Sharma (Placement Head) of NITRA Technical Campus, Ghaziabad for his full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

## REFERENCES

1. Rajeshwari Sundar, Santhoshs Hebbar, and Varaprasad Golla, "Implementing Intelligent Traffic Control System," Sensor Journal, IEEE (Volume:15, Issue:2)
2. Ms. Sarika B. Kale and Prof. Gajanan P. Dhok, "Design of Intelligent Traffic Light Controller Using Embedded System," Second International Conference on.Emerging Trends in Engineering and Technology, ICETET- 09.
3. Xia and Shao, "Modelling of traffic flow and air pollutions ion with application to Hong Kong Island", Journal of Environmental Modelling & Software. Vol. 20, 2005. pp 1175-1188.
4. G.Sathya, Fathima Shameema S, Jyothi MolSebastian, Jemsya K S"Automatic Rescue System for Ambulance and Authoritative Vehicles, Vol.2 -Issue 4 April.
5. Md. Rokebul Islam*, Nafis Ibn Shahid*, Dewan Tanzim ul Karim*, Abdullah Al Mamun**, Dr. Md. Khalilur Rh "An Efficient Algorithm for Detecting Traffic Congestion and a Framework for Smart Traffic Control System" Journal, IEEE Jan 20