Vision-based Efficient Traffic Control and Scheduling System for Smart City Intersections with Emphasis on Emergency Vehicles

Mahdi Seyfipoor[1], Sayyed Muhammad Jaffry[2], Siamak Mohamadi[3]

[1] *PhD Student at School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Iran*
mahdiseyfipoor@ut.ac.ir
[2] *Undergraduate Student in Computer Engineering at Faculty of Engineering, College of Farabi, University of Tehran, Iran*
smjaffry@ut.ac.ir
[3] *Associate Professor at School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Iran*
smohamadi@ut.ac.ir

Abstract

Smart cities move towards automating tasks to make people's life easier. Signalized intersections are one of the most common issues people face on a daily basis. Using sensors to improve the time allotment for different roads at the intersection can reduce waiting time for the vehicles, which can in turn reduce pollution and simply help people reach their destinations faster. Leveraging computer vision can aid in this process. We can use object detection to analyze the busyness of different roads, emergency vehicles and accidents, to allocate suitable times to each road. It can also help reduce collisions due to premature green lights, resulting in cars entering the intersection crashing with those leaving it.

Keywords: *Smart City, Computer Vision, Edge Computing, Traffic Lights.*

1 Introduction

Smart cities are cities that use various sensors to provide its main operating systems with real-time data. This may include water, sewerage, traffic, law enforcement, security and surveillance systems [1]. The data that are collected can be used by city officials to aid in decision-making, efficient resource allocation, troubleshooting, or automate certain tasks. Automating the city's traffic control system can have many benefits: improve traffic flow, reduce CO_2 emissions, and lower the travel-time of passengers. One of the most valuable parts of a smart traffic control system, in controlling the intersections,

is using smart traffic lights. During different times of the day, traffic is distributed differently. Therefore, using a static time allotment scheme, that does not change based on the different circumstances, is inefficient. A dynamic time allotment policy takes into account various parameters such as, the busyness of a road, emergency vehicles that are arriving, and the cars in the intersection. Many countries have adopted some form of smart traffic lights. The goal of such systems is to use data available at intersections and roads to act like a traffic police officer, and hopefully better, since the officer may not have as much data as a system does. The control that the system has over the road needs to be backed by an accurate knowledge base, and these data can be obtained using sensors, such as, cameras. The use of computer vision can help us design such a system. It can conduct the necessary operations we require for a smart traffic light. Using object detection, we can detect the busyness of a road, identify any emergency vehicle (EV) and the path it is taking, and also find out if there are any cars in the intersection.

2 Related Works

Azad and Ramazani [9] propose an algorithm based on Q-Learning combined with deep neural networks, which resulted in a 34% decrease in queue waiting time. Wiering [10] proposed reinforcement learning algorithms, which reduced average waiting times by 25% in experiments where the cars were identical and travelled at similar speeds. Ferreira [11] introduced a infrastructure-less, vehicle-to-vehicle-communication based approach, and was able to show a reduction of CO_2 emissions up to nearly 20%. Younes and Boukerche [12] propose an arterial traffic light (ATL) controller, intelligent traffic lights communicate with each other to generate an efficient traffic light algorithm for the entire network. They also proposed intelligent traffic light controlling (ITLC) algorithm which schedules each isolated traffic light efficiently, resulting in a 30% increase in traffic flow fluency compared to the online OAF [13]. Gradinescu et al. [14] have also used a similar approach, resulting in significant reductions in waiting time, fuel consumption and pollutant emissions. Huang et al. [15] use syncronized timed Petri nets (STPN) to design an urban traffic network control system. They use a modular technique so that the network can be extended easily.

3 Methods

Our proposed smart traffic light control system consists of 4 units: Physical unit, object detection unit, processing unit and scheduling unit. The physical unit controls the flow of data from the sensors towards the object detection unit. It is the lowest level in the system. The object detection unit is tasked with identifying different objects and their count in the data provided by the physical unit. There are many models used for object detection, most famously Single-Shot-Detectors (SSD) [2], Residual Networks (ResNet)

[3] and You-Only-Look-Once (YOLO) [4]. YOLO is well-known for its fast and efficient performance, and lightweight versions of it such as YOLO-LITE [5] are used in embedded and real-time systems. The processing unit will gather the information of the vehicles, pedestrians, and active emergency vehicles, and process the information for the scheduling unit, which will allot green light-times to each road in the intersection. The units can be implemented in two major ways: edge computing (embedded), and server processing. The former uses a processor embedded inside the device that is controlling the intersection to store, process and analyze without using a dedicated external server, and in the latter the data are sent to an external server which will send an output back to the device [6]. Each method has its own advantages and disadvantages. In edge computing, we do not rely on network or cellular connections to send and receive information from the server, rather the device will compute the output on-site, making it favorable for real-time applications [6]. Using edge computing is also better for security, is more cost-effective, and lower in power consumption compared to traditional computing [6].

There is a lot of information that is available at an intersection. The data and conditions that our scheduler requires are as follows: active emergency vehicles and the roads they are approaching from, accidents that have occurred at the intersection or on any roads connected to the intersection, the existence of cars in the intersection, the number of cars on the roads connected to the intersection, and the pedestrians in or at the intersection. The scheduler treats each of the roads and pedestrian crossings as a set of *tasks* that need to be scheduled. Each of the roads and crossings (tasks) contain conditions, which are basically the vehicles or pedestrians occupying it. It will prioritize the roads based on the conditions as mentioned in table 1. The nature of the scheduler is similar to that of a Round Robin scheduling policy [7] because we use a circular queue. The difference is that each road will receive a time-slice that is decided by the conditions mentioned above, i.e. busyness, EVs and accidents. The effect of accidents on traffic and the proper methods of dealing with them need to be studied further, therefore in this paper, we have focused on emergency vehicles. Ghazal et al. [8] propose using a handheld controller to trigger the emergency vehicle protocol. This controller will be used by the emergency vehicles approaching the intersection. We propose using object detection to identify the emergency vehicles, but handheld devices can also be used to assist decision making.

As we can see, the vehicles that are already inside the intersection hold the highest priority, and will not be preempted by any other vehicles. A similar statement is also true for pedestrians inside the intersection. If any task is scheduled to activate next, it will have to wait until the higher priority tasks have been completed. A flowchart for the general functionality of the system is provided in fig. 1.

In fig. 1, we see that if the system spots an EV (Emergency Vehicle), it will attempt to open the signal for the road containing the EV, and keep that road open until the EV crosses the intersection. We understand that this shift cannot always happen instantaneously, as there may still be cars inside the intersection. Instead, the system

Table 1: Priorities of tasks in the scheduler, a greater priority shows higher importance

Conditions	Priority
Vehicles/pedestrians inside the intersection	3
Emergency Vehicles	2
Vehicles/pedestrians at the intersection	1

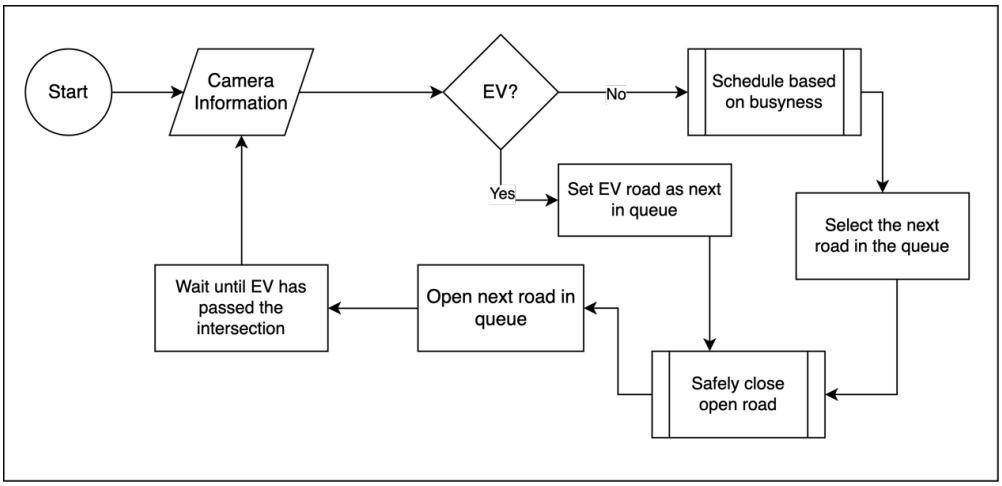

Figure 1: Flowchart of the traffic control system

will initiate a closing process for the currently open road, which is explained in fig. 2. It is important to note that the next road will open only once the safe closing process is completed, meaning that the intersection is clear of vehicles and pedestrians. In the scheduling process, the total period is divided among the roads, with busyness being the main factor. Newly arriving vehicles may or may not extend the time of a signal, but the signal will definitely stay open for a newly arriving EV.

Fig. 2 shows the safe road closing process. The light turns yellow for 3 seconds before turning red, after which it checks whether the intersection is clear of vehicles and pedestrians or not. The system will keep monitoring the intersection until it is empty.

We can define a syntax for possible scenarios: a road leading to an intersection I_i is denoted by L_i. The time remaining until the signal for L_i turns to green is $R_i(t)$. The time remaining until the signal for L_i turns yellow is $G_i(t)$. Busyness can be measured by a $busy\_factor$, where the sum of the $busy\_factor$ of all the roads equals 1.

There are 3 main approaches we can take in order to schedule times for each road. These approaches differ based on the total interval that is divided between the roads. One approach is to have a constant interval or period, denoted by P. After P seconds, the scheduler will reevaluate the intersection, and allot the times based on the new conditions. In this approach the sum of green light-times of all the roads are equal to P, such as shown in (1). Extending this approach can help us arrive at a superior method, which is a preemptive version of the one we just mentioned, meaning that the

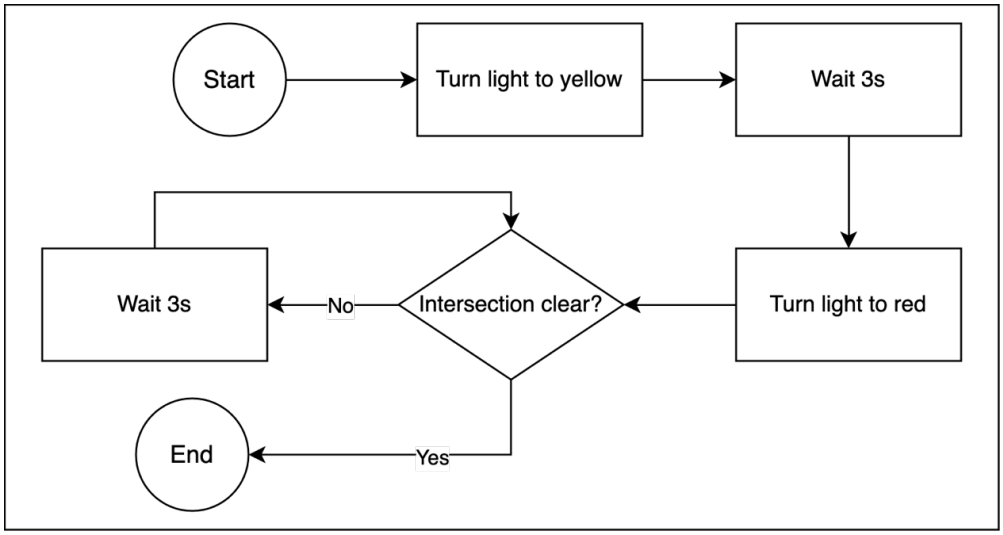

Figure 2: Safe road-closing process

scheduler will preempt the remaining allotted time to a task if it finishes prematurely. This can also be implied by using accurate data and car speeds in the simpler method.

$$\sum_{i=1}^{4} \max\{G_i(t)\} = \sum_{i=1}^{4} window\,(L_i) = P \tag{1}$$

$$Allocated\_Time = busy\_factor \times P \tag{2}$$

Another approach is to use memory for the scheduler. This way, the scheduler will remember the time allocation that was used during previous periods. If one road had stayed open for much longer than usual due to reasons such as an EV or a crash, the other roads will be compensated as they did not get a fair amount of time. The drawback to this approach is that even though the scheduler tries to be fairer than the first approach, this level of fairness may not be necessary most of the time, since the other roads might not be as crowded as the road with the longest time-slice in the previous period. Using memory can even cause a higher waiting time compared to not using memory, because the conditions can completely change during an interval.

A third approach is to have a dynamic period, in which the sum of the time-slices of the roads can vary depending on the conditions. For example, if all the roads leading to an intersection are more vacant than usual, then using a constant period will have a higher average waiting time for the vehicles at the intersection when compared to using a shorter period. Using a dynamic period can help by reducing the waiting time for the vehicles at an intersection. A question that naturally follows is how the period should be calculated at each interval, for which we propose using a set of predefined periods that can be used by the scheduler based on the current conditions. For example, an intersection can use the following periods based on the conditions {60s, 100s, 120s, 180s,

300s, etc.}. If the intersection is relatively empty, the scheduler can choose a smaller period so the utilization rate of the intersection stays high. From an OS perspective, this is similar to assigning the CPU to a new task when a task finishes before the time allotted to it ends, and in contrast to waiting for the allotted time to end, even if the task finishes sooner than expected. Using predefined periods simplifies the scheduling process, while providing a "good enough" scheduling scheme. Extending this idea, the scheduling process itself can also use a smaller quantum of time, and each road receives a multiple of the quantum. For Example, time-slices can be multiples of a 10 second quantum.

One point to keep in mind is that an accident or the arrival of an EV can occur at any time. If the system only updates the condition values after an interval, the effectiveness of the scheduling scheme will be reduced, and may even prove to be counter-productive to its objective. This leads us to constantly check for a change in high-priority conditions, so the system can only take action if a condition, such as an EV or cars inside an intersection, changes.

4 Simulation

We modeled the intersection as a task set, with each road leading to the intersection as a task. The vehicles are subtasks, with the EV subtasks having the ability to preempt other tasks in the intersection. While an EV subtask may not be the first subtask of a task, the task will keep running until the EV subtask is completed. Table 2 contains the information of the simulation setup. The simulation was done using Python. The preemptive versions of the algorithms were simulated.

The set of periods available for dynamic period selection was {60, 90,120,180,240}, and the choice was uniformly spread over the $busy\_factor$ axis.

The results of the simulation suggested that there was not a huge difference among the methods when it came to scenarios where the intersection was regarded as crowded. The waiting time and total number of cars passed was also similar. The main reason for this was that all 3 methods were preemptive in nature, meaning, that if there were no more cars on a road, the scheduler would stop the execution of that task. On the contrary, for simple traffic signals, the traffic light would keep on executing the same task until the allotted time for it was over, which resulted in a waste of valuable time.

5 Conclusion

We introduced 3 approaches to schedule the allocation of the intersection to the roads leading to it. The main points of difference in the methods were the use of memory and the selection of the scheduling period. Using memory will incur a cost, and is not needed anyways. Constantly gathering information causes all 3 methods to perform similarly, and therefore the different periods will not affect the system greatly.

Table 2: Simulation Setup Parameters

Parameter	Value
Period (not applicable to dynamic period)	240s
Time added per car	2s
Time it takes a car to cross the intersection	2s
Probability of new car arriving	10-40%
Probability of EV arriving	10%

References

[1] Moura, Filipe & de Abreu e Silva, Joao. (2021). Smart Cities: Definitions, Evolution of the Concept, and Examples of Initiatives.

[2] Wei Liu et al, "SSD: Single Shot MultiBox Detector", arXiv:1512.02325, 2015

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", arXiv:1512.03385, 2015.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", arXiv:1506.02640, 2015.

[5] Jonathan Pedoeem, Rachel Huang, "YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers", arXiv:1811.05588, 2018

[6] K. Cao, Y. Liu, G. Meng and Q. Sun, "An Overview on Edge Computing Research", in *IEEE Access*, vol. 8, pp. 85714-85728, 2020.

[7] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. 2012. Operating System Concepts (9th ed.). Wiley Publishing.

[8] B. Ghazal, K. ElKhatib, K. Chahine and M. Kherfan, "Smart traffic light control system", 2016 Third International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA), Beirut, Lebanon, 2016, pp. 140-145.

[9] Azad, Seyedeh & Ramazani, Abbas. (2023). Smart control of traffic lights based on traffic density in the multi-intersection network by using Q learning. Discover Artificial Intelligence.

[10] Wiering, M.; Vreeken, J.; Van Veenen, J.; Koopman, A. Simulation and optimization of traffic in a city. In Proceedings of the IEEE Intelligent Vehicles Symposium, Parma, Italy, 14–17 June 2004;

[11] M. Ferreira and P. M. d'Orey, "On the Impact of Virtual Traffic Lights on Carbon Emissions Mitigation", in *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 284-295, March 2012.

[12] M. Bani Younes and A. Boukerche, "Intelligent Traffic Light Controlling Algorithms Using Vehicular Networks", in *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 5887-5899, Aug. 2016.

[13] K. Pandit, D. Ghosal, H. M. Zhang and C. -N. Chuah, "Adaptive Traffic Signal Control With Vehicular Ad hoc Networks", in *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1459-1471, May 2013.

[14] V. Gradinescu, C. Gorgorin, R. Diaconescu, V. Cristea and L. Iftode, "Adaptive Traffic Lights Using Car-to-Car Communication", *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*, Dublin, Ireland, 2007.

[15] Y. -S. Huang, Y. -S. Weng and M. Zhou, "Modular Design of Urban Traffic-Light Control Systems Based on Synchronized Timed Petri Nets", in *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 530-539, April 2014.