Vrije Universiteit Brussel

Internet of smart-cameras for traffic lights optimization in smart cities

Tchuitcheu, Willy Carlos; Christophe Bobda; Md Jubaer Hossain Pantho

Published in:
Internet of Things

DOI:
10.1016/j.iot.2020.100207

Publication date:
2020

Document Version:
Accepted author manuscript

Link to publication

Citation for published version (APA):
Tchuitcheu, W. C., Christophe Bobda, & Md Jubaer Hossain Pantho (2020). Internet of smart-cameras for traffic lights optimization in smart cities. *Internet of Things*, *11*, [100207]. https://doi.org/10.1016/j.iot.2020.100207

Copyright
No part of this publication may be reproduced or transmitted in any form, without the prior written permission of the author(s) or other rights holders to whom publication rights have been transferred, unless permitted by a license attached to the publication (a Creative Commons license or other), or unless exceptions to copyright law apply.

Take down policy
If you believe that this document infringes your copyright or other rights, please contact openaccess@vub.be, with details of the nature of the infringement. We will investigate the claim and if justified, we will take the appropriate steps.

Download date: 27. Aug. 2022

Internet of Smart-Cameras for Traffic Lights Optimization in Smart Cities

Willy Carlos Tchuitcheu [a,*], Christophe Bobda[b], Md Jubaer Hossain Pantho[b]

[a] *Camertronix, Yaounde, Cameroon*
[b] *Department of Electrical and Computer Engineering,University of Florida, USA*

Abstract

Smart and decentralized control systems have recently been proposed to handle the growing traffic congestion in urban cities. Proposed smart traffic light solutions based on Wireless Sensor Network and Vehicular Ad-hoc NETwork are either unreliable and inflexible or complex and costly. Furthermore, the handling of special vehicles such as emergency is still not viable, especially during busy hours. Inspired by the emergence of distributed smart cameras, we present a novel fuzzy logic approach to traffic control at intersections . Our approach uses smart cameras at intersections along with image understanding for real-time traffic monitoring and assessment. Besides understanding the traffic flow, the cameras can detect and track special vehicles and help prioritize emergency cases. Traffic violations can be identified as well and traffic statistics collected. In this paper, we introduce a flexible, adaptive and phase-free distributed traffic-control algorithm that uses the information provided by distributed smart cameras to efficiently control traffic signals. Experimental results show that our collision-free approach outperforms the state-of-the-art of the average user's waiting time in the queue and improves the routing of emergency vehicles in a cross congestion area.

Keywords:
distributed smart-cameras, smart city, image processing, traffic signal, intelligent traffic management system, emergency vehicles, fuzzy logic

1. Introduction

The rapid growth in urbanization is leading to a tremendous increase in automobiles in cities [1]. Unfortunately, infrastructure development has not kept up with the growth in transportation. With lack and limited availability of public transportation, traffic congestion on public roads during rush hours has become a critical problem in many countries. This problem will be unmanageable if no effort is undertaken [2][3]. Congestion results from traffic demand that approaches or exceeds the capacity of the available infrastructure. There are essentially two categories of traffic congestion: 1) recurring traffic congestion that appears at the same place and the same time every day and 2) non-recurring traffic congestion caused by a random unplanned event or temporary disruptions that take away part of the roadway. The US Federal Highway Administration defines six sources of congestion[4] as shown in Table 1. Figure 1 shows the sources of congestion and their contribution to congestion in percent on y-axes.

Table 1: Congestion sources and terms.

Termed	**Source**
Bottlenecks	Road with inadequate physical capacity (roadway narrows, enclave).
Traffic incidents	Vehicles crashes and stalls.
Work area	Road repairs, building of new roads and maintenance activities.
Bad weather	Flood, snowfall and fog.
Rare Events	Strikes and marathons.
Poor signal timings	Empty lane with green light [5] and time allocated with respect to the volume of the traffic on the lane.

Recurring and non-recurring traffic congestion contribute to urban traffic congestion at almost the same rate. Heavy traffic congestion leads to waste of time, increase pollution, waste of fuel, increased cost of transportation and driving-related stress, inefficient supply chains [6], with an adverse effect on the economy [4][7][8]. Intelligent Traffic Management System can alleviate traffic congestion by 1) collecting traffic data in real-time at intersection, for instance through the use of Wireless Sensor Networks (WSNs) [9], RFIDs, ZigBee [10], Vehicular ad-hoc NETwork(VANETs) [11], Bluetooth devices and cameras and infrared signals, whereas WSNs have gained increasing attention in traffic detection and avoiding road congestion [12]; 2) using adaptive algorithm to control traffic with the goal of minimizing the average waiting time in queues of users; 3) incorporating a mechanism to allow emergency vehicles to easily cross congestion areas. As perspective work to alle-

*Corresponding author
Email addresses: twilly@aims.ac.rw (Willy Carlos Tchuitcheu), cbobda@ece.ufl.edu (Christophe Bobda), mpantho@ufl.edu (Md Jubaer Hossain Pantho)

viate traffic congestion, Paolo Santi [13] relies on VANET and proposes a promising concept LightTraffic which aims to reinvent the traffic light by challenging today's model of grinding to a halt every time. LightTraffic allows vehicles to safely cross an intersection in coordination with others approaching and could double the capacity of the current system, minimize travel delays, reduce emissions, and eliminate queues without lowering current car volumes.

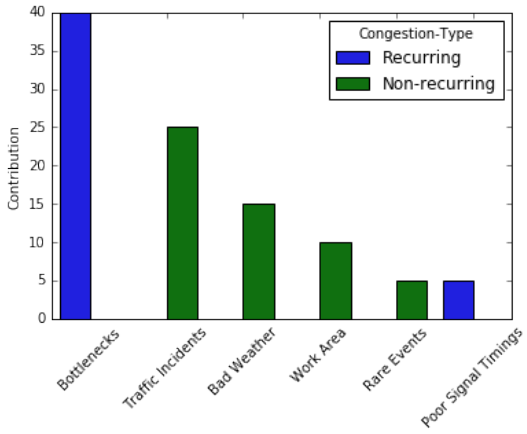


Figure 1: Traffic congestion contribution as a percentage of congestion source.

Despite various techniques and research for alleviating traffic congestion including "anywhere working" [1], "Markov chain traffic assignment" [14], and government's policies such as congestion pricing, driving restrictions, vehicle purchase restrictions, and public transit investment, the Intelligent Traffic Management System (ITMS) [15] continues to face significant challenges namely:

- Congestion : can ITMS react quickly to non-recurring congestion problem.
- Traffic incident notification (Traffic violations, traffic rules) : can ITMS send real-time information to police to act upon the situation.
- How to maintain coordination between intersections for a safety smart city.

Table 2 presents several technologies used for traffic control. Most of those systems rely on Wireless Sensor Network (WSN) for traffic control coupled with cameras for video surveillance [16]. These methods are tedious and involve a large amount of hardware.

In this paper, we propose a new and versatile method that infers with relevant critical scenarios at the road-intersection. Our contribution relies on fuzzy logic to propose: 1) a model of the state queue taking into account the priority of emergency vehicles and traffic load behavior; 2) a phase-free distributed algorithm for traffic signals control at the road-intersection. Our work addresses the system-level architecture and methods. The cameras are used to provide real-time and contextual data needed for system management. Figure 2 illustrates how our method uses distributed smart cameras along with advanced image understanding to supply the waiting queue in real-time with traffic data (vehicles count, types, density, etc). This information can then be used by a central authority to control the whole traffic infrastructure.

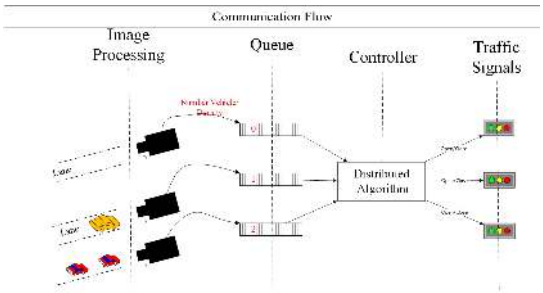


Figure 2: Communication flow.

The target of this initial paper is a smart city with multiple connected infrastructure, including traffic light and networking aspect as figure 3 illustrates.

Figure 3: Smart city configuration with smart infrastructure, cloud-based data collection and decision making.

Data collected from those infrastructures are sent to the could for analysis and decision making. In the specific case of traffic, data collected from all traffic light can be used for inter-intersection control, tracking of special vehicles, traffic offenders and address the non-recurring congestion area by changing traffic rules of the adjacent intersections. The current paper focuses on smart control at road-intersections. The global management of the infrastructure and decision making in the whole context of a smart city is not in the scope of this work and will be addressed in future research.

The rest of the paper is organized as follows. Section 2 discusses the related work. In section 3, we describe our modeling system. In sections 4, 5, and 6, we explain our implemented algorithm, and the simulation and results in section 7.

2. Related Works

In this section, we discuss related work on smart traffic control. We divide the work in two categories: data

Table 2: Technology along with traffic data collected [17].

Technology	Vehicle Count	Presence	Speed	Output Data	Classification	Multiple Lane, Multiple Detection Zone Data	Communication Bandwidth
Inductive loop	√	√	√ *	√	√ &		Low to modest
Magnetometer	√	√	√ *	√			Low
Magnetic induction coil	√	√ $	√ *	√			Low
Microwave radar	√	√ #	√	√ #	√ #	√ #	Moderate
Active infrared	√	√	√ @	√	√	√	Low to modest
Passive infrared	√	√	√ @	√			Low to modest
Ultrasonic	√	√		√			Low
Acoustic array	√	√	√	√		√ ˆ	Low to modest
Video image processor	√	√	√	√	√	√	Low to high

* Two sensors can be used to measure speed; & With specific electronics device that categorizes vehicles;
$ By using distinct sensor layouts and data processing software; # By using a microwave radar sensor and suitable signal processing unit;.
@ With multi detection region; ˆBy suitable beam forming models and data processing unit

collection and traffic control solutions.

2.1. Traffic Data Collection Technologies

In [17], a review of technologies for traffic data at the intersections is performed. For each technology, the authors consider the ability to count vehicles, detect vehicles, detect the speed of vehicles and the ability to distinguish different vehicle types. The authors also consider whether the technology could be used in a multi-lane scenario, and the bandwidth required to communicate the information to a central control. The summary of the survey is shown in table 2.

Anilloy Frank and, al,. [18] proposed a system based on IoT to control the traffic density control using image processing techniques. The system uses Raspberry-Pi for communication between the server and traffic lights. It first captures the image using USB cameras and sends it to the server where an algorithm is developed to determine the traffic density on each side of the road. Nevertheless, the authors did not address the multi-lane traffic density.

The presented technologies are not suitable to detect special vehicles such as emergency and police cars. Furthermore, to detect the entire intersection, multiple sensor nodes must be deployed to increase coverage. As a result, efficient coordination with the central system becomes very challenging. Our approach in this work is based on video, which is free of the previous-mentioned challenges.

2.2. Traffic Control Algorithm

There is a large amount of literature on the subject of traffic control algorithms. We provide an overview of the representative examples.

In [19], the authors propose WSN architecture and an algorithm for controlling green lights on a single intersection. Their solution is designed for an isolated intersection and reduces waiting times without introducing congestion.

The approach has been extended in [20] to an algorithm called TAPIOCA *(distribuTed and Adaptative IntersectiOns Control Algorithm)* that considers multiple adjacent intersections that communicate with adjacent intersections using WSN.

In [21], an improvement of TAPIOCA was proposed for the multi-intersection case by defining mechanisms to ease offloading between close intersections and to create green waves. In [22], a novel approach to traffic control at the intersection was proposed. Rather than solving the optimization problem of green light scheduling, vehicles compete for the privilege of passing by exchanging messages. In this case, the vehicles must have an extra device that allows them to communicate.

In [23], an adaptive algorithm was proposed from the back-pressure routing, which has been mainly applied to communication and power networks. In [24], an adaptive traffic control for both single and multiples intersections was proposed.

In [12] a survey has been made on adaptive algorithms for traffic control. However, these approaches haven't considered the priority for emergency vehicles, and in case of changing traffic rules or maintenance at the intersection, the algorithm fails to adapt.

In general, those algorithms relying on a set of phase(group of queue without conflict of collision in green-light) and the WSN are not resilient to certain critical scenarios such as changing the highway code at the intersection, maintenance work on pavement and less optimize due the green-light with empty queue observed within a phase. Furthermore, they do not provide a safety gateway to act under certain critical cases occurring at the intersection remotely. The phase-free algorithm we present in this work is resilient capable of withstanding several critical cases that the existing approach cannot handle.

3. Design and Modeling

In this section, we will first describe the component of traffic light system we intend to optimize. A description of our overall control architecture will follow.

3.1. Ecosystem

The intersections considered in this way are 4-way right-side driving intersections. Each way has the three-color traffic light located at the right top and a smart camera

installed in the face of the road. All possible movements are allowed. An intersection is where multiple roads cross. A road is a set of lanes. Figure 4 shows an intersection with four lanes marked N (North), S(South), W (West) and E (East) that intersect. Each path has three lanes in the incoming direction, which are turn-left(L), go-forward (F) and turn-right(R). A passing vehicle can have a path P of $\{E, S, N, W\}$ and a direction D of $\{L, F, R\}$. Thus, a lane that has a vehicle can be determined by a pair of $\{P, D\}$. There are twelve lanes with labels pair (P, D) : $\{WR, WF, WL, ER, EF, EL, NR, NF, NL, SR, SF, SL\}$. Modeling the intersection to comply with road regulations is provided in the following table.

Table 3: Matrix of conflict.

	WR	WF	WL	ER	EF	EL	NR	NF	NL	SR	SF	SL
WR						⊗		⊗				
WF						⊗		⊗	⊗	⊗	⊗	⊗
WL				⊗	⊗			⊗	⊗		⊗	⊗
ER			⊗								⊗	
EF			⊗				⊗	⊗	⊗		⊗	⊗
EL	⊗	⊗						⊗	⊗		⊗	⊗
NR					⊗							⊗
NF	⊗	⊗	⊗		⊗	⊗						⊗
NL		⊗	⊗		⊗	⊗				⊗	⊗	
SR		⊗							⊗			
SF		⊗	⊗	⊗	⊗	⊗			⊗			
SL		⊗	⊗		⊗	⊗	⊗	⊗				

⊗ *Not Allowed* — *Illegal Case*

Table 3 shows conflict direction matrix [24] . An empty box $\{blank\}$ means that both lanes can proceed without a possibility of collision. A crossed circle $\{\otimes\}$ means that a collision may occur if both lanes are allowed to proceed. Because of this possibility of collision, lanes that have a crossed circle will not be allowed to proceed simultaneously.

3.2. Proposed Architecture

Our proposed architecture system is shown on Figure 6. The proposed framework comprises two components: the intersection unit and the cloud center. Within the intersection, there are three sub-components namely the controller, calibrated smart cameras, and the traditional traffic signal control unit. We designed each smart camera (see figure 5) by connecting a MIPI CSI-2 camera to a Zynq UltraScale+ MPSoC board (ZU3EG) [25]. The MPSoC board hosts a 64-bit Arm Cortex-A53 processor infused within the programmable logic. We designed the video pipeline on the programmable logic to receive image frames on the processor. Within the programmable logic, image data is transmitted through an AXI stream link. The embedded processor performs video traffic analyzer algorithm on image frame for extracting traffic-related knowledge (number and type of vehicles) within its calibrated space (waiting queues) in real-time. This information is sent to the controller unit through the communication medium (i.e. Wi-Fi). The edge module within the controller unit which is implemented on a similar 64-bit arm processor receives the information from different calibrated smart cameras and relies on traffic rules(matrix of conflict see table 3) for optimizing the traffic light through the traffic signal control. The layered architecture of our controller unit indicating the data flow from the bottom external modules to the application layer is shown in figure 6. It shows how the intersection can be remotely controlled by changing traffic rules from the cloud.

4. Image Processing Module

In this section, we discuss the image processing used for traffic analytic. This module is embedded within the smart cameras at intersections and perform processing in-situ. The advantage of this approach is reducing data transport. This section describes algorithms within the camera needed to extract knowledge such as number, types, and direction of cars in each lane as well as the density of traffic.

4.1. Detecting Vehicles

The goal is to identify the lanes separated by the white lines on the road, then detect and count the number of vehicles in each lane. Upon installing cameras at the intersection, calibration is first performed to define the regions of the pictures corresponding to the different lanes. While this detection could be done automatically, without calibration, the computation overhead is too high for run-time in-situ computation.

Camera calibration is spatial and consists of determining what parts of the scene are viewed by each camera at the intersection [26].
The algorithm works on the set of lines $Q = \{l_0, l_1, ..., l_{2p-1}\}$ used to determine the p Waiting Queues (WQ). Each line is represented by two random points (x_1, y_1) and (x_2, y_2) on that line: (l_i) : $a_i x + b_i y + c_i = 0$ where $a_i = y_2 - y_1$, $b_i = x_1 - x_2$, and $c_i = -a_i x_1 - b_i y_1$. Each lane $L_p = (l_{2i}, L_{2i+1})$ is bounded by two lines left l_{2i} and right L_{2i+1}. Furthermore, for each lane L_p we assign a unique color C_p. Algorithm 1 provides details on the vehicle counting in the lanes for the purpose of building the waiting queue.

Because the focus of the work is on the infrastructure for efficient control of the traffic, we will not dive into the details of the machine learning algorithms for car detection. This algorithm is part of the available machine learning package and can be integrated into any framework.

4.2. Density Measurement

In this section, we propose an approach which extends Anilloy Frank and, al,. [18] achievement by measuring in real-time the traffic density of multi-lane(queue) using background subtraction operation.

Density of a queue: We define the density d_a of a queue $\{a\}$ as the proportions of space occupied by the vehicles in $\{a\}$. Background Subtraction(BS) technique [28] is used to compute the foreground mask and measure the traffic density from the mask as shown in Figure 7. In the first step, the background model is computed when the

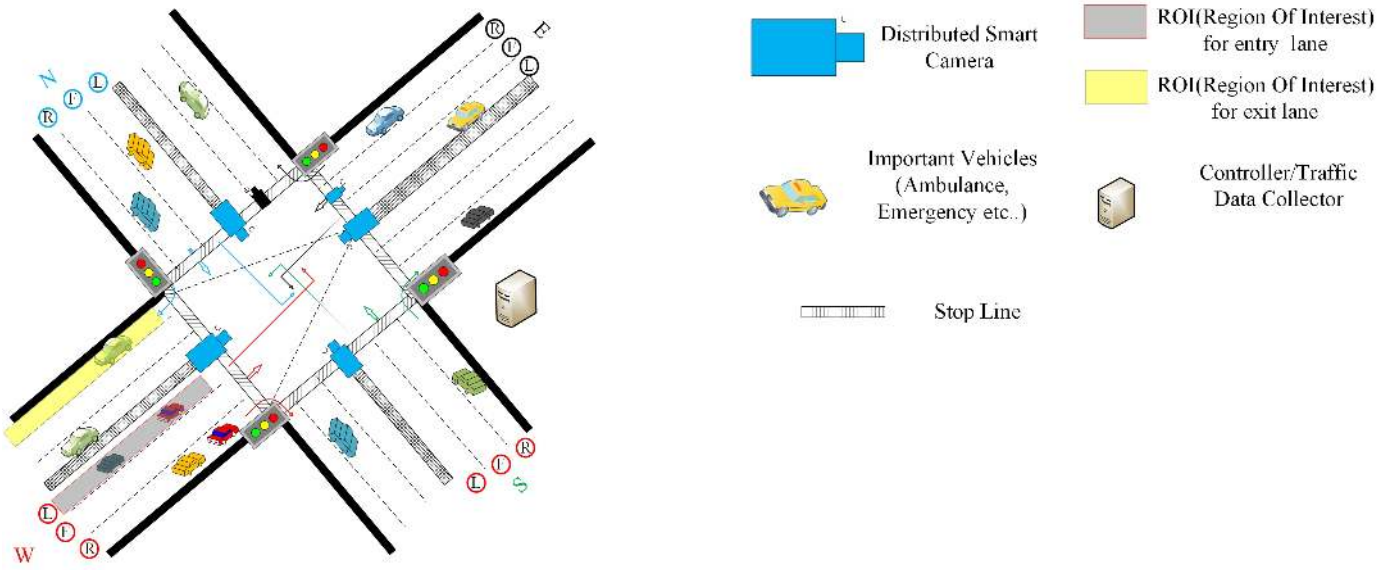


Figure 4: Distributed Smart-Camera based ITMS.

Figure 5: Smart camera setup. The ultra96 MPSoC with the camera connected to the extension board.

Algorithm 1: Algorithm Count Number Vehicle on each Waiting Queue

```
// The input of this algorithm in a frame or a picture
Data: Frame F or image
// The output is the number of vehicle on each Waiting
   queue
Result: Number of vehicle in each Lane
// For each frame F, use Mask-RCNN [27] to detect objects
   and return the set of rectangles where vehicles have
   been detected
ObjectDetect ⟵ Net.setInput(F);
// Initialize the Number of vehicles to zero
NumVehicle[0, ..., p − 1] ⟵ 0;
// Classify Objects with respect to lanes
for rect(x0, y0, width, height) ∈ ObjectDetect do
    // find the center of the rectangle
    A(xc, yc) ⟵ (x0 + width/2, y0 − height/2);
    // find the nearest line in Q to p by computing the
       distance
    j ⟵ arg min d(A, li), where d(A, li) =
      {|ai xc + bi yc + ci| / sqrt(a_i^2 + b_i^2)}; li ∈ Q;
    // localize the Lane of the vehicle Lk
    k ⟵ quotion(j/2) ;
    assign the color Ck to the vehicle;
    // Increase the number of vehicle of the Lane Lk
    NumVehicle[k] ⟵ NumVehicle[k] + 1 ;
end
```

road is free. In the second step, we are comparing the current frame to the background model in order to detect the objects(vehicles, truck, etc.) on the scene. Since the foreground mask is a binary image $\{0 = \text{Black}, 1 = \text{White}\}$, we can compute the density by counting the proportion of white over that area of ROI.

$$d = \frac{countNonZero()}{Area_{ROI}} \tag{1}$$

4.3. Implementation

We implemented our algorithm based on the Region based Convolution Neural Network [27] to output an object label, bounding box, and the mask. We have used open-source OpenCV [29] version 4.0.0 in C++ language. Figure 8 presents the flowchart of our implantation for density measurement and counting vehicles. We calibrated the camera in space in order to set the set of foreground mask and bounded of a lane for a future feature classification [26]. The results of implementation are shown on Figure 9a and 9b as snapshot from the video processing.

Having completed the first components of our control infrastructure, namely extracting knowledge from images to supply waiting queues in real-time, we first perform some analysis of performance before devoting the rest of the paper to the remaining modules of our architecture.

4.4. Performance Analysis

We measured the computation time of our model as illustrated in Table 4. We tested our model as a single-core implementation on a 64-bit x86 processor with a clock frequency of 3.60GHz. The total computation time does not include the calibration time since this will be only performed once only at the beginning of the computation. For calibration and testing purposes, We used a 4-minutes

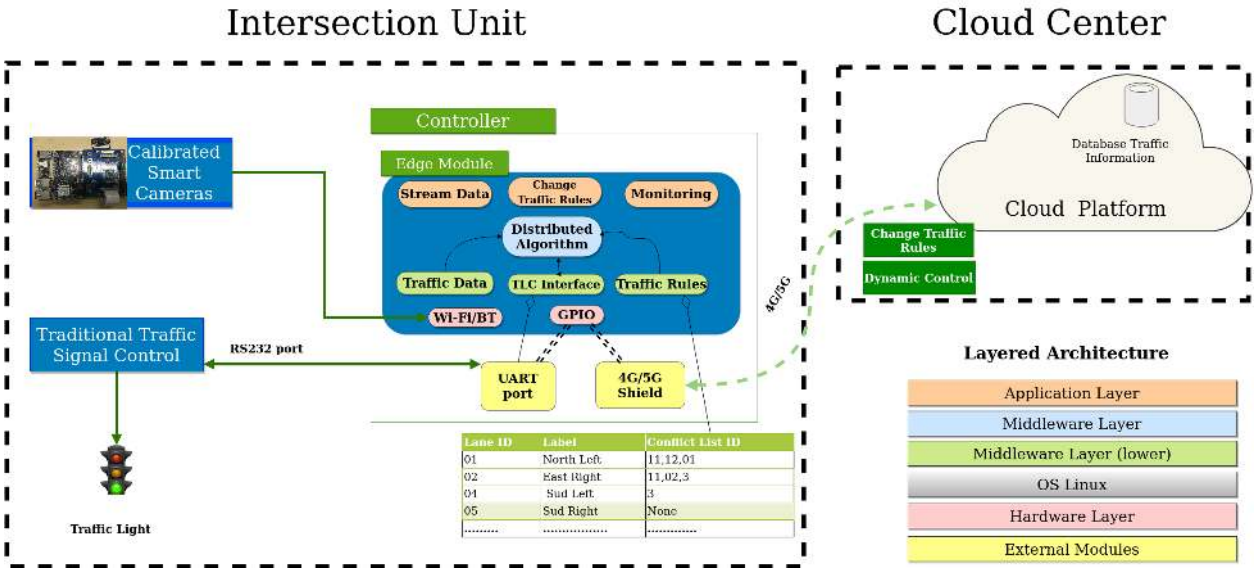


Figure 6: Proposed architecture system.

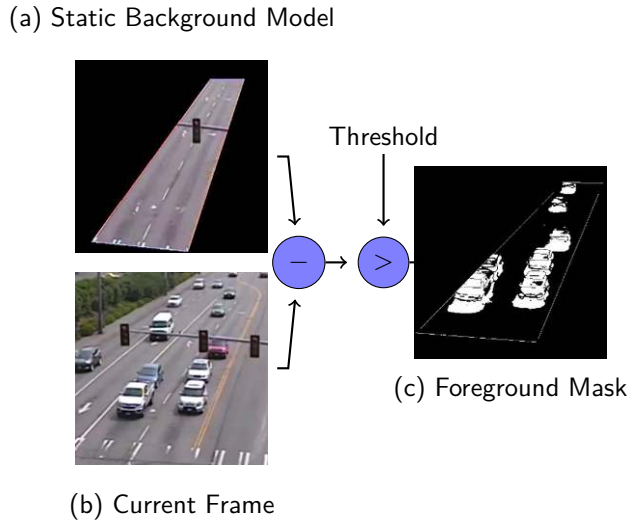


Figure 7: Density modeling. Figure (a) shows the static background model generated during the calibration step. (b) shows a test frame at any time t. (c) illustrates the foreground mask extracted from the image.

video clip of a 4 -way transaction as input. The results suggest that the model can achieve close to 4 frames per second. This can be improved by trading the detection model with a lighter one. We were able to achieve 13 frames per second by using the YOLO-V3-tiny model to detect vehicles. However, this reduces the accuracy of the model as well.

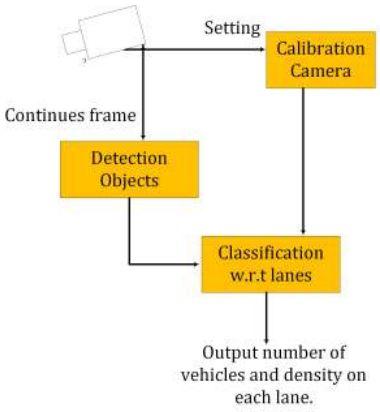


Figure 8: Design implementation.

Table 4: Performance Analysis

-	Computation time (ms)
Detection time	224ms
Calculating traffic density	40ms
Total Computation time	262ms
Frames per second	3.82fps

5. Waiting Queue

Our algorithm operates on waiting queues, which are updated using knowledge gained from the image processing module. This section provides details of the various waiting queues (WQ) used in our model.

A waiting queue represents vehicles in a given lane. We consider two types of waiting queues in the target system: Entry queues and exit queues:

5.1. Exit Queue

An Exit Queue noted Q_O or output queue at the intersection can be in one of two states :

(a) Frame captured before applying the algorithm.

(b) Output frame after applying the algorithm.

Figure 9: Result produced from recorded traffic data.

- **Open** Meaning this queue can accommodate vehicles.
- **Closed** Meaning this queue can not accommodate new vehicles.

For a queue in open state, we find the free space by using the density over the lane space.. This information is important for finding the maximum number of vehicles that the queue can accommodate.

5.2. Entry Queue

An Entry queue Q_I or Input queue at the intersection has three possible states :

- $\{A\}$: *Active* means that the signal light is **green** (G) for this queue.
- $\{WA\}$: *Waiting Active* meaning that the signal light should be **green** but for some reason(Q_O is closed or important vehicle has been detected) it has been blocked and the light remains red.
- $\{IA\}$: *Inactive* meaning that the signal light is **red** (R) for this queue.

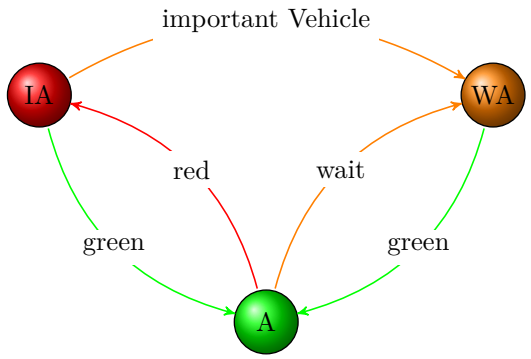
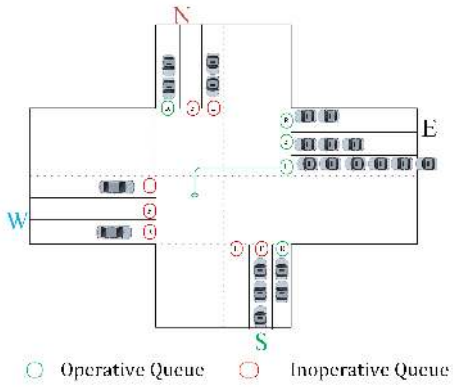
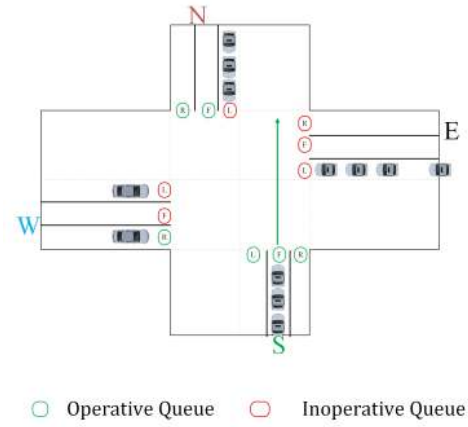


Figure 10: Simple state diagram of queue.

Figure 10 shows the states a queue can be in at any time. The controller algorithm chooses the queue with the highest number of vehicles to move from state $\{IA\}$ to $\{A\}$ and the state $\{WA\}$ precedes $\{IA\}$. However, the logic to select the waiting queue with a large number of vehicles has some problems, namely long waiting time in queue with low traffic. Figure 11a shows that at time t, $\{EL\}$ has the highest number of cars in it's queue so the algorithm turns this light green, at the same time $\{EF, ER, NR, SR\}$ can be open without conflict. In Figure 11b a time $t+x$, $\{SF\}$ is the queue with most of the cars, so this turns green. Observe that $\{EL\}$ is filling up with vehicles and will be opened next time, while queues such as $\{NL\}$ and $\{WL\}$ with low traffic are still waiting and may wait forever.



(a) Our approach consists of setting the green light to queue with highest number of car. At time t: $\{NL, WR, WL\}$ are waiting.



(b) Apply same principle as shown on figure 11a. At time $t + x$: $\{NL, WR\}$ are still waiting and next queue to be opened is $\{EL\}$. So, $\{NL, WR\}$ might wait forever.

Figure 11: Dead wait of low traffic queue along heavy traffic queue for simple state diagram queue modeling.

We have introduced the notion of internal state $IA \in \{0, 1, 2, 3, 4\}$ (the light is still red for those states) to take into account the fact that a queue with low traffic can at certain time have the highest priority. As describe in Figure 12 where

- E is a transition caused by the presence of an emergency vehicle.

- $Y_i(t)$ is elapsed time for a queue to change his internal state during the cycle i.
- $X_i(t)$ is the time allocated to empty the queue during a cycle i.

A cycle is when a queue made a complete turn through state $\{A\}$. An example of a possible cycle can be $\{0 \to 1 \to 2 \to A \to 0\}$, or$\{0 \to 1 \to 2 \to 3 \to WA \to A \to 0\}$. Hence $Y_i(t)$ is computing each cycle and is constant for a time slot t. The figure 12 shows the difference transitions and state of a waiting queue Q_I.

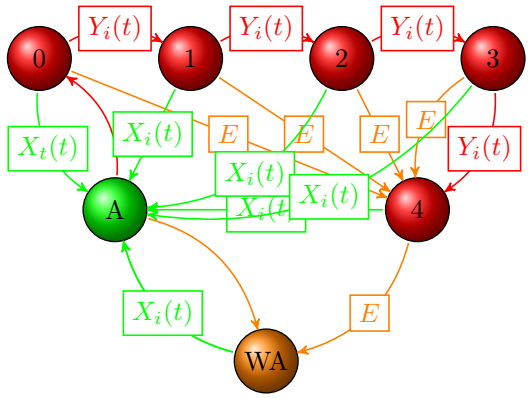


Figure 12: Extend state diagram of a queue.

The set $\{0, 1, 2, 3, 4\}$ are the states that a waiting queue Q_I goes trough cyclically whenever transition conditions are fulfilled. Only the controller or the master can set the status of the queue as $\{A\}$ and afterwards to zero to mark the end of a cycle.

Let's consider this cycle as the first running time. $\{0 \to 1 \to 4 \to A \to 0\}$. The queue starts in state $\{0\}$, we assume that, the queue is not empty (id the queue is empty, the state remains in zero). After $Y_1(t)$ time has elapsed (index 1 means cycle number 1) the queue moves to state $\{1\}$. The presence of an emergency vehicle upgrades it to state $\{4\}$, and after some time it becomes the only queue in state $\{4\}$, and it moves to state $\{A\}$. After the time taken to empty the queue noted $X_1(t)$, it returns to state $\{0\}$ and $Y_2(t)$ is computing to mark the end of the cycle.

Let $I_i^a(t)$ be the internal state of queue $\{a\}$ during the cycle i, the equation is given by:

$$I_i^a(t) = \begin{cases} I_i^a(t-1), & \text{if } I_i^a(t-1) < 0 \text{ or } I_i^a(t-1) > I_{max} \\ \min\{I_i^a(t-1) + 1_{[Y_i^a \text{ elapsed } \&\ d_I^a(t) \neq 0]}, I_{max}\}, & \text{Otherwise} \end{cases} \quad (2)$$

Where $I \in \{0, 1, \cdots, I_{max}\}$, in particular $I_{max} = 4$, $1_{[X]}$ is the indicator function that takes the value 1 if X is true and 0 otherwise, and $d_I^a(t)$ the current density of the entry queue $\{a\}$. There are 12 queues in our target system: $\{WR, WF, WL, ER, EF, EL, NR, NF, NL, SR, SF, SL\}$, each of which has an internal state. This internal state is relayed back to the main controller, where the algorithm sets the queue with the highest state to have a green light. If this is not possible, it selects the next highest state queue. After the queue has been emptied, the controller then reassesses which queue should be given priority.

5.3. Calculating $X_i(t)$: time needed to empty a queue

$X_i(t)$ is the time needed to empty the Q_I at time slot t during a cycle i. This time depends on the exit and the entry queue.

There are situations where the exit queue is not ready to accommodate vehicles but the light is green for the entry queue. On the figure 13, The queue $\{EL\}$ has the highest number of vehicles and has been selected, then we find the queue in green can operate without conflict, especially the arrival queue $\{EF\}$ but the departure queue is not ready to accommodate vehicles.

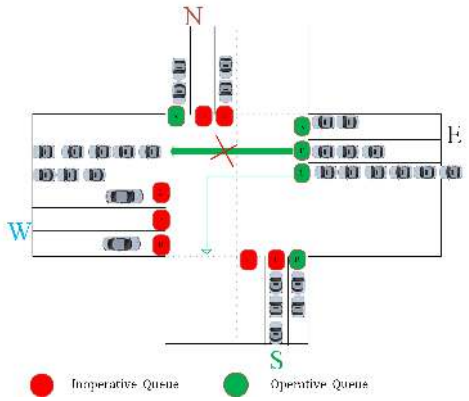


Figure 13: Case of a queue EF with green light but with exit queue that cannot accommodate new vehicle.

Hence, we take into account all these parameters and find out that the formula for computing the time $X_i^a(t)$ needed to empty a queue $\{a\}$ during a cycle $i \in \mathbb{N}$ at time slot t is:

$$X_i^a(t) = \frac{\min\{(1 - d_O^a(t))L_O^a, d_I^a(t).L_I^a\}}{V}. \quad (3)$$

Where d_I^a and L_I^a are the current density of the vehicle on input traffic queue $\{a\}$ and the length of the queue $\{a\}$, d_O^a and L_O^a are the current density on the output traffic queue(exit queue) and the length of the queue, and V is the velocity to cross the crossroad.

- $(1 - d_O^a(t))L_O^a$ is the proportion of space available to accommodate vehicles on exit (output) queue.
- $d_I^a(t).L_I^a$ is the proportion of space occupied by vehicles on the entry (Input) queue.

The function minimum guaranty the time needed to empty a queue is enough in order to avoid vehicles to remain in middle of the intersection after that time elapsed.

Note that, $X_i^a(t) = 0$ implies either $d_I^a = 0$ meaning there is not vehicles on the input queue, or $d_O^a = 1$ meaning the Output queue is full and can not accommodate vehicles.

5.4. Calculating $Y_i(t)$

Let Y_i^a be the time needed for a queue $\{a\}$ to change internal state. Y_i^a can be written as a function of the time needed to empty the queue:

$$Y_i^a = f(X_{i-1}^a). \tag{4}$$

Our goal is to find an appropriate function f that minimizes the time needed to empty the queue. If this time is high, this queue will be assume to have heavy traffic, and its internal state should change quickly. That is, if X_{i-1}^a is large, then Y_i^a should be small, and vice versa. Thus

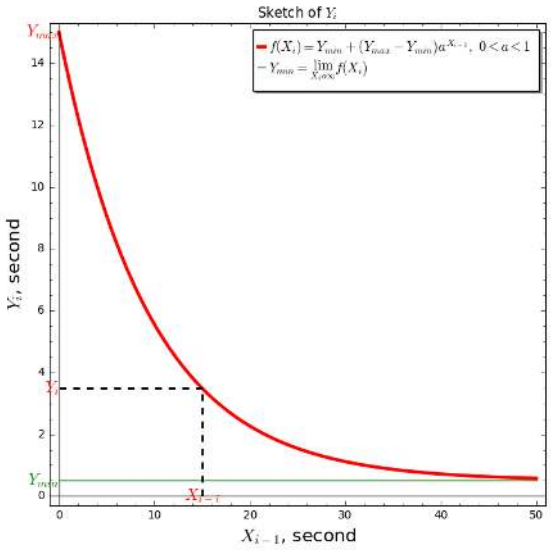


Figure 14: Plot of a possible choice for $Y_i^a = f(X_{i-1}^a)$

equation 4 becomes :

$$Y_i^a = Y_{min} + (Y_{max} - Y_{min})a^{X_{i-1}} \quad \text{where} \quad 0 < a < 1 \tag{5}$$

The figure 14 was obtained with $a = \frac{4.5}{5}, Y_{min} = 0.5$, and $Y_{max} = 15$. Here we will make an assumption that the model 14 is appropriate. Our goal is to vary the parameters $\{a, Y_{min}, Y_{max}\}$ in order to minimize the average waiting time.

Note that if a queue has only one car, it's internal state will still change, but the time taken to change will be longer than that of a queue with many cars. This way both the number of cars and the time spent are taken into account when assigning an internal state.

The complete algorithm is described as flowchart on the figure 15. Note that, each queue is supposed to run independently this algorithm.

Table 5: Sub functions used on flowchart Figure 15.

Sub-process	
Init()	$Y_i = f(X_{i-1})$;Status=0; Event=$False$; T=Time().
ChangeLevel()	Check if Y_i time elapsed and $d \neq 0$ and status $\neq \{A, WA\}$.
IncreaseStatus()	Status = $I_i(t+1)$, get it from equation 2
VehicleImportant()	Event= (emergency vehicle detected) ? True : False
SetPriority()	Set Status = (Status <4) ? 4 : WA.
ResetEvent()	Event=False

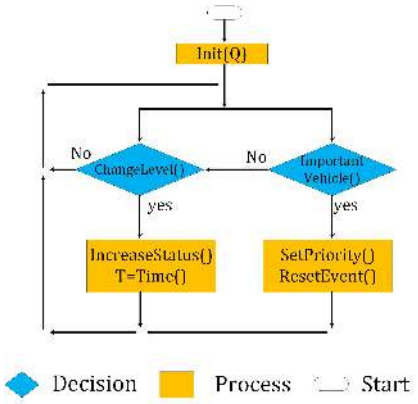


Figure 15: Flowchart queue insight.

Table 6: Variables used on flowchart figure 15.

Variables	
T	Record the time when increase Status occurs.
Status	Current Internal state of a queue, $\in \{0, \ldots, T_{max}, WA, A\}$
Y_i	Time elapsed for changing the internal state during current cycle.
d	Current density of the queue.
X	Time needed to empty queue during the previous cycle

6. Distributed Control Algorithm

In this section we describe the distributed algorithm for traffic light, based on the conflict matrix 3.

6.1. Modeling

We have described the different states of a queue in section 12, taking into account the density of traffic and the time elapsed. The road code was modeled through the conflict table 3 where entries in the diagonal were considered the illegal case. In the rest of this article, this table will be seen as a square symmetric matrix where indexes are queuing labels and the diagonal entries will hold the state of the corresponding queue. This matrix named here M contains the configuration of the intersection and the current state of the system.

Thus the square matrix M corresponds to the states of the queue and being constructed based on the following formula:

$$M_{[a,b]}(t) = \begin{cases} I_i^a(t), & \text{if } a = b \\ \otimes, & \text{if queue } \{a\} \text{is in conflict with queue. } \{b\} \\ 0, & \text{Otherwise.} \end{cases} \tag{6}$$

where $I_i^a(t)$ is the current state of queue $\{a\}$ from 2. Note that, from the equation (6) are follow :

$$M_{[a,b]} = M_{[b,a]} \text{ and } M_{[a,a]} = I_i^a$$

To solve this, we introduce a numerical values for the active state $A : -1$. and similarly for the waiting active state, we assign it a value of $I_{max} + 1$ which is 5 in particular.

$$\{A\} \leftarrow -1 \text{ and } \{WA\} \leftarrow 5$$

$\{A\}$ and $\{WA\}$ are respectively the status Active and Waiting Active from our state diagram of Figure 10. The

matrix M of equation 7 represents the state of the system at a specific time where the diagonal is the only dynamic part. The queues $\{ER, NR\}$ in red are operative, thus their internal state is set as $\{A\}$ meaning $M_{[ER,ER]} = -1$, and the next queue to open is $\{SF\}$ because of its highest internal state obtaining with following formula $\arg\max\{diagonal(M)\} = \arg\{M_{[i,i]} = 3\} = \{SF\}$.

$$M = \begin{array}{c} \begin{matrix} WR & \cdots & ER & \cdots & NR & \cdots & SF \end{matrix} \\ \begin{pmatrix} 2 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & -1 & \cdots & 0 & \cdots & \otimes \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & -1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \otimes & \cdots & 0 & \cdots & 3 \end{pmatrix} \end{array} \begin{matrix} WR \\ \vdots \\ ER \\ \vdots \\ NR \\ \vdots \\ SF \end{matrix} \tag{7}$$

This model with the matrix gives us a flexibility in case of modification of the road code at the intersection, it will be enough for us to modify the static part of the matrix by performing an operation to exchange between $\{0\}$ and $\{\otimes\}$. These operates can be extend in remotely control the intersection for a smart city.

6.2. Algorithm for updating the matrix M

The matrix M is made up of a diagonal and non-diagonal entries which hold respectively the current internal state of queues and the configuration of conflict between queues. The diagonal entries are dynamic as queues state change over time(see figure 15). Algorithm 2 describes how the matrix M is updated during run-time by computing the internal state of each queue (see equation 2) and assigning it to its corresponding cell of the matrix M.

Algorithm 2: Update the matrix M

```
Data: set of the queue
Result: Matrix M
for q ∈ setOfQueue do
    // Compute the internal state of the queue from the
       flowchart 15
    Compute I^q ;
    // Update the diagonal of the matrix M
    M[q, q] ← I^q;
end
```

6.3. Notations and Sub-functions

Now that we have digitized any intersection via our Matrix M, before writing the distributed algorithm, we are going to show variables and sub-functions used in the algorithm.

Table 7: Notations used in algorithm 3.

Variables	Meaning
Q	Set of all queue i.e $Q = \{WR, WF, WL, ER, EF, EL, NR, NF, NL, SR, SF, SL\}$
M	Square Matrix of size $\|\|Q\|\| \times \|\|Q\|\|$
ListOpen	Set of queue to open
RunList	Set of queue in process
S	Set of forbidden queue
U_a	set a queue without conflict with queue $\{a\}$

For the **Sub functions**, we have two. $Sort(L)$: which Sorts in ascending order the items in the list L, and $Open(a)$.

$$Open(a) = \begin{cases} \text{True}, & if X^a(t) \neq 0 \text{ we obtain } X^a \text{ from } equation3 \\ & X^a(t) \text{: time needed to empty the queue } \{a\} \\ \text{False}, & \text{Otherwise.} \end{cases}$$

Basically, the sub-function $open(a)$ is a Boolean function that returns true if the time needed to empty the queue is different to zero and false otherwise. Consider a scenario where an entry queue is full and the exit queue is also full. Then at a certain time, we realize that the queue has the highest priority. If we set a queue to green, the vehicles will stand in the middle of the intersection and that will cause traffic congestion. In order to avoid that, the time X_i^a needed to empty a queue is taking into account this scenario.

The distributed algorithm 3 is based on the matrix M essentially the diagonal of the matrix. It operates according to the principle that the queue having a high priority level (internal state) will be selected, and this algorithm is supposed to be executed by an irregular interval of time.

7. Simulation

This section evaluates our algorithm using a combination of image processing and the SUMO (Simulation of Urban MObility,[30]) framework. SUMO is an open-source, discrete-time, continuous space, microscopic simulator entirely coded in C++ to model traffic flow as well as supporting tools, mainly for network import and demand modeling. In short, it allows placing sensors and retrieving real-time traffic data using tools TraCI (**Tra**ffic **C**ontrol **I**nterface) and to act on the behavior of the Traffic Light Control on-line. Our distributed algorithm is based on an intersection modeled from the city Amiens in France at rush hour, between 8 AM and 9 AM, which according to Sebastien Faye and al. [20] statistically have in SUMO new vehicles arrival-rate of $\lambda = 0.8$ per second on the intersection. Relying on this statistical assumption, our simulation was conducted on an intersection with 4 directions and 12 possible movements: from each direction, a vehicle could go straight, turn left or turn right. In addition to that, we injected emergency vehicles with an arrival rate of $\lambda = 0.025$ per second in order to evaluate the impact of such vehicles on our distributed algorithm. Each simulation ran for 3600 program steps representing the 3600s. The results presented below are the average waiting time computed on the 3000 first vehicles that left the intersection. This allows us to evaluate our algorithm with a realistic traffic. Note that, the TraCI tool allows us to gather traffic data but in the real deployment of our proposed solution, those traffic data are provided by the distributed smart-cameras installed at the intersection.

[1] https://www.youtube.com/watch?v=RLTZmprHFnw

Algorithm 3: Distributed Algorithm.

Data: Matrix M
Result: Return the set of queue to open $ListOpen$
// initialization
$ListOpen \leftarrow \oslash$;
// get set a queue in process
$RunList \leftarrow \arg\{diag(M) < 0\}$;
$S \leftarrow RunList$;
// find set of queue which is in conflict with queues in Process
for $q \in RunList$ **do**
  $p \leftarrow \arg_{i \in Q} \{M[q,i] = \otimes\}$;
  $S \leftarrow S \cup \{p\}$;
end
$listOrder \leftarrow \arg_{i \in Q \setminus S} \{Sort(M[i,i])\}$;
// find exactly one queue without constraint to open
for $q \in ListOrder$ **do**
  $S \leftarrow S \cup \{q\}$;
  if $Open(q)$ **then**
    $M[q,q] \leftarrow WA = 5$;
    // Take the next queue with highest level
    $continue$;
  else
    $ListOpen \leftarrow \{q\}$;
    $break$;
  end
end
if $ListOpen = \oslash$ **then**
  // End the program if there no queue to open
  $exit()$;
else
  // find the list of queue without conflict with $q \in ListOpen$
  $U_q \leftarrow \arg_{i \in Q \setminus S} \{M[q,i]) \neq \otimes\}$;
  // Find the set of queue to open in parallel with q without conflict
  while $U_q \neq \oslash$ **do**
    // Among those queues, chose the one with highest internal state
    $p \leftarrow \arg\max_{i \in U_q} \{M[i,i]\}$;
    if $Open(p)$ **then**
      // add p in the list of queue to open
      $ListOpen \leftarrow ListOpen \cup \{p\}$;
    end
    // remove p from U_q
    $U_q \leftarrow U_q \setminus \{p\}$;
    // find the list of queue without conflict with p
    $U_p \leftarrow \arg_{i \in U_q} \{M[p,i] \neq \otimes\}$;
    $U_q \leftarrow U_q \cap U_p$
  end
end

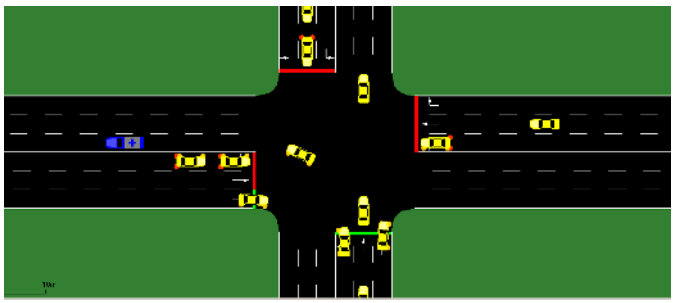
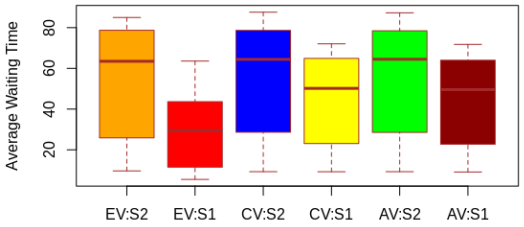


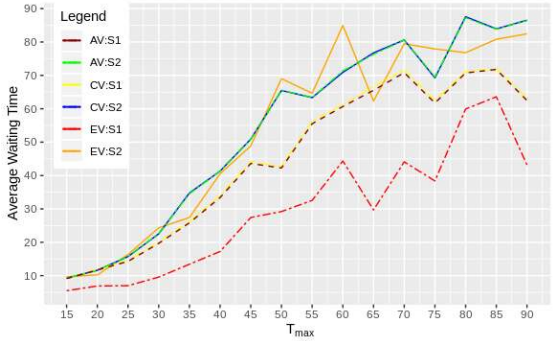
Figure 16: Capture of simulation after 150 steps(150 seconds), Emergency vehicle(EV) in blue and Classic vehicles(CV) in yellow. The video as support materials of the simulation is available here [1].

7.1. Parameters

The green light limit time T_{max} are expected to have a strong influence on the performance result. In case $T_{max} < X_i$ (define time needed to empty queue i see equation 3), the green light queue will be T_{max}. Afterwards, the internal state of the queue will not be set to zero but proportionally to the ratio between T_{max} and X_i. We chose to evaluate the algorithm with two scenarios :(1) S_1 when taking into account the priority of the emergency vehicles; (2) S_2 without considering priority on any vehicles. For both scenarios, we output the average waiting time of emergency vehicles(EV), classic vehicles (CV) and both vehicles (AV). These simulation is conducted for $T_{max} \in [15, 90]$ in steps of 5 seconds.



(a) Multiple average waiting time for scenarios S_1 and S_2 with collision forbidden.



(b) (Vehicle type, scenario), T_{max} influence on average waiting time with collision forbidden.

Figure 17: Performance results.

Figures 17a and 17b present the average waiting time at an intersection for different class of vehicles with and without emergency vehicles. We can first notice that, the scenario S_1 allows reaching the better average waiting time especially the emergency vehicles (EV), which is the target class of vehicle for this scenario. We can also notice the fact the average waiting time is proportional to T_{max}.

7.2. Performance

We compare the results of the distributed algorithm presented above to other achieved works [24],[19],[20] on the same isolated-intersection data set of Amiens city in France. Provided the correct value of T_{max}, our distributed algorithm achieves the best average waiting time with a record of an average vehicle of 2651 instead of 2138 for 3600 steps (3600 s). Moreover, our algorithm is more efficient as it can fast evacuate the emergency vehicles at

the intersection while fulfilling the constraints of zero collisions and no green light with an empty queue. In addition to that, this faster evacuation will be done without significantly increasing the average waiting time of non-emergency vehicles.

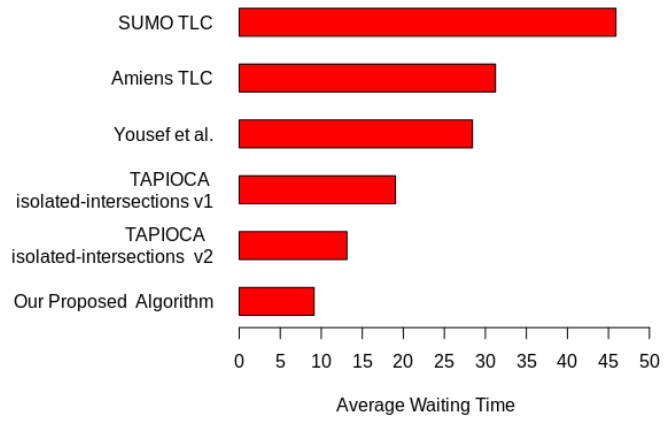


Figure 18: Performance analysis of our result to other result on Amiens isolated dataset [2].

8. Conclusion and Future Works

In this work, we presented a vision-based infrastructure of a decentralized approach for the Intelligent Transportation System(ITS). The main challenges associated with traffic congestion and emergency vehicles were discussed and an adaptive algorithm was presented.

We have modeled an intersection through the conflict matrix, thus giving flexibility in case of a modification of the code of the road, it will be enough to perform an exchange operation on the matrix. The proposed optimized phase-free distributed algorithm for traffic signal control is based on this matrix and it will provide a set of queues without conflict of collisions and the time needed to empty each queue. We also took into account the balance between the queue with heavy traffic and low traffic in order to avoid the problem of an empty queue with a green light.

We also implemented an algorithm to monitor real-time traffic information using cameras. This low cost and smart vision-based infrastructure approach for gathering traffic data replaces the use of WSN coupled with cameras for video-surveillance which are complex to establish.

Our future works will attempt to first extend the simulation to multiple intersections data-set while introducing new elements like pedestrian crossing, and secondly port this work on edge devices to perform computation on the edge.

References

[1] J. L. Hopkins, J. McKay, Investigating 'anywhere working'as a mechanism for alleviating traffic congestion in smart cities, Technological Forecasting and Social Change 142 (2019) 258–272.

[2] J. Wan, Y. Yuan, Q. Wang, Traffic congestion analysis: A new perspective, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 1398–1402.

[3] V. Pattanaik, M. Singh, P. K. Gupta, S. K. Singh, Smart real-time traffic congestion estimation and clustering technique for urban vehicular roads, in: 2016 IEEE Region 10 Conference (TENCON), 2016, pp. 3420–3423.

[4] F. F. H. Administration, 2013 status of the nation's highways, bridges, and transit: Conditions & performance, in: Rep. to Congress, 2013.

[5] J. R. Srivastava, T. Sudarshan, Intelligent traffic management with wireless sensor networks, in: 2013 ACS International Conference on Computer Systems and Applications (AICCSA), IEEE, 2013, pp. 1–4.

[6] J. L. Hopkins, J. McKay, Alleviating traffic congestion around our cities; how can supply chains address the issue?, in: Paper Presented at the 6th International Conference on Operations and Supply Chain Management, Bali, Indonesia, 2014.

[7] A. K. Emo, G. Matthews, G. J. Funke, The slow and the furious: Anger, stress and risky passing in simulated traffic congestion, Transportation research part F: traffic psychology and behaviour 42 (2016) 1–14.

[8] C. Loong, D. van Lierop, A. El-Geneidy, On time and ready to go: An analysis of commuters' punctuality and energy levels at work or school, Transportation research part F: traffic psychology and behaviour 45 (2017) 1–13.

[9] H. M. Sherif, M. A. Shedid, S. A. Senbel, Real time traffic accident detection system using wireless sensor network, in: 2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR), 2014, pp. 59–64.

[10] Aisha Al-Abdallah, Asma Al-Emadi, Mona Al-Ansari, Nassma Mohandes, Qutaibah Malluhi, Real-time traffic surveillance using zigbee, in: 2010 International Conference On Computer Design and Applications, Vol. 1, 2010, pp. V1–550–V1–554.

[11] Z. Lu, G. Qu, Z. Liu, A survey on recent advances in vehicular network security, trust, and privacy, IEEE Transactions on Intelligent Transportation Systems 20 (2) (2019) 760–776.

[12] K. Nellore, G. P. Hancke, A survey on urban traffic management system using wireless sensor networks, Sensors 16 (2) (2016) 157.

[13] P. Santi, Reinvent the traffic light, in: Onward: mobility in the next New York, Urban Design Forum editions, 2018, pp. 135–137.

[14] S. Salman, S. Alaswad, Alleviating road network congestion: Traffic pattern optimization using markov chain traffic assignment, Computers & Operations Research 99 (2018) 191–205.

[15] D. Singh, A. M. Alberti, Developing novagenesis architecture for internet of things services: Observation, challenges and itms application, in: 2014 International Conference on Information and Communication Technology Convergence (ICTC), 2014, pp. 1009–1014.

[16] M. Franceschinis, L. Gioanola, M. Messere, R. Tomasi, M. A. Spirito, P. Civera, Wireless sensor networks for intelligent transportation systems, in: VTC Spring 2009 - IEEE 69th Vehicular Technology Conference, 2009, pp. 1–5.

[17] G. Padmavathi, D. Shanmugapriya, M. Kalaivani, A study on vehicle detection and tracking using wireless sensor networks, Wireless Sensor Network 2 (02) (2010) 173.

[18] A. Frank, Y. S. K. Al Aamri, A. Zayegh, Iot based smart traffic density control using image processing, in: 2019 4th MEC International Conference on Big Data and Smart City (ICBDSC), IEEE, 2019, pp. 1–4.

[19] S. Faye, C. Chaudet, I. Demeure, A distributed algorithm for adaptive traffic lights control, in: Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, IEEE, 2012, pp. 1572–1577.

[20] S. Faye, C. Chaudet, I. Demeure, A distributed algorithm for multiple intersections adaptive traffic lights control using a wire-

[2] http://tapioca.sfaye.com/

less sensor networks, in: Proceedings of the first workshop on Urban networking, ACM, 2012, pp. 13–18.
[21] S. Faye, C. Chaudet, I. Demeure, Multiple intersections adaptive traffic lights control using a wireless sensor networks (2014).
[22] W. Wu, J. Zhang, A. Luo, J. Cao, Distributed mutual exclusion algorithms for intersection traffic control, IEEE Transactions on Parallel and Distributed Systems 26 (1) (2015) 65–74.
[23] T. Wongpiromsarn, T. Uthaicharoenpong, Y. Wang, E. Frazzoli, D. Wang, Distributed traffic signal control for maximum network throughput, in: Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, IEEE, 2012, pp. 588–595.
[24] K. M. Yousef, M. N. Al-Karaki, A. M. Shatnawi, Intelligent traffic light flow control system using wireless sensors networks., J. Inf. Sci. Eng. 26 (3) (2010) 753–768.
[25] Xilinx, Zynq ultrascale+ devices, in: Reference Manual, 2019.
[26] C. Bobda, S. Velipasalar, et al., Distributed Embedded Smart Cameras, Springer, 2014.
[27] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988. doi:10.1109/ICCV.2017.322.
[28] G. Bradski, A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library, " O'Reilly Media, Inc.", 2008.
[29] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools (2000).
[30] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker, Recent development and applications of SUMO - Simulation of Urban MObility, International Journal On Advances in Systems and Measurements 5 (3&4) (2012) 128–138.
URL `http://elib.dlr.de/80483/`