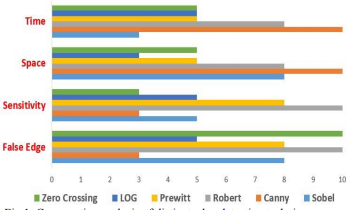
As the problem of urban traffic congestion intensifies, there is a pressing need for the introduction of advanced technology and equipment to improve the state-of-theart of traffic control. The current methods used such as timers or human control are proved to be inferior to alleviate this crisis. In this paper, a system to control the traffic by measuring the realtime vehicle density using canny edge detection with digital image processing is proposed. This imposing traffic control system offers significant improvement in response time, vehicle management, automation, reliability and overall efficiency over the existing systems. Besides that, the complete technique from image acquisition to edge detection and finally green signal allotment using four sample images of different traffic conditions is illustrated with proper schematics and the final results are verified by hardware implementation.

**INTRODUCTION** Traffic congestion is one of the major modern-day crisis in every big city in the world. Recent study of World Bank has shown that average vehicle speed has been reduced from 21 km to 7 km per hour in the last 10 years in Dhaka [1]. Intermetropolitan area studies suggest that traffic congestion reduces regional competitiveness and redistributes economic activity by slowing growth in county gross output or slowing metropolitan area employment growth [2].As more and more vehicles are commissioning in an already congested traffic system, there is an urgent need for a whole new traffic control system using advanced technologies to utilize the already existent infrastructures to its full extent. Since building new roads, flyovers, elevated expressway etc. needs extensive planning, huge capital and lots of time; focus should be directed upon availing existing infrastructures more efficiently and diligently. Previously different techniques had been proposed, such as infra-red light sensor, induction loop etc. to acquire traffic date which had their fair share of demerits. In recent years, image processing has shown promising outcomes in acquiring real time traffic information using CCTV footage installed along the traffic light. Different approaches have been proposed to glean traffic data. Some of them count total number of pixels [3], some of the work calculate number of vehicles [4- 6].These methods have shown promising results in collecting traffic data. However, calculating the number of vehicles may give false results if the intravehicular spacing is very small (two vehicles close to each other may be counted as one) and it may not count rickshaw or auto-rickshaw as vehicles which are the quotidian means of traffic especially in South-Asian countries. And counting number of pixels has disadvantage of counting insubstantial materials as vehicles such as footpath or pedestrians. Some of the work have proposed to allocate time based solely on the density of traffic. But this may be disadvantageous for those who are in lanes that have less frequency of traffic. Edge detection technique is imperative to extract the required traffic information from the CCTV footage. It can be used to isolate the required information from rest of the image. There are several edge detection techniques available. They have distinct characteristics in terms of noise reduction, detection sensitivity, accuracy etc. Among them, Prewitt [7], canny [8],Sobel [9], Roberts and LOG are most accredited operators. It has been observed that the Canny edge detector depicts higher accuracy in detection of object with higher entropy, PSNR(Peak Signal to Noise Ratio), MSE(Mean Square Error) and execution time compared with Sobel, Roberts, Prewitt, Zero crossing and LOG [10-12].Here is a comparison between distinct edge detection techniques



**I**NDIA is the second most populous Country in the World and is a fast growing economy. It is seeing terrible road congestion problems in its cities. Infrastructure growth is slow as compared to the growth in number of vehicles, due to space and cost constraints [1]. Also, Indian traffic is non lane based and chaotic. It needs a traffic control solutions, which are different from the developed Countries. Intelligent management of traffic flows can reduce the negative impact of congestion. In recent years, wireless networks are widely used in the road transport as they provide more cost effective options [2]. Technologies like ZigBee, RFID and GSM can be used in traffic control to provide cost effective solutions. RFID is a wireless technology that uses radio frequency electromagnetic energy to carry information between the RFID tag and RFID reader. Some RFID systems will only work within the range inches or centimeters, while others may work for 100 meters (300 feet) or more. A GSM modem is a specialized type of modem, which accepts a SIM card and operates over a subscription to a mobile operator, just like a mobile phone. AT commands are used to control modems. These commands come from Hayes commands that were used by the Hayes smart modems. The ZigBee operates at low-power and can be used at all the levels of work configurations to perform predefined tasks. It operates in ISM bands (868 MHz in Europe, 915 MHz in USA and Australia, 2.4 GHz in rest of the world). Data transmission rates vary from 20 Kilobits/second in the 868 MHz frequency band to 250 Kilobits/second in the 2.4 GHz frequency band [3], [4]. The ZigBee uses 11 channels in case of 868/915 MHz radio frequency and 16 channels in case of 2*.*4 GHz radio frequency. It also uses 2 channel configurations, CSMA/CA and slotted CSMA/CA [5].

**SURVEY**

Traffic congestion is a major problem in cities of developing Countries like India. Growth in urban population and the middle-class segment contribute significantly to the rising number of vehicles in the cities [6]. Congestion on roads eventually results in slow moving traffic, which increases the time of travel, thus stands-out as one of the major issues in metropolitan cities. In [7], green wave system was discussed, which was used to provide clearance to any emergency vehicle by turning all the red lights to green on the path of the emergency vehicle, hence providing a complete green wave to the desired vehicle. A ‘green wave’ is the synchronization of the green phase of traffic signals. With a ‘green wave’ setup, a vehicle passing through a green signal will continue to receive green signals as it travels down the road. In addition to the green wave path, the system will track a stolen vehicle when it passes through a traffic light. Advantage of the system is that GPS inside the vehicle does not require additional power. The biggest disadvantage of green waves is that, when the wave is disturbed, the disturbance can cause traffic problems that can be exacerbated by the synchronization. In such cases, the queue of vehicles in a green wave grows in size until it becomes too large and some of the vehicles cannot reach the green lights in time and must stop. This is called over-saturation [12], [13]. In [8], the use of RFID traffic control to avoid problems that usually arise with standard traffic control systems, especially those related to image processing and beam interruption techniques are discussed. This RFID technique deals with multivehicle, multilane, multi road junction areas. It provides an efficient time management scheme, in which, a dynamic time schedule is worked out in real time for the passage of each traffic column. The real-time operation of the system emulates the judgment of a traffic policeman on duty. The number of vehicles in each column and the routing are proprieties, upon which the calculations and the judgments are done. The disadvantage of this work is that it does not discuss what methods are used for communication between the emergency vehicle and the traffic signal controller. In [9], it proposed a RFID and GPS based automatic lane clearance system for ambulance. The focus of this work is to reduce the delay in arrival of the ambulance to the hospital by automatically clearing the lane, in which, ambulance is travelling, before it reaches the traffic signal. This can be achieved by turning the

traffic signal, in the path of the ambulance, to green when the ambulance is at a certain distance from the traffic junction. The use of RFID distinguishes between the emergency and non-emergency cases, thus preventing unnecessary traffic congestion. The communication between the ambulance and traffic signal post is done through the transceivers and GPS. The system is fully automated and requires no human intervention at the traffic junctions. The disadvantage of this system is it needs all the information about the starting point, end point of the travel. It may not work, if the ambulance needs to take another route for some reasons or if the starting point is not known in advance. Traffic is a critical issue of transportation system in most of all the cities of Countries. This is especially true for Countries like India and China, where the population is increasing at higher rate as show in figure 1. For example, Bangalore city, has witnessed a phenomenal growth in vehicle population in recent years. As a result, many of the arterial roads and intersections are operating over the capacity (i.e., v/c is more than 1) and average journey speeds on some of the key roads in the central areas are lower than 10 Km/h at the peak hour. In [10], some of the main challenges are management of more than 36,00,000 vehicles, annual growth of 7–10% in traffic, roads operating at higher capacity ranging from 1 to 4, travel speed less than 10 Km/h at some central areas in peak hours, insufficient or no parking space for vehicles, limited number of policemen. In [11], currently a video traffic surveillance and monitoring system commissioned in Bangalore city. It involves a manual analysis of data by the traffic management team to determine the traffic light duration in each of the junction. It will communicate the same to the local police officers for the necessary actions.

**LITERATURE REVIEW:**

**In “Violation detection method for vehicular ad hoc networking,”** Every hour, nearly 40 people under the age of 25 die in road accidents around the world. According to the World Health Organization, this is the second most important cause of death for 5- to 29-year-olds. In India, drunk driving and inefficient law enforcements are major contributing factors. The current system of visual identification of traffic violation, conducted by the traffic authorities, cannot work everywhere and every time. There is a great demand for simple and cost-effective solutions to traffic safety problem. In this paper, we propose a traffic violation detection technique for vehicular ad hoc networks to detect crossing speed limits and analyzing the behavior of driver. In this work, we used a sensor device, a digital map and GPS-based system for area of 1000 m × 1000 m. We analyzed the behavior of each vehicle in the network. Here, we have divided a network into a number of clusters, and each cluster has an infrastructure node (base station); the infrastructure node will be the point of contact for all the vehicles in that area. All infrastructure nodes communicate with a control center (master control room). If the driver violates traffic rule(s), then the infrastructure node will send an alert message to the control center. We have simulated our proposed model on a graphics package, and the simulation result suggests that drunken drivers can no longer escape from the law enforcers, which is the foundation for traffic safety.

**In “Traffic light control in non-stationary environments based on multi agent Q-learning,”** In many urban areas where traffic congestion does not have the peak pattern, conventional traffic signal timing methods does not result in an efficient control. One alternative is to let traffic signal controllers learn how to adjust the lights based on the traffic situation. However this creates a classical non-stationary environment since each controller is adapting to the changes caused by other controllers. In multi-agent learning this is likely to be inefficient and computationally challenging, i.e., the efficiency decreases with the increase in the number of agents (controllers). In this paper, we model a relatively large traffic network as a multi-agent system and use techniques from multi-agent reinforcement learning. In particular, Q-learning is employed, where the average queue length in approaching links is used to estimate states. A parametric representation of the action space has made the method extendable to different types of intersection. The simulation results demonstrate that the proposed Q-learning outperformed the fixed time method under different traffic demands.

**In “A novel approach to implement green wave system and detection of stolen vehicles”** In today's world, traffic jams during rush hours is one of the major concerns. During rush hours, emergency vehicles like Ambulances, Police cars and Fire Brigade trucks get stuck in jams. Due to this, these emergency vehicles are not able to reach their destinations in time, resulting into a loss of human lives. We have developed a system which is used to provide clearance to any emergency vehicle by turning all the red lights to green on the path of the emergency vehicle, hence providing a complete green wave to the desired vehicle. A `green wave' is the synchronization of the green phase of traffic signals. With a `green wave' setup, a vehicle passing through a green signal will continue to receive green signals as it travels down the road. Around the world, green waves are used to great effect. Often criminal or terrorist vehicles have to be identified. In addition to the green wave path, the system will track a stolen vehicle when it passes through a traffic light. In contrast to any traditional vehicle tracking system, in which the Global Positioning System (GPS) module requires battery power, our tracking system, installed inside the vehicle, does not require any power. The information regarding the vehicle has to be updated in the system database. So, it is an autonomous 2-tier system which will help in the identification of emergency vehicles or any other desired vehicle. It is a novel system which can be used to implement the concept of the green wave

**In “Traffic light priority control for emergency vehicle using RFID,”** The proposed RFID traffic control avoids problems that usually arise with standard traffic control systems, especially those related to image processing and beam interruption techniques. This RFID technique deals with a multivehicle, multilane, multi road junction area. It provides an efficient time management scheme, in which a dynamic time schedule is worked out in real time for the passage of each traffic column. The real time operation of the system emulates the judgment of a traffic policeman on duty. The number of vehicles in each column and the routing are proprieties, upon which the calculations and the judgments are based. Keywords-EV (Emergency vehicle),PIC (Priority Intersection Control),RFID TAGS, VTL (Virtual traffic light)

**In “RFID and GPS based automatic lane clearance system for ambulance,”** The exponential growth of the metropolitan cities of the country has generated and magnified urban sprawl into problematic proportions. Lack of efficient traffic control and management has many a times lead to loss of lives due to ambulances getting stuck in traffic jams. To overcome this problem, we propose a RFID and GPS based Automatic Lane Clearance System for Ambulance. The focus of this paper is to reduce the delay in arrival of the ambulance to the hospital by automatically clearing the lane in which ambulance is travelling, before it reaches the traffic signal. This can be achieved by turning the traffic signal, in the path of the ambulance, to green when the ambulance is at a certain distance from the traffic junction. The use of RFID distinguishes between the emergency and non-emergency cases, thus preventing unnecessary traffic congestion. The communication between the ambulance and the traffic signal post is done through transceivers and GPS. The system is fully automated and thus, requires no human intervention at the traffic junctions Bangalore has grown exponentially in the past two decades. Improvement in the quality of life along with substandard public transportation has resulted in spiralling growth of private automobiles. The resultant offshoot of such a high automobile growth is that now Bangalore is one of the most accident-prone cities in India [1].Moreover, the ambulances often get stuck at the traffic signals where all other vehicles try to squeeze in to all the available space so as to move ahead as soon as the signal turns green. Unlike western countries, Indian cities cannot think of having separate lanes for emergency purpose due to lack of road planning and infrastructure. With the lives of the patients depending on the speedy arrival of the ambulances to hospital, an alternative solution to the above problem is the need of the hour. The problem of ambulance getting stuck in a traffic jam can be addressed by ensuring that the lane in which the ambulance is travelling is cleared. That is, the arrival of the ambulance is to be communicated to the nearest traffic signal, so that it can turn the light to green and hence clear the traffic. However, all the ambulances will not be carrying emergency cases. Hence, the traffic clearing system, if done for all the ambulances, will certainly pose a traffic problem. To overcome this difficulty, we propose to make a system combining RFID (Radio Frequency Identification) and GPS

**IN “M. SWEET, “TRAFFIC CONGESTION’S ECONOMIC IMPACTS: EVIDENCE FROM US METROPOLITAN REGIONS,” URBAN STUDIES, VOL. 51, NO. 10, PP. 2088–2110, OCT. 2013.”** Traffic congestion alleviation has long been a common core transport policy objective, but it remains unclear under which conditions this universal byproduct of urban life also impedes the economy. Using panel data for 88 US metropolitan statistical areas, this study estimates congestion’s drag on employment growth (1993 to 2008) and productivity growth per worker (2001 to 2007). Using instrumental variables, results suggest that congestion slows job growth above thresholds of approximately 4.5 minutes of delay per one-way auto commute and 11,000 average daily traffic (ADT) per lane on average across the regional freeway network. While higher ADT per freeway lane appears to slow productivity growth, there is no evidence of congestion-induced travel delay impeding productivity growth. Results suggest that the strict policy focus on travel time savings may be misplaced and, instead, better outlooks for managing congestion’s economic drag lie in prioritising the economically most important trips (perhaps through road pricing) or in providing alternative travel capacity to enable access despite congestion. Planners and policy-makers use both congestion alleviation and mitigating congestion’s economic drag as core justifications for publicly desired and politically favoured transport programmes. Yet while congestion and its potential costs serve as powerful discourses to frame transport policy debates, the precise relationship between road gridlock and economic outcomes is unclear. This research contributes empirically to this gap. Does congestion impede the prospects of a regional economy? Economic and travel behaviour theories reason that congestion is a diseconomy and is inconvenient, but little research explores the more extensive impact of congestion (and congestion alleviation policy) on second-order outcomes, including the economy. This study uses panel data models to estimate congestion’s drag on economic growth in comparison with other explanations of economic outcomes. Understanding the link between congestion and the economy is critical to improving the leveraging of transport and land use policy to support more fundamental social objectives. US federal legislation explicitly identifies congestion reduction and economic support as primary surface transport policy objectives. Yet research on the link between congestion and economic growth is conflicted. The largest urban economies are also among the most congested. Many suggest that traffic congestion reduces city competitiveness and that only peak-period pricing, a highly unpopular tool, can reduce congestion to increase economic function (Boarnet, 1997; Hymel, 2009; Winston and Langer, 2006). Others question the assumption that congestion is an indicator of unsuccessful places and poor social outcomes (Mondschein et al., 2009; Taylor, 2002) or whether long-term congestion alleviation is feasible (Downs, 1992). Instead, the extent and conditions under which congestion impedes social outcomes remain unclear and effective politically acceptable solutions to this murky problem remain even more elusive. There are many important causes of economic growth, of which this study focuses on traffic congestion’s potential drag on regional employment growth and productivity growth per worker. Each of these two metrics of economic activity contributes to total productivity (the sum of employment and productivity per worker), thereby focusing on how congestion might influence regional economic growth by either impeding worker productivity or slowing the hiring of new workers. Existing literature on congestion’s diseconomy focuses on different scales at which congestion can potentially slow economic growth (within or between regions). Road congestion is an external byproduct of other common causes of economic growth, including big-city agglomeration benefits, social preferences and affluence, urban spatial structure, and municipal governance and therefore separating ‘good’ from ‘bad’ congestion is challenging Research on congestion’s economic consequences explores differences in regional or firm productivity, city growth and relocation responses by individuals and firms. The relationship between metropolitan economic activity and traffic congestion is complex and unclear (Taylor, 2002). Large regional economies lead to more congestion, while congestion may impede economic activities by degrading mobility services. Travel is a direct economic input which also leads to the congestion externality. In econometrics, this issue is called endogeneity and captures the methodological challenges of separating the competing benefits of big-city access and dense travel patterns from the drag of big-city road gridlock which raises travel costs or increases unreliability. Congestion reduces national (Fernald, 1999) and regional (Boarnet, 1997; Hymel, 2009) economic competitiveness across regions, but firms and workers adapt within regions through location decisions and bearing commuting burdens (Cervero, 1996; Gordon et al., 1989). Thus, while congestion can potentially lead to travel and economic inefficiencies, it is unclear under what circumstances urbanisation benefits and adaptations by individuals, firms or through policy can no longer outweigh congestion’s potential drag (Sweet, 2011). Intrametropolitan studies of traffic congestion’s economic consequences suggest that it shapes regional geographies, but that it is unclear whether resident and firm adjustments can overcome the impact of congestion on urban function. According to the co-location hypothesis, congestion simply induces employer-employee suburban co-location (Crane and Chatman, 2003; Gordon et al., 1989; Levinson and Kumar, 1994). In contrast, empirical research on job–housing imbalance (Cervero, 1996; Cervero and Wu, 1998; Schwanen et al., 2004) suggests significant commuting burdens while theoretical urban economic models likewise imply congestion-induced urban economic inefficiencies (Arnott, 2007; Anas and Xu, 1999; Fujita and Thisse, 2002; Weisbrod, Vary, and Treyz, 2001), most notably by reducing agglomeration benefits (Graham, 2007). Moreover, research suggests industry-variant sensitivity to congestion’s potential drag—most notably, service industries are least sensitive while manufacturing industries are most sensitive, indicating that industry mix is important. Intermetropolitan area studies suggest that traffic congestion reduces regional competitiveness and redistributes economic activity by slowing growth in county gross output (Boarnet, 1997) or slowing metropolitan area employment growth (Hartgen and Fields, 2009; Hymel, 2009). Boarnet (1997) finds that congestion reduces productivity in California counties. Similarly, using panel data for major American metropolitan areas, Hymel (2009) finds that higher congestion leads to slower employment growth, but that its short-term job growth impacts are stronger than those over the longer term—implying regional adaptation. Thus, while intrametropolitan studies suggest that firms and residents adapt to congestion, intermetropolitan studies suggest that such adaptations may not overcome congestion’s regionally scaled drag.

**IN “MD. MUNIR HASAN, GOBINDA SAHA, AMINUL HOQUE AND MD. BADRUDDOJA MAJUMDER, “SMART TRAFFIC CONTROL SYSTEM WITH APPLICATION OF IMAGE PROCESSING TECHNIQUES,”IN 3RD INTERNATIONAL CONFERENCE ON INFORMATICS, ELECTRONICS & VISION, DHAKA, MAY 2014.”** In this paper we propose a method for determining traffic congestion on roads using image processing techniques and a model for controlling traffic signals based on information received from images of roads taken by video camera. We extract traffic density which corresponds to total area occupied by vehicles on the road in terms of total amount of pixels in a video frame instead of calculating number of vehicles. We set two parameters as output, variable traffic cycle and weighted time for each road based on traffic density and control traffic lights in a sequential manner Traffic congestion has become a major problem in every large city of the world. To ensure a reliable transportation system it is important to have an intelligent traffic control system. The very first step to do that is to acquire traffic data. Traffic data may come from different sensors. Some examples are use of induction loop, infra-red light sensor, optical flow etc. However in recent days image processing techniques [1] has been very important and promising topic to deal with traffic related problems because of its ease of maintenance and being more intelligent system. Different techniques [2]– [5] have been proposed to acquire traffic information. Most of the work detects edge of the vehicles and counts the number of traffic on the road. However the disadvantage of the method is that counting the number of vehicles may give faulty results when space between the vehicles on the road are very small (i.e. two cars very close to each other may be counted as one vehicle). Moreover most of the methods treat only cars as traffic but in many part of the world rickshaws, autorickshaws, bikes are major part of everyday traffic such as in south asian countries. In this paper we propose a method that finds out total amount of pixels in a video frame which corresponds to the amount of area of occupied by vehicles on the road rather than finding number of vehicles. The greater the amount of area occupied by vehicles on the road the greater the amount of traffic congestion. This way every kind of vehicles can be accounted for traffic density. Using this traffic data we propose a model for traffic signal control depending on the amount of traffic on the road. Time allocated for each road is made variable by weighing its time allocation depending on the traffic density In this section we present a process of extracting traffic information from image. We assume that a video camera, placed at appropriate position, is employed for image accusation. From the camera video stream data is processed frame by frame. Our goal is to determine how much traffic is on the road. The amount of traffic will be termed Traffic Density in this paper. To accomplish this task we will use background subtraction method [6]. The background subtraction method is particularly suitable for detecting a foreground objects on fixed background. Here the empty road will be the background image and subsequent frames from the video camera will be the foreground image. By subtracting background image from the foreground image we can find out traffic density present in a frame. We present two methods to find traffic density and both methods will be used simultaneously. One is using gradient magnitude and other using direct subtraction.

**IN “VISMAY PANDIT, JINESH DOSHI, DHRUV MEHTA, ASHAY MHATRE AND ABHILASH JANARDHAN, “SMART TRAFFIC CONTROL SYSTEM USING IMAGE PROCESSING,”INTERNATIONAL JOURNAL OF EMERGING TRENDS & TECHNOLOGY IN COMPUTER SCIENCE (IJETTCS), VOL. 3, ISSUE 1, JANUARY – FEBRUARY 2014”** –The fact is that, the population of city and numbers of vehicles on the road are increasing day by day. With increasing urban population and hence the number of vehicles, need of controlling streets, highways and roads is major issue. The main reason behind today’s traffic problem is the techniques that are used for traffic management. Today’s traffic management system has no emphasis on live traffic scenario, which leads to inefficient traffic management systems. This project has been implemented by using the Mat lab software and it aims to prevent heavy traffic congestion. Moreover, for implementing this project Image processing technique is used. At first, film of a lane is captured by a camera. A web camera is placed in a traffic lane that will capture images of the road on which we want to control traffic. Then these images are efficiently processed to know the traffic density. According to the processed data from mat lab, the controller will send the command to the traffic LEDs to show particular time on the signal to manage traffic. Fast transportation systems and rapid transit systems are nerves of economic developments for any nation. Mismanagement and traffic congestion results in long waiting times, loss of fuel and money. It is therefore utmost necessary to have a fast, economical and efficient traffic control system for national development. The monitoring and control of city traffic is becoming a major problem in many countries. With the ever increasing number of vehicles on the road, the Traffic Monitoring Authority has to find new methods of overcoming such a problem. One way to improve traffic flow and Safety of the current transportation system is to apply automation and Intelligent control methods. As the Number of road users constantly increases, and resources provided by current infrastructures are limited, intelligent control of traffic will become a very important issue in the future [1]. Traffic congestion may result due to heavy traffic at a junction. To avoid congestion there are so many traffic management techniques available. But no technique is perfect by itself as the real time situations are generally continuously changing and the system has to adapt itself to change in the continuously changing circumstances. We have made an attempt to provide some traffic management strategy which is self-changing in nature, so as to fit into continuously changing real time traffic scenarios. In this system time is assigned to traffic light of particular lane according to the traffic density on the road with priority given to ambulance. Also we can indicate signal break in a particular lane. If there is an obstacle LCD is used to display the message of obstacle detection to avoid inconvenience. Objective of proposed system is to improve efficiency of existing automatic traffic signalling system. The system will be image processing based adaptive signal controlling. The timing will be calculated each time change automatically depending upon the traffic load. Proposed system will be functioning based on traditional system along with automated signalling. System will have artificial vision with the help of digital camera mounted on motor for its rotation to face lanes and sense the traffic on the road. The camera is controlled by PC through microcontroller to change its direction in steps of 90 degree to face each lane and capture image. This single image of lane will be processed using image processing techniques to estimate traffic load. Estimated traffic load on particular road will be used to calculate the required time duration for controlling of signal lights based on in comparison with experimental results. System will be intelligent and will calculate the time every time and operate in a cyclic clockwise signal lights control. Maximum and minimum time limit will be maintained to prevent over waiting of vehicle in queue of other lanes which would be found out experimentally. Controls of the signal will be routed through the microcontroller. MATLAB programming will be used for simulating and developing the proposed system. The signal will be controlled by interrupting the normal functioning. The emergency will set the priority and the requested lane will be open closing all others. After emergency is removed the system starts normal functioning. The main aim in designing and developing of the Smart Traffic Signal Simulator is to reduce the waiting time of each lane of the cars and also to maximize the total number of cars that can cross an intersection given the mathematical function to calculate the waiting time. The traffic signal system consists of three important parts. • The first part is the controller which represents the brain of the traffic system.

**IN “PALLAVI CHOUDEKAR, SAYANTI BANERJEE AND M. K. MUJU, “IMPLEMENTATION OF IMAGE PROCESSING IN REAL TIME TRAFFIC LIGHT CONTROL,” IN3RD INTERNATIONAL CONFERENCE ON ELECTRONICS COMPUTER TECHNOLOGY, APRIL, 2011.”** — As the problem of urban traffic congestion spreads, there is a pressing need for the introduction of advanced technology and equipment to improve the state-of-the-art of traffic control. Traffic problems nowadays are increasing because of the growing number of vehicles and the limited resources provided by current infrastructures. The simplest way for controlling a traffic light uses timer for each phase. Another way is to use electronic sensors in order to detect vehicles, and produce signal that cycles. We propose a system for controlling the traffic light by image processing. The system will detect vehicles through images instead of using electronic sensors embedded in the pavement. A camera will be installed alongside the traffic light. It will capture image sequences. Setting image of an empty road as reference image, the captured images are sequentially matched using image matching. For this purpose edge detection has been carried out using Prewitt edge detection operator and according to percentage of matching traffic light durations can be controlled. Automatic traffic monitoring and surveillance are important for road usage and management. Traffic parameter estimation has been an active research area for the development of intelligent Transportation systems (ITS). For ITS applications traffic- information needs to be collected and distributed. Various sensors have been employed to estimate traffic parameters for updating traffic information. Magnetic loop detectors have been the most used technologies, but their installation and maintenance are inconvenient and might become incompatible with future ITS infrastructure. It is well recognized that vision-based camera system are more versatile for traffic parameter estimation [l,4]. In addition to qualitative description of road congestion, image measurement can provide quantitative description of traffic status including speeds, vehicle counts, etc. Moreover, quantitative traffic parameters can give us complete traffic flow information, which fulfills the requirement of traffic management theory. Image tracking of moving vehicles can give us quantitative description of traffic flow

BACKGROUND:

Digital image processing is an area characterized by the need for extensive experimental work to establish the viability of proposed solutions to a given problem. An important characteristic underlying the design of image processing systems is the significant level of testing & experimentation that normally is required before arriving at an acceptable solution. This characteristic implies that the ability to formulate approaches &quickly prototype candidate solutions generally plays a major role in reducing the cost & time required to arrive at a viable system implementation.

**What is IMAGE PROCESSING?**

An image may be defined as a two-dimensional function f(x, y), where x & y are spatial coordinates, & the amplitude of f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x, y & the amplitude values of f are all finite discrete quantities, we call the image a digital image. The field of DIP refers to processing digital image by means of digital computer. Digital image is composed of a finite number of elements, each of which has a particular location & value. The elements are called pixels.

Vision is the most advanced of our sensor, so it is not surprising that image play the single most important role in human perception. However, unlike humans, who are limited to the visual band of the EM spectrum imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves. They can operate also on images generated by sources that humans are not accustomed to associating with image.

There is no general agreement among authors regarding where image processing stops & other related areas such as image analysis& computer vision start. Sometimes a distinction is made by defining image processing as a discipline in which both the input & output at a process are images. This is limiting & somewhat artificial boundary. The area of image analysis (image understanding) is in between image processing & computer vision

There are no clear-cut boundaries in the continuum from image processing at one end to complete vision at the other. However, one useful paradigm is to consider three types of computerized processes in this continuum: low-, mid-, & high-level processes. Low-level process involves primitive operations such as image processing to reduce noise, contrast enhancement & image sharpening. A low- level process is characterized by the fact that both its inputs & outputs are images. Mid-level process on images involves tasks such as segmentation, description of that object to reduce them to a form suitable for computer processing & classification of individual objects. A mid-level process is characterized by the fact that its inputs generally are images but its outputs are attributes extracted from those images. Finally higher- level processing involves “Making sense” of an ensemble of recognized objects, as in image analysis & at the far end of the continuum performing the cognitive functions normally associated with human vision

Digital image processing, as already defined is used successfully in a broad range of areas of exceptional social & economic value.

**WHAT IS AN IMAGE?**

An image is represented as a two dimensional function f(x, y) where x and y are spatial co-ordinates and the amplitude of ‘f’ at any pair of coordinates (x, y) is called the intensity of the image at that point.

**Gray scale image:**

A grayscale image is a function I(xylem) of the two spatial coordinates of the image plane.

I(x, y) is the intensity of the image at the point (x, y) on the image plane.

I (xylem)takes non-negative values assume the image is bounded by arectangle[0, a] ×[0, b]I: [0, a] × [0, b] → [0, info)

**Color image:**

**It** can be represented by three functions, R (xylem)for red,G (xylem)for green *and* B (xylem)for blue.

An image may be continuous with respect to the x and y coordinates and also in amplitude. Converting such an image to digital form requires that the coordinates as well as the amplitude to be digitized. Digitizing the coordinate’s values is calledsampling. Digitizing the amplitude values is called quantization.

**Coordinate convention:**

The result of sampling and quantization is a matrix of real numbers. We use two principal ways to represent digital images. Assume that an image f(x, y) is sampled so that the resulting image has M rows and N columns. We say that the image is of size M X N. The values of the coordinates (xylem) are discrete quantities. For notational clarity and convenience, we use integer values for these discrete coordinates. In many image processing books, the image origin is defined to be at (xylem)=(0,0).The next coordinate values along the first row of the image are (xylem)=(0,1).It is important to keep in mind that the notation (0,1) is used to signify the second sample along the first row. It does not mean that these are the actual values of physical coordinates when the image was sampled. Following figure shows the coordinate convention. Note that x ranges from 0 to M-1 and y from 0 to N-1 in integer increments.

The coordinate convention used in the toolbox to denote arrays is different from the preceding paragraph in two minor ways. First, instead of using (xylem) the toolbox uses the notation (race) to indicate rows and columns. Note, however, that the order of coordinates is the same as the order discussed in the previous paragraph, in the sense that the first element of a coordinate topples, (alb), refers to a row and the second to a column. The other difference is that the origin of the coordinate system is at (r, c) = (1, 1); thus, r ranges from 1 to M and c from 1 to N in integer increments. IPT documentation refers to the coordinates. Less frequently the toolbox also employs another coordinate convention called spatial coordinates which uses x to refer to columns and y to refers to rows. This is the opposite of our use of variables x and y.

# Image as Matrices:

The preceding discussion leads to the following representation for a digitized image function:

f (0,0) f(0,1) ……….. f(0,N-1)

f(1,0) f(1,1) ………… f(1,N-1)

f(xylem)= . . .

. .

f(M-1,0) f(M-1,1) ………… f(M-1,N-1)

The right side of this equation is a digital image by definition. Each element of this array is called an image element, picture element, pixel or pel. The terms image and pixel are used throughout the rest of our discussions to denote a digital image and its elements.

A digital image can be represented naturally as a MATLAB matrix:

f(1,1) f(1,2) ……. f(1,N)

f(2,1) f(2,2) …….. f(2,N)

. . .

f = . . .

f(M,1) f(M,2) …….f(M,N)

Where f(1,1) = f(0,0) (note the use of a monoscope font to denote MATLAB quantities). Clearly the two representations are identical, except for the shift in origin. The notation f (p ,q) denotes the element located in row p and the column q. For example f (6, 2) is the element in the sixth row and second column of the matrix f. Typically we use the letters M and N respectively to denote the number of rows and columns in a matrix. A 1xN matrix is called a row vector whereas an Mx1 matrix is called a column vector. A 1x1 matrix is a scalar.

Matrices in MATLAB are stored in variables with names such as A, a, RGB, real array and so on. Variables must begin with a letter and contain only letters, numerals and underscores. As noted in the previous paragraph, all MATLAB quantities are written using mono-scope characters. We use conventional Roman, italic notation such as f(x ,y), for mathematical expressions

# Reading Images:

Images are read into the MATLAB environment using function imread whose syntax is

imread(‘filename’)

**Format name Description recognized extension**

TIFF Tagged Image File Format .tif, .tiff

JPEG Joint Photograph Experts Group .jpg, .jpeg

GIF Graphics Interchange Format .gif

BMP Windows Bitmap .bmp

PNG Portable Network Graphics .png

XWD X Window Dump .xwd

**What is Image Segmentation?**

Let’s understand image segmentation using a simple example. Consider the below image:

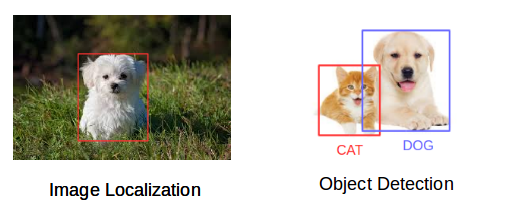


There’s only one object here – a dog. We can build a straightforward cat-dog classifier model and predict that there’s a dog in the given image. But what if we have both a cat and a dog in a single image?



We can train a multi-label classifier, in that instance. Now, there’s another caveat – we won’t know the location of either animal/object in the image.

That’s where image localization comes into the picture (no pun intended!). It helps us to identify the location of a single object in the given image. In case we have multiple objects present, we then rely on the concept of [object detection](https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/?utm_source=blog&utm_medium=introduction-image-segmentation-techniques-python) (OD). We can predict the location along with the class for each object using OD.

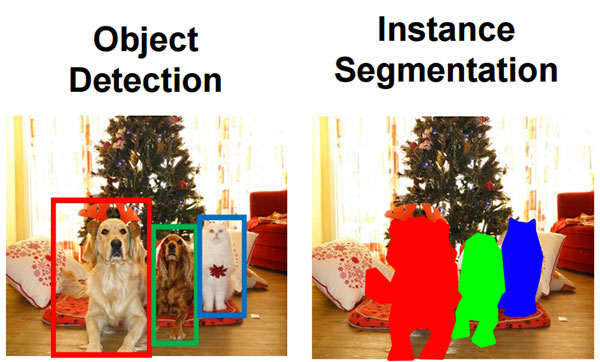


Before detecting the objects and even before classifying the image, we need to understand what the image consists of. Enter – Image Segmentation.

**So how does image segmentation work?**

We can divide or partition the image into various parts called segments. It’s not a great idea to process the entire image at the same time as there will be regions in the image which do not contain any information. By dividing the image into segments, we can make use of the important segments for processing the image. That, in a nutshell, is how image segmentation works.

An image is a collection or set of different pixels. We group together the pixels that have similar attributes using image segmentation. Take a moment to go through the below visual (it’ll give you a practical idea of image segmentation):

*Source : cs231n.stanford.edu*

Object detection builds a bounding box corresponding to each class in the image. But it tells us nothing about the shape of the object. We only get the set of bounding box coordinates. We want to get more information – this is too vague for our purposes.

Image segmentation creates a pixel-wise mask for each object in the image. This technique gives us a far more granular understanding of the object(s) in the image.

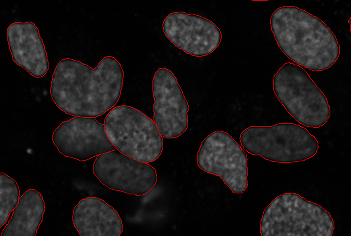
Why do we need to go this deep? Can’t all image processing tasks be solved using simple bounding box coordinates? Let’s take a real-world example to answer this pertinent question.

**Why do we need Image Segmentation?**

Cancer has long been a deadly illness. Even in today’s age of technological advancements, cancer can be fatal if we don’t identify it at an early stage. Detecting cancerous cell(s) as quickly as possible can potentially save millions of lives.

The shape of the cancerous cells plays a vital role in determining the severity of the cancer. You might have put the pieces together – object detection will not be very useful here. We will only generate bounding boxes which will not help us in identifying the shape of the cells.

Image Segmentation techniques make a MASSIVE impact here. They help us approach this problem in a more granular manner and get more meaningful results. A win-win for everyone in the healthcare industry.

*Source: Wikipedia*

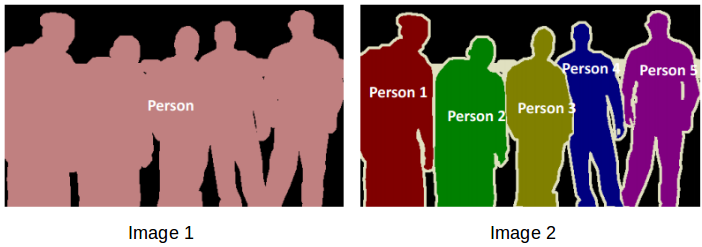
Here, we can clearly see the shapes of all the cancerous cells. There are many other applications where Image segmentation is transforming industries:

* Traffic Control Systems
* Self Driving Cars
* Locating objects in satellite images

There are even more applications where Image Segmentation is very useful. Feel free to share them with me in the comments section below this article – let’s see if we can build something together.

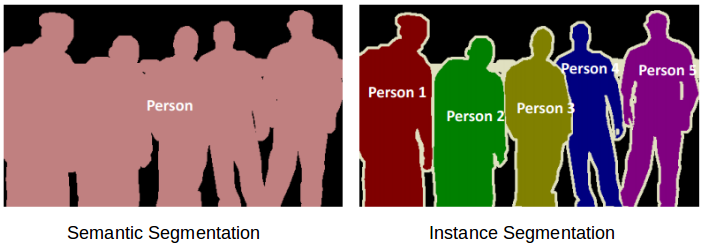
**The Different Types of Image Segmentation**

We can broadly divide image segmentation techniques into two types. Consider the below images:



Can you identify the difference between these two? Both the images are using image segmentation to identify and locate the people present.

* In image 1, every pixel belongs to a particular class (either background or person). Also, all the pixels belonging to a particular class are represented by the same color (background as black and person as pink). This is an example of semantic segmentation
* Image 2 has also assigned a particular class to each pixel of the image. However, different objects of the same class have different colors (Person 1 as red, Person 2 as green, background as black, etc.). This is an example of instance segmentation



Let me quickly summarize what we’ve learned. If there are 5 people in an image, semantic segmentation will focus on classifying all the people as a single instance. Instance segmentation, on the other hand. will identify each of these people individually.

So far, we have delved into the theoretical concepts of image processing and segmentation. Let’s mix things up a bit – we’ll combine learning concepts with implementing them in Python. I strongly believe that’s the best way to learn and remember any topic.

**Region-based Segmentation**

One simple way to segment different objects could be to use their pixel values. An important point to note – the pixel values will be different for the objects and the image’s background if there’s a sharp contrast between them.

In this case, we can set a threshold value. The pixel values falling below or above that threshold can be classified accordingly (as an object or the background). This technique is known as **Threshold Segmentation**.

*If we want to divide the image into two regions (object and background), we define a single threshold value. This is known as the****global threshold****.*

*If we have multiple objects along with the background, we must define multiple thresholds. These thresholds are collectively known as the****local threshold****.*

Let’s implement what we’ve learned in this section. Download [**this image**](https://drive.google.com/open?id=1aM4otWKSsDz1Rof3LZkY055YkYXeO-vf) and run the below code. It will give you a better understanding of how thresholding works (you can use any image of your choice if you feel like experimenting!).

First, we’ll import the required libraries.

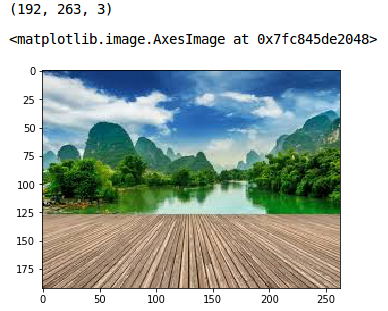
|  |  |
| --- | --- |
|  | from skimage.color import rgb2gray |
|  | import numpy as np |
|  | import cv2 |
|  | import matplotlib.pyplot as plt |
|  | %matplotlib inline |
|  | from scipy import ndimage |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/import_library.py)[**import\_library.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-import_library-py) hosted with  by [**GitHub**](https://github.com/)

Let’s read the downloaded image and plot it:

|  |  |
| --- | --- |
|  | image = plt.imread('1.jpeg') |
|  | image.shape |
|  | plt.imshow(image) |

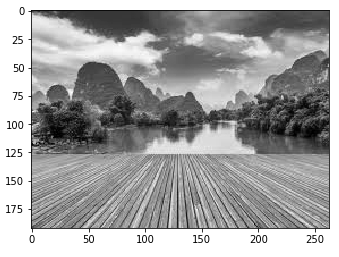
[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/read_image_1.py)[**read\_image\_1.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-read_image_1-py) hosted with  by [**GitHub**](https://github.com/)



It is a three-channel image (RGB). We need to convert it into grayscale so that we only have a single channel. Doing this will also help us get a better understanding of how the algorithm works.

|  |  |
| --- | --- |
|  | gray = rgb2gray(image) |
|  | plt.imshow(gray, cmap='gray') |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/gray_scale.py)[**gray\_scale.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-gray_scale-py) hosted with  by [**GitHub**](https://github.com/)



Now, we want to apply a certain threshold to this image. This threshold should separate the image into two parts – the foreground and the background. Before we do that, let’s quickly check the shape of this image:

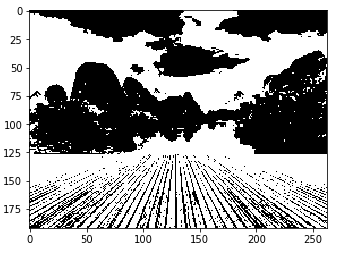
gray.shape

(192, 263)

The height and width of the image is 192 and 263 respectively. **We will take the mean of the pixel values and use that as a threshold.** If the pixel value is more than our threshold, we can say that it belongs to an object. If the pixel value is less than the threshold, it will be treated as the background. Let’s code this:

|  |  |
| --- | --- |
|  | gray\_r = gray.reshape(gray.shape[0]\*gray.shape[1]) |
|  | for i in range(gray\_r.shape[0]): |
|  | if gray\_r[i] > gray\_r.mean(): |
|  | gray\_r[i] = 1 |
|  | else: |
|  | gray\_r[i] = 0 |
|  | gray = gray\_r.reshape(gray.shape[0],gray.shape[1]) |
|  | plt.imshow(gray, cmap='gray') |

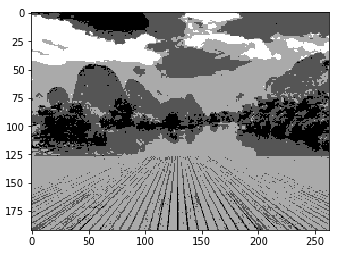
[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/global_threshold.py)[**global\_threshold.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-global_threshold-py) hosted with  by [**GitHub**](https://github.com/)



Nice! The darker region (black) represents the background and the brighter (white) region is the foreground. We can define multiple thresholds as well to detect multiple objects:

|  |  |
| --- | --- |
|  | gray = rgb2gray(image) |
|  | gray\_r = gray.reshape(gray.shape[0]\*gray.shape[1]) |
|  | for i in range(gray\_r.shape[0]): |
|  | if gray\_r[i] > gray\_r.mean(): |
|  | gray\_r[i] = 3 |
|  | elif gray\_r[i] > 0.5: |
|  | gray\_r[i] = 2 |
|  | elif gray\_r[i] > 0.25: |
|  | gray\_r[i] = 1 |
|  | else: |
|  | gray\_r[i] = 0 |
|  | gray = gray\_r.reshape(gray.shape[0],gray.shape[1]) |
|  | plt.imshow(gray, cmap='gray') |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/local_threshold.py)[**local\_threshold.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-local_threshold-py) hosted with  by [**GitHub**](https://github.com/)



There are four different segments in the above image. You can set different threshold values and check how the segments are made. Some of the advantages of this method are:

* Calculations are simpler
* Fast operation speed
* When the object and background have high contrast, this method performs really well

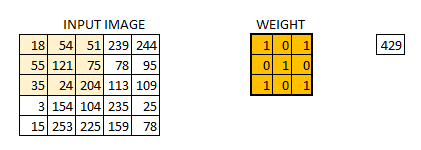
But there are some limitations to this approach. When we don’t have significant grayscale difference, or there is an overlap of the grayscale pixel values, it becomes very difficult to get accurate segments.

**Edge Detection Segmentation**

What divides two objects in an image? There is always an edge between two adjacent regions with different grayscale values (pixel values). The edges can be considered as the discontinuous local features of an image.

We can make use of this discontinuity to detect edges and hence define a boundary of the object. This helps us in detecting the shapes of multiple objects present in a given image. Now the question is how can we detect these edges? This is where we can make use of filters and convolutions. Refer to [this article](https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/?utm_source=blog&utm_medium=image-segmentation-article) if you need to learn about these concepts.

The below visual will help you understand how a filter colvolves over an image :



Here’s the step-by-step process of how this works:

* Take the weight matrix
* Put it on top of the image
* Perform element-wise multiplication and get the output
* Move the weight matrix as per the stride chosen
* Convolve until all the pixels of the input are used

The values of the weight matrix define the output of the convolution. My advice – it helps to extract features from the input. Researchers have found that choosing some specific values for these weight matrices helps us to detect horizontal or vertical edges (or even the combination of horizontal and vertical edges).

One such weight matrix is the sobel operator. It is typically used to detect edges. The sobel operator has two weight matrices – one for detecting horizontal edges and the other for detecting vertical edges. Let me show how these operators look and we will then implement them in Python.

Sobel filter (horizontal) =

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

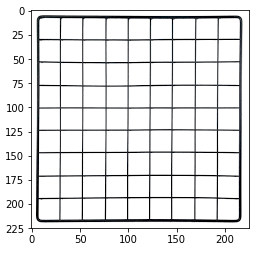
Sobel filter (vertical) =

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Edge detection works by convolving these filters over the given image. Let’s visualize them on [this article](https://drive.google.com/file/d/1gohzcl0AI1yVCF3zdGZnfcbvI_1poM32/view?usp=sharing).

|  |  |
| --- | --- |
|  | image = plt.imread('index.png') |
|  | plt.imshow(image) |

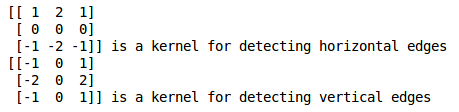
[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/read_image_2.py)[**read\_image\_2.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-read_image_2-py) hosted with  by [**GitHub**](https://github.com/)



It should be fairly simple for us to understand how the edges are detected in this image. Let’s convert it into grayscale and define the sobel filter (both horizontal and vertical) that will be convolved over this image:

|  |  |
| --- | --- |
|  | # converting to grayscale |
|  | gray = rgb2gray(image) |
|  |  |
|  | # defining the sobel filters |
|  | sobel\_horizontal = np.array([np.array([1, 2, 1]), np.array([0, 0, 0]), np.array([-1, -2, -1])]) |
|  | print(sobel\_horizontal, 'is a kernel for detecting horizontal edges') |
|  |  |
|  | sobel\_vertical = np.array([np.array([-1, 0, 1]), np.array([-2, 0, 2]), np.array([-1, 0, 1])]) |
|  | print(sobel\_vertical, 'is a kernel for detecting vertical edges') |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/sobel_filters.py)[**sobel\_filters.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-sobel_filters-py) hosted with  by [**GitHub**](https://github.com/)



Now, convolve this filter over the image using the *convolve* function of the *ndimage* package from *scipy*.

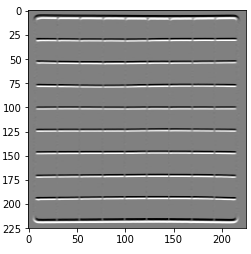
|  |  |
| --- | --- |
|  | out\_h = ndimage.convolve(gray, sobel\_horizontal, mode='reflect') |
|  | out\_v = ndimage.convolve(gray, sobel\_vertical, mode='reflect') |
|  | # here mode determines how the input array is extended when the filter overlaps a border. |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/convolving_sobel_filters.py)[**convolving\_sobel\_filters.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-convolving_sobel_filters-py) hosted with  by [**GitHub**](https://github.com/)

Let’s plot these results:

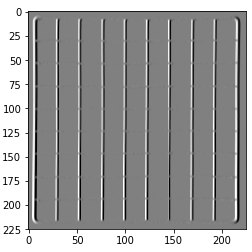
|  |  |
| --- | --- |
|  | plt.imshow(out\_h, cmap='gray') |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/plot_1.py)[**plot\_1.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-plot_1-py) hosted with  by [**GitHub**](https://github.com/)



|  |  |
| --- | --- |
|  | plt.imshow(out\_v, cmap='gray') |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/plot_2.py)[**plot\_2.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-plot_2-py) hosted with  by [**GitHub**](https://github.com/)



Here, we are able to identify the horizontal as well as the vertical edges. There is one more type of filter that can detect both horizontal and vertical edges at the same time. This is called the laplace operator:

|  |  |  |
| --- | --- | --- |
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

Let’s define this filter in Python and convolve it on the same image:

|  |  |
| --- | --- |
|  | kernel\_laplace = np.array([np.array([1, 1, 1]), np.array([1, -8, 1]), np.array([1, 1, 1])]) |
|  | print(kernel\_laplace, 'is a laplacian kernel') |

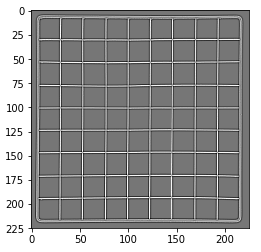
[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/laplacian_filter.py)[**laplacian\_filter.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-laplacian_filter-py) hosted with  by [**GitHub**](https://github.com/)

laplacian filter

Next, convolve the filter and print the output:

|  |  |
| --- | --- |
|  | out\_l = ndimage.convolve(gray, kernel\_laplace, mode='reflect') |
|  | plt.imshow(out\_l, cmap='gray') |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/convolving_laplacian_filter.py)[**convolving\_laplacian\_filter.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-convolving_laplacian_filter-py) hosted with  by [**GitHub**](https://github.com/)



Here, we can see that our method has detected both horizontal as well as vertical edges. I encourage you to try it on different images and share your results with me. Remember, the best way to learn is by practicing!

**Image Segmentation based on Clustering**

This idea might have come to you while reading about image segmentation. Can’t we use clustering techniques to divide images into segments? We certainly can!

In this section, we’ll get an an intuition of what clustering is (it’s always good to revise certain concepts!) and how we can use of it to segment images.

Clustering is the task of dividing the population (data points) into a number of groups, such that data points in the same groups are more similar to other data points in that same group than those in other groups. These groups are known as clusters.

One of the most commonly used clustering algorithms is [k-means](https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/). Here, the k represents the number of clusters (not to be confused with k-nearest neighbor). Let’s understand how k-means works:

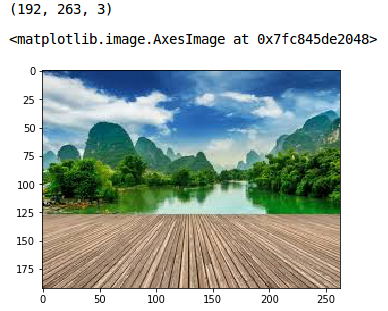
1. First, randomly select k initial clusters
2. Randomly assign each data point to any one of the k clusters
3. Calculate the centers of these clusters
4. Calculate the distance of all the points from the center of each cluster
5. Depending on this distance, the points are reassigned to the nearest cluster
6. Calculate the center of the newly formed clusters
7. Finally, repeat steps (4), (5) and (6) until either the center of the clusters does not change or we reach the set number of iterations

**The key advantage of using k-means algorithm is that it is simple and easy to understand.** We are assigning the points to the clusters which are closest to them.

Let’s put our learning to the test and check how well k-means segments the objects in an image. We will be using [**this image**](https://drive.google.com/open?id=1aM4otWKSsDz1Rof3LZkY055YkYXeO-vf), so download it, read it and and check its dimensions:

|  |  |
| --- | --- |
|  | pic = plt.imread('1.jpeg')/255 # dividing by 255 to bring the pixel values between 0 and 1 |
|  | print(pic.shape) |
|  | plt.imshow(pic) |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/read_image_3.py)[**read\_image\_3.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-read_image_3-py) hosted with  by [**GitHub**](https://github.com/)



It’s a 3-dimensional image of shape (192, 263, 3). For clustering the image using k-means, we first need to convert it into a 2-dimensional array whose shape will be (length\*width, channels). In our example, this will be (192\*263, 3).

|  |  |
| --- | --- |
|  | pic\_n = pic.reshape(pic.shape[0]\*pic.shape[1], pic.shape[2]) |
|  | pic\_n.shape |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/reshaping_image.py)[**reshaping\_image.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-reshaping_image-py) hosted with  by [**GitHub**](https://github.com/)

**(50496, 3)**

We can see that the image has been converted to a 2-dimensional array. Next, fit the k-means algorithm on this reshaped array and obtain the clusters. The *cluster\_centers\_ function* of k-means will return the cluster centers and *labels\_ function* will give us the label for each pixel (it will tell us which pixel of the image belongs to which cluster).

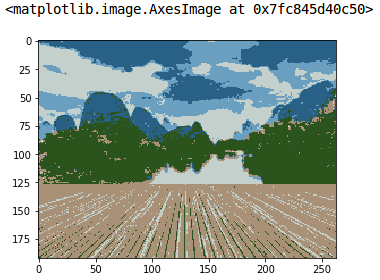
|  |  |
| --- | --- |
|  | from sklearn.cluster import KMeans |
|  | kmeans = KMeans(n\_clusters=5, random\_state=0).fit(pic\_n) |
|  | pic2show = kmeans.cluster\_centers\_[kmeans.labels\_] |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/Kmeans.py)[**Kmeans.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-kmeans-py) hosted with  by [**GitHub**](https://github.com/)

I have chosen 5 clusters for this article but you can play around with this number and check the results. Now, let’s bring back the clusters to their original shape, i.e. 3-dimensional image, and plot the results.

|  |  |
| --- | --- |
|  | cluster\_pic = pic2show.reshape(pic.shape[0], pic.shape[1], pic.shape[2]) |
|  | plt.imshow(cluster\_pic) |

[**view raw**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3/raw/464dd408528ba094fa672e2cfd4a45e22e7d8840/clusters.py)[**clusters.py**](https://gist.github.com/PulkitS01/b92a41d29a6c496c87919479bffc33c3#file-clusters-py) hosted with  by [**GitHub**](https://github.com/)



Amazing, isn’t it? We are able to segment the image pretty well using just 5 clusters. I’m sure you’ll be able to improve the segmentation by increasing the number of clusters.

k-means works really well when we have a small dataset. It can segment the objects in the image and give impressive results. But the algorithm hits a roadblock when applied on a large dataset (more number of images).

It looks at all the samples at every iteration, so the time taken is too high. Hence, it’s also too expensive to implement. And since k-means is a distance-based algorithm, it is only applicable to convex datasets and is not suitable for clustering [non-convex clusters](https://en.wikipedia.org/wiki/Convex_set).

Finally, let’s look at a simple, flexible and general approach for image segmentation.

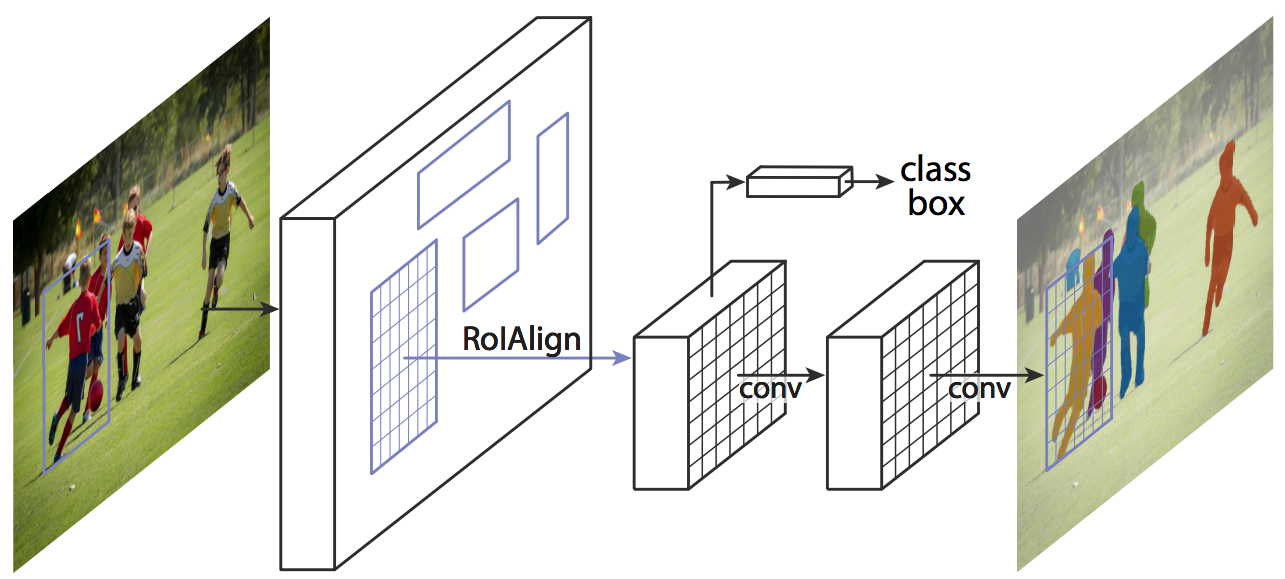
**Mask R-CNN**

Data scientists and researchers at Facebook AI Research (FAIR) pioneered a deep learning architecture, called Mask R-CNN, that can create a pixel-wise mask for each object in an image. This is a really cool concept so follow along closely!

Mask R-CNN is an extension of the popular [Faster R-CNN](https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/?utm_source=blog&utm_medium=image-segmentation-article) object detection architecture. Mask R-CNN adds a branch to the already existing Faster R-CNN outputs. The Faster R-CNN method generates two things for each object in the image:

* Its class
* The bounding box coordinates

Mask R-CNN adds a third branch to this which outputs the object mask as well. Take a look at the below image to get an intuition of how Mask R-CNN works on the inside:

*Source: arxiv.org*

1. We take an image as input and pass it to the ConvNet, which returns the feature map for that image
2. Region proposal network (RPN) is applied on these feature maps. This returns the object proposals along with their objectness score
3. A RoI pooling layer is applied on these proposals to bring down all the proposals to the same size
4. Finally, the proposals are passed to a fully connected layer to classify and output the bounding boxes for objects. It also returns the mask for each proposal

*Mask R-CNN is the current state-of-the-art for image segmentation and runs at 5 fps.*

**IMPLEMENTATION**

In this paper, a system in which density of traffic is measured by comparing captured image with real time traffic information against the image of the empty road as reference image is proposed. Here, in figure 1, the block diagram for proposed traffic control technique is illustrated.

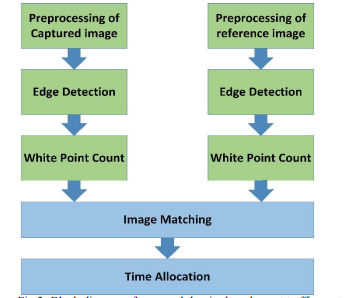


Fig : Block diagram of proposed density based smart traffic control system.

Each lane will have a minimum amount of green signal duration allocated. According to the percentage of matching allocated traffic light duration can be controlled. The matching is achieved by comparing the number of white points between two images. The entire image processing before edge detection i.e. image acquisition, image resizing, RGB to gray conversion and noise reduction is explained in section II. At section III, canny edge detection operation and white point count are depicted. Canny edge detector operator is selected because of its greater overall performance. Percentage matching for different sample images and traffic time allocation for them are demonstrated in section IV. The content of this paper completely serves the purpose of demonstrating the limitations of current traffic control techniques and the solution of this limitations with detailed explanation. Image matching by comparing detected edges is a novel approach to identify the vehicular density with propitious accuracy. As far as we know, matching images by comparing detected edges has not been used before for smart traffic control application.

**ALGORITHMS**

**CANNY EDGE DETECTION**

When it comes to image classification, the human eye has the incredible ability to process an image in a couple of milliseconds, and to determine what it is about (label). It is so amazing that it can do it whether it is a drawing or a picture.



Drawing of a car (Left) — Real car picture (Right): The human eye is able to classify both.

The idea today is to build an algorithm that can sketch the edges of any object present on a picture, using the Canny edge detection algorithm.

First of all, let’s describe what is the Canny Edge Detector:

*The****Canny edge detector****is an*[*edge detection*](https://en.wikipedia.org/wiki/Edge_detection)*operator that uses a multi-stage*[*algorithm*](https://en.wikipedia.org/wiki/Algorithm)*to detect a wide range of edges in images. It was developed by*[*John F. Canny*](https://en.wikipedia.org/wiki/John_F._Canny)*in 1986. Canny also produced a*computational theory of edge detection*explaining why the technique works. (Wikipedia)*

The Canny edge detection algorithm is composed of 5 steps:

1. Noise reduction;
2. Gradient calculation;
3. Non-maximum suppression;
4. Double threshold;
5. Edge Tracking by Hysteresis.

After applying these steps, you will be able to get the following result:



Original image on the left — Processed image on the right

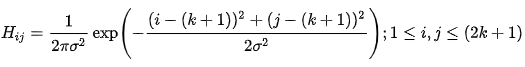
One last important thing to mention, is that the algorithm is based on grayscale pictures. Therefore, the pre-requisite is to convert the image to grayscale before following the above-mentioned steps.

# Noise Reduction

Since the mathematics involved behind the scene are mainly based on derivatives (cf. Step 2: Gradient calculation), edge detection results are highly sensitive to image noise.

One way to get rid of the noise on the image, is by applying Gaussian blur to smooth it. To do so, image convolution technique is applied with a Gaussian Kernel (3x3, 5x5, 7x7 etc…). The kernel size depends on the expected blurring effect. Basically, the smallest the kernel, the less visible is the blur. In our example, we will use a 5 by 5 Gaussian kernel.

The equation for a Gaussian filter kernel of size (2k+1)×(2k+1) is given by:



Gaussian filter kernel equation

Python code to generate the Gaussian 5x5 kernel:

Gaussian Kernel function

After applying the Gaussian blur, we get the following result:



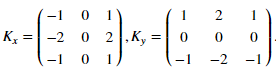
Original image (left) — Blurred image with a Gaussian filter (sigma=1.4 and kernel size of 5x5)

# Gradient Calculation

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators.

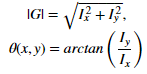
Edges correspond to a change of pixels’ intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y)

When the image is smoothed, the derivatives **Ix** and **Iy** w.r.t. **x** and **y** are calculated. It can be implemented by convolving **I** with Sobel kernels**Kx** and **Ky**, respectively:



Sobel filters for both direction (horizontal and vertical)

Then, the magnitude **G** and the slope **θ** of the gradient are calculated as follow:



Gradient intensity and Edge direction

Below is how the Sobel filters are applied to the image, and how to get both intensity and edge direction matrices:



Blurred image (left) — Gradient intensity (right)

The result is almost the expected one, but we can see that some of the edges are thick and others are thin. Non-Max Suppression step will help us mitigate the thick ones.

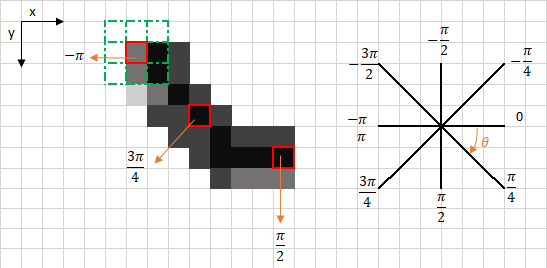
Moreover, the gradient intensity level is between 0 and 255 which is not uniform. The edges on the final result should have the same intensity (i-e. white pixel = 255).

# Non-Maximum Suppression

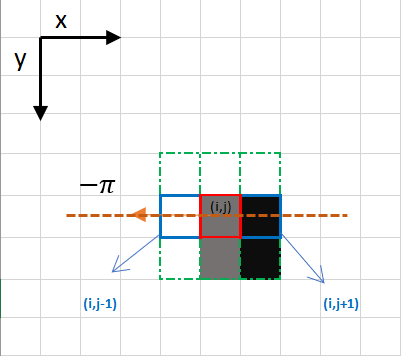
Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges.

The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.

Let’s take an easy example:



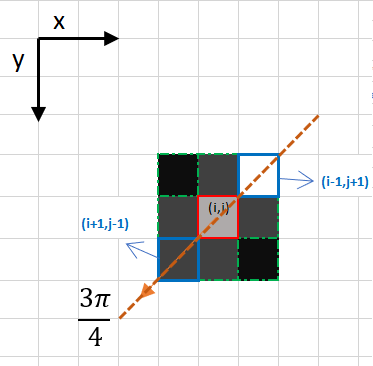
The upper left corner red box present on the above image, represents an intensity pixel of the Gradient Intensity matrix being processed. The corresponding edge direction is represented by the orange arrow with an angle of -pi radians (+/-180 degrees).



Focus on the upper left corner red box pixel

The edge direction is the orange dotted line (horizontal from left to right). The purpose of the algorithm is to check if the pixels on the same direction are more or less intense than the ones being processed. In the example above, the pixel **(i, j)**is being processed, and the pixels on the same direction are highlighted in blue **(i, j-1)** and **(i, j+1).**If one those two pixels are more intense than the one being processed, then only the more intense one is kept. Pixel **(i, j-1)**seems to be more intense, because it is white (value of 255). Hence, the intensity value of the current pixel **(i, j)** is set to 0. If there are no pixels in the edge direction having more intense values, then the value of the current pixel is kept.

Let’s now focus on another example:

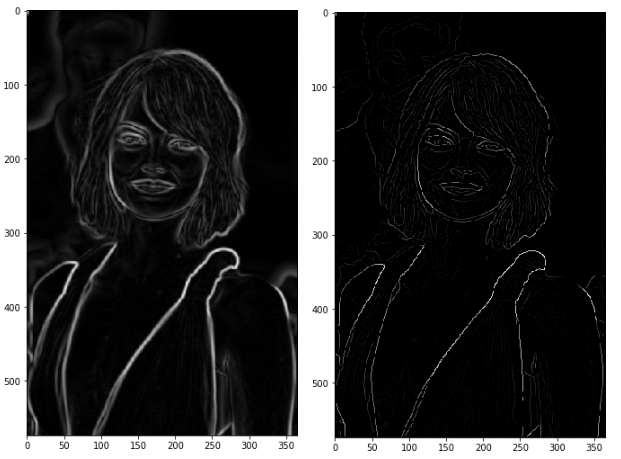


In this case the direction is the orange dotted diagonal line. Therefore, the most intense pixel in this direction is the pixel **(i-1, j+1).**

Let’s sum this up. Each pixel has 2 main criteria (edge direction in radians, and pixel intensity (between 0–255)). Based on these inputs the non-max-suppression steps are:

* Create a matrix initialized to 0 of the same size of the original gradient intensity matrix;
* Identify the edge direction based on the angle value from the angle matrix;
* Check if the pixel in the same direction has a higher intensity than the pixel that is currently processed;
* Return the image processed with the non-max suppression algorithm.

The result is the same image with thinner edges. We can however still notice some variation regarding the edges’ intensity: some pixels seem to be brighter than others, and we will try to cover this shortcoming with the two final steps.



Result of the non-max suppression.

# Double threshold

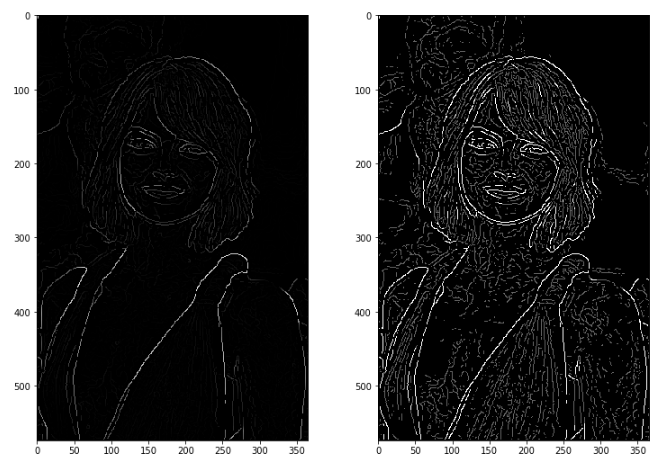
The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant:

* Strong pixels are pixels that have an intensity so high that we are sure they contribute to the final edge.
* Weak pixels are pixels that have an intensity value that is not enough to be considered as strong ones, but yet not small enough to be considered as non-relevant for the edge detection.
* Other pixels are considered as non-relevant for the edge.

Now you can see what the double thresholds holds for:

* High threshold is used to identify the strong pixels (intensity higher than the high threshold)
* Low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold)
* All pixels having intensity between both thresholds are flagged as weak and the Hysteresis mechanism (next step) will help us identify the ones that could be considered as strong and the ones that are considered as non-relevant.

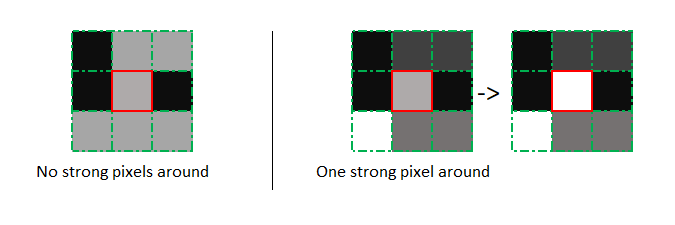
The result of this step is an image with only 2 pixel intensity values (strong and weak):



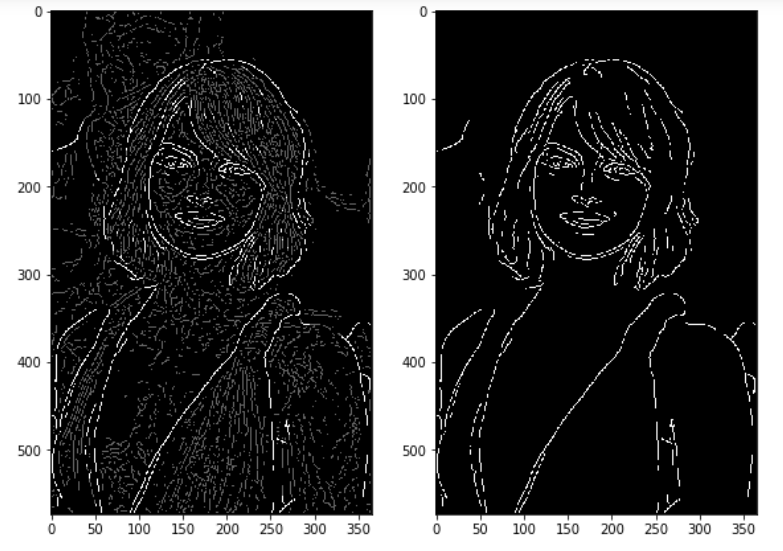
Non-Max Suppression image (left) — Threshold result (right): weak pixels in gray and strong ones in white.

# Edge Tracking by Hysteresis

Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one, as described below:



Hysteresis function

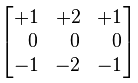
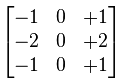


Results of hysteresis process

Sobel Edge Detection

Sobel edge detector is a gradient based method based on the first order derivatives. It calculates the first derivatives of the image separately for the X and Y axes.

The operator uses two 3X3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. The picture below shows Sobel Kernels in x-dir and y-dir:

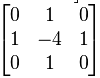


For more details on Sobel operation, please check [Sobel operator](http://en.wikipedia.org/wiki/Sobel_operator).

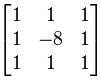
Laplacian Edge Detection

Unlike the Sobel edge detector, the Laplacian edge detector uses only one kernel. It calculates second order derivatives in a single pass.

A kernel used in this Laplacian detection looks like this:



If we want to consider the diagonals, we can use the kernel below:



cv2.Laplacian(src, ddepth, other\_options...)

where **ddepth** is the desired depth of the destination image.

Code for Edge Detection

Here is a code that can do edge detection:

import cv2

import numpy as np

from matplotlib import pyplot as plt

# loading image

#img0 = cv2.imread('SanFrancisco.jpg',)

img0 = cv2.imread('windows.jpg',)

# converting to gray scale

gray = cv2.cvtColor(img0, cv2.COLOR\_BGR2GRAY)

# remove noise

img = cv2.GaussianBlur(gray,(3,3),0)

# convolute with proper kernels

laplacian = cv2.Laplacian(img,cv2.CV\_64F)

sobelx = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5) # x

sobely = cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5) # y

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')

plt.title('Laplacian'), plt.xticks([]), plt.yticks([])

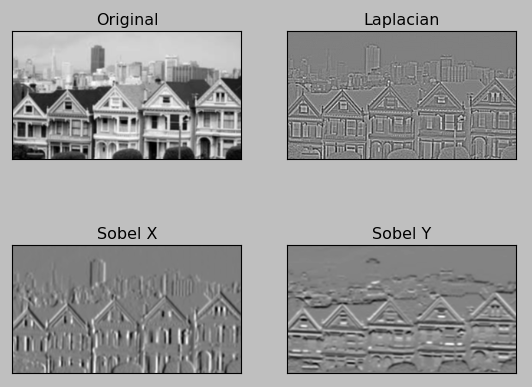
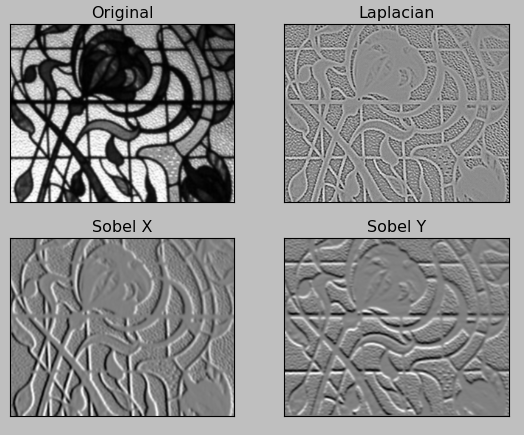
plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')

plt.title('Sobel X'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')

plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

plt.show()

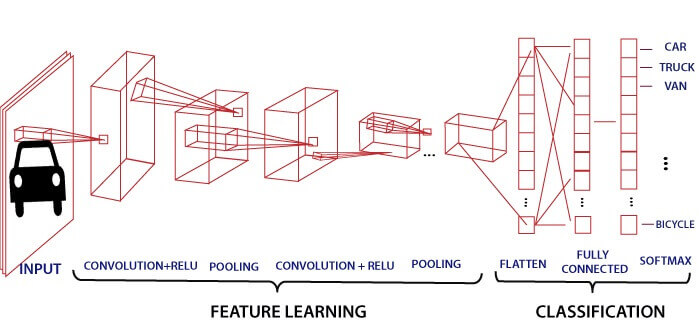
  
  
  


# Images used: CONVOLUTIONAL NEURAL NETWORK

**Convolutional Neural Network** is one of the main categories to do image classification and image recognition in neural networks. Scene labeling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.

CNN takes an image as input, which is classified and process under a certain category such as dog, cat, lion, tiger, etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution, it will see as **h \* w \* d**, where h= height w= width and d= dimension. For example, An RGB image is **6 \* 6 \* 3** array of the matrix, and the grayscale image is **4 \* 4 \* 1** array of the matrix.

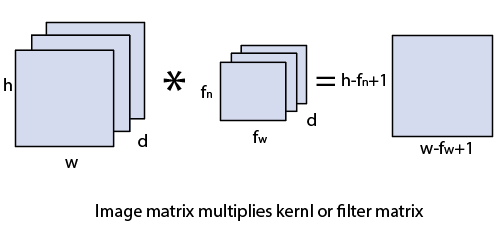
In CNN, each input image will pass through a sequence of convolution layers along with pooling, fully connected layers, filters (Also known as kernels). After that, we will apply the Soft-max function to classify an object with probabilistic values 0 and 1.



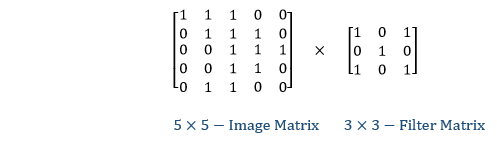
## Convolution Layer

Convolution layer is the first layer to extract features from an input image. By learning image features using a small square of input data, the convolutional layer preserves the relationship between pixels. It is a mathematical operation which takes two inputs such as image matrix and a kernel or filter.

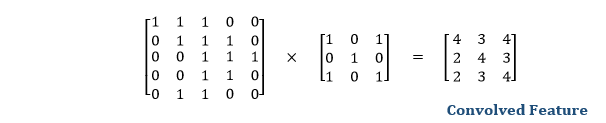
* The dimension of the image matrix is **h×w×d**.
* The dimension of the filter is **fh×fw×d**.
* The dimension of the output is **(h-fh+1)×(w-fw+1)×1**.



Let's start with consideration a 5\*5 image whose pixel values are 0, 1, and filter matrix 3\*3 as:



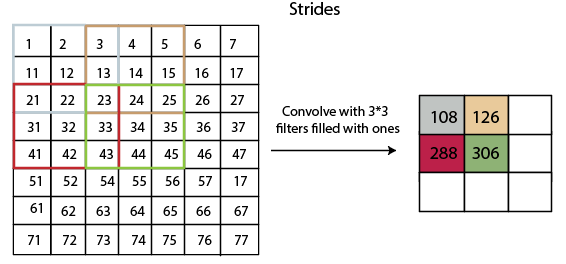
The convolution of 5\*5 image matrix multiplies with 3\*3 filter matrix is called "**Features Map**" and show as an output.



Convolution of an image with different filters can perform an operation such as blur, sharpen, and edge detection by applying filters.

## Strides

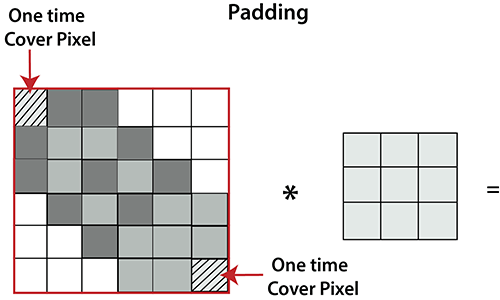
Stride is the number of pixels which are shift over the input matrix. When the stride is equaled to 1, then we move the filters to 1 pixel at a time and similarly, if the stride is equaled to 2, then we move the filters to 2 pixels at a time. The following figure shows that the convolution would work with a stride of 2.



## Padding

Padding plays a crucial role in building the convolutional neural network. If the image will get shrink and if we will take a neural network with 100's of layers on it, it will give us a small image after filtered in the end.

If we take a three by three filter on top of a grayscale image and do the convolving then what will happen?



It is clear from the above picture that the pixel in the corner will only get covers one time, but the middle pixel will get covered more than once. It means that we have more information on that middle pixel, so there are two downsides:

* Shrinking outputs
* Losing information on the corner of the image.

To overcome this, we have introduced padding to an image. **"Padding is an additional layer which can add to the border of an image."**

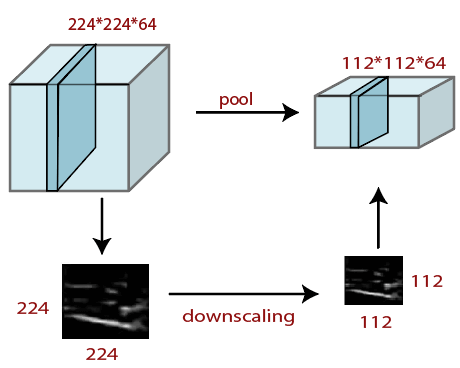
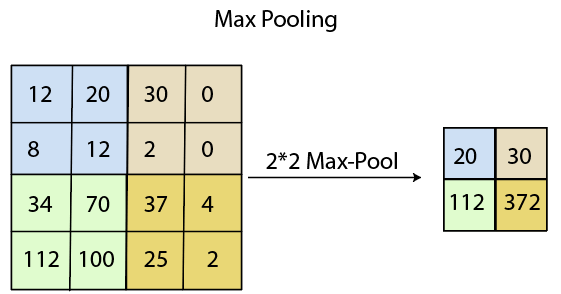
## Pooling Layer

Pooling layer plays an important role in pre-processing of an image. Pooling layer reduces the number of parameters when the images are too large. Pooling is "**downscaling**" of the image obtained from the previous layers. It can be compared to shrinking an image to reduce its pixel density. Spatial pooling is also called downsampling or subsampling, which reduces the dimensionality of each map but retains the important information. There are the following types of spatial pooling:

### Max Pooling

Max pooling is a **sample-based discretization process**. Its main objective is to downscale an input representation, reducing its dimensionality and allowing for the assumption to be made about features contained in the sub-region binned.

Max pooling is done by applying a max filter to non-overlapping sub-regions of the initial representation.



### Average Pooling

Down-scaling will perform through average pooling by dividing the input into rectangular pooling regions and computing the average values of each region.

**Syntax**

layer = averagePooling2dLayer(poolSize)

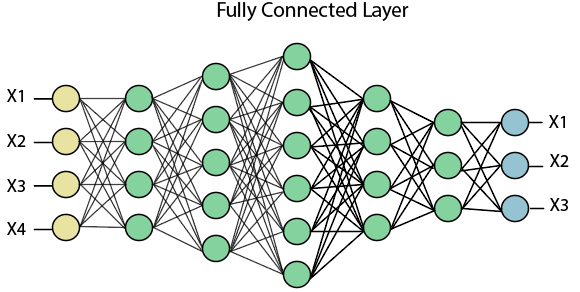
layer = averagePooling2dLayer(poolSize,Name,Value)

### Sum Pooling

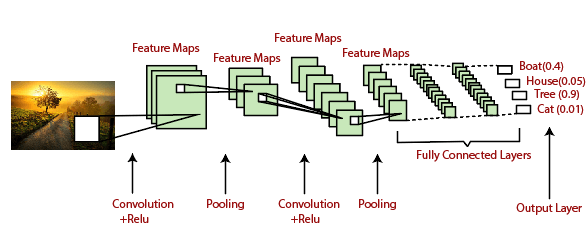
The sub-region for **sum pooling** or **mean pooling** are set exactly the same as for **max-pooling** but instead of using the max function we use sum or mean.

## Fully Connected Layer

The fully connected layer is a layer in which the input from the other layers will be flattened into a vector and sent. It will transform the output into the desired number of classes by the network.



In the above diagram, the feature map matrix will be converted into the vector such as **x1, x2, x3... xn** with the help of fully connected layers. We will combine features to create a model and apply the activation function such as **softmax** or **sigmoid** to classify the outputs as a car, dog, truck, etc.



**CONCLUSION**

In this paper, a smart traffic control system availing image processing as an instrument for measuring the density has been proposed. Besides explaining the limitations of current near obsolete traffic control system, the advantages of proposed traffic control system have been demonstrated. For this purpose, four sample images of different traffic scenario have been attained. Upon completion of edge detection, the similarity between sample images with the reference image has been calculated. Using this similarity, time allocation has been carried out for each individual image in accordance with the time allocation algorithm. In addition, similarity in percentage and time allocation have been illustrated for each of the four sample images using Python programming language. Besides presenting the schematics for the proposed smart traffic control system, all the necessary results have been verified by hardware implementation

REFERENCES [1] Siddique, A. (2017). WB: Dhaka’s average traffic speed 7kmph. [online] Dhaka Tribune. Available at: http://www.dhakatribune.com/bangladesh/dhaka/2017/07/19/dhakaaverage-traffic-speed-7-kmph/ [Accessed 25 Jul. 2017]. [2] M. Sweet, “Traffic Congestion’s Economic Impacts: Evidence from US Metropolitan Regions,” Urban Studies, vol. 51, no. 10, pp. 2088–2110, Oct. 2013. [3] Md. Munir Hasan, Gobinda Saha, Aminul Hoque and Md. Badruddoja Majumder, “Smart Traffic Control System with Application of Image Processing Techniques,”in 3rd International Conference on Informatics, Electronics & Vision, Dhaka, May 2014. [4] Vismay Pandit, Jinesh Doshi, Dhruv Mehta, Ashay Mhatre and Abhilash Janardhan, “Smart Traffic Control System Using Image Processing,”International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Vol. 3, Issue 1, January – February 2014. [5] Pallavi Choudekar, Sayanti Banerjee and M. K. Muju, “Implementation of image processing in real time traffic light control,” in3rd International Conference on Electronics Computer Technology, April, 2011. [6] T. Semertzidis, K. Dimitropoulos, A. Koutsia and N. Grammalidis, “Video sensor network for real-time traffic monitoring and surveillance,”The Institution of Engineering and Technology, Volume 4, Issue 2, June 2010, pp. 103 – 112. [7] Lei Yang, Dewei Zhao and Xiaoyu Wu, “An improved Prewitt algorithm for edge detection based on noised image,”in 4th International Congress on Image and Signal Processing, Shanghai, China, October 2011. [8] Weibin Rong, Zhanjing Li and Wei Zhang, “An improved canny edge detection algorithm,”in IEEE International Conference on Mechatronics and Automation, August 2014. [9] Wenshuo Gao, Xiaoguang Zhang and Lei Yang, “An improved Sobel edge detection,” in3rd International Conference on Computer Science and Information Technology, July 2010. [10] G.T. Shrivakshan and Dr.C. Chandrasekar, “A Comparison of various Edge Detection Techniques used in Image Processing,”International Journal of Computer Science Issues, Vol. 9, Issue 5, No 1, September 2012.