As per UGC guidelines an electronic bar code is provided to seure your paper

International Journal for Modern Trends in Science and Technology, 9(05): 269-274, 2023
Copyright © 2023International Journal for Modern Trends in Science and Technology
ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0905044
Available online at: http://www.ijmtst.com/vol9issue05.html

Check for updates

Traffic Light Optimization using OpenCV

Isha Nimje, Mrunal Wankhede, Nandini shahare, Nupur Khadse, Roshni Shahu

Department of Information Technology, Rajiv Gandhi College of Engineering and Research, Nagpur, Maharashtra

To Cite this Article
Isha Nimje, Mrunal Wankhede, Nandini shahare, Nupur Khadse, Roshni Shahu. Traffic Light Optimization using OpenCV. International Journal for Modern Trends in Science and Technology 2023, 9(05), pp. 269-274. https://doi.org/10.46501/IJMTST0905044

Article Info
Received: 06 April 2023; Accepted: 02 May 2023; Published: 14 May 2023.

ABSTRACT

To provide an optimal solution for traffic congestion due to high waiting time at signals sometimes an empty road is given green light allowing the other lane vehicles to wait for the signal to go green. The project aims at minimizing the waiting time on a particular lane by deciding the pattern and time in which a lane is given green light. OpenCV is a technology used to give vision to the computer which allows the computer to detect objects by its own. After detecting the objects, the unnecessary objects areignored and the necessary ones are considered.

The project is designed in such a way that a particular lane with a maximum number of the vehicle is given less waiting time. The number of vehicles is detected by OpenCV and after counting the number of vehicles a certain amount of time is decided for which green light is to be given. The time wasted in waiting at a signal is minimized. Training the module in such a way that equal chances should be givento each lane and after completion of a loop, the vehicle is given chance again. In such a way our projectis developed.

1. INTRODUCTION

One important application of computer vision is traffic monitoring and control. Here we are presenting asystem for detection of moving vehicles approaching an intersection or in an highway by camera in the context of traffic light control systems. As the system is dedicated to outdoor applications, efficient and robust vehicle detection under various weather and illumination conditions is examined. To deal with these ever-changing conditions, vehicle detection relies on motion segmentation and color mapping to achieve feature space segmentation. Experimental results using real outdoor sequences of images demonstrate the system's robustness under various environmental conditions. It detects the number of vehicles on each road and depending on the vehicles load on each road, this system assigns optimized amount of waiting time (red signal light) and running time (green signal light). This system is a fully automated system that can replace the conventional pre- determined fixed-time based traffic system with a dynamically managed traffic system. It can also detect vehicle condition on road and auto-adjust the system according to the changing road conditions which makes the system intelligent. The designed system can help solving traffic problems in busy cities to a great extent by saving a significant number of man-hours that get lost waiting on jammed roads. This research focuses on factors, low-cost image processing and traffic load balancing. Moreover, we are also replacing the conventional traffic light systemwith a more efficient system using LCD projector. This computer vision technology can be used to reducethe traffic congestion and also helps to detect people who are not wearing helmets to a extent We made 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. The image sequence will then be analyzed using digital image processing for vehicle detection, and according to traffic conditions on the road trafficlight can be controlled.

LITERATURE SURVEY

Traffic signals are essential to guarantee safe driving at road intersections. However, they disturb and reduce the traffic fluency due to the queue delay at each traffic flow. In this work, we introduce an Intelligent Traffic Light Controlling (ITLC) algorithm. This algorithm considers the real-timetraffic characteristics of each traffic flow that intends to cross the road intersection of interest, whilst schedulingthe time phases of each traffic light. The introduced algorithm aims at increasing the traffic fluency by decreasing the waiting time of travelling vehicles at the signalized road intersections. Moreover, it aims to increase the number of vehicles crossing the road intersection per second. In modern life we have to face with many problems one of which is traffic congestion becoming more serious day after day. Traffic flow determination can play a principle role in gathering information about them.

This data is used to establish censorious flow time periodsuch as the effect of large vehicle, specific part on vehicular traffic flow and providing a factual record of traffic volume trends. This recorded information also useful for process the better traffic in terms of periodic time of traffic lights. There are many routes to count the number of vehicles passed in a particular time, and can give judgment of traffic flow. Now aday's camera-based systems are better choices for tracing the vehicles data. This project focuses on a firmware-based novel technique for vehicle detection. Thisapproach detects the vehicles in the source image, and applies an existing identifier for each of the vehicle. Later it classifies each vehicle on its vehicle-type group and counts them all by individually. The developed approach was implemented in a firmware platform which results is better accuracy, high reliability and less errors. Traffic lights play a very significant role in traffic control and regulation on a daily basis.

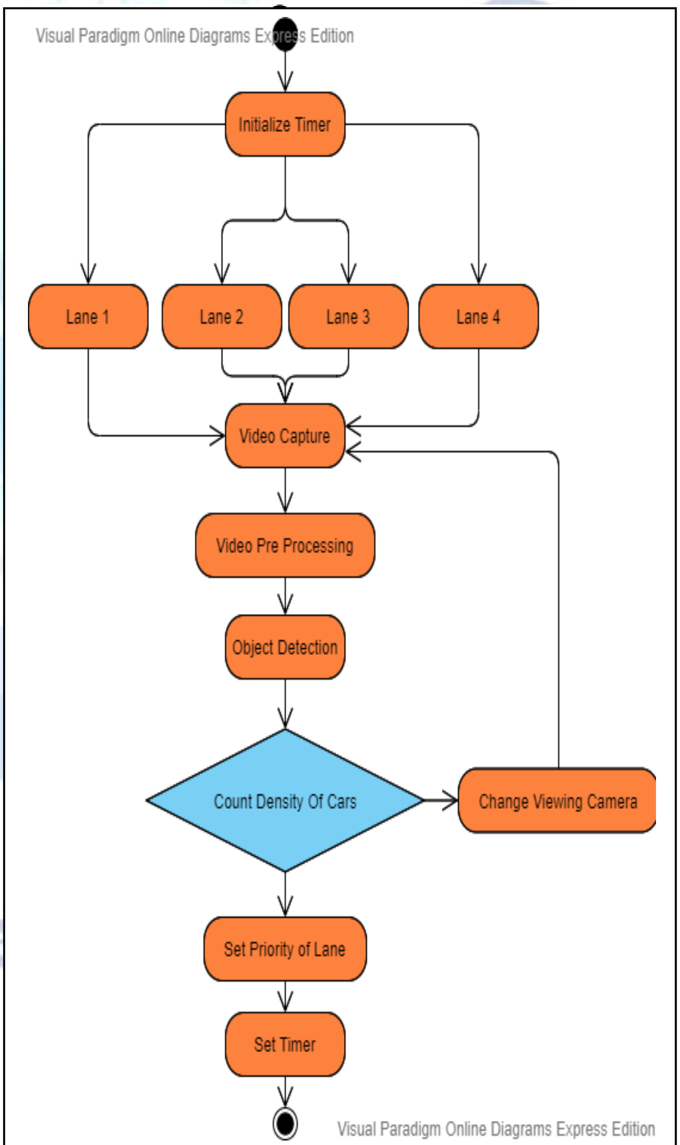


Fig 2. Implemantation of flow chart

DESIGN / IMPLEMENTATION / MODELING

For Implementation of the project first we need to set the total timer of the traffic light so that in what interval can all lanes light take for one complete cycle. Then we need to capture the video as input from each lane so that we could identify the total number of vehicles in each lane. After we get the lane-wise videos then we need to identify the cars which are done by using the YOLO Object detection Module. And after the cars are detected, then we need to count the number of vehicles and store them. After we get lane wise Vehicle count we need to set the priority to the lane with the maximum number of vehiclesand also we have to manipulate the green signal timing depending upon the number of vehicles For Example:- The lane with the maximum number of the vehicle is given first priority and also the timer is set as such that the lane gets maximum time from the total timer, so that maximum number of vehicles can go through the signal. This all is done by using our Algorithm. This is a continuous process in whichthe video capture will be taken after every lane is given a green signal so that the Real-time decisions can be made.

METHODOLOGY

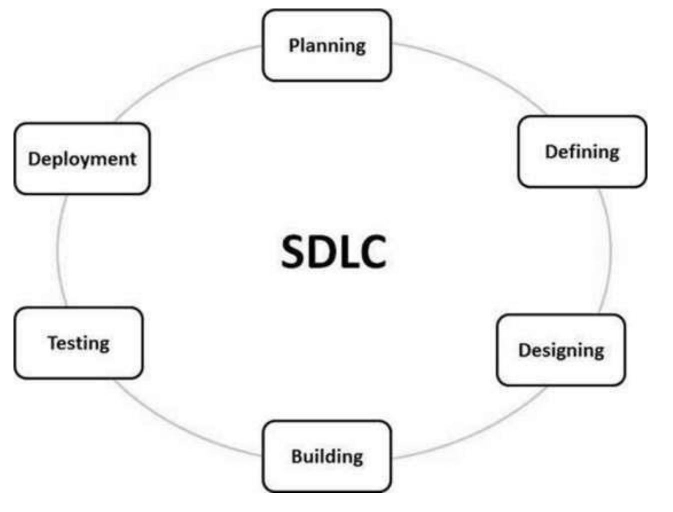


Fig -3: Workflow

The Interface is going to be a real-time application. The project is based on a live collection of data and then processed through a particular algorithm by which it is trained. The first step in making a project isdefining the project into modules which later can be combined to work accordingly.

Module

Machine learning projects are highly iterative; as we progress through both ML lifecycles supervised learning and unsupervised, we will do the iterating on a section until reaching a satisfactory level of performance, then proceeding forward to the next task (which may be circling back to an even earlier step). Moreover, the project isn't complete after we ship the first version; we get feedback from real- world interactions and redefine the goals for the nextiteration of deployment.

Project Planning
Collection of Data
Designing model Implementation of models
Testing
Deployment
Maintenance

OpenCV

OpenCV (Open-Source Computer Vision Library) isan open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD- licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

OpenCV is used to give vision to a particular system. It is used to detect objects in front of the computer vision system to easily recognize the vehicles waiting in the signal or coming in sight of the compute

YOLO

You only look once (YOLO) is a state-of-the-art, real-time object detection System YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. The object detection task consists in determining the location on the image where certain objects are present, as well as classifying those objects.

Previous methods for this, like R- CNN and its variations, used a pipeline to perform this task in multiple steps. This can be slow to run and also hard to optimize because each individual component must be trained separately. YOLO does it all with a single neural network. The YOLO library can detectmany objects and classify them according to the algorithm it uses. To classify only the vehicles the YOLO library is used.

Turtle

Turtle graphics is a popular way of introducing programming to kids. It was part of the original Logo programming language developed by Wally Feurzeig, Seymour Papert and Cynthia Solomon in 1967. Imagine a robotic turtle starting at (0, 0) in the x-y plane. After an import turtle, give it the command turtle.forward(15), and it moves (on-screen!) 15 pixels in the direction it is facing, drawing a line as it moves. Give it the command turtle.right(25), and it rotates in-place 25 degrees clockwise.

Turtle star

Turtle can draw intricate shapes using programs that repeatsimple moves.

```
../_images/turtle-star.pngfrom turtle import * color('red', 'yellow') begin_fill()
while True:
forward(200)left(170)
if abs(pos())<1:break
end_fill()done()
```

By combining together these and similar commands, intricateshapes and pictures can easily be drawn

Data Collection / Tools / Platform used

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. Data collection is a component of research in all fields of study including physical and social sciences, humanities, and business. While methods vary by discipline, the emphasis on ensuring accurate and honest collection remains the same. The goal for all data collection is to capture quality evidence that allows analysis to lead to the formulation of convincing and credible answers to the questions that have been posed. The data is collected from live video feeds from the CCTV cameras which would be fixed in each square. Hence, we will be requiring live video feeds

The video is fed into the model and then our own algorithms helps us to find the priority of lane with greensignal timer

The feasibility is first checked by implementing the model in ATOM. The results are first checked for any errors and then were implemented into the project.

The multiple technologies are used to create the complete project:

Python Technology Stack used to develop the platform

1. OpenCV
2. YOLO
3. Pytorch
4. Turtle

OpenCV- Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays

YOLO (You Only Look Once) is a method / way to do object detection. It is the algorithm /strategy behindhow the code is going to detect objects in the image

YOLO takes entirely different approach. It looks at the entire image only once and goes through the networkonce and detects objects. Hence the name. It is very fast. That's the reason it has got so popular

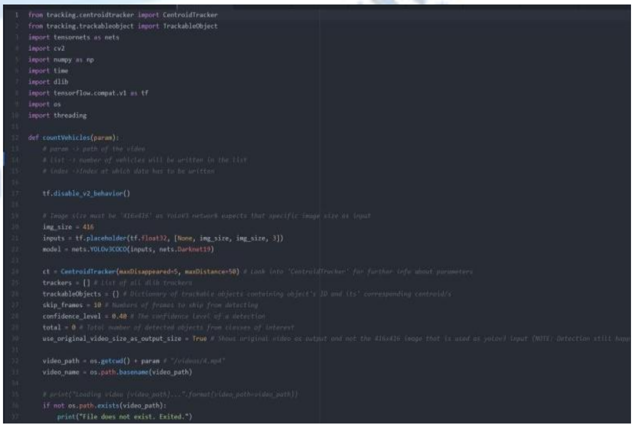
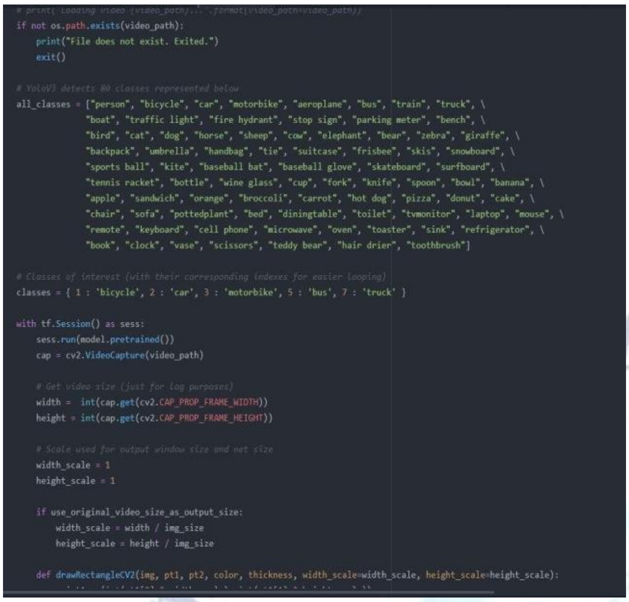


Fig 1.Object Detection Using OpenCV/YOLO



```python
# print("Loading video {video_path}...".format(video_path=video_path))
if not os.path.exists(video_path):
    print("File does not exist. Exited.")
    exit()

# YoloV3 detects 80 classes represented below
all_classes = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck", \
            "boat", "traffic light", "fire hydrant", "stop sign", "parking meter", "bench", \
            "bird", "cat", "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe", \
            "backpack", "umbrella", "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard", \
            "sports ball", "kite", "baseball bat", "baseball glove", "skateboard", "surfboard", \
            "tennis racket", "bottle", "wine glass", "cup", "fork", "knife", "spoon", "bowl", "banana", \
            "apple", "sandwich", "orange", "broccoli", "carrot", "hot dog", "pizza", "donut", "cake", \
            "chair", "sofa", "pottedplant", "bed", "diningtable", "toilet", "tvmonitor", "laptop", "mouse", \
            "remote", "keyboard", "cell phone", "microwave", "oven", "toaster", "sink", "refrigerator", \
            "book", "clock", "vase", "scissors", "teddy bear", "hair drier", "toothbrush"]

# Classes of interest (with their corresponding indexes for easier looping)
classes = { 1 : 'bicycle', 2 : 'car', 3 : 'motorbike', 5 : 'bus', 7 : 'truck' }

with tf.Session() as sess:
    sess.run(model.pretrained())
    cap = cv2.VideoCapture(video_path)

    # Get video size (just for log purposes)
    width =  int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    # Scale used for output window size and net size
    width_scale = 1
    height_scale = 1

    if use_original_video_size_as_output_size:
        width_scale = width / img_size
        height_scale = height / img_size

    def drawRectangleCV2(img, pt1, pt2, color, thickness, width_scale=width_scale, height_scale=height_scale):
```

Fig 2.Object Detection Using OpenCV/YOLO

```python
    # Get video size (just for log purposes)
    width =  int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    # Scale used for output window size and net size
    width_scale = 1
    height_scale = 1

    if use_original_video_size_as_output_size:
        width_scale = width / img_size
        height_scale = height / img_size

    def drawRectangleCV2(img, pt1, pt2, color, thickness, width_scale=width_scale, height_scale=height_scale):
        point1 = (int(pt1[0] * width_scale), int(pt1[1] * height_scale))
        point2 = (int(pt2[0] * width_scale), int(pt2[1] * height_scale))
        return cv2.rectangle(img, point1, point2, color, thickness)

    def drawTextCV2(img, text, pt, font, font_scale, color, lineType, width_scale=width_scale, height_scale=height_scale):
        pt = (int(pt[0] * width_scale), int(pt[1] * height_scale))
        cv2.putText(img, text, pt, font, font_scale, color, lineType)

    def drawCircleCV2(img, center, radius, color, thickness, width_scale=width_scale, height_scale=height_scale):
        center = (int(center[0] * width_scale), int(center[1] * height_scale))
        cv2.circle(img, center, radius, color, thickness)

    # Python 3.5.6 does not support f-strings (next line will generate syntax error)
    #print(f"Loaded {video_path}. Width: {width}, Height: {height}")
    # print("Loaded {video_path}. Width: {width}, Height: {height}".format(video_path=video_path, width=width, height=height))

    skipped_frames_counter = 0

    while(cap.isOpened()):
        try :
            ret, frame = cap.read()
            img = cv2.resize(frame, (img_size, img_size))
```

Fig 3.Object Detection Using OpenCV/YOLO

```python
    # print("Loaded {video_path}. Width: {width}, Height: {height}".format(video_path=video_path, width=width, heigh

    skipped_frames_counter = 0

    while(cap.isOpened()):
        try :
            ret, frame = cap.read()
            img = cv2.resize(frame, (img_size, img_size))
        except:
            print(total_str)

        output_img = frame if use_original_video_size_as_output_size else img

        tracker_rects = []

        if skipped_frames_counter == skip_frames:

            # Detecting happens after number of frames have passes specified by 'skip_frames' variable value
            # print("[DETECTING]")

            trackers = []
            skipped_frames_counter = 0 # reset counter

            np_img = np.array(img).reshape(-1, img_size, img_size, 3)

            start_time=time.time()
            predictions = sess.run(model.preds, {inputs: model.preprocess(np_img)})
            # print("Detection took %s seconds" % (time.time() - start_time))

            # model.get_boxes returns a 80 element array containing information about detected classes
            # each element contains a list of detected boxes, confidence level ...
            detections = model.get_boxes(predictions, np_img.shape[1:3])
            np_detections = np.array(detections)

            # Loop only through classes we are interested in
            for class_index in classes.keys():
```

Fig 4.Object Detection Using OpenCV/YOLO

```python
                class_name = classes[class_index]

                # Loop through detected infos of a class we are interested in
                for i in range(len(np_detections[class_index])):
                    box = np_detections[class_index][i]

                    if np_detections[class_index][i][4] >= confidence_level:
                        # print("Detected ", class_name, " with confidence of ", np_detections[class_index][i][4])

                        local_count += 1
                        startX, startY, endX, endY = box[0], box[1], box[2], box[3]

                        drawRectangleCV2(output_img, (startX, startY), (endX, endY), (0, 255, 0), 1)
                        drawTextCV2(output_img, class_name, (startX, startY), cv2.FONT_HERSHEY_SIMPLEX, .5, (0, 0, 255), 1)

                        # Construct a dlib rectangle object from the bounding box coordinates and then start the dlib correlation
                        tracker = dlib.correlation_tracker()
                        rect = dlib.rectangle(int(startX), int(startY), int(endX), int(endY))
                        tracker.start_track(img, rect)

                        # Add the tracker to our list of trackers so we can utilize it during skip frames
                        trackers.append(tracker)

                # Write the total number of detected objects for a given class on this frame
                # print(class_name," : ", local_count)
        else:

            # If detection is not happening then track previously detected objects (if any)
            # print("[TRACKING]")

            skipped_frames_counter += 1 # Increase the number frames for which we did not use detection

            # Loop through tracker, update each of them and display their rectangle
            for tracker in trackers:
                tracker.update(img)
                pos = tracker.get_position()
```

Fig 5.Object Detection Using OpenCV/YOLO

```
            # Store the trackable object in our dictionary
            trackableObjects[objectID] = to

            # Draw both the ID of the object and the centroid of the object on the output frame
            object_id = "ID {}".format(objectID)
            drawTextCV2(output_img, object_id, (centroid[0] - 10, centroid[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)
            drawCircleCV2(output_img, (centroid[0], centroid[1]), 2, (0, 255, 0), -1)

            # Display the total count so far
            total_str = str(total)
            drawTextCV2(output_img, total_str, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

        # Display the current frame (with all annotations drawn up to this point)
        cv2.imshow(video_name, output_img)

        key = cv2.waitKey(1) & 0xFF
        if key  == ord('q'): # QUIT (exits)
            break
        elif key == ord('p'):
            cv2.waitKey(0) # PAUSE (Enter any key to continue)
    cap.release()
    cv2.destroyAllWindows()
    print("Exited")

    """
    function which will run our code

    will write the number of veicles in the list provided
    """

if _name_ == "_main_":

    countVehicles("/videos/1.1.mp4")

    # Logic for setting the time for each signal
```

Fig 6.Object Detection Using OpenCV/YOLO

Testing & Summary of Results

We have done the testing of our project by using the different signals video and also we have tested our project with manual input of the number of vehicles and lanes. we got the optimized Traffic light signal, Our project saved time as well as helped to avoid the traffic congestion on the signal

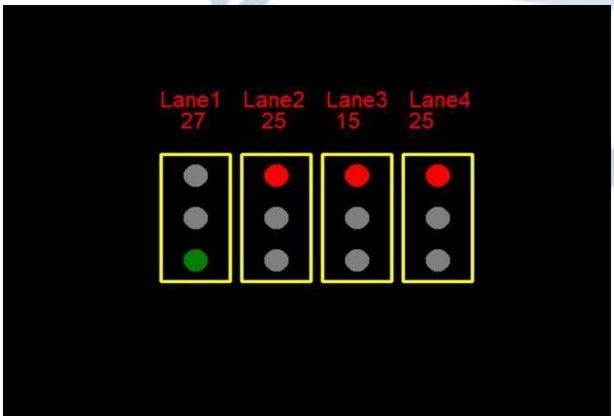


Fig 4.Result

CONCLUSION

By the study given, the Initialization of the research; it can be concluded that applying the required methodology, the required analytics in the field of OpenCV can be implemented.It does have a future scope and can be introduced to new features and algorithms. The research with algorithms provided here make sure that others can refer to this and also use this part to optimize the outcome in their research workand save time in determining the optimum algorithm.

Hence, we have successfully implemented our project with livevideo footage from traffic signal and alsogot the traffic lights optimized

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

References

[1] Mădălin-Dorin Pop, Politehnica University of Timişoara, Bd. Vasile Pârvan 2, Timişoara 300223, Romania Traffic Lights Management Using Optimization tool.

[2] K.T.K. Teo, W.Y. Kow and Y.K. Chin School of Engineering and Information Technology, Universiti Malaysia Sabah, Kota Kinabalu, Malaysia Optimization of Traffic Flow within an Urban Traffic Light Intersection with Genetic Algorithm

[3] L. Adacher, Roma Tre University, Via della Vasca Navale 79, Roma 00146, Italy A globaloptimization approach to solve the traffic signal synchronization problem.

[4] Kasun N. Hewage and Janaka Y. Ruwanpura Department of Civil Engineering University of Calgary 2500, University Drive, NW, Calgary, AB T2N1N4, CANADA OPTIMIZATION OF TRAFFIC SIGNAL LIGHT TIMING USING SIMULATION