AI-Powered Traffic Flow Optimization for Smart Cities

Shashwat Pratap Singh[1], Dr. Shikha Singh[2]
[1]*BTech in Computer Science and Engineering, Amity University, Lucknow, India*
[2]*Professor, Amity University, Lucknow, India*

***Abstract*—The rapid growth in urban populations and the corresponding surge in vehicle numbers have led to significant traffic congestion issues in modern cities. Traffic congestion not only led to delays and driver annoyance but also result in higher fuel usage and increased air pollution levels. While this problem affects cities worldwide, megacities are particularly vulnerable. Addressing this challenge necessitates real-time traffic density measurement to optimize signal control and enhance overall traffic management. The efficiency of traffic flow largely hinges on the performance of traffic controllers, making their optimization critical. This research introduces a system that leverages live imagery from traffic cameras to calculate vehicle density using advanced image processing and artificial intelligence techniques. The system employs algorithms to adjust traffic light timings based on vehicle density, thereby reducing congestion, facilitating quicker commutes, and lowering pollution levels.**

***Index Terms*—Traffic control, Traffic light system, Traffic management, Intelligent transport systems, Smart surveillance, Computer Vision, Machine Learning, Object detection, YOLO.**

I. INTRODUCTION

Urbanization has led to a significant increase in vehicle numbers, resulting in road networks experiencing reduced capacity and deteriorating service levels. A major contributor to traffic congestion is the reliance on fixed-timer-based traffic control systems at intersections, which follow repetitive phase sequences without accounting for real-time variations in traffic conditions. As the demand for road infrastructure grows, there is a pressing need to develop innovative traffic control solutions within the scope of Intelligent Transportation Systems (ITS).

To illustrate the severity of traffic issues, consider cities like Mumbai and Bangalore. Bangalore has been identified as the city with the worst traffic congestion globally, while Mumbai ranks fourth, as per a report covering 416 cities across 57 countries. In Bangalore, commuting during peak hours can extend travel time by 71%, while in Mumbai, it increases by 65% [1].

Currently, three primary traffic control methods are in use:

1. Manual Control: This method relies on traffic police officers managing the flow of vehicles using signboards, signal lights, and whistles. While effective in certain scenarios, it demands substantial manpower, which is often unavailable in many urban areas.
2. Conventional Static Timers: These systems operate on fixed timers, where the duration of green, yellow, and red lights is preset and does not adjust to real-time traffic conditions.
3. Electronic Sensors: Technologies like loop detectors and proximity sensors collect traffic data to adjust signals. However, these systems often struggle with accuracy, require sophisticated (and costly) technology, and have limited coverage due to budget constraints and sensor range limitations.

Each of these methods faces significant limitations. Manual control is labor-intensive and impractical for widespread use in large urban environments. Static timers fail to accommodate fluctuating traffic volumes, leading to inefficiencies. Sensor-based systems, though more advanced, are often cost-prohibitive and limited in scope due to their dependency on hardware installations.

In the last few years, video traffic surveillance systems have become increasingly used in traffic management for security, ramp metering, and disseminating real-time information to road users. These systems offer potential for traffic density estimation and vehicle classification, enabling adaptive control of traffic signals to optimize flow and reduce congestion. This research proposes a traffic light control system utilizing computer vision techniques, which adapts signal timings based on real-time traffic data captured

through CCTV cameras at intersections. By detecting vehicle density and adjusting light phases dynamically, the system aims to minimize congestion, reduce delays, and lower pollution levels.

II. LITERATURE REVIEW

Numerous approaches have been proposed to address traffic congestion using technological advancements, particularly in the fields of image processing, machine learning, and adaptive control systems.

- Video-Based Traffic Control: As outlined in [2], video feeds are processed to extract traffic data, which is then analyzed using a C++ algorithm to optimize signal timings. The study compared hardcoded and dynamic algorithms, with the latter demonstrating a 35% improvement in traffic flow management.
- Arduino-UNO and MATLAB Integration: Reference [3] discusses a system that captures images via cameras and processes them in MATLAB. The images undergo thresholding by filtering out hues and saturation to estimate traffic density. This data is then relayed to an Arduino-UNO, which adjusts green light durations accordingly. However, this method faces challenges such as vehicle overlap and misidentification of non-vehicular objects like billboards or trees, leading to inaccuracies.
- Fuzzy Logic-Based Control: A fuzzy logic-driven system is described in [4], employing two fuzzy controllers to adapt signal timings based on real-time traffic conditions. Simulations conducted using VISSIM and MATLAB indicated improved traffic conditions, particularly under low-density scenarios.
- Artificial Neural Networks (ANN) and Fuzzy Systems: Reference [5] proposes a hybrid model using ANN and fuzzy logic controllers. Images captured at intersections are processed through grayscale conversion, normalization, and segmentation techniques. The ANN processes the segmented images to determine vehicle counts, which the fuzzy controller then uses to adjust signal timings. This system achieved an average error rate of 2% and an execution time of 1.5 seconds.
- Support Vector Machines (SVM): Another approach [6] utilizes SVM algorithms coupled with image processing techniques to analyze traffic density and detect red-light violations. Real-time video feeds are segmented into frames, converted to grayscale, and processed using OpenCV before applying SVM classification.
- Adaptive Light Timer Using Image Processing: Reference [7] presents a system incorporating microcontroller-based timers, high-resolution imaging devices, and MATLAB processing. While effective in adjusting signal timings based on traffic density, this system lacks capabilities for prioritizing emergency vehicles or detecting accidents at intersections.
- Review of Intelligent Traffic Management Techniques: A comprehensive review in [8] highlights various methodologies for traffic light management. These include:
  - Vehicular Ad-hoc Networks (VANETs): Vehicles communicate their location to nearby intelligent traffic lights using GPS. However, high deployment costs limit feasibility.
  - Infrared Sensor Systems: Sensors detect vehicle IDs for traffic control and emergency vehicle prioritization, though the system's reliance on line-of-sight limits flexibility.
  - Fuzzy Logic Controllers: Utilized for optimizing and extending green light phases based on real-time traffic data from video cameras or road sensors.
  - Photoelectric Sensors: Measure traffic load by calculating the weight of roads but incur high maintenance costs.
  - Video Imaging and Dynamic Background
  - Subtraction: While effective in vehicle detection, this method struggles with occlusion and shadow overlap.

III. PROPOSED SYSTEM

A. Proposed System Overview

The proposed system leverages real-time imagery from CCTV cameras at traffic intersections to dynamically manage traffic signal timings. By integrating image processing and object detection techniques, the system calculates traffic density and adjusts green light durations accordingly. Figure 1 illustrates the architecture of the system, where live images are fed into a vehicle detection algorithm powered by YOLO (You Only Look Once). The algorithm classifies vehicles into categories such as cars, bikes, buses, trucks, and rickshaws, providing a detailed density analysis for each lane. This data is then processed by a signal-switching algorithm that determines the optimal green light duration based on current traffic conditions, while ensuring no lane experiences excessive delays.

B. Vehicle Detection Module

The core of the detection system is the YOLO algorithm, a real-time object detection model known for its accuracy and speed. A custom YOLO model

was trained to identify various vehicle classes, including cars, motorcycles, heavy vehicles (buses and trucks), and rickshaws.

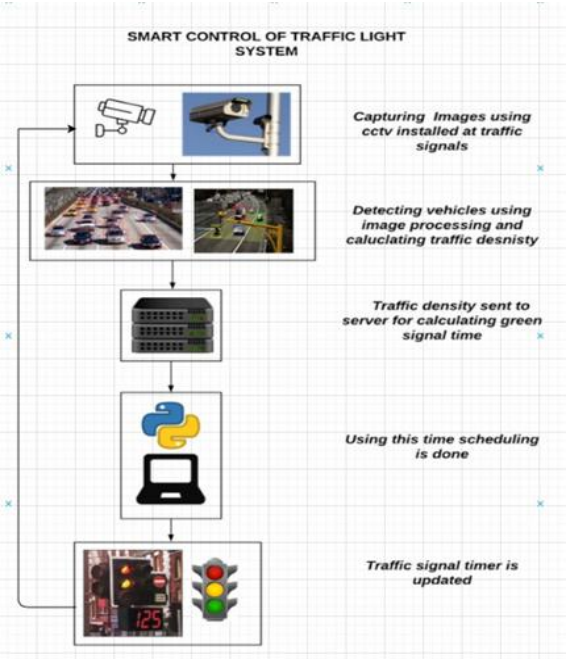

Fig. 1. Proposed System Model

YOLO [9] uses one convolutional neural network (CNN) over the whole image and splits itintoregions, then predicts bounding boxes and class probabilities for each of them. This method ensures high detection accuracy while maintaining real-time performance. The "single look" approach refers to YOLO's efficiency in requiring only one forward pass through the network to generate predictions.

For enhanced processing rate, the system leverages the Darknet framework, which is an open-source neural network developed in C and CUDA. Darknet produces excellent accuracy records on the ImageNet dataset with 72.9% top-1 accuracy and 91.2% top-5 accuracy.

The dataset used for training the model was curated from online sources and manually annotated using Label IMG, an image labeling tool. The YOLO configuration files were modified to accommodate four vehicle classes, adjusting the number of filters and output neurons accordingly. Once trained, the model was integrated with OpenCV to facilitate vehicle detection. A confidence threshold ensures that only high-certainty detections are considered, and the results are output in JSON format, listing detected vehicles along with their confidence scores and bounding box coordinates.

Fig. 2 Our vehicle detection model was used on test images as displayed in fig 2. The left side of the figure presents the original image while on the right side, the corresponding image after applying the vehicle detection model is shown. The output is drawn with bounding boxes.

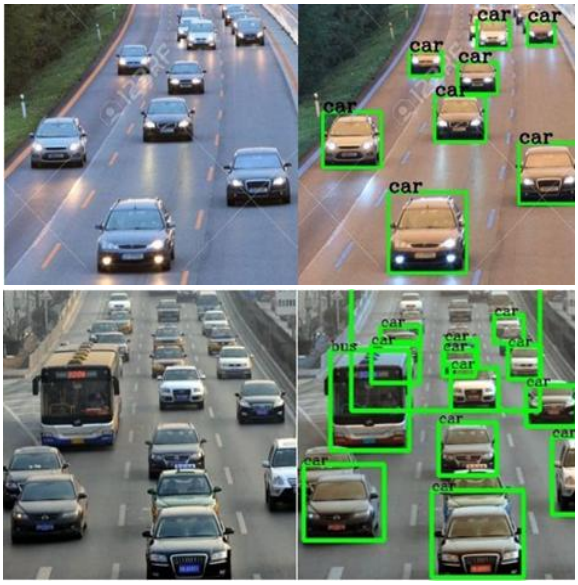

Fig. 2. Vehicle Detection Results

C. Signal Switching Module

The signal-switching algorithm dynamically adjusts traffic light phases based on real-time vehicle detection data. The algorithm processes JSON data from the detection module, which includes the count and classification of vehicles at each intersection.

Key factors considered by the algorithm include:

1. Processing Time: Ensuring the system processes traffic data and adjusts signals promptly.
2. Number of Lanes: Different intersections may have varying lane configurations.
3. Vehicle Classification: Differentiating between vehicle types (e.g., cars, trucks, bikes) for precise density calculation.
4. Traffic Density Calculation: Using detected vehicle counts to determine overall congestion levels.
5. Vehicle Start-Up Lag: Accounting for delays as vehicles begin moving, with increased lag for those further back in the queue.
6. Average Speed by Vehicle Class: Estimating how quickly different vehicles can clear the intersection.
7. Minimum and Maximum Green Light Duration:

Ensuring fairness by preventing any lane from being deprived of green time for too long.

The algorithm operates in cycles, with each signal transitioning through the typical sequence: Red → Green → Yellow → Red. While the system adapts green light durations based on traffic density, the overall cycle order remains consistent with traditional systems to avoid confusing drivers. To prevent delays, vehicle detection occurs five seconds before a signal is scheduled to turn green, allowing the system to process data and adjust timings in real-time. Our vehicle detection model was used on test images as displayed in fig 2. The left-hand side of the figure depicts the original image whereas on the right-hand side, the respective image after processing with the vehicle detection model is displayed in equation (1). The output is depicted with bounding boxes and respective labels.

The green light duration is calculated using the following formula:

$$GST = \frac{\sum(noOfLanesOfveh.class * AvgTimeOfveh.class)}{(\text{NoOFLanes} + 1)}$$

……. equation (1)

Where:

- GST is the green signal time.
- noOfVehiclesOfClass represents the count of each vehicle class detected.
- averageTimeOfClass refers to the estimated time it takes for vehicles of that class to clear the intersection.
- noOfLanes indicates the number of lanes at the intersection.

This dynamic approach optimizes signal timings, reduces unnecessary delays, and enhances overall traffic flow.

In order to enhance traffic control, the typical time taken by each category of vehicle to move through an intersection can be predefined according to location, i.e., region-wise, city-wise, locality-wise, or even intersection-wise depending on the nature of the intersection. Thisu0 can be done by analyzing data from the relevant transport authorities. Instead of switching in the densest direction initially, the signals do so cyclically. This aligns with the present system, in which individuals do not have to modify their actions or get confused since the signals change to green sequentially in a known pattern. The yellow signals have also been taken into consideration, and the signal order is identical to the current system.

Order of signals: Red →Green →Yellow →Red

D. Simulation Module

Pygame was used to create a simulation from the ground up that-mimics actual traffic. It assists in visualization of the system and wheelers come from every corner. Some of the vehicles from the far-right lane turn across the intersection to make the simulation more realistic. Random numbers also decide if and when a generated vehicle will turn. It has a timer showing how long has elapsed since simulation started. A snapshot of the simulation's ultimate output is displayed in Fig. 3. A cross-platform library of Python modules named Pygame was developed solely for the development of video games. It includes sound and graphics libraries developed solely for the Python programming language. Pygame augments the wonderful SDL library with additional features. This allows users to create multimedia applications and fully featured games with the Python programming language. Pygame is very portable and compatible with nearly all operating systems and platforms. It is LGPL-licensed and free [11].

IV. RESULTS AND ANALYSIS

A. Performance of the Vehicle Detection Module

The vehicle detection component of the system underwent rigorous testing using a diverse set of images featuring varying traffic densities and vehicle types. The accuracy of the detection algorithm consistently ranged between 75% and 80%. While these results are promising, there's room for enhancement. The primary limitation stems from the dataset used for model training, which lacked comprehensive real-world traffic scenarios. To further improve detection accuracy, incorporating actual CCTV footage from urban intersections would allow for more robust training, enabling the model to better recognize vehicles under diverse conditions such as different lighting, weather variations, and occlusions.

Sample detection outputs are illustrated in Figure 3, showcasing the system's ability to accurately identify and classify vehicles like cars, motorcycles, buses, trucks, and rickshaws. The bounding boxes and corresponding labels demonstrate the model's effectiveness in real-time scenarios.

B. Evaluation of the proposed adaptive system

To assess the efficacy of the proposed adaptive traffic control system, a series of simulations were conducted, comparing its performance to that of traditional fixed-timer systems. Fifteen distinct simulation scenarios were executed, each lasting **five minutes** and featuring varying traffic distributions across four intersection lanes.

KeyPerformance Indicator:

The primary metric for performance evaluation was the number of vehicles successfully passing through the intersection during each simulation. This metric directly reflects the system's ability to minimize idle green light time and reduce overall vehicle wait times.

Traffic Distribution Methodology:

Traffic flow was distributed probabilistically across four lanes. For instance, in Table 1, a distribution of [300, 600, 800, 1000] indicates that the probability of a vehicle being in Lane 1 is [300/1000], in Lane 2 in comparison with the existing static system. There are four traffic lights and a four-way crossing there. Above every signal is a timer that displays how many seconds it will take for the light to turn from green to yellow, from yellow to red, or from red to green. The number of vehicles that have passed through the crossing is also displayed beside every signal. Motorcars, motorbikes, buses, lorries, and three-s [300/1000], in Lane 3 is [200/1000], and in Lane 4 is [200/1000]. Table II indicates the stimulation result of the Adaptive System.

TABLE I. SIMULATION RESULTS OF CURRENT STATIC SYSTEM

Exp. No.	Distribution	Lane1	Lane 2	Lane 3	Lane 4	Sum
1	[300,600,800,1000]	70	52	52	65	239
2	[500,700,900,1000]	112	49	48	31	240
3	[250,500,750,1000]	73	53	63	62	251
4	[300,500,800,1000]	74	44	65	71	254
5	[700,800,900,1000]	90	32	25	41	188
6	[500,900,950,1000]	95	71	15	14	195
7	[300,600,900,1000]	73	63	69	24	229
8	[200,700,750,1000]	54	89	10	67	220
9	[940,960,980,1000]	100	10	8	4	122
10	[400,500,900,1000]	81	29	88	37	235
11	[200,400,600,1000]	42	47	54	86	229
12	[250,500,950,1000]	39	52	93	22	206
13	[850,900,950,1000]	74	10	13	17	114
14	[350,500,850,1000]	49	46	69	50	214
15	[350,700,850,1000]	51	64	37	43	195

C. Analysis of Results

The simulation results highlight the superior performance of the adaptive traffic control system compared to the static system across all tested scenarios. Key observations include:

- Uniform Traffic Distribution (Simulations 1–4): When traffic was evenly distributed across lanes, the adaptive system showed a modest improvement of about 9%. This is expected as balanced traffic flow doesn't significantly benefit from dynamic adjustments.
- Moderate Traffic Skew (Simulations 5–8, 14–15): In scenarios where traffic density varied moderately across lanes, the adaptive system outperformed the static system by approximately 22%. This reflects the system's ability to prioritize lanes with higher congestion effectively.

Skewed Traffic Distribution (Simulations 9 and 13): Under conditions of extreme traffic imbalance, the adaptive system demonstrated a performance improvement of up to 36%. This significant gain underscores the importance of real-time traffic density assessment in mitigating congestion.

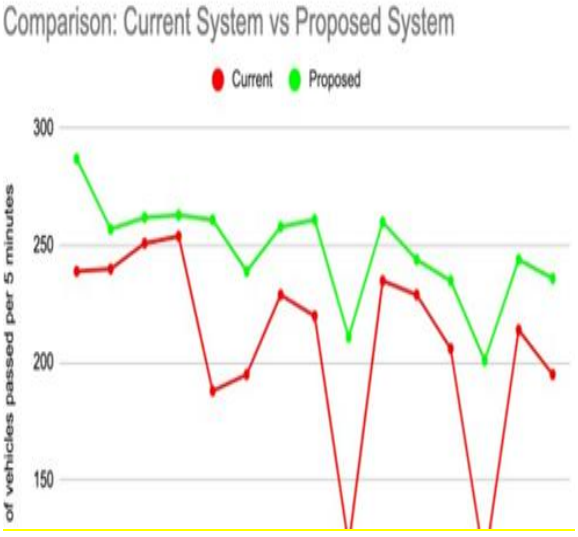

Fig. 3. Comparison of current static system and proposed adaptive system

TABLE II. SIMULATION RESULTS – PROPOSED ADAPTIVE SYSTEM

No.	Distribution	Lane1	Lane 2	Lane 3	Lane 4	Sum
1	[300,600,800,1000]	87	109	41	50	287
2	[500,700,900,1000]	128	55	49	25	257
3	[250,500,750,1000]	94	50	60	58	262
4	[300,500,800,1000]	89	46	69	59	263
5	[700,800,900,1000]	185	25	23	28	261
6	[500,900,950,1000]	94	118	11	16	239
7	[300,600,900,1000]	87	68	70	33	258
8	[200,700,750,1000]	56	108	19	78	261
9	[940,960,980,1000]	193	6	5	7	211
10	[400,500,900,1000]	97	29	100	34	260
11	[200,400,600,1000]	26	52	67	99	244
12	[250,500,950,1000]	52	75	101	7	235
13	[850,900,950,1000]	154	17	12	18	201
14	[350,500,850,1000]	64	53	80	47	244
15	[350,700,850,1000]	66	82	40	48	236

D. Average Performance Gain

Across all simulations, the adaptive system consistently facilitated 23% more vehicles passing through intersections compared to the fixed-timer system. This suggests a substantial reduction in idle green signal time and shorter waiting periods for vehicles at red lights.

Comparative Benchmarking:

When compared to other adaptive systems:

- Reference [2] achieved a detection accuracy of 70%, whereas the proposed system achieved 80% accuracy.
- Reference [3] claimed an improvement in average performance of 12% compared to static systems, whereas our adaptive system performed with 23% improvement.

V. CONCLUSION AND FUTURE WORK

The adaptive traffic control system proposed in this study dynamically adjusts green light durations based on real-time traffic density, resulting in significantly improved traffic flow. By utilizing live feeds from CCTV cameras and employing YOLO-based object detection, the system ensures that intersections with higher traffic volumes receive proportionally longer green signals.[10] This approach minimizes delays, reduces fuel consumption, and contributes to lower pollution levels.

Simulation results confirm the system's 23% performance improvement over conventional fixed-timer traffic lights, with even greater efficiency in cases of uneven traffic distribution. The system's reliance on existing CCTV infrastructure also makes it a **cost-effective solution** compared to other intelligent traffic control technologies like pressure mats or infrared sensors.

Future Work

- Several enhancements can further improve the system’s functionality and impact:
- Traffic Violation Detection: Integrating algorithms to detect red-light violations by monitoring vehicles that cross a defined boundary during red signals. Additionally, identifying illegal lane changes using background subtraction and image processing can enhance law enforcement.
- Accident and Breakdown Detection: The system can be trained to recognize stalled vehicles, indicating potential accidents or breakdowns. By identifying vehicles that remain stationary in inappropriate locations (e.g., the middle of intersections), the system can trigger alerts for rapid response, thereby reducing congestion.
- Synchronization Across Multiple Intersections: Implementing signal coordination between adjacent intersections can create "green waves," allowing vehicles to pass through multiple signals with minimal stopping. This approach can further enhance urban traffic flow.[12]
- Emergency Vehicle Prioritization: By training the system to recognize emergency vehicles such as ambulances or fire trucks, traffic lights may be dynamically tuned to offer a safe route for such vehicles, shortening response times in emergency situations.
- Utilization of Real-World CCTV Data: Expanding the dataset to include real-life footage from various urban intersections will improve the system’s detection accuracy, making it more adaptable to diverse traffic scenarios, lighting

conditions, and weather variations.

REFERENCES

[1] TomTom.com, 'Tom Tom World Traffic Index', 2019. [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/ranking/

[2] Khushi, "Smart Control of Traffic Light System using Image Processing," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Mysore, 2017, pp. 99-103, doi: 10.1109/CTCEEC.2017.8454966.

[3] A. Vogel, I. Oremović, R. Šimić and E. Ivanjko, "Improving Traffic Light Control by Means of Fuzzy Logic," 2018 International Symposium ELMAR, Zadar, 2018, pp. 51-56, doi: 10.23919/ELMAR.2018.8534692.

[4] A. A. Zaid, Y. Suhweil and M. A. Yaman, "Smart controlling for traffic light time," 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Aqaba, 2017, pp. 1-5, doi: 10.1109/AEECT.2017.8257768.

[5] Renjith Soman "Traffic Light Control and Violation Detection Using Image Processing"." IOSR Journal of Engineering (IOSRJEN), vol. 08, no. 4, 2018, pp. 23-27

[6] (N.d.). Researchgate.net. Retrieved May 14, 2025, from https://www.researchgate.net/publication/269310721_Smart_traffic_lights_switching_and_traffic_density_calculation_using_video_processing

[7] Srivastava, S. (2016, March 29). *Adaptive traffic light timer control (ATLTC)*. NERD. https://www.iitk.ac.in/nerd/web/technology/adaptive-traffic-light-timer-control-atltc/

[8] Shinde, S., Tech Student, S. J., & Assistant Professor. (n.d.). *ISSN (online): 2349-6010*. Ijirst.org. Retrieved May 14, 2025, from https://ijirst.org/articles/IJIRSTV2I9077.pdf

[9] ODSC-Open Data Science. (2018, September 25). *Overview of the YOLO object detection algorithm*. Medium. https://odsc.medium.com/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0

[10] J. Hui, 'Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3', 2018. [Online]. Available: https://medium.com/ @jonathan_hui/ real-time-object-detection-with-yolo-yolov2-28b1b93e2088

[11] 'Pygame Library', 2019. [Online]. Available: https://www.pygame.org/wiki/about

[12] 'Traffic Signal Synchronization'. [Online]. Available: https://www.cityofirvine.org/signal-operations-maintenance/traffic- signal-synchronization