COMPARITIVE ANALYSIS OF HEURISTIC TECHNIQUES FOR VEHICLES MOVEMENT DETECTION

- **BTP CODE : B24SMP03**

OUR TEAM

Mentor

Dr. S Manipriya

S20210010173

Peddineni Rupesh chowdary

S20210010208

S.V.S.Apparao

S20210010164

Oothuru khadar basha

RECAP

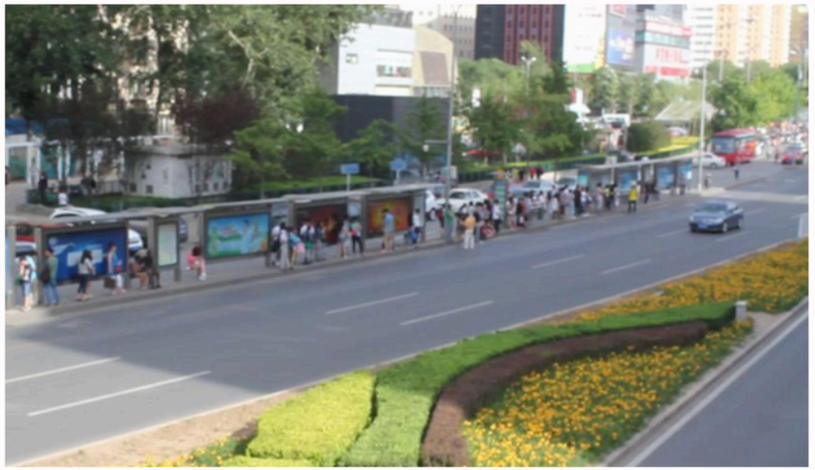
01 Exploring research papers and Dataset collection

02 Developed Road Surface segmentation Algorithm

03 Developed 3 Adaptive Algorithms to solve the problem statement

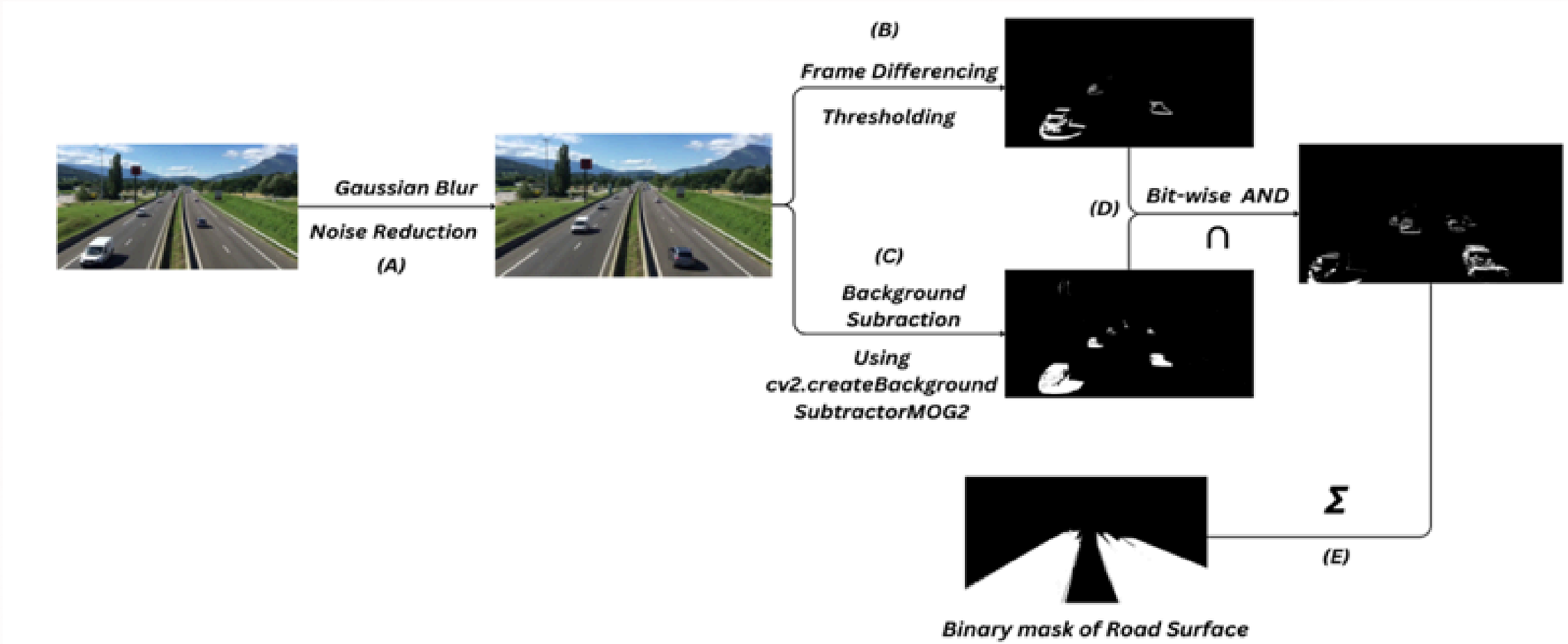
04 Comparison of 3 Adaptive algorithms based on Evaluation metrics

UA-DETRAC Dataset



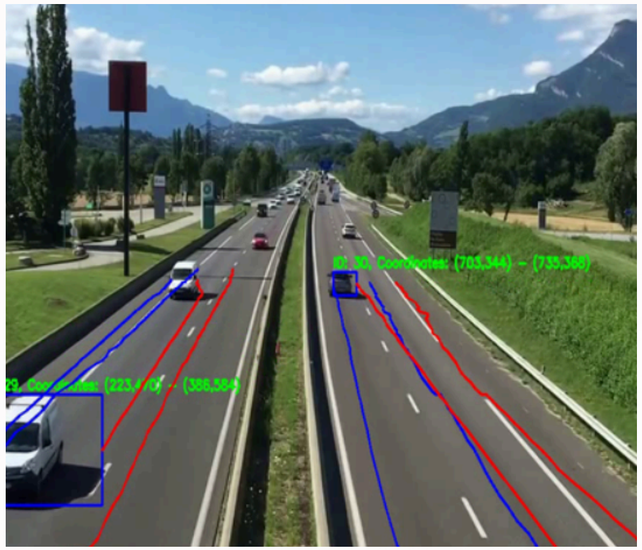



Road Surface segmentation



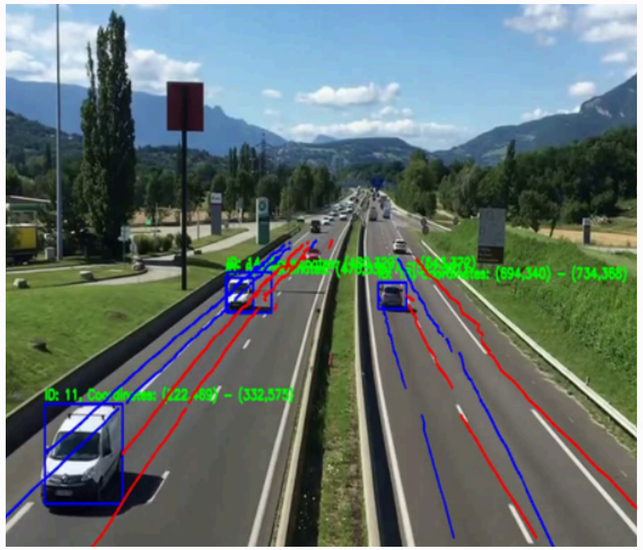
Adaptive Algorithms

Adaptive Blob Tracking



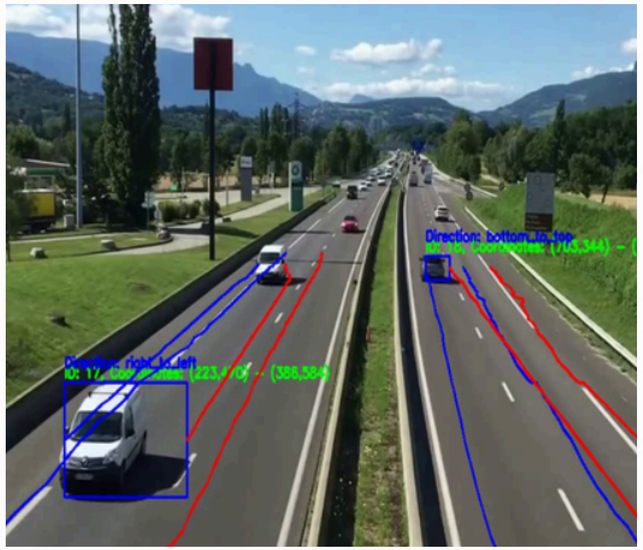
This method detects moving vehicles by identifying regions of connected pixels (blobs) that differ from the background. It is effective in stationary camera setups where objects stand out from the background.

Adaptive Yolo and Sort



YOLO is a real-time object detection algorithm that classifies and locates vehicles within video frames, while SORT tracks those detected objects across frames, assigning unique IDs to each vehicle for continued tracking .

Adaptive Optical Flow



This method estimates the motion of vehicles by analyzing the displacement of pixel intensities between consecutive frames, making it useful for tracking moving objects based on their movement patterns using lucas kanade optical flow

Comparison of Results

Videos	Adaptive Optical FLow	Adaptive Blob Tracking	Adaptive YOLO and SORT
1.) 	MSE : 1384.38 RMSE : 37.21	MSE : 648.25 RMSE : 25.46	MSE : 594.75 RMSE : 24.39
2.) 	MSE : 2467.25 RMSE : 49.671	MSE : 742.5 RMSE : 27.25	MSE : 3407.75 RMSE : 58.36
3.) 	MSE : 1133.75 RMSE : 33.67	MSE : 2260.05 RMSE : 47.54	MSE : 1889.25 RMSE : 43.46

*MSE: Mean Square Error

*RMSE: Root Mean Square Error

Comparative Analysis by space and time

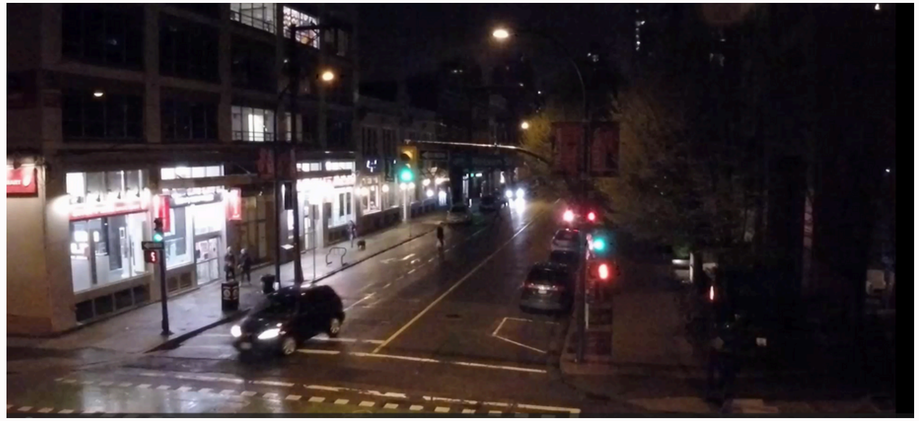
Contd..

- Analyzing an algorithm based on time and space is crucial
- Because these factors directly impact the algorithm’s practical usability, especially in real-world applications.
- Three kinds of videos are being considered for analyzing the algorithms
    - Well illuminated video
    - Moderately illuminated video
    - Weak illuminated video

Contd..

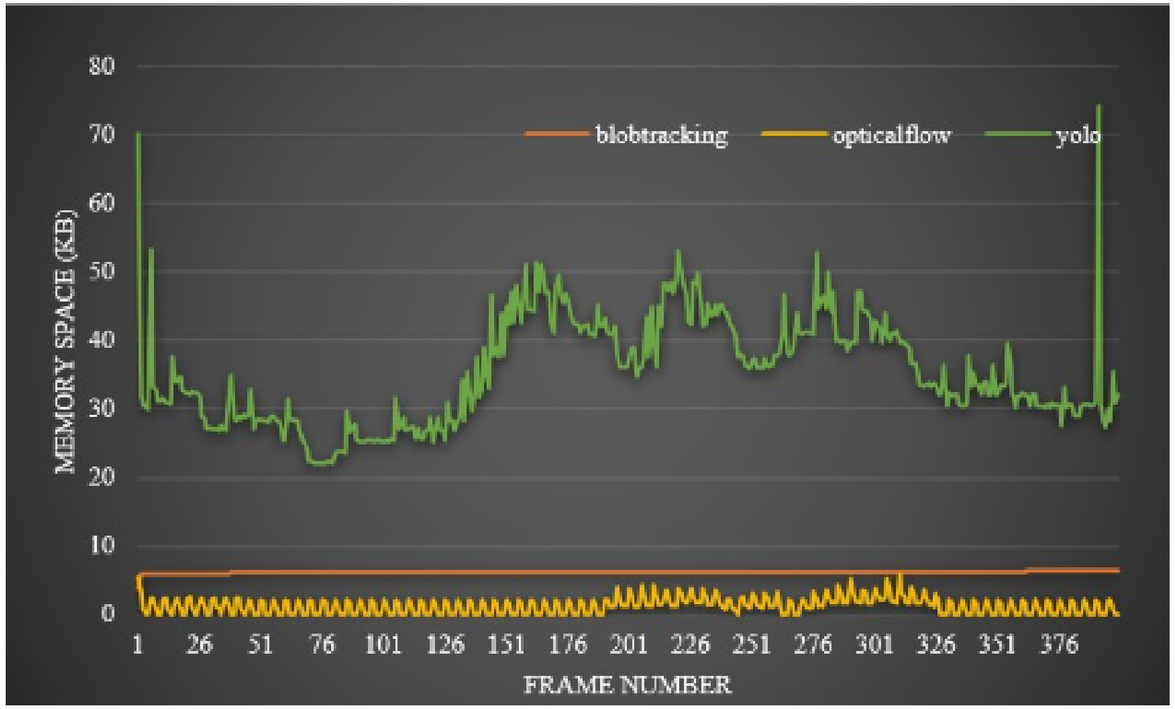
Well illuminated video frame

Moderately illuminated video frame



Weak illuminated video frame

Comparative Analysis by Memory Usage

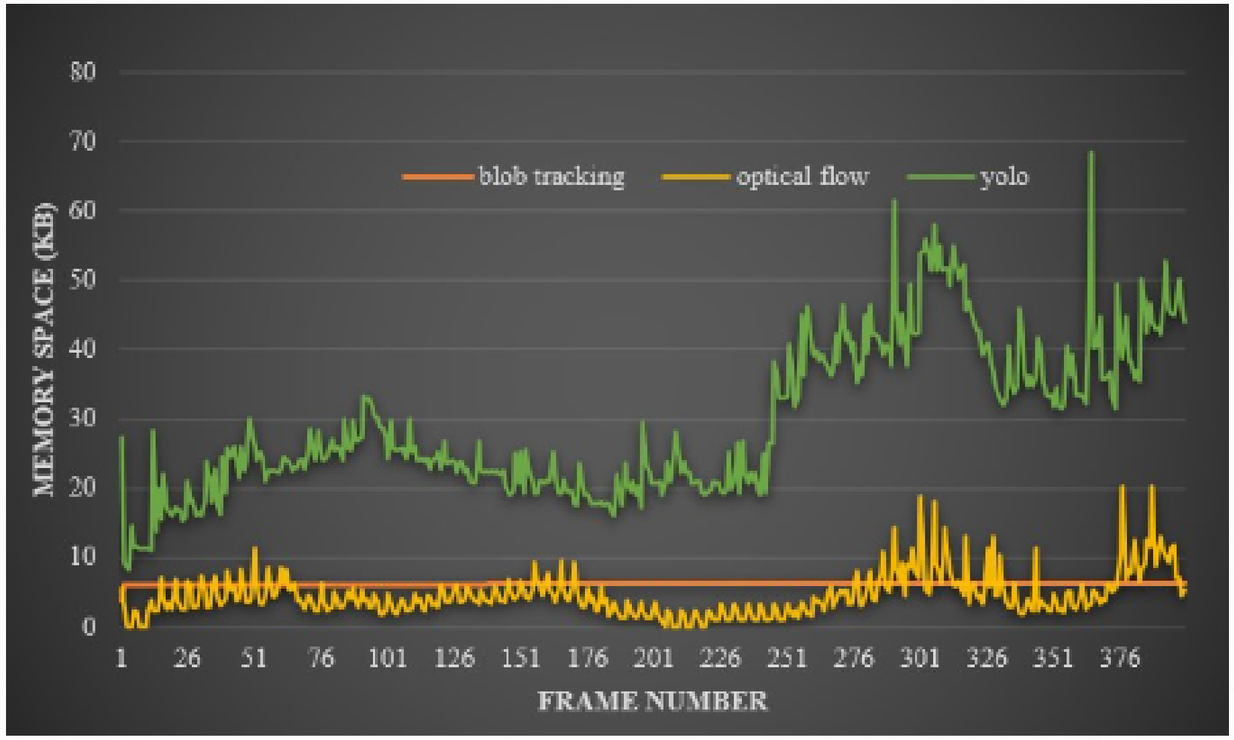


For well illuminated videos

- Optical flow follows closely with a maximum of **5.35 KB**
- Blob-tracking demonstrates memory usage, peaking at **6.3 KB.**
- YOLO exhibits significantly higher memory consumption, peaking at **70.96 KB**.
- This increase is due to YOLO’s deep neural network architecture

- Therefore, Blob tracking and optical flow excel in memory efficiency suitable for well-illuminated traffic videos in comparison with YOLO performance

Contd...

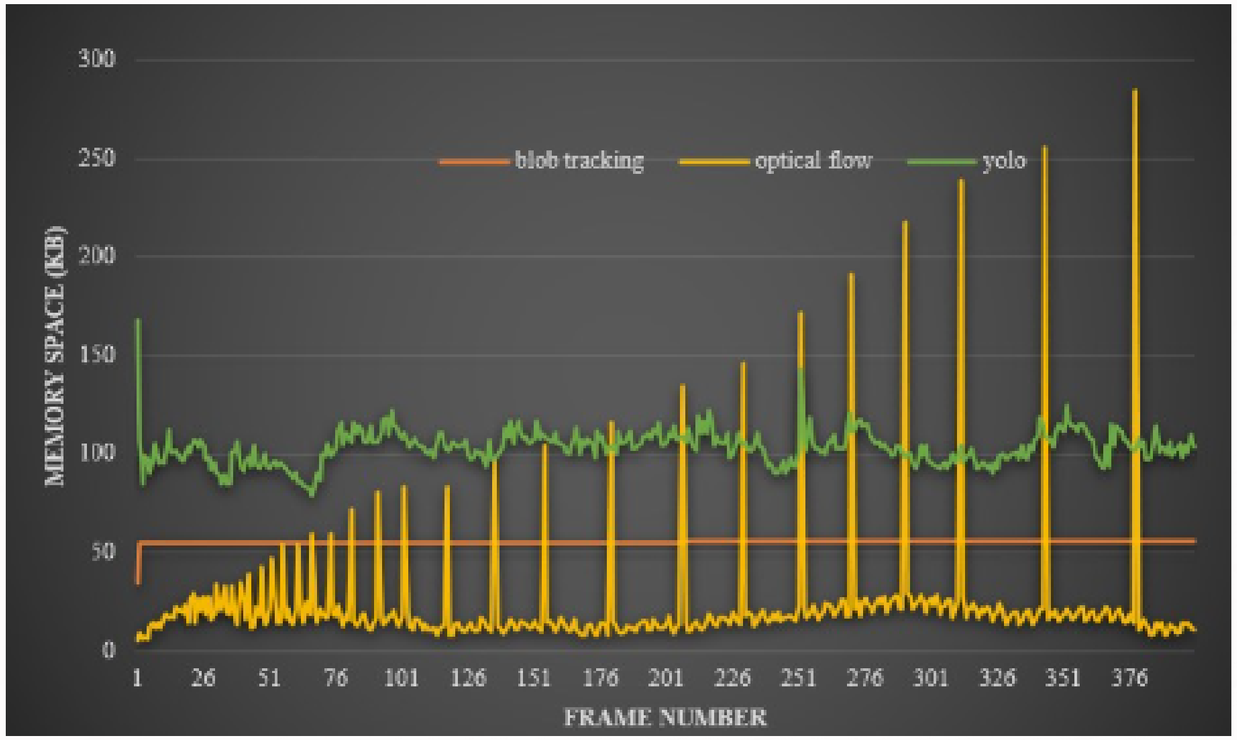


For moderately illuminated videos

- Here, Blob-tracking shows the lowest space complexity, with measurements ranging mostly between **3.87** and **6.25 KB**.
- Optical flow demonstrates moderate space usage, generally between **0** and **11.63 units,** occasionally peaking up to **20.69 KB**.
- In contrast, YOLO consistently exhibits the highest space complexity, ranging widely from **8.33** to **69.42 KB,**

- Therefore, Blob tracking and optical flow utilize minimal space suitable for moderately-illuminated traffic videos in comparison with YOLO algorithm

Contd...

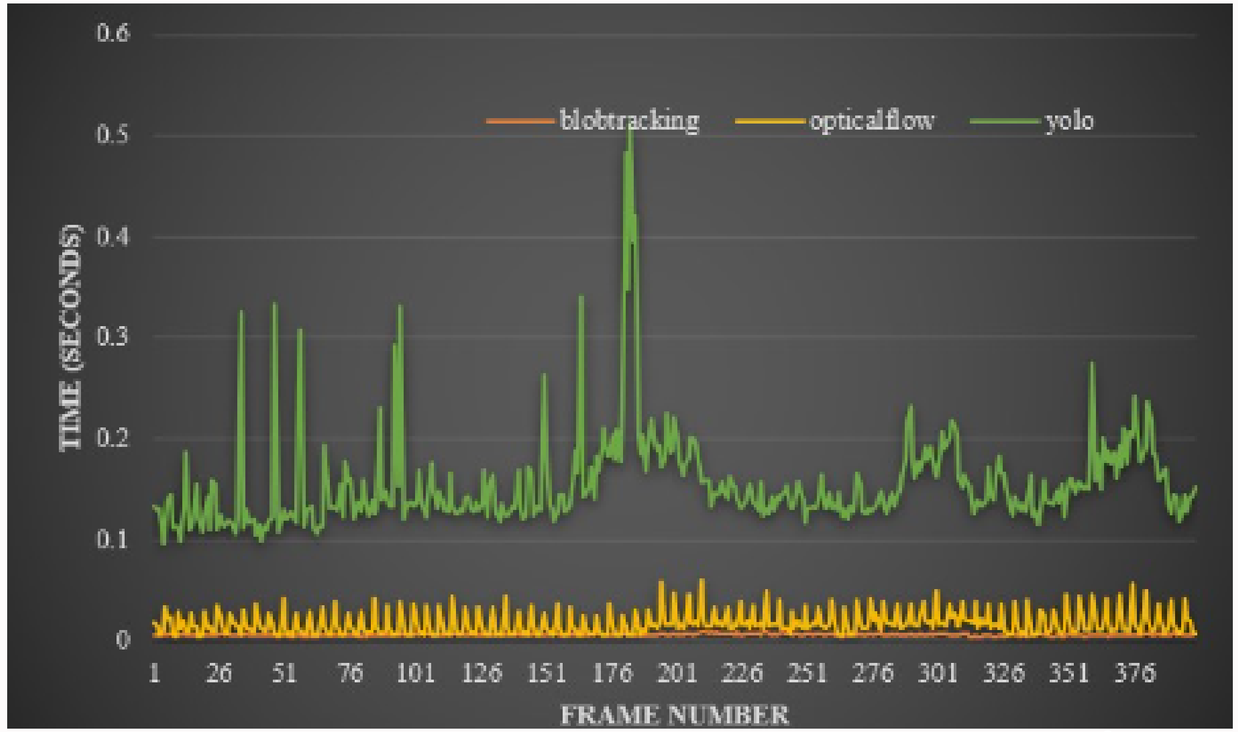


For weak illuminated videos

- Blob tracking generally requires the least memory, with measurements ranging from approximately **34.84 KB** to **55.61 KB.**
- Optical flow's memory usage varies more widely, from around **5.55 KB** to **256.04 KB**.
- YOLO consistently demands the most memory , with values ranging from **78.80 KB** to **145.84 KB**.
- These variations highlight the differing memory demands of each algorithm

- In conclusion, the memory usage is influenced by factors such as image illumination, processing complexity of the algorithm.

Comparative Analysis by Computational Time

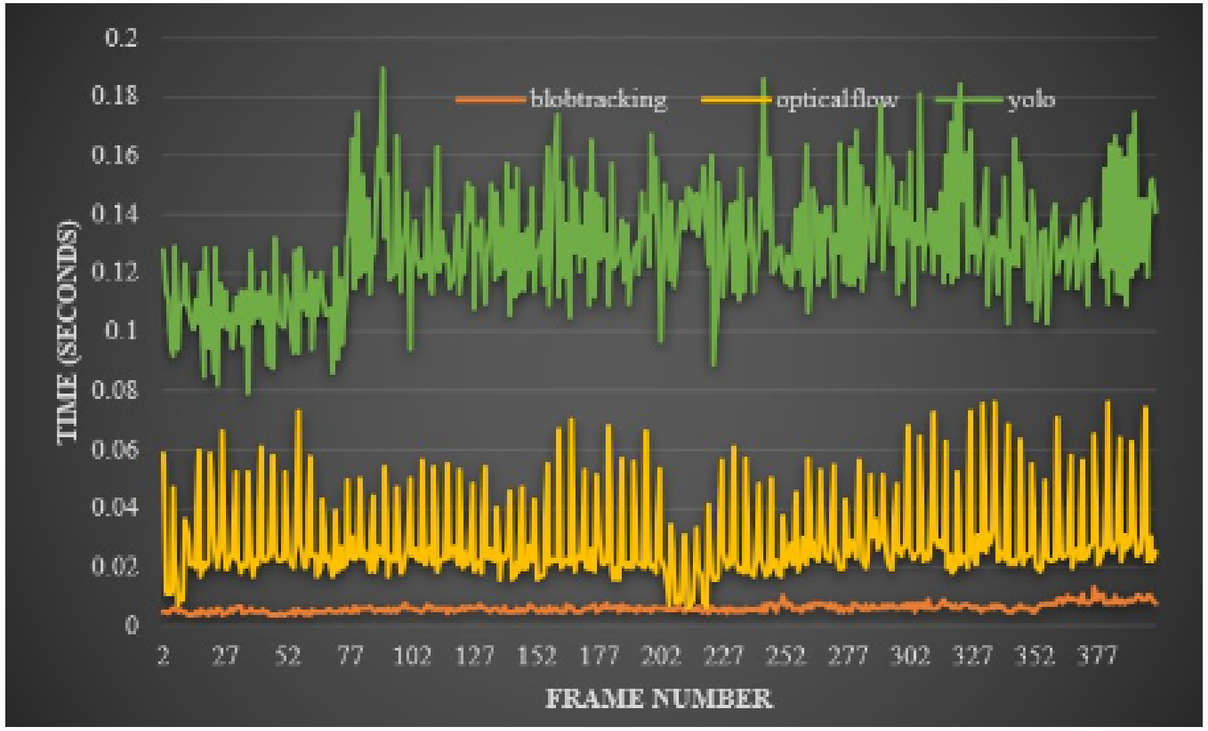


For well illuminated videos

- Blob Tracking consistently exhibits the lowest time complexity, with an average runtime of approximately **0.005 seconds per frame.**
- Optical Flow, with an average runtime of **0.016 seconds per frame**, demonstrates moderate performance
- Conversely, YOLO, despite its significantly higher average runtime of **0.145 seconds per frame**

- Therefore, considering this scenario blob tracking and optical flow are excel in computational time efficiency compared with YOLO

Contd...

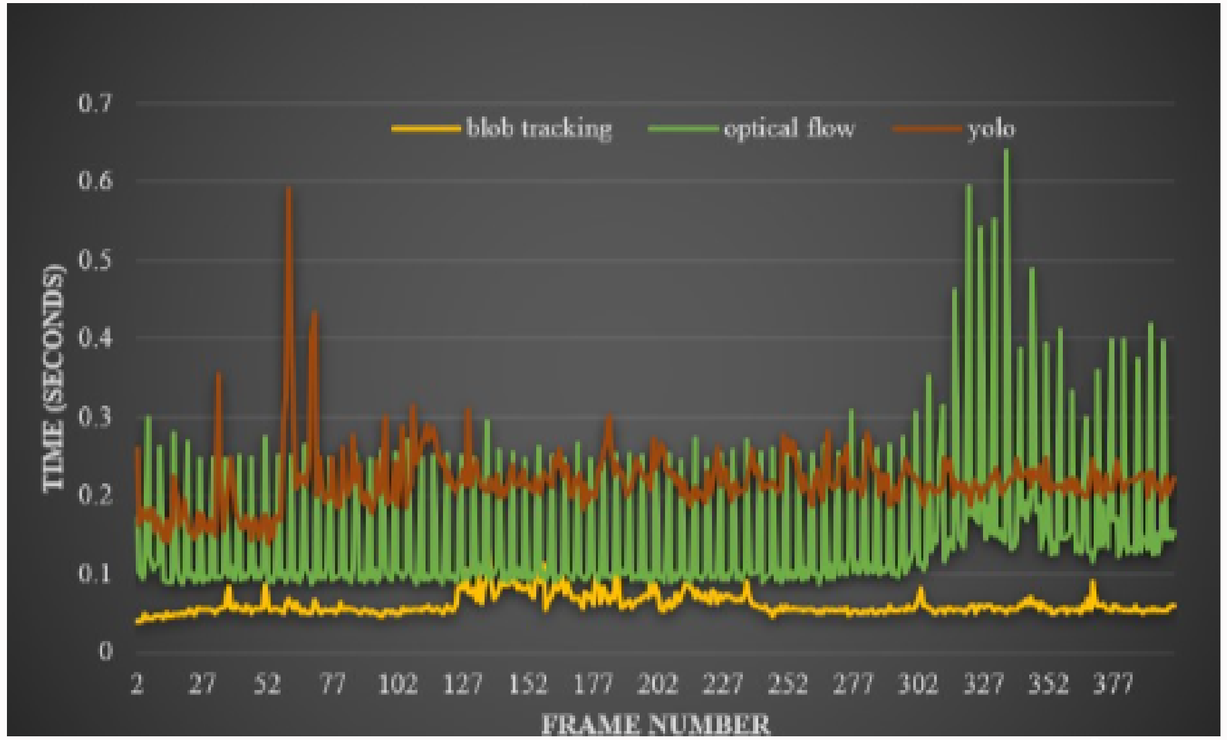


For moderately illuminated videos

- Blob Tracking consistently exhibits the lowest time complexity, with an average runtime of approximately **0.005 seconds per frame**.
- Optical flow, with an average of about **0.025 seconds per frame** remains relatively efficient.
- In contrast, YOLO exhibits significantly higher execution times, averaging around **0.125 seconds per frame**.

- This suggests Blob tracking is the most computationally efficient among the three methods.

Contd...



For weak illuminated videos

- Blob tracking shows the lowest average time complexity of approximately **0.055 seconds.**
- Optical flow estimation averages around **0.102 seconds.**
- YOLO object detection being the most computationally intensive with an average of **0.221 seconds.**

- These measurements highlight YOLO's higher computational demands compared to Blob tracking and optical flow.

Summary of Analysis

- From the analysis performed it can be concluded that Adaptive Blob Tracking emerged as the most computationally efficient
- It’s averaging a **0.005 seconds per frame**, ideal for real-time applications.
- Blob Tracking also demonstrated superior memory efficiency **(peak of 6.3 KB)**
- Optical flow demonstrated relative good memory efficiency but not as much as Blob tracking approach
- Whereas YOLO required significantly more memory in comparison with Blob tracking and Optical flow

Need of GUI

Ease of Use:

- A GUI simplifies complex tasks, making it easy for users to interact with the system through buttons rather than typing commands.

Flexible:

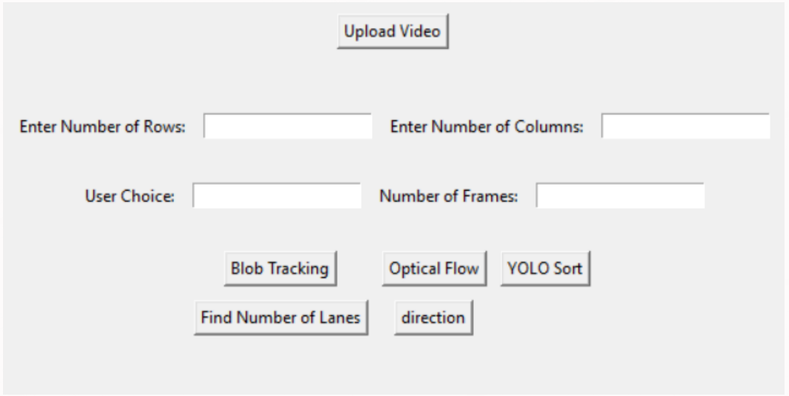
- It offers different options, such as choosing algorithms and entering the number of frames.
- This makes it useful for both automatic and custom tasks.

Time-Saving:

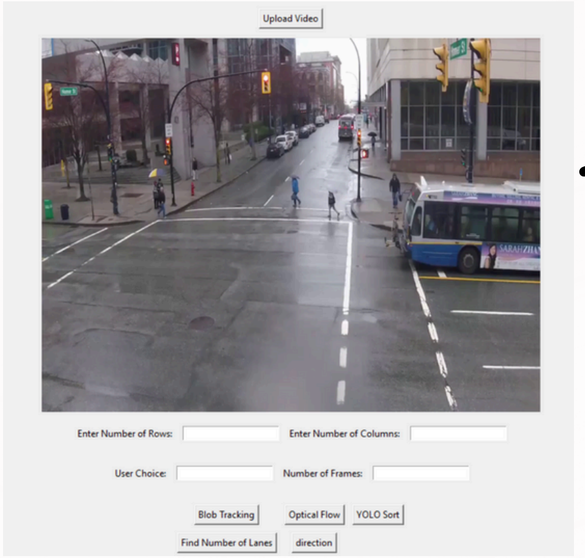
- Users can quickly choose and run algorithms without needing to change any code, which makes the process faster and easier

The layout is clean and easy to understand, so users won't feel confused when using it

Work Flow of GUI

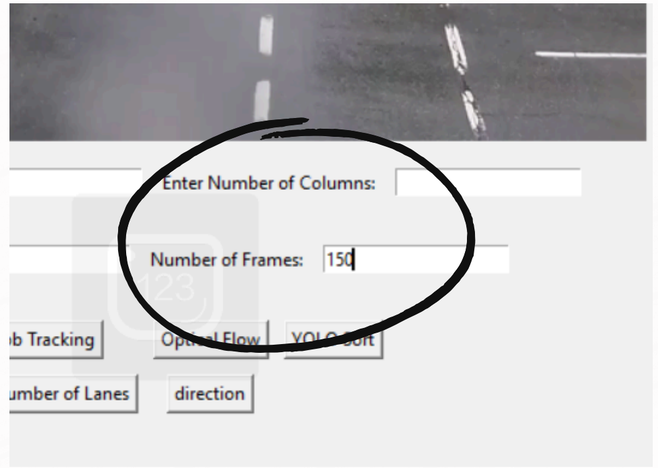


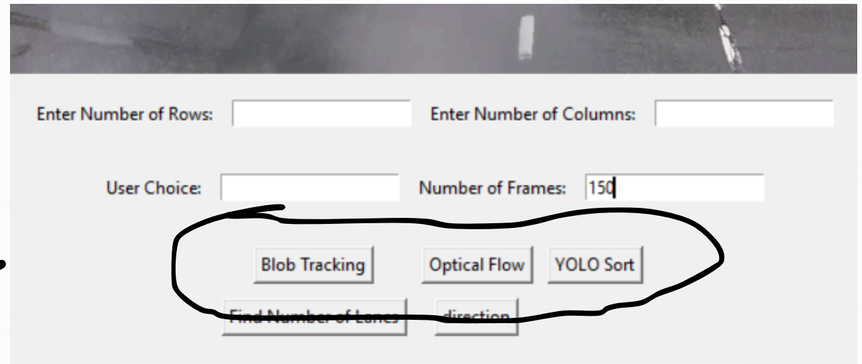
User Interface



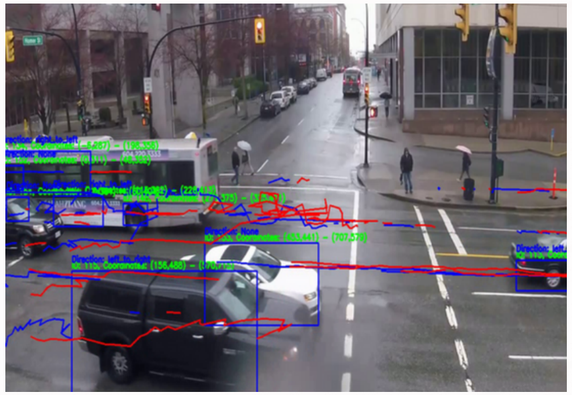
Uploading Video

Selecting number of frames





Selecting methods



output of the method

Timeline

Review 1

Understand various research works, explore the dataset and conduct basic implementation

Review 2

Explore more research papers and implement different approaches of our problem statement

Review 3

Build GUI for user experience and Uncover areas for improvement by performing Analysis

Review 4

Implement dynamic thresholding. Try out the solution on detection technique and Document the research work

References

- Huansheng Song, Haoxiang Liang* , Huaiyu Li, Zhe Dai and Xu Yun, **“Vision-based vehicle detection and counting system using deep learning in highway scenes”**, European Transport Research Review, SpringerOpen, pp., 2019.

- Hadi Ghahremannezhad, Hang Shi, Chengjun Liu, **“Robust Road Region Extraction in Video Under Various Illumination and Weather Conditions”**, IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS) pp., 2020.

- Hadi Ghahremannezhad, Hang Shi, Chengjun Liu, **“Automatic Road Detection in Traffic Videos”**, IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS) pp., 2021.

THANK YOU

- **BTP CODE : B24SMP03**