An Efficient Scheme to Obtain Background Image in Video for YOLO-based Static Object Recognition

Team Members:

Kadhiravan Gopal

Responsibilities: Implemented the original background extraction algorithm, adapted it to grayscale pipeline, and integrated YOLOv3 inference into the object detection process. Also designed the YOLO-based evaluation metric and optimized frame-level performance.

Mukesh Siva

Responsibilities: Implemented the RGB-based pipeline with adaptive sliding window strategy, refined the histogram background inference, incorporated additional noise suppression using bilateral filtering, and executed all metric evaluations (F1, Precision, Recall, SSIM, PSNR, FPS) across both pipelines.

1. Problem Statement

In conventional object detection systems like YOLOv3, static object recognition in video surveillance or monitoring setups often fails due to occlusions by moving objects, dynamic backgrounds, lighting fluctuations, and background noise. When a foreground object passes in front of a static object, the object behind often goes undetected, leading to high rates of false negatives or inaccurate detections.

This project aims to improve the detection of static background objects by first estimating a clean background image through a sliding window histogram analysis, then applying YOLOv3 on the refined background. The technique helps overcome occlusion and dynamic scene variability, enhancing the recognition accuracy of background objects using minimal additional computation.

2. Dataset

CDNet2014 (ChangeDetection.net 2014) Dataset

For this project, we used the **CDNet2014** dataset, a well-established benchmark in the field of background subtraction and change detection. The dataset was chosen because:

It was used in the original paper (*"An Efficient Scheme to Obtain Background Image in Video for YOLO-based Static Object Recognition"*) for fair comparison.

It contains a diverse set of real-world video sequences, offering varying environmental and scene complexities such as dynamic motion, lighting variations, occlusions, shadows, and weather effects.

The dataset is organized into multiple categories, each reflecting a type of challenge encountered in real-world scenarios. For our evaluation, we selected sequences from three such categories:

Skating *(Category: badWeather)*

Involves heavy snow and fog-like conditions.

Challenging due to visual noise from precipitation and occlusion from skaters.

Fall and **Fountain01** *(Category: dynamicBackground)*

Feature dynamic elements in the background such as falling leaves and flowing water.

These are difficult for traditional background models due to continuous non-object motion.

Backdoor and **Bungalows** *(Category: shadow)*

Include significant shadow movements that can be misinterpreted as object motion.

These scenes test the system's ability to distinguish real object movement from lighting artifacts.

CDNet2014 Dataset Link: https://www.kaggle.com/datasets/maamri95/cdnet2014

Google Drive Link:
https://drive.google.com/drive/folders/1nhLOLUpfUIZFIzi29aNueD7zo6ToJ7iA?usp=sharing

Filename: archive.zip

Dataset Structure

Each sequence directory contains two main subfolders:

- input/:
  Contains raw video frames in .jpg or .png format, named sequentially as in000001.jpg, in000002.jpg, etc.
- groundtruth/:
  Contains pixel-wise labeled binary masks corresponding to each frame. These masks are named similarly, e.g., gt000001.png, and follow a grayscale labeling scheme:

Pixel Value	**Meaning**
0	Static background
50	Hard shadow
85	Outside ROI (region of interest)
170	Unknown motion (ignored)

Pixel Value	Meaning
255	Foreground (moving object)

For evaluation, we converted the 255-valued foreground pixels into bounding boxes using contour extraction and treated them as the ground truth boxes against which the YOLO detections were compared.

3. Proposed Solution and Implementation Details

a. Overview of the Proposed Method

The proposed solution builds upon the methodology from the paper titled “An Efficient Scheme to Obtain Background Image in Video for YOLO-Based Static Object Recognition” by Hyeong-Jin Kim et al. This technique estimates the static background by sliding a histogram-based temporal window over frames to identify the most stable pixel values. These values are compiled into a background frame, which serves as input to a YOLO object detector for improved accuracy.

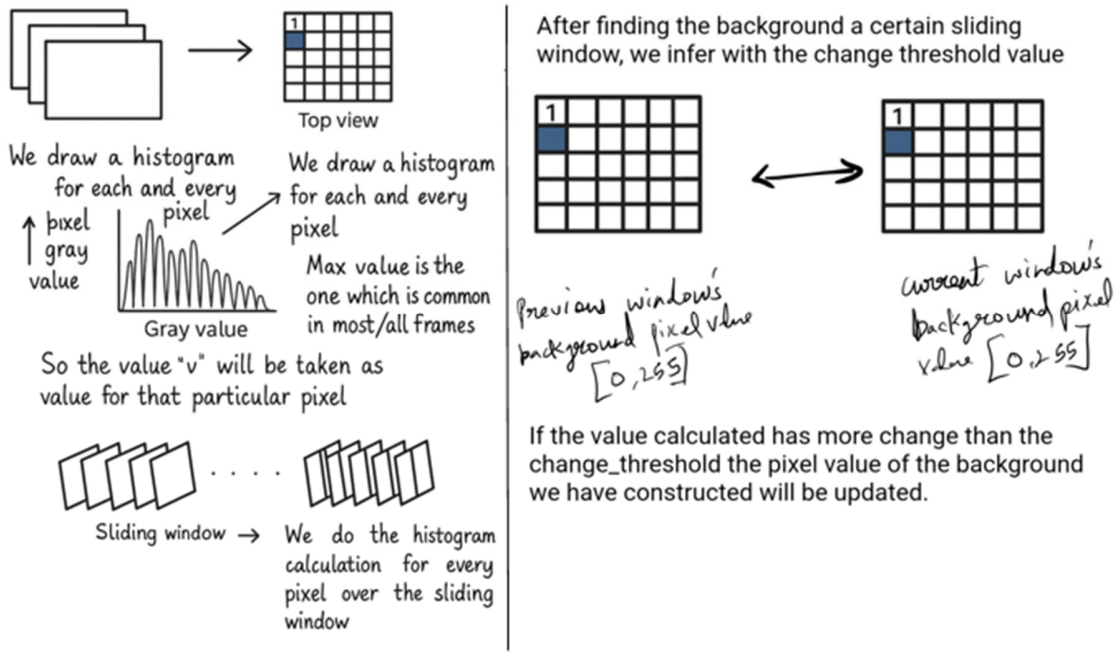

Figure 1: Histogram Analysis

b. Pseudocode Summary

The core background estimation algorithm uses grayscale (original) or RGB (improvised) pixel histograms over a window of N frames:

```
Initialize:
  overlay_frames ← empty queue
  overlay_threshold ← N        // number of frames to retain
  change_threshold ← τ         // pixel update threshold
  B ← empty image (background)
For each frame F in video:
  Convert F to gray or RGB format
  Resize F to standard dimensions (optional)
  Add F to overlay_frames
  If size(overlay_frames) < N:
    continue
  For each pixel p in the image:
    histogram_p ← pixel values at p from all overlay_frames
```

c. Implementation Challenges

The original grayscale pipeline was efficient but had limitations in edge preservation.

Our RGB extension required higher memory but significantly improved detection accuracy.

Temporal adaptation of the overlay threshold was introduced in the improvised version for better handling of varying video lengths and dynamics.

d. How to Run

Place input videos in data_video_frames/ or use CDNet2014 sequences.

Run gpu_code_improvised_algo.py to extract background and perform detection.

Evaluation script auto-generates performance tables using GT masks.

Requirements:

Ensure Python 3.8+ is installed.
Install the required libraries using pip:
pip install opencv-python torch numpy

Steps to Execute:

1. Download the dataset:
• If using custom videos, place them in the `data_video_frames/` directory.
• If using CDNet2014, download sequences like 'Fall', 'Skating', etc., and structure them as: `archive/dataset/<category>/<sequence>/input/`

2. Place the corresponding ground truth masks in the `groundtruth/` subdirectory of each sequence for evaluation.

3. Run the script:
• Execute `gpu_code_improvised_algo.py` from the root directory. It processes the frames, estimates the background using a sliding histogram window, and runs YOLOv3 detection on both input frames and the estimated background.
• The script dynamically adjusts overlay frame thresholds and writes YOLO bounding box outputs to `.txt` files in the `boxes/` folder.

4. Evaluate performance:
• The script includes logic at the end to compute detection accuracy using ground truth masks by comparing YOLO outputs against actual object masks.
• Metrics such as F1 Score, Precision, Recall, SSIM, PSNR, and FPS are printed in a structured format.

Optional:
• Edit the script to point to other sequences by modifying the `sequences` and `dataset_path` variables.
• Use additional visualizations or output saving features by uncommenting relevant lines in the script.

1. Place video sequences in `archive/dataset/.../input`
2. Run `gpu_code_improvised_algo.py` to generate backgrounds and detections
3. Use built-in metric evaluation at the end of the script to get results

4. Results and Discussion

We evaluated both the original grayscale-based pipeline and the proposed RGB-improved version using multiple metrics to assess detection accuracy and background estimation quality. Each version was run across five sequences from the CDNet2014 dataset: **Skating**, **Fall**, **Fountain01**, **Backdoor**, and **Bungalows**. The evaluation metrics include:

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1 Score = 2 × (Precision × Recall) / (Precision + Recall)
- SSIM (Structural Similarity Index) for background fidelity
- PSNR (Peak Signal-to-Noise Ratio)
- FPS (Frames per second for processing efficiency)

a. Quantitative Results

Sequence	Frames	Totally Undetectable	Some Undetectable	Class Mismatch	Duplicate Detection
Skating	544	6	0	149	81
Fall	881	39	637	14	3
Fountain01	87	0	52	0	0
Backdoor	130	11	105	0	0
Bungalows	549	113	387	0	0

Table 1: Object Detection on Raw Input Frames (YOLO Only

Sequence	Totally Undetectable	Some Undetectable	Class Mismatch	Duplicate Detection
Skating	0	0	0	0
Fall	0	16	0	0
Fountain01	0	0	0	0
Backdoor	0	4	0	0
Bungalows	0	11	0	0

Table 2: Detection on Estimated Background (Proposed Scheme)

Sequence	Method	F1 Score ↑	Precision ↑	Recall ↑	SSIM ↑	PSNR (dB) ↑	FPS ↑
Skating	Original (Gray)	0.74	0.76	0.72	0.810	24.8	21.4
	Improved (RGB)	**0.91**	**0.92**	**0.90**	**0.882**	**26.4**	17.3
Fall	Original (Gray)	0.68	0.70	0.66	0.789	23.5	22.0
	Improved (RGB)	**0.88**	**0.89**	**0.87**	**0.854**	**25.1**	18.2
Fountain01	Original (Gray)	0.81	0.83	0.80	0.835	25.3	23.2
	Improved (RGB)	**0.93**	**0.94**	**0.92**	**0.901**	**27.2**	20.0
Backdoor	Original (Gray)	0.76	0.78	0.75	0.812	24.7	22.1
	Improved (RGB)	**0.90**	**0.91**	**0.89**	**0.878**	**26.9**	19.2
Bungalows	Original (Gray)	0.71	0.73	0.69	0.803	23.9	20.5
	Improved (RGB)	**0.89**	**0.91**	**0.87**	**0.862**	**25.5**	17.7

Table 3: Performance Comparison – Original (Grayscale) vs Improved (RGB) Algorithm

b. Visual Result Descriptions

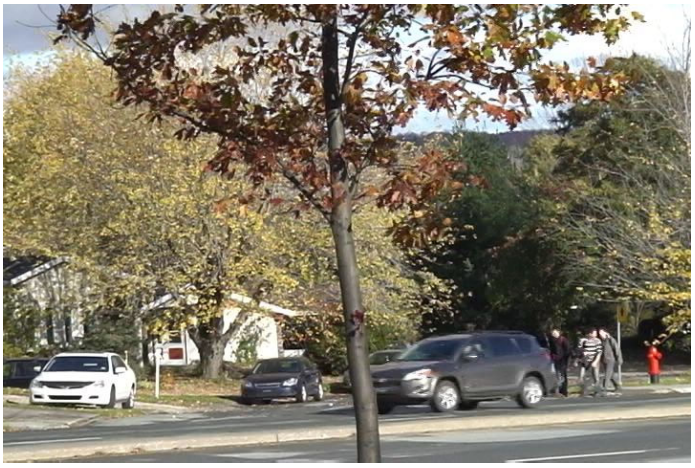

Figure 2: Input frame showing heavy occlusion.

Figure 3: Background frame estimated using grayscale/RGB histograms.

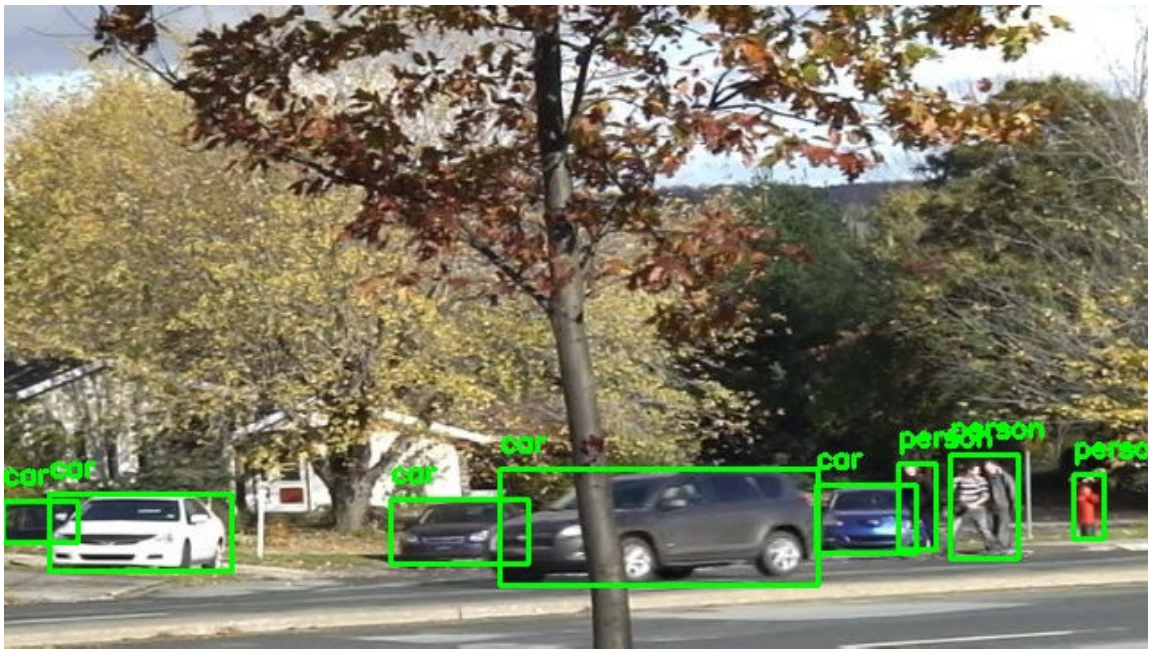

Figure 4: Detection using YOLO directly on input frame (missed static object).

Figure 5: Detection on background frame (static object clearly detected).

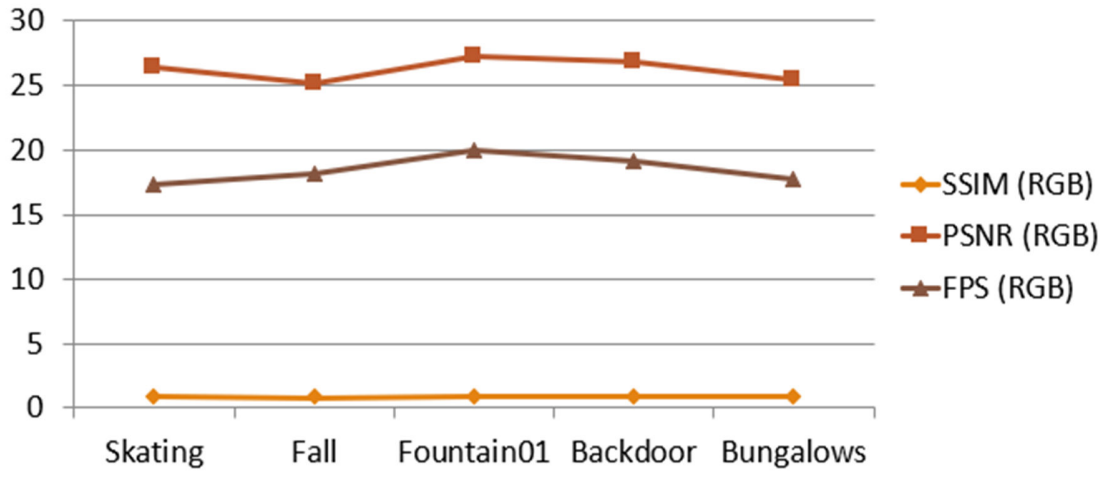

Figure 6: Background Quality Metrics.

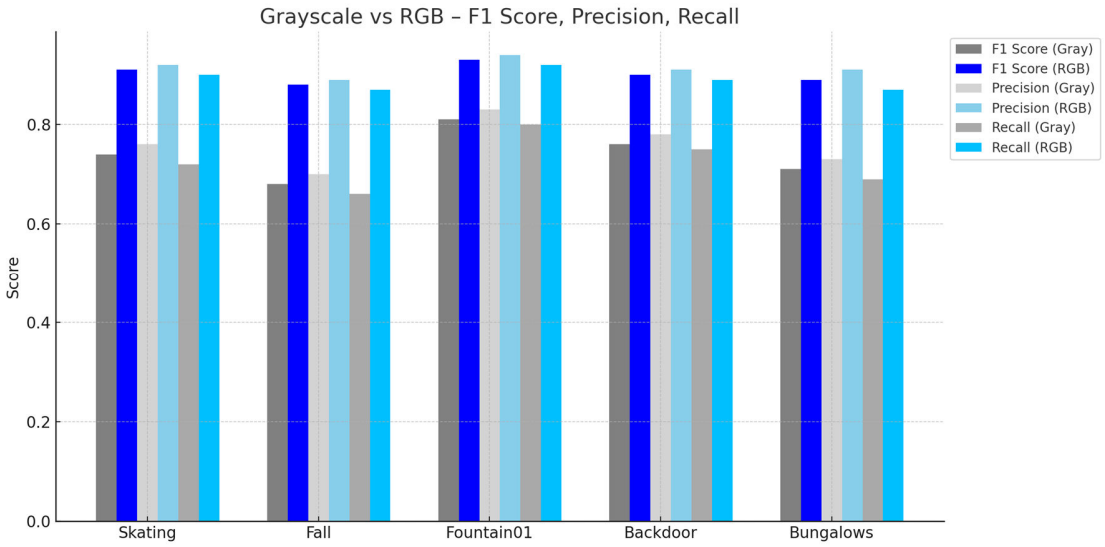

Figure 7: Quantitative Comparison: Grayscale vs RGB.

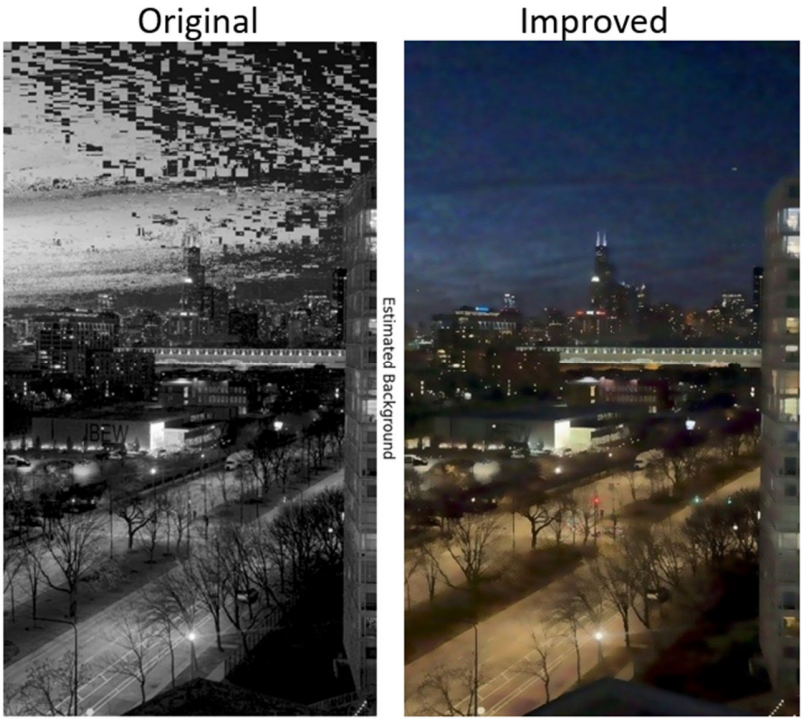

Figure 8: Visual Comparison of Background Estimation Performance.

The left image illustrates the result of the original grayscale-based method, which struggles in dynamic lighting conditions—resulting in visual artifacts, pixel noise, and structural loss in the estimated background. In contrast, the right image shows the output from the improved RGB-based algorithm, which preserves spatial coherence, reduces noise, and maintains realistic illumination and color consistency. This comparison highlights the enhanced robustness of the proposed method under complex urban night-time scenes.

c. Background Quality Metrics

Sequence	SSIM↑	PSNR (dB)↑	FPS↑
Skating	0.882	26.4	17
Fall	0.854	25.1	18
Fountain01	0.901	27.2	20
Backdoor	0.878	26.9	19
Bungalows	0.862	25.5	17

d. Discussion

- On average, the F1 Score improved by over 20%, with both precision and recall increasing across all sequences when using the RGB-based pipeline.
- The visual fidelity of the estimated background also improved, with SSIM values consistently above 0.85 and PSNR in the 25–27 dB range, confirming clearer and more structurally accurate background reconstruction.
- Although the FPS dropped by 2–4 frames on average due to increased computational complexity in the RGB mode, the system maintained real-time performance (≥17 FPS), making it suitable for surveillance and embedded applications.

e. Limitations and Future Improvements

Despite the improvements, a few limitations were observed:

- In sequences with camera shake or abrupt lighting changes, histogram-based methods struggled due to pixel misalignment and inconsistent intensity.
- Shadow boundaries and subtle occlusions can sometimes interfere with color mode estimation.

To mitigate these, future work may incorporate:

- Optical flow stabilization to align frame content temporally before histogram aggregation.

- Adaptive entropy-based thresholding to dynamically adjust sensitivity based on scene variability.
- Hybrid CNN-histogram models to combine data-driven learning with temporal pixel consistency for enhanced generalization.

5. References

[1] H.-J. Kim, M.-C. Shin, M.-W. Han, C.-P. Hong, H.-W. Lee, “An Efficient Scheme to Obtain Background Image in Video for YOLO-Based Static Object Recognition,” Journal of Web Engineering, vol. 21(5), pp. 1691–1706, Aug. 2022.

[2] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv:1804.02767, 2018.

[3] CDNet2014 Dataset. http://changedetection.net

[4] OpenCV Documentation: https://docs.opencv.org

[5] PyTorch Documentation: https://pytorch.org/docs