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

CS512 FINAL PROJECT PRESENTATION

KADHIRAVAN GOPAL | MUKESH SIVA

1. Problem Statement

- Traditional YOLO object detection struggles with static object recognition when moving objects occlude the background.
- Dynamic backgrounds, lighting fluctuations, and scene clutter cause false detections and missed objects.
- The aim is to extract a clean background using histogram-based analysis and apply YOLO on this refined image.
- Result: Improved detection of static objects in complex scenes.

2. Steps to follow

Video Input →
Frame Extraction →
Frame Buffering →
Histogram Analysis →
Background Estimation →
YOLO Detection →
Metric Evaluation

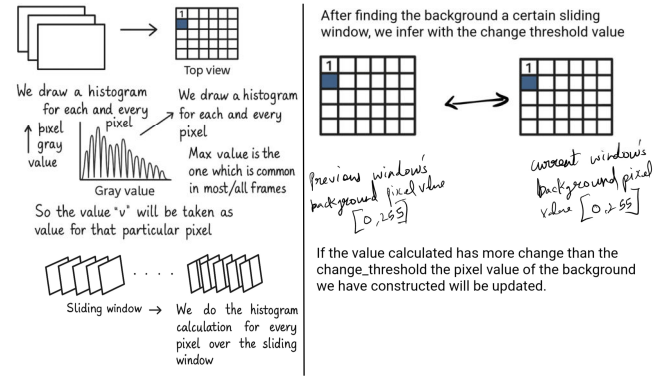
3. Dataset: CDNet2014

- Public benchmark dataset for background subtraction and change detection.
- Sequences used:
  - Skating (badWeather), Fall, Fountain01 (dynamicBackground)
  - Backdoor, Bungalows (shadow)
- Ground truth provided as pixel-wise masks for evaluation.
- Used to generate bounding boxes and compute detection metrics.

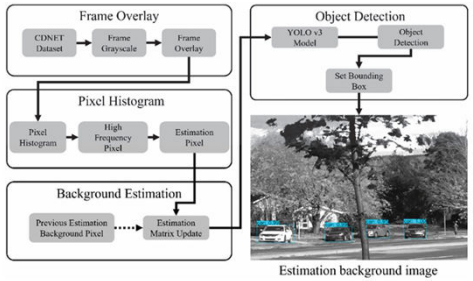
4. Proposed Solution

- Estimate background using temporal histograms of pixel values.
- Original version used grayscale histograms.
- Improved version used RGB histograms with dynamic thresholds.
- Run YOLOv3 on estimated background to detect static objects with higher precision.

4. Histogram Analysis



5. Background Estimation Logic



Estimation background image

6.Pseudocode

```
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
    dominant ← mode(histogram_p)
    If |B(p) - dominant| > τ:
      B(p) ← dominant

Return B as estimated background
```

7. Implementation

- 1. Install Dependencies:
- • Python 3.8+, pip install opencv-python torch numpy
- 2. Prepare Dataset:
- • Place custom videos in 'data_video_frames/'
- • Or use CDNet2014: archive/dataset/<category>/<sequence>/input/
- 3. Run Main Script:
- • Execute 'gpu_code_improvised_algo.py'
- • Background is estimated and YOLO detection is performed
- 4. Evaluate Results:
- • Bounding boxes saved as .txt files
- • Metrics (F1, Precision, Recall, SSIM, PSNR, FPS) are printed
- 5. Optional:
- • Tweak thresholds, sequences, or export settings in script

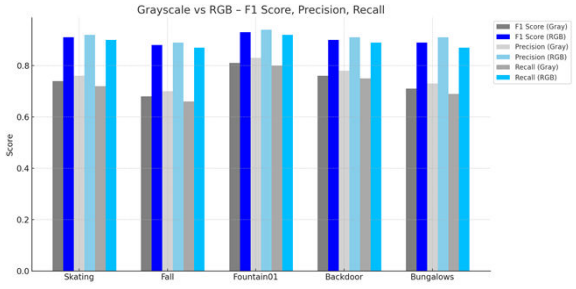
8. Evaluation Metrics

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

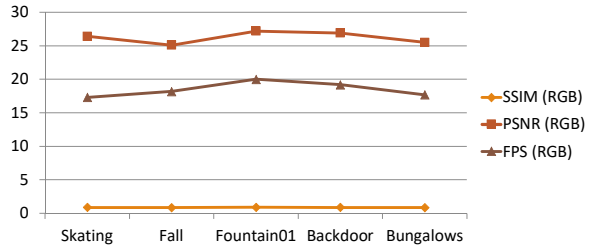
9. Results Summary (RGB vs Grayscale)

- RGB algorithm improved F1 Score by up to 20%.
- SSIM consistently > 0.85, PSNR 25–27 dB.
- FPS dropped slightly (~2–4 FPS) but still real-time (>17 FPS).
- Significant gains in object detection precision and robustness.

10. Quantitative Comparison: Grayscale vs RGB

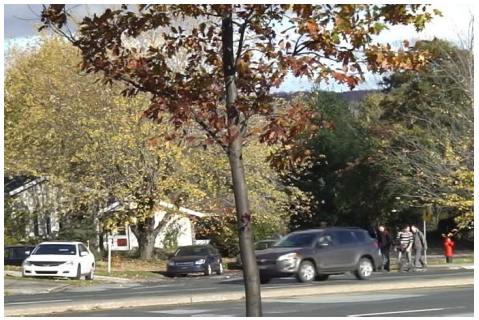


11. Background Quality Metrics



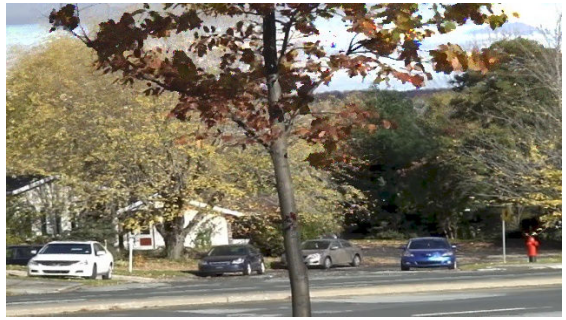
12. Visual Comparison

- Figure 1: Input frame (with occlusion)



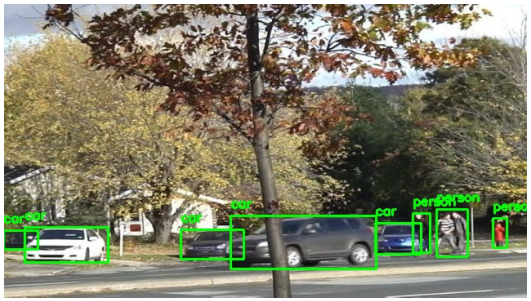
12. Visual Comparison

- Figure 2: Estimated background (RGB)



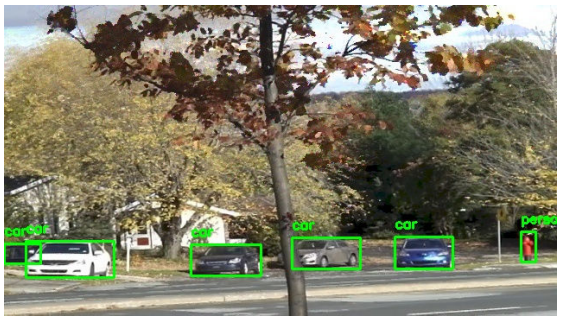
12. Visual Comparison

- Figure 3: YOLO detection on input (missed)

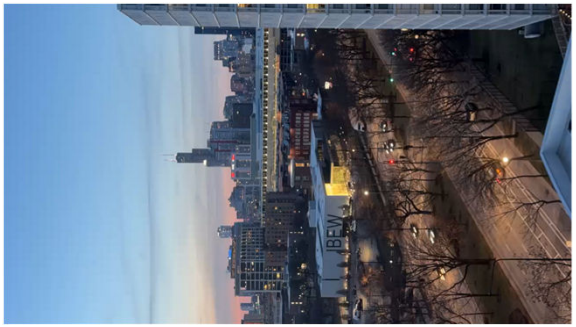


12. Visual Comparison

- Figure 4: YOLO detection on background (detected)

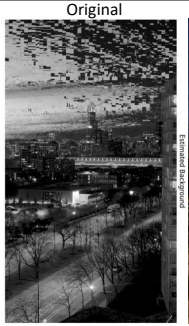


12.Dynamic lighting data



12.Original vs Improved

Original



Improved

As see from the visual comparison our new method shows significant improvement in estimating background in dynamic light conditions.

13. Limitations and Future Work

- Struggles under camera shake or abrupt lighting.
- Cannot handle all forms of shadows or occlusions.

Future Enhancements:

- Optical flow-based stabilization.
- Adaptive entropy thresholding.
- Combine CNN features with histogram background model.

14. Conclusion

The RGB-improved background estimation significantly enhanced object detection accuracy in video sequences.

- Our method reduced undetectable objects and improved F1 scores by up to 20%.
- Background quality (SSIM and PSNR) remained high while maintaining real-time performance (>17 FPS).
- Histogram-based RGB analysis proved robust for static object recognition under diverse conditions.
- The approach is scalable and suitable for practical applications such as surveillance and robotics.

15. References

[1] H.-J. Kim et al., Journal of Web Engineering, 2022.

[2] J. Redmon and A. Farhadi, YOLOv3, arXiv:1804.02767.

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

[4] OpenCV Documentation

[5] PyTorch Documentation