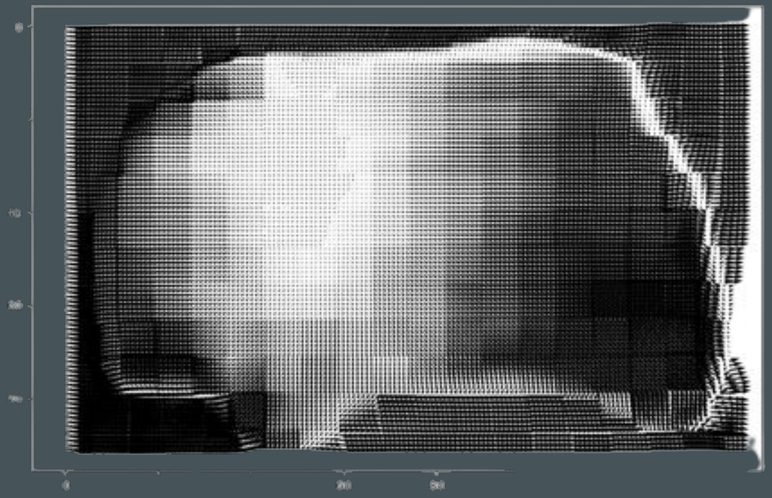
VEHICLE SPEED CALCULATION USING OPTICAL FLOW

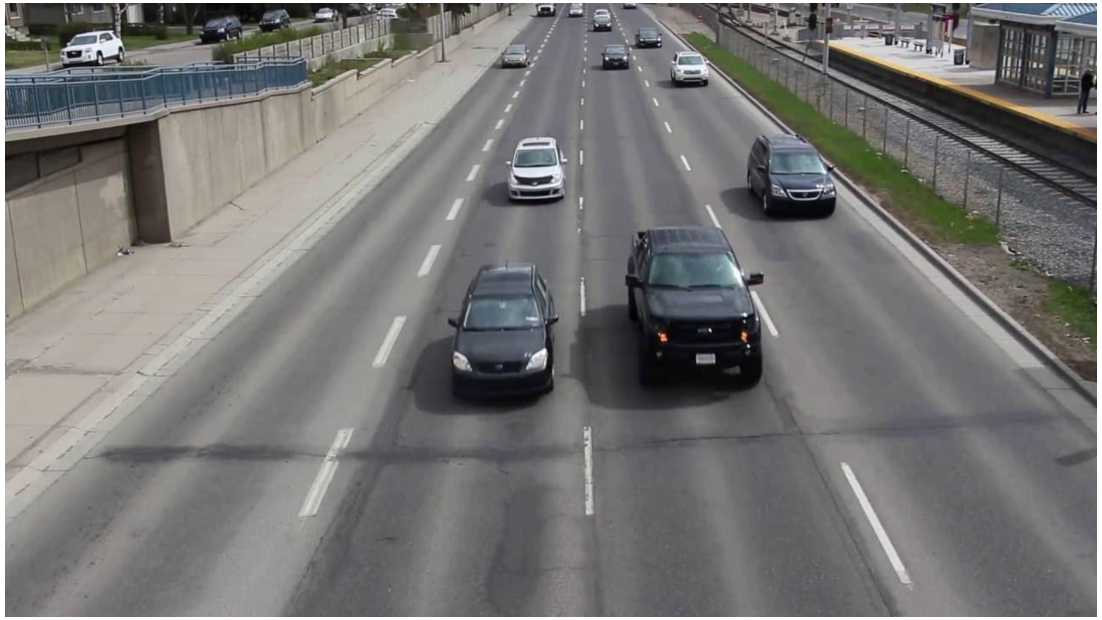
BY ~
NISHANT PANDEY
AMOGH WYAWAHARE
JOSEPH THOMAS
JAYASURIYA SURESH
RISHIKESH JADHAV



GROUP 13

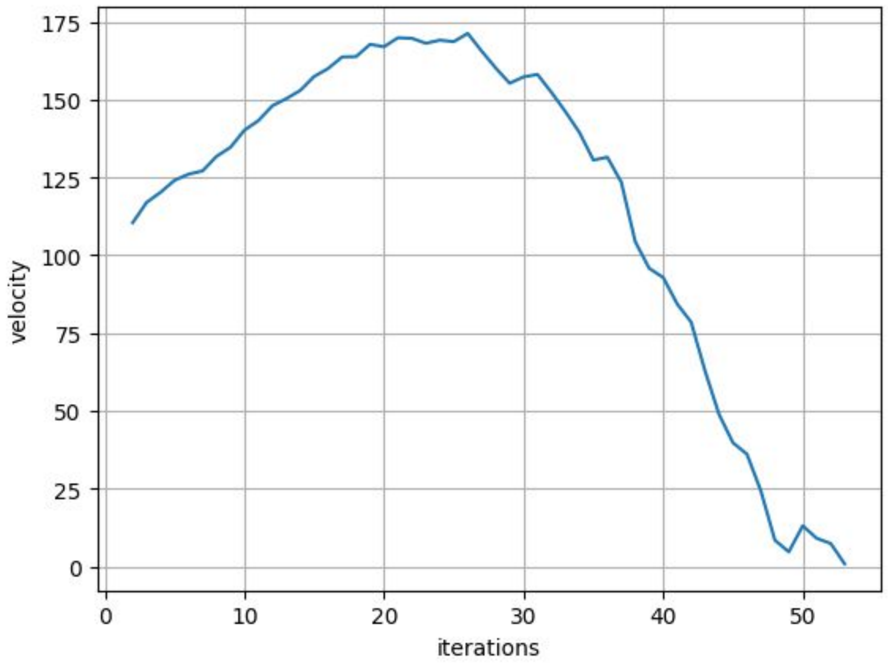
INTRODUCTION

- Estimating the accurate speed of vehicles is very important in the automotive industry to improve efficiency in road connectivity and to increase road safety.
- Optical flow is a promising approach to calculate the vehicle speed using video data from onboard cameras.
- Three different algorithms are utilized for the estimation of speed – Lucas Kanade Algorithm, Farneback Gunnar Algorithm and finally RAFT which is short for Recurrent All-Pairs Field Transforms



LUKAS-KANADE

- Lukas-Kanade is an optical flow algorithm that estimates the motion between two frames of an image sequence by tracking a set of feature points.



Input: Image frames I_1, I_2
Output: Velocity vector $v = (u, v)$

foreach *pixel* (x, y) *in* I_1 **do**

$I_x(x,y) \leftarrow \frac{1}{2}(I_1(x+1,y) - I_1(x-1,y))$;
$I_y(x,y) \leftarrow \frac{1}{2}(I_1(x,y+1) - I_1(x,y-1))$;
$I_t(x,y) \leftarrow I_2(x,y) - I_1(x,y)$;

end

foreach *pixel* (x, y) *in a window around each feature point* (x_0, y_0) **do**

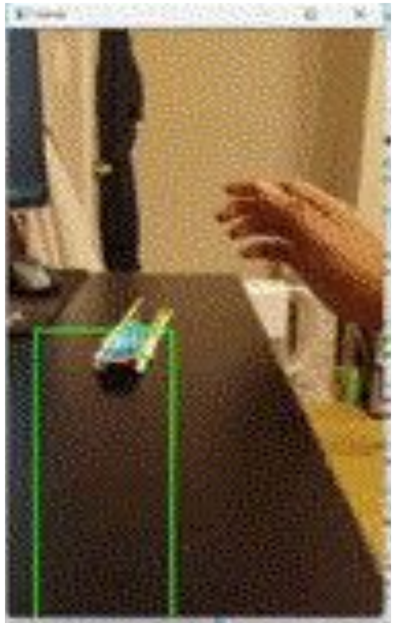
$A \leftarrow I_x(x_0,y_0)I_y(x_0,y_0)$; $b \leftarrow -I_t(x_0,y_0)$; $v \leftarrow (u,v) \leftarrow (0,0)$;
foreach *pixel* (x, y) *in the window* **do**
$A \leftarrow A\ I_x(x,y)I_y(x,y)$; $b \leftarrow b\ - I_t(x,y)$;
end
$v \leftarrow (A^T A)^{-1} A^T b$;

end
return v;

Algorithm 1: Lucas-Kanade Optical Flow

FARNEBACK GUNNAR

- The Farneback Gunnar is a dense optical flow algorithm that estimates the optical flow for every pixel in the image.



Input: Previous frame I_{t-1}, current frame I_t, number of pyramid layers n, size of window w, regularization parameter λ

Output: Optical flow u, v

$I_{t-1}, I_t \leftarrow$ grayscale(I_{t-1}), grayscale(I_t); $G_x, G_y \leftarrow$ compute image gradients of I_t; u,$v \leftarrow$ initialize flow vectors to 0; **for** $i = n$ *down to 1* **do**

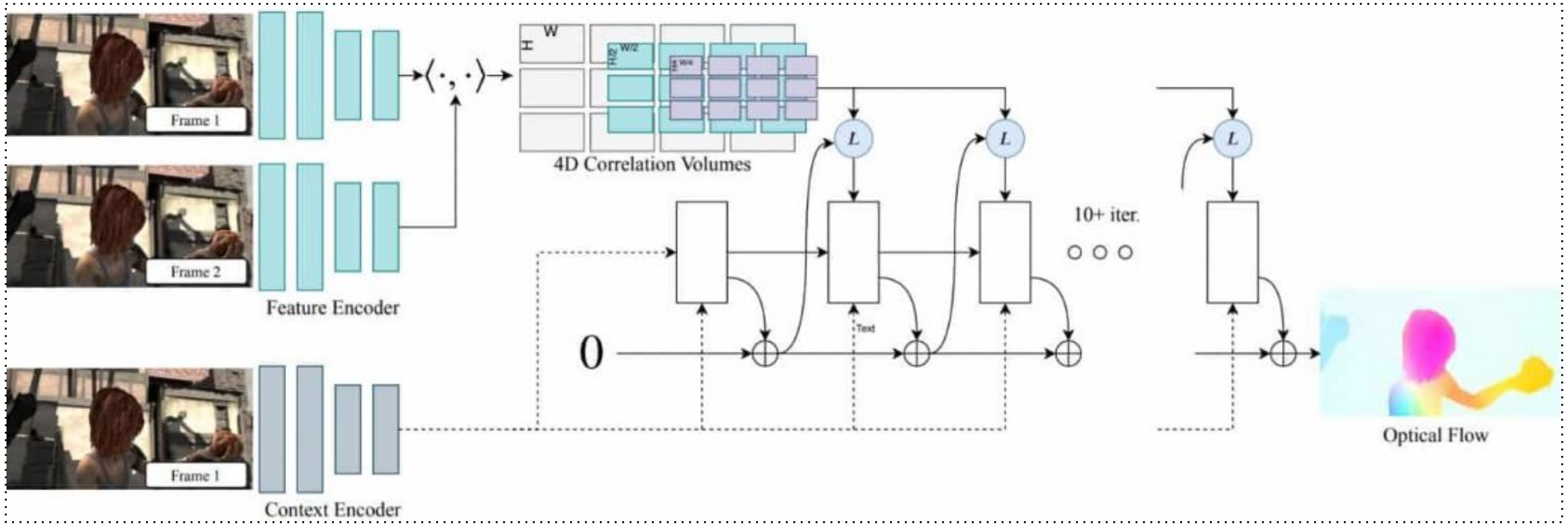
$I^i_{t-1}, I^i_t \leftarrow$ downsample(I_{t-1},I_t) to level i; u^i, $v^i \leftarrow$ resize(u, v) to level i; G^i_x, $G^i_y \leftarrow$ downsample(G_x,G_y) to level i; A, B, C, D, E, F $\leftarrow$ compute Farneback polynomial terms using I^i_{t-1}, I^i_t, G^i_x, G^i_y, u^i, v^i, w, λ; $solve(Au^i + Bv^i + C, Du^i + Ev^i + F) \leftarrow$ solve for flow vectors u^i, v^i using polynomial terms; u, $v \leftarrow$ resize(u^i, v^i) to level 0;

end

return u, v

Algorithm 2: Farneback Optical Flow

RAFT

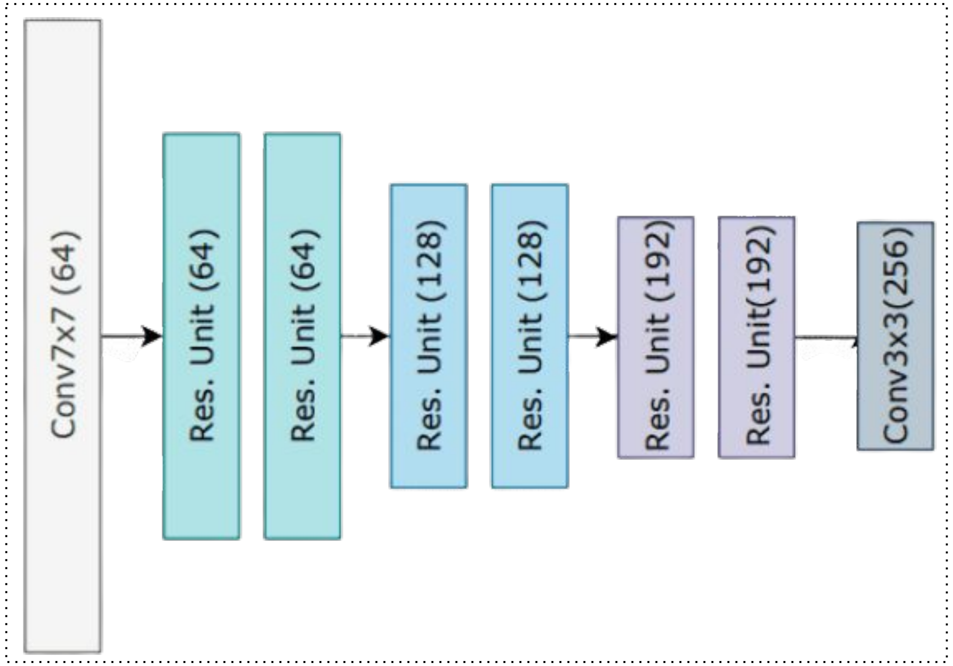


- The Recurrent All-Pairs Field Transforms is a deep network architecture for optical flow. It can produce multi-scale 4D correlation volumes for every pair of pixels, extract per-pixel characteristics and repeatedly update a flow field using a recurrent unit that searches the correlation volumes.

RAFT ARCHITECTURE

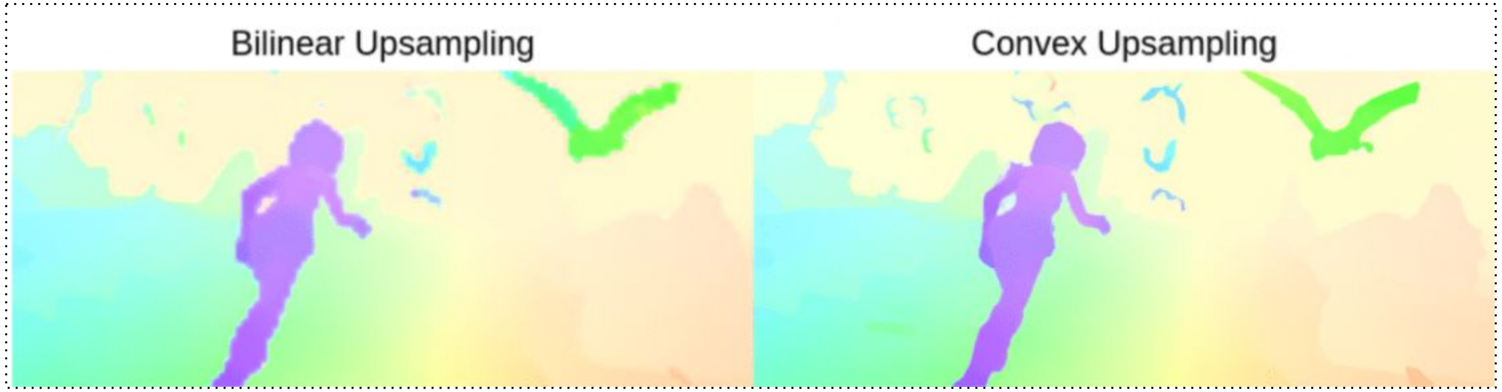
RAFT can be divided into three stages:

1. Feature Extractors
2. Visual Similarity
3. Iterative Updates



UPSAMPLING MODULE

- Two different approaches are used for upsampling the optical flow output of a GRU cell.
- The first method is called bilinear interpolation, which is a simple and fast technique but may not produce the best quality results. It essentially computes the values of new pixels by taking weighted averages of neighboring pixels.
- The second approach is called Convex Upsampling, which is a learned upsample module.



ADVANTAGES OF USING RAFT OVER OTHER ALGORITHMS

- Highly efficient – inference time, training speed and parameter count.
- Excellent cross-data generalization
- There are some open-source Optical Flow datasets and benchmarks that depict the predominance of Deep Learning solutions over the classical ones in terms of quality and inference time. Regarding the RAFT architecture, it took the first place in SINTEL benchmark at the time when it came out.

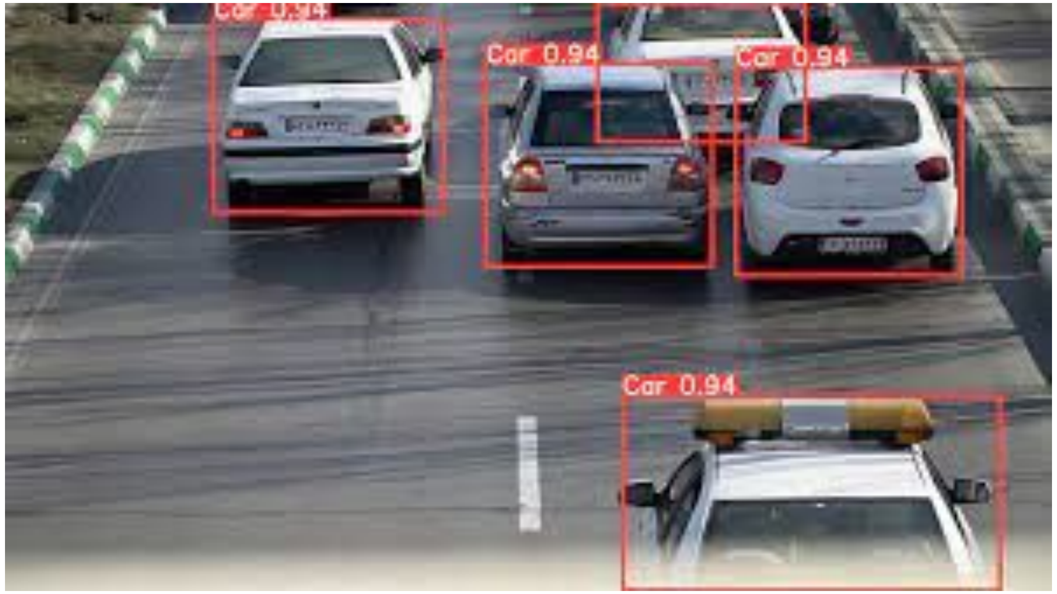
YOLO

- You only look once is a real-time object detection system. It applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region that are weighted by the predicted probabilities.

HOW YOLO WORKS

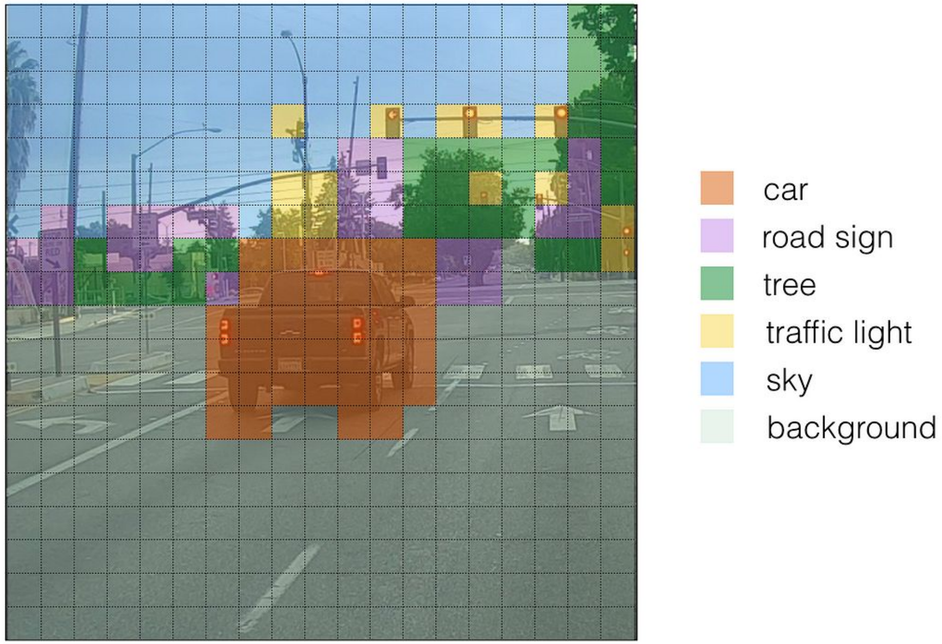
Uses the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)



FINDING THE REGION OF INTEREST USING YOLO

- Pre-processed and annotated images are used to train the model.
- A pre-processed image is passed through a deep convolutional network, non-max suppression is applied, and the detected objects are outputted with a bounding box around them.
- Only one box is selected when several boxes overlap with each other and detect the same object. A filter by thresholding is applied to do this.

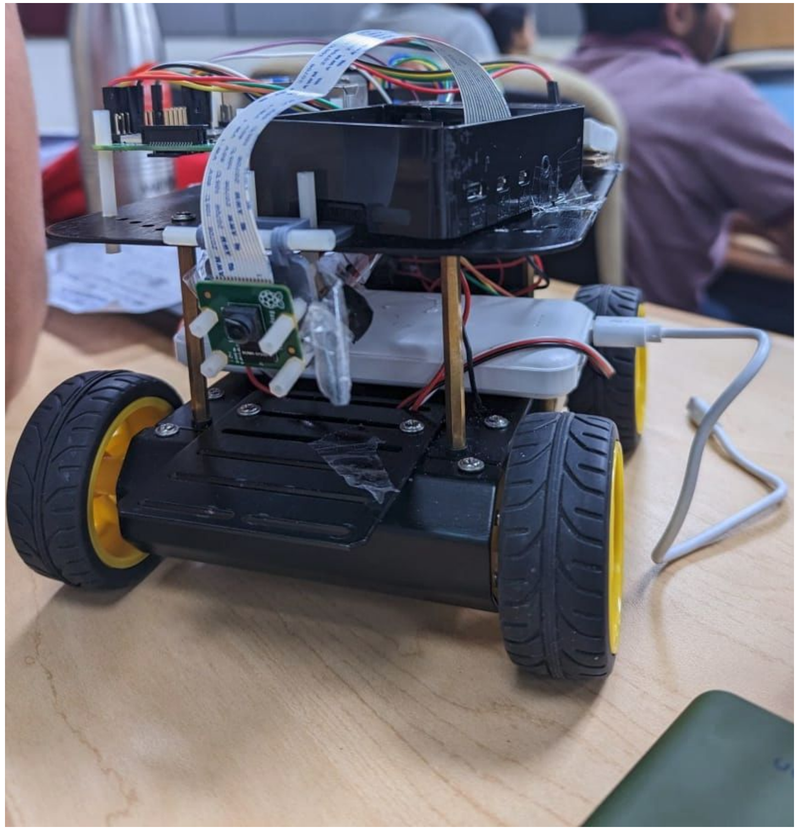


WHY YOLO ?

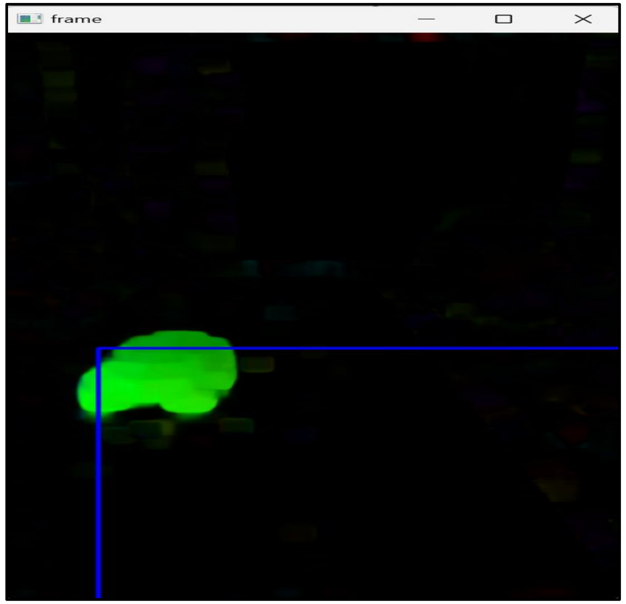
- Speed: This algorithm improves the speed of detection because it can predict objects in real-time.
- High accuracy: YOLO is a predictive technique that provides accurate results with minimal background errors.
- Learning capabilities: The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

Hardware

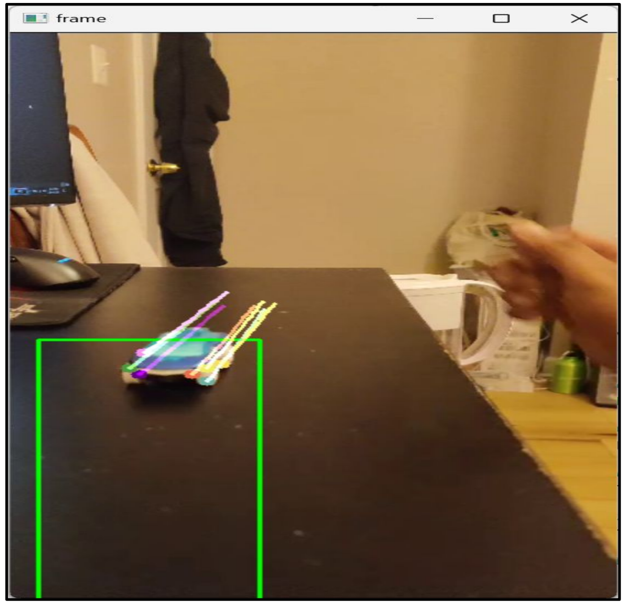
- Raspberry Pi camera V2.1 was used.
- Video was recorded at 480P , 30FPS.
- Classical Optical methods were ran on Raspberry Pi 4.
- Not a very good camera for this purpose.



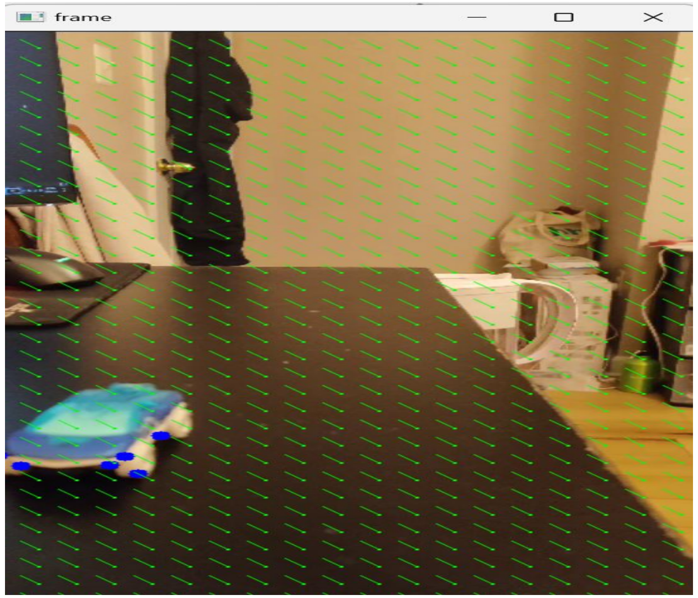
LUCAS KANADE AND FARNEBACK RESULTS



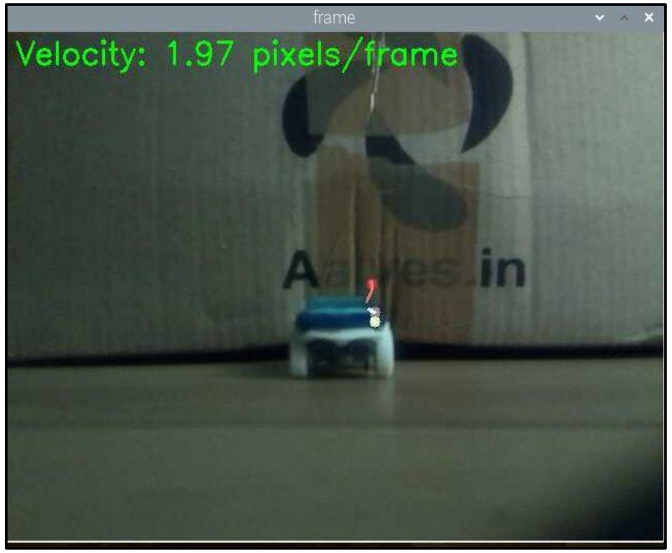
Optical Flow



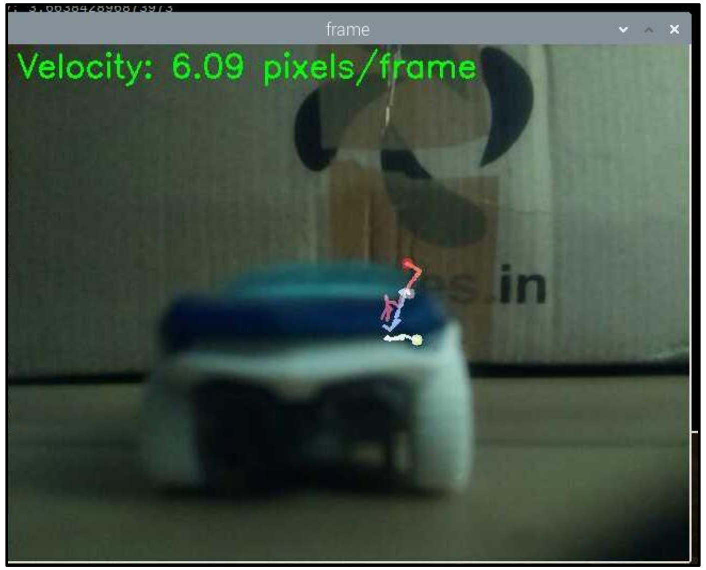
Feature Points



Optical Shade

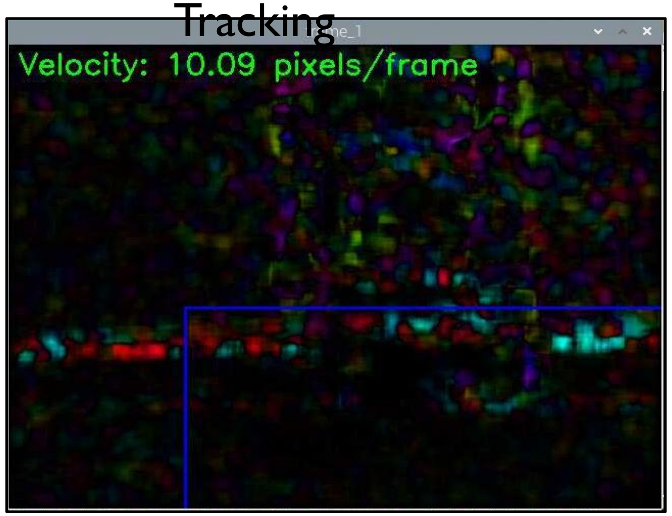


Feature Extraction



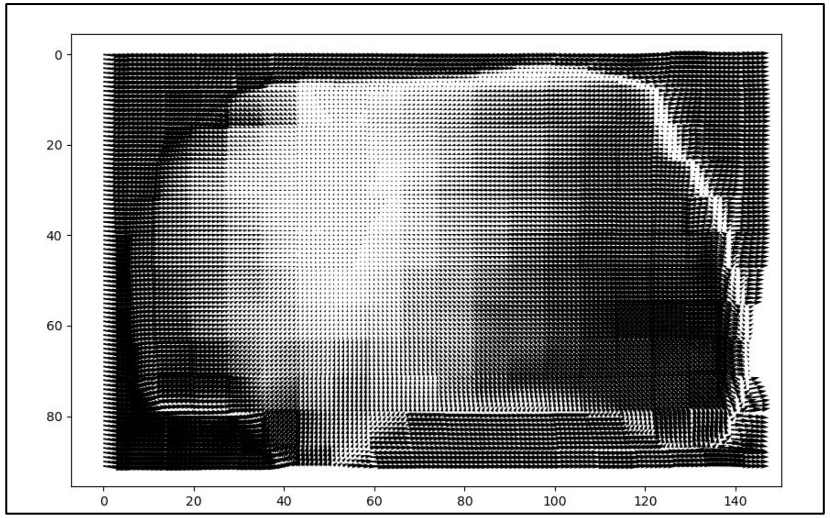
Feature Tracking

Optical Shade



Optical Flow

RAFT RESULTS - Using frames from test video



Optical Shade

Optical Flow

RAFT RESULTS - LIVE TESTING

https://drive.google.com/file/d/1vhBZ9c4gu8jtySNnI1wsDmVi3Eo-FCLR/view?usp=sharing
https://drive.google.com/file/d/17wc-j3qBRNW92vrLINJXw5YU8ts39wP5/view?usp=sharing

Future Work

- Making a leader-follower implementation using calculated velocity values.
- Improving the accuracy of velocity calculations.
- Modify RAFT to determine the velocity without needing of additional processing of the output.
- Determining velocity of multiple cars in a single video.

REFERENCES

- Andreas Geiger and Philip Lenz and Raquel Urtasun, Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite, {Conference on Computer Vision and Pattern Recognition (CVPR), 2012, https://www.kaggle.com/datasets/klemenko/kitti-dataset
- N. Sharmin and R. Brad, "Optimal Filter Estimation for Lucas-Kanade Optical Flow," Sensors, vol. 12, no. 9, pp. 12694–12709, Sep. 2012, doi: https://doi.org/10.3390/s120912694.
- Wu, Zhao, Gan, and Ma, "Measuring Surface Velocity of Water Flow by Dense Optical Flow Method," Water, vol. 11, no. 11, p. 2320, Nov. 2019, doi: https://doi.org/10.3390/w11112320.
- A. Dosovitskiy et al., "FlowNet: Learning Optical Flow with Convolutional Networks." Accessed: May 11, 2023. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2015/papers/Dosovitskiy_FlowNet_Learning_Optical_ICCV_2015_paper.pdf
- A. M. Mathew and T. Khalid, "Ego Vehicle Speed Estimation using 3D Convolution with Masked Attention," arXiv:2212.05432 [cs], Dec. 2022, Accessed: May 11, 2023. [Online]. Available: https://arxiv.org/abs/2212.05432
- A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras," Robotics: Science and Systems XIV, Jun. 2018, doi: https://doi.org/10.15607/RSS.2018.XIV.062.
- https://learnopencv.com/optical-flow-using-deep-learning-raft/#raft
- arXiv:2003.12039
- https://github.com/princeton-vl/RAFT
- https://github.com/pjreddie/darknet
- https://www.cvlibs.net/datasets/kitti/
- https://drive.google.com/drive/folders/1a-v4os2Ekr-IezLE-pGNJ7R0pIZyf6bE