Real-Time Object Detection at the Edge

Project Introduction and Goals

Colin Burdine

June 14, 2021

SULI Intern, Hosted at Argonne National Laboratory

Introduction

About Myself

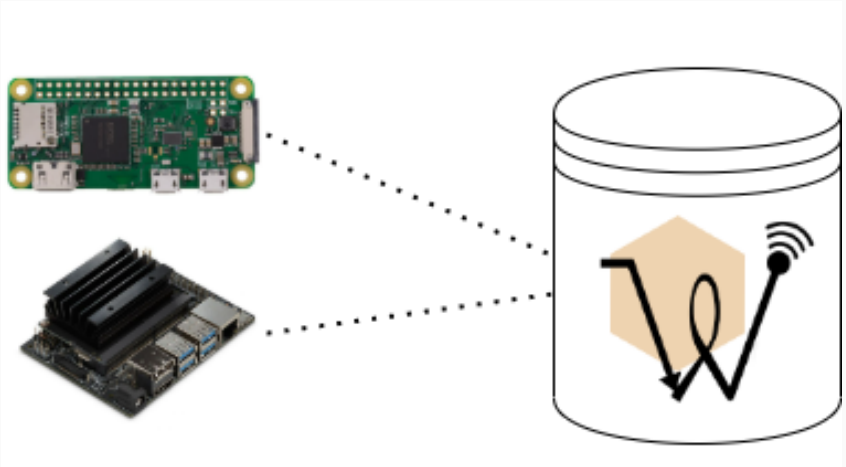
- I am a recent graduate of Baylor University (Sic 'Em Bears!)
- I majored in Mathematics and Computer Science.
- I am currently working on my Master's at Baylor, doing research in the field of quantum computing.

Problem Statement

- Over the past three to four decades, advances in computer vision systems and machine learning have enabled high-fidelity object detection, classification, and tracking.
- However, these computer vision models and methods are often quite complex and perform poorly on small computing devices.

Problem Statement

- Over the past three to four decades, advances in computer vision systems and machine learning have enabled high-fidelity object detection, classification, and tracking.
- However, these computer vision models and methods are often quite complex and perform poorly on small computing devices.
- This poses a challenge if we want to integrate these methods into an edge computing system like Waggle:



Problem Statement

- This summer, I will be working on a general-purpose plugin for Waggle that is capable of real-time detection of moving objects.
- There are a few constraints I will be operating within when developing this plugin:

Requirements and Constraints

1. Must be able to detect movement of reasonably large objects, but not background movements (e.g. waving branches).
2. Must be able to detect multiple objects at once and identify their bounding box in the scene.
3. Must be able to optimize the system across a trade-off of resource utilization versus accuracy.

Proposed Methods

Proposed Methods

- I have already written a prototype motion detector system that uses a few different methods of varying complexity.
- These methods use OpenCV's Python3 API and Tensorflow2.
- In the plugin, these methods are realized as Python classes that conform to a common interface.

Proposed Methods

- I have already written a prototype motion detector system that uses a few different methods of varying complexity.
- These methods use OpenCV's Python3 API and Tensorflow2.
- In the plugin, these methods are realized as Python classes that conform to a common interface.

Motion Detector Methods

1. Background Subtraction (via K-nearest Neighbors [ZvdH06])
2. Dense Optical Flow (Farnebäck's method [Far03])
3. TinyYOLOv2 (CNN-based object classifier [RDGF16]).

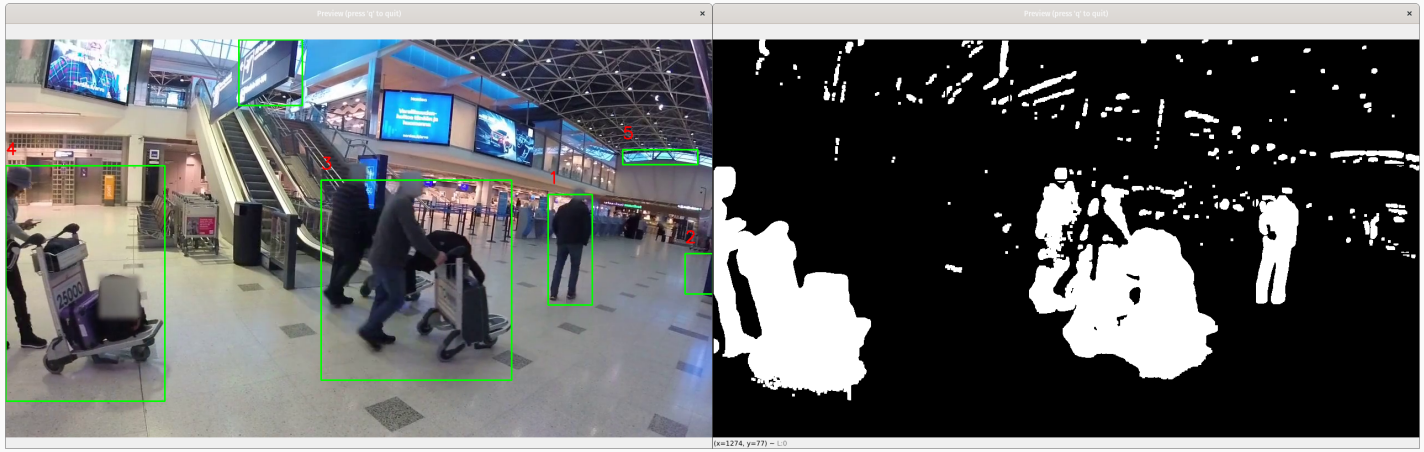
Background Subtraction

- Background Subtraction is performed by constructing a probability distribution representing the background pixels of an image [ZvdH06].

  There are several implementations available in OpenCV:

  1. Gaussian Mixture Model
  2. Kernel Density Estimation
  3. K-Nearest Neighbors

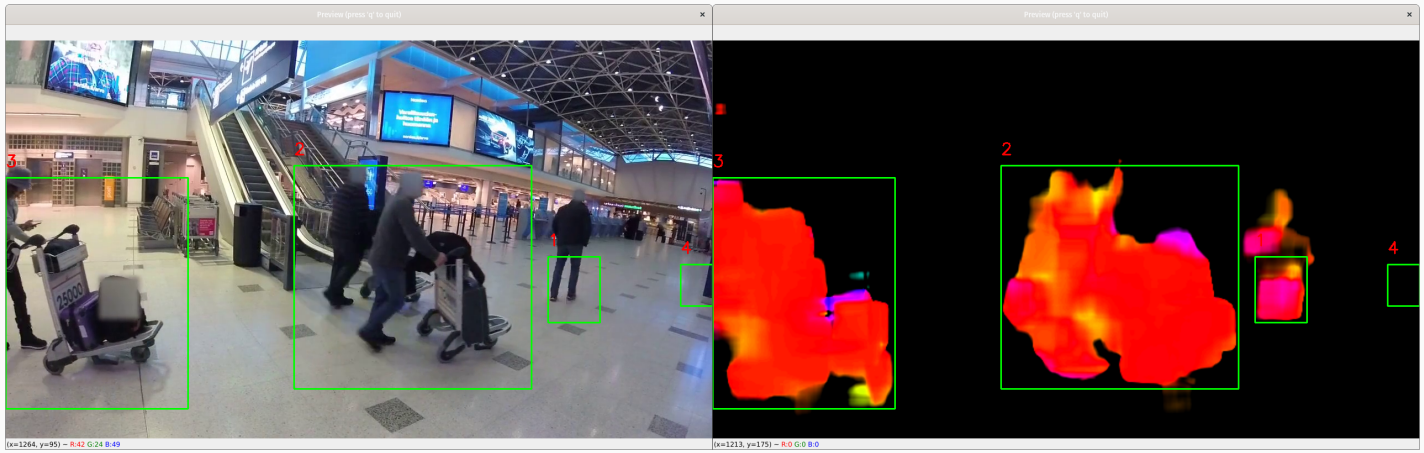
Example (TAU Urban A/V Scenes Dataset [WMHV21])



Dense Optical Flow

- In addition to where motion is occurring, Dense Optical Flow estimates the direction and velocity of motion relative to the 2D image.
- Optical Flow is more computationally demanding, but can handle occluded objects with different trajectories.
- Optical Flow can be computed efficiently through Farnebäck's method [?].

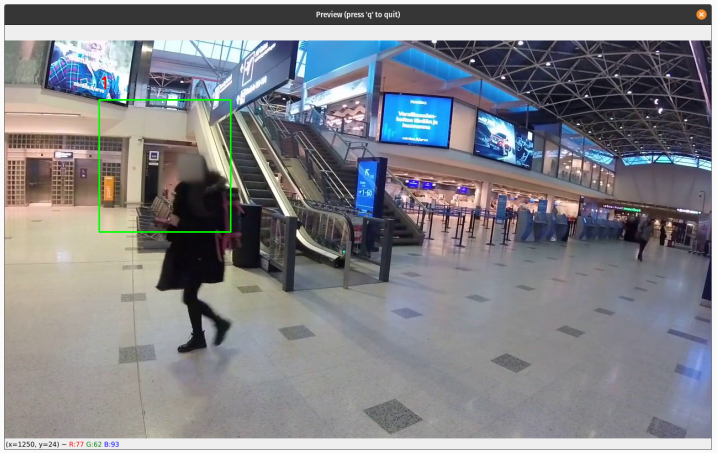
Example (TAU Urban A/V Scenes Dataset [WMHV21])



TinyYOLO (CNN Object Detector)

- The YOLO Model ("*You Only Look Once*") is a Convolution Neural Network (CNN) that can identify and classify objects in a scene.
- It is computationally demanding, but can be trained to identify only moving objects that are of interest in the scene.
- The plugin prototype currently uses the pre-trained "TinyYOLOv2" network, which has 9 layers, and over 15,000,000 trainable parameters.

Example (TAU Urban A/V Scenes Dataset [WMHV21])



Goals and Future Work

Goals

My Goals for this internship are the following:

1. I have little experience with computer vision, so I want to learn as much as possible about some of tools used in this field (OpenCV, etc.)
2. I'd like to produce a usable final product that will contribute to the Waggle project in a meaningful way.
3. I'd like to learn more about the mathematics of computer vision.
4. I want to meet some great people at Argonne and learn as much as I can from them.
5. I also want to be a great person to work with.

Future Work

My current priority list for my Waggle plugin project is the following:

1. Integrate the plugin prototype with PyWaggle so that it can publish object data to other plugins.
2. Begin work on more thorough documentation of the plugin.
3. Explore some ways of improving upon the aforementioned methods.
4. Allow for the plugin to import custom-trained object detection models.
5. Implement any suggestions and feedback from my mentors!

Questions?

My Daily Progress Log:
https://github.com/waggle-sensor/summer2021/tree/main/Burdine

Motion Detector Plugin:

https://github.com/waggle-sensor/plugin-motion-detector

References i

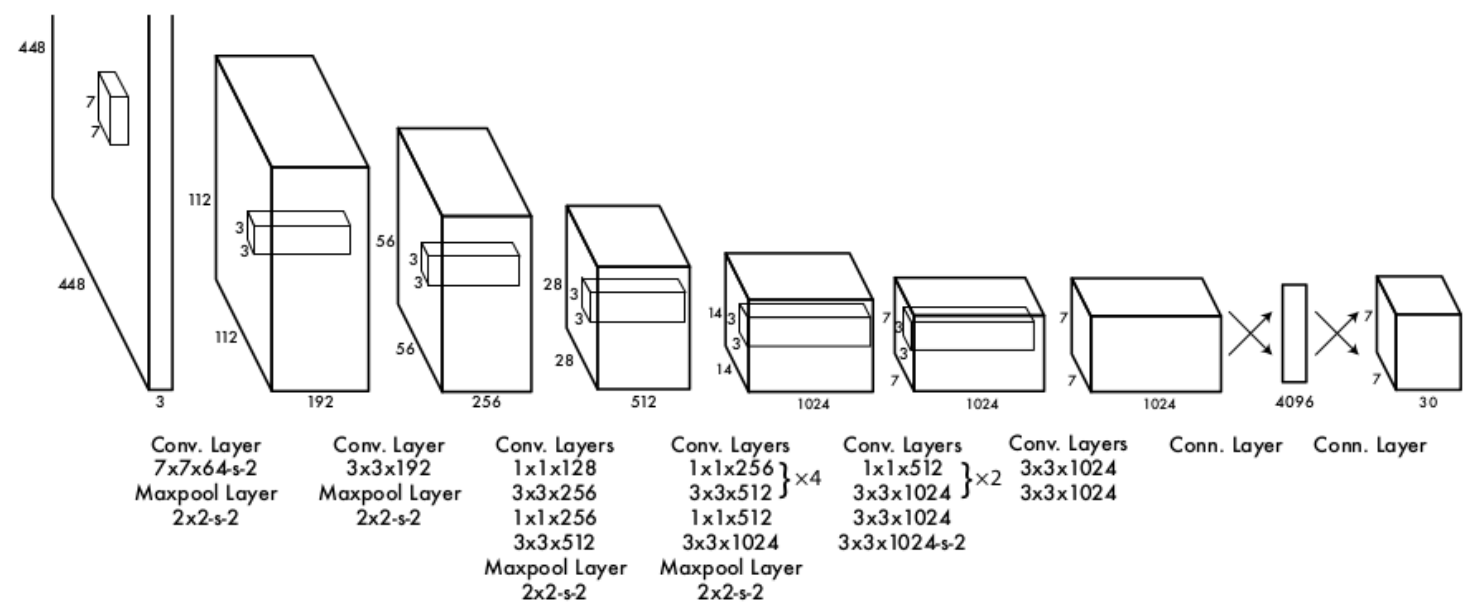
- Gunnar Farnebäck, *Two-frame motion estimation based on polynomial expansion*, vol. 2749, 06 2003, pp. 363–370.
- Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, *You only look once: Unified, real-time object detection*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
- Shanshan Wang, Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen, *A Curated Dataset of Urban Scenes for Audio-Visual Scene Analysis*, ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Toronto, ON, Canada), IEEE, June 2021, pp. 626–630.

References ii


Zoran Zivkovic and Ferdinand van der Heijden, *Efficient adaptive density estimation per image pixel for the task of background subtraction*, Pattern Recognition Letters **27** (2006), no. 7, 773–780 (en).

Appendix

YOLOv1 Architecture



(This image is from the original YOLO paper [RDGF16])