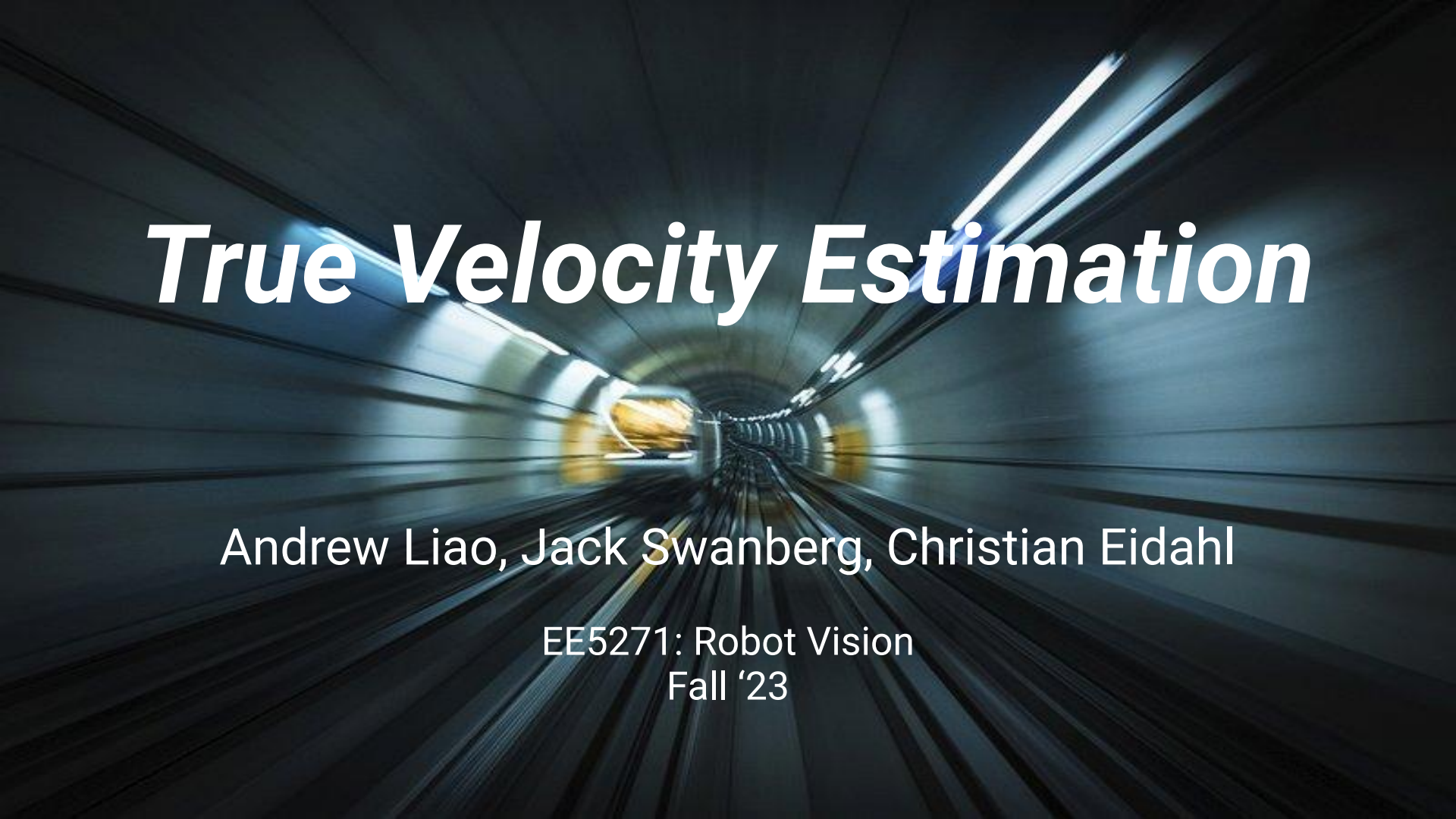


True Velocity Estimation

Andrew Liao, Jack Swanberg, Christian Eidahl

EE5271: Robot Vision
Fall ‘23

Project Goal

Utilize depth information gained from stereo vision along with image plane location gathered using object detection and instance segmentation to estimate velocity of an object.

Methodology

To get x-axis, or side-to-side, movement we used a CNN-based computer vision model, YOLOv8, to do object tracking and segmentation

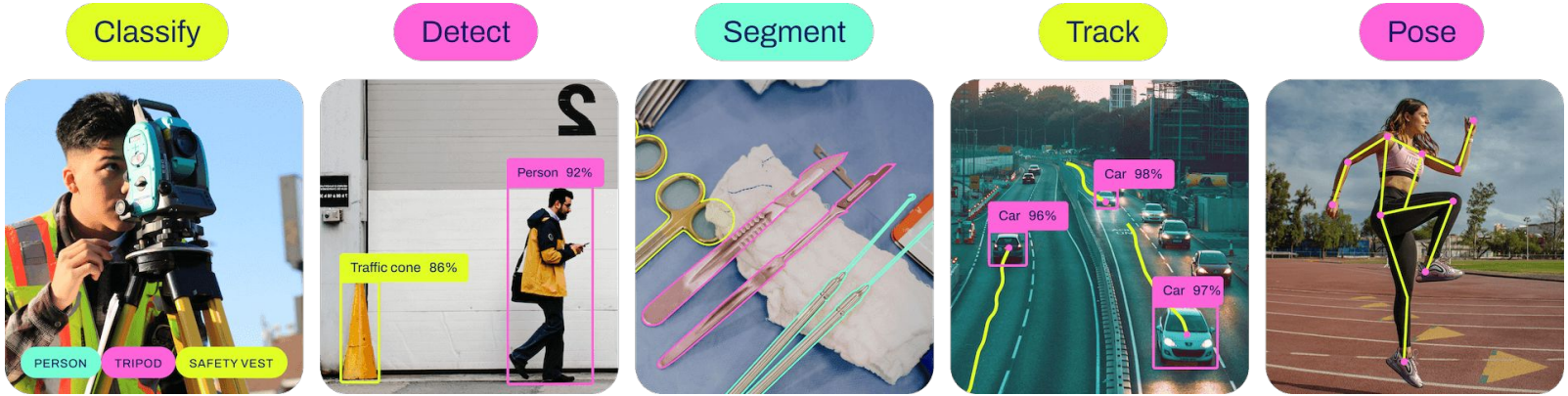
We use a sequence of frames to calculate “instantaneous” velocity, and average velocity.

We use the x position data and knowledge of frames rate to calculate speed

Object Tracking & Segmentation

YOLOv8: the latest iteration of the popular YOLO family of computer vision algorithms

+ Runs quickly, easy to use
+ CNN-based architecture
+ Many capabilities



Hardware Setup

Stereo system using two DSLR cameras

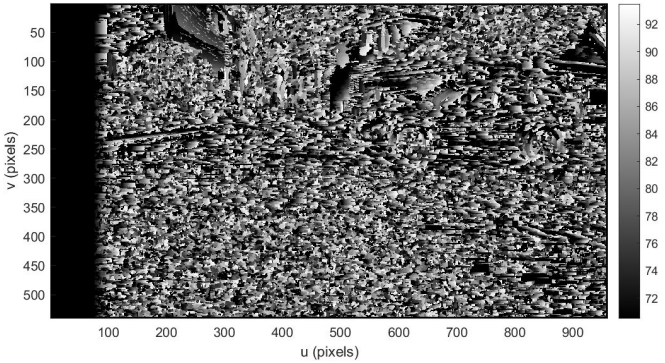
Baseline set to 20 inches

Both cameras shooting 1920x1080 at 60 fps

Cameras were unsynchronized

Problems and Solutions

Input images



Bad disparity map!

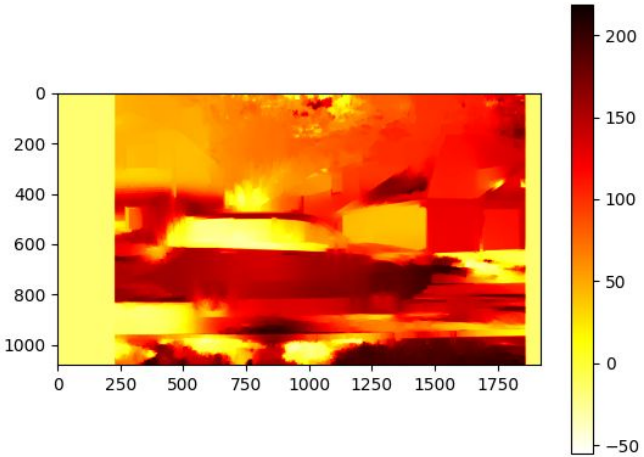
Initial disparity images were unusable so several methods were used to try and improve the output

- Calibration
- Stereo image rectification: warping original images/frames to account for misalignment, distortions, etc.
- Disparity filter (Weighted Least Squares filter)
- Tuning parameters (e.g. template size, disparity range, etc.)
- Segmentation mask: taking the average disparity over the segmented object

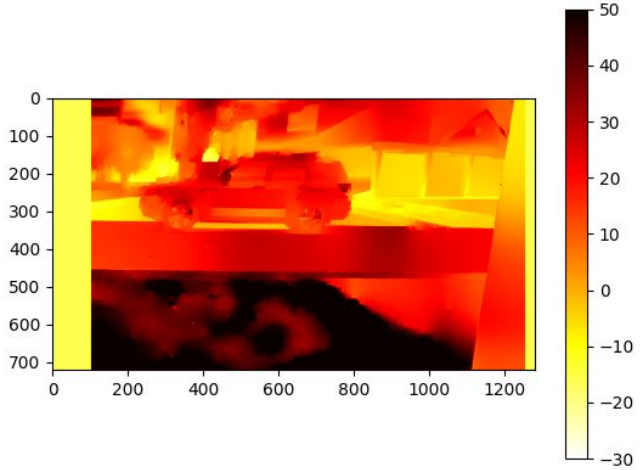
Stereo Rectification

Two approaches to calculating rectification transformation:

1) Calibrated: Checkerboard used to find camera intrinsics/extrinsics
2) Uncalibrated: uses feature matching e.g. SIFT + nearest neighbors



Calibrated



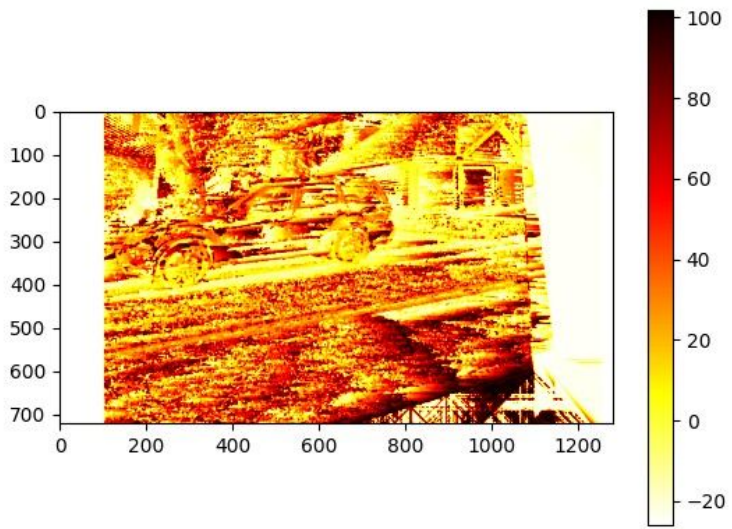
Uncalibrated

Rectification Results

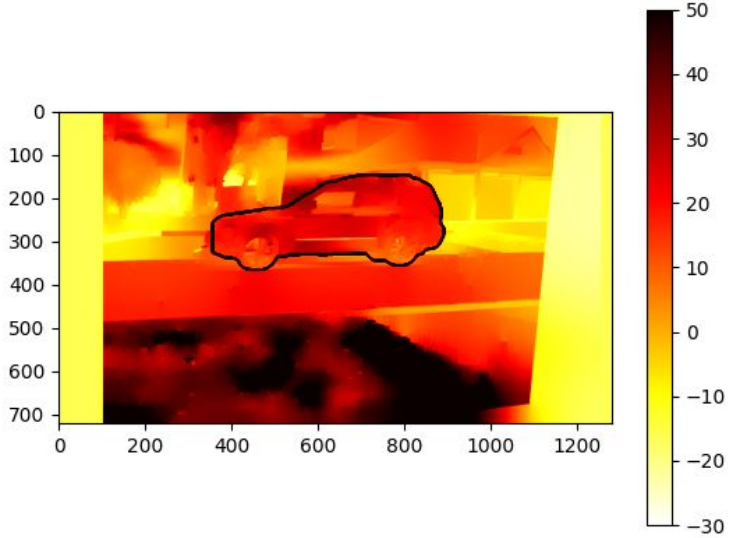
Pre-rectification

Post-rectification

Disparity filter, segmentation mask, parameter tuning, etc.

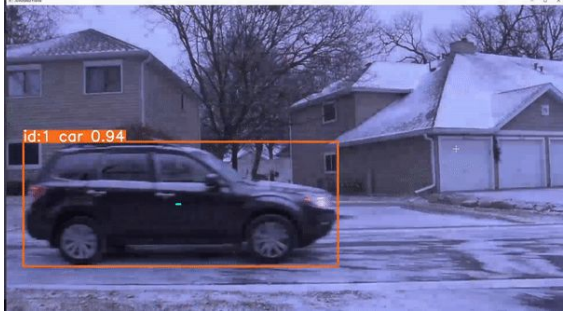


Unfiltered, unsegmented,
some parameter tuning



Filtered, segmented,
some parameter tuning

Calculating Velocity



- Use the same point on the car between images
- Calculate real world coordinates (x-y pixel coordinates -> meters, disparity values -> depth)

$$X = \frac{b\left({}^{L}u - u_0\right)}{d}, \quad Y = \frac{b\left({}^{L}v - v_0\right)}{d}, \quad Z = \frac{fb}{\rho_u d}$$

- Calculate velocity e.g. X_Vel = (X1-X0)/(frames/fps)
- Results: Average velocity over all frames has an error of 2.3%

Improvements

We assumed the car was traveling in a straight line, ignoring the y-axis

- easily included by using y-value in object tracking output

Improved stereo setup (synchronization and improved camera mounting)

Dynamic stereo parameters

Camera movement