Two-step tomographic reconstructions of temperature and species concentration in a flame based on tunable diode laser absorption tomography measurements using **Deep Neural Networks**

Zeeshan Nadir (Purdue University)

Nicola J. Ferrier (Argonne National Lab)

Pete Beckman (Argonne National Lab)

Pedestrian Detection & Tracking Using Kalman Filter & KLT Tracker

Zeeshan Nadir (Purdue University)

Nicola J. Ferrier (Argonne National Lab)

Pete Beckman (Argonne National Lab)

Problem Description

- Find pedestrians in the videos and track them

Problem Description

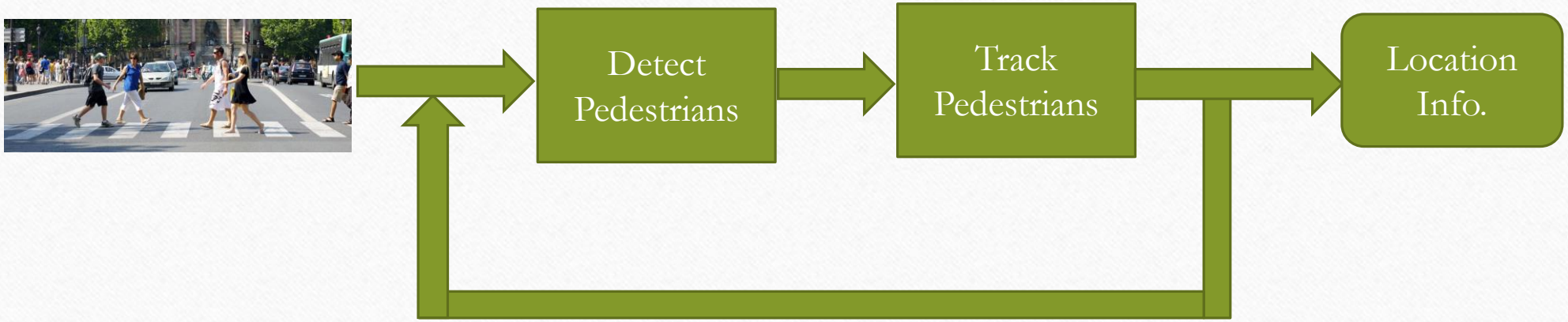
- What does that mean for a computer?

Problem Description

- In each frame, detect pedestrians
- Subsequently, locate pedestrians based upon their last known position
- May have to deal with following:
  - False negatives i.e., missed detections
  - Occlusion due to obstacles
  - Keeping track of identity
    - Model for motion and appearance

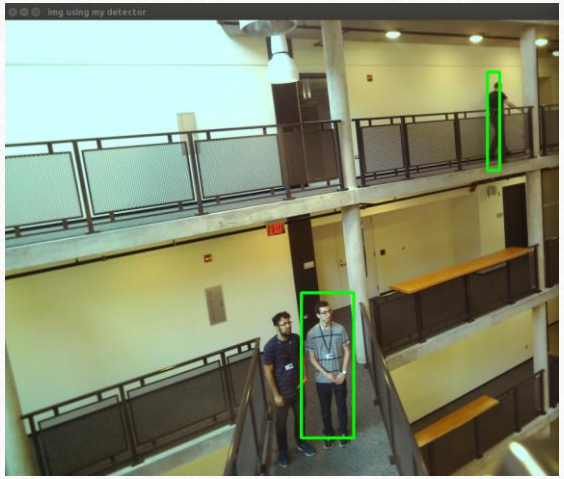
Breakdown of problem

- **Detection** and **Tracking** are two different problems



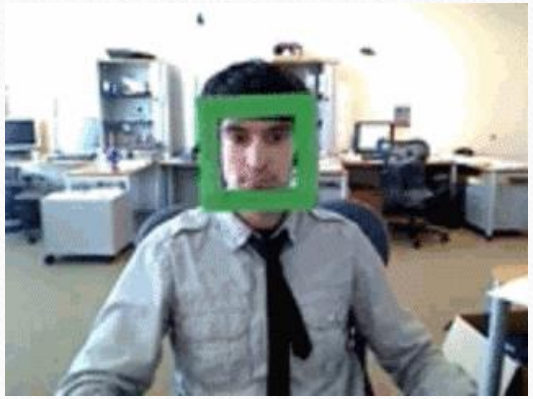
Detection Problem

- Detection is concerned with detecting an object of interest in an image
- Detections involves finding distinguishing features of objects
- Detections are usually computationally expensive
- To perform detection, we need a detection model
- The detection model is trained using training data



Tracking Problem

- Tracking is defined as finding a fixed object in each frame
- Tracking is a computational less expensive (**but not necessarily easy!!**)
- Tracking can use a model for motion
- Because of model mismatches, it's possible to lose track of the object
- To correct for such errors, need detection measurements every n^{th} frame



Tracking with detection

- Can we not detect people in every frame to solve this problem?

Tracking with detection

- The answer to this question is both Yes and No
- Technically, one can keep detecting in every frame

- **Caveats:**
  - Computationally expensive as you try to find out people in all locations from scratch
  - Missed detections
  - No identity associated with each detection

Detection & Tracking Combined

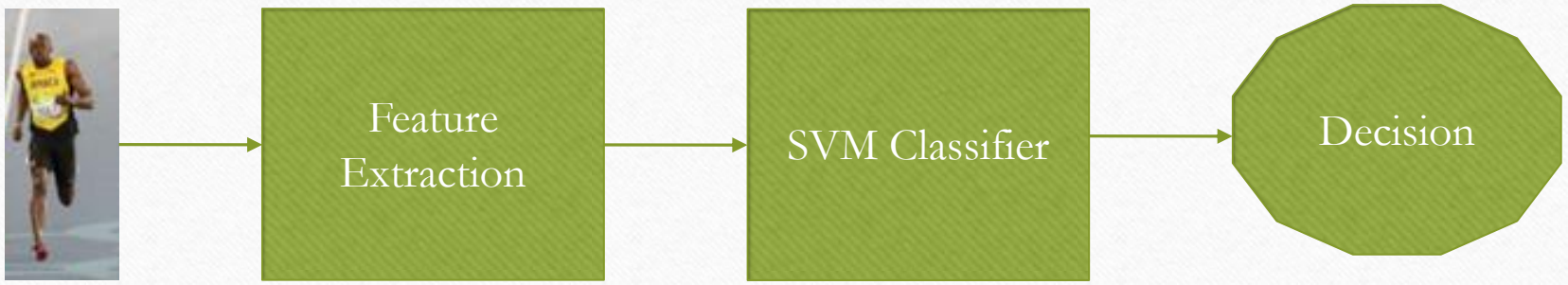
- Detection and tracking combined
- Detection using gradient based features
- Tracking using Kalman Filter

What's coming in rest of the slides?

- Object Detection
- Background Subtraction
- Kalman Filter Tracking
- KLT Tracking
- Algorithm/Approach to combine all the blocks
- Results

Theory/Formulation of Detection Problem

Pedestrian Detection Block Diagram



Feature Extraction

- Histogram of gradients:
  - First compute the gradient of images

Feature Extraction

- Histogram of gradients:
  - First compute the gradient of images
  - Divide the image into different cells

Feature Extraction

- Histogram of gradients:
  - First compute the gradient of images
  - Divide the image into different cells
  - Compute a histogram of gradients in these cells

Feature Extraction

- Histogram of gradients:
  - First compute the gradient of images
  - Divide the image into different cells
  - Compute a histogram of gradients in these cells
  - Normalize the gradients

Feature Extraction

- Histogram of gradients:
  - First compute the gradient of images
  - Divide the image into different cells
  - Compute a histogram of gradients in these cells
  - Normalize the gradients
  - Combine all the cells to make a big vector

Feature Extraction

- Histogram of gradients:
  - First compute the gradient of images

(a) (b) (c) (d)

-1	0	1

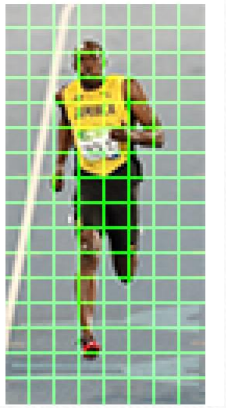
-1
0
1

Filter Kernels

(a) shows the original image, (b) shows the x-gradient image, (c) shows the y-gradient image, (d) shows the magnitude of the gradient

Feature Extraction

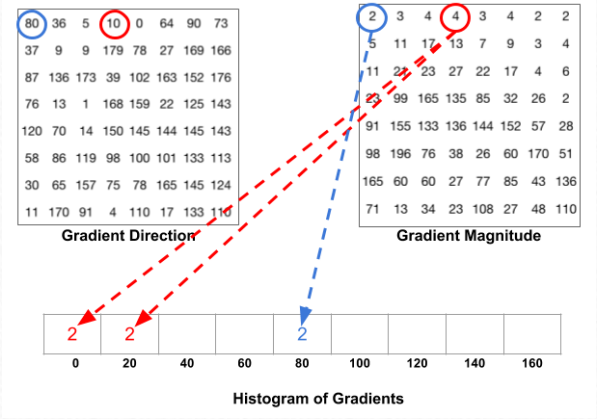
- Histogram of gradients:
  - First compute the gradient of images
  - Divide the image into different cells

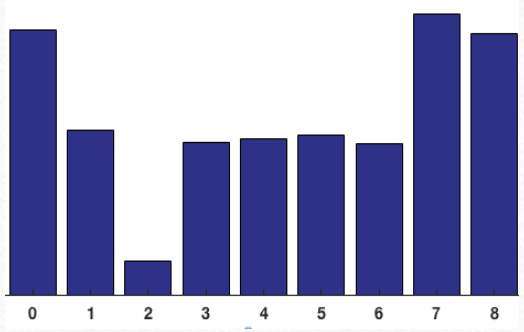


Pedestrian image divided in to cells

Feature Extraction

- Histogram of gradients:
  - First compute the gradient of images
  - Divide the image into different cells
  - Compute a histogram of gradients in these cells

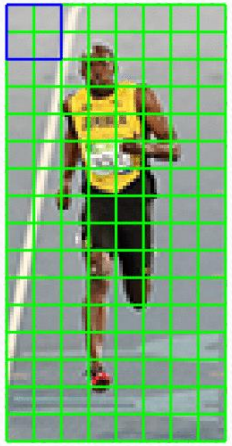




Histogram computed for each 8x8 cell

Feature Extraction

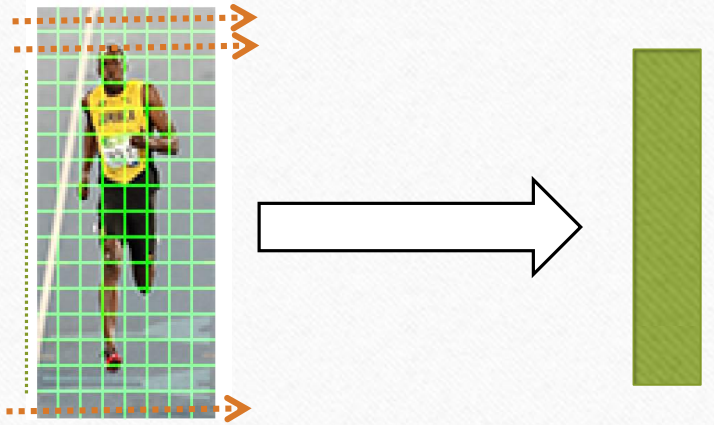
- Histogram of gradients:
  - First compute the gradient of images
  - Divide the image into different cells
  - Compute a histogram of gradients in these cells
  - Normalize the gradients



Normalize each cell using a block around it

Feature Extraction

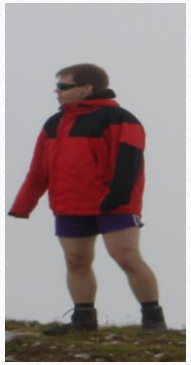
- Histogram of gradients:
  - First compute the gradient of images
  - Divide the image into different cells
  - Compute a histogram of gradients in these cells
  - Normalize the gradients
  - Combine all the cells to make a big vector



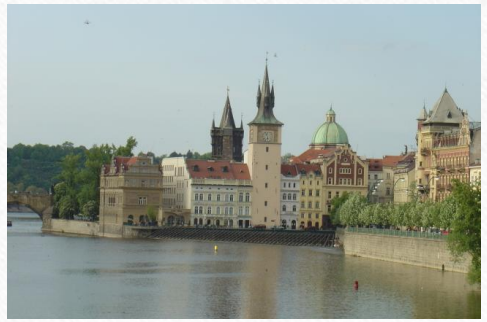
Cells are read in raster order to produce a vector

SVM Classifier

- Collect a lot of images with/without pedestrians





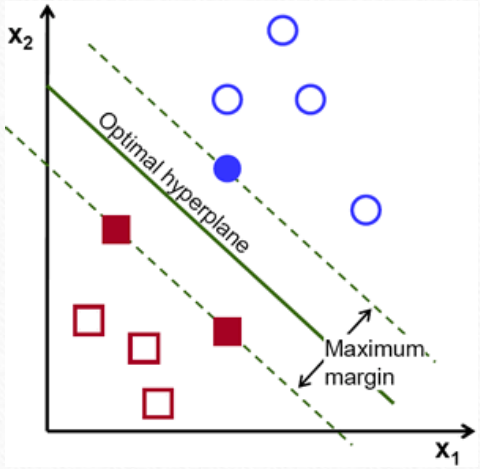



SVM Classifier Input

- Convert all the training images into feature vectors

SVM Classifier Input

- Convert all the training images into feature vectors
- Use these features to train a support vector classifier



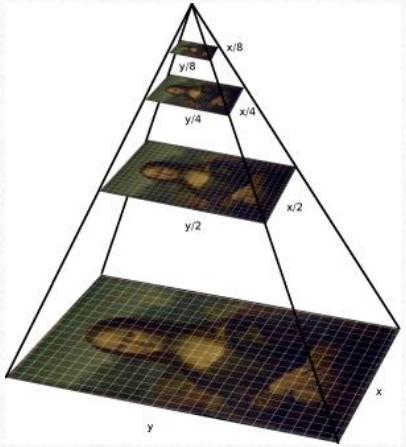
SVM Classifier 2D example

Image Pyramid

- So far we have a classifier that can detect a person in the image using a fixed window size
- What if the person appears too big with respect to our window?

Image Pyramid

- So far we have a classifier that can detect a person in the image using a fixed window size
- What if the person appears too big with respect to our window?

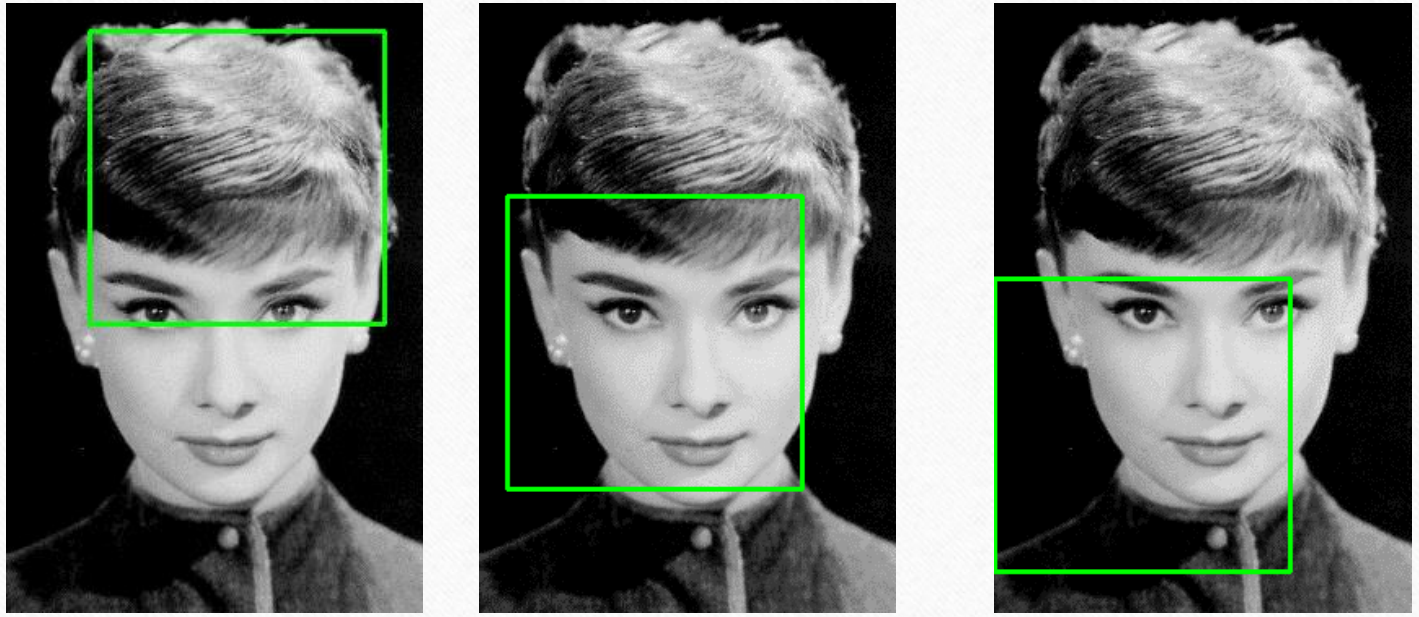


Window Sliding

- What if the person is not in the center of the image?
- What if there are more than one person in the image?

Sliding Window

- We slide a window across the entire image



Final Classifier

Obtain the input image

For each scale

    For each sliding window

        Determine if there is a pedestrian or not

    End For

End For

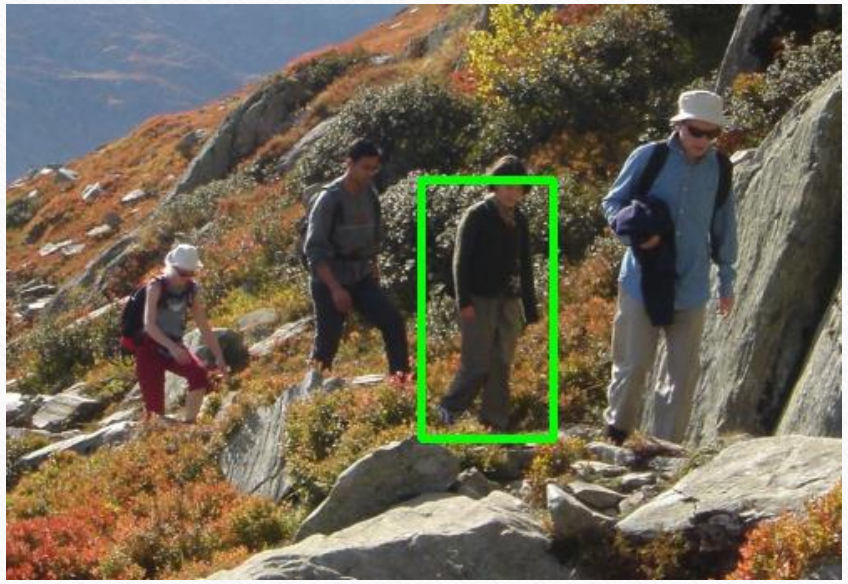
Get rid of overlapping pedestrian windows

Detection Results

(INRIA Dataset)

Our trained detector

OpenCV default detector



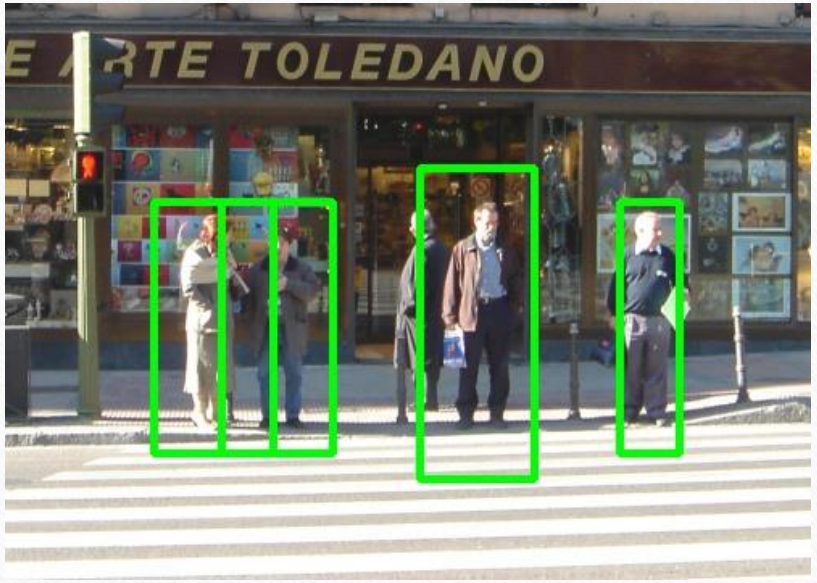
Our trained detector

OpenCV default detector

Our trained detector

OpenCV default detector

Our trained detector

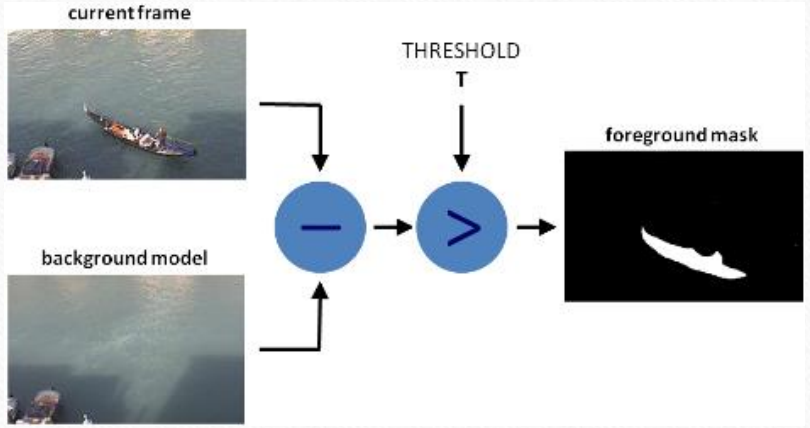


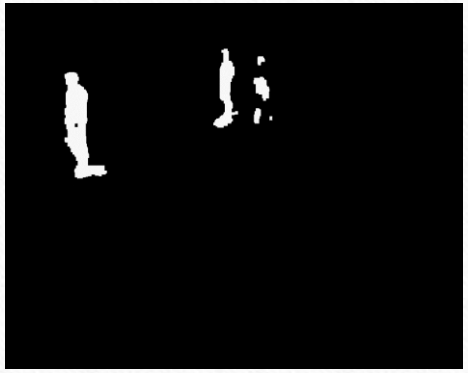
OpenCV default detector

Questions?

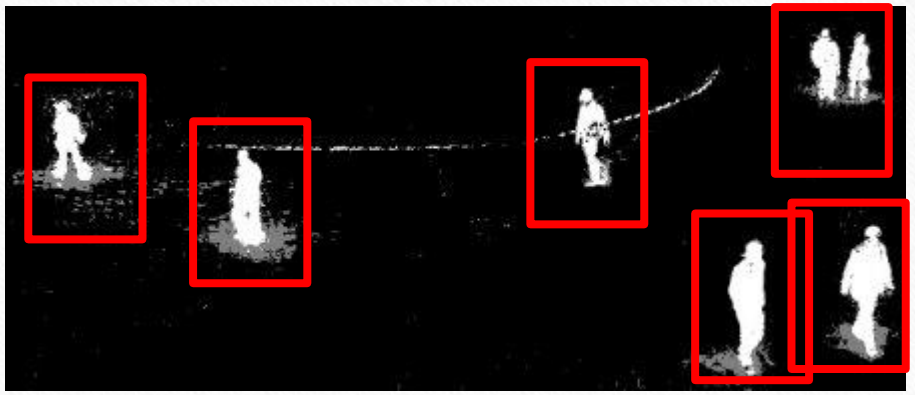
Improving Detection With Background Subtraction

- Can increase the speed of detection by ignoring background regions
- Can further improve the performance by ignoring pedestrian free regions of images using apriori information





Background Subtraction Result



- Find people by searching in relatively big boxes around the foreground regions

Theory/Formulation of Tracking Problem

Tracking Overview

- Want to track detected objects in each frame and maintain identity
- Different approaches to track objects
  - Point Tracking e.g., Kalman Filter
  - Kernel Tracking e.g., KLT Tracking
  - Silhouette Tracking e.g., Shape based models, Contour matching
- Tracking Challenges:
  - Abrupt motion of objects
  - Changing appearance of objects
  - Object-to-object and object-to-scene occlusion

Tracking Overview

- Want to track detected objects in each frame and maintain identity
- Different approaches to track objects
  - Point Tracking e.g., Kalman Filter
  - Kernel Tracking e.g., KLT Tracking

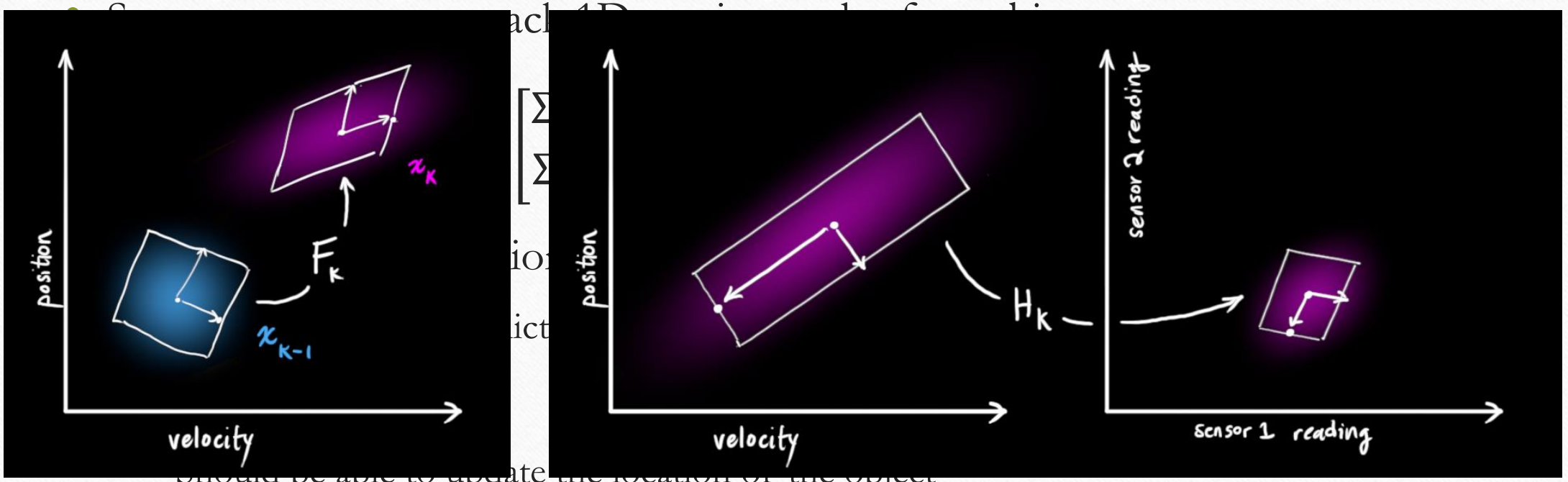
Tracking Overview

- Point Tracking e.g., Kalman Filter
- Kernel Tracking e.g., KLT Tracking

- Requires a measurement every frame
- Requires a motion model
- We use constant velocity with a limit on maximum velocity

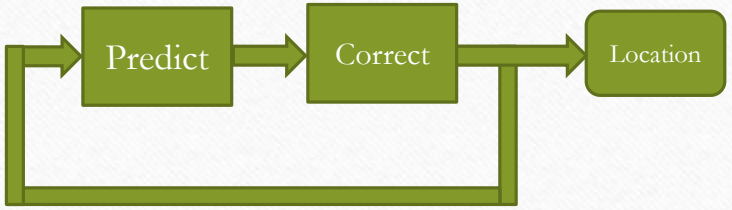
- Requires some salient points
- Matches the points between each frames

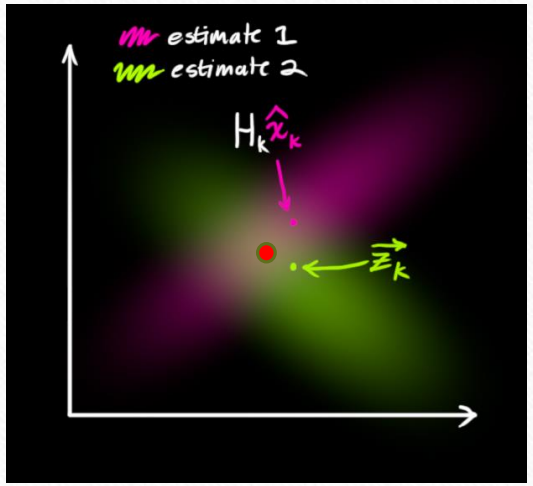
Kalman Filtering



Kalman Filter Updates

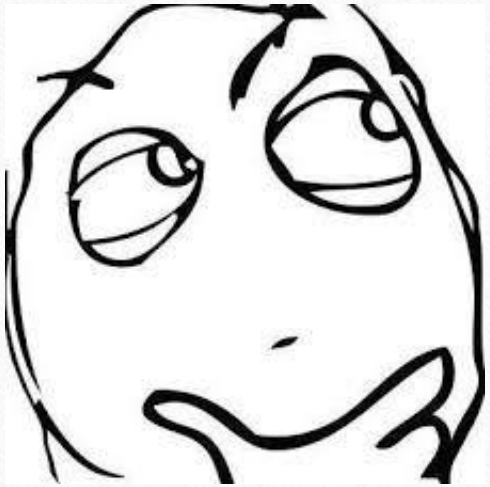
- First we predict the location of the object
- Next, we acquire a (noisy?) measurement of the object
- We then combine the two to get our "best" known estimate of location





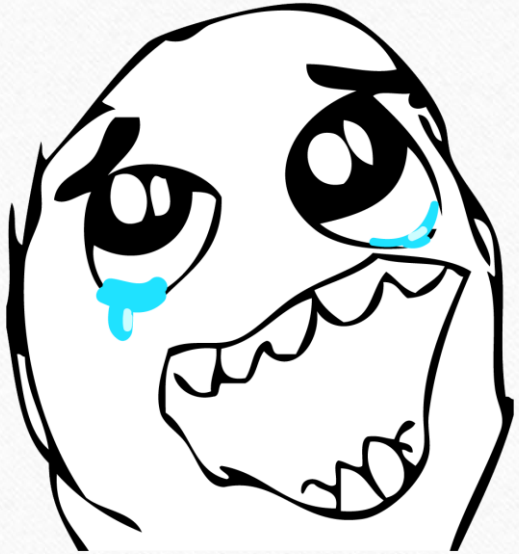
Measurements for Kalman Filter

- So where does Kalman Filter get it's measurements from?



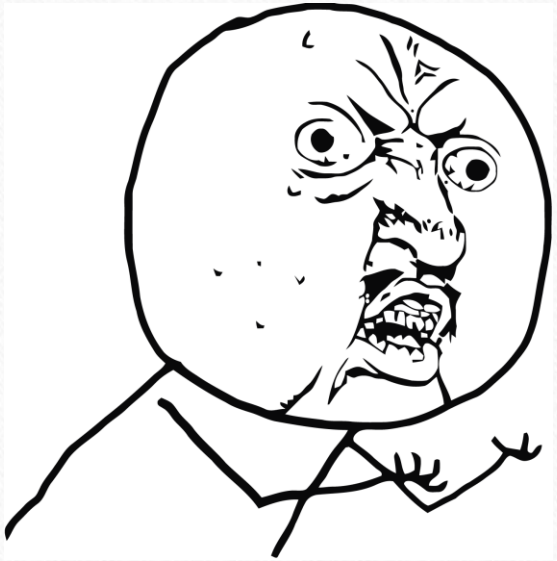
Measurements for Kalman Filter

- Could use pedestrian detection to feed measurements to Kalman Filter (right?)



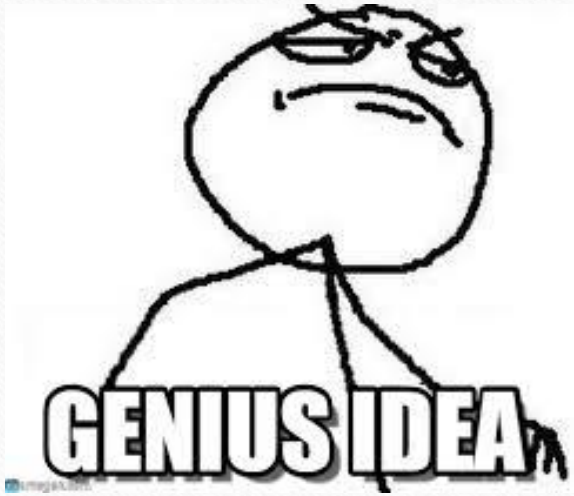
Measurements for Kalman Filter

- However, running a detection every frame can be computationally expensive and reduce the performance …



Kanade Lucas Tomasi (KLT) Tracker

- So what do we do?
- Use a relatively cheaper tracking method to get measurements for Kalman Filter
- Use detections less often to correct the course
- Shall use KLT tracking to track the pedestrians



KLT Tracking

- KLT Tracking is not very accurate in comparison to detection
- **Final Strategy:**
  - Use detection less often and KLT tracking more often
- **This has advantage of higher speed and better accuracy**

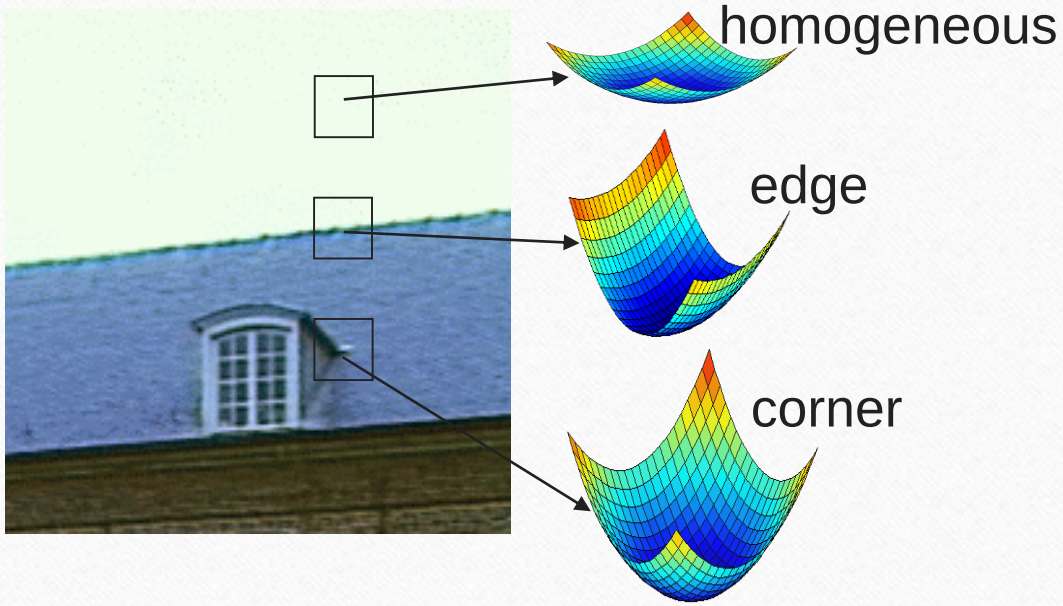
Nuts and Bolts of KLT Tracking

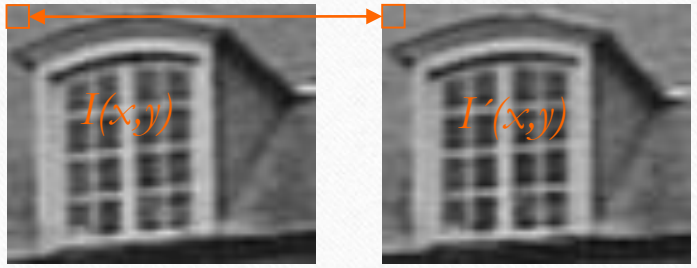
- Want to find some important features in the image
- As the video doesn't change much frame to frame, easy to track these feature points
- **Optical Flow Equation:**

$$f(x,y,t) = f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt$$

Good Features to Track (Tomasi)

- Use “strong” corner points

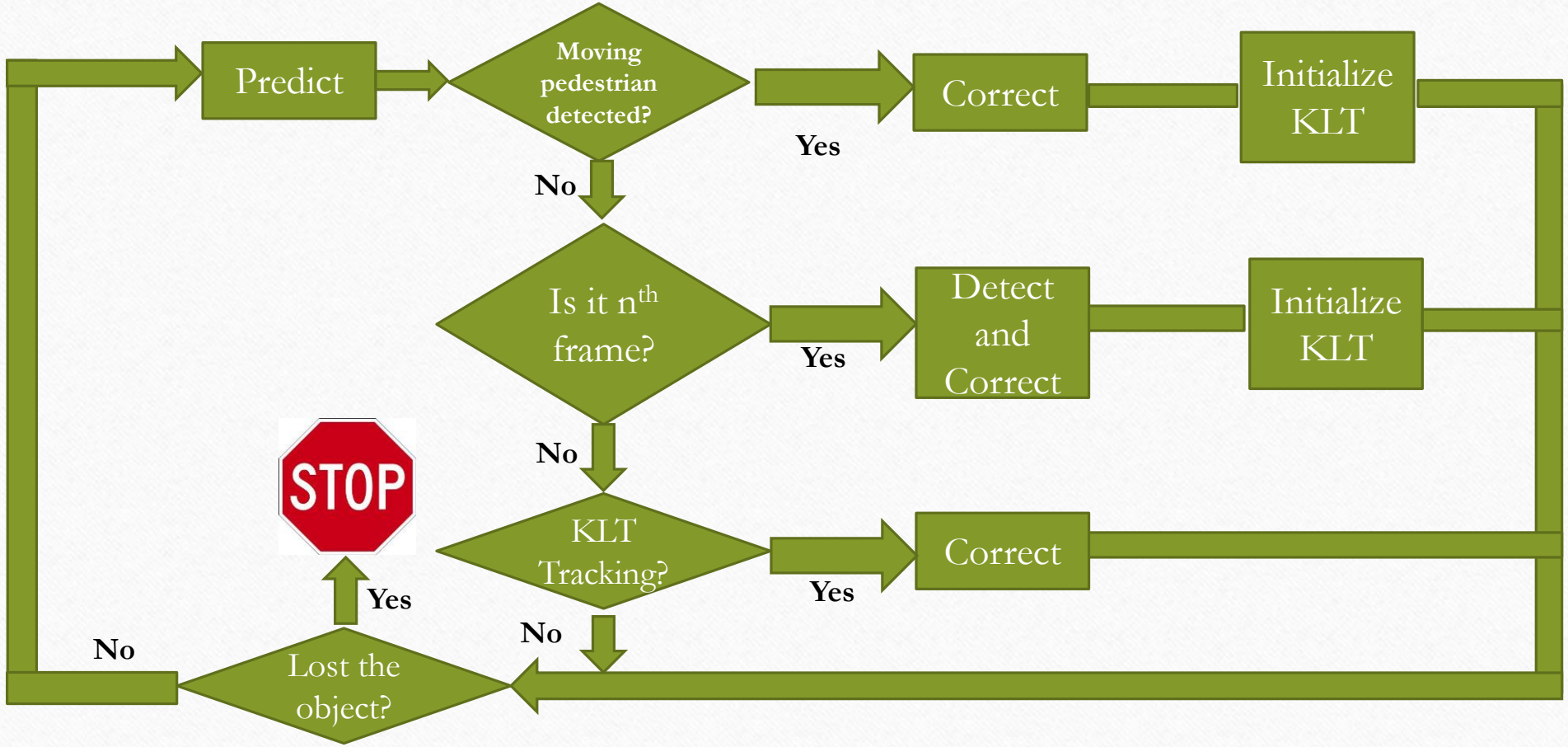




Comparing smooth regions

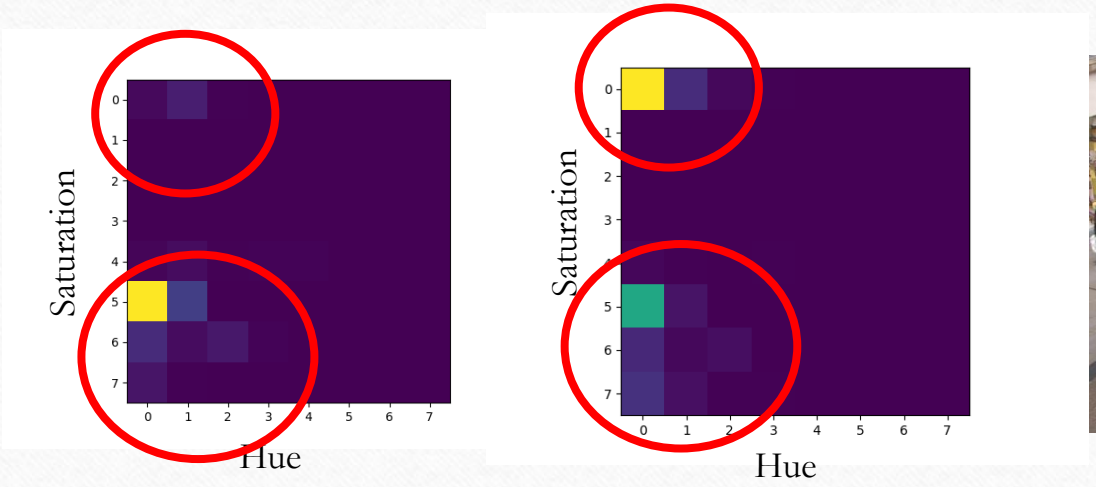
Comparing corners

Tracking Algorithm



Mapping Measurements to Existing Pedestrians

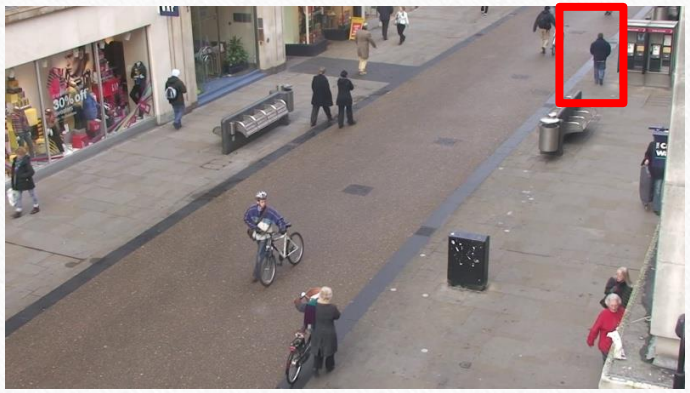
- How do we map new measurements to existing pedestrians?
- Look for overlap
- **Appearance Model:**
  - Use an appearance model based on color distribution of the pedestrian



Are we done?

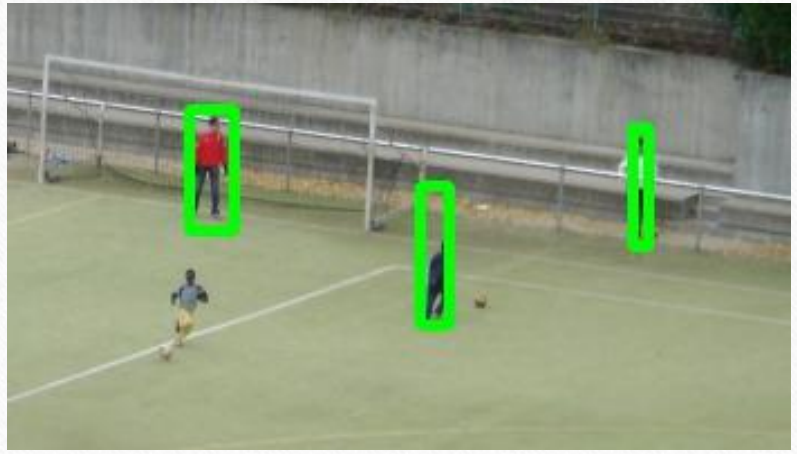
- What about pedestrians too small for 128x64 window size?
- Use a hierarchy of pedestrian detection models

32x64 → 128x64



Multiple Models for Pedestrian Detection

- Following example shows the utility of using multiple models



Pedestrian detected using 64x32 window

Pedestrian detected using 128x64 window

Results

Pedestrian Tracking Using Background Subtraction Only

Pedestrian Tracking Using Kalman Filter Only

Pedestrian Tracking Using Kalman Filter and KLT Tracking

Key Contributions

- A unified framework incorporating KLT tracker with Kalman Filter
- A strategy to propagate the feature points from frame to frame
- Putting all the building blocks together and showed that it works (decently?) ☺

Future Work

- Try the custom trained detection model to study it's speed/accuracy tradeoff
- Write a report/paper to document all the work
- Upload the code on Github and write documentation/read me
- Run the code on Waggle node to resolve any issues

Thanks


THANK YOU FOR YOUR ATTENTION
YESSSS, FINALLY OVER!
by 4memes.net