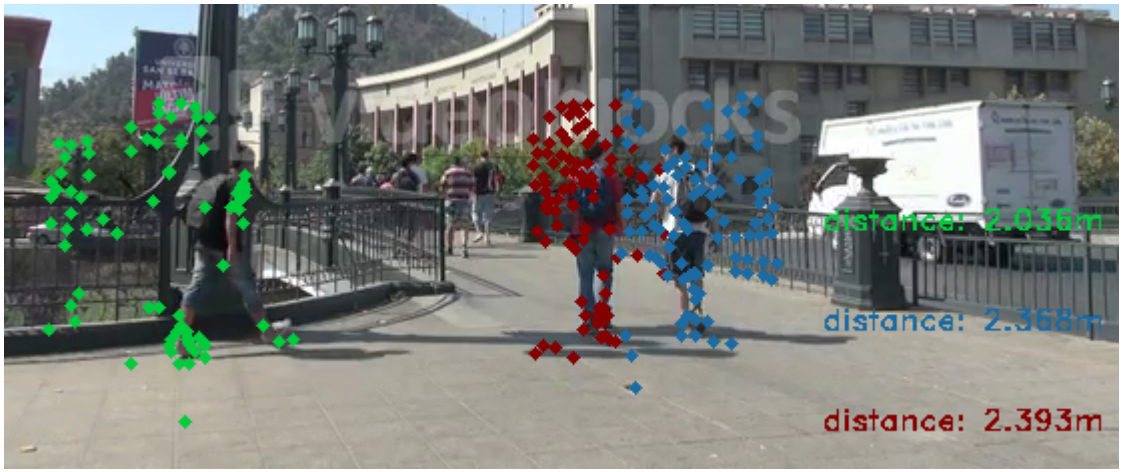
Real Time Pedestrian Detection, Tracking and Distance Estimation

Keywords: HOG, Lukas Kanade, Pinehole Camera, OpenCV

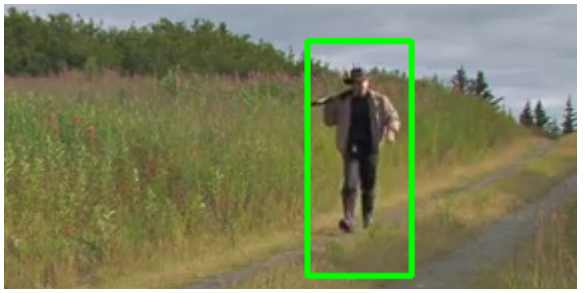


Omid.Asudeh@mavs.uta.edu
University of Texas At Arlington

# of slides : 30

Problem definition (Goal):

In this project, given a **stream of video**, we want to **detect people, track** them, and find their **distance** in a **real-time** manner.

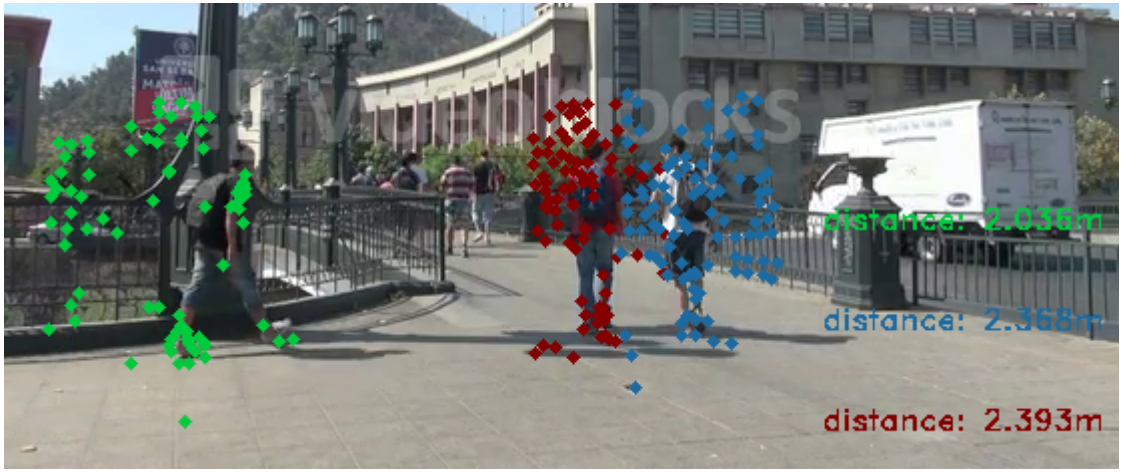


Issues:

- **Pedestrian Detection**

- **Pedestrian Tracking**

- **Distance Detection**

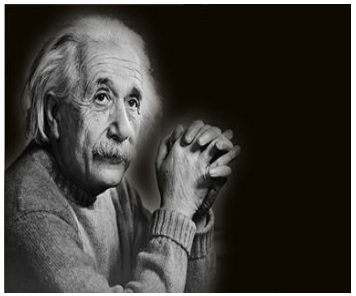


People Detection

What is HOG (Histograms of Oriented Gradients)?:

Answer: It is a kind of feature descriptor.

Then what is a feature descriptor?

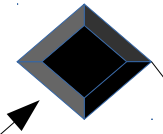


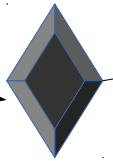
The goal of a feature descriptor is to **generalize an object** in a way that same objects (e.g. pedestrians) make as close as possible descriptors. That way the task of classification becomes easier.

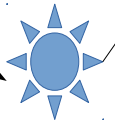
Objects

Descriptors

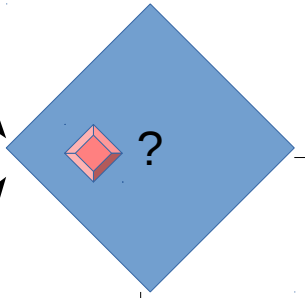
HOG







Classifier (SVM)



Yes

Is Pedestrian

No

Is Not Pedestrian

Pedestrian detection Process

Challenges of HOG and SVM Classifier:

1. HOG is Computationally Complex
2. It needs the whole body to be detected
3. Classifiers are not exact (it may classify wrongly)

HOG in Open CV :

Input : an image

Output : **a list of rectangles each bounding a pedestrian**

Slow

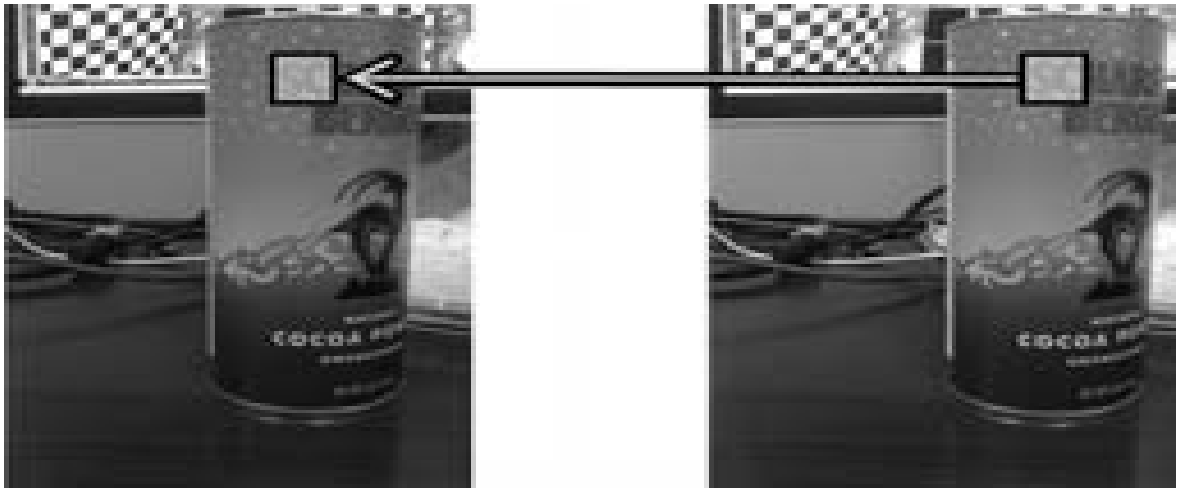
Idea : Read a video stream frame by frame and do HOG on each frame and show the results

Lets try the idea! To make sure

Feature point Tracking

What Does a feature tracker do ?

Answer: Suppose we are given two images. We have the position of some points on the first image. In the second image the scene has changed a little. The goal of the tracker is to find the position of the points in the second image.

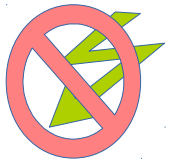


Benefit : It is not computationally complex

Tracking in OpenCV: OpenCV has implemented the Lucas-Kanade (LK) traking algorithm

Input: Image A, list of feature points in image A, Image B

Output : List of tracked feature points in the Image B



Idea : Forget about the HOG. Read the video stream frame by frame, do corner detection, track corners using LK algorithm from the current frame to the next in a loop.

seems that you forgot about the **pedestrian**. Using this algorithm, the program will track all the feature points (corners) of the image and not specially those related to the people.

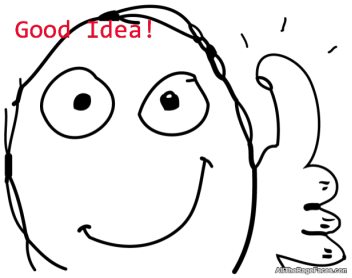
Lets **Not** try the idea To make sure!

- **HOG detects people but is not fast enough to be used in real-time applications**
- **Tracking corners is a relatively fast operation but it is not specified to people**

Lets combine the two ideas!!

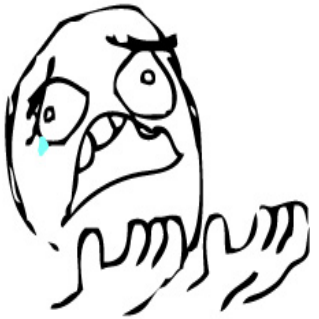
Idea :

1) Read the video stream frame by frame
2) Do people detection using HOG until finding people.
3) Do corner detection for each people
4) Track corners using LK from the current frame to the next.

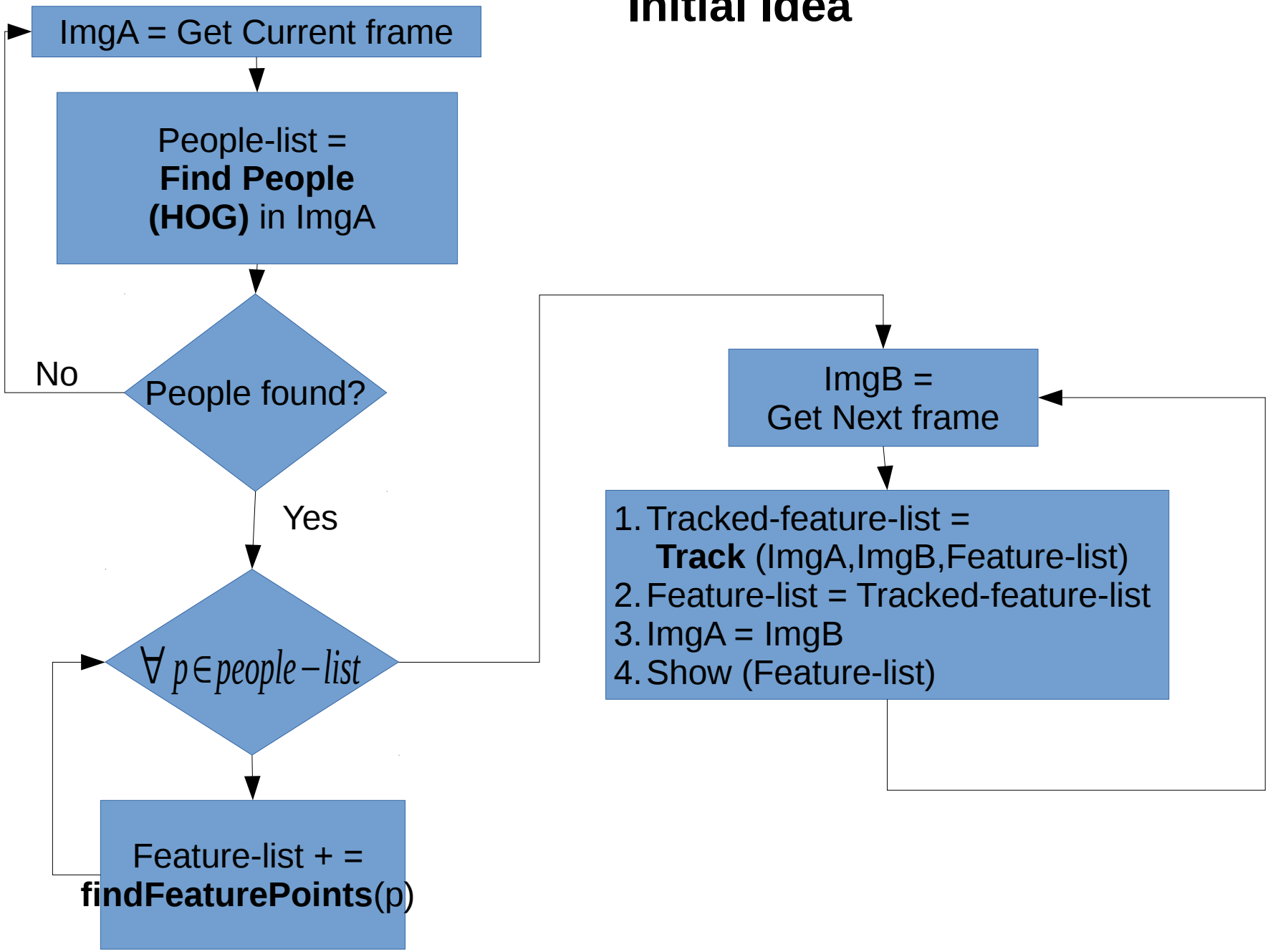


Good idea, but it still have some issues!!

But Why??



Initial Idea



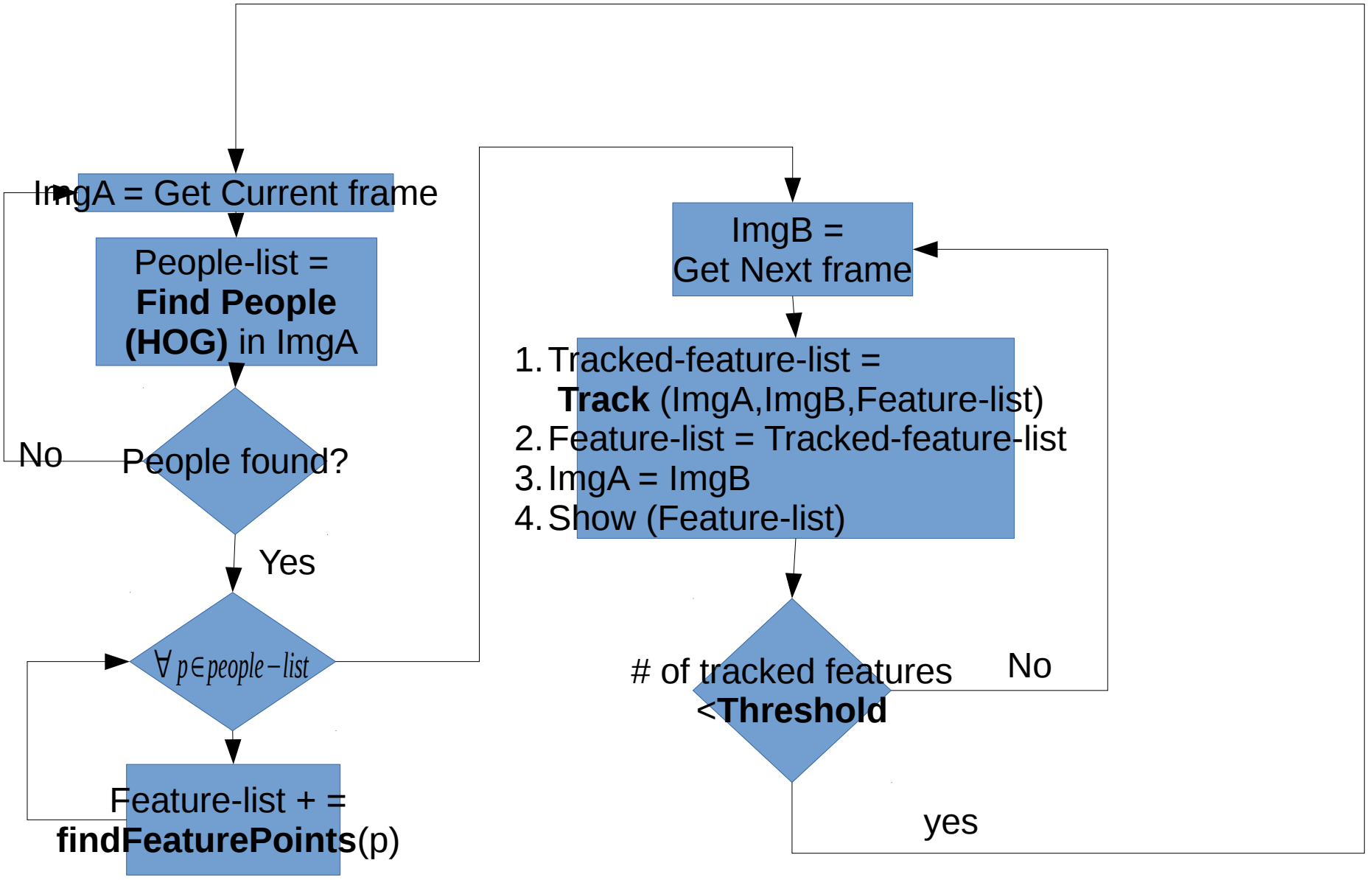
Detection

Tracking loop

? What if some feature points lost during tracking loop?

Idea : go back and do the pedestrian detection again

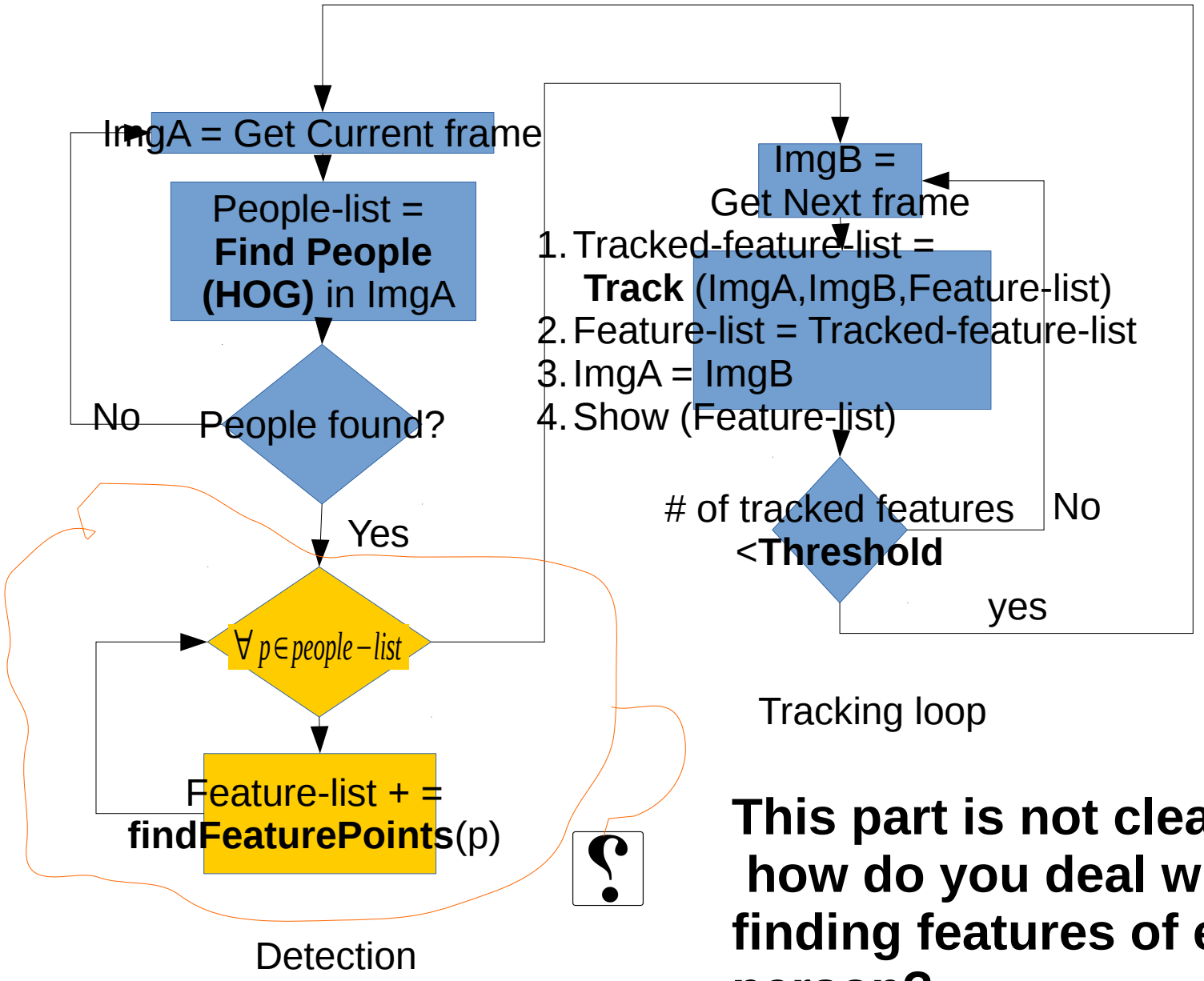
Developing Idea...



Detection

Tracking loop

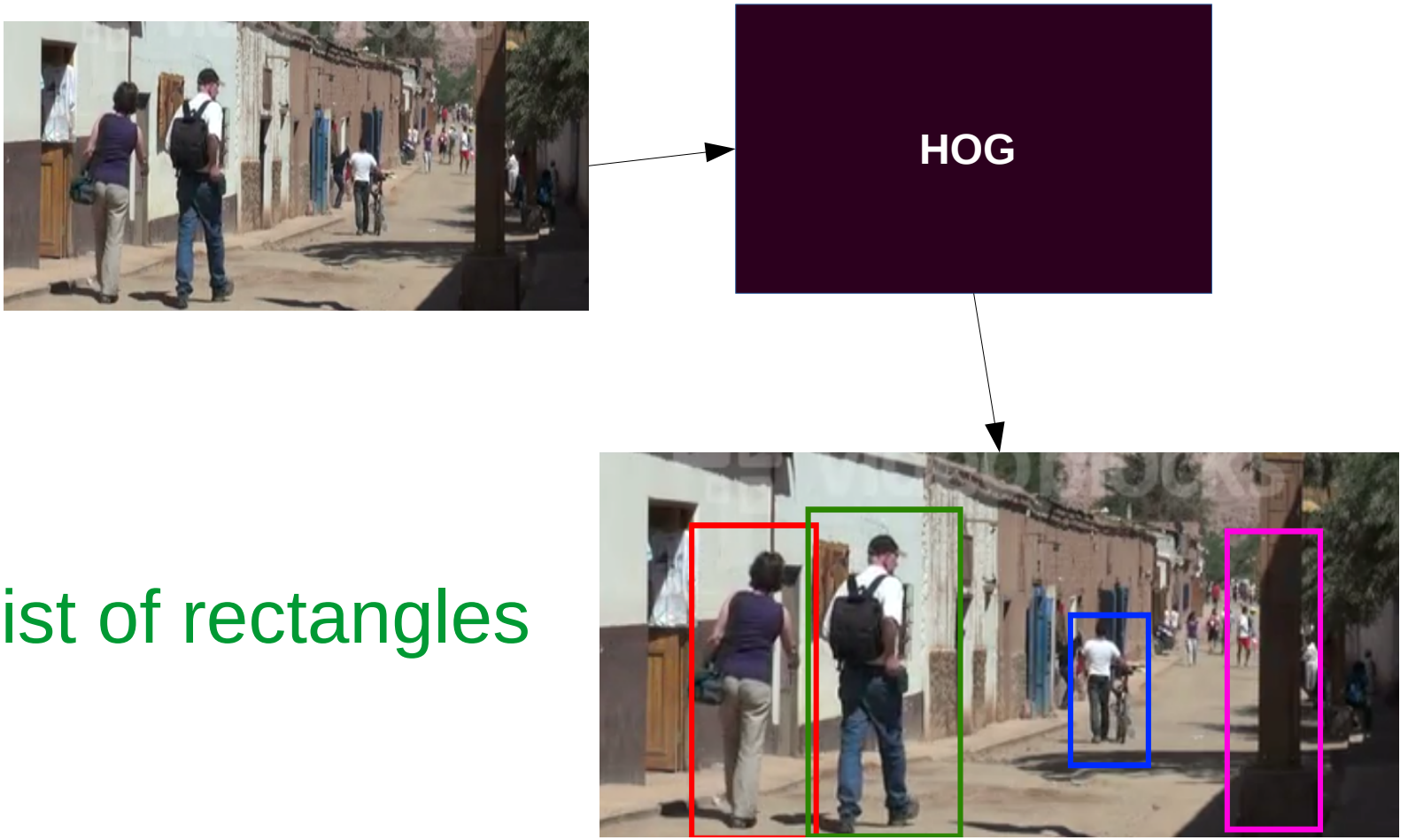
Developing Idea...



This part is not clear, how do you deal with finding features of each person?

Okay, Let's talk about more details

What does HOG give us in openCV?

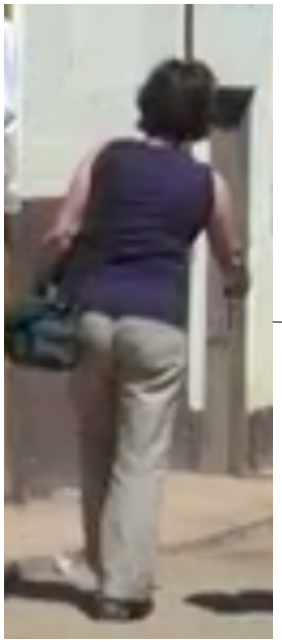


A list of rectangles

What is a rectangle? <X,Y,W,H>

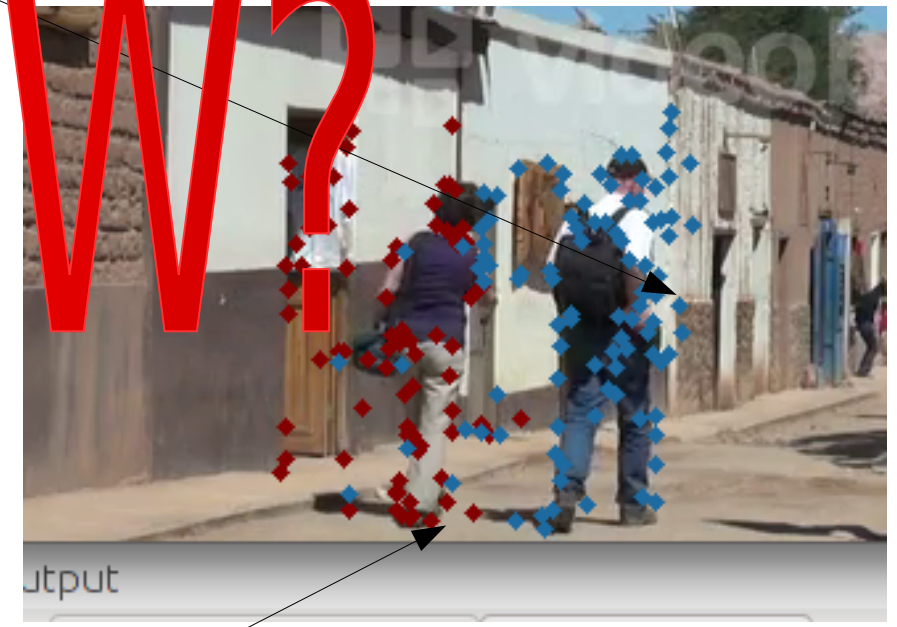
So, HOG gives us : **<(x1,y1,w1,h1);(x2,y2,w2,h2);(x3,y3,w3,h3),...>**

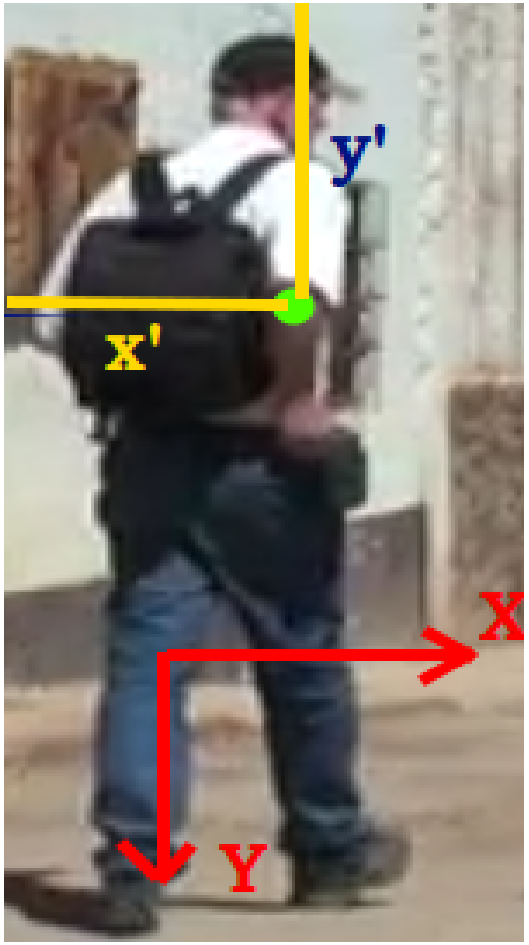
Crop each rectangle and do feature detection for it



Then map features of each rectangle to the original image

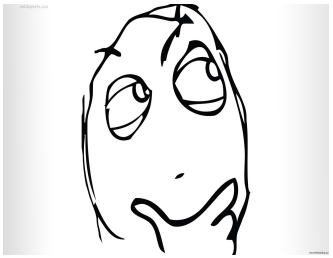
HOW?





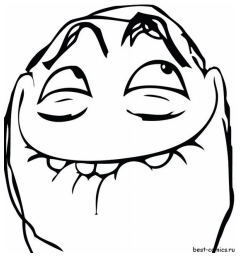
Coordinates in original image

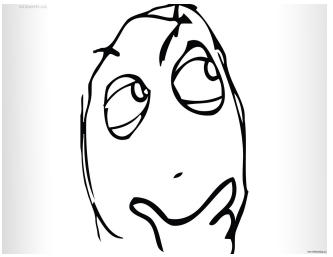
$$\begin{cases} X = X1+x' \\ Y = Y1+y' \end{cases}$$



It seems that you tackled the complexity issue of HOG so far, and the algorithm can detect and track people relatively fast. Also it can track people even if their full body is not in the screen.

Yes, I did.





You know what? I think there is something **badly** wrong with your algorithm

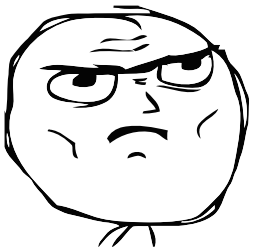
What??????

Challenge :

- HOG is not a perfect algorithm. It may consider wrong objects (like a tree) as people.

- The tree is not moving, so it rarely loses its feature points during tracking loop.

- Therefore, **the algorithm recur tracking loop infinite times (trap), and it will never go to the detection phase again!!**



You may say this is the end of the story, but there is always hope!

- The wrongly recognized object, is not moving, **so it rarely loses its feature points.**

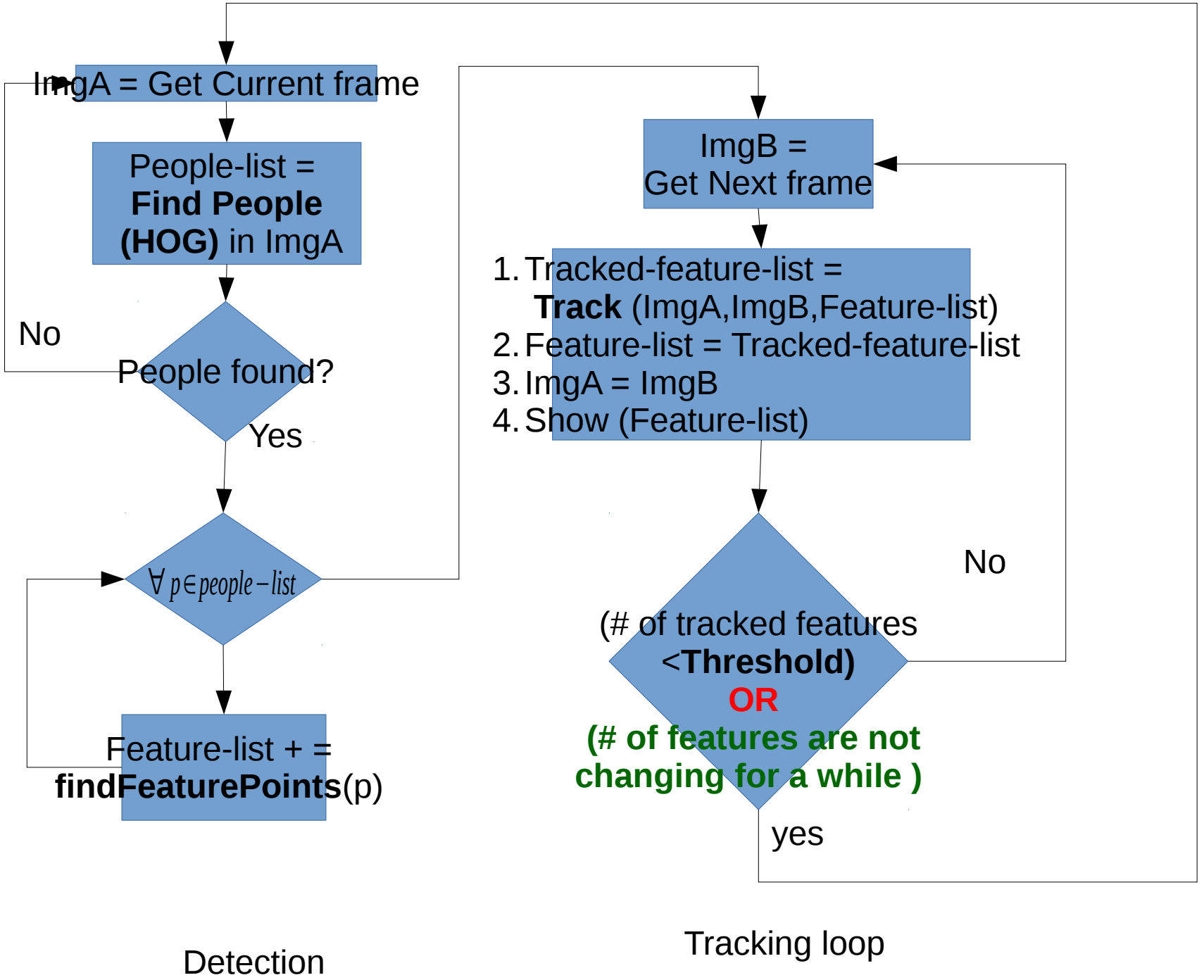
This is the key!

Idea:

If the number of the feature points are not changing during a while, say in 200 frames, then the algorithm should suspect a trap state and break the tracking loop.

Therefore, we can tackle this problem by adding a simple condition to the previous algorithm

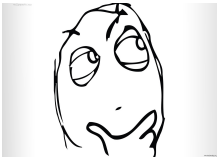
Dealing with wrong object detection challenge



Distance recognition

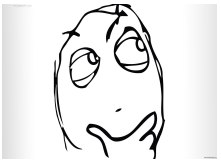
Problem definition: Using a pinhole camera, we want an strategy to estimate the distance of the pedestrians

Challenge: that is impossible to get the distance of objects using just a pinhole camera!!

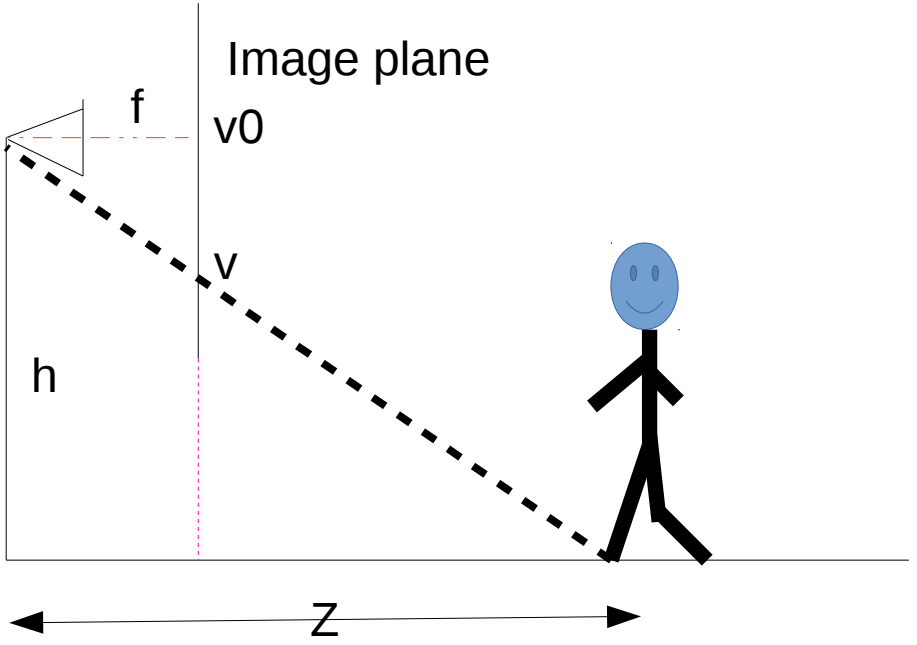


Yes, but what if we have a piece of information?

What kind of information are you talking about?



Like the height of the camera from the ground!



It can be proved that:

$$Z = \frac{f * k_v * h}{v - v_0}$$

Where f and k_v are the focal length and the pixel density(pixel/meter) of the camera.

Challenge: We should find V which is the height of the **base point** (where feet of the pedestrian touch the ground) in the image plane

But How to find the bottom point?

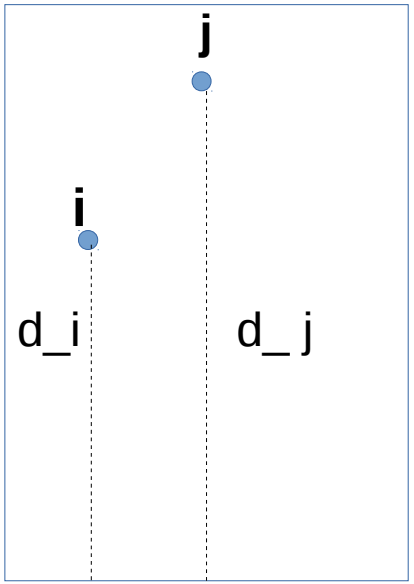
Let's define the bottom of the rectangle obtained from detection phase as the **base line.**

Base line

Idea: the average height of the feature points from the base line remains unchanged in the video stream.

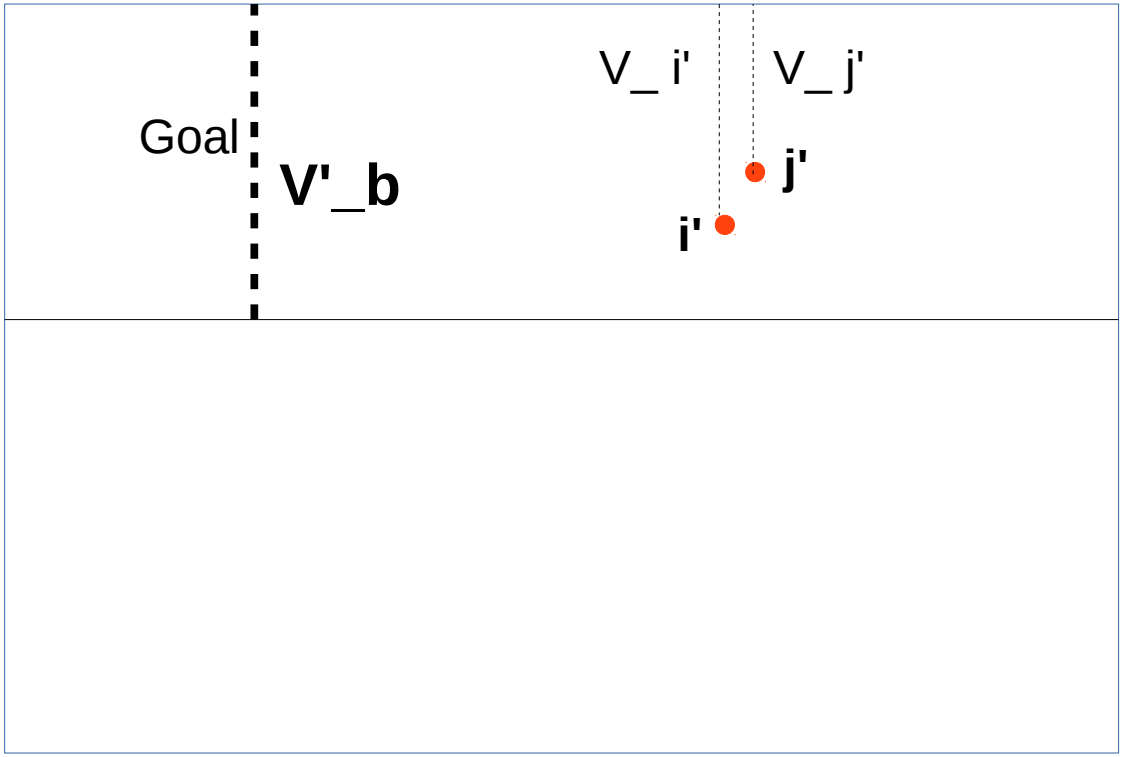
Base line Estimation

Example for 2 points



Detection Rectangle

Keep <d_i, d_j>



The whole image while tracking

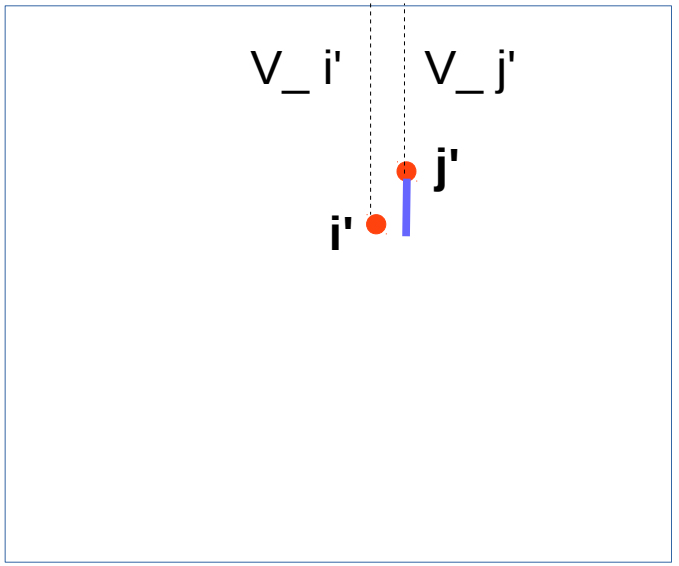
V'_b = Average(V_ j'+d_ j , V_ I' +d_ i)

Scaling

The closer the pedestrian to the camera the more difference between the points.

Scaling factor:

$$\alpha = Average_{\forall_{i,j \in people[p]}} \frac{V_{j'} - V_{i'}}{d_i - d_j}$$



$$V_b = \alpha * V'_b$$

Let's see some videos of the results of this work!

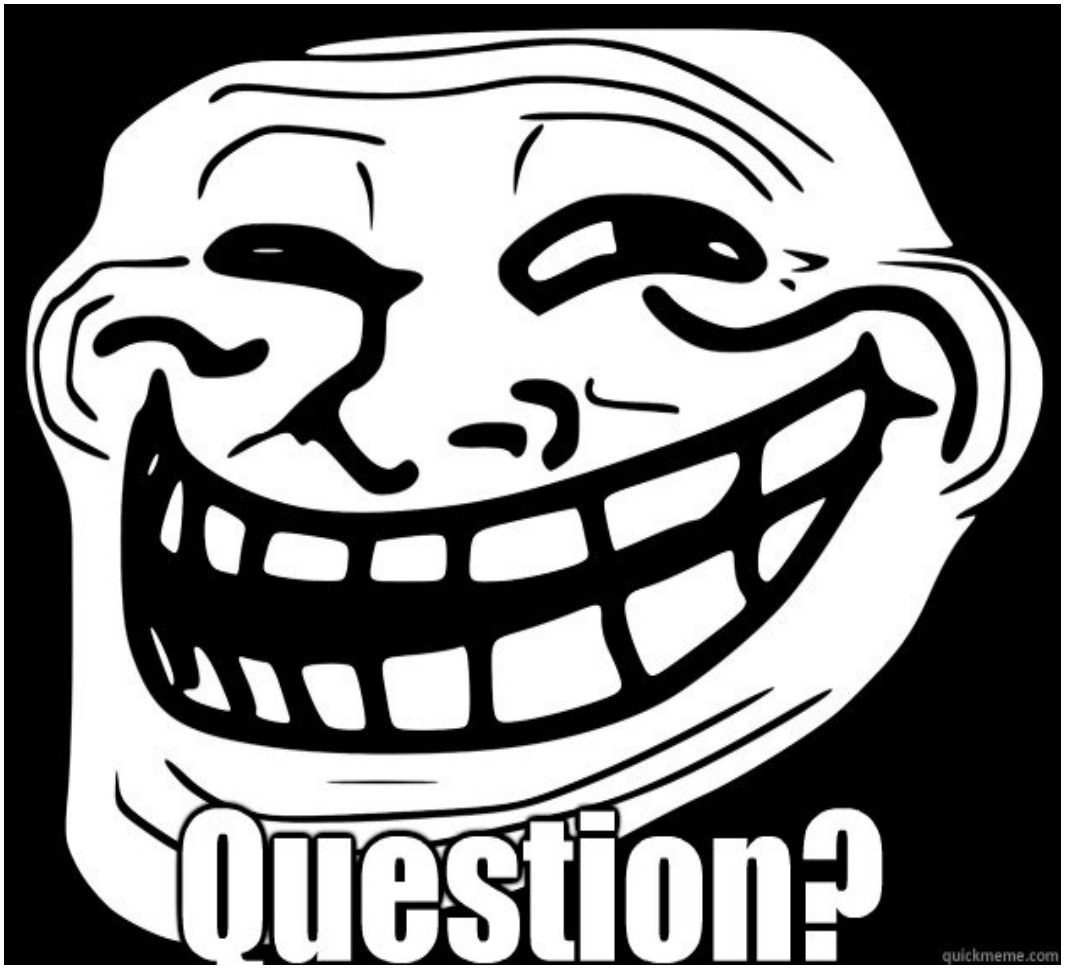
References:

- Learning opencv, 1st edition , O'Reilly Media, Inc. ©2008
ISBN: 9780596516130, Dr. Gary Rost Bradski, Adrian Kaehler

- http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

- https://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/

- http://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method



Omid.Asudeh@mavs.uta.edu