UofT Computer Vision Club

Paper Breakdown Series

Object Tracking

youtube.com/@uoftcv

github.com/ut-cvdg/paper-breakdown

Follow us on social media

UNIVERSITY OF TORONTO

UofT CV is sponsored by GitHub

Sign up for the GitHub Student Developer Pack:
https://education.github.com/benefits

UNIVERSITY OF TORONTO | CVDG

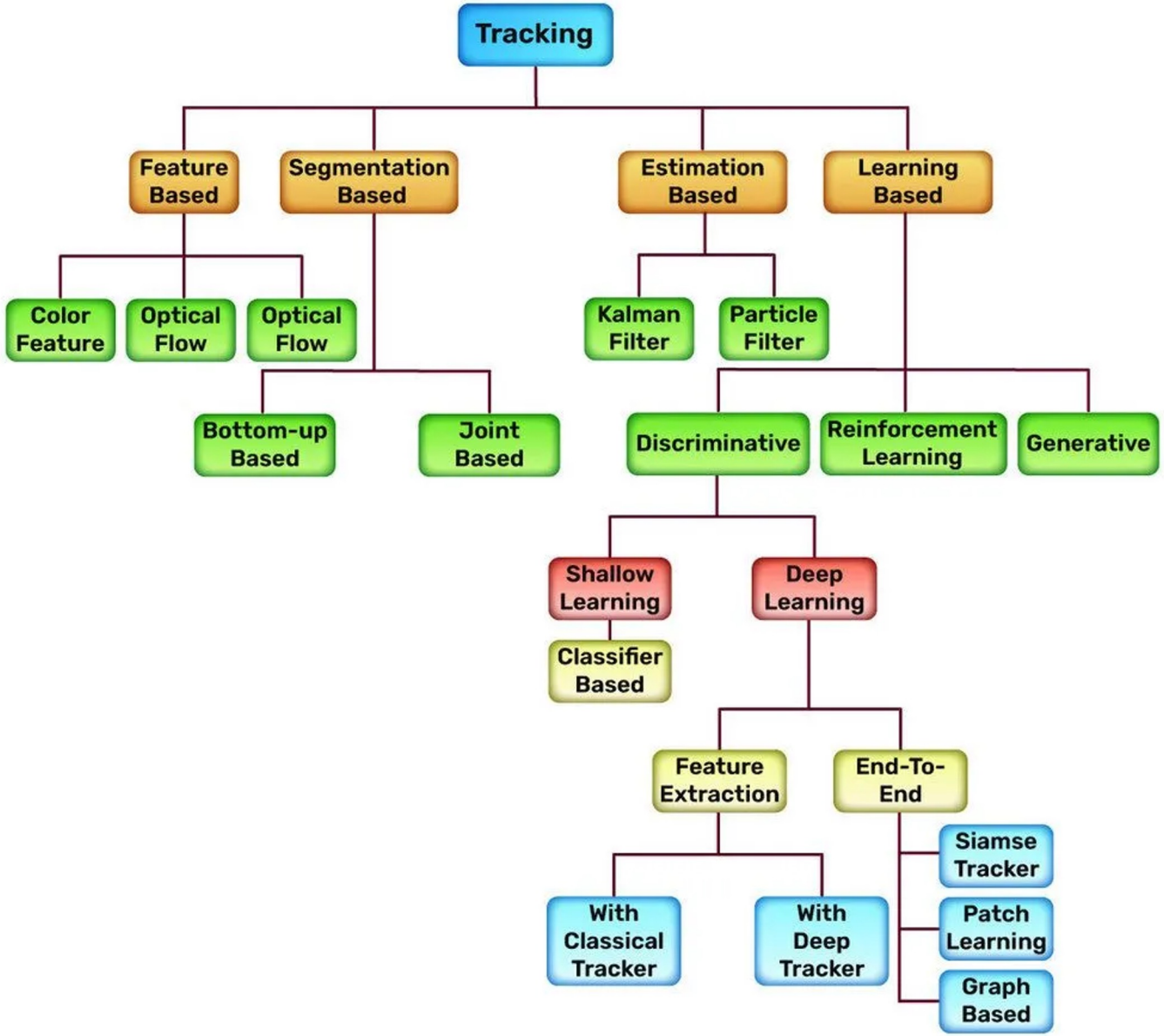
About the Club

- We are a student club for Computer Vision and Machine Learning affiliated with the University of Toronto.
- We run:
  - paper “discussions” for paper in the past 1 week
  - conference-style research orals for pre-prints
  - Now, paper breakdown series

If you want to help run our club, talk to any of us or drop an email contact@cvdg.tech.

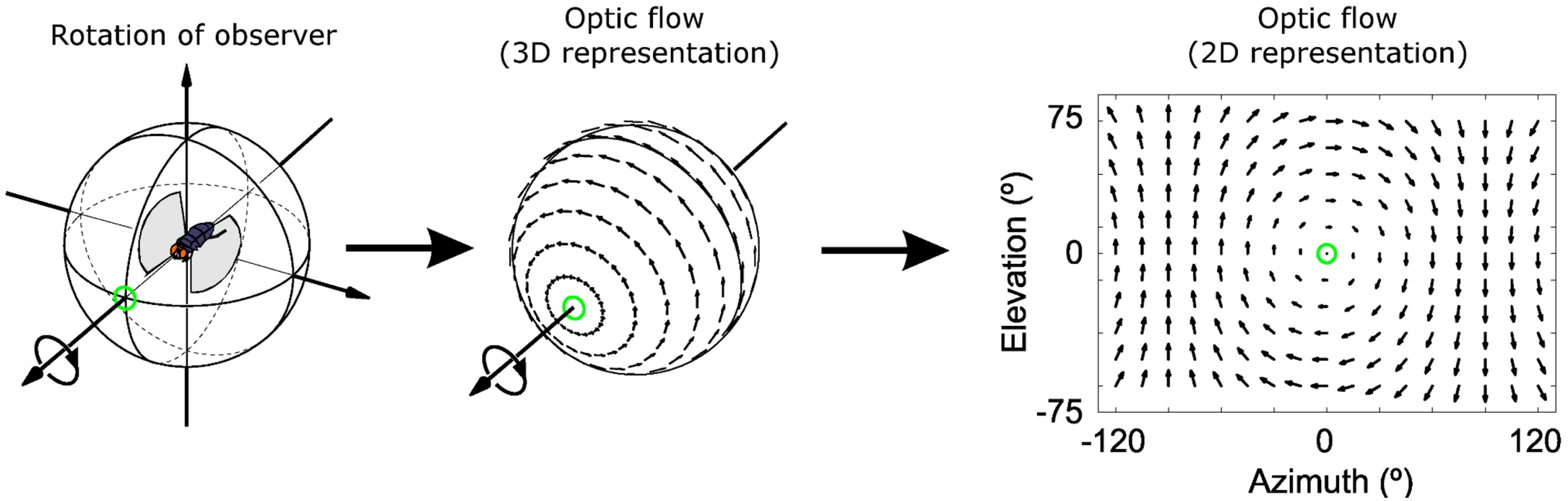
What is the Object Tracking ?

Continuously monitoring and recording an object's spatial position and directional alignment, both in the moment and over time. This tracking persists even when the object is partially or fully hidden from view, the camera is moving, or lighting conditions fluctuate.


Tracking
Feature Based
Segmentation Based
Estimation Based
Learning Based
Color Feature
Optical Flow
Optical Flow
Kalman Filter
Particle Filter
Bottom-up Based
Joint Based
Discriminative
Reinforcement Learning
Generative
Shallow Learning
Deep Learning
Classifier Based
Feature Extraction
End-To-End
With Classical Tracker
With Deep Tracker
Siamse Tracker
Patch Learning
Graph Based

Optical Flow

Optical flow or optic flow is the pattern of apparent motion of objects, edges and surface in a visual scene caused by the relative motion between an observer (an eye or a camera) and the scene



Source: Optical Flow Measurement using Lucas kanade Method

The Optical Flow Equation

Optical flow methods try to calculate motion between two image frames at time t and $t + \Delta t$.

Considering the pixel (x, y) at time t –

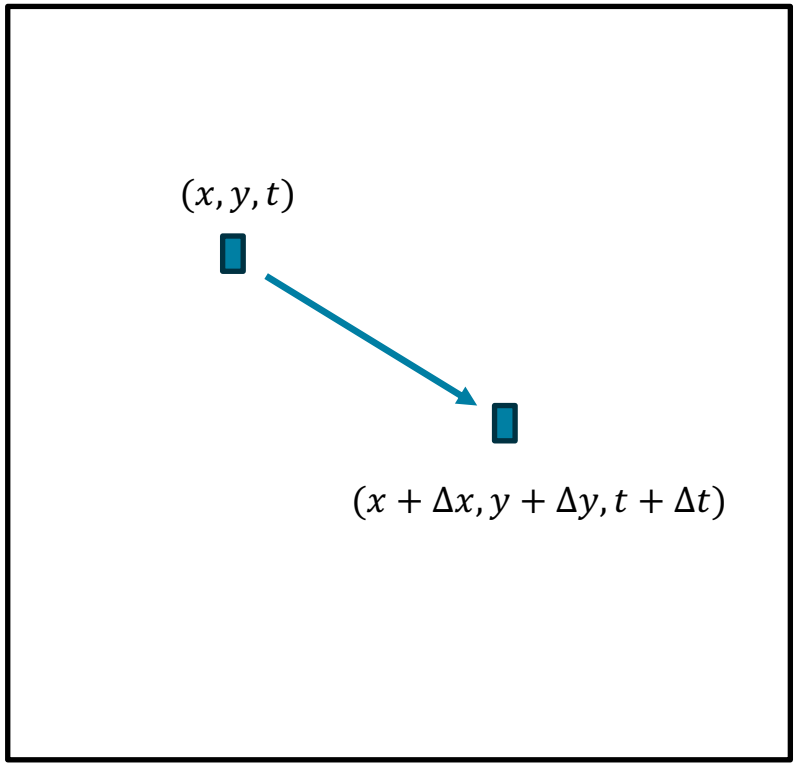
$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

Using the Taylor Series

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \cdots$$

By removing the higher order terms

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0$$



Source: Optical Flow Measurement using Lucas kanade Method

The Optical Flow Equation - Continued

Dividing by Δt

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0$$

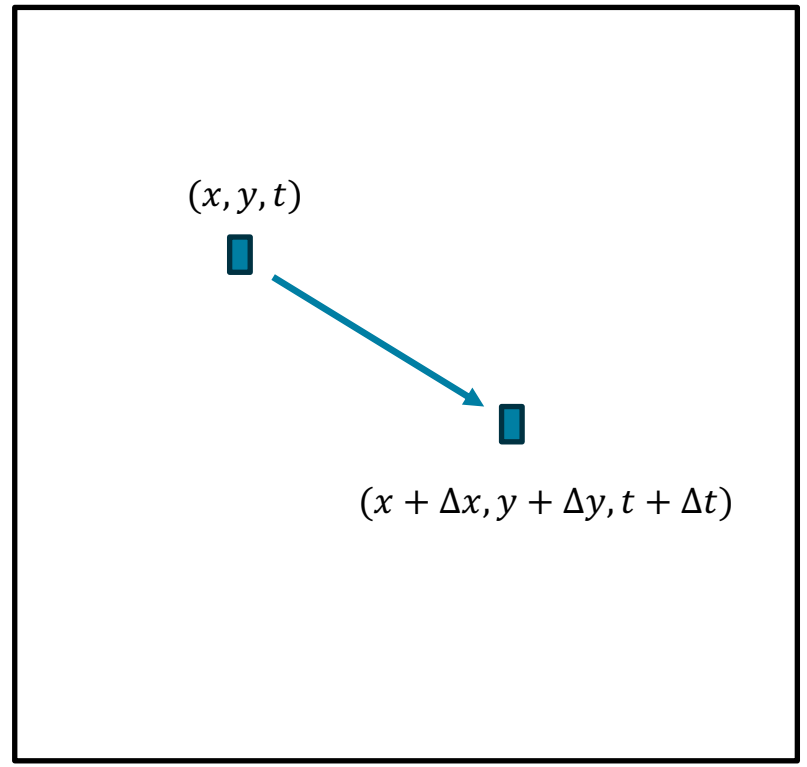
Giving

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

The values V_x and V_y are the x and y components of the velocity – optical flow of $I(x, y, t)$

The Equation

$$I_x V_x + I_y V_y = -I_t$$



Source: Optical Flow Measurement using Lucas kanade Method

The Lucas Kanade Method for Optical Flow

Assumes that all neighbouring pixels of (x, y) have similar motion
Take a patch of pixels $P_1, \ldots P_n$ around (x, y)

$$I_x\,(P_1)V_x + I_y(P_1)V_y = -I_t(P_1)$$
$$\vdots$$
$$I_x\,(P_n)V_n + I_y(P_n)V_y = -I_t(P_n)$$

Can be written as $Av = b$, where

$$\mathrm{A} = \begin{bmatrix} I_x\,(P_1) & I_y(P_1) \\ \vdots & \vdots \\ I_x\,(P_n) & I_y(P_n) \end{bmatrix}, v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, b = \begin{bmatrix} -I_t(P_1 \\ \vdots \\ -I_t(P_n) \end{bmatrix}$$

Source: Optical Flow Measurement using Lucas kanade Method

The Lucas Kanade Method for Optical Flow

Using least squares

$$A^t A v = A^t b$$
$$v = (A^t A)^{-1} A^t b$$

Giving

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(P_i)^2 & \sum_i I_x(P_i) I_y(P_i) \\ \sum_i I_x(P_i) I_y(P_i) & \sum_i I_y(P_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(P_i) I_t(P_i) \\ -\sum_i I_y(P_i) I_t(P_i) \end{bmatrix}$$

Source: Optical Flow Measurement using Lucas kanade Method

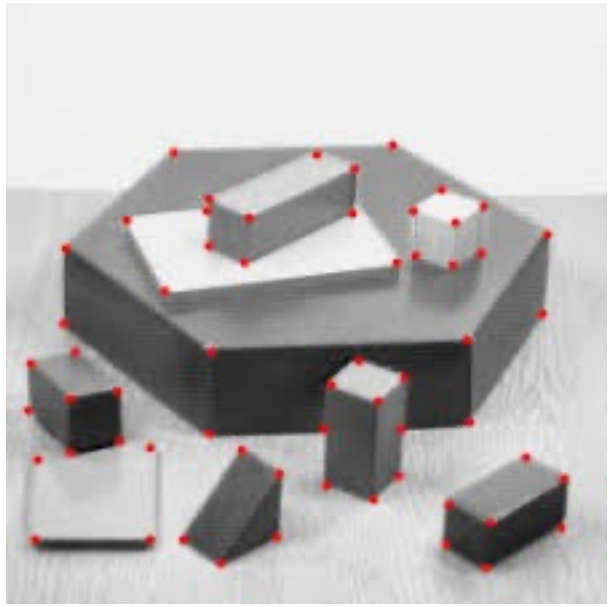
When is this Solvable ?

- A^tA should be invertible
- A^tA should not be too small - the eigenvalues λ_1 , λ_2 should not be too small
- A^tA should be well conditioned - $\frac{\lambda_1}{\lambda_2}$ should not be too large where λ_1 is the larger eigenvalue

Source: Lucas-Kanade Optical Flow

Corners !!!

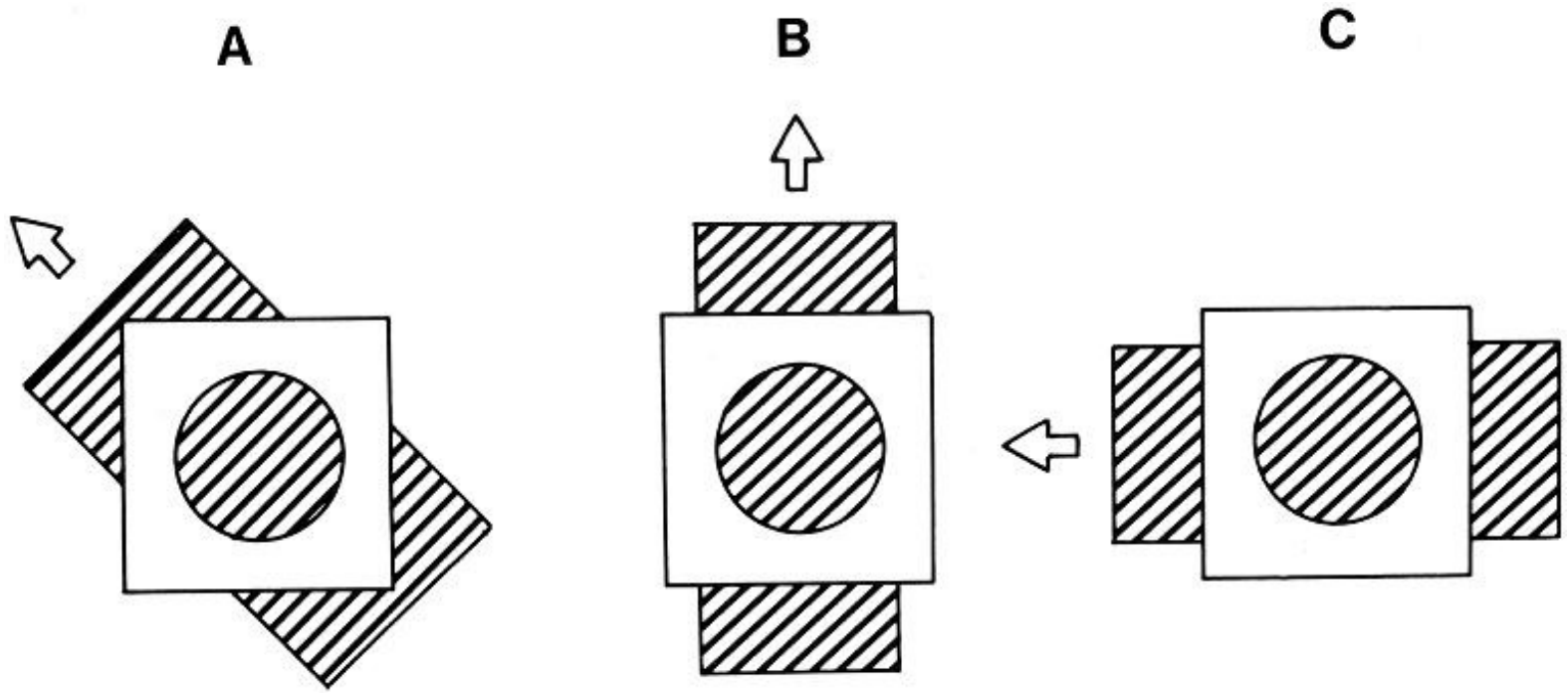
- Corners are when λ_1 , λ_2 are big
- Corners are regions with two different directions of gradient (at least)
- Corners are good places to compute optical flow



Source: Lucas-Kanade Optical Flow

Disadvantages

- Computationally expensive
- Aperture Problem



Source: Lucas-Kanade Optical Flow

Scale Invariant Feature Transform Tracking

Tracking features with a rotation and scale invariant algorithm

Source: Object Tracking with SIFT using OpenCV

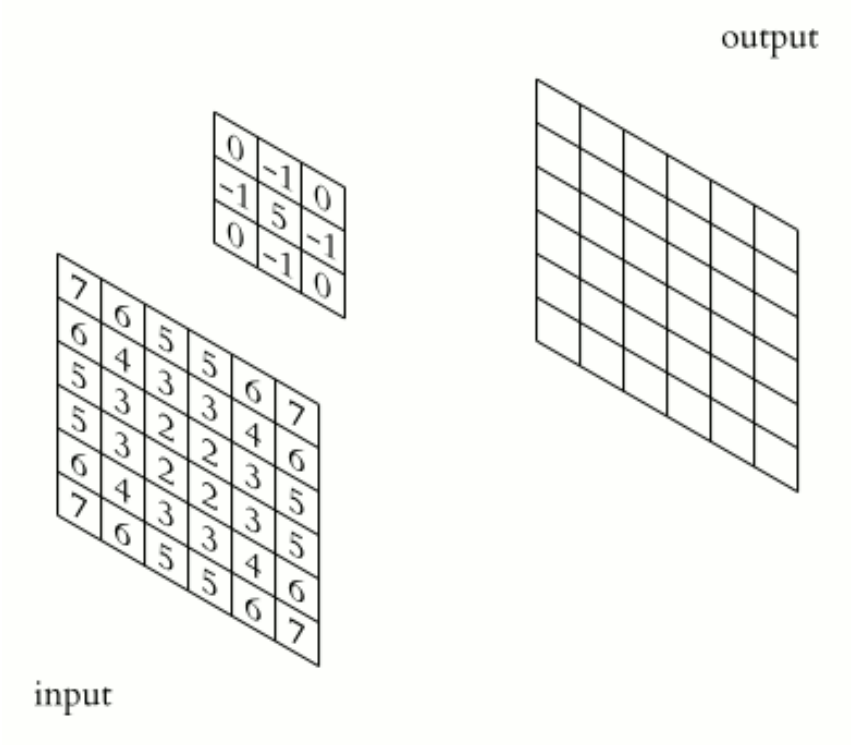
Scale Invariant Feature Transform Tracking

- Scale-space extrema detection: Potential location for finding features.
- Keypoint Localization: Accurately locating the feature keypoints.
- Orientation Assignment: Assigning orientation to keypoints.
- Keypoint descriptor: Describing the keypoints as a high dimensional vector.
- Keypoint Matching

Source: Distinctive Image Features from Scale-Invariant Keypoints

Applying a Filter – 2D Convolution

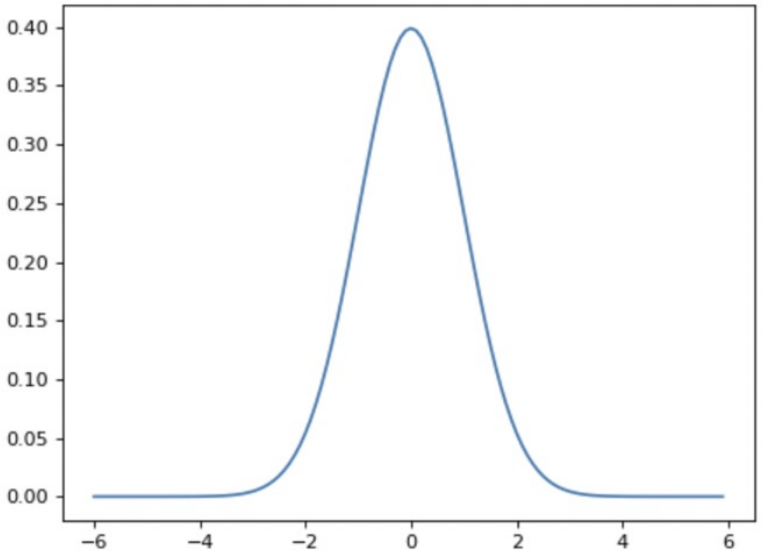
$$g(x,y) = \sum_i \sum_j h(i,j) f(x-i, y-j)$$

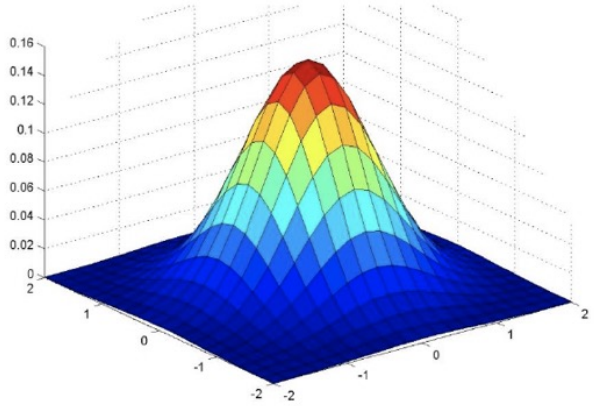


Source: Wikipedia

The Gaussian Filter

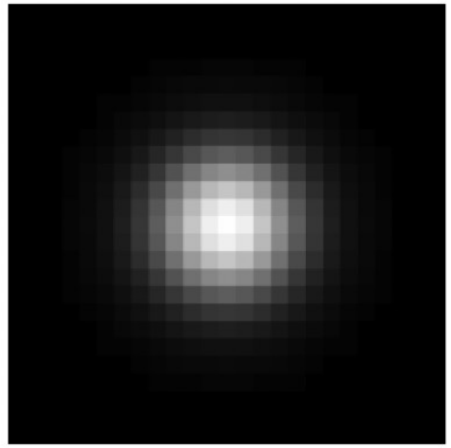
Produces a blurring effect and can be controlled by changing the sigma value





Gaussian kernel

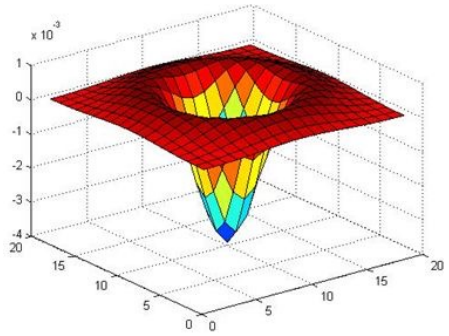
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

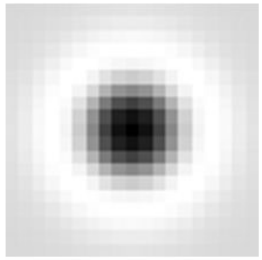


Source: Image Processing Medium Blog

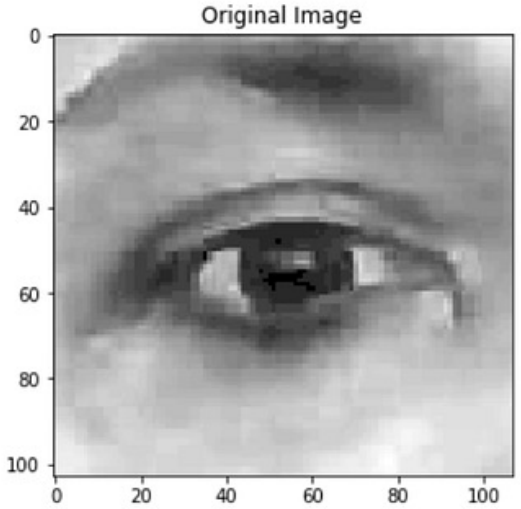
The normalised Laplacian of the Gaussian Filter

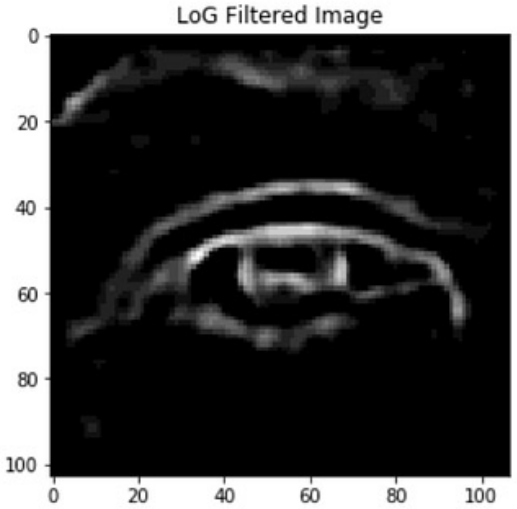
Highlights regions of rapid intensity change





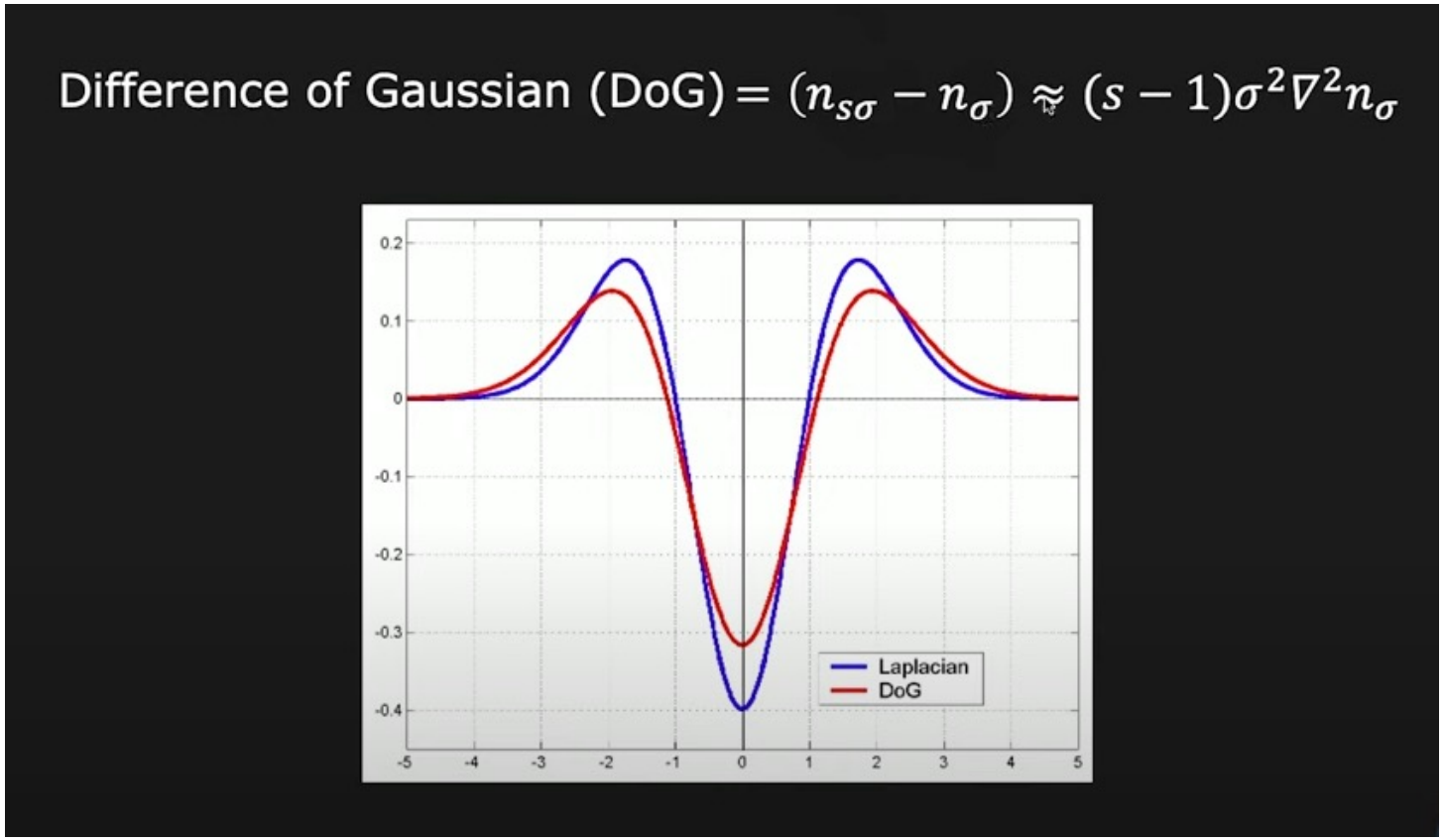
Scale-normalized: $\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$





The Difference of Gaussian Filter

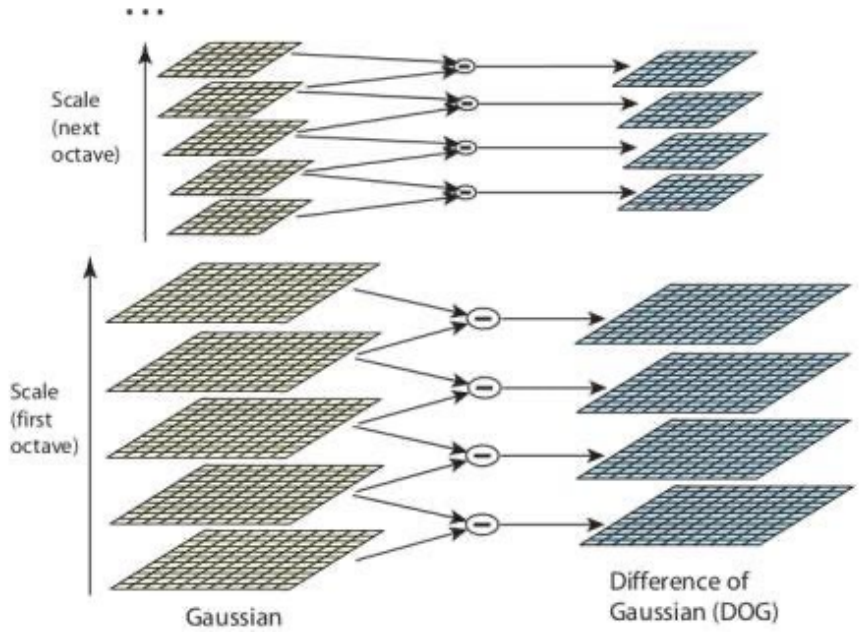
This filter is easier to compute and is a good approximation of the nLOG



Scale Space extrema detection

The image is blurred using the gaussian filter with different sigma values to get a Gaussian Scale Space – multiple images blurred at different “levels”

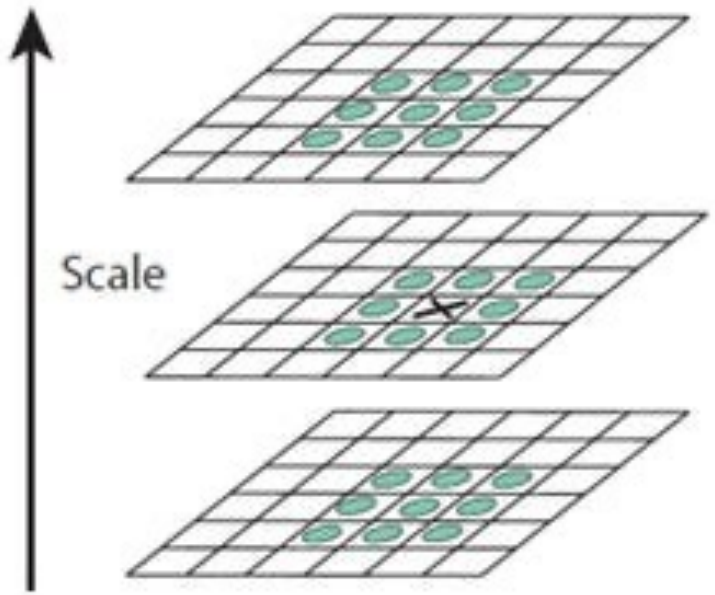
These images are then subtracted to calculate the difference of gaussians



Source: Distinctive Image Features from Scale-Invariant Keypoints

Scale Space extrema detection

Keypoints are found by searching a 3x3x3 neighbourhood of pixels across scale and space to find extrema



Source: Distinctive Image Features from Scale-Invariant Keypoints

Keypoint Localization

Interest points are filtered out using a threshold on their intensities (< 0.03)

The DOG has a high response to edges and these must be filtered out as well. This is done using a 2x2 Hessian Matrix

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

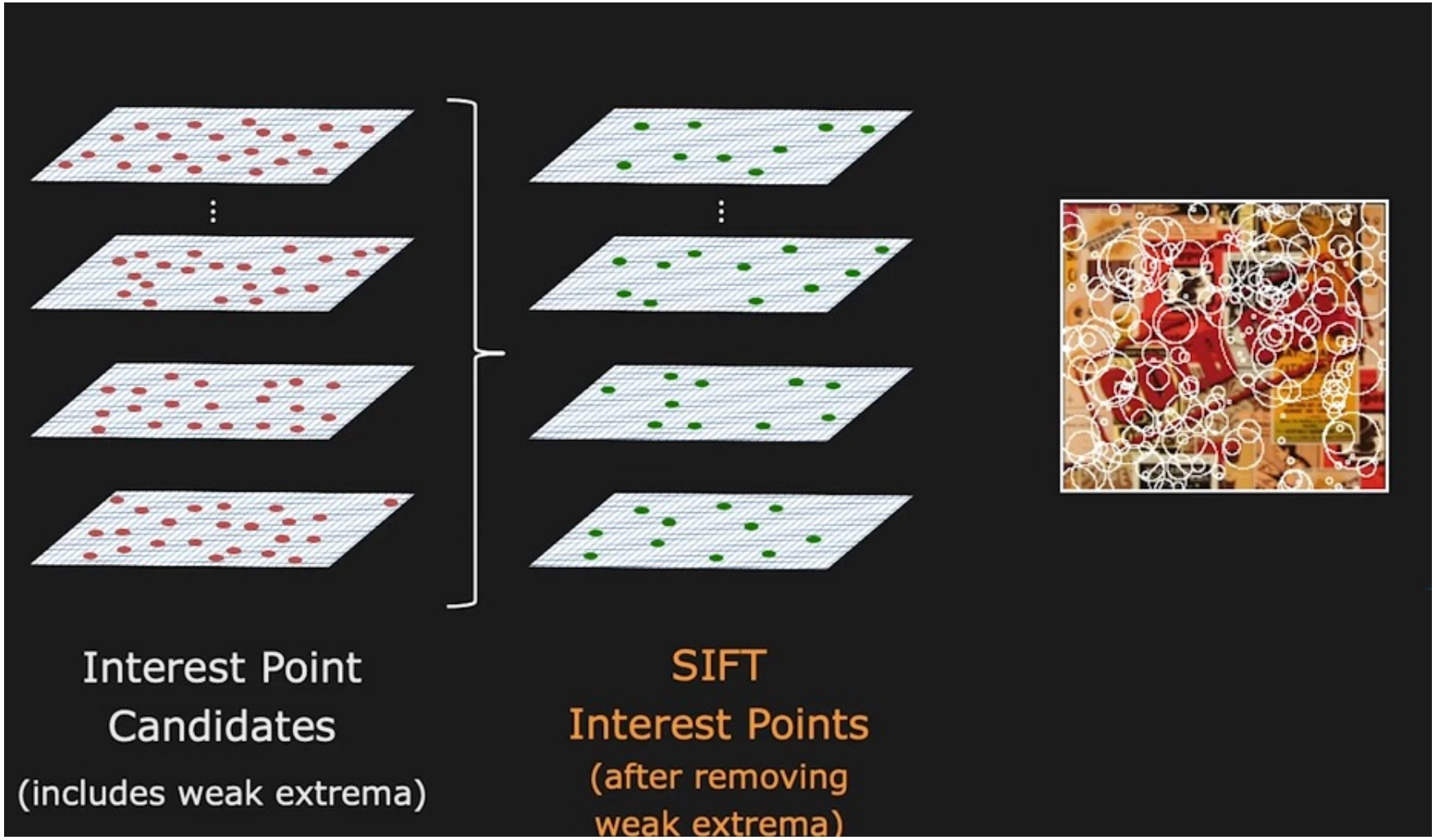
Let a and b be eigenvalues of H with a $\geq b$

$$Tr(H) = a + b$$
$$Det(H) = ab$$
$$\frac{Tr(H)^2}{Det(H)} = \frac{(a+b)^2}{ab} = \frac{(r+1)^2}{r}$$

Where $r = \frac{a}{b}$ and values of $r < 10$ are kept.

Source: Distinctive Image Features from Scale-Invariant Keypoints

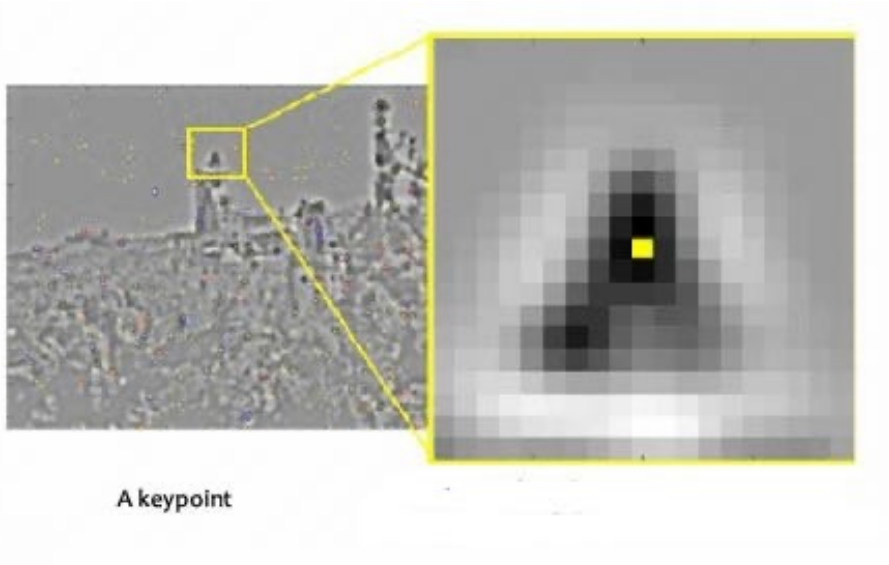
Keypoint Localization

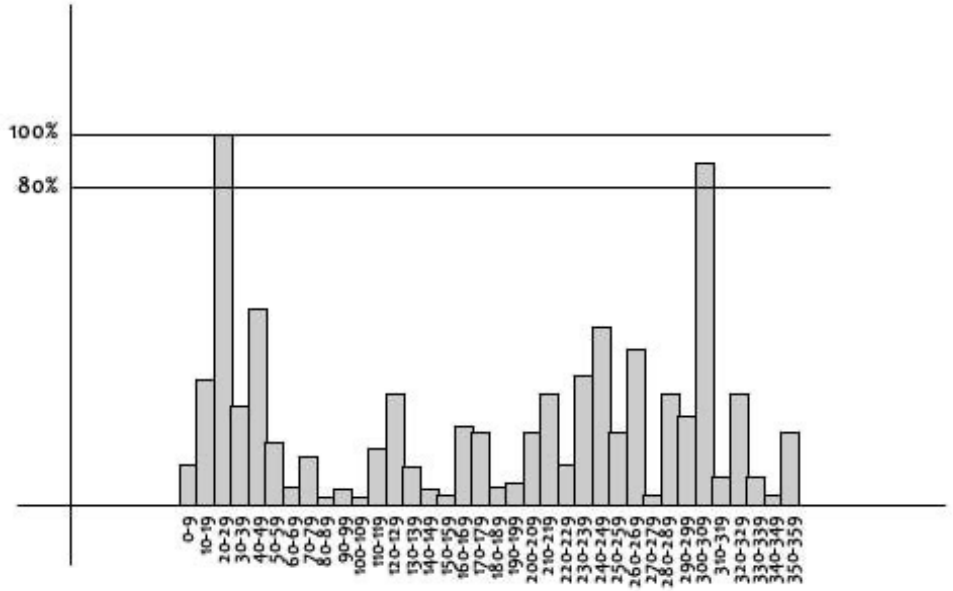


Orientation Assignment

A neighbourhood is taken around the keypoint location depending on the scale and the gradient magnitude and direction is calculated in that region. This is then plotted in a histogram with 36 bins covering 360 degrees.

The direction with the highest number of pixels is chosen as the principal orientation

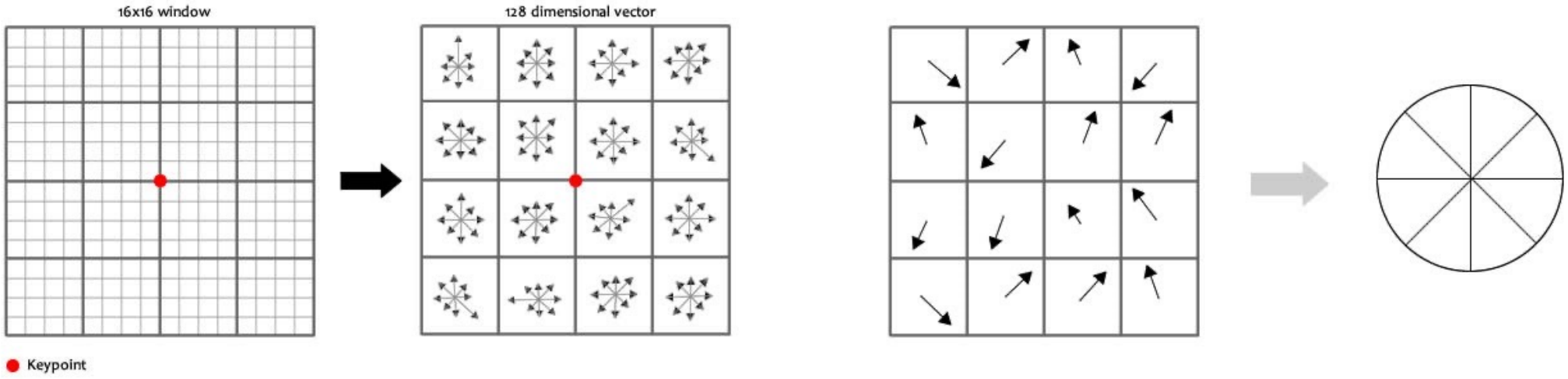




Source: Distinctive Image Features from Scale-Invariant Keypoints

Keypoint Descriptor

- A vector/signature that “uniquely” defines each SIFT feature
- A 16x16 window is taken around the keypoint and is divided into 16 blocks of size 4x4
- For each block, an 8 bin orientation histogram is created
- This gives use 128 values, and is represented as a feature vector



Source: Distinctive Image Features from Scale-Invariant Keypoints

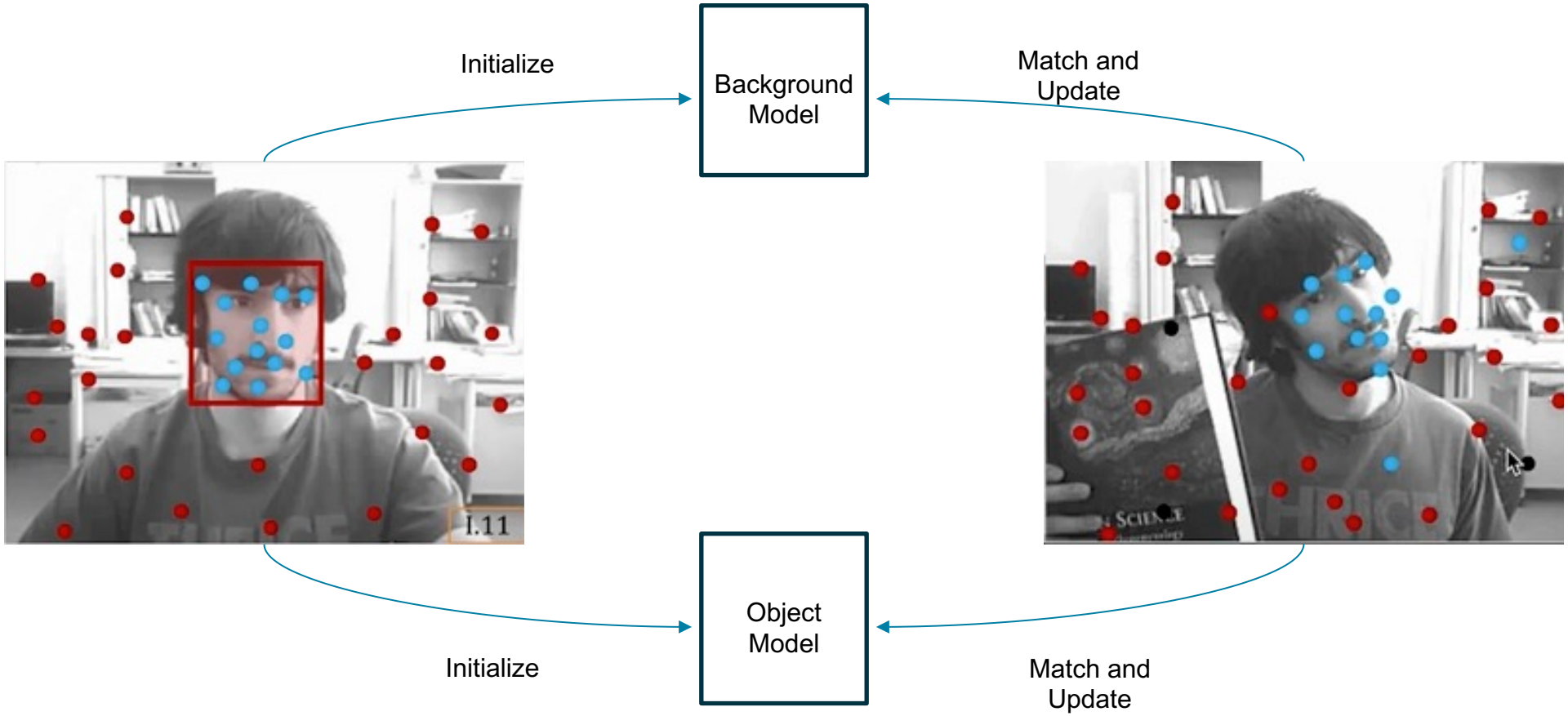
Keypoint Matching

Keypoint descriptors are matched using their Euclidean distance with the nearest neighbours algorithm

UNIVERSITY OF TORONTO | CVDG

Source: Feature Extraction and Matching of Slam Image Based on Improved SIFT Algorithm

Tracking by Feature Detection

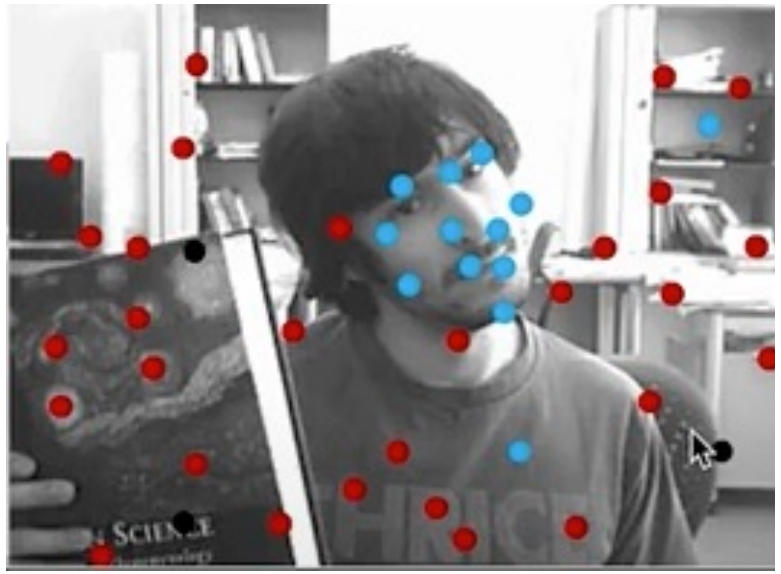


Source: [Tracking by Feature Detection | Object Tracking]

Tracking by Feature Detection

At frame t

1. Compute the SIFT descriptors
2. For each SIFT descriptor – compute the distance with best match from the object set and background set
3. $$C(v_i) = \begin{cases} +1 & if\ \frac{d_o}{d_b} < 0.5 \\ -1 & otherwise \end{cases}$$



Source: Tracking by Feature Detection | Object Tracking

Tracking by Feature Detection

For each search window W

1. Compute $\varphi(W) = \sum C(v_i)$ for al features v_i inside W
2. Compute the heuristic $\tau(W, W_{t-1})$

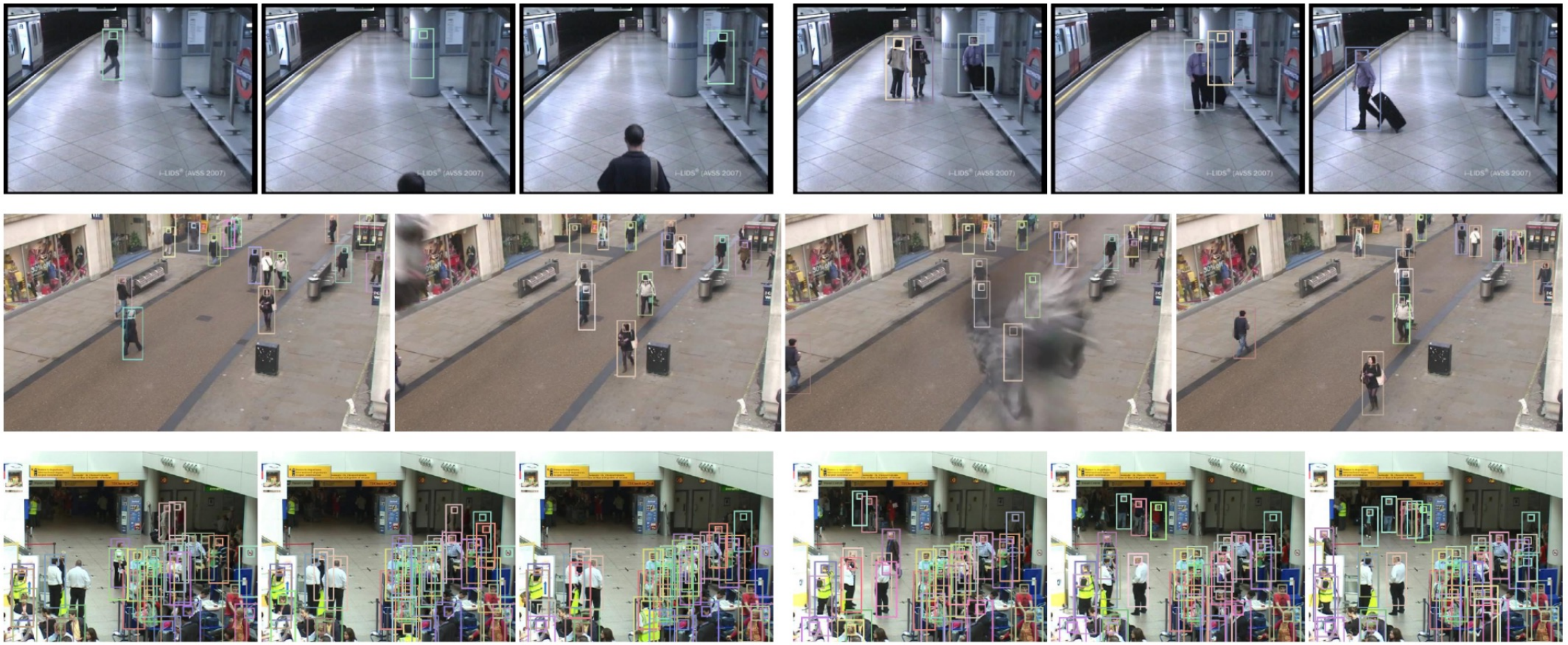
$$\tau(W, W_{t-1}) = \gamma(\left|\left|O_1 - O_2\right|\right| + |h_1 - h_2| + |w_1 - w_2| + s(W, W_{t-1})$$

$$s(W, W_{t-1}) = \max\{\left|\frac{h_1}{w_1} - \frac{h_2}{w_2}\right|, \left|\frac{w_1}{h_1} - \frac{w_2}{h_2}\right|\}$$

3. Compute the Match Score - $\mu(W) = \varphi(W) - \tau(W, W_{t-1})$
4. Select the window with the best match score
5. Update the object model with newly detected feature points

Source: Tracking by Feature Detection | Object Tracking

Results

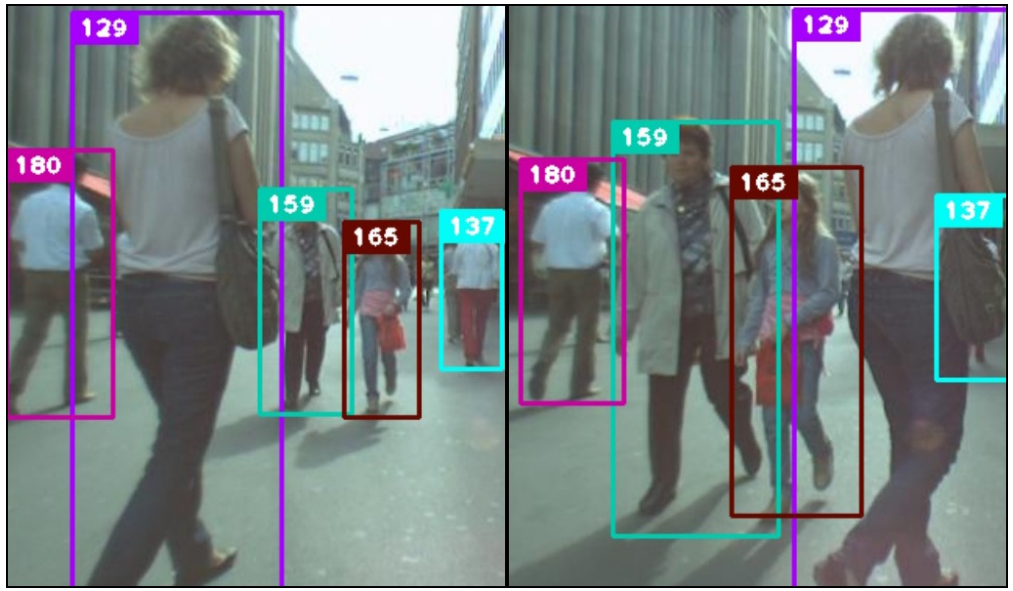


UNIVERSITY OF TORONTO | CVDG

Source: Stable Multi-Target Tracking in Real-Time Surveillance Video

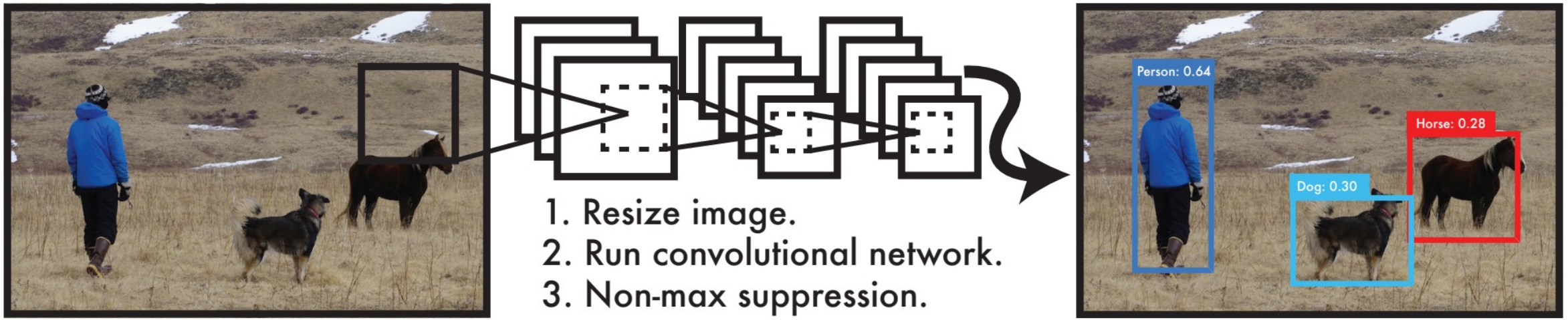
Simple Online and Real-Time Tracking (SORT)

- Detecting objects of interest
- Estimating object states into future frames
- Associating current detection’s with existing objects
- Managing the lifespan of tracked objects



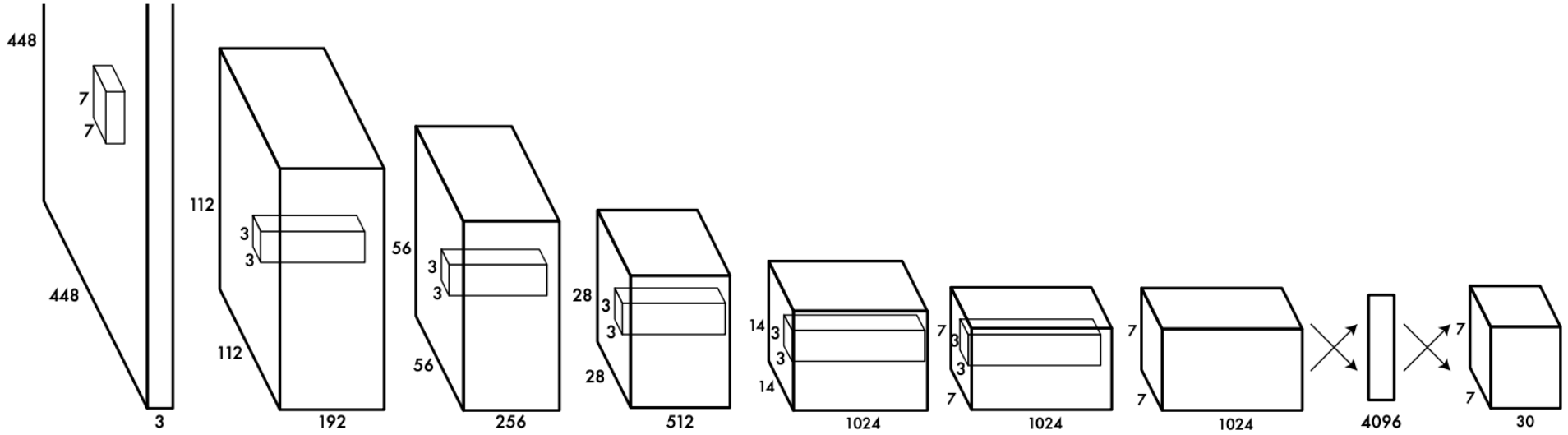
Source: SIMPLE ONLINE AND REALTIME TRACKING

Detection - YOLO



Source: [You Only Look Once: Unified, Real-Time Object Detection]

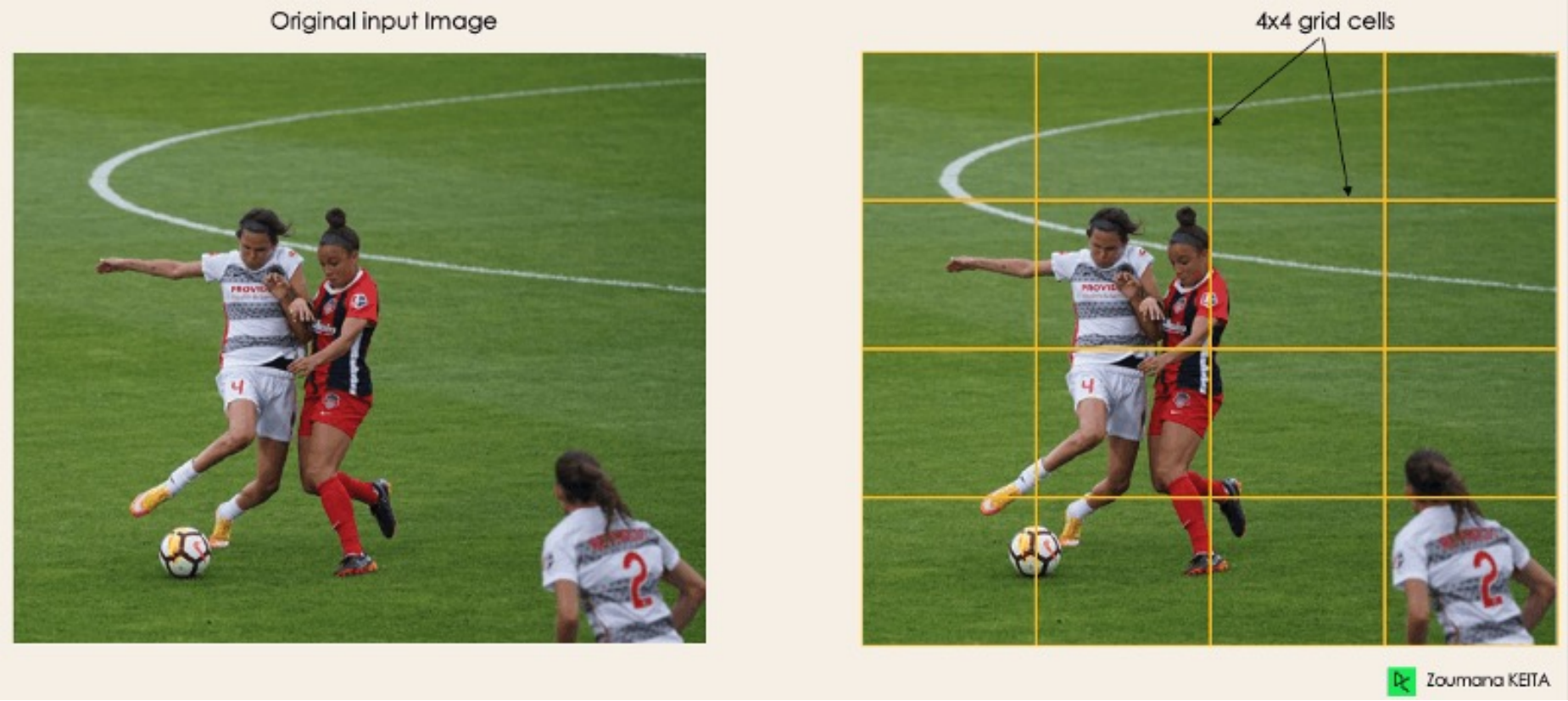
Detection - YOLO



Source: You Only Look Once: Unified, Real-Time Object Detection

Detection - YOLO

Residual Blocks



Source: [YOLO Object Detection Explained]

Detection - YOLO

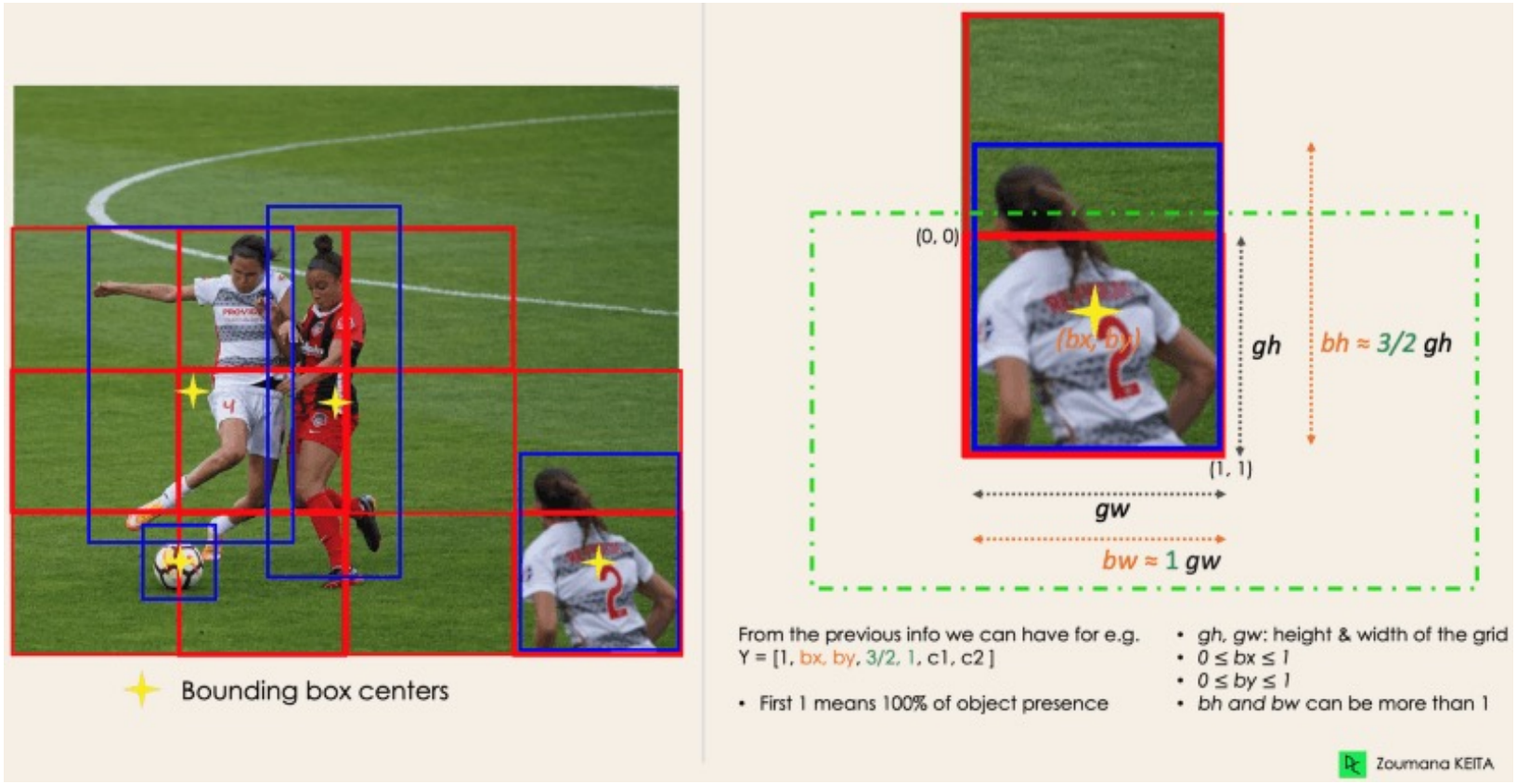
Bounding Box Regression - [p, x, y, h, w, c1, c2, …]

- p is the probability of an object contained in the grid
- (x, y, w, h) is the location, height and width
- (c1, c2, …) the presence of classes in the grid

Source: YOLO Object Detection Explained

Detection - YOLO

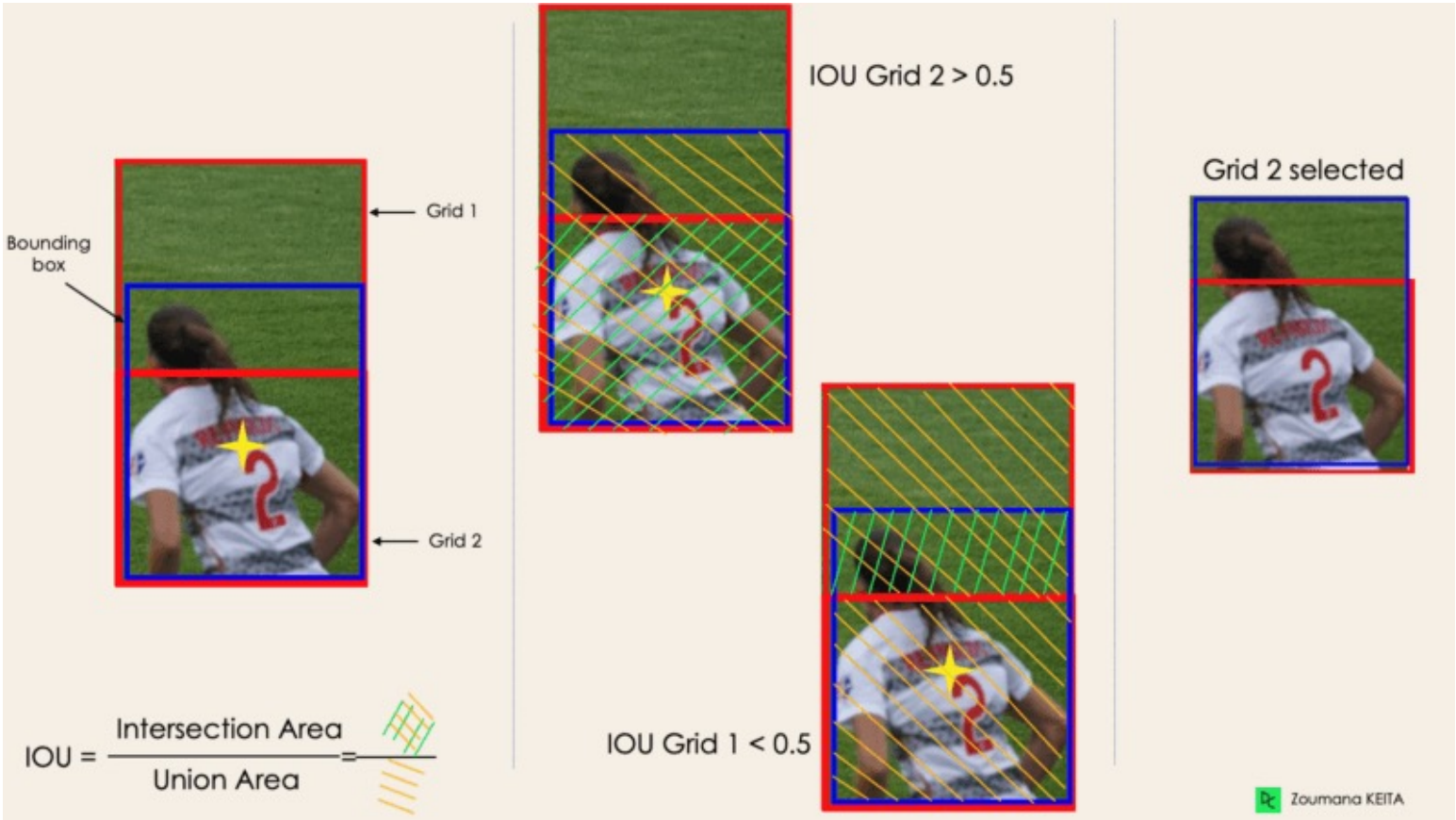
Bounding Box Regression



Source: YOLO Object Detection Explained

Detection - YOLO

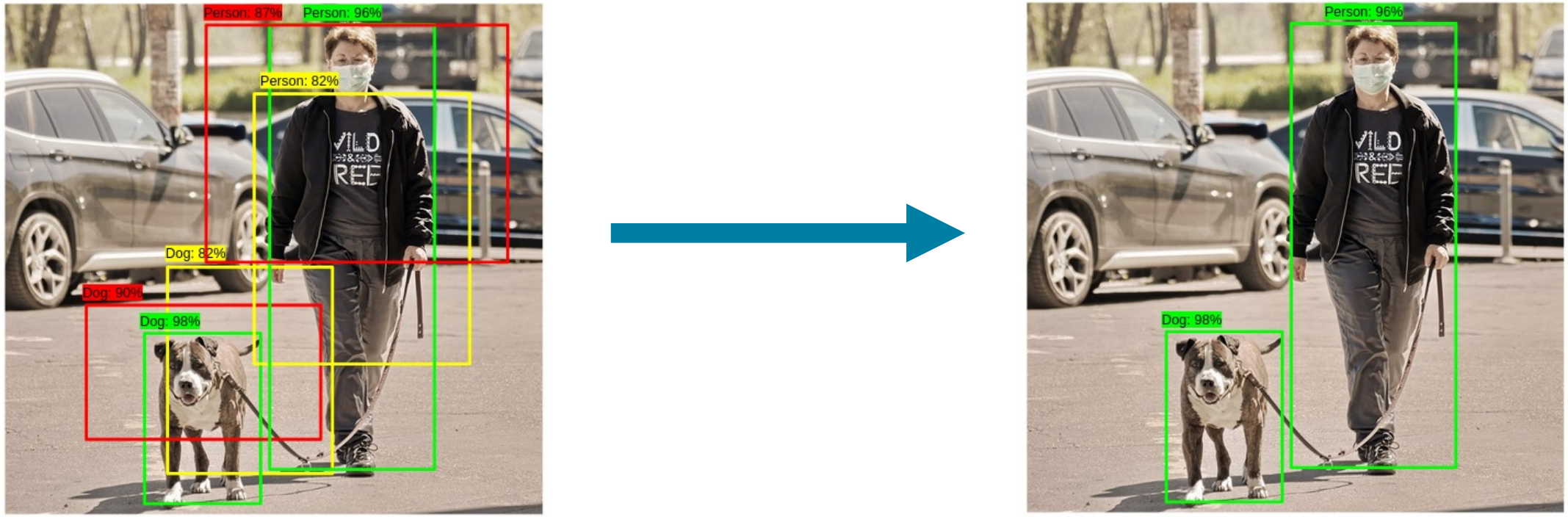
Intersection over Union – Choose boxes with high IOU



Source: YOLO Object Detection Explained

Detection - YOLO

Non-Max Suppression



Estimation – Kalman Filter

- Define an 8 dimensional state space – (x, y, a, h, x’, y’, a’, h’)
- (x,y) is the object’s position, a is the aspect ratio, h is the height, and (x’, y’, a’, h’) are their respective velocities
- The Kalman Filter predicts this vector for an upcoming frame with the current state using a constant velocity motion and a linear measurement model.

Source: SIMPLE ONLINE AND REALTIME TRACKING

Kalman Filter – One Dimensional Case

Consider a car moving on a straight road. It's state vector is

$$x = \begin{bmatrix} position \\ velocity \end{bmatrix}$$

We can calculate the new position of car with

$$x = x_i + v\Delta t$$

This can be modeled with a state transition matrix $F = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$

$$x_{pred} = F \cdot x_{prev}$$

Source: KalmanFilter.net

Kalman Filter – Prediction Step

The uncertainty in our prediction can be calculated with a covariance matrix P, which is initially set to

$$P = \begin{bmatrix} {\sigma^2}_x & Cov(x,v) \\ Cov(x,v) & {\sigma^2}_v \end{bmatrix}$$

The uncertainties are also affected by noise which can be modelled with a process noise covariance matrix Q.

P is updated using

$$P_{pred} = F \cdot P_{prev} \cdot F^T + Q$$

Source: KalmanFilter.net

Kalman Filter – Update Step

We adjust our prediction based on a new measurement of the car's position - z

Let our prediction be $H \cdot x_{pred}$, Where $H = [0 \quad 1]$ to chose the position prediction.

The **Innovation** or the residual is $z - H \cdot x_{pred}$

The **Kalman Gain**, K, determines how much we should adjust our prediction based on the measurement

$$K = P_{pred} \cdot H^T \cdot (H \cdot P_{pred} \cdot H^T + R)^{-1}$$

The state and covariance matrix are updated by

$$x_{new} = x_{pred} + K \cdot innovation = (I - K \cdot H) \cdot x_{pred} + K \cdot z$$

$$P = (I - K \cdot H) \cdot P_{pred}$$

Source: KalmanFilter.net

Association

The bounding boxes predicted using the Kalman filter from the previous frame need to be matched with the detections from the current frame.

The assignment cost matrix is computed using intersection over union.

Source: SIMPLE ONLINE AND REALTIME TRACKING

Association – The Hungarian Algorithm

1. Subtract the smallest entry in each row from all the other entries in the row. This will make the smallest entry in the row now equal to 0.
2. Subtract the smallest entry in each column from all the other entries in the column. This will make the smallest entry in the column now equal to 0.
3. Draw lines through the row and columns that have the 0 entries such that the fewest lines possible are drawn.
4. If there are n lines drawn, an optimal assignment of zeros is possible and the algorithm is finished. If the number of lines is less than n, then the optimal number of zeroes is not yet reached. Go to the next step.
5. Find the smallest entry not covered by any line. Subtract this entry from each row that isn't crossed out, and then add it to each column that **is** crossed out. Then, go back to Step 3.

Association – The Hungarian Algorithm

We want to solve a maximization problem so before performing the algorithm, we subtract all entries from the largest entry in the matrix making it a minimization problem.

Runtime -

1. Row Reduction - $O(n^2)$
2. Column Reduction - $O(n^2)$
3. Test for Optimal Assignment – $\mathrm{n} \cdot O(n^2) = O(n^3)$
4. Shift Zeros – $\mathrm{n} \cdot O(n^2) = O(n^3)$
5. Final Assignment - $O(n)$

Managing Lifespan of Tracklets

1. Unique identities for objects are created or destroyed as they enter and leave the image, with untracked objects indicated by detections having an overlap less than the minimum Intersection over Union (IOU) threshold.

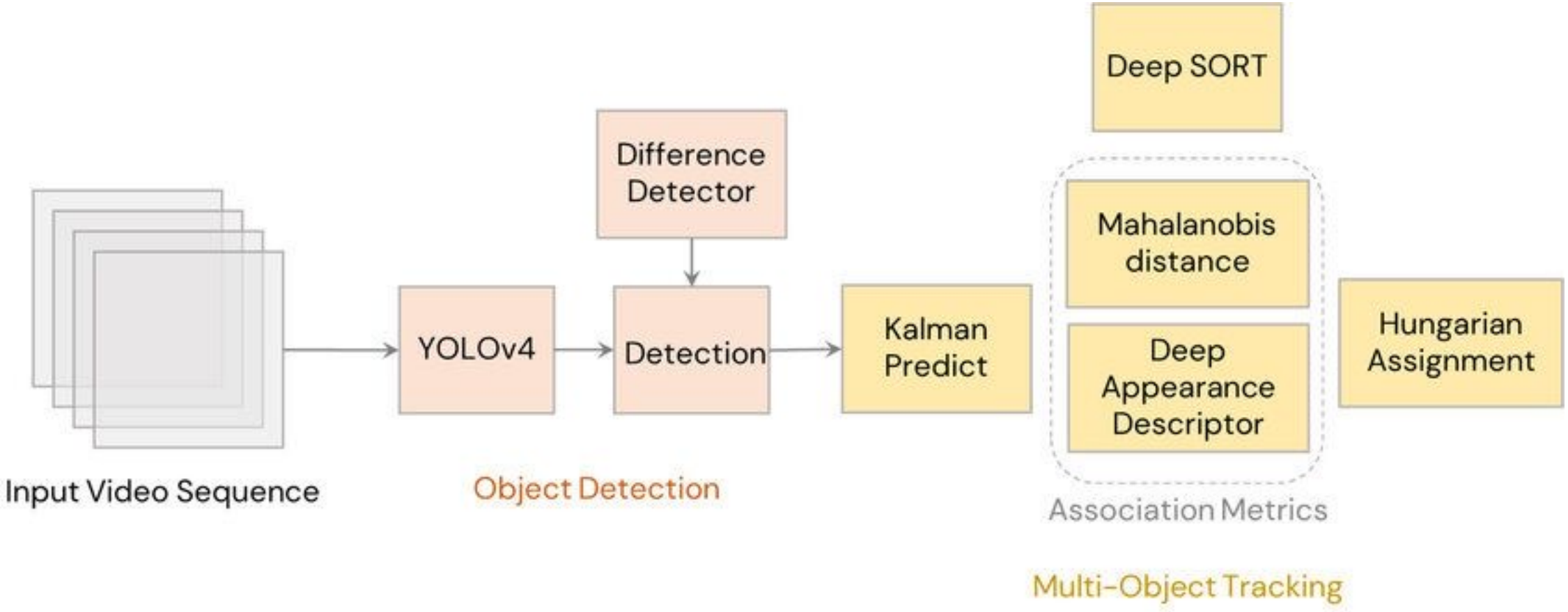
2. Trackers are initialized using the bounding box geometry, starting with a velocity of zero and large covariance values to reflect uncertainty about the velocity.

3. A probationary period is required for new trackers, during which they must be associated with detections to gather sufficient evidence and avoid false positives.

4. Tracks are terminated after being undetected for a specified number of frames (T Lost) to prevent excessive tracker growth and localization errors from prolonged predictions without detector corrections.

Source: SIMPLE ONLINE AND REALTIME TRACKING

Simple Online Realtime Tracking with a Deep Association Metric - DeepSort



Source: Understanding Multiple Object Tracking using DeepSORT

Improvements in DeepSORT

Assignment – Mahalanobis distance is used between predicated Kalman states and new detections.

$$d^1(i,j) = (d_j - y_i)^t S_i^{-1} (d_j - y_i)$$

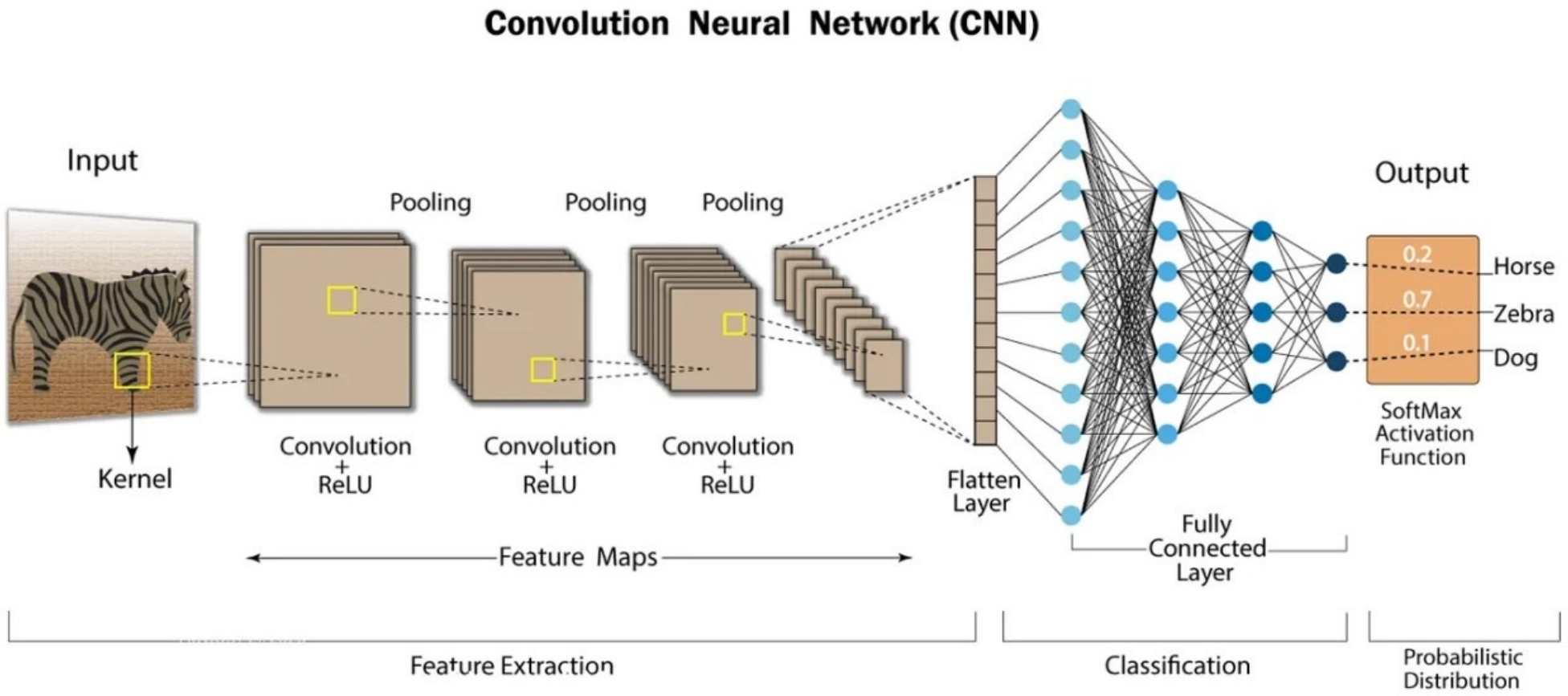
S_i is the covariance matrix.

Unlikely Associations are discarded by thresholding this distance at a 95% confidence interval from χ^2

Source: Understanding Multiple Object Tracking using DeepSORT

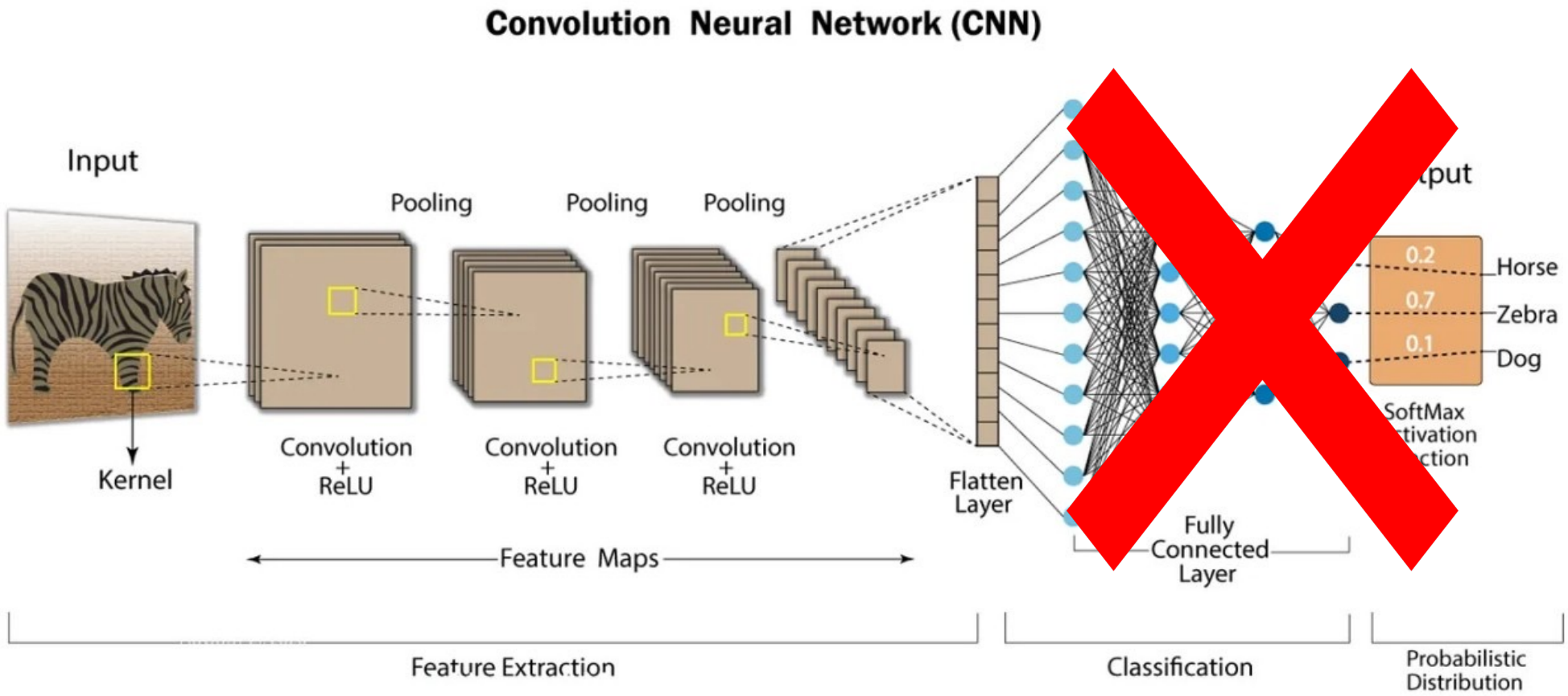
Improvements in DeepSORT

Deep Appearance Descriptor



Improvements in DeepSORT

Deep Appearance Descriptor



Improvements in DeepSORT

Deep Appearance Descriptor – Cosine Similarity

$$d^2(i,j) = \min\{1 - r_j \cdot r_k{}^{(i)} | r_k{}^{(i)} \in R_i\}$$

Where R_i is a list of the last n appearance descriptors for each track i.

Again, unlikely associations are discarded using a threshold.

Source: Understanding Multiple Object Tracking using DeepSORT

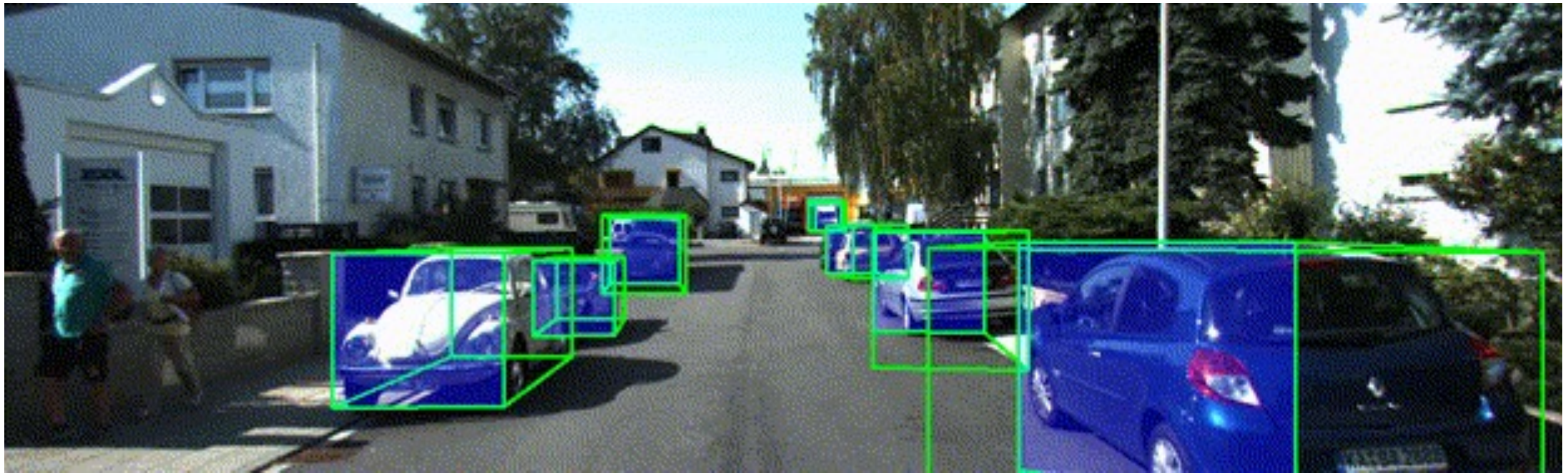
Improvements in DeepSORT

The final cost for each association

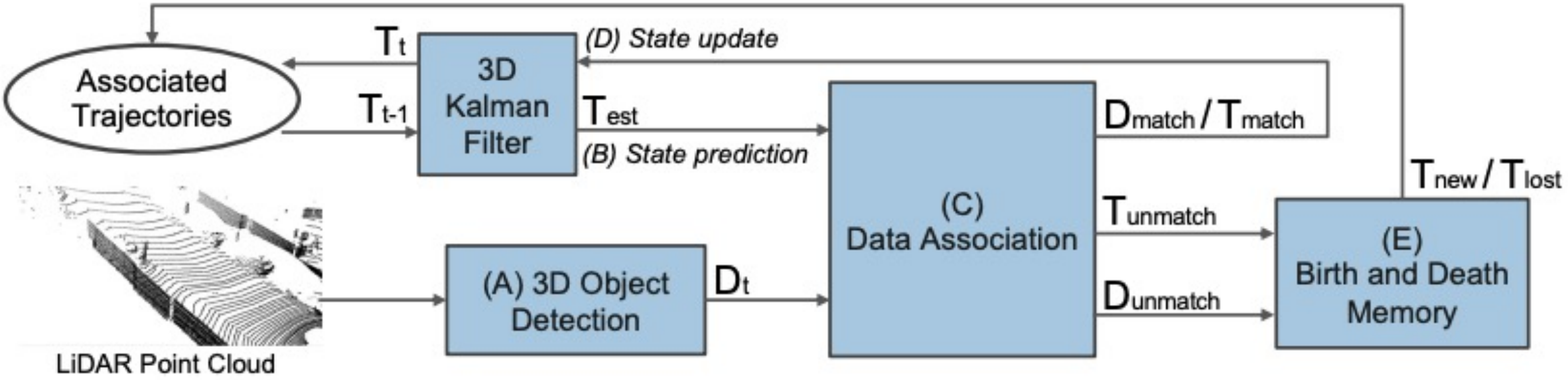
$$c_{i,j} = \lambda d^1(i,j) + (1-\lambda) d^2(i,j)$$

Source: Understanding Multiple Object Tracking using DeepSORT

3D Object Tracking



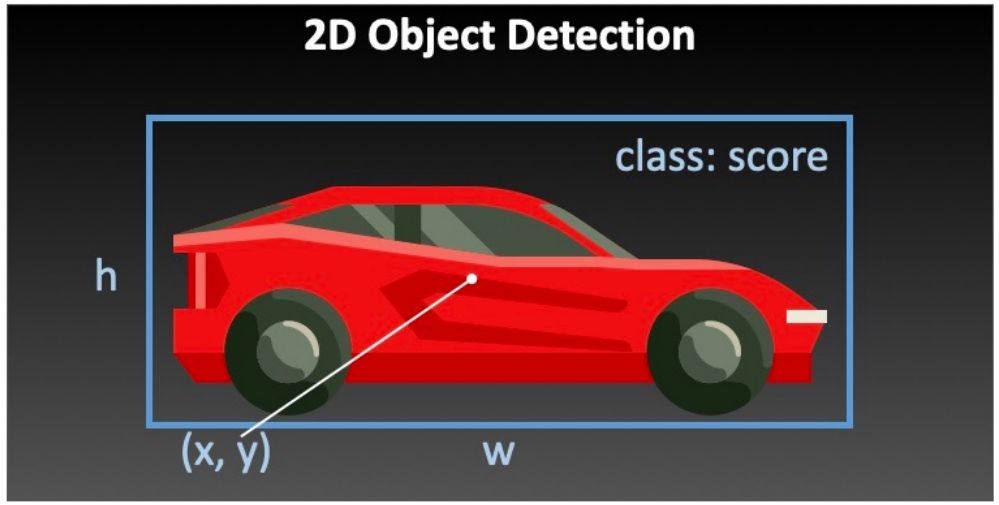
3D Object Tracking – How does the problem translate ?

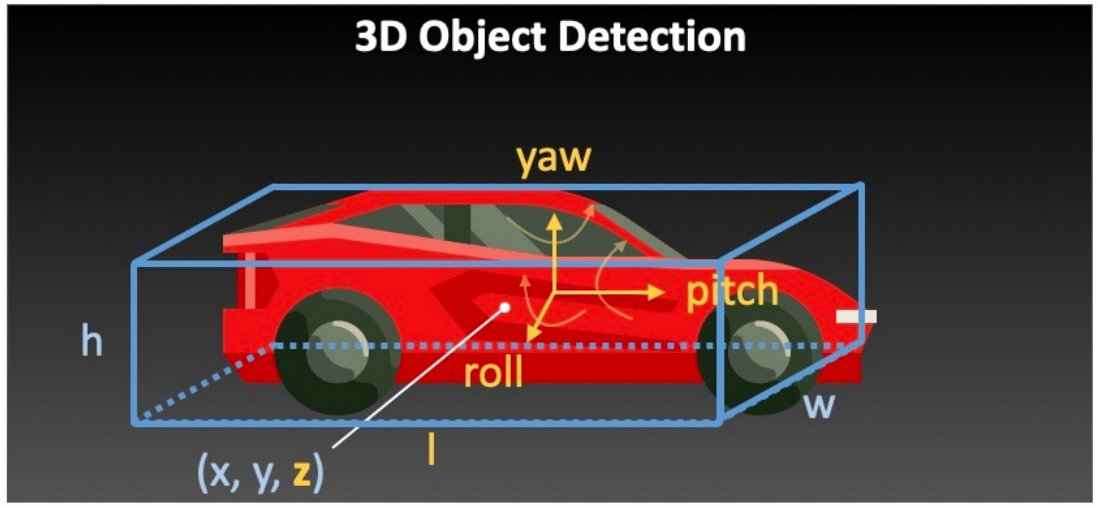


Source: 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics

3D Object Tracking – How does the problem translate ?

3D Object Detection – We need (x,y,z) coordinates, (h,w,l) values as well as yaw, pitch and roll angles.





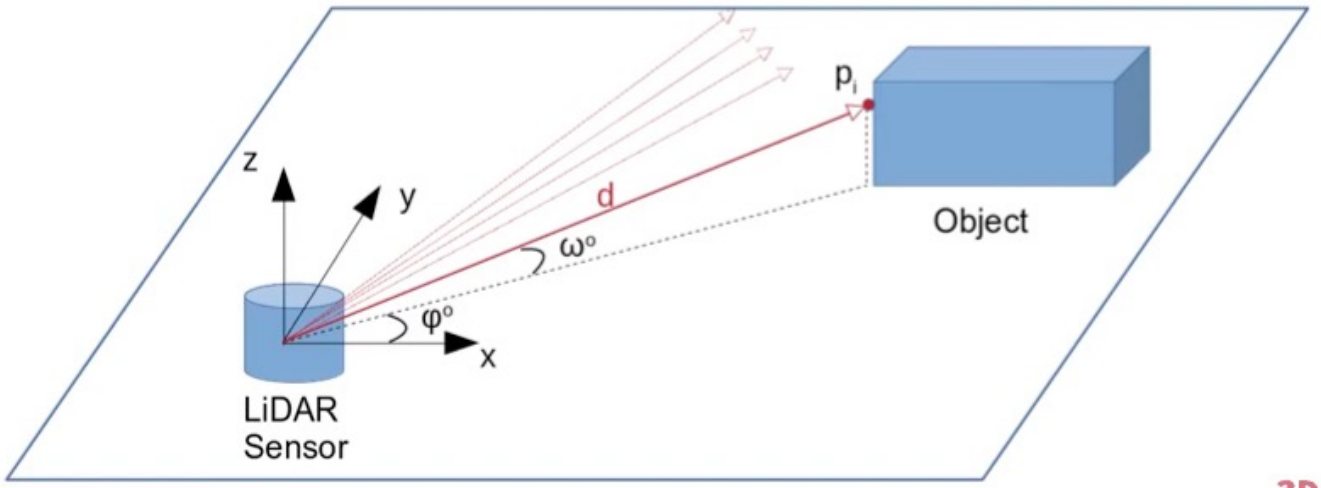
Source: An introduction to 3D Object Tracking (Advanced)

LiDAR – Light Detection and Ranging – Object Detection

Computes a set of 3D points – point cloud – from the time used to bounce light waves for each angle

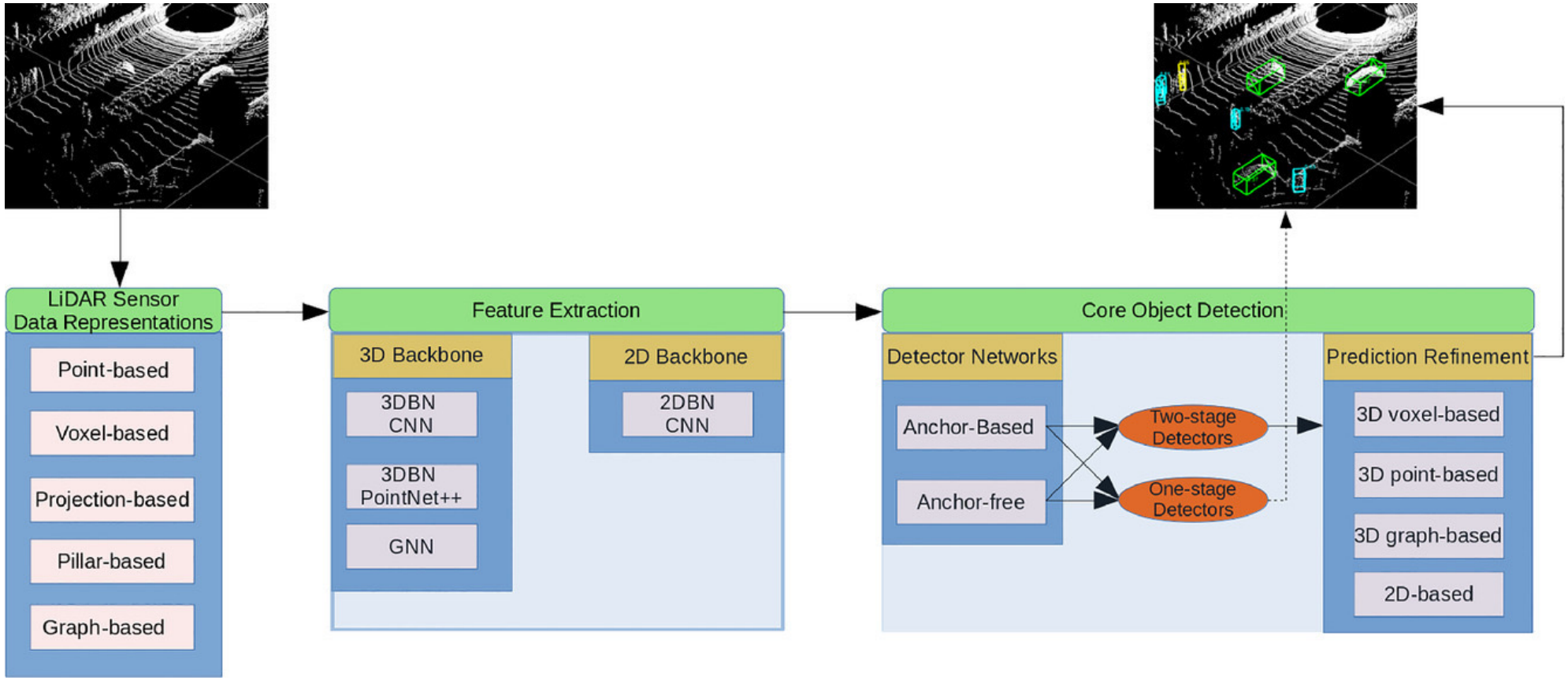
$$d = c\,(t_2 - t_1) / 2$$

$$x = d * \cos(\omega) * \cos(\varphi)$$
$$y = d * \cos(\omega) * \sin(\varphi)$$
$$z = d * \sin(\omega)$$



3D OR

LiDAR – Light Detection and Ranging – Object Detection

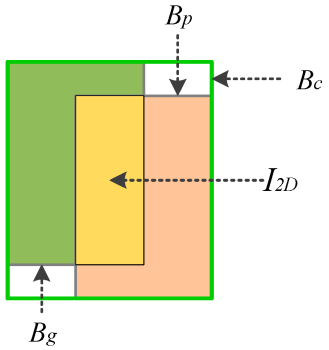


Source: A Comprehensive survey of LIDAR base detection models

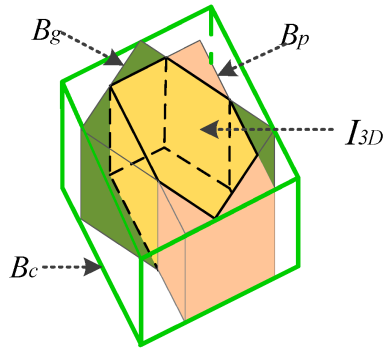
We have the bounding boxes – now what ?

Run the tracking algorithm in 3D with two changes –

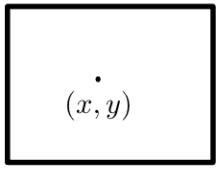
- The Intersection over Union calculation
- The Kalman Filter



(a) 2D

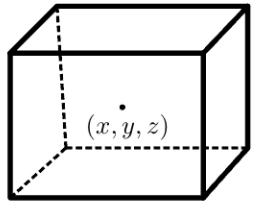


(b) 3D



$$\mu = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\sigma = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$



$$\mu = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\sigma = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Questions

Thank You

Follow us on social media

UNIVERSITY OF TORONTO

CVDG