VIT-AP UNIVERSITY

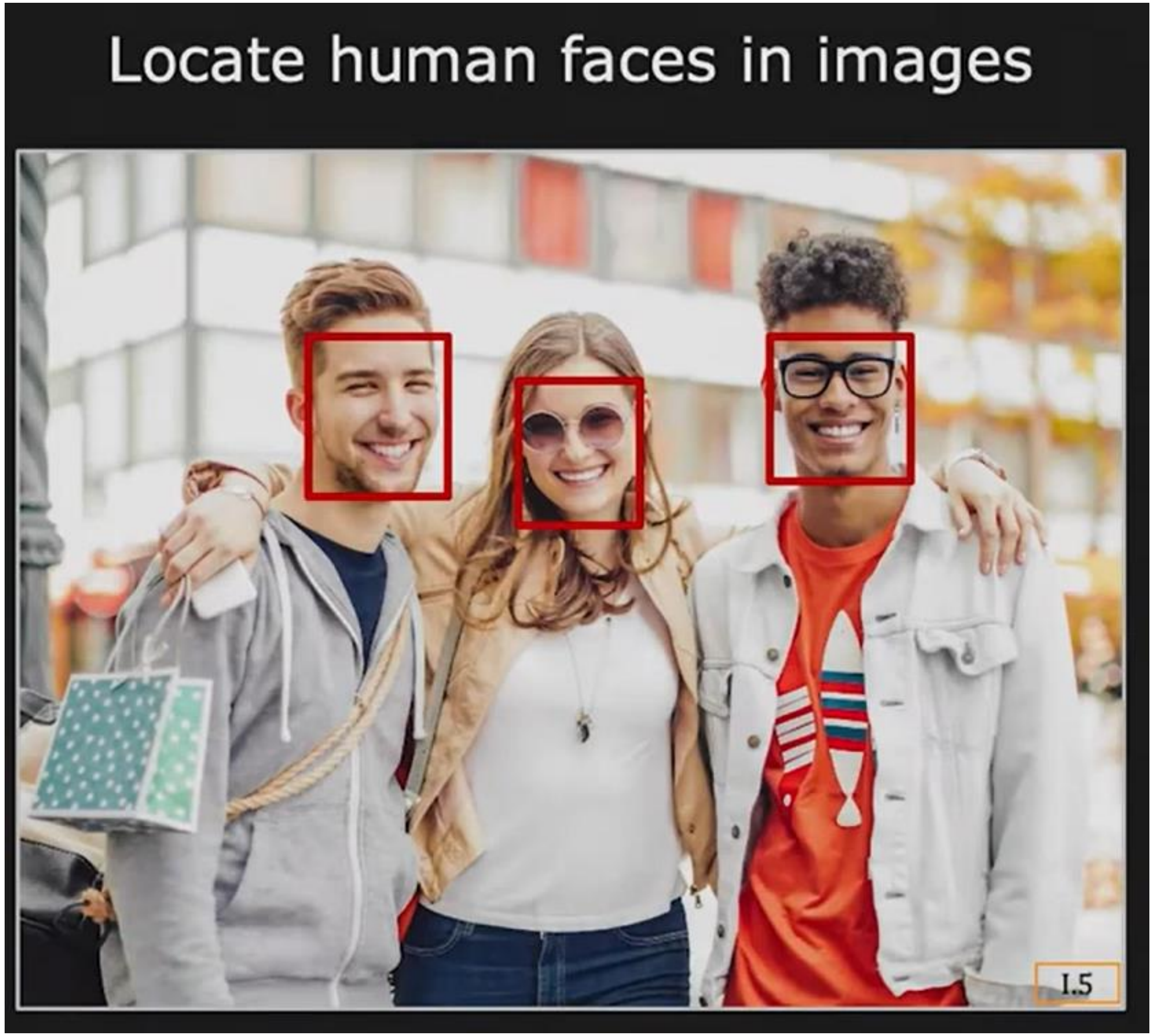
Computer Vision

(Course Code: 4047)

Module-4:Lecture-1: Face Detection

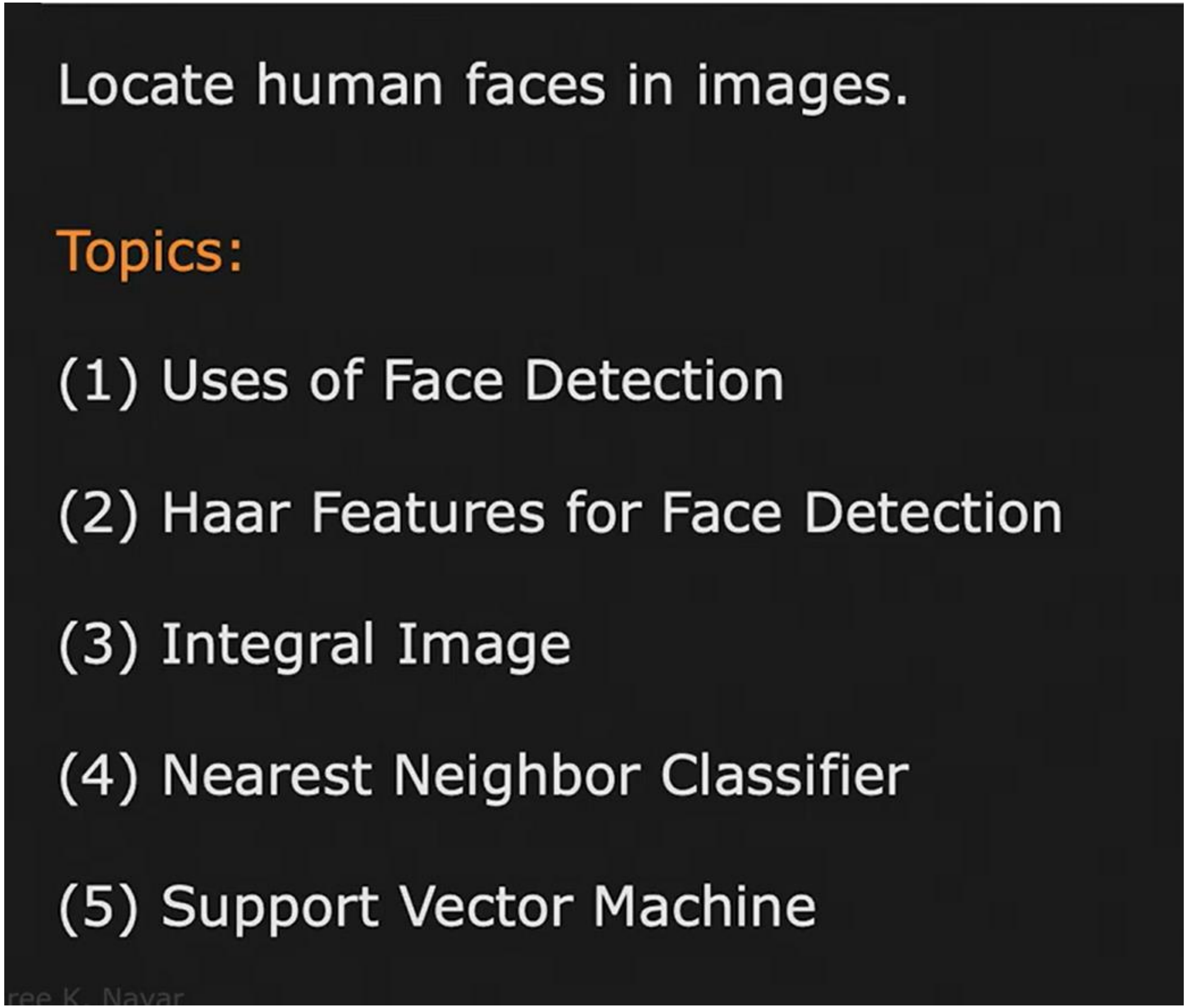
Gundimeda Venugopal, Professor of Practice, SCOPE

What is face detection



Locate human faces in images

Face Detection



Locate human faces in images.

Topics:

(1) Uses of Face Detection

(2) Haar Features for Face Detection

(3) Integral Image

(4) Nearest Neighbor Classifier

(5) Support Vector Machine

ree K. Nayar

What is Face Detection?

- Face Detection solution should be:
  - Insensitive to lighting changes
    - E.g., Should be able to detect faces in different class rooms
  - Able to handle faces of different sizes
    - Kids face vs Elder’s face
    - People could be at different distances from the camera
      - E.g., Should be able to detect faces in front as well as in back of a class room
  - Able to handle the rotations of the head with respect to the camera

Let us assume that we are looking at frontal faces.

Faces have a particular type of appearance (eyes, nose, lips , eye brows etc).

Our goal is to simply discriminate between faces and non-faces.

Face Detection - Preprocessing

- Firstly the image is imported by providing the location of the image.
- The picture is then transformed from RGB to Grayscale because it is easy to detect faces in the grayscale.
- After that, the image manipulation used, in which the resizing, cropping, blurring and sharpening of the images done if needed. The next step is image segmentation, which is used for contour detection or segments the multiple objects in a single image so that the classifier can quickly detect the objects and faces in the picture.

Converting RGB image to Grayscale

Face Detection Steps

- For Face Detection, we use the following steps:

1) Haar features are computed using Haar filters which are based on Haar wavelets or the Square functions.

Haar Filters are very effective and they are very efficient to compute.

2) The speed up of Haar filter computations comes with the use of **Integral Images**.

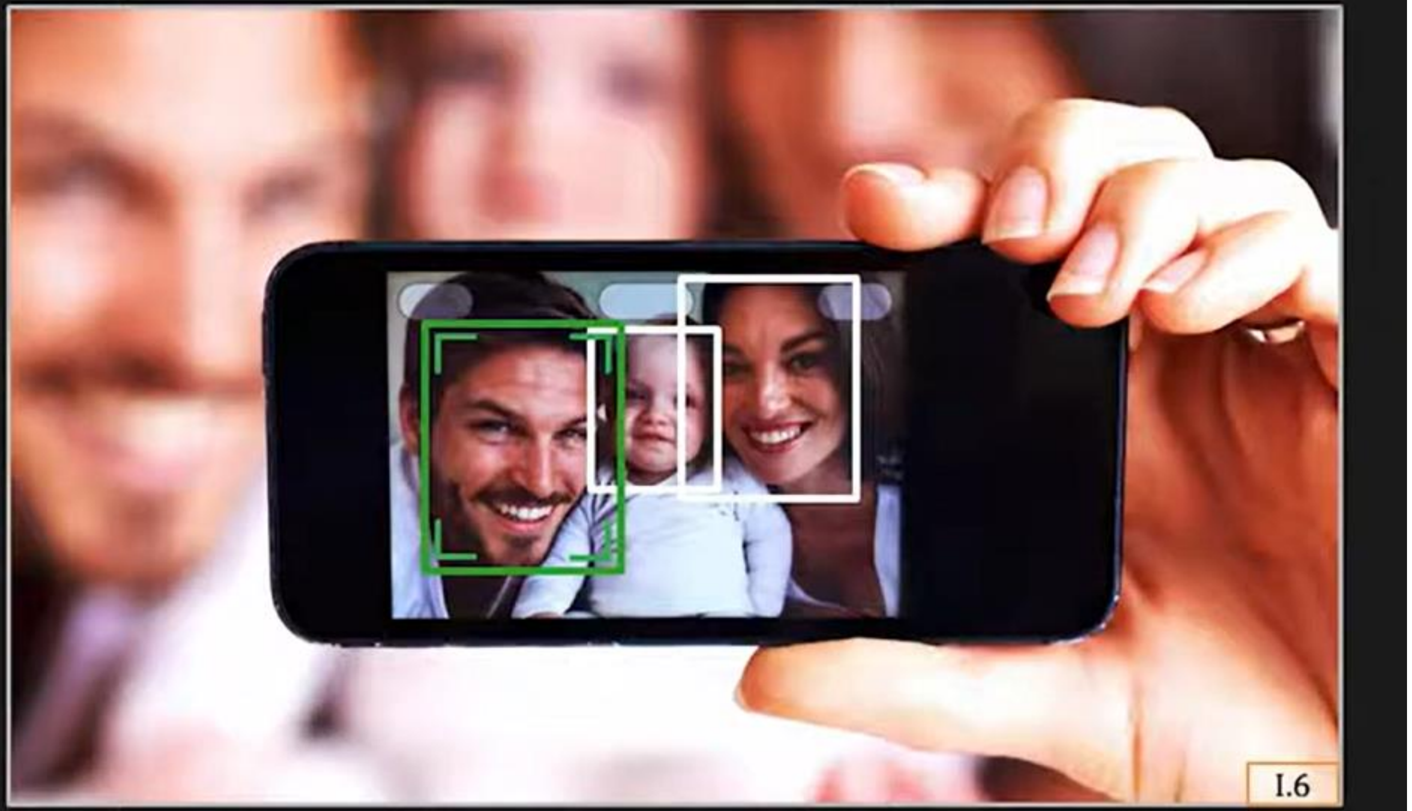
Integral images can calculate Haar filter output with the same speed irrespective of the size of Haar filters or size of images used.

3) Once the features are extracted, the features are sent to a classification algorithm to detect if the image has a face or not

    - Nearest neighbour
    - SVM

4) Bounding box(es) is/are generated to locate the coordinate(s) of human face(s).

Where the face detection is used?



Automatic Selection of Camera Settings
(Autofocus, Exposure, Color Balance, etc.)

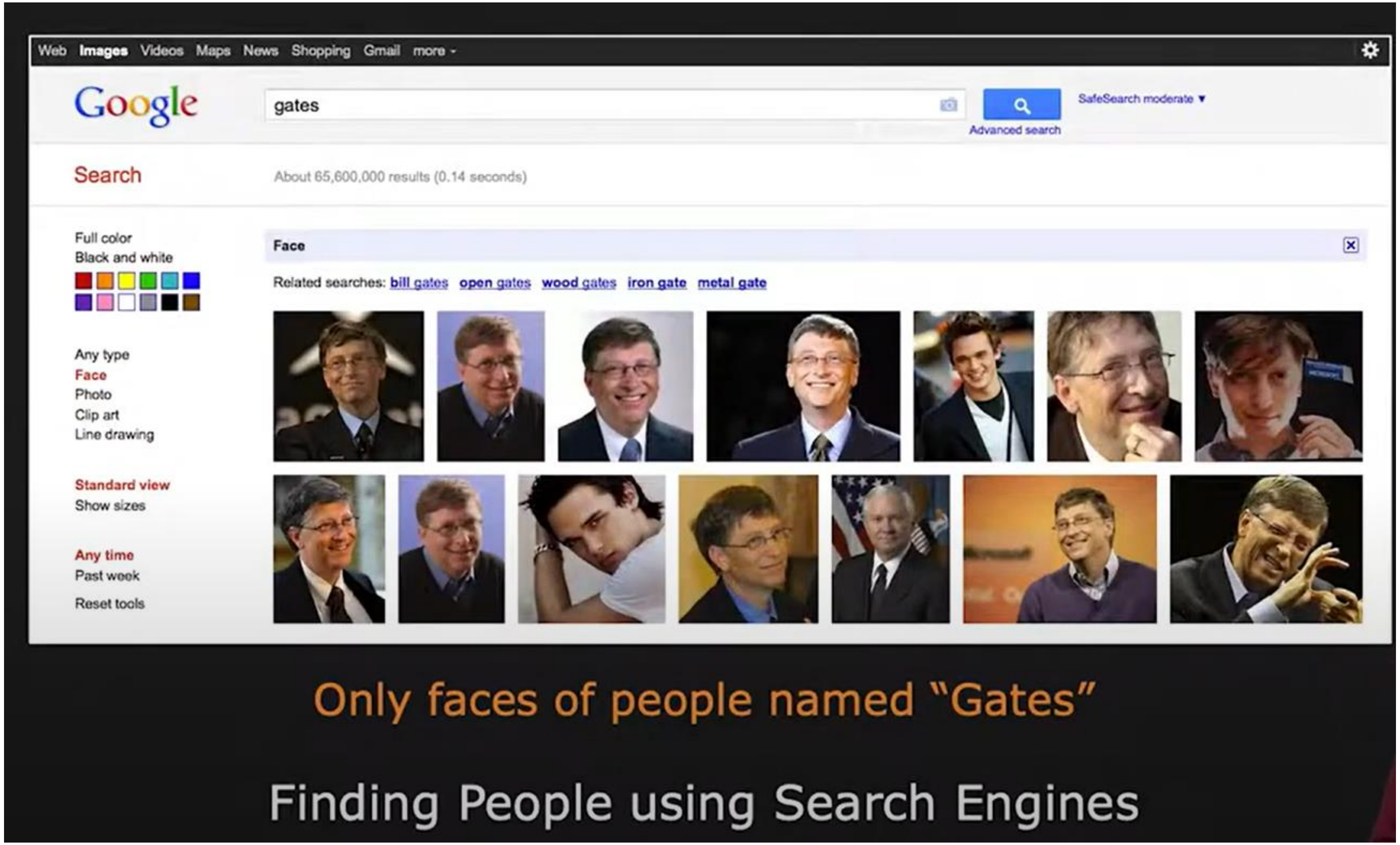
Every Smart phone has Face detection running in the background.

When we take pictures of a scene with people in it, we use a preview mode

The camera app preview mode does use real time face detection and faces are detected and shown to the users.

We need faces of high quality. So, the parameters of the camera are adjusted (e .g., Focus, exposure, color balance) to have a clearer focus on human faces.

Where Face detection is used?



gates

About 65,600,000 results (0.14 seconds)

Face ☒

Related searches: bill gates open gates wood gates iron gate metal gate

Only faces of people named "Gates"

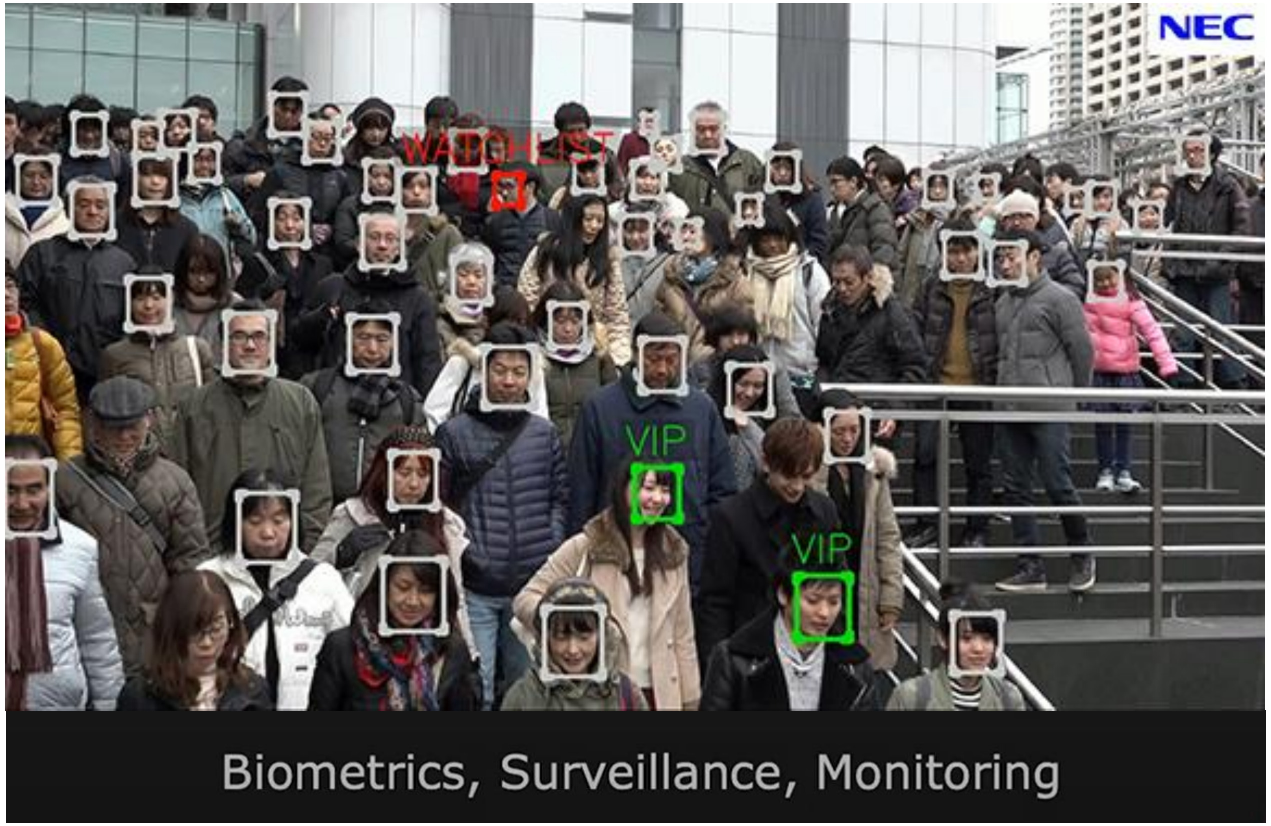
Finding People using Search Engines

Where Face detection is used?

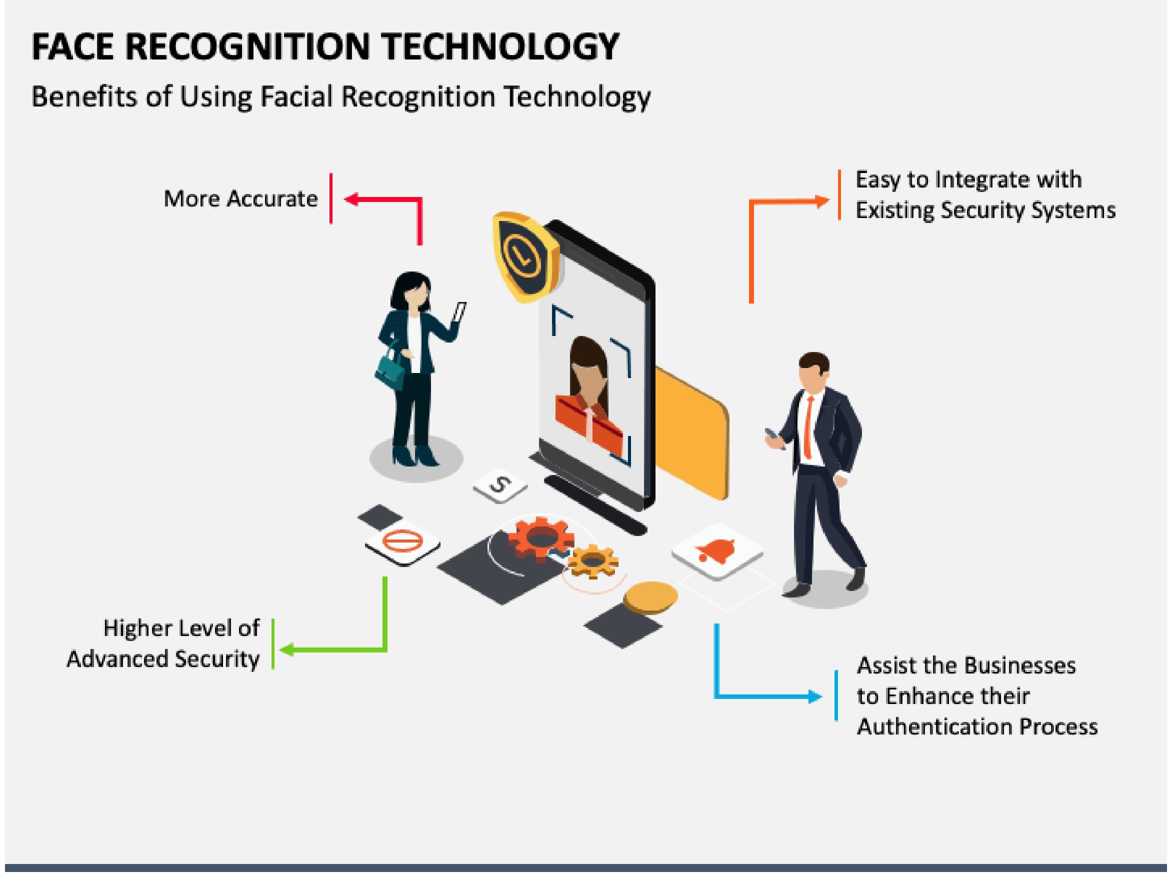
Intelligent Marketing

Detects customer’s gender and ~age
Based on demographic information, it displays various products that are of interest.

Where is Face Detection used?



Biometrics, Surveillance, Monitoring



FACE RECOGNITION TECHNOLOGY

Benefits of Using Facial Recognition Technology

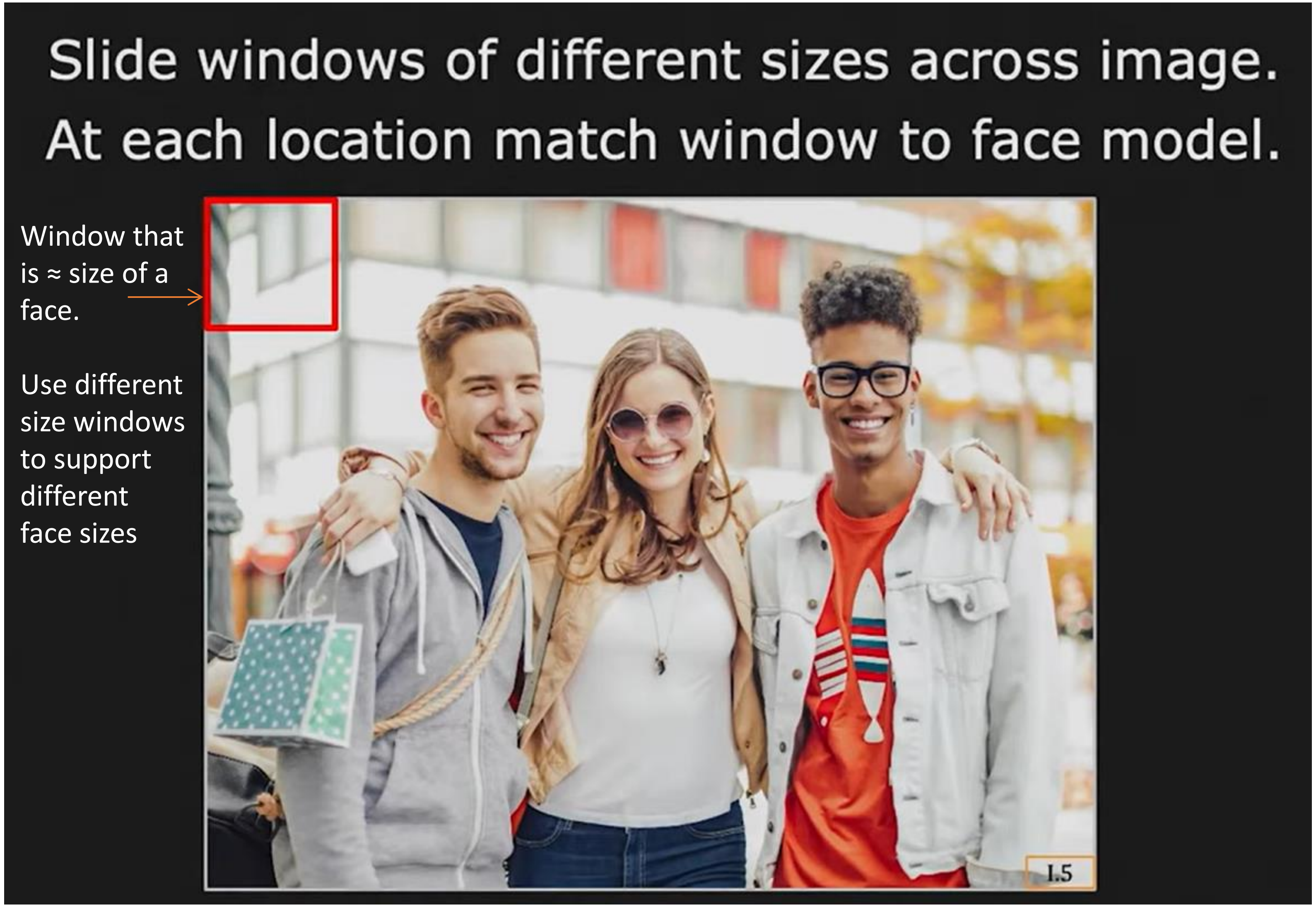
More Accurate

Easy to Integrate with Existing Security Systems

Higher Level of Advanced Security

Assist the Businesses to Enhance their Authentication Process

Face Detection in Computers



Slide windows of different sizes across image.
At each location match window to face model.

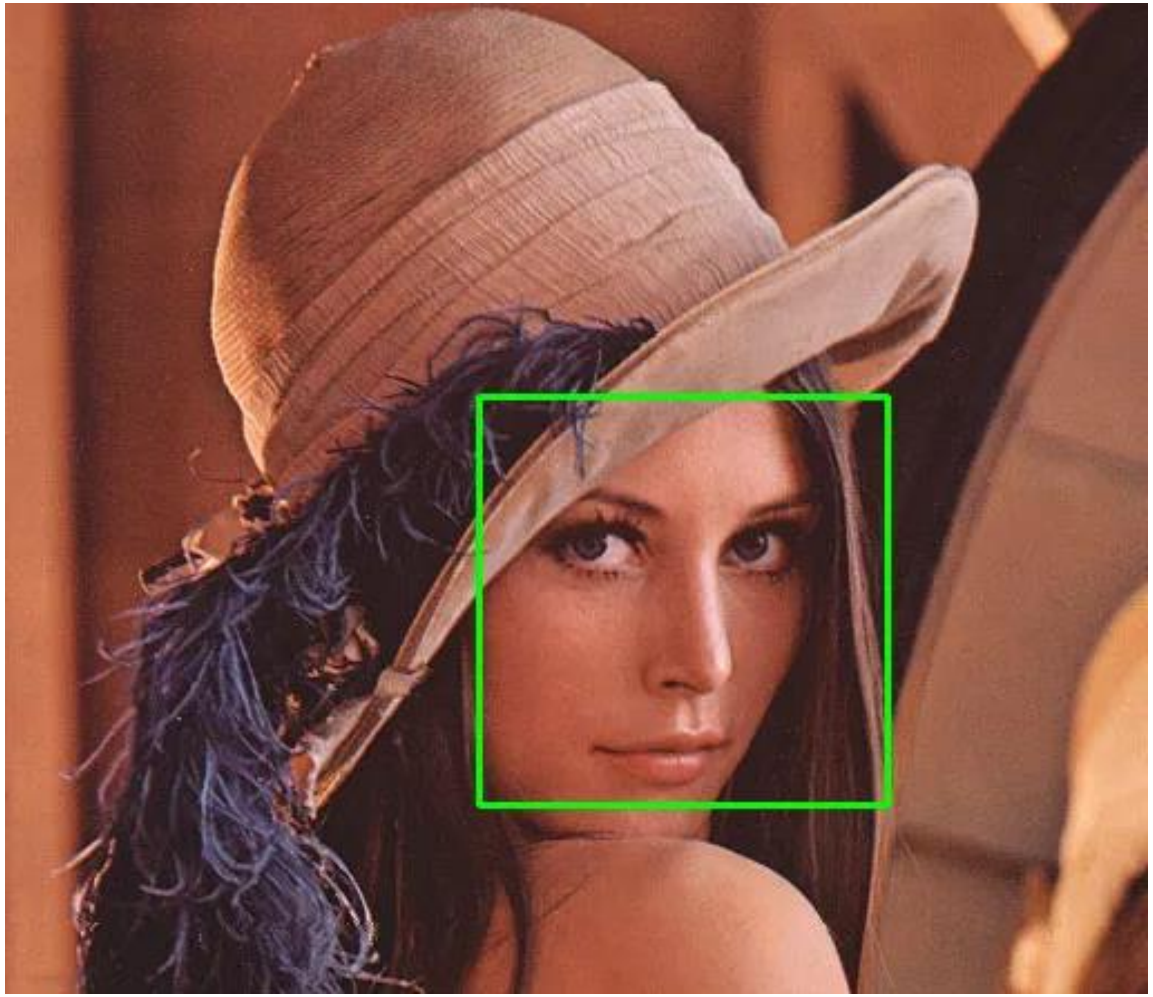
Window that is ≈ size of a face.

Use different size windows to support different face sizes

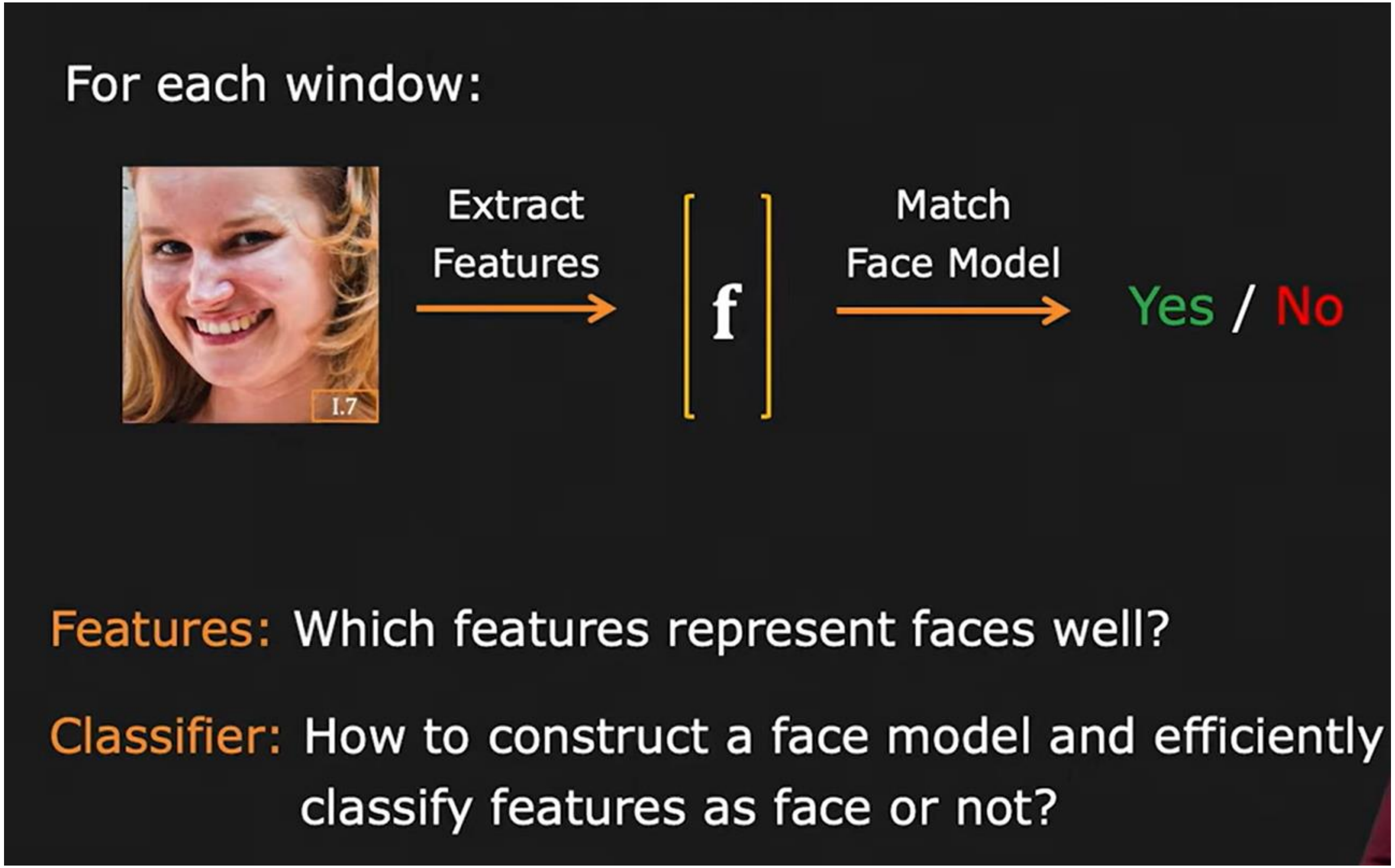
Run this window across this image in a raster scan fashion and at each pixel extract features and classify the features that have been extracted as face or non-face

Face Detection: Bounding Box

The next step is to give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face or we can say that to show the region of interest in the image. After this, it can make a rectangle box in the area of interest where it detects the face.



Face Detection Framework



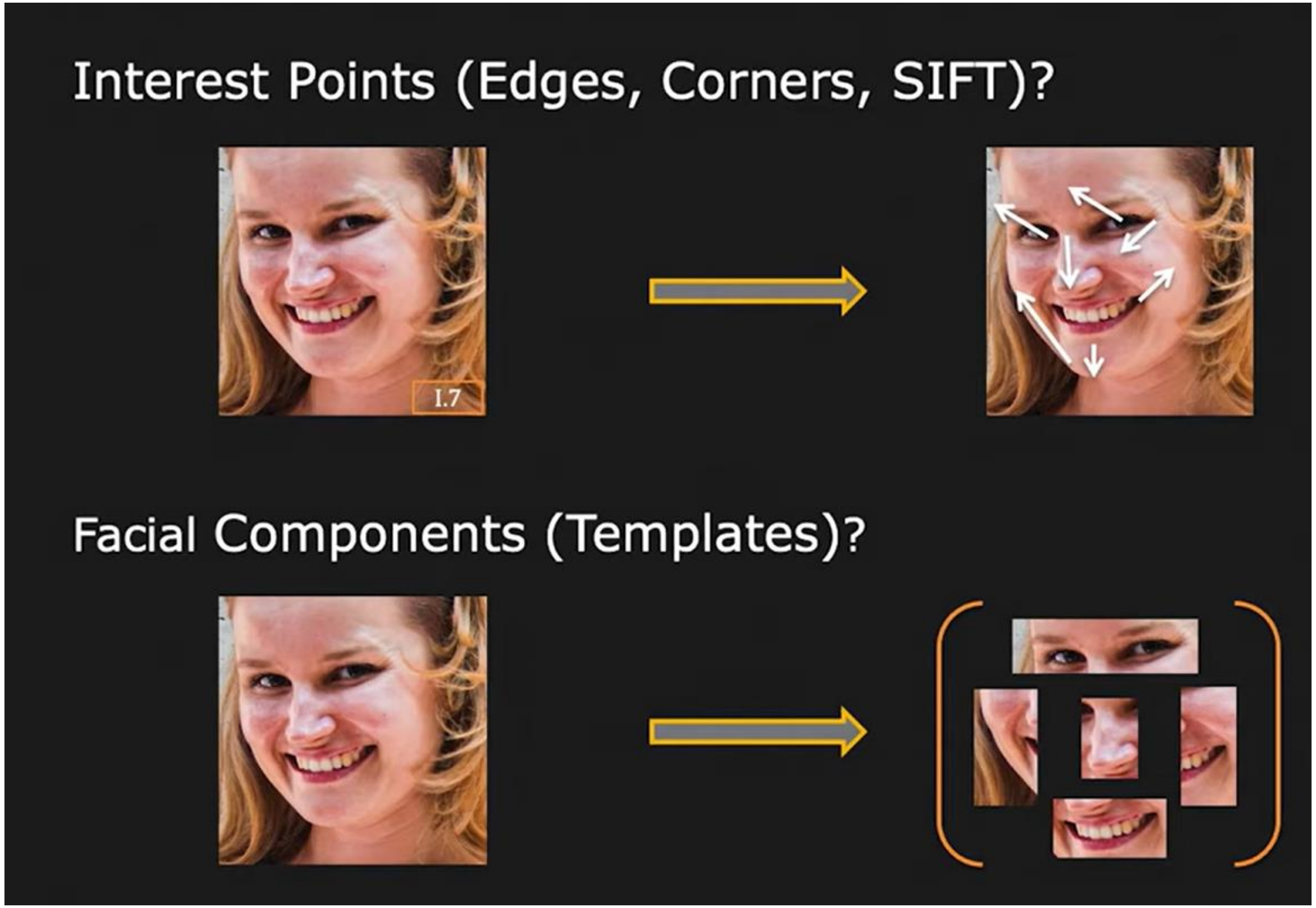
For each window:

Extract Features → [f] → Match Face Model → Yes / No

Features: Which features represent faces well?

Classifier: How to construct a face model and efficiently classify features as face or not?

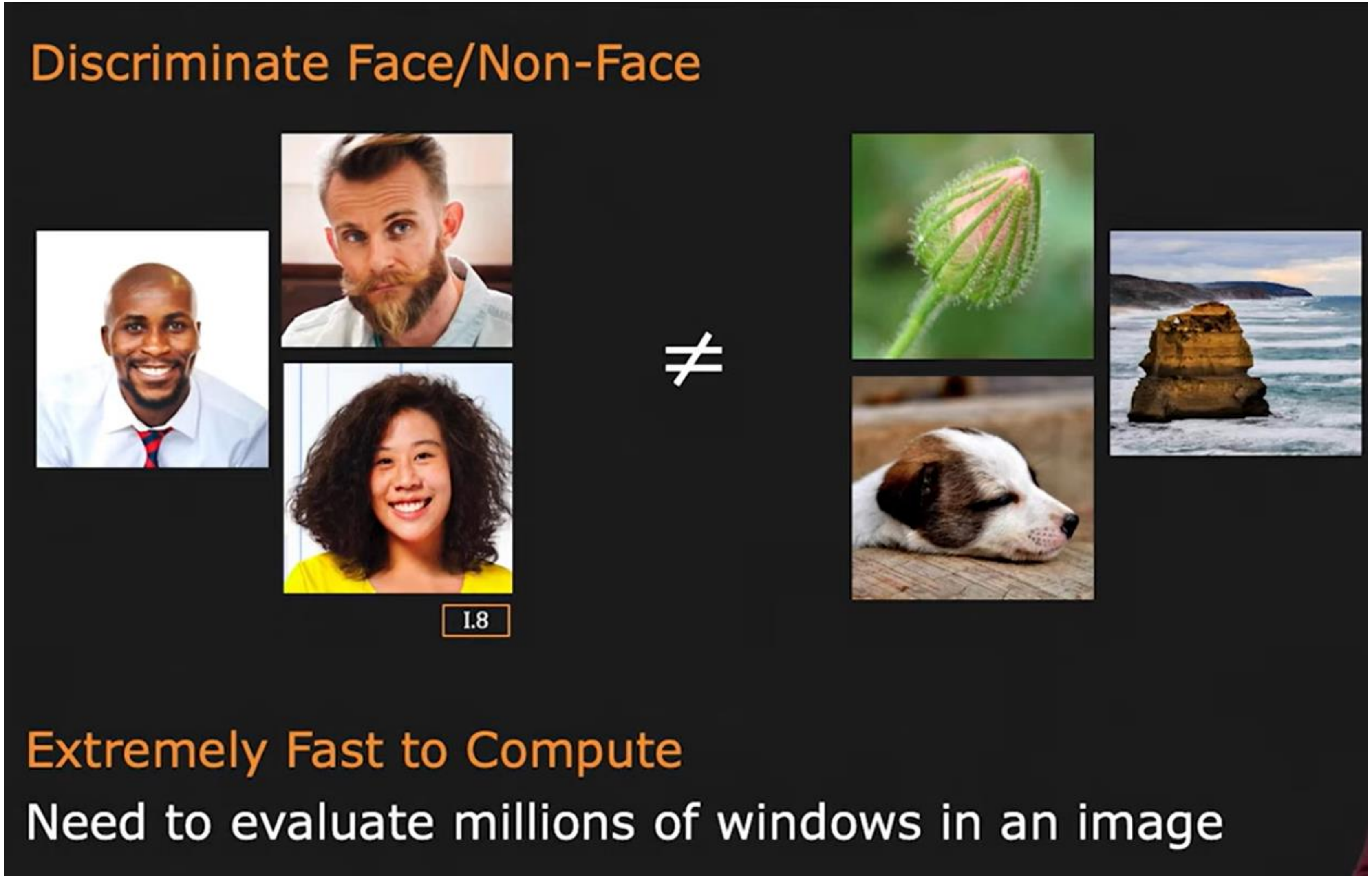
What are good features for face detection?


Interest Points (Edges, Corners, SIFT)?
1.7
Facial Components (Templates)?

Feature Options

- ❖ Interest Points (Edges / corners): lots of edges with many objects
- ❖ Interest Points (SIFT): good for matching appearances
- ❖ Facial Components (Templates)
  - ➢ Components: Eyes, nose, mouth, right side, left side
  - ➢ Lot of variability within components (e.g., eyes, eyebrows)

Characteristics of Good Features



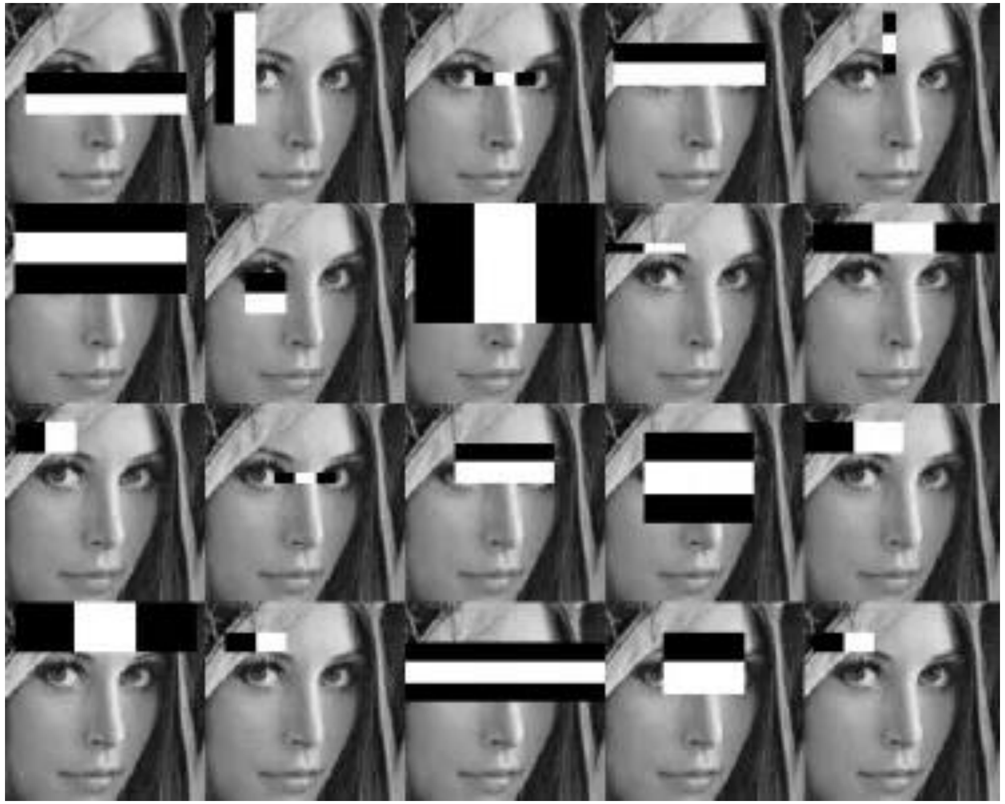
Discriminate Face/Non-Face

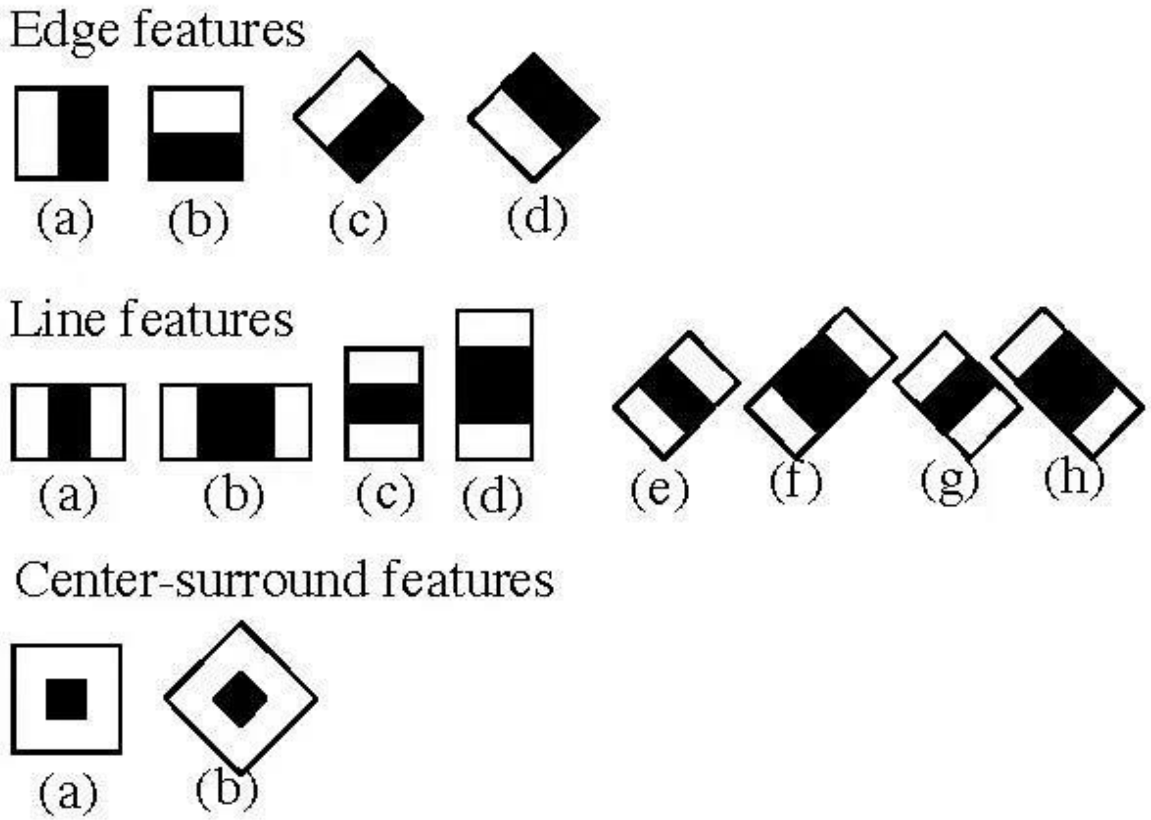
Extremely Fast to Compute

Need to evaluate millions of windows in an image

- When we take pictures, we focus on people (we focus on faces)
- Camera parameters have to be adjusted to have a clearer focus on human faces.
- Face detection is running in the background while we are setting the camera parameters and clicking the button to click the pictures.

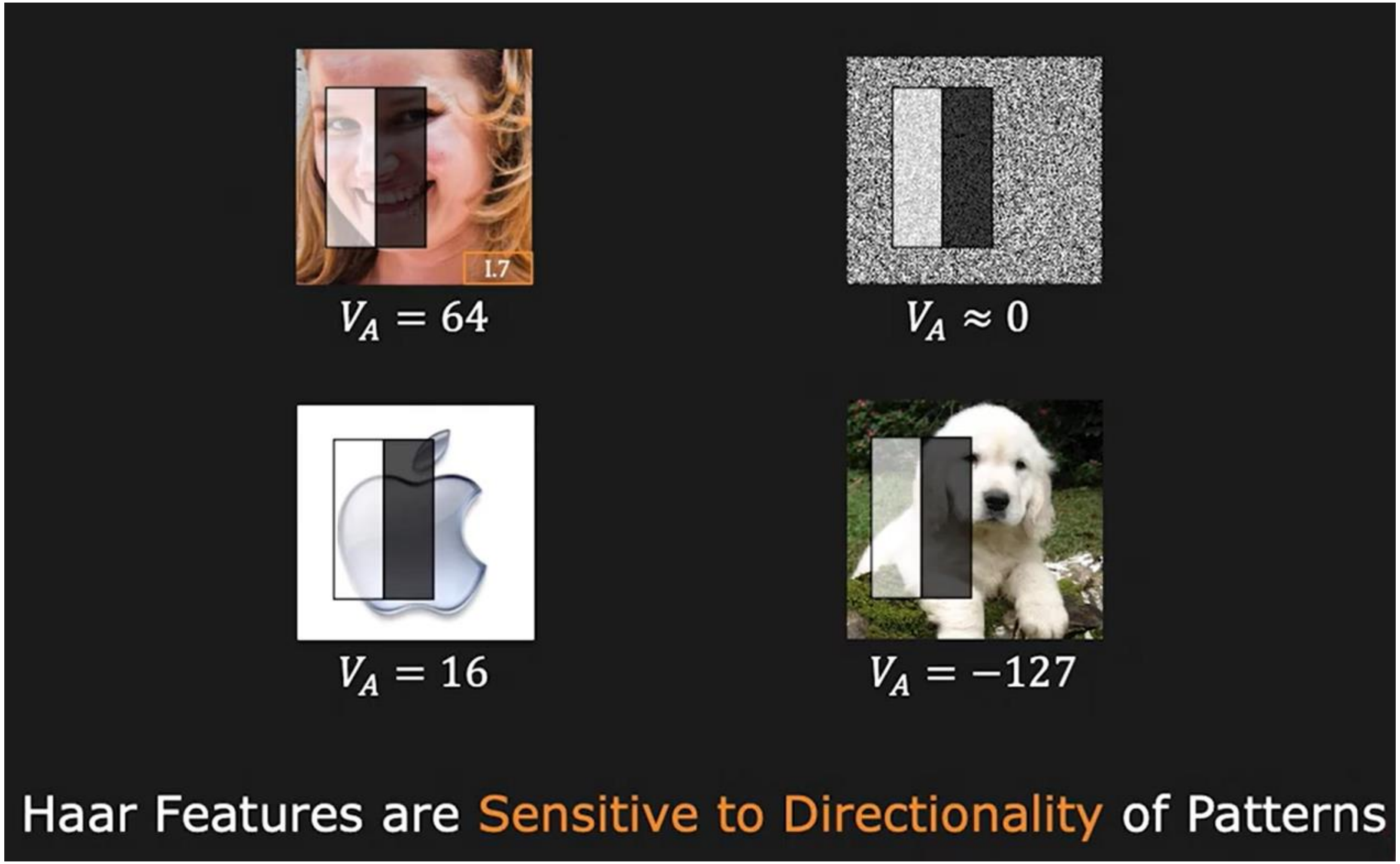
Haar Like features





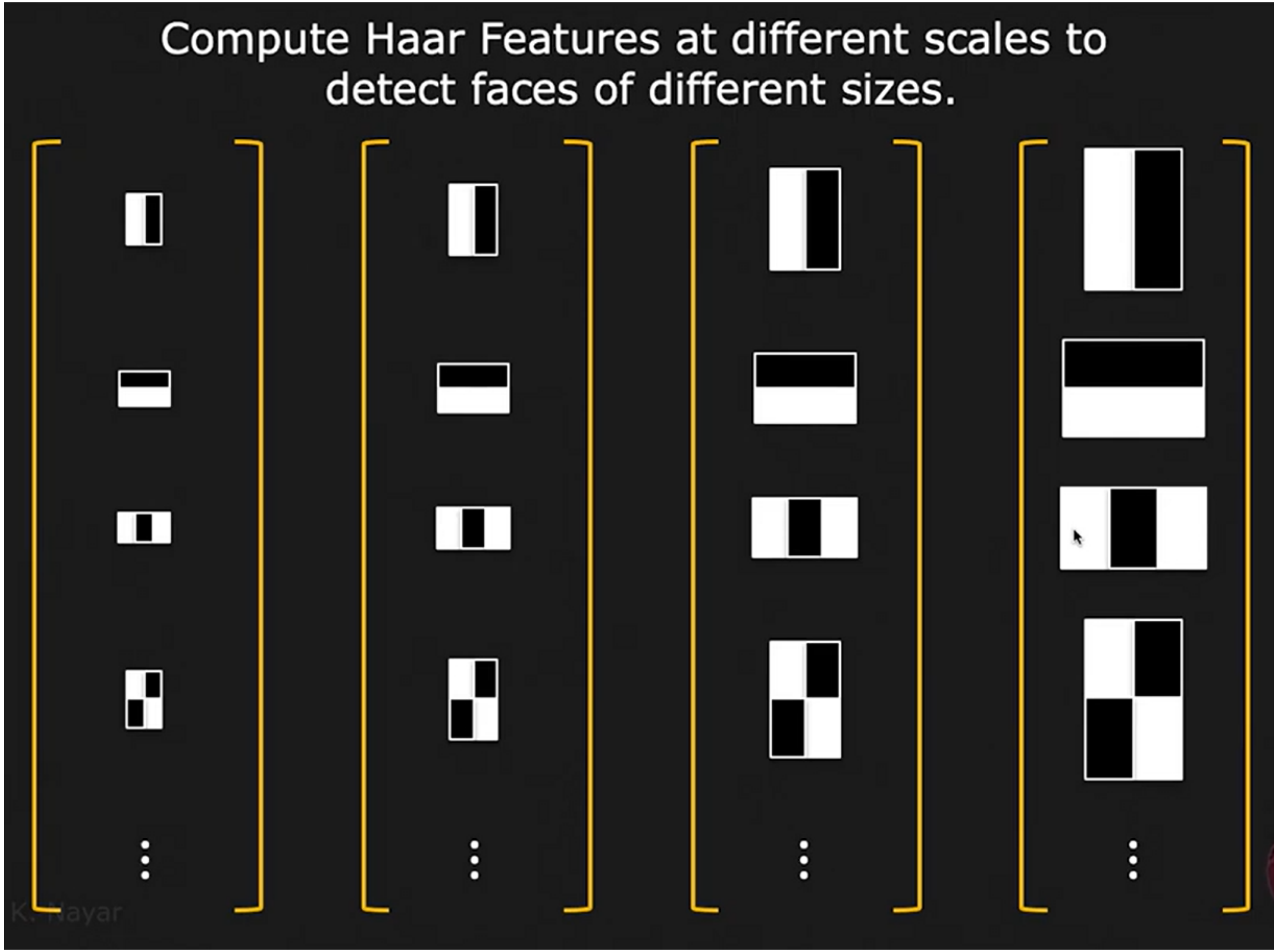
- Originally, Viola Jones used Haar-Like features for face detection.
- All human faces shares some universal properties of the human face like the eyes region is darker than its neighbour pixels and nose region is brighter than eye region.

Discriminative Ability of Haar Feature



Haar Features are Sensitive to Directionality of Patterns

Detecting Faces of Different Size



Compute Haar Features at different scales to detect faces of different sizes.

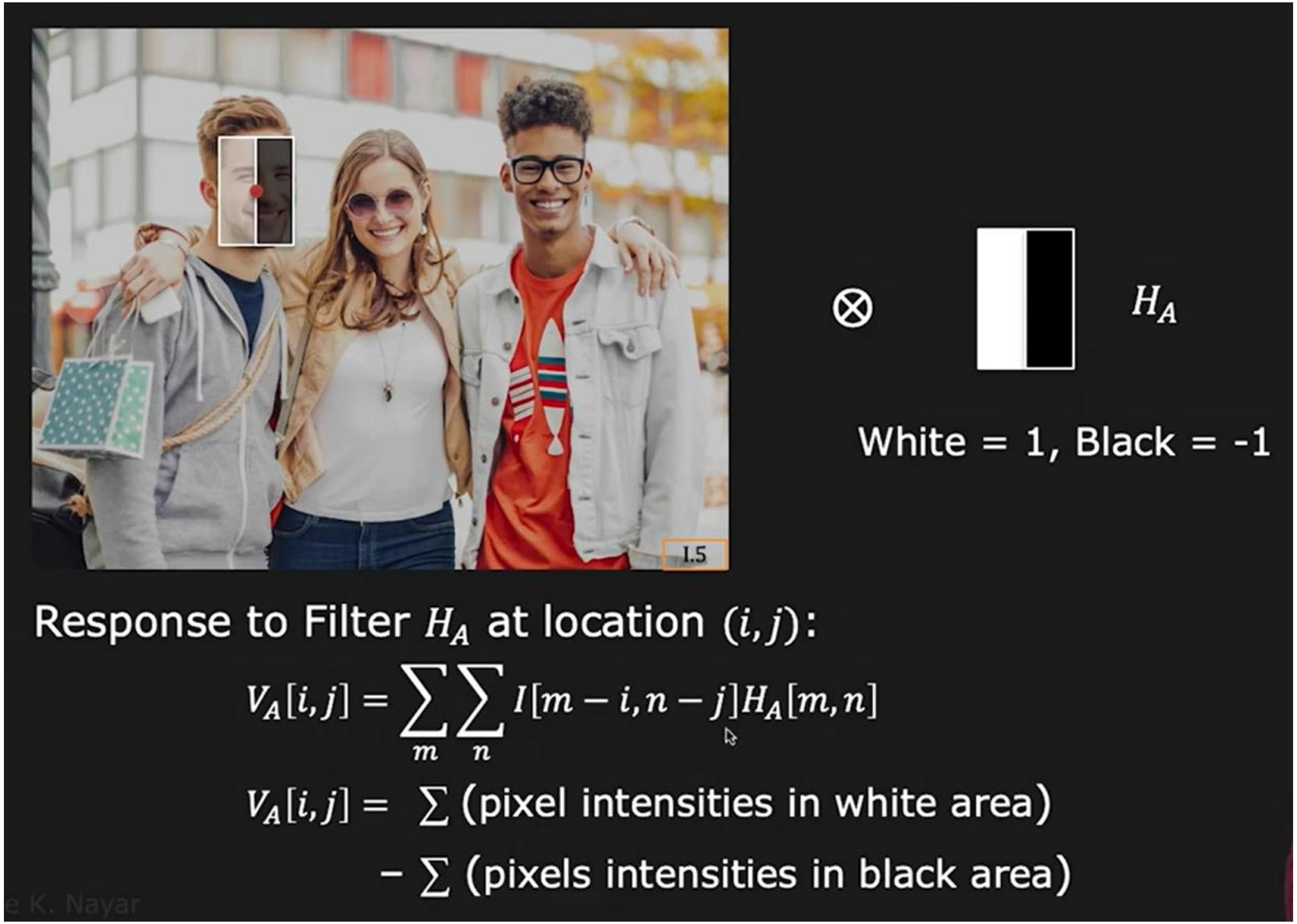
Large scale edge detector
Looks for Change in x-direction

Looks for Change in y-direction

Looks like a Laplacian

Two valued Higher order filters
(higher derivative filters)

Computing a Haar Feature



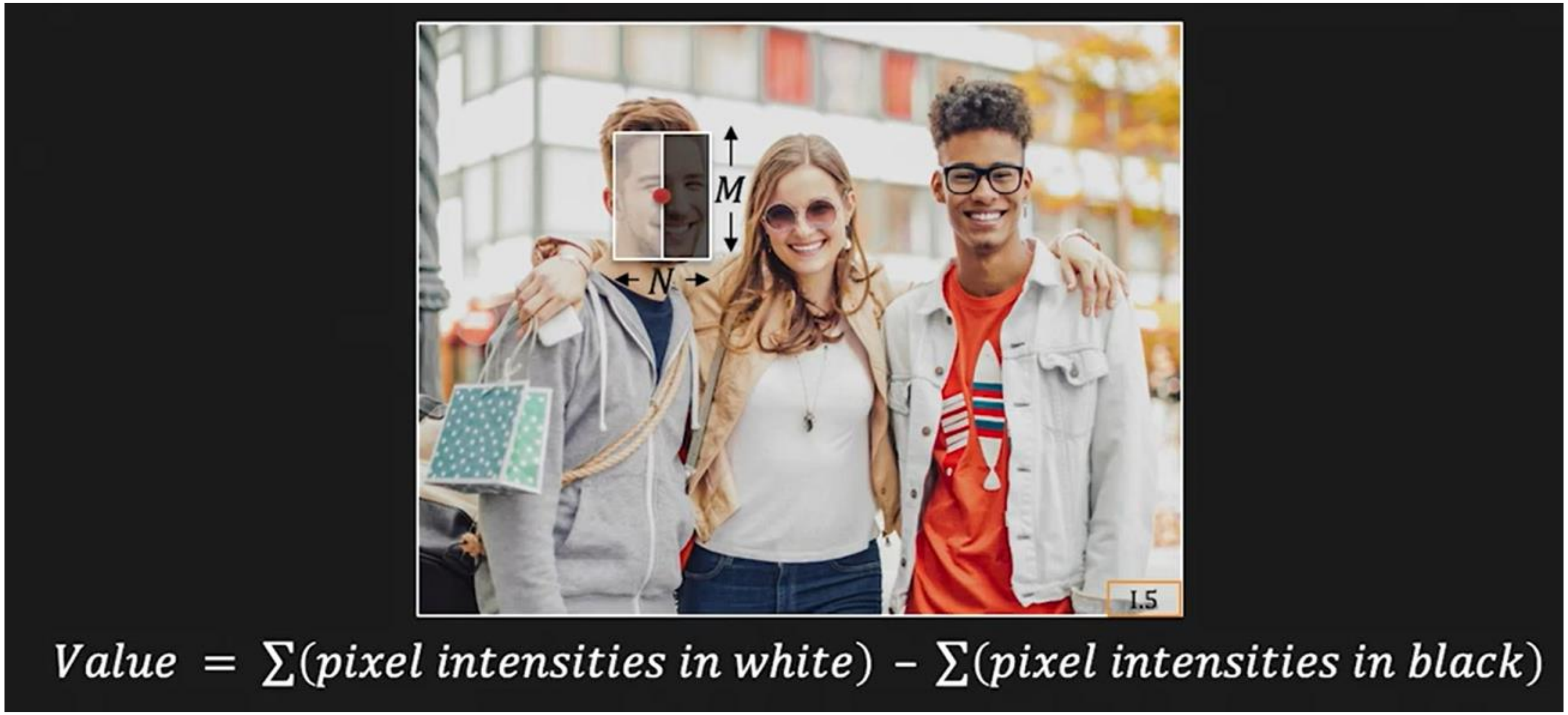
White = 1, Black = -1

Response to Filter H_A at location (i,j):

$$V_A[i,j] = \sum_m \sum_n I[m-i, n-j] H_A[m,n]$$

$$V_A[i,j] = \sum(\text{pixel intensities in white area}) - \sum(\text{pixels intensities in black area})$$

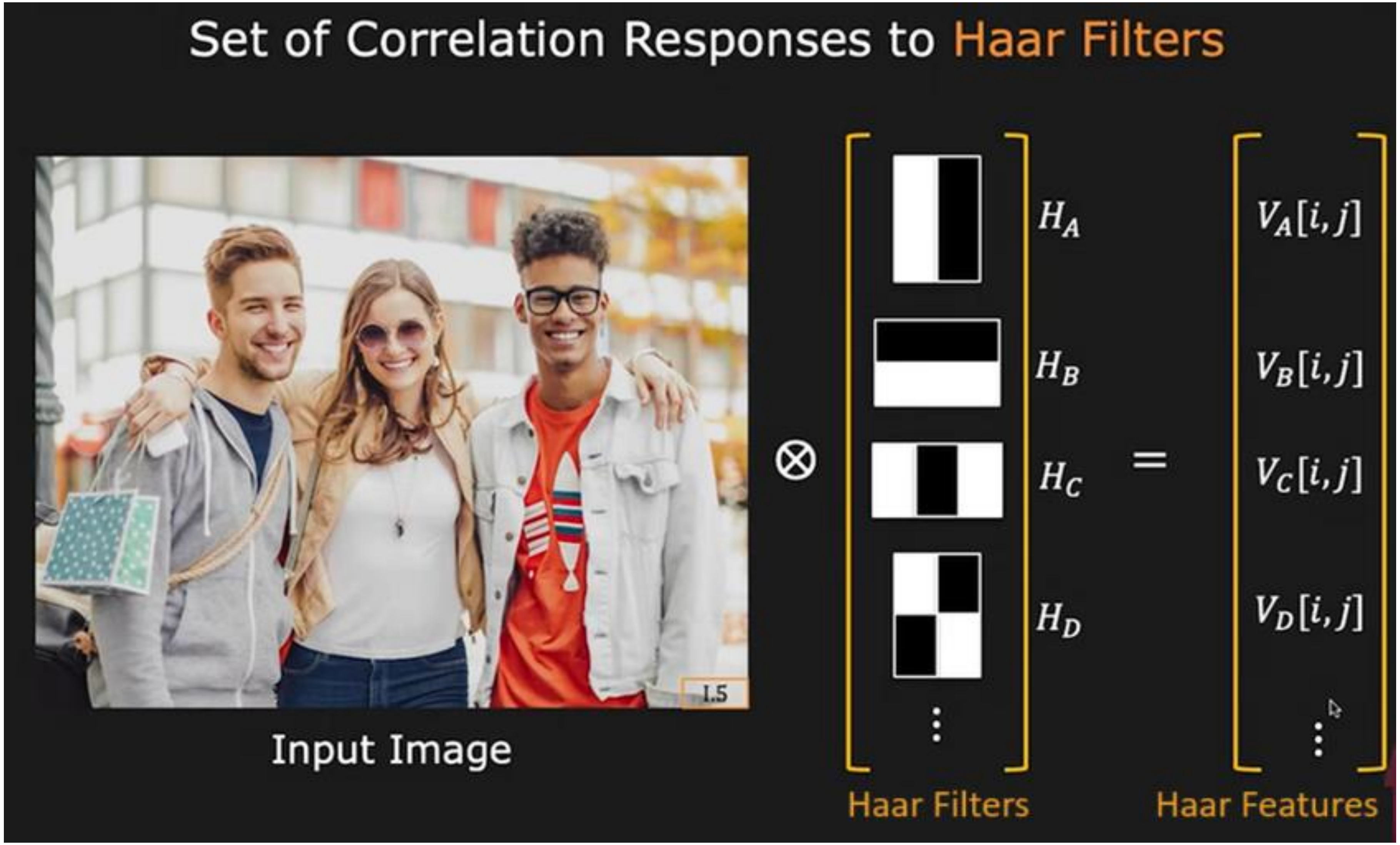
Haar Feature: Computation Cost



$$Value = \sum(pixel\ intensities\ in\ white) - \sum(pixel\ intensities\ in\ black)$$

Computation cost = (N X M – 1) additions per pixel per filter per scale

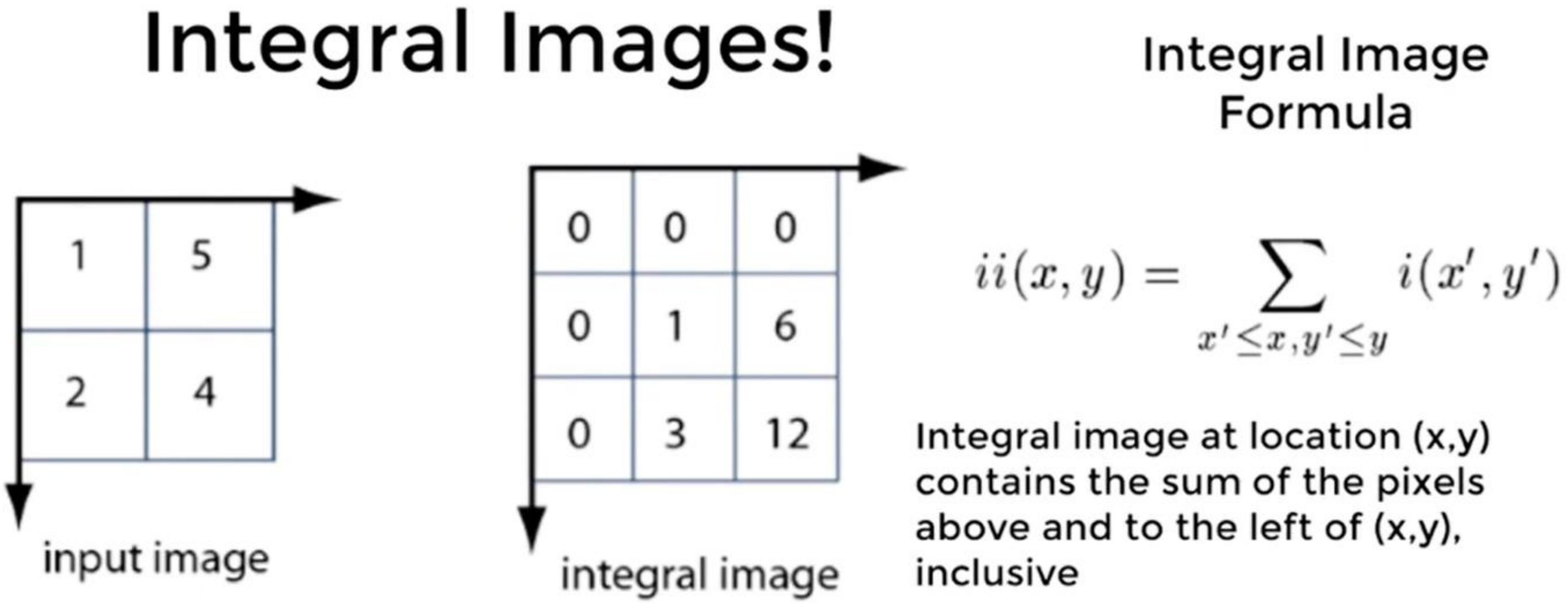
Haar Feature Vector (per pixel per scale)



Set of Correlation Responses to Haar Filters

Integral Images

Integral Image - Methodology



Integral Images!

1	5
2	4

input image

0	0	0
0	1	6
0	3	12

integral image

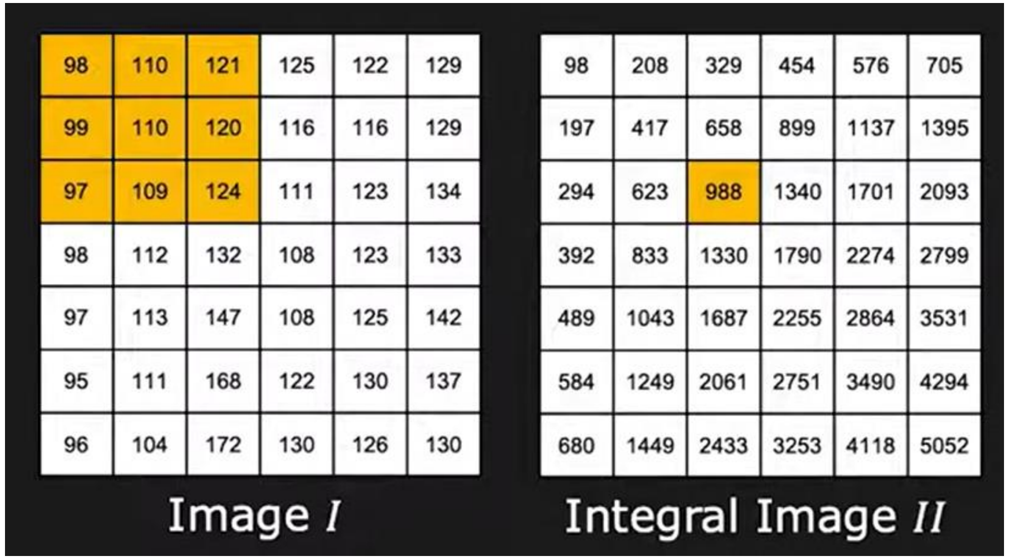
Integral Image Formula

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Integral image at location (x,y) contains the sum of the pixels above and to the left of (x,y), inclusive

Integral Image

❖ A table that holds the sum of all pixel values to the left and top of a given pixel, inclusive.

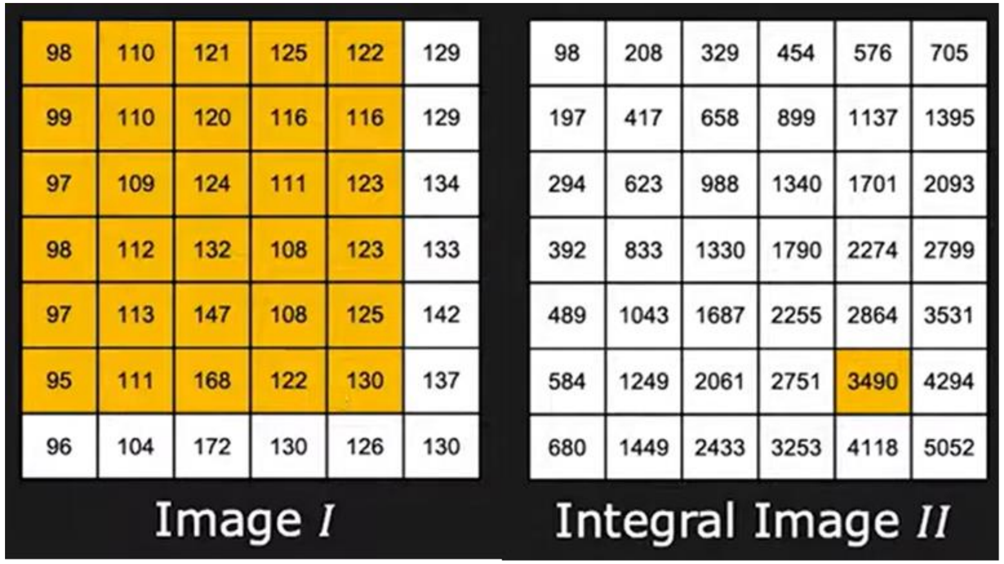


98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II



98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

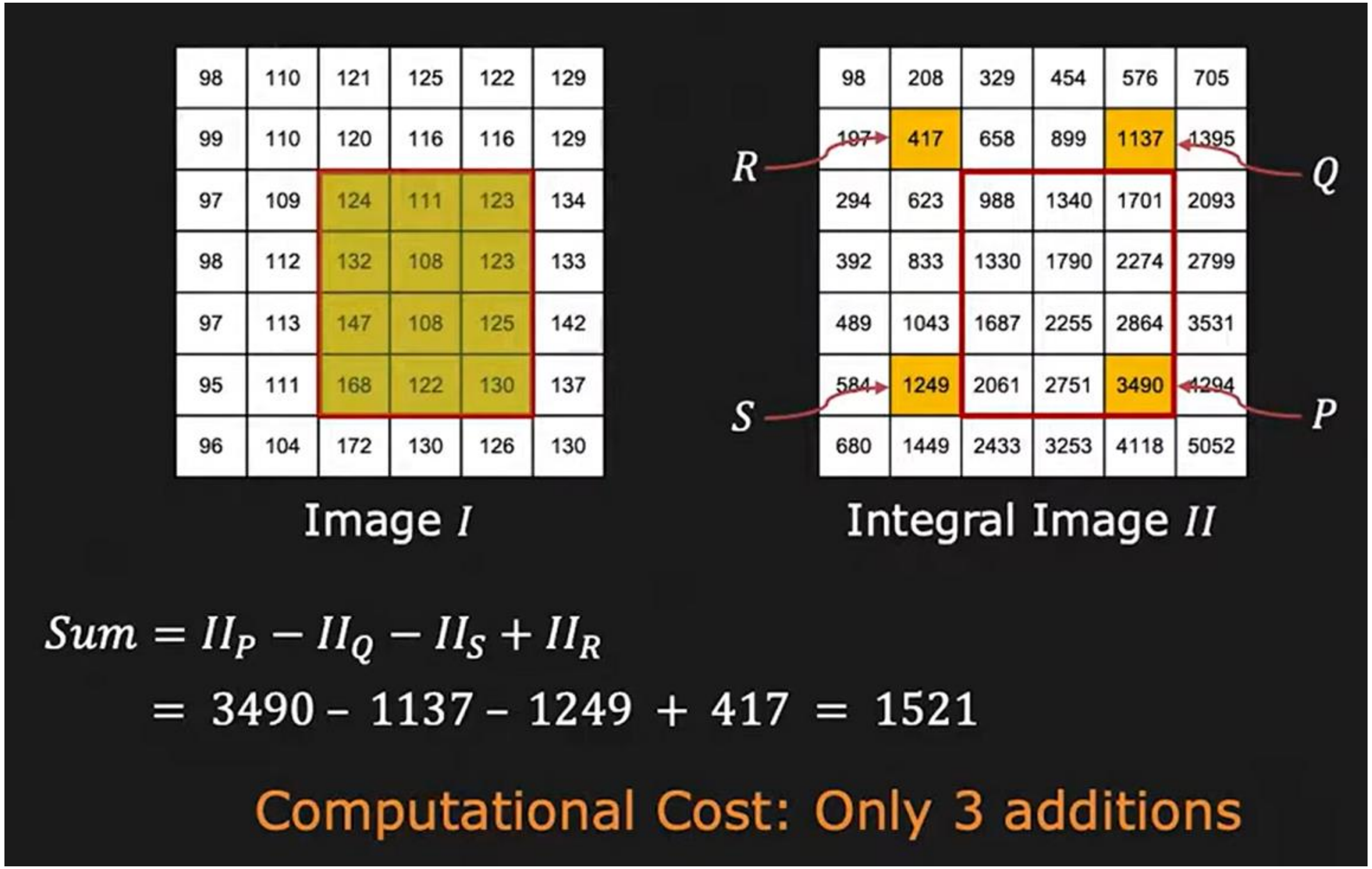
Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

Summation within a Rectangle

Fast summations of arbitrary rectangles using integral images



98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

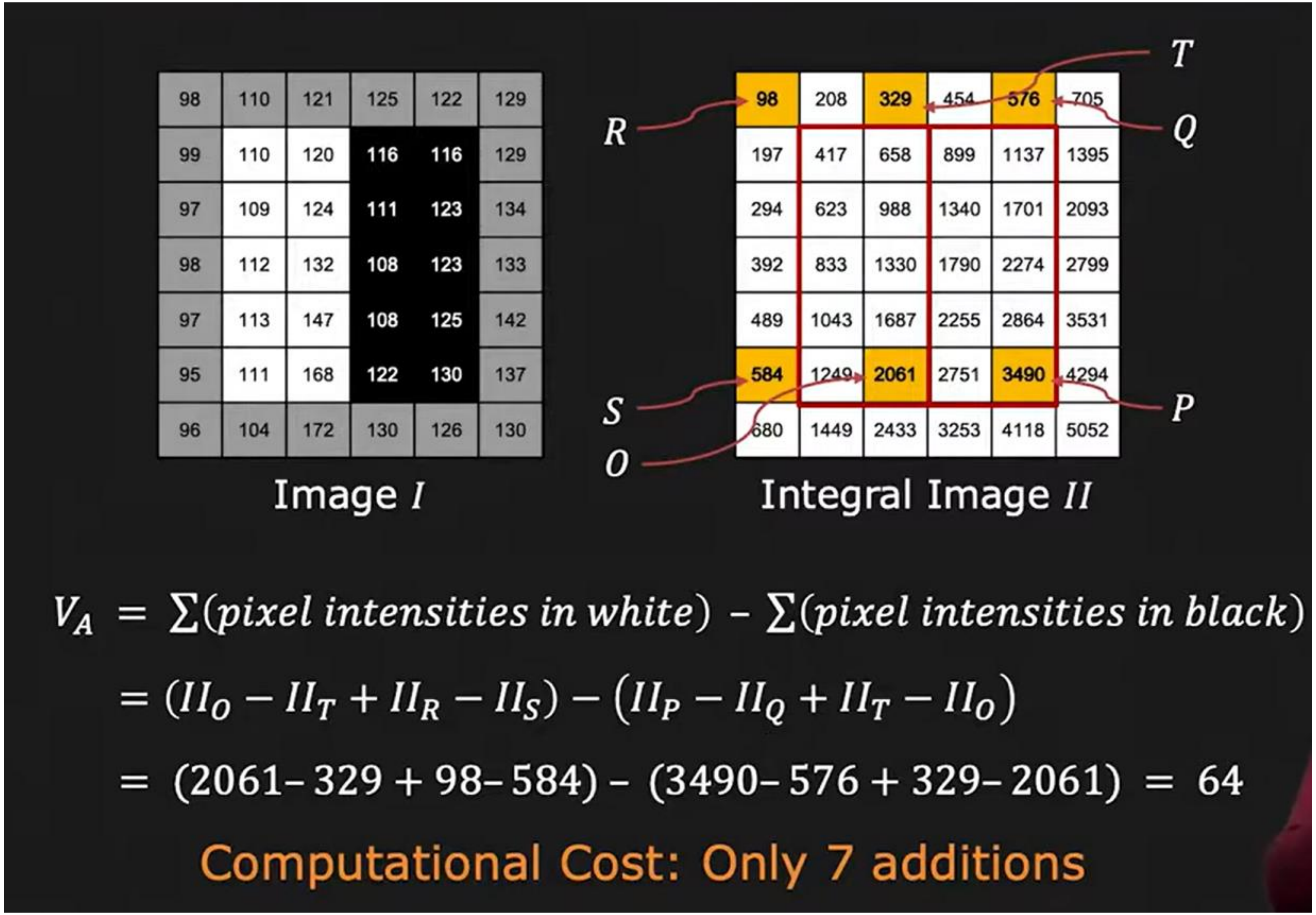
Integral Image II

$$Sum = II_P - II_Q - II_S + II_R$$

$$= 3490 - 1137 - 1249 + 417 = 1521$$

Computational Cost: Only 3 additions

Haar Response using Integral Image



98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

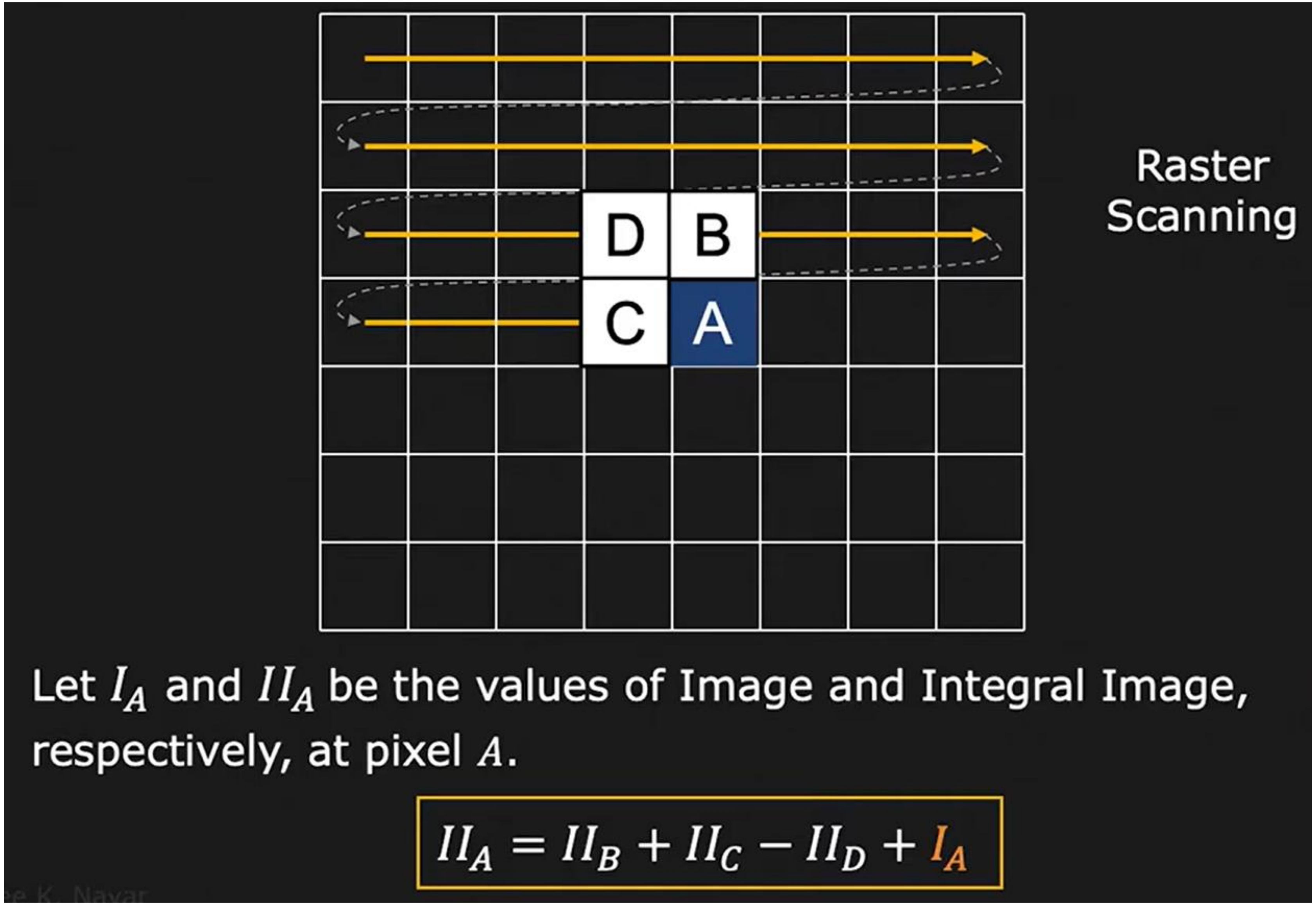
$$V_A = \sum(pixel\ intensities\ in\ white) - \sum(pixel\ intensities\ in\ black)$$

$$= (II_O - II_T + II_R - II_S) - (II_P - II_Q + II_T - II_O)$$

$$= (2061-329+98-584) - (3490-576+329-2061) = 64$$

Computational Cost: Only 7 additions

Computing the Integral Image



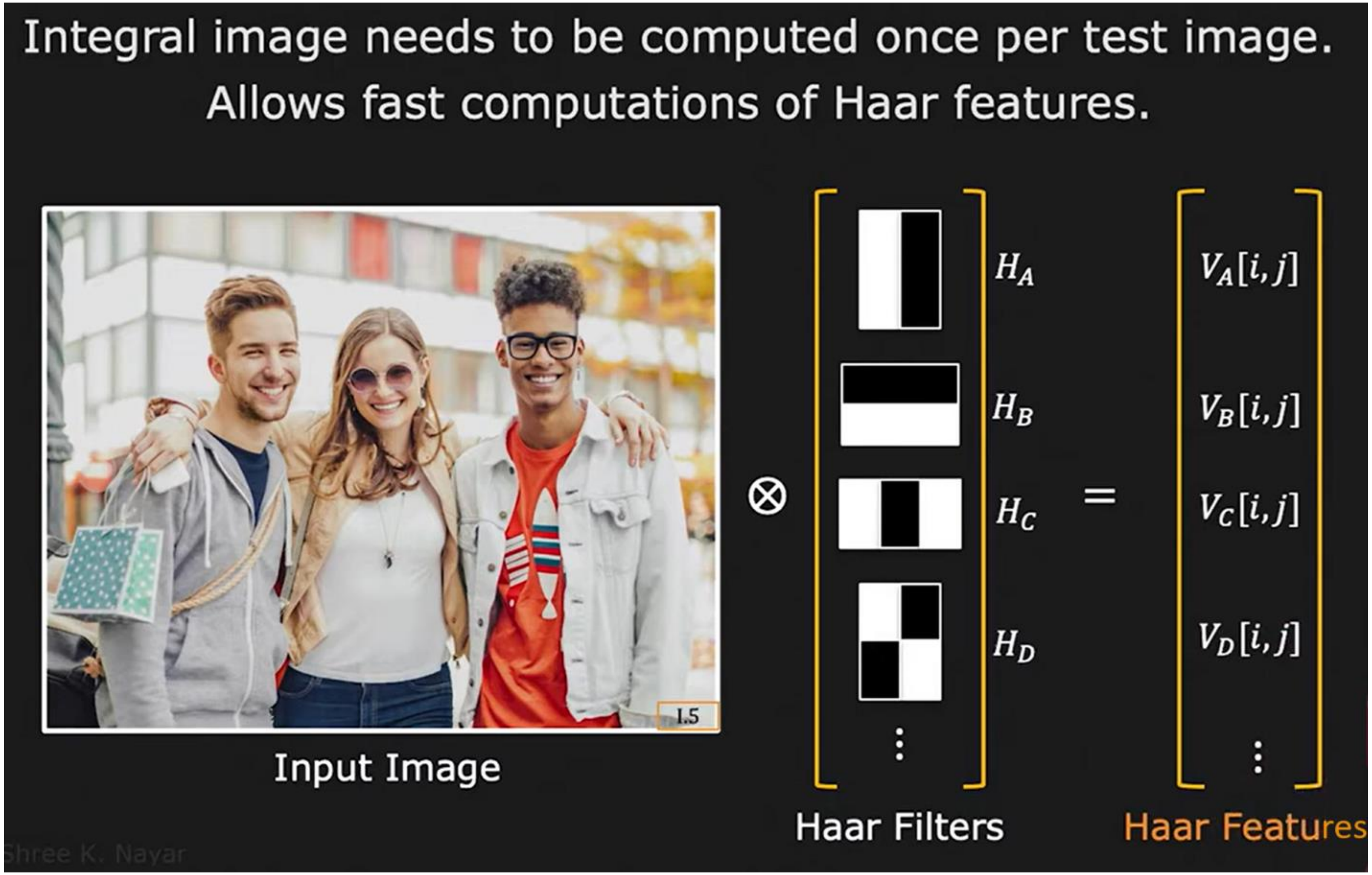
Raster Scanning

Let I_A and II_A be the values of Image and Integral Image, respectively, at pixel A.

$$\boxed{II_A = II_B + II_C - II_D + I_A}$$

Haar Features using Integral Images

Haar feature Vector at a point in Image. For feature comparison, you may have to use different vectors for different scale.



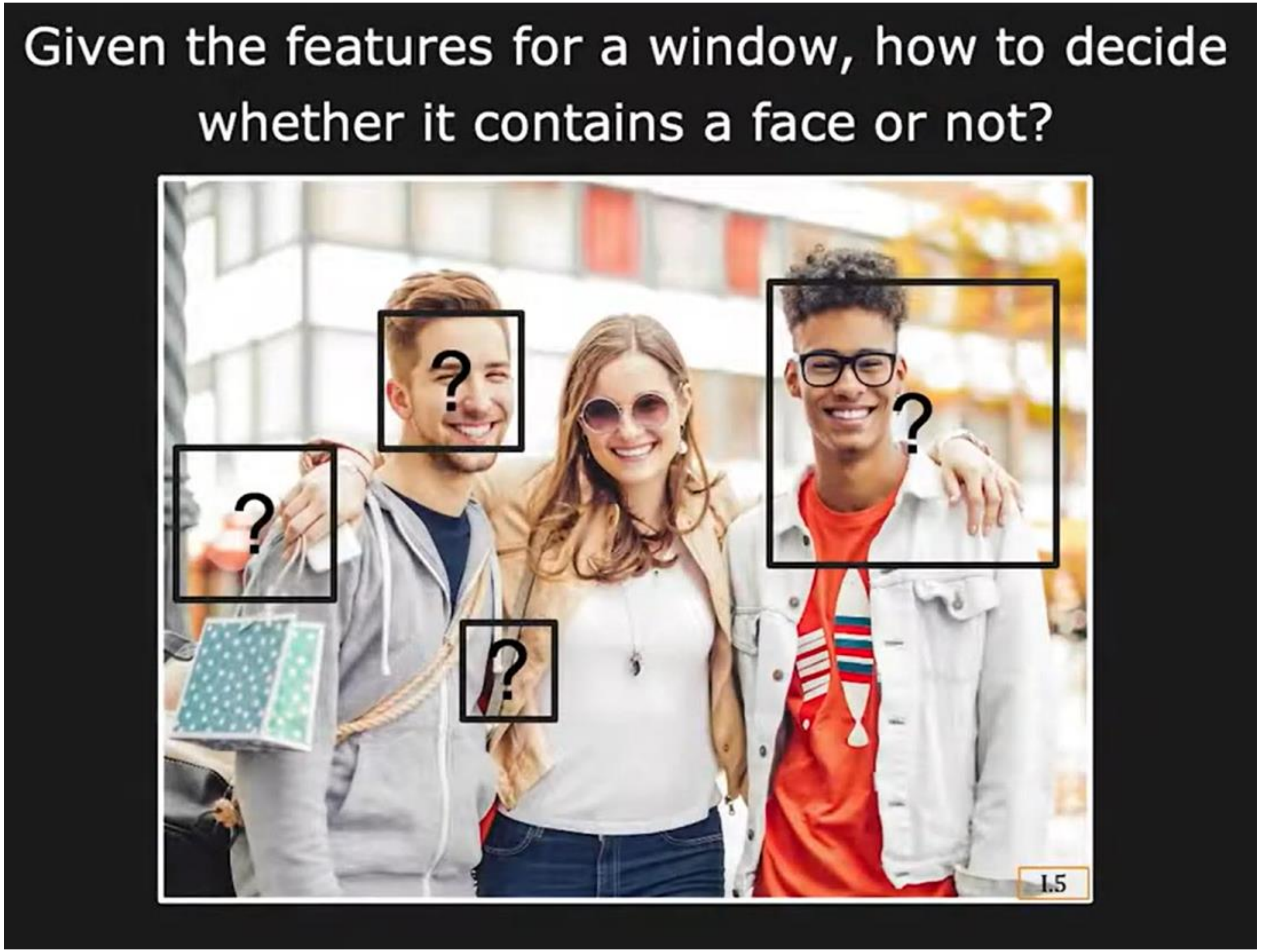
Integral image needs to be computed once per test image. Allows fast computations of Haar features.

Nearest Neighbor Classifier

Now we have a feature vector at each pixel in the image.

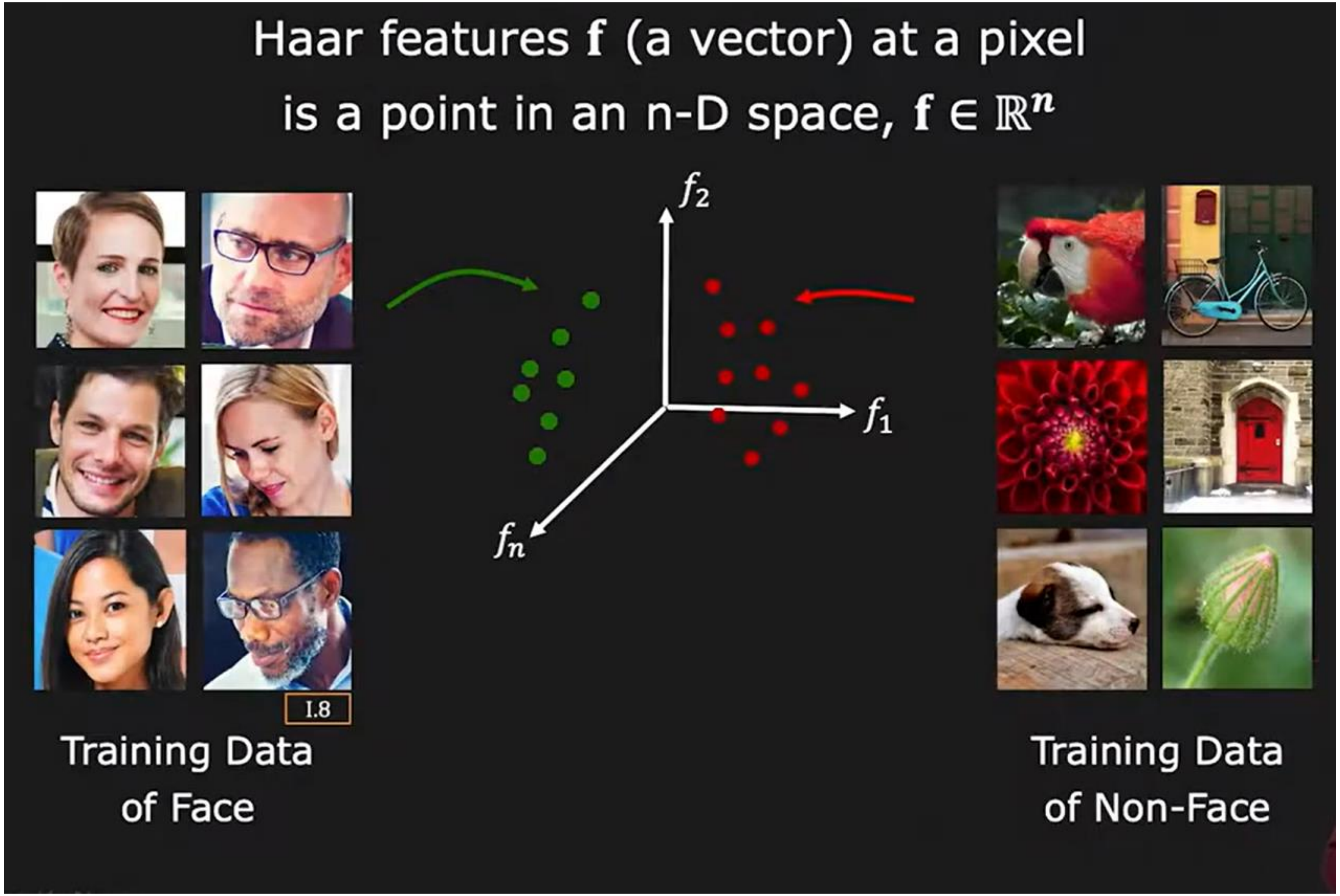
Use feature vector to classify: face vs non-face

Classifier for face Detection



Given the features for a window, how to decide whether it contains a face or not?

Feature Space

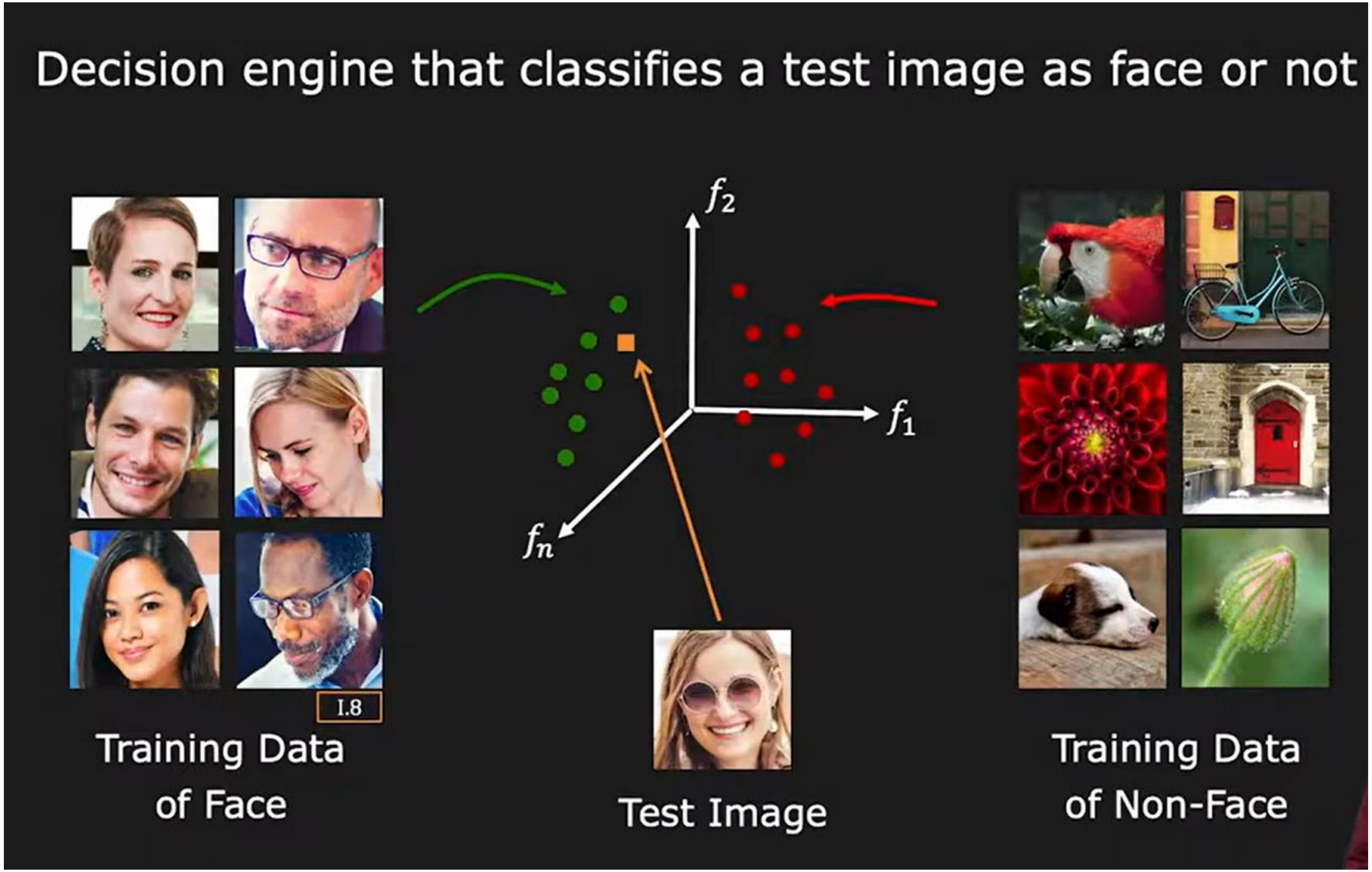


Haar features **f** (a vector) at a pixel is a point in an n-D space, $\mathbf{f} \in \mathbb{R}^n$

Training Data of Face

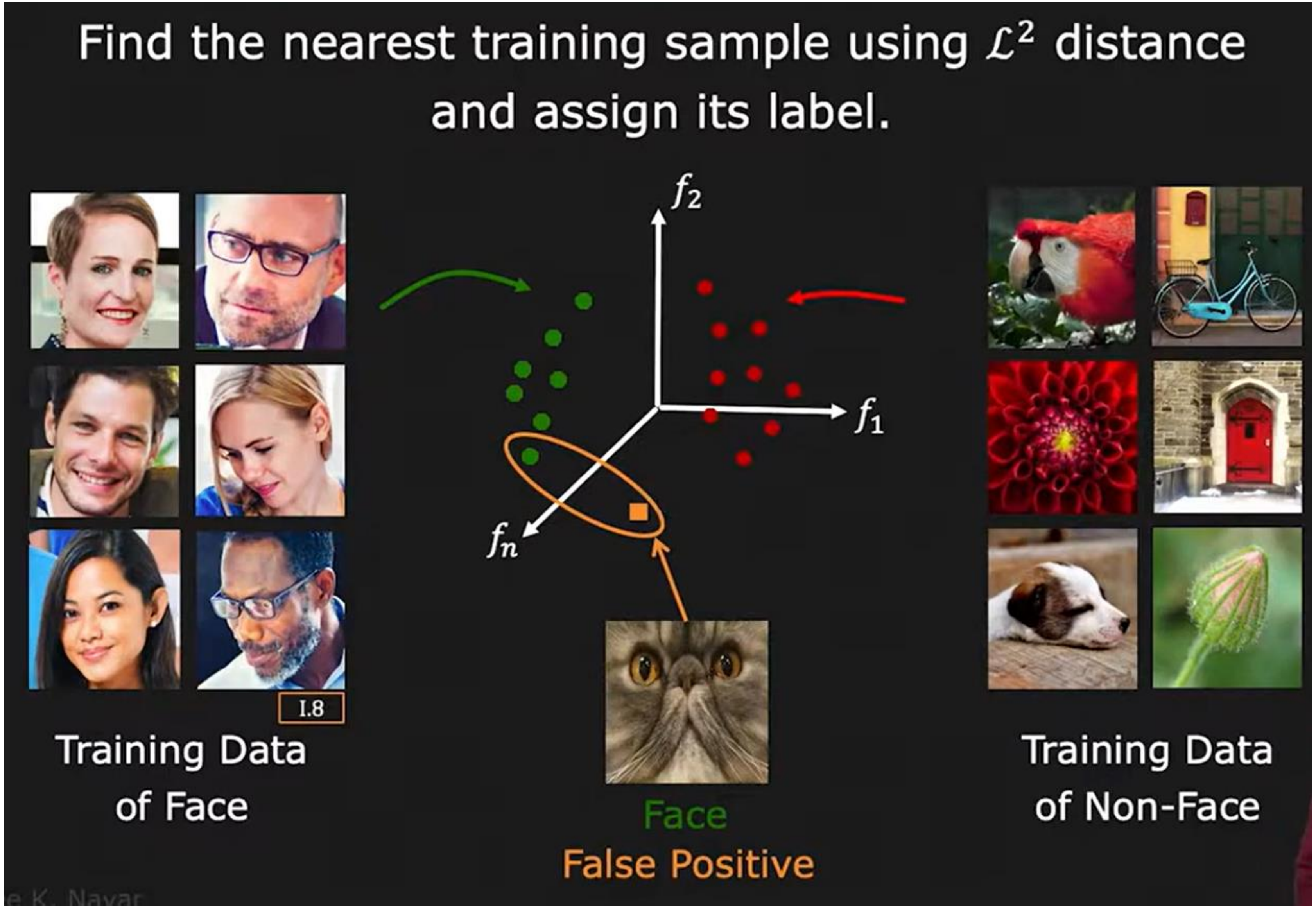
Training Data of Non-Face

Nearest Neighbor Classifier



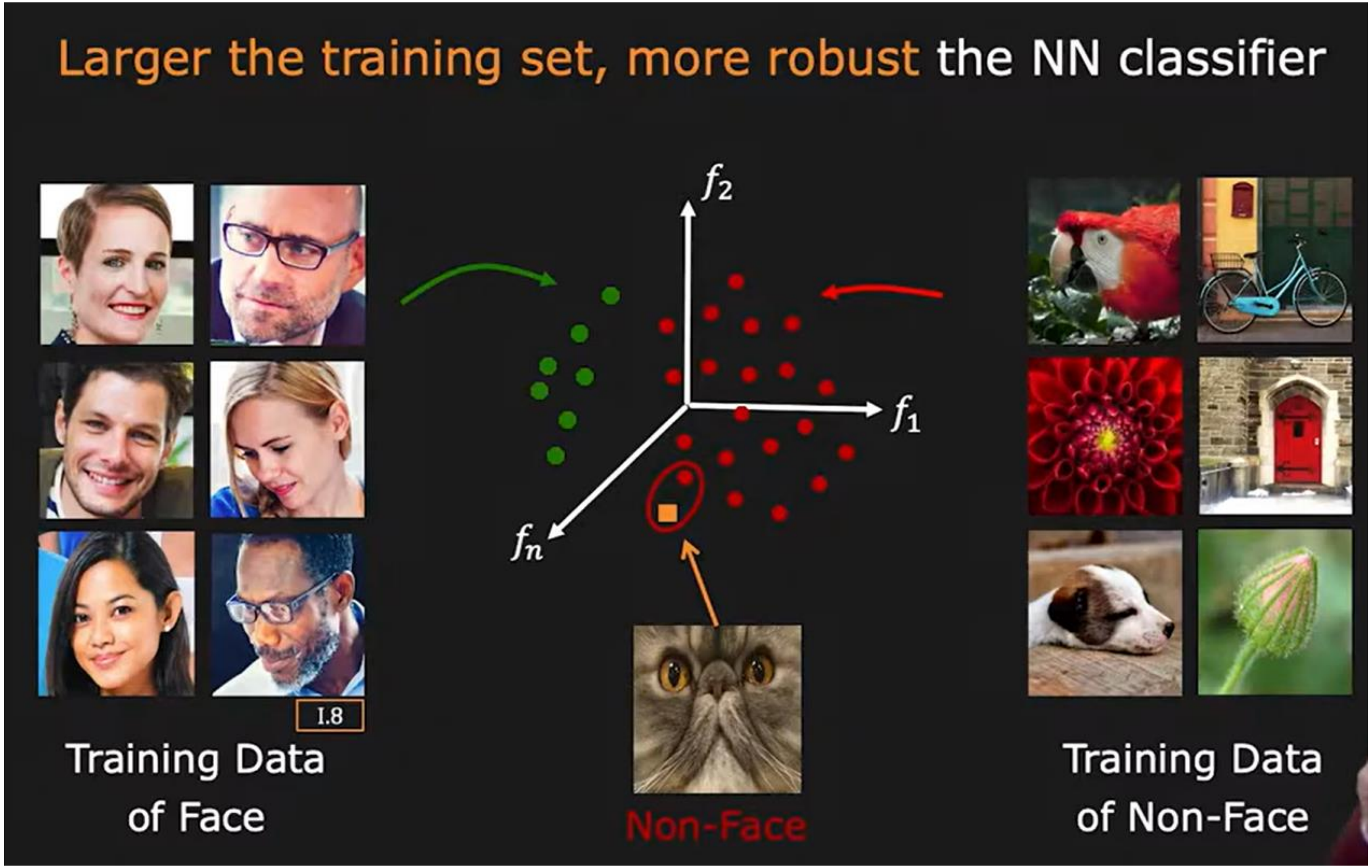
Decision engine that classifies a test image as face or not

Nearest Neighbor Classifier



Find the nearest training sample using $\mathcal{L}^2$ distance and assign its label.

Nearest Neighbor Classifier



Larger the training set, more robust the NN classifier

Expensive
From a
computation
stand point

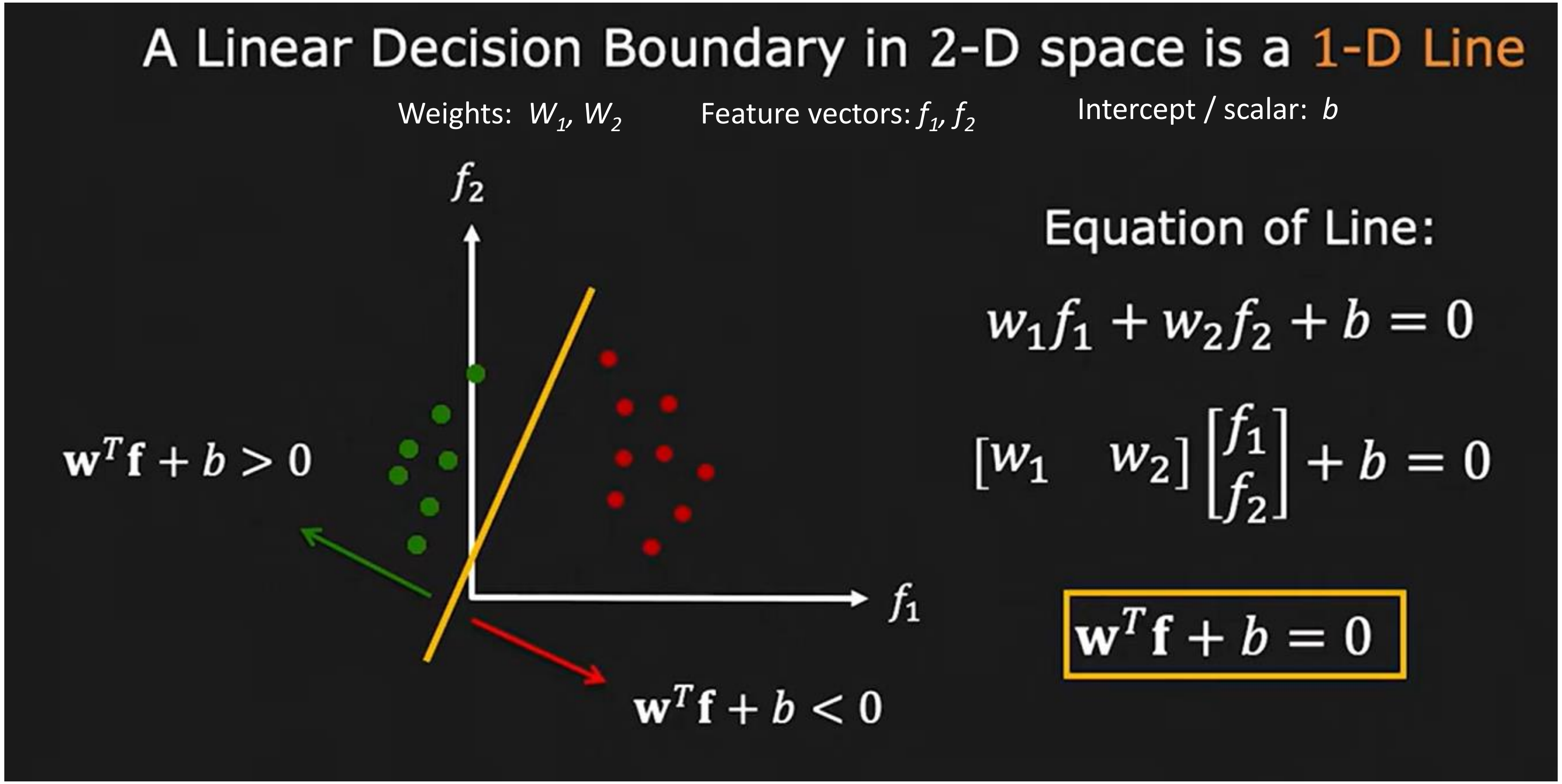
Support Vector Machines

(classify by decision boundaries)

Now we have a feature vector at each pixel in the image.

Use feature vector to classify: face vs non-face

Linear Decision Boundaries



A Linear Decision Boundary in 2-D space is a 1-D Line

Weights: W_1, W_2

Feature vectors: f_1, f_2

Intercept / scalar: b

f_2
f_1
$\mathbf{w}^T\mathbf{f} + b > 0$
$\mathbf{w}^T\mathbf{f} + b < 0$

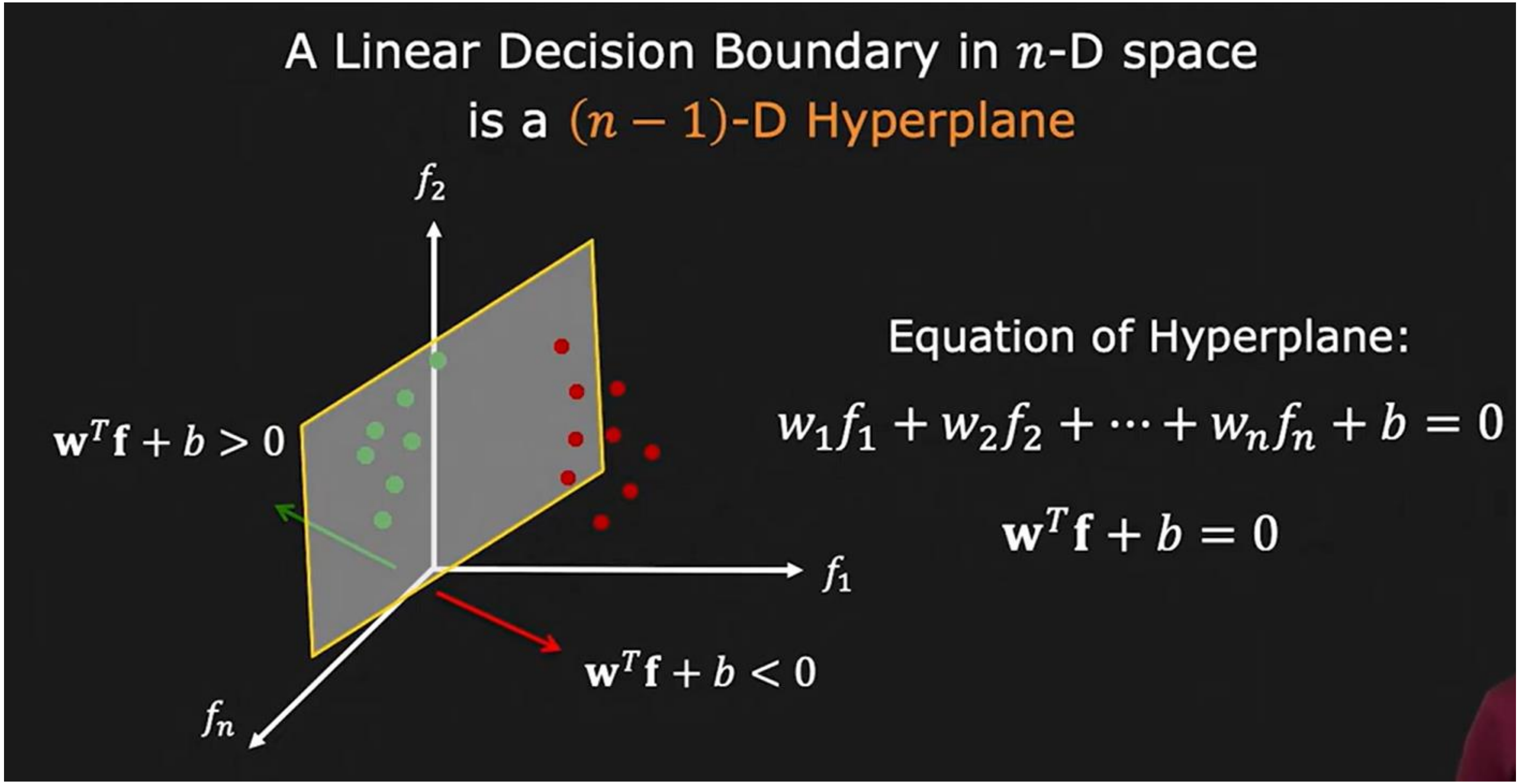
Equation of Line:

$$w_1 f_1 + w_2 f_2 + b = 0$$

$$\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} + b = 0$$

$$\boxed{\mathbf{w}^T\mathbf{f} + b = 0}$$

Linear Decision Boundaries



A Linear Decision Boundary in n-D space is a $(n-1)$-D Hyperplane

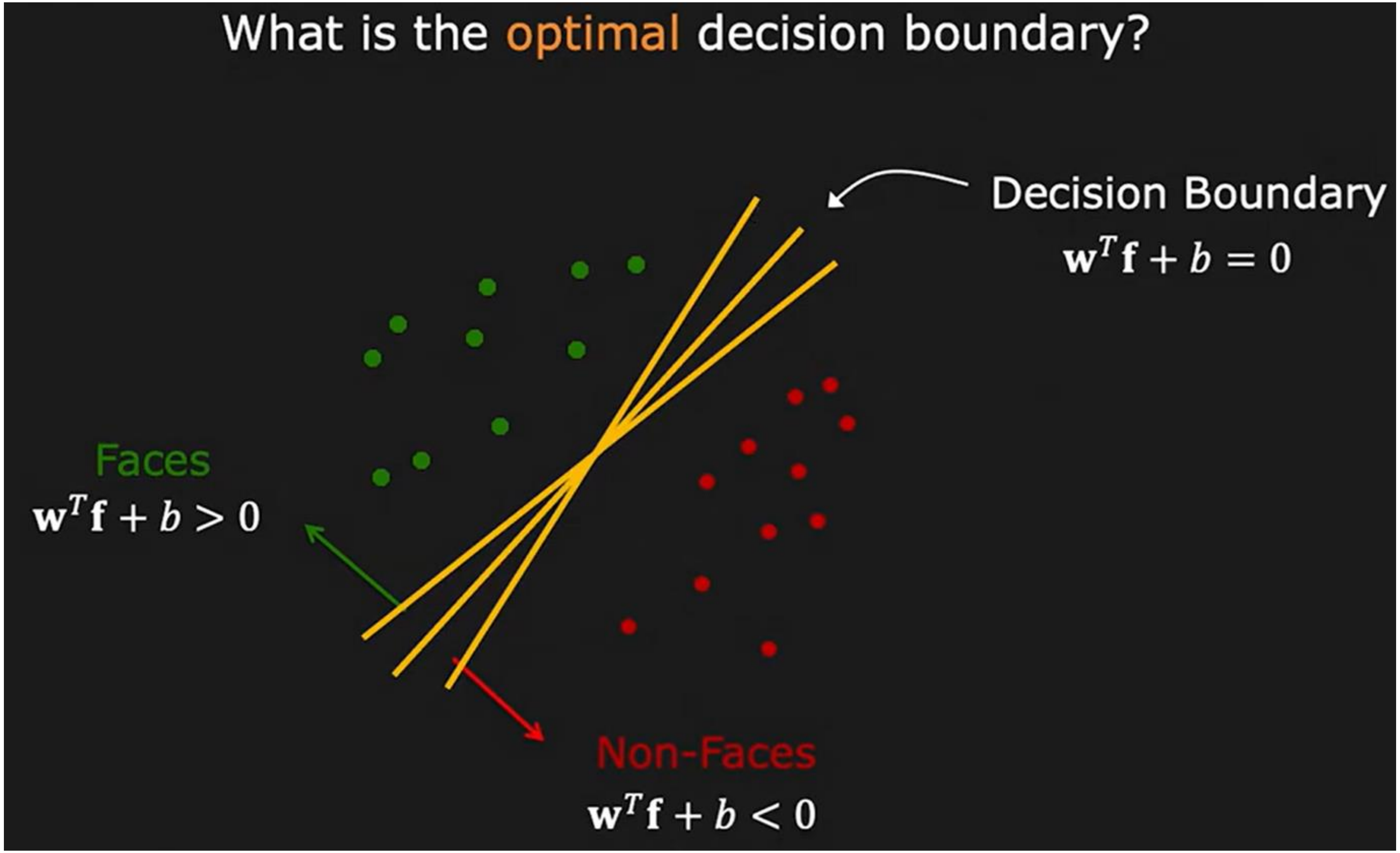
Equation of Hyperplane:

$$w_1 f_1 + w_2 f_2 + \cdots + w_n f_n + b = 0$$

$$\mathbf{w}^T \mathbf{f} + b = 0$$

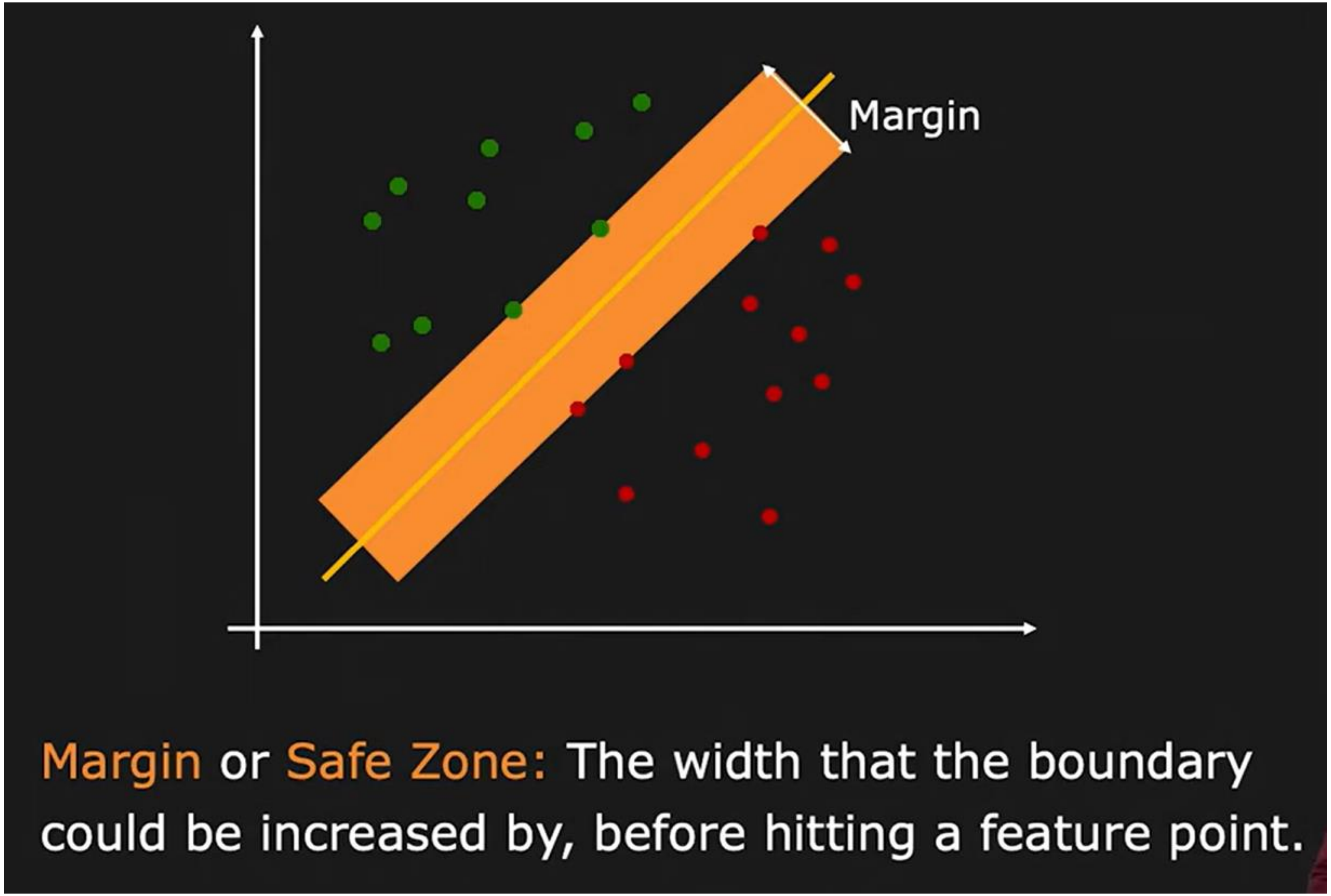
f_2
$\mathbf{w}^T \mathbf{f} + b > 0$
f_1
$\mathbf{w}^T \mathbf{f} + b < 0$
f_n

Decision Boundary (*w*, *b*)



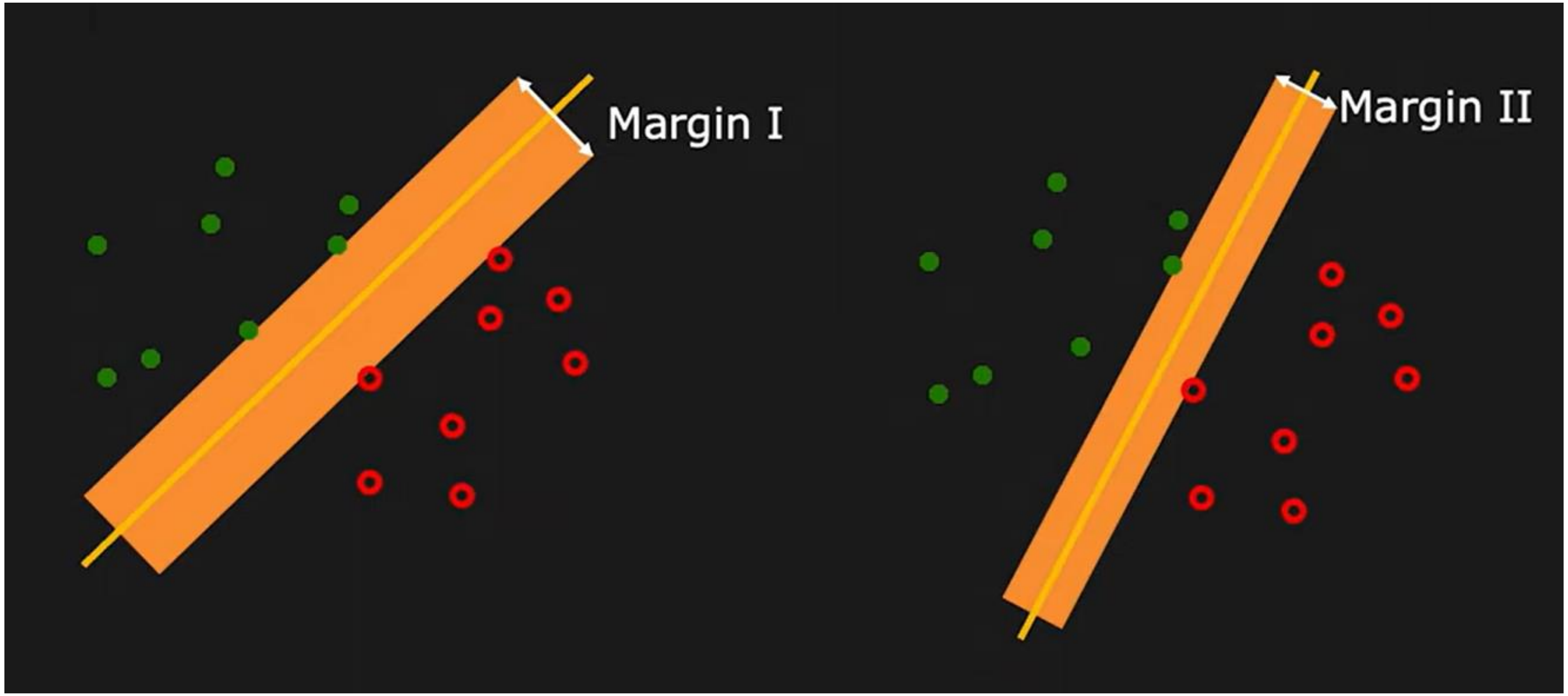
What is the optimal decision boundary?
Decision Boundary
$\mathbf{w}^T\mathbf{f} + b = 0$
Faces
$\mathbf{w}^T\mathbf{f} + b > 0$
Non-Faces
$\mathbf{w}^T\mathbf{f} + b < 0$

Evaluating a Decision Boundary

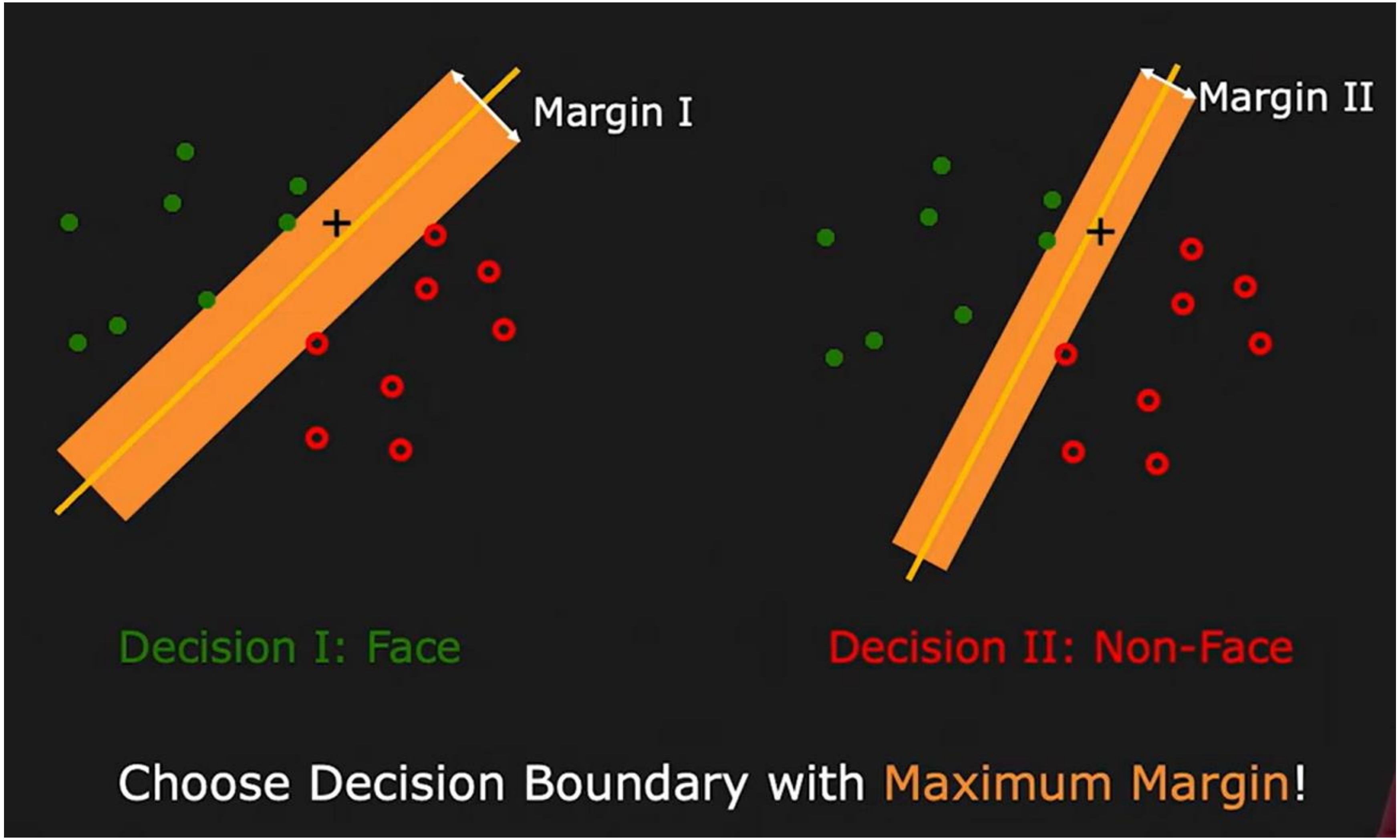


Margin or Safe Zone: The width that the boundary could be increased by, before hitting a feature point.

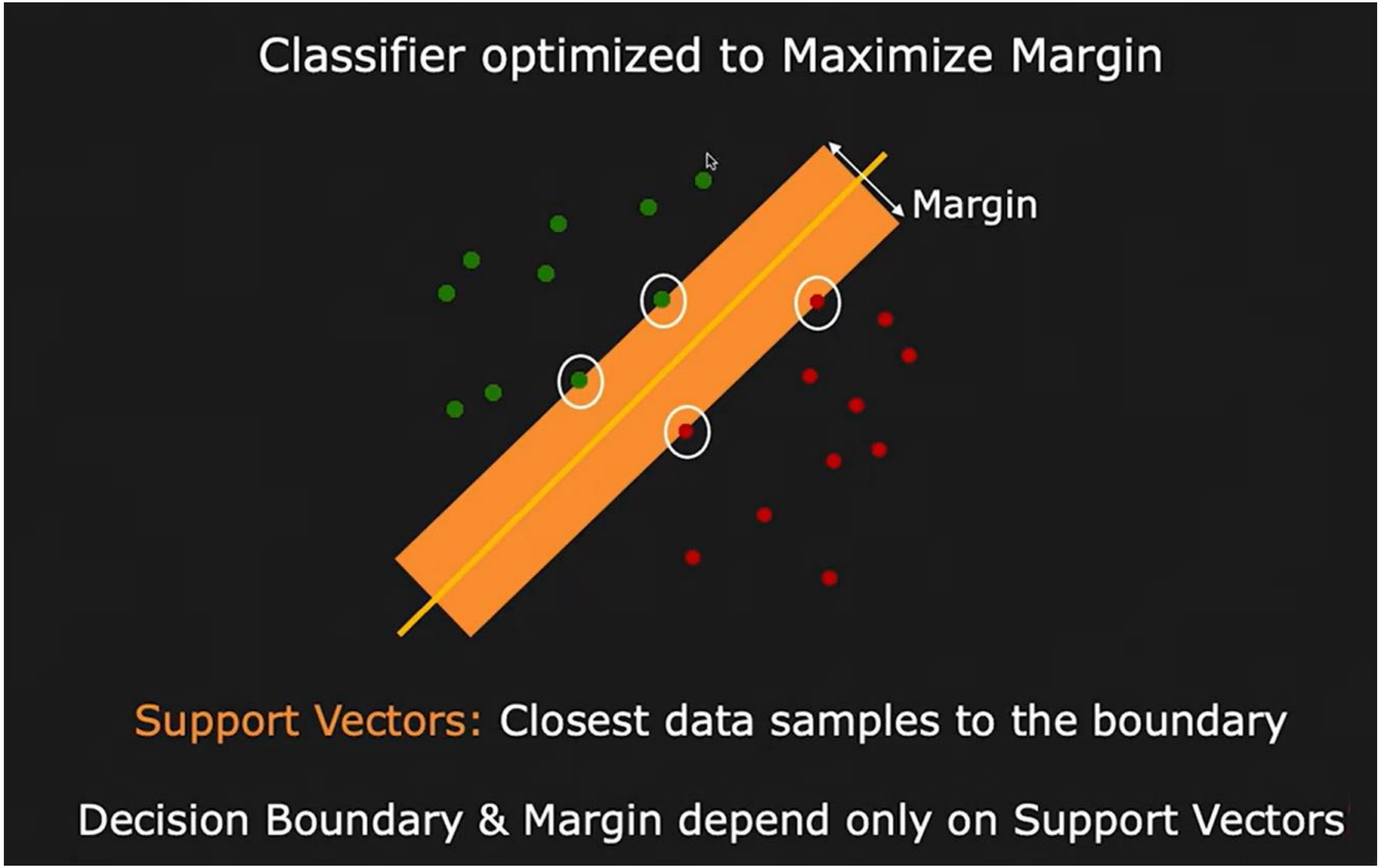
Evaluating a Decision Boundary



Evaluating a Decision Boundary



Support Vector Machine (SVM)

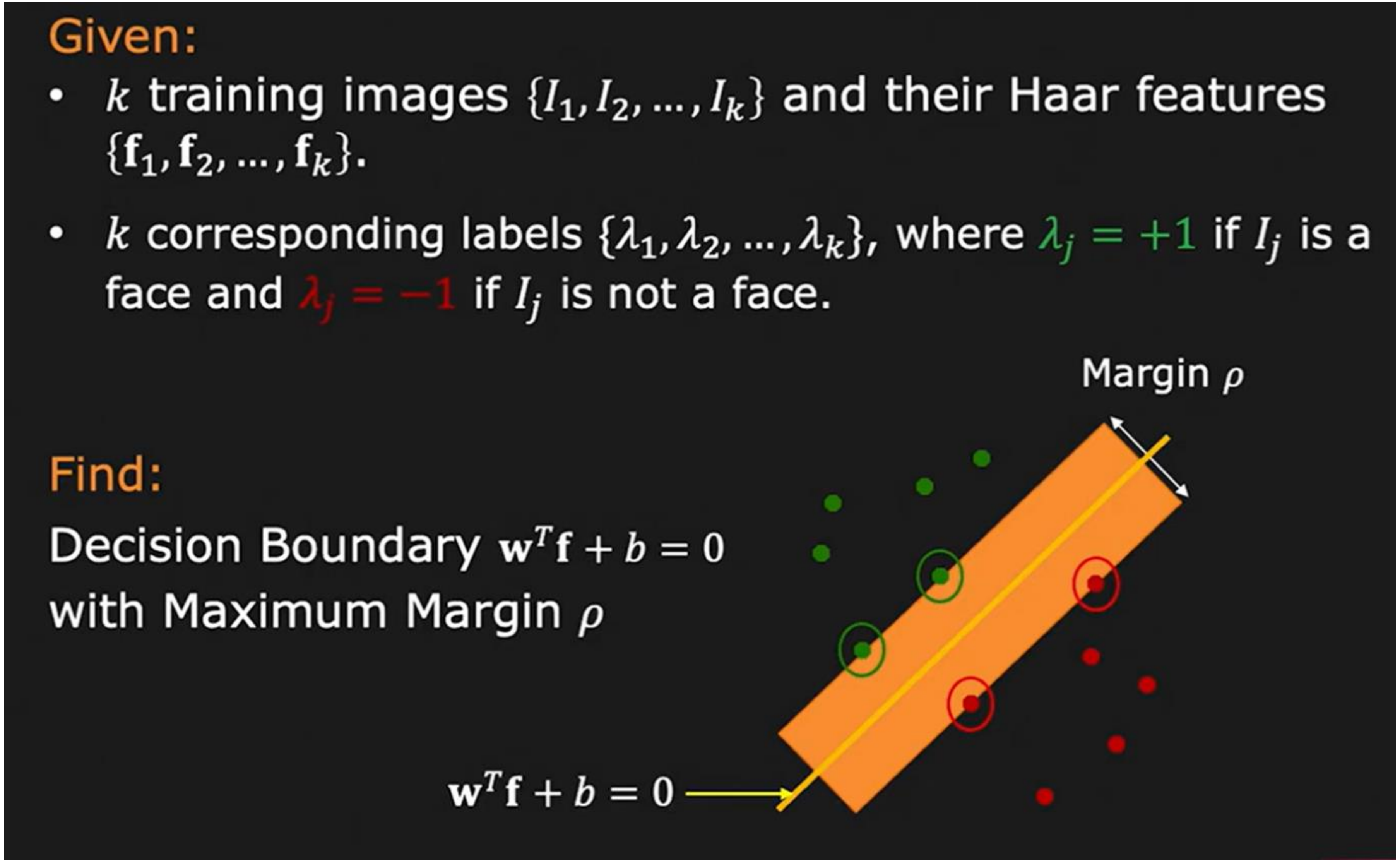


Classifier optimized to Maximize Margin

Support Vectors: Closest data samples to the boundary

Decision Boundary & Margin depend only on Support Vectors

Support Vector Machine (SVM)



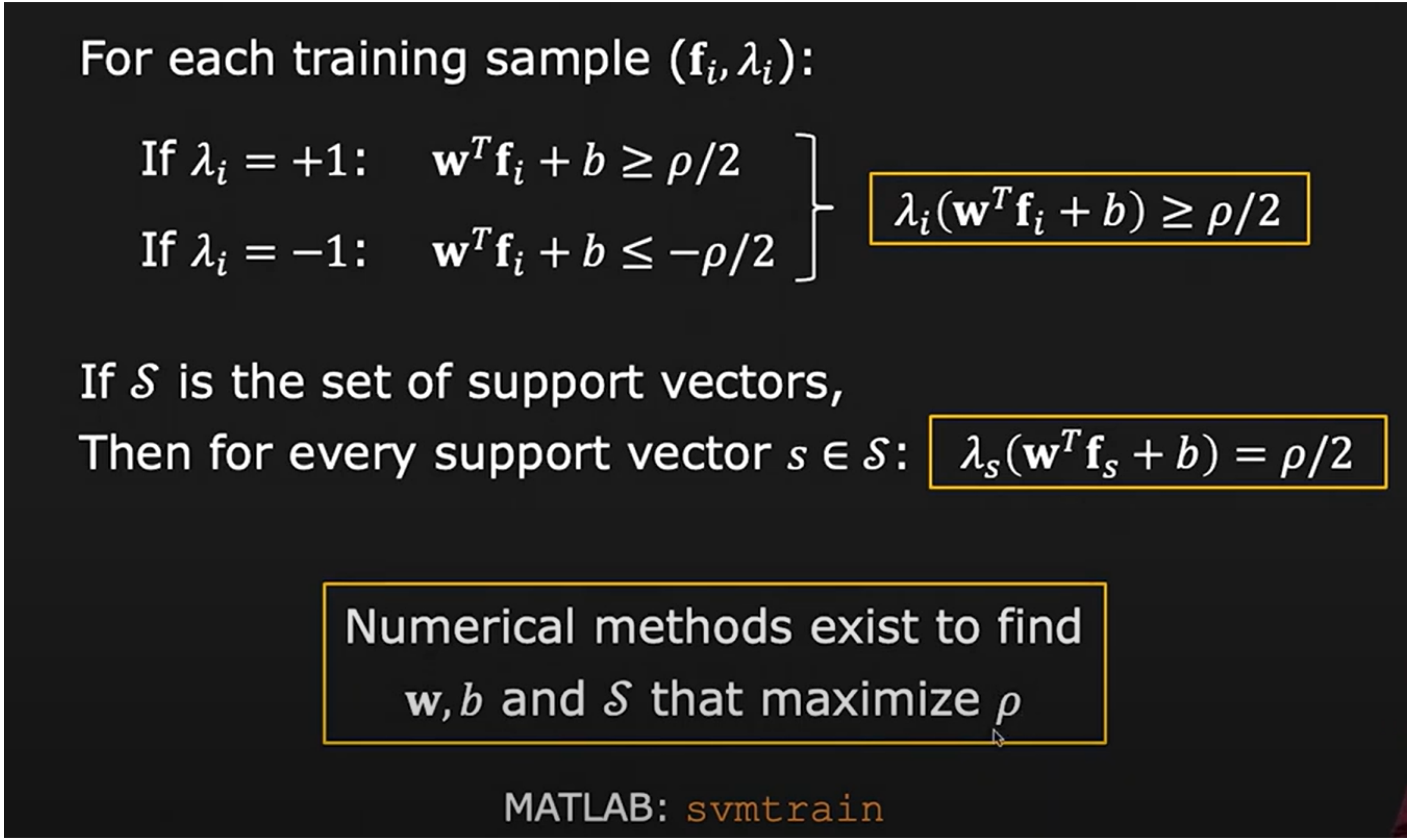
Given:

- k training images $\{I_1, I_2, \dots, I_k\}$ and their Haar features $\{\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_k\}$.
- k corresponding labels $\{\lambda_1, \lambda_2, \dots, \lambda_k\}$, where $\lambda_j = +1$ if I_j is a face and $\lambda_j = -1$ if I_j is not a face.

Find:

Decision Boundary $\mathbf{w}^T\mathbf{f} + b = 0$ with Maximum Margin ρ

Finding the decision boundary (*W*, *b*)



For each training sample $(\mathbf{f}_i, \lambda_i)$:

$$\left.\begin{array}{ll} \text{If } \lambda_i = +1: & \mathbf{w}^T\mathbf{f}_i + b \geq \rho/2 \\ \text{If } \lambda_i = -1: & \mathbf{w}^T\mathbf{f}_i + b \leq -\rho/2 \end{array}\right\} \boxed{\lambda_i(\mathbf{w}^T\mathbf{f}_i + b) \geq \rho/2}$$

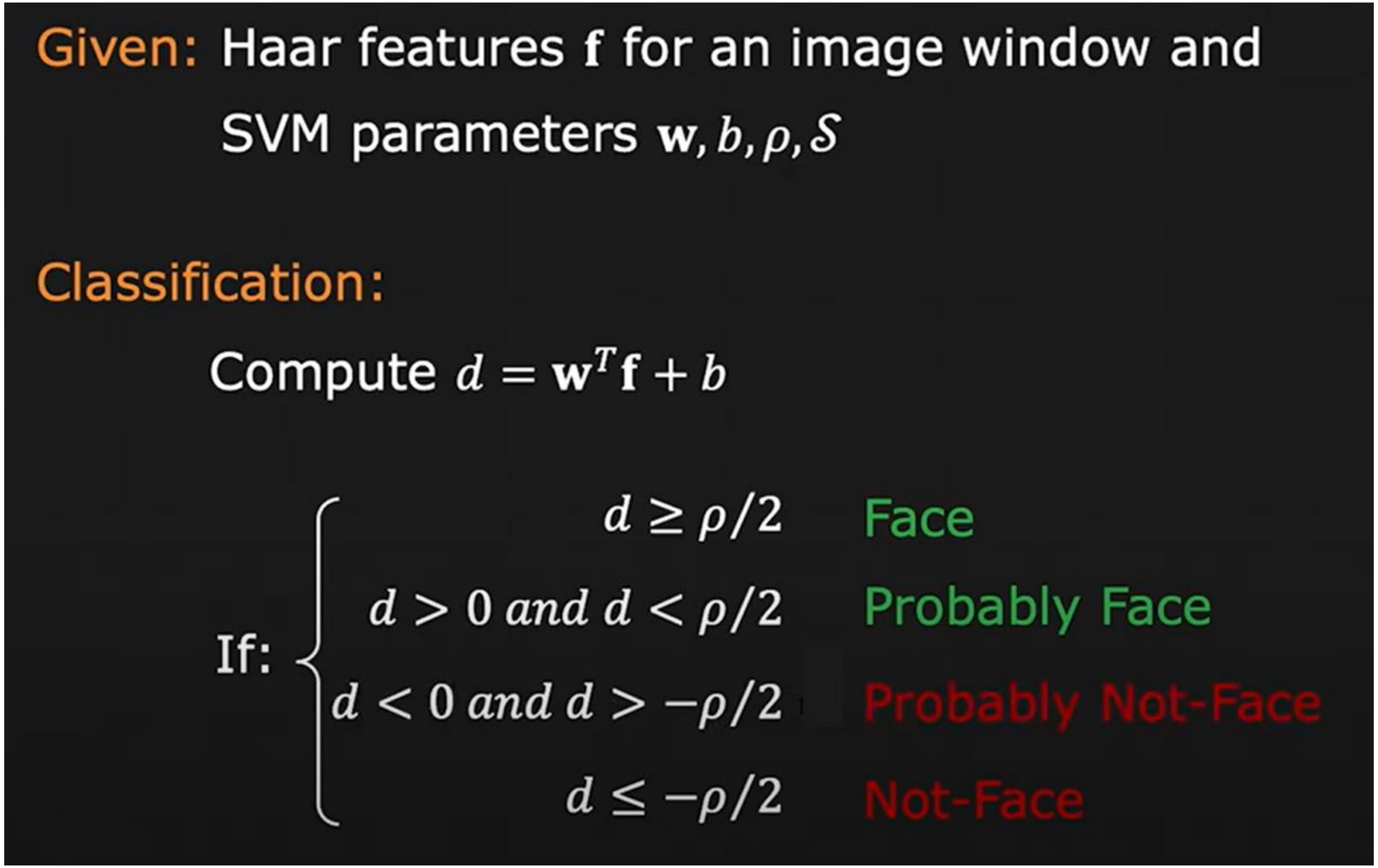
If $\mathcal{S}$ is the set of support vectors,

Then for every support vector $s \in \mathcal{S}$: $\boxed{\lambda_s(\mathbf{w}^T\mathbf{f}_s + b) = \rho/2}$

Numerical methods exist to find $\mathbf{w}, b$ and $\mathcal{S}$ that maximize ρ

MATLAB: `svmtrain`

Classification using SVM



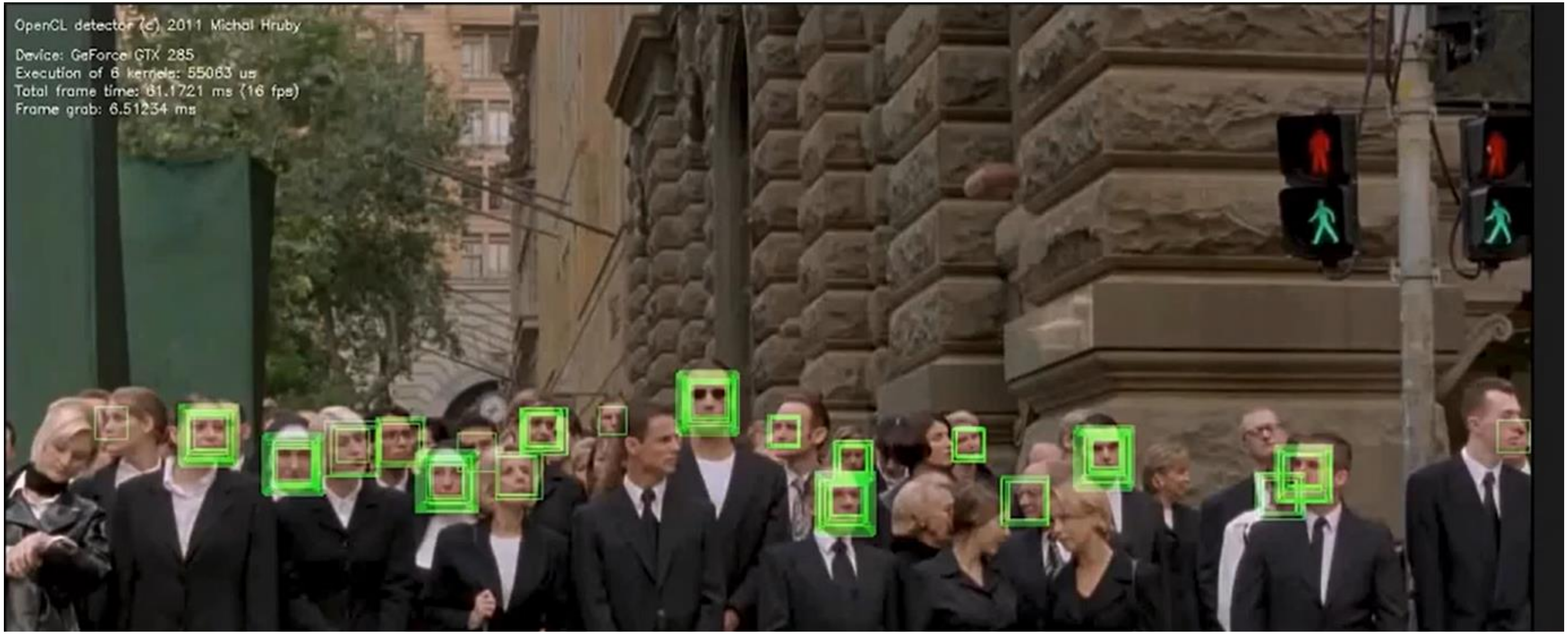
Given: Haar features $\mathbf{f}$ for an image window and SVM parameters $\mathbf{w}, b, \rho, \mathcal{S}$

Classification:

Compute $d = \mathbf{w}^T\mathbf{f} + b$

$$\text{If: } \begin{cases} d \geq \rho/2 & \text{Face} \\ d > 0 \; and \; d < \rho/2 & \text{Probably Face} \\ d < 0 \; and \; d > -\rho/2 & \text{Probably Not-Face} \\ d \leq -\rho/2 & \text{Not-Face} \end{cases}$$

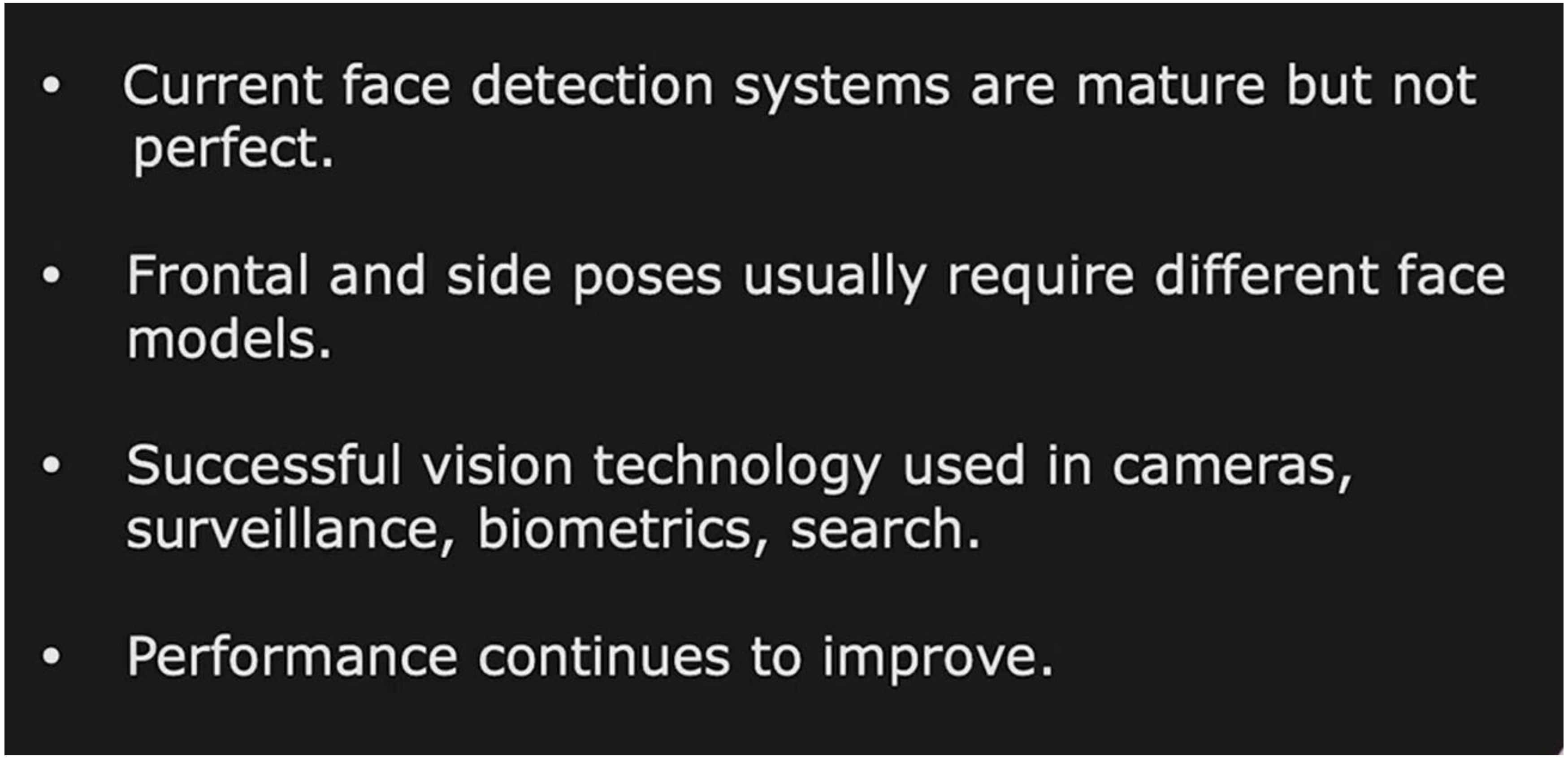
Face Detection Results



```
OpenCL detector (c) 2011 Michal Hruby

Device: GeForce GTX 285
Execution of 6 kernels: 55063 us
Total frame time: 61.1721 ms (16 fps)
Frame grab: 6.51234 ms
```

Remarks



- Current face detection systems are mature but not perfect.
- Frontal and side poses usually require different face models.
- Successful vision technology used in cameras, surveillance, biometrics, search.
- Performance continues to improve.

References

- Image Stitching | face Detection play list by Shree K Nayar (Columbia university): https://www.youtube.com/playlist?list=PL2zRqk16wsdp8KbDfHKvPYNGF2L-zQASc
- https://medium.com/@aaronward6210/facial-detection-understanding-viola-jones-algorithm-116d1a9db218
- https://www.mygreatlearning.com/blog/viola-jones-algorithm/