# Lecture 07 - Image Descriptors

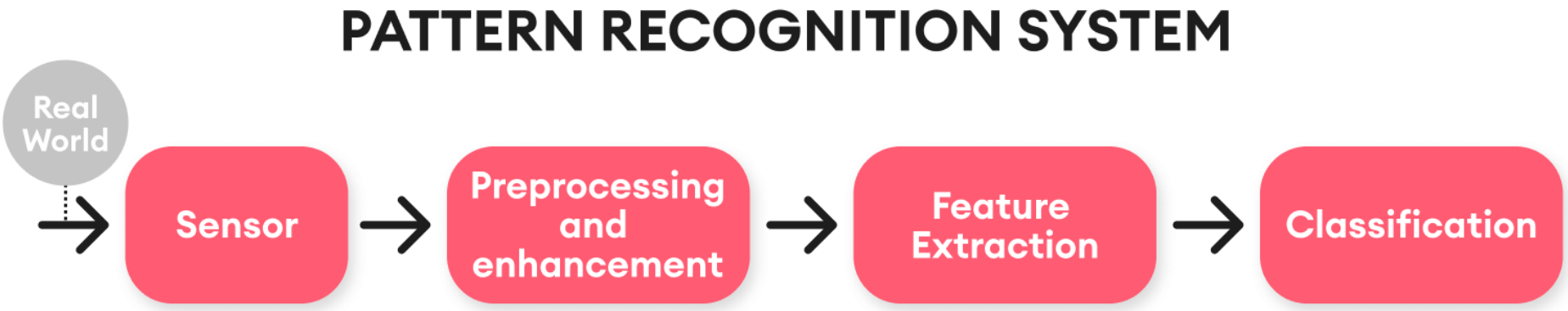
Prof. André Gustavo Hochuli

gustavo.hochuli@pucpr.br
aghochuli@ppgia.pucpr.br

# Topics

- Discussion of Lecture #06
  - Feature Vector
  - Horizontal and Vertical Projections
- Image Descriptors
  - Shape (HoG)
  - Textures (LBP, GABOR)
- Classification
  - K-NN
- Practice

# Computer Vision & Pattern Recognition Pipeline



# Image Descriptors – Shape

- Moments
  - Values that carry both spatial and intensity information (shape)
    - Weighted average of all pixel's intensities
    - I(x,y) ➔ pixel coordinates of input
    - Powers, *p* and *q,* are the weights of the horizontal and vertical dimensions

$$M_{pq} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} x^p y^q I(x,y)$$

- HuMoments (Hu 1962)
  - Translation and Scale Invariant

$$h_1 = \eta_{20} + \eta_{02}$$

$$h_2 = (\eta_{20} - \eta_{02})^2 + 4(\eta_{11})^2$$

$$h_3 = (\eta_{30} - 3\eta_{12})^2 + 3(\eta_{03} - 3\eta_{21})^2$$

$$h_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2$$

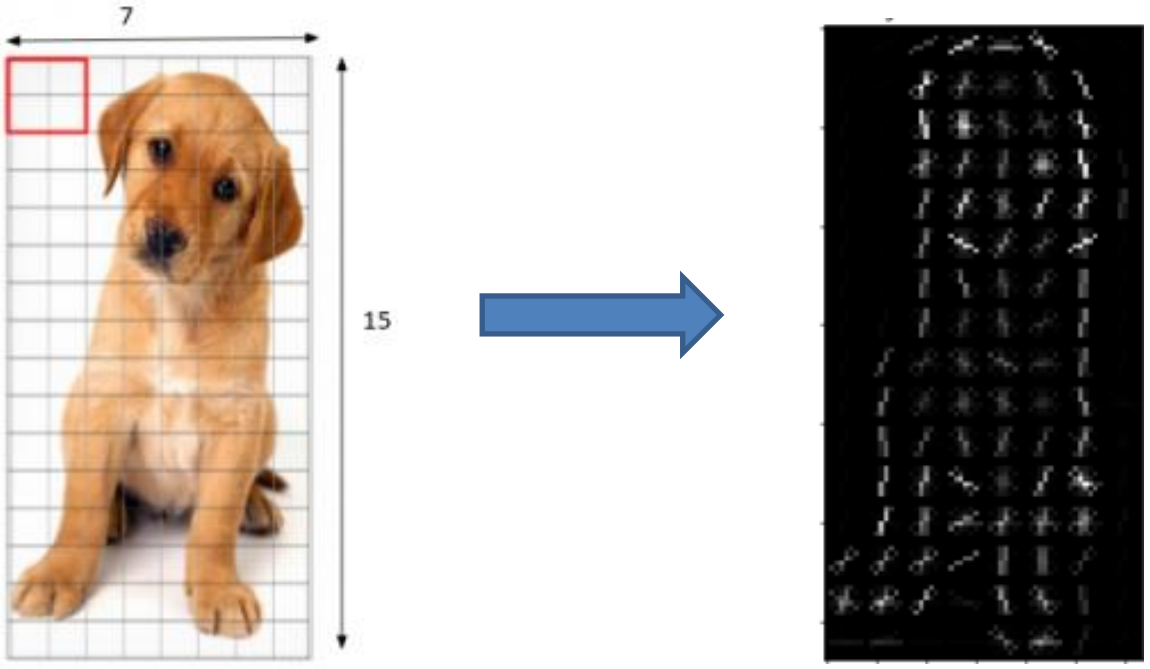
$$h_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + (3\eta_{21} - \eta_{03})(\eta_{03} + \eta_{21})[3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]$$

$$h_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - 7(\eta_{03} + \eta_{21})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})$$

$$h_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + (\eta_{30} - 3\eta_{12})(\eta_{03} + \eta_{21})[3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]$$

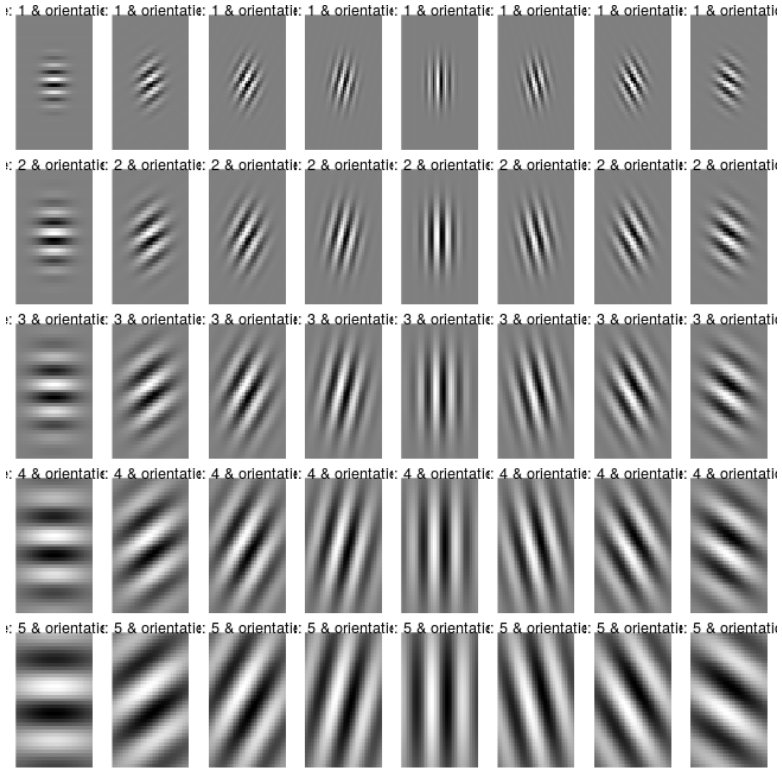
# Image Descriptors – Shape

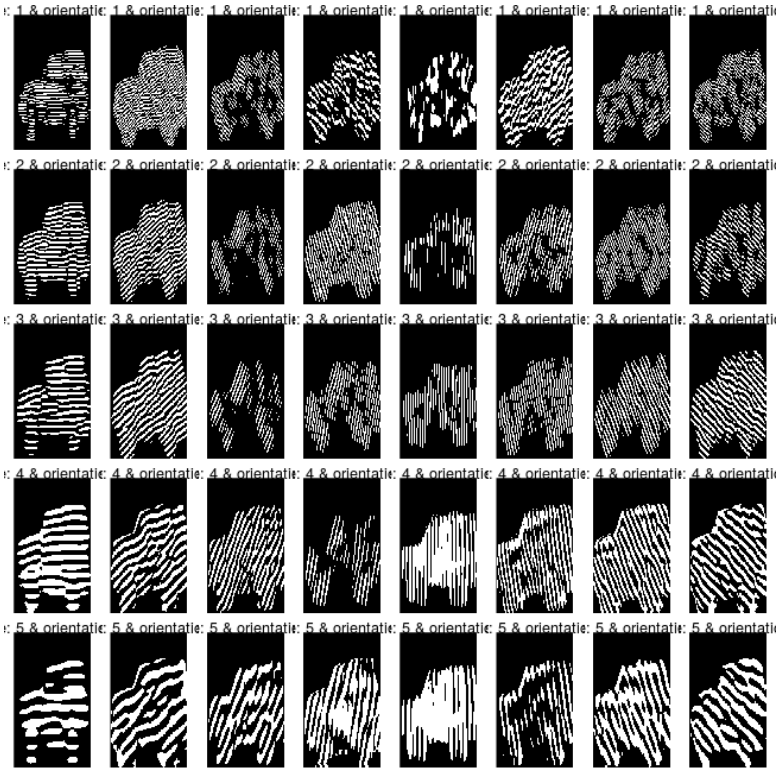
- HoG – Histogram of Oriented Gradients
  - Computes the gradient and orientation of edges
  - Use a kernel to compute the Gradients (i.e 9x1)
  - Patch-Based Histogram (8x8, 16x16..)



# Image Descriptors – Texture

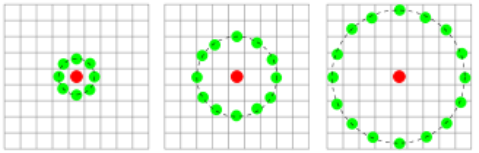
- Gabor Filters
  - Convolves the image using several Gaussian Kernels (Kernel Bank)

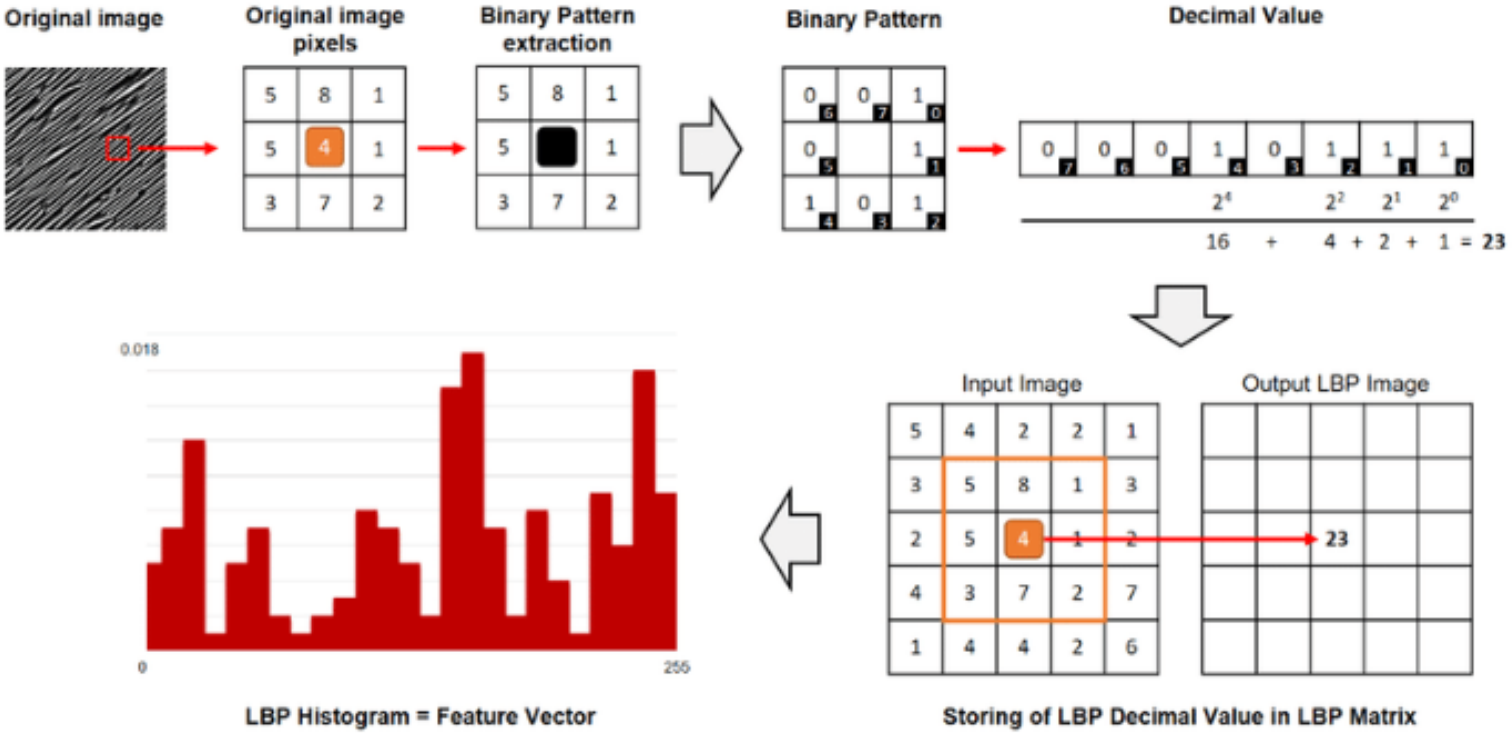




# Image Descriptors – Texture

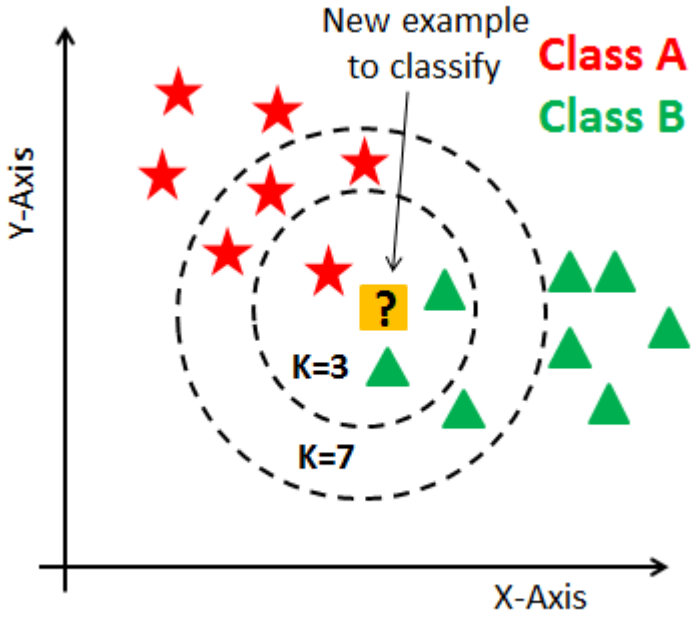
- Local Binary Patterns
  - Convolves the image using a Circular Kernel
  - The resulting pixel is computed in the binary neighborhood





# Classification

- KNN
  - Computes the similarity in a feature space (Euclidian Distance, Manhattan….)
  - The K-Nearest Neighbors determines the class (Majority Vote)




$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

# Let’s Code

- LINK