Intel AI Academy

IMAGE FEATURES

LEGAL NOTICES AND DISCLAIMERS

WHAT ARE IMAGE FEATURES?

IMAGE FEATURES

Image features are *interesting* locations in an image.

We want to be able to apply…

- Feature detection
  Finding regions, when moved, that cause *maximal variation*
- Features description
  Building a *good context* around a feature

We use this information about these *interesting* locations for a wide variety of uses.

FEATURE DETECTION

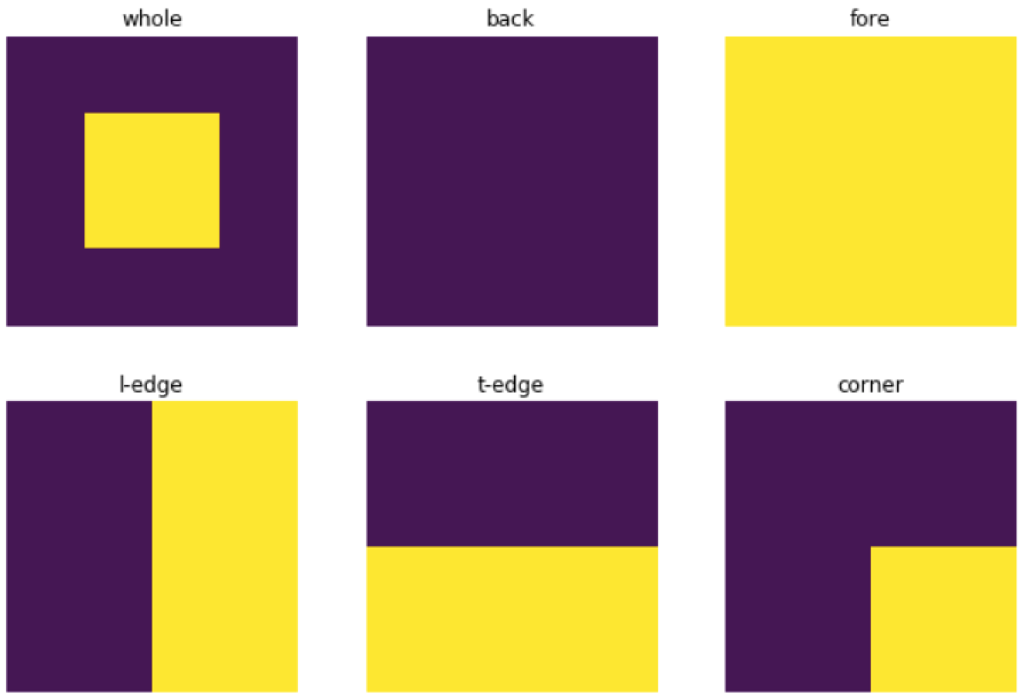
Background feature matches any background cell.

Foreground feature matches any foreground cell.

L-edge matches any vertical edge.

T-edge matches any horizontal edge.

But, CORNER only has one match!!!



USES OF IMAGE FEATURES: MATCHING

CLASSIFICATION (OBJECT RECOGNITION)

It is a Crayon

What is this object?

TRACKING OF FEATURES IN MOTION

Image reference: https://commons.wikimedia.org/wiki/File:Madison_Square_Garden_food_court.jpg

IMAGE STITCHING

OBJECT DETECTION

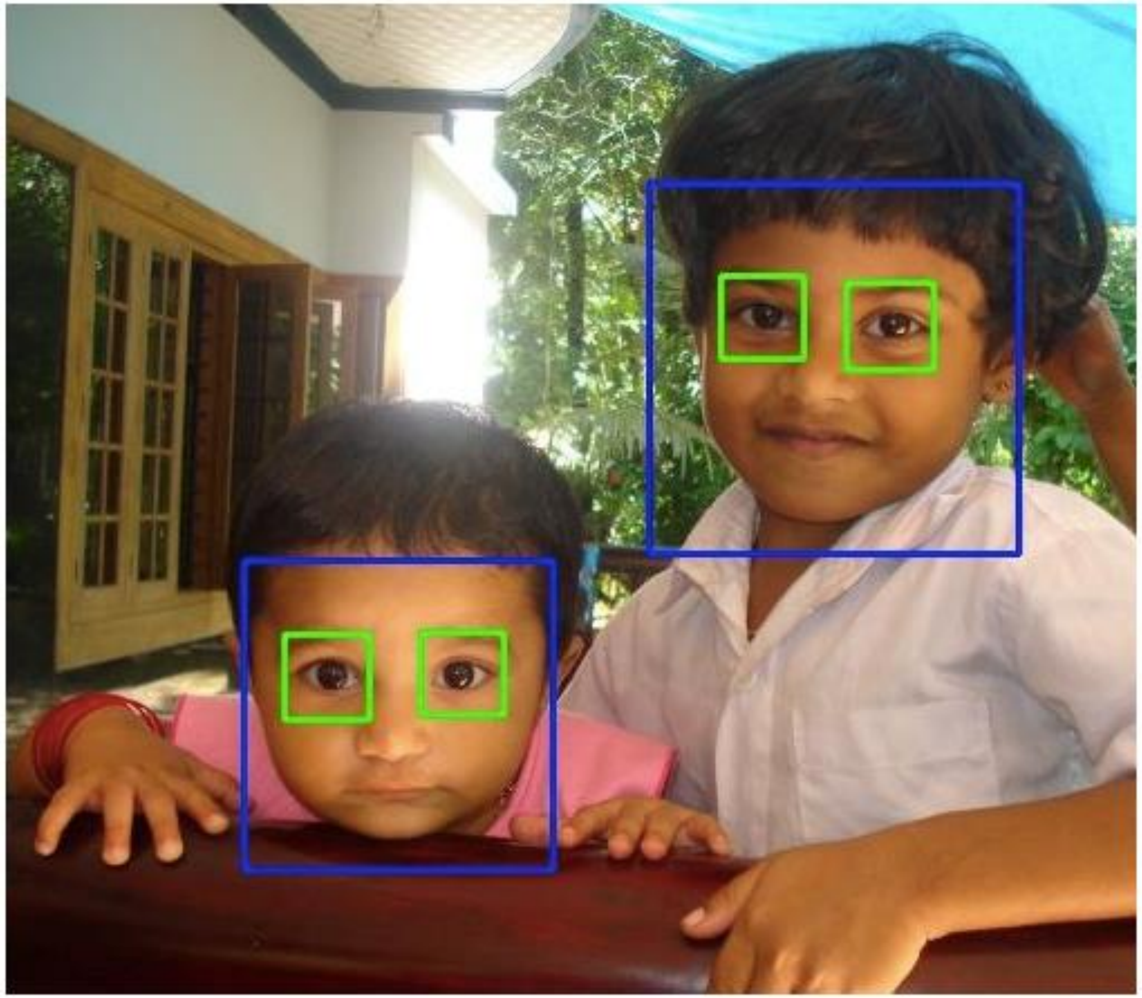


Image from OpenCV documentation: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html

TECHNIQUES FOR IMAGE FEATURES

Techniques that can be used for *any* of these use-cases:

- Corners
- HOG
- SIFT
- SURF
- FAST
- BRIEF
- However, these techniques are typically developed by a research group to solve one use case.

CORNERS

CORNERS

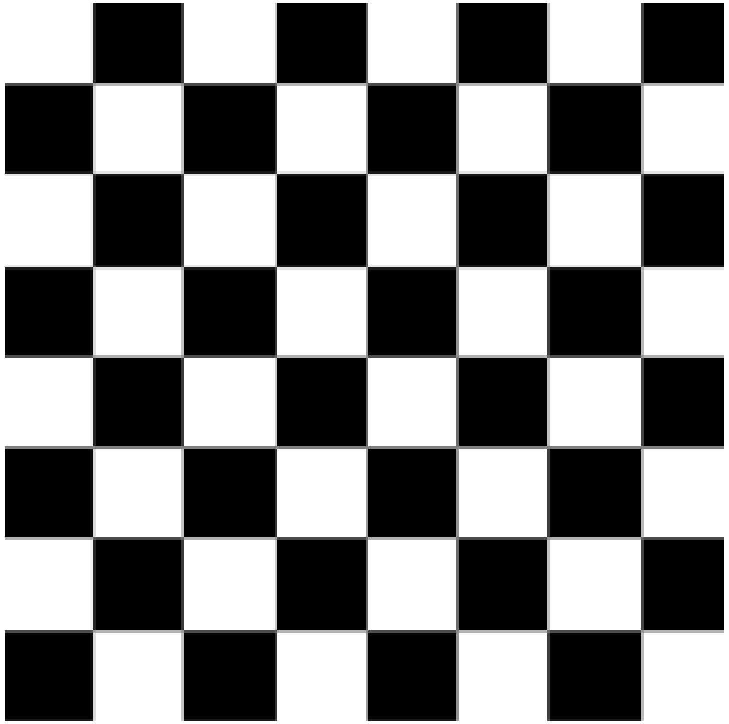
An edge is indicated by a strong derivative.

- This is helpful to find an edge, but all features along that edge may look the same.
- We want a better way to identify an *interesting* feature.

Corners are *edges* in two orthogonal directions.

- Hopefully, a corner will have enough information to identify our feature over multiple images, or frames.

HARRIS CORNERS



Harris Corners

Corner on Original Image

HARRIS CORNERS

Used for sparse feature matching.

Use a Gaussian weighted window.

- Allows detection when there is in-plane rotation of an *interesting* feature
- Down-weights edge-like features

Compute a scalar interest measure.

Find local maxima.

HARRIS CORNERS

Harris corners are defined as second-order derivatives.

We create a second-derivative *Hessian* image.

A corner is identified when this matrix has two large eigenvalues.

Look at eigenvalues of the autocorrelation matrix.

- Eigenvectors tell us the orientation of a matrix.
- Eigenvalues tell us the scale in those directions.
- Either minimum eigenvalue, or something more complicated like:

$$\lambda_0\lambda_1 - 0.06(\lambda_0\lambda_1)^2$$

HARRIS CORNERS AND SHI-TOMASI

Shi-Tomasi modified the Harris corner.

In both cases we take the:

- Correlation between gradients in x and y directions

Harris used:

$$\lambda_0\lambda_1 - 0.06(\lambda_0\lambda_1)^2 < Threshold$$

Shi-Tomasi used:

$$\lambda_0 < Threshold$$

- Invariant to rotation (because of eigenvalues)

SUB-PIXEL METHODS

Harris and Shi-Tomasi corners are good for feature recognition and used…

- When you are extracting geometric measurements
- When you need higher resolution feature recognition

If I am a…

Biologist trying to generate a 3D reconstruction of a blood vessel

Or a marine determining the location of a target in a satellite image

I need sub-pixel coordinates.

SUB-PIXEL METHODS

If we know an approximate (pixel level) location C_{approx} of a corner we can find a better corner $C_{improved}$ by:

- Sampling a region of points P around C_{approx}.
- Setting up a series of equations that capture the directional similarities between each p of P and $C_{improved}$
  Note, we don't know $C_{improved}$!
- Solve the equations. The result is $C_{improved}$.
- Repeat this process.

SUB-PIXEL METHODS

Similarity measure has these important properties:

- Similarity(0,any) = Similarity(any,0) = 0
- Similarity(perpendicular directions) = 0

For us, our directions are:

1. The gradient at a region point p
2. The direction from C_approx to p

SUB-PIXEL METHODS

When the similarity between the direction of the gradient and the direction from C_approx to p is 0

- The directions must be perpendicular.

In turn, this means that the direction lies along an edge

- Multiple edges meet at a corner!

The similarity measure is the dot product between the gradient at p and the direction from C_{approx} to p.

$$<\nabla I(p), q - p \geq 0$$

HISTOGRAM OF ORIENTED GRADIENTS (HOG)

HISTOGRAM OF ORIENTED GRADIENTS

Developed to detect pedestrians in a scene

Techniques focus on detection accuracy instead of computing speed and efficiency.

- Details about an object are determined as accurately as possible.

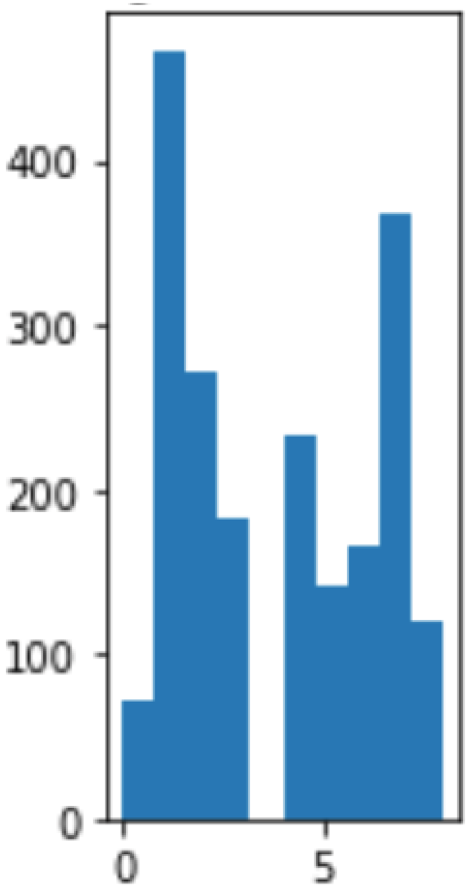
HOGs are a distribution of the directions of the gradients.

The histogram confers distribution

The oriented gradients confer direction of gradient

HISTOGRAM OF ORIENTED GRADIENTS

Features produced by HOG are the histograms from the cells.



CONSTANTS ARE TUNED FOR RECOGNIZING HUMANS

HOG is described in terms of one ROI, but many steps can be precomputed on the whole image.

Because HOG was originally developed for pedestrian recognition, many HOG ROIs are humans.

Humans are basically tall rectangles.

Therefore, we will make patches around an ROI by using a…

- Patch aspect of 2 (tall) to 1 (wide)
- Cell size of 8 x 8 because it captures human features like eyes

YOU CAN USE HOG ON OTHER, NON-HUMAN, FEATURES

Image

↓

ROI

↓

Frames for Normalization

↓

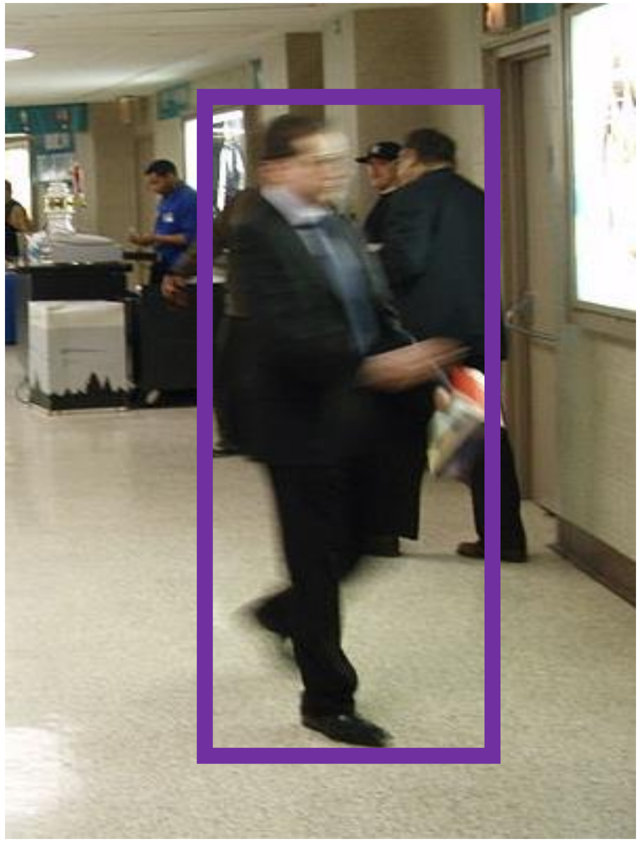
Cells

Tree image from: https://commons.wikimedia.org/wiki/Christmas_tree#/media/File:Piazza_Portanova_Natale_2008.jpg

HOG: AN EXAMPLE

Step 1: Take the image and create a patch of the ROI with a fixed aspect ratio of 1:2.

- Essentially, you are cropping the image around the ROI.

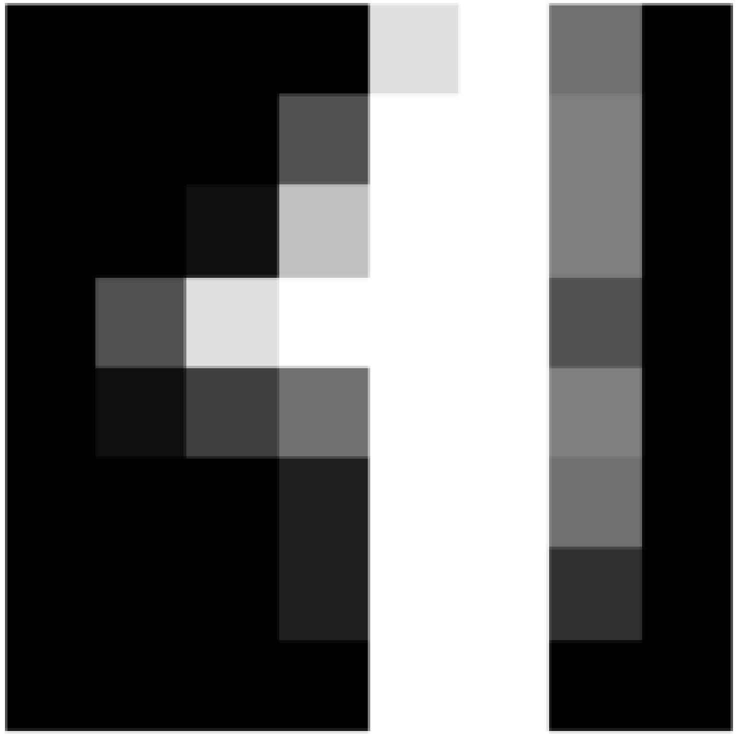


Patch (or ROI)

HOG: AN EXAMPLE

Step 2: Calculate the horizontal and vertical gradients for each color in the patch.

- Convolve with [-1,0,1] kernels
- For instance, you can use Sobel for this step



HOG: AN EXAMPLE

Step 3: Consider the component-wise gradients in polar coordinates.

- Take x, y gradients and convert to Θ
  - x, y → r, Θ [graphic]
  - At every pixel, take max r over the three color channels
    - r is magnitude
  - Associated direction is the theta of that max r

HOG: AN EXAMPLE

Step 4: The patch is divided into 8 x 8 cells, from which we make a histogram of gradients.

Each bucket, or bin, are angles (Θ)

- Θ = 0, 20, 40, ..., 160
  - This gives us nine total bins
- We assign the magnitude (r) of a pixel to its angle (th) bin.
- For angles between bins, we divide it proportionally to distance from the bins.

120	20	50
40	80	15

Direction (Q)

6	4	6
3	14	10

Magnitude (r)

0	20	40	60	80	100	120	140	160
	4	3	3			6		

HOG: AN EXAMPLE

Step 5: Create a mega-cell and associated mega-histogram with 36 total values.

- Use a sliding 16 x 16 window (of four histograms each time you slide the window).
- Normalize this window with respect to L2 norm to create one block of features.
- Slide the window over 8 pixels and repeat normalization.
  - Some pixels will be repeated normalized, but with respect to different neighborhoods extending differed directions.

FEATURES PER PATCH: SOME SIMPLE MATH

The result is 3,780 total features for one patch.

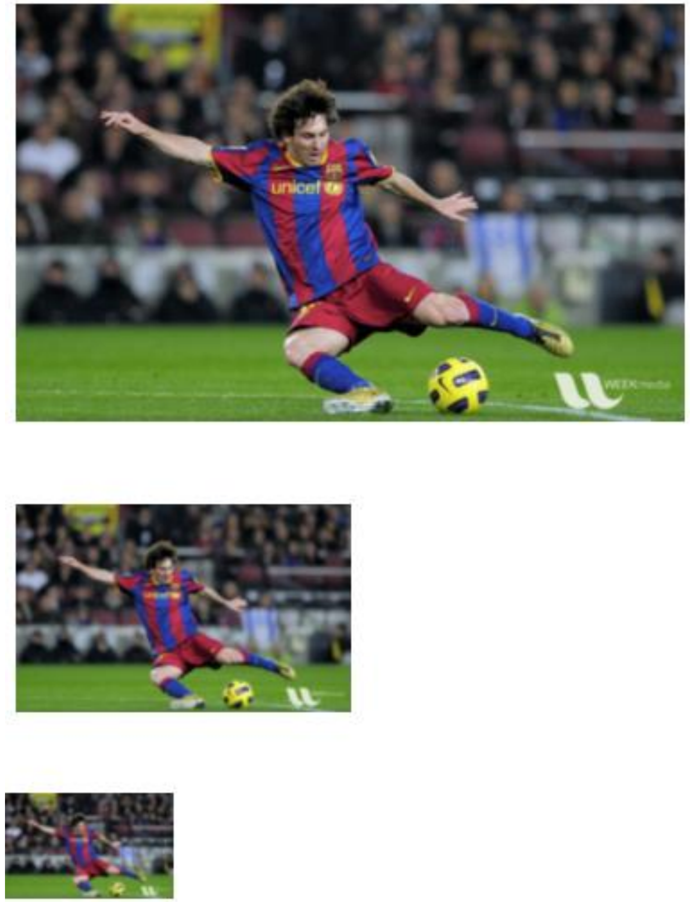
- On a 128 x 64 image we have 16 x 8 cells that are 8 x 8
- We have 15 x 7 mega-cells
- Which gives us 105 blocks of 36 features

SIFT, SURF, FAST, BRIEF

SCALE INVARIANT FEATURE TRANSFORM (SIFT)

Is a feature detection and description method that is invariant to scale

- Particularly helpful when analyzing larger corners



Scale-space filtering is used to detect keypoints with different scales.

- Uses difference of Gaussians (DoG) to approximate Laplacian of Gaussians (LoG).
- Finds maxima with respect to space and scale.
  - Space is across an image
  - Scale is up and down a DoG pyramid

SIFT: FEATURE DETECTION

First, we need to find candidate keypoints (feature detection).

We do this by finding extrema versus neighbors in a Gaussian pyramid.

This is on the same scale but can be high or lower resolution.

Then, we find candidates by filtering out points with low contrast and on edges.

SIFT: FEATURE DESCRIPTION

Now that feature detection is complete, we need to describe the feature.

- The neighborhood is proportional to level of the extrema
- Estimate orientation in neighborhood around the keypoint
- Compute a histogram of gradients in this neighborhood
- Normalize histogram

SIFT: MATCHING

Lastly, we will use the feature description to match the feature.

- We will use the Euclidean distance between keypoint descriptors

SPEEDED UP ROBUST FEATURES (SURF)

Similar to SIFT, but relies on several approximations and tricks to speed the processes.

Uses a box filter to approximate Gaussian smoothing.

- Integral images are easy to compute

Hessian matrix is used to find both extrema and scale.

- Matrix of mixed second-order derivatives

SURF: FEATURE DETECTION

Computed via its *determinant*

- Eigenvalues are like scale factors

- Determinant is like a volume
  - Multiply the eigenvalues together like dimensions of a cube

- The determinant of the matrix of partial second-order derivatives! (LOL)

SURF: FEATURE DESCRIPTION

Uses wavelet responses for orientation and description

- In turn, these calculations are sped-up by the use of the integral image.

FEATURE FROM ACCELERATED SEGMENT TEST (FAST)

High-speed corner detector is even faster than SURF.

Imagine the corner of a building against a blue sky.

- From around 9 o'clock to 6 o'clock

    The sky will be blue.

    The center point will be building color (tan, brown, and so on).

FAST

Uses a trivial first test to rule out the big arc.

- Checks for 12 o'clock and 6 o'clock, then 3 o'clock and 9 o'clock, for sameness.
- Finds corners
  - But there is no real descriptor built in to FAST.
- FAST is not good with high-level noise.
  - Identifies points where much of an arc around it is all the same color (and different from the center).

BINARY ROBUST INDEPENDENT ELEMENTARY FEATURES (BRIEF)

When we are looking at many features, feature description takes up too much memory and time.

- The descriptors can be too verbose.
- They are often hard to learn from, hard to match, and have inefficient space use.
- There are various means to compress this data: PCA, hashing, and so on.
- Some methods will apply hashing after feature description.
  - This still requires too much memory
- BRIEF is designed to solve this problem.

BRIEF

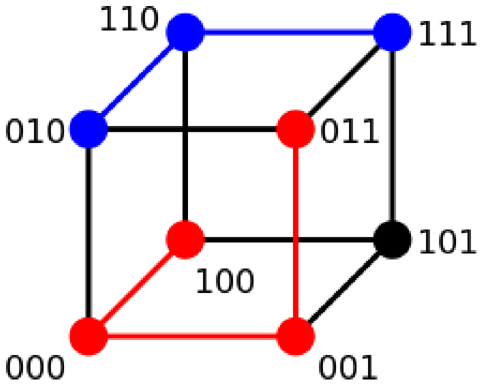
BRIEF uses hashing to convert vectors to binary strings.

- The strings are then often compared with Hamming distance: the number of characters that are different for two strings.
  - Fast and efficient process as it is binary (off/on)

BRIEF goes directly from pixels to binary numbers.

- Fixed set of pairs of points (p_i, p_j)
  - Selected once and permanently
  - Sampled from a random distribution
- The binary values

$$p_i > p_j = 1$$
$$p_i \leq p_j = 0$$



Example Hamming distances:
- 100→011: distance 3
- 010→111: distance 2

Source: Wikipedia

Image source: https://upload.wikimedia.org/wikipedia/commons/6/6e/Hamming_distance_3_bit_binary_example.svg

ORIENTED FAST AND ROTATED BRIEF (ORB)

Developed by OpenCV labs and is not patented like SIFT and SURF

Enhances FAST keypoint detectors

- Adds orientation

Enhances BRIEF descriptors

- Integrates orientation information into BRIEF

ORIENTED FAST AND ROTATED BRIEF (ORB)

Uses FAST to find keypoints.

Applies Harris corner measure to pick keypoints.

- Finds top N points among the keypoints
- Uses a pyramid to produce multiscale features

Builds BRIEF descriptors by incorporating keypoint orientation.

FEATURE MATCHING

FEATURE MATCHING: MATCHING STRATEGY

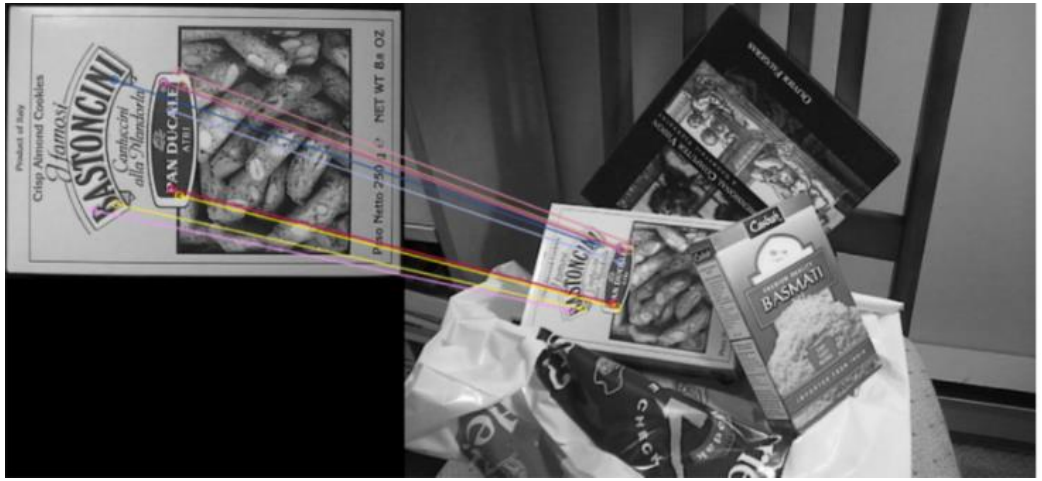
We have seen many techniques for feature detection and description.

How do we match them?

First, we need to pick a matching strategy.

Brute force

- Compare all features against each other
- Quadratic time



FEATURE MATCHING: MATCHING STRATEGY

Use a maximal distance threshold with a Euclidean distance metric.

- Simplest procedure
- Thresholds can be difficult to set
- Need to optimize against false positives/negatives

Use nearest neighbor in feature space.

- Nearest neighbor distance ratio

FEATURE MATCHING: INDEXING STRUCTURE

Efficient strategies need an indexing structure.

Multidimensional hashing

- New features are hashed into buckets
- Buckets are matched between images
- Return buckets like *me* (family address, check members of family)

Locally sensitive hashing

- Indexes features to unions of independently computed hashing functions

FEATURE MATCHING: INDEXING STRUCTURE

Parameter-sensitive hashing

- More sensitive to distribution points

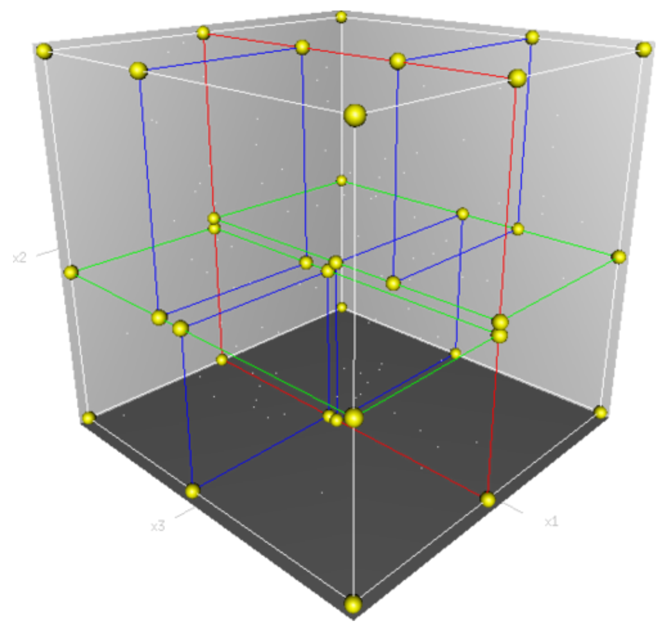
Mulitdimensional search trees

- kd-trees
  - Divides features into alternating axis-aligned hyperplanes

Slicing

- Series of 1D binary searches

Metric tree



Example KD tree in three dimensions
Source: Wikipedia

Image source: https://upload.wikimedia.org/wikipedia/commons/b/b6/3dtree.png

EVALUATION OF ALGORITHMS: ACCURACY

Determining the accuracy of our feature-matching algorithm:

Need to evaluate its accuracy

- Percentage of correct prediction of all predictions
- 95 percent accuracy is a good job

Example

- With 54 percent Democrats and 46 percent Republicans
  - Predict party using votes
  - How many times did I get it right?

EVALUATION OF ALGORITHMS: ACCURACY

Predicting low probability events

Example:

- 1 percent have leukemia, 99 percent are healthy
- Predict leukemia using health records and tests
- Imagine stupidest possible answer (Healthy = Always)

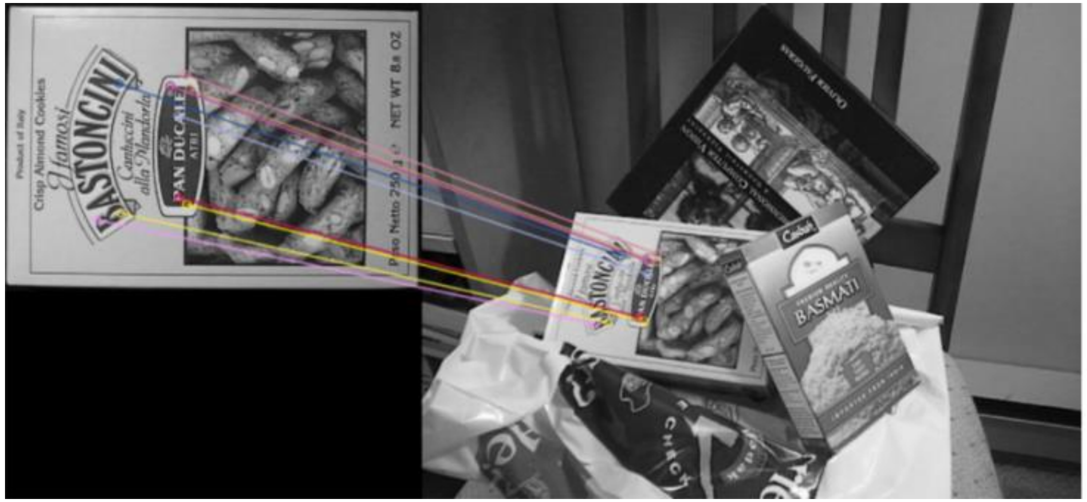
  99 percent accuracy

  You will be correct 99 percent of the time, but you will NOT catch sick people

  USELESS!

EVALUATION USING A CONFUSION MATRIX

	Bastoncini Box (Predicted)	Not - Bastoncini Box (Predicted)	Row-Wise Accuracy (Correctness)
Bastoncini Box (Actual)	27	6	81.81
Not - Bastoncini Box (Actual)	10	57	85.07
Overall Accuracy			83.44



CONFUSION MATRIX

Predicted
(our claim, diagnosis, our result)

Actual
(real-world, actual
sickness, relevance)

	Positive	Negative
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

SENSITIVITY AND SPECIFICITY

Predicted

Actual

	P	N		
P	TP	FN	TP/ (TP+FN)	Sensitivity
N	FP	TN	TN/ (TN+FP)	Specificity

PRECISION AND RECALL

	Predicted		
	P	N	
P	TP	FN	TP/ (TP+FN)
N	FP	TN	
	TP/ (TP+FP)		

Actual

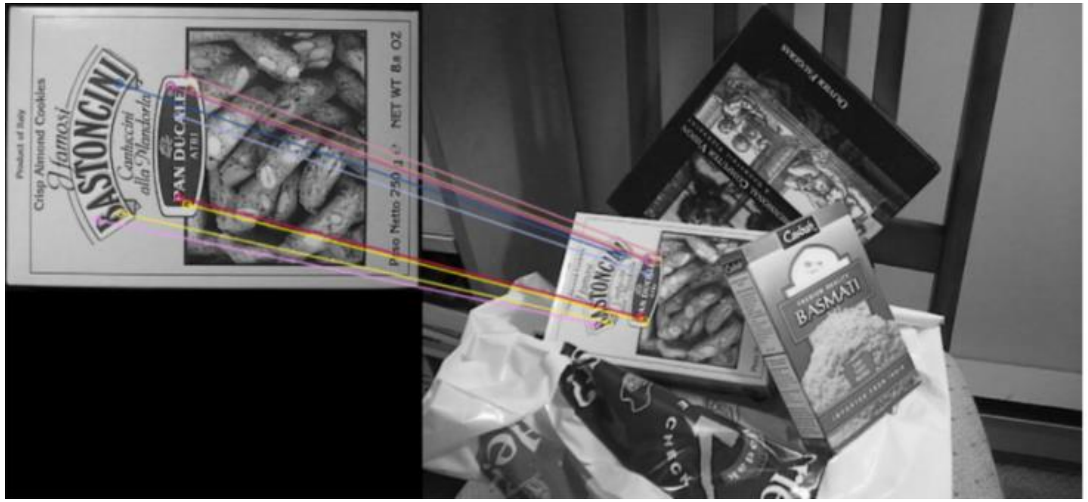
Precision

Recall

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

EVALUATION USING A CONFUSION MATRIX

	Bastoncini Box (Predicted)	Not - Bastoncini Box (Predicted)	Accuracy
Bastoncini Box (Actual)	27	6	81.81
Not - Bastoncini Box (Actual)	10	57	85.07
Overall Accuracy			83.44



$$\textit{Precision} = 27/37 = 73.0\%$$

$$\textit{Recall} = 27/33 = 81.8\%$$

HOW TO TUNE PRECISION AND RECALL

Google.

Forget Recall. There are millions of relevant search results. Catching all of them is not important: Most people only look at the first page of results.

Precision is way more important. If one of those ten links in the first page is <u>not relevant</u> to the search, you'll lose customers.

HOW TO TUNE PRECISION AND RECALL

HIV Tests.

Recall is very, very important. We want to catch all sick people. Imagine an HIV+ person taking the test and being told they are HIV-. Worst outcome. Nope.

Precision is also important. Telling someone that they are HIV+ when they are not is a devastating mistake. Not as bad as letting HIV+ slip through, but we still cannot turn the knob all the way to recall without a care about this, either.

HOW TO TUNE PRECISION AND RECALL

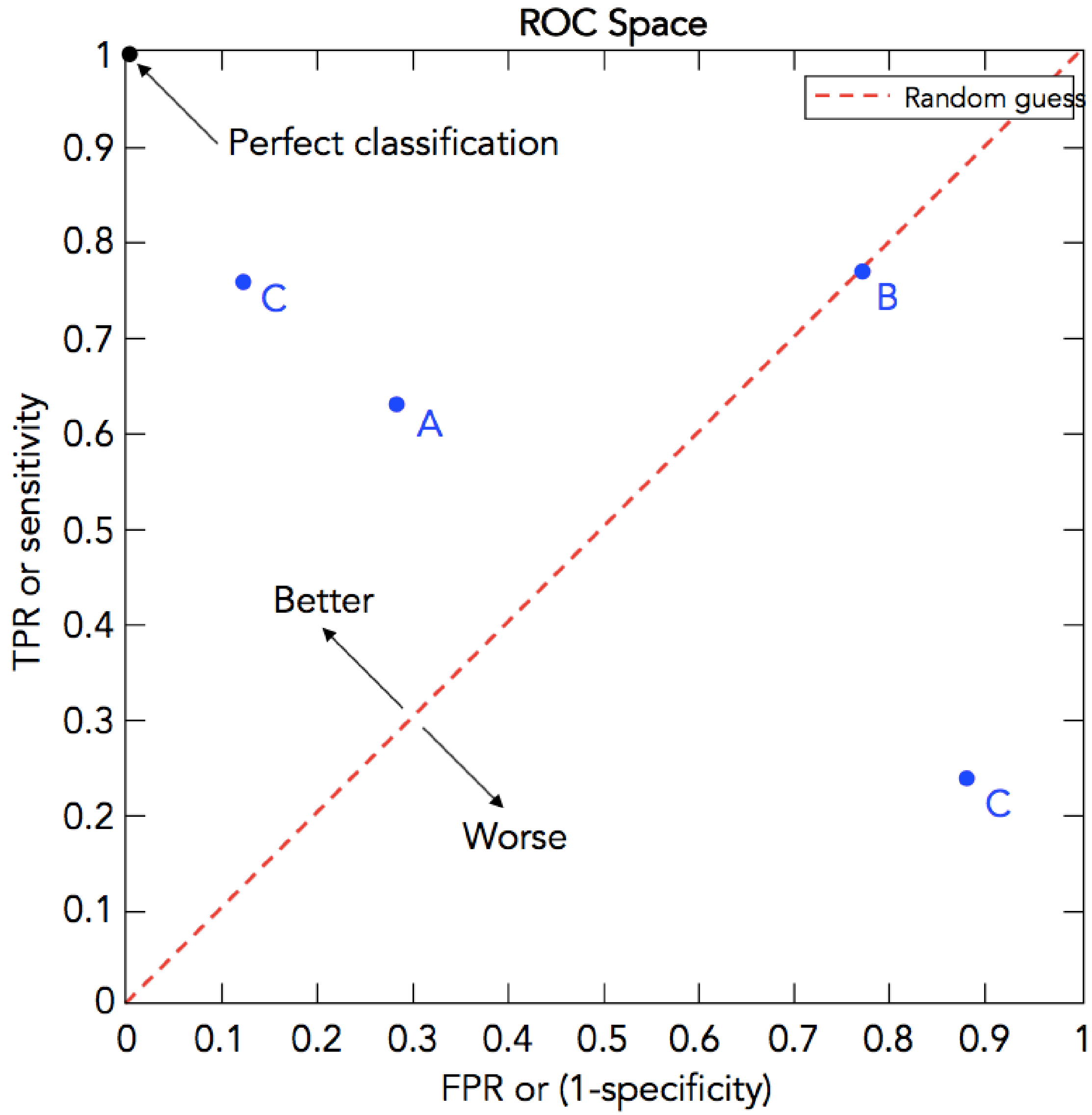
United States Legal System.

Higher Precision at the cost of Recall. Burden of proof. Reasonable doubt = Acquittal. These mean that the *threshold* for classifying someone with the *criminal* label is set very high.

High precision means making sure everyone sent to jail is indeed guilty.

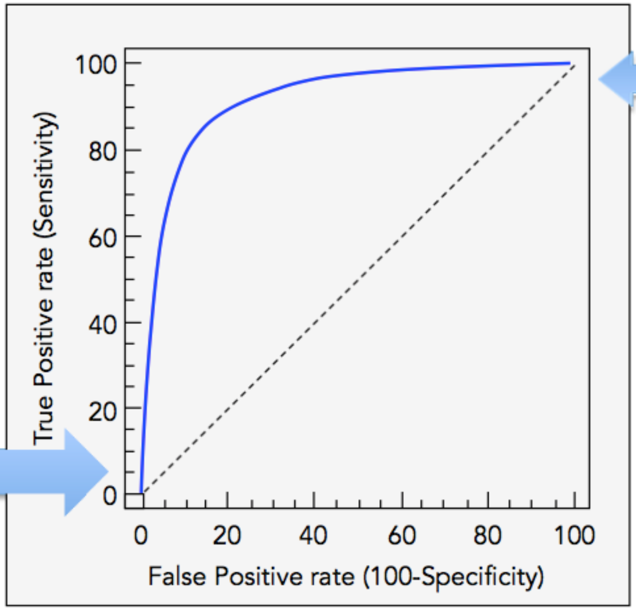
High recall means making sure all criminals go to jail. The legal system is ready to pay the cost of letting some criminals go to ensure no innocents go to jail.

RECEIVER OPERATING CHARACTERISTIC



TUNE PRECISION AND RECALL: THRESHOLDS

Lower threshold: Ease criteria for calling it positive.
Can catch more positives by accepting edge cases.
higher recall, lower precision; higher true positive rate, higher false positive rate.



Higher threshold: Only call it positive if absolutely sure.
To make most positive calls right, give up on
lower recall, higher precision; lower true positive rate, lower false positive rate.

AREA UNDER CURVE

An evaluation of a classification algorithm

- including all possible thresholds