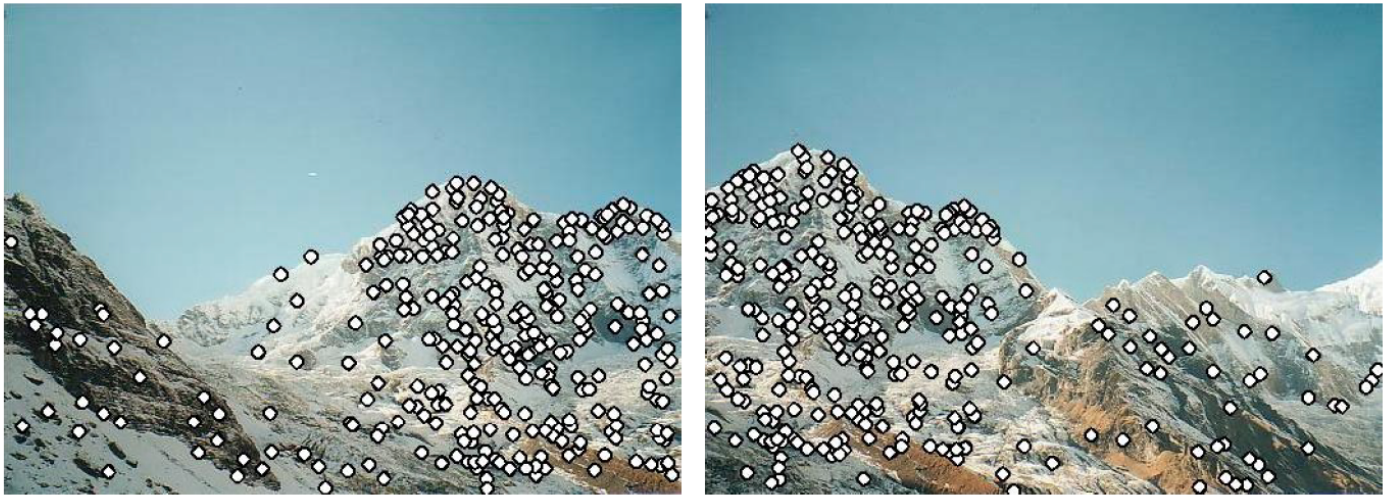


COMP9517: Computer Vision

Feature Representation

Outline

- What and why of feature representation

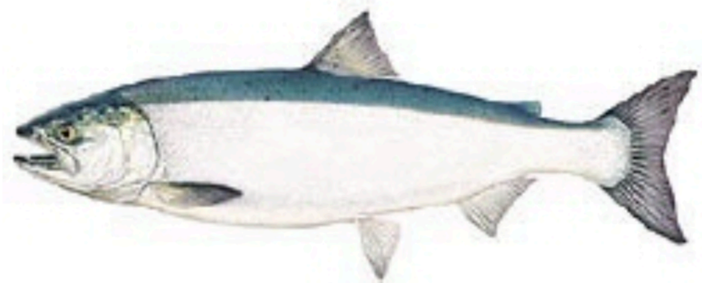
- How of feature representation
  - Different feature extractors/descriptors
    - Classical approaches
    - Representation learning

  - Application cases in various computer vision applications

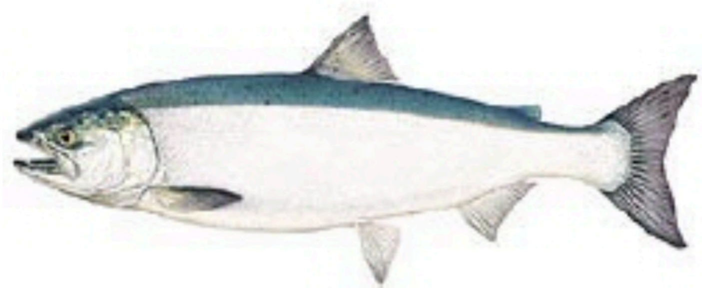
Why and What of Feature Representation

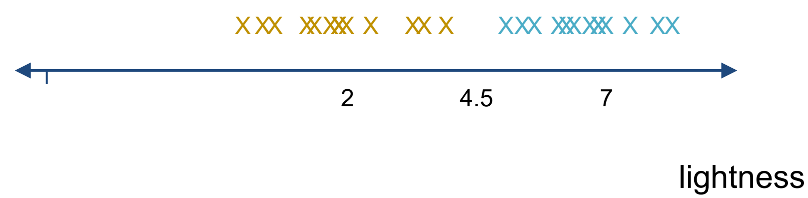
sea bass and salmon

Classification?

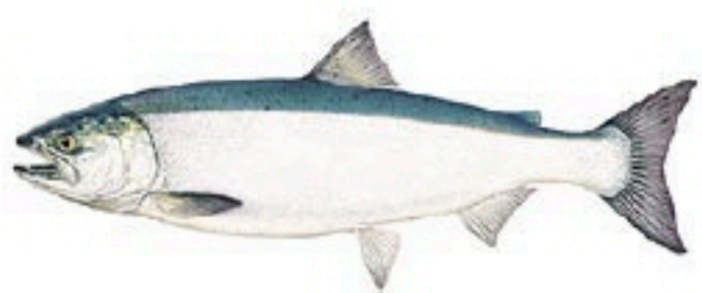


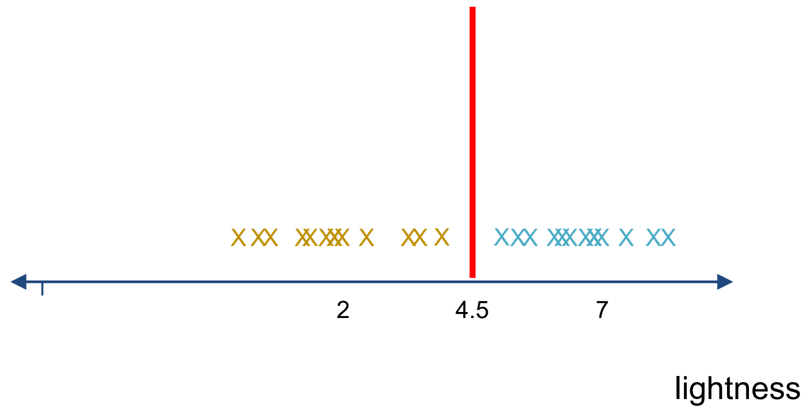
sea bass and salmon

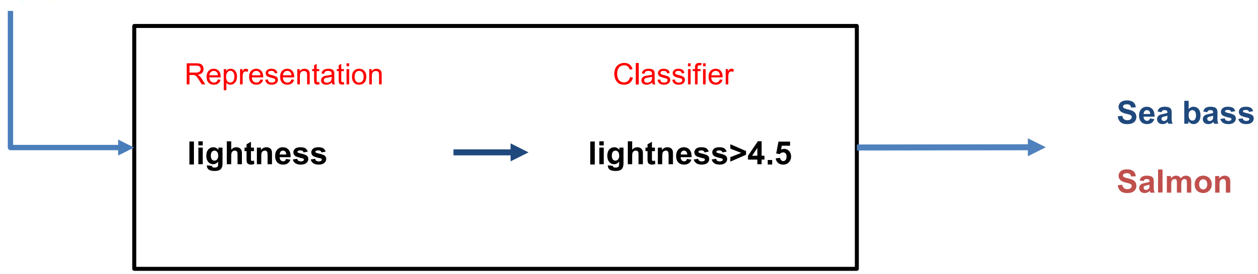




sea bass and salmon







Feature representation for a cat detector?
-- not always easy

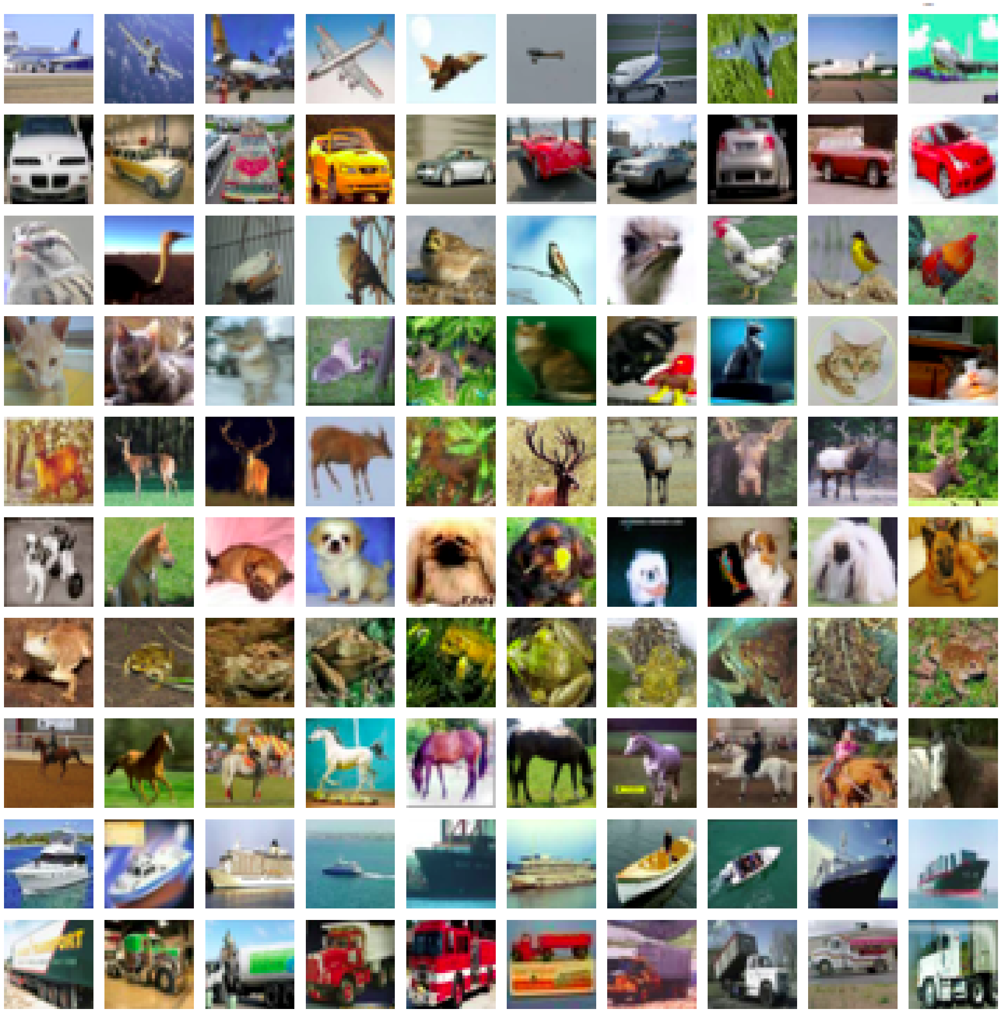
Image Features

- Image features are essentially **vectors** that are a **compact representation** of images
- They represent important information shown in an image
- Intuitive examples of image features:
  - Colour/brightness
  - Edges
  - Corners
  - Lines
  - Shape
  - Texture
  - etc…

Image Features

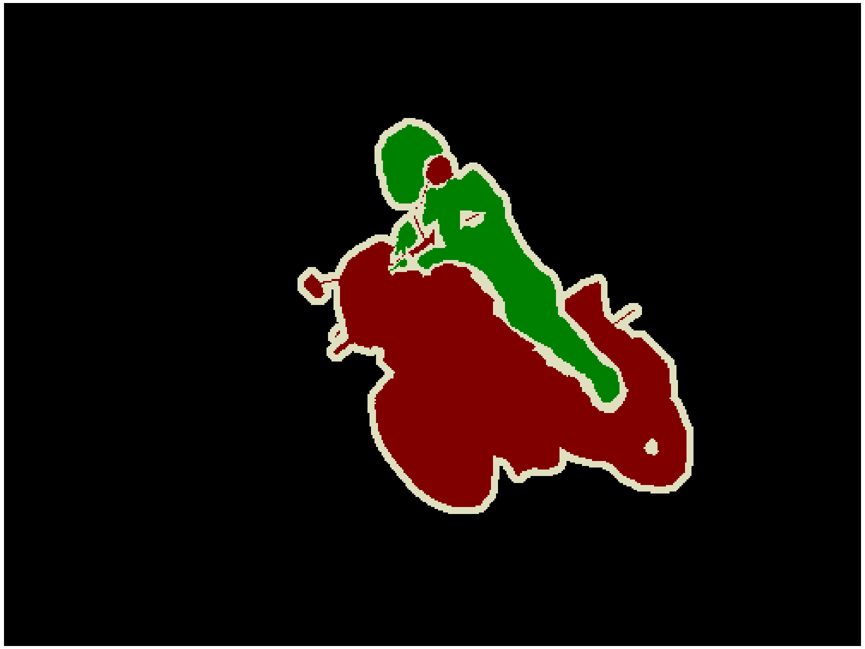
- We need to represent images as feature vectors for further processing in a more efficient and robust way
  - Pixel values -> more informative representations
- Examples of further processing include:
  - Object detection
  - Image segmentation
  - Image classification
  - Content-based image retrieval
  - Image stitching
  - Object tracking

Image Classification



Object Detection

Segmentation



Content-Based Image Retrieval

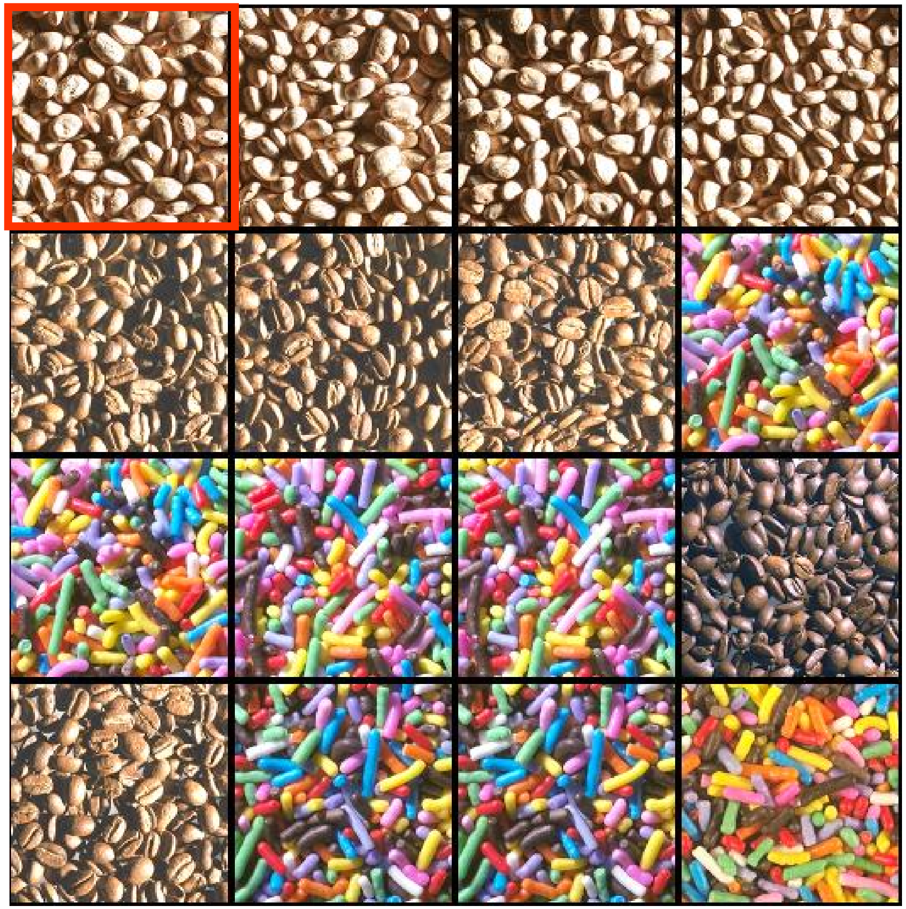
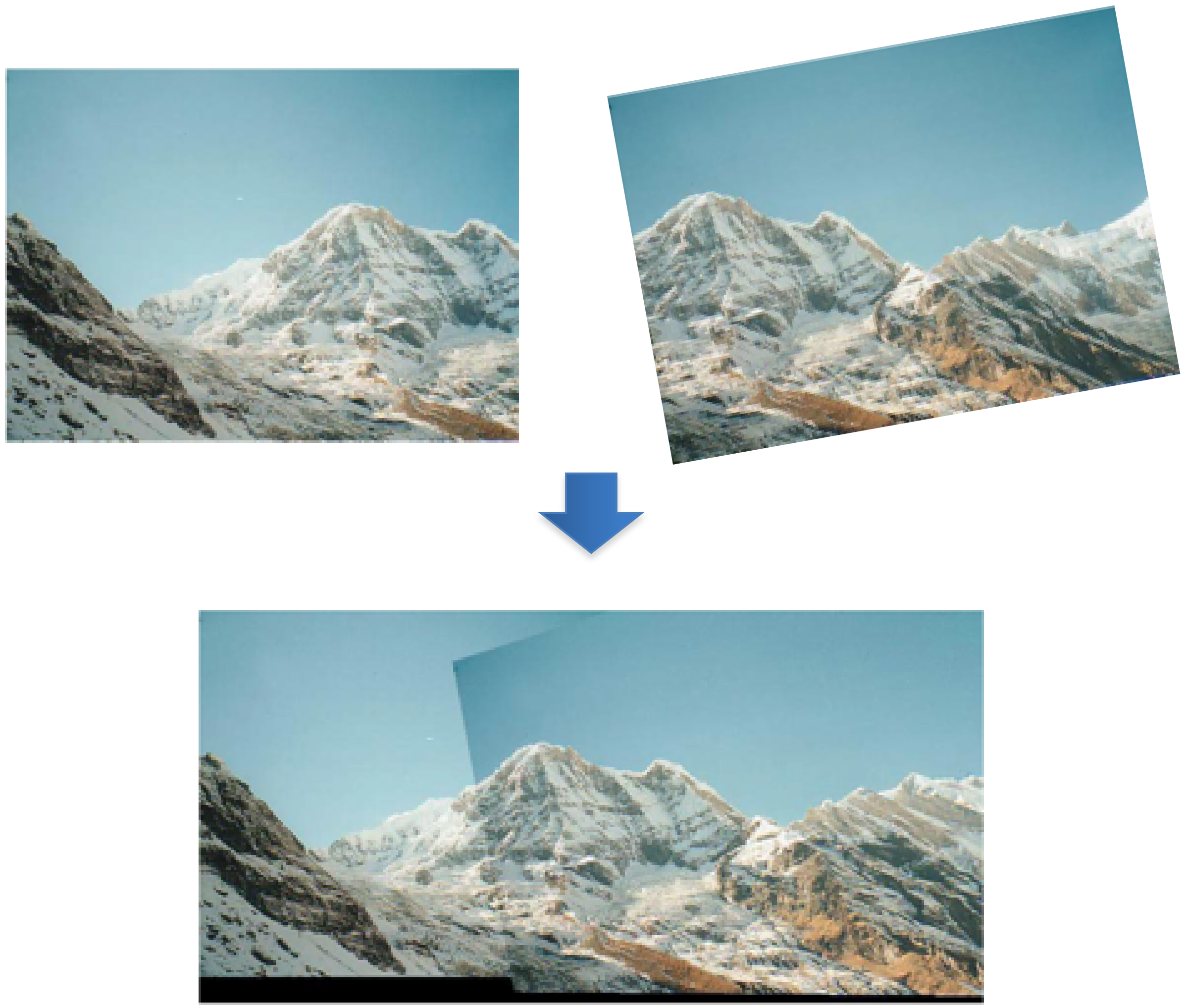
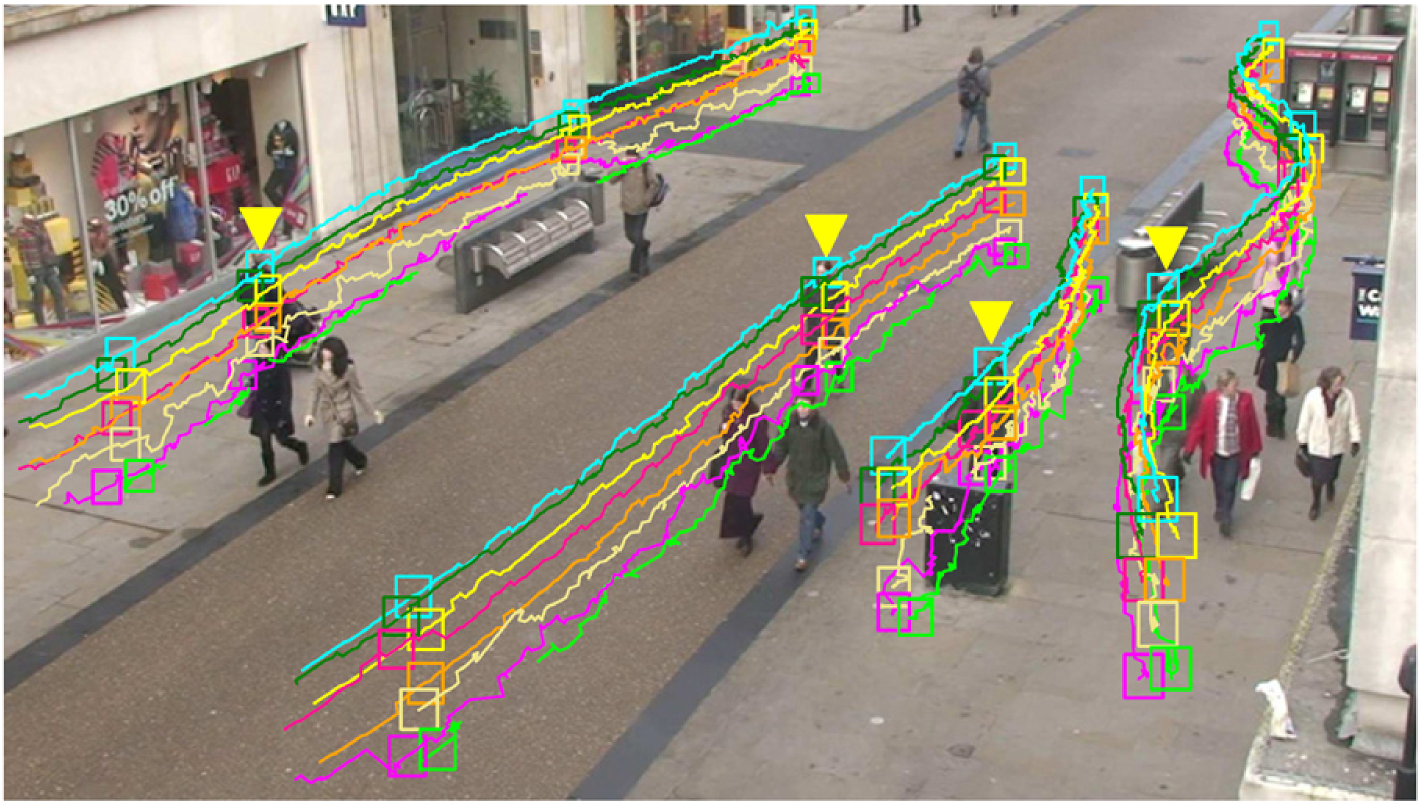


Image Stitching



Object Tracking



https://heartbeat.fritz.ai/

Properties of Features

- Why not just use pixels values directly?
  - Pixel values change with light intensity, colour and direction
  - They also change with camera orientation
  - And they are highly redundant

May be irrelevant to tasks

- Repeatability (robustness)
  - Should be detectable at the same locations in different images despite changes in illumination and viewpoint
- Saliency (descriptiveness)
  - Each feature should have a distinctive and matchable description
- Compactness (efficiency)
  - Fewer features
  - Smaller features

General Framework

Object detection

Image segmentation

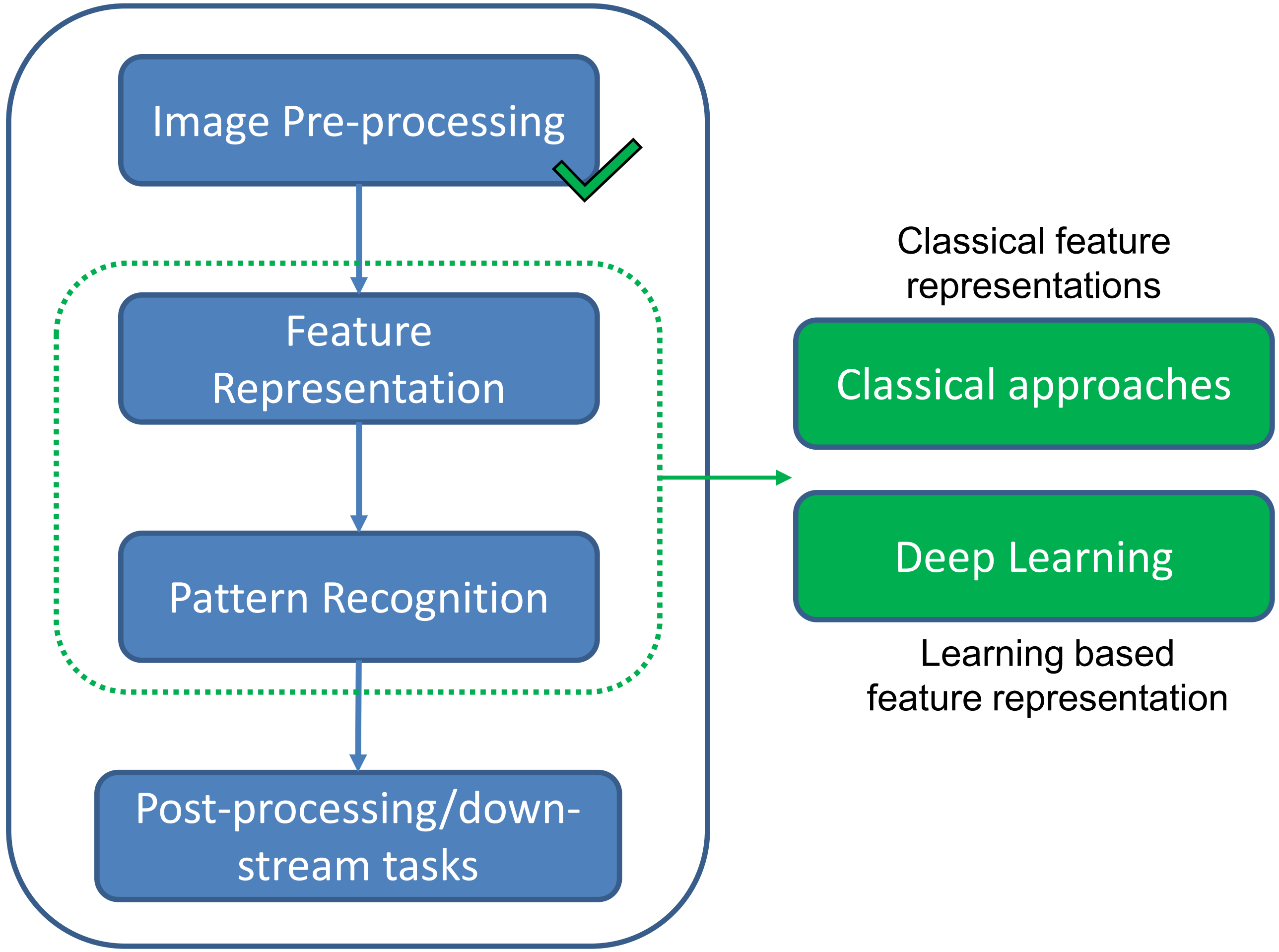
Image classification

Image retrieval

Image stitching

Object tracking

…



How of Feature Representation

- Colour features (based on pixel value)
  - Colour histogram
  - Colour moments
- Feature Descriptors (based on pixel gradients/textures)
  - Haralick texture features
  - Local binary patterns (LBP)
  - Scale-invariant feature transform (SIFT)
  - Bag-of-words (BoW)
  - Histogram of oriented gradients (HOG)
  - Shape descriptors
- Learning based feature representation
  - Unsupervised representation learning
  - Supervised representation learning

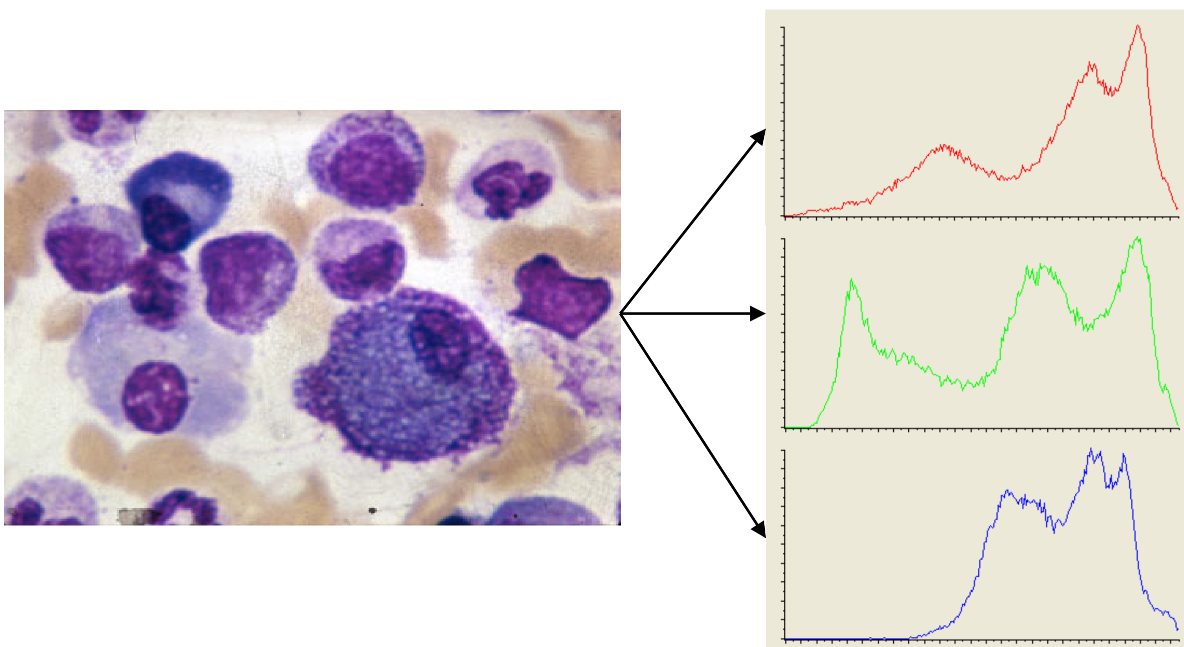
Colour Features

- **<u>Colour</u>** is the simplest feature to compute, and is **invariant** to image scaling, translation and rotation
- Color-sensitive tasks
- Example: colour-based image retrieval

http://labs.tineye.com/multicolr/

Colour Histogram

- Represent the global distribution of pixel colours in an image
  - Step 1: Construct a histogram for each colour channel (R, G, B)
  - Step 2: Concatenate the histograms (vectors) of all channels as the final feature vector



Histogram of **R** channel

Histogram of **G** channel

Histogram of **B** channel

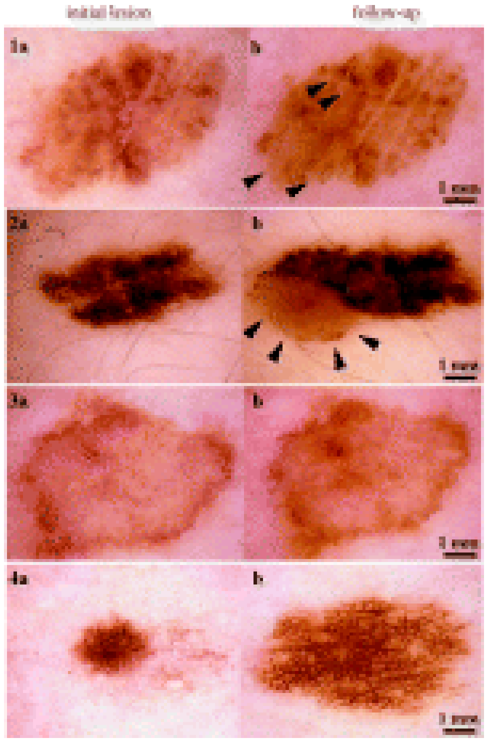
Colour Moments

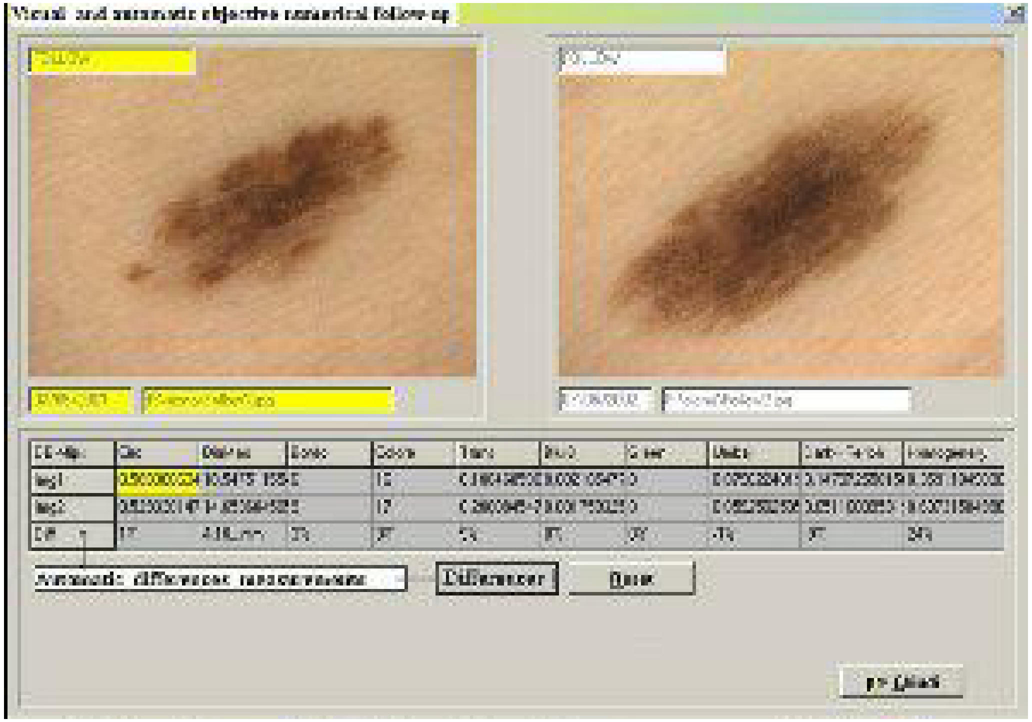
f_{ij} is the value of the i-th colour component of pixel j and N is the number of pixels in the image

- Another way of representing colour distributions
  - Based on statistical moments (summarization of a whole image)
  - First-order moment $\boxed{\mu_i = \frac{1}{N}\sum_{j=1}^{N} f_{ij}}$ (*mean*)
  - Second-order moment $\boxed{\sigma_i = (\frac{1}{N}\sum_{j=1}^{N}(f_{ij} - \mu_i)^2)^{\frac{1}{2}}}$ (*variance*)
  - Third-order moment $\boxed{s_i = (\frac{1}{N}\sum_{j=1}^{N}(f_{ij} - \mu_i)^3)^{\frac{1}{3}}}$ (*skewness*)
- Moments based representation of colour distributions
  - Gives a feature vector of only 9 elements (for RGB images)
  - Lower representation capability than the colour histogram

Application Example

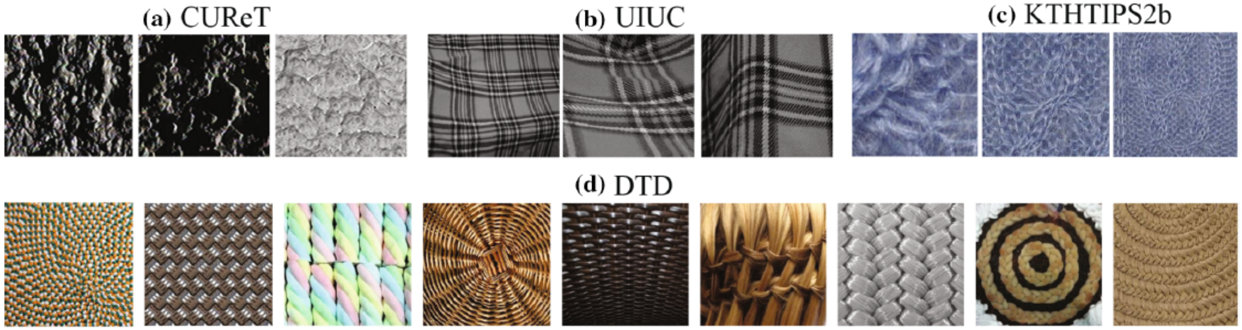
- Colour-based image retrieval





Pixel Gradient-based Features

- Feature descriptor relying on local patterns at "textural" level.
- Local appearance reflected in pixel gradients
- **Texture** is a powerful discriminating feature for identifying **visual patterns** with properties of homogeneity that cannot result from the presence of only a single color or intensity
- Many successful classical feature descriptors



https://arxiv.org/abs/1801.10324

Haralick Features

- Haralick features give an **array of statistical descriptors** of image patterns to capture the **spatial relationship between neighbouring pixels**, that is, textures.
  - Step 1: Construct the **gray-level co-occurrence matrix** (GLCM)
  - Step 2: Compute the Haralick feature descriptors from the GLCM
- Representing the feel, appearance, or consistency of a surface, such as distinguishing rough and smooth surfaces.

https://doi.org/10.1109/TSMC.1973.4309314

610 IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. SMC-3, NO. 6, NOVEMBER 1973

Textural Features for Image Classification

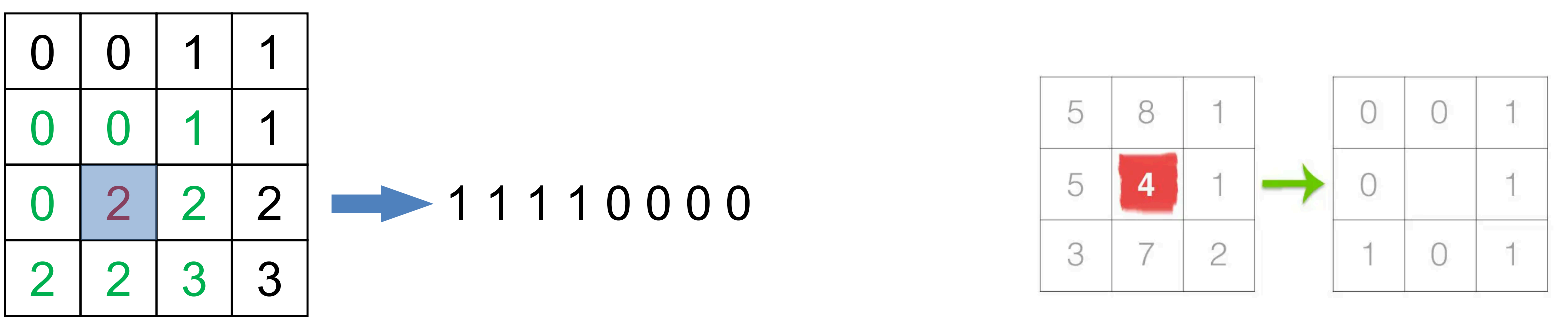
ROBERT M. HARALICK, K. SHANMUGAM, AND ITS'HAK DINSTEIN

***Abstract*—Texture is one of the important characteristics used in identifying objects or regions of interest in an image, whether the image be a photomicrograph, an aerial photograph, or a satellite image. This paper describes some easily computable textural features based on gray-tone spatial dependancies, and illustrates their application in category-**

array. If $L_x = \{1,2,\cdots,N_x\}$ and $L_y = \{1,2,\cdots,N_y\}$ are the X and Y spatial domains, then $L_x \times L_y$ is the set of resolution cells and the digital image I is a function which assigns some gray-tone value $G \in \{1,2,\cdots,N_g\}$ to each and every

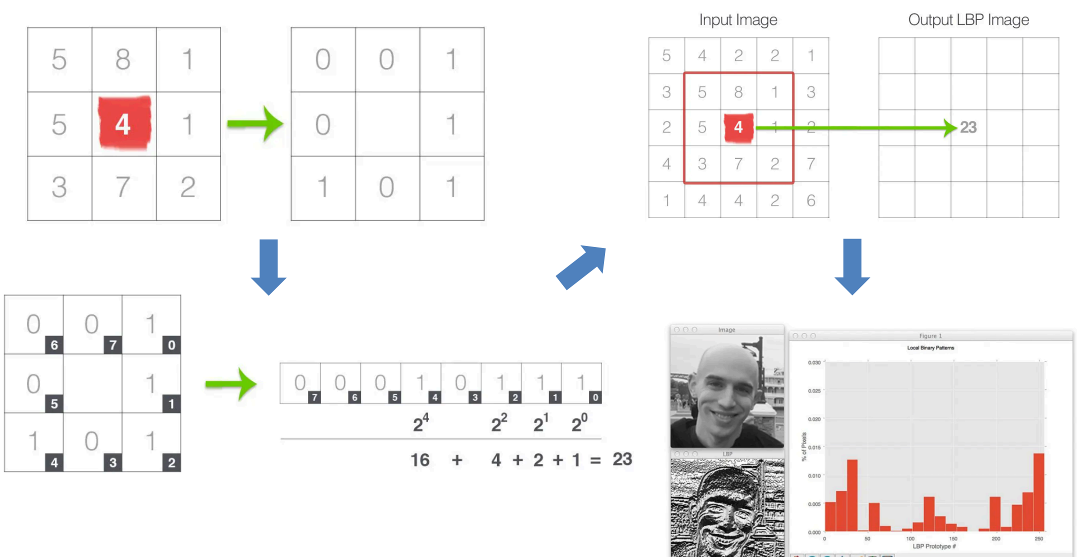
Local Binary Patterns

- Describe the spatial structure of local image texture
  - Divide the image into cells of N x N pixels (e.g. N = 16 or 32)
  - Compare each pixel in a cell to each of its 8 neighbouring pixels: If the centre pixel's value is greater than the neighbour's value, write "0", otherwise write "1"
  - This gives an 8-digit binary pattern per pixel after comparing with all 8 neighbouring pixels, representing a value in the range 0…255



Local Binary Patterns

- Describe the spatial structure of local image texture (cont.)
  - Generate the histogram for all pixels in the cell, computing the frequency of each 8-digit binary number occurring in the cell
  - This gives a 256-bin histogram (the LBP feature vector)
  - Combine the histograms of all cells to obtain the image-level LBP feature descriptor

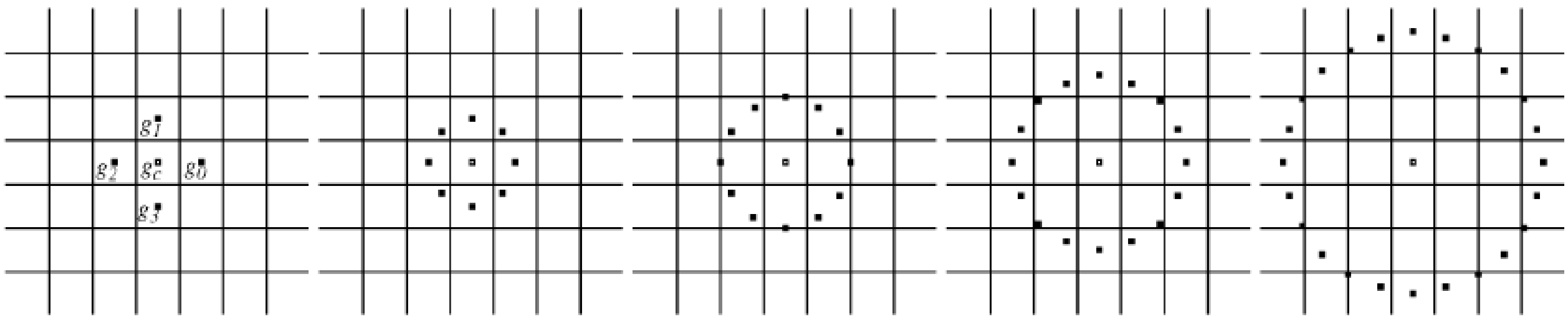


A histogram of 256 elements

Ref: https://pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

Local Binary Patterns

- LBP can be multi-resolution and rotation-invariant
  - Multi-resolution: varying the distance between the centre pixel and neighbouring pixels, and the number of neighbouring pixels



(P=4,R=1.0) (P=8,R=1.0) (P=12,R=1.5) (P=16,R=2.0) (P=24,R=3.0)

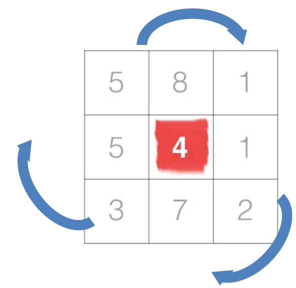
T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):971-987, 2002. https://doi.org/10.1109/TPAMI.2002.1017623

Local Binary Patterns

- LBP can be multi-resolution and rotation-invariant
  - Rotation-invariant: varying the way of constructing the 8-digit binary number, e.g. performing bitwise shift to derive the smallest number

Example:

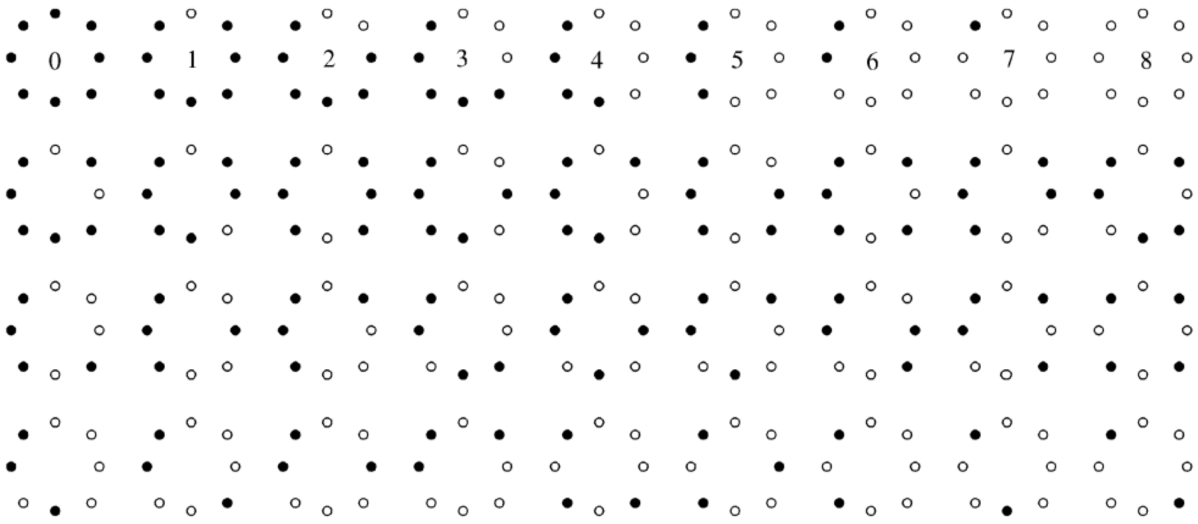
$$\left.\begin{array}{rcr} 1\;1\;1\;1\;0\;0\;0\;0 & = & 240 \\ 1\;1\;1\;0\;0\;0\;0\;1 & = & 225 \\ 1\;1\;0\;0\;0\;0\;1\;1 & = & 195 \\ 1\;0\;0\;0\;0\;1\;1\;1 & = & 135 \\ 0\;0\;0\;0\;1\;1\;1\;1 & = & 15 \\ 0\;0\;0\;1\;1\;1\;1\;0 & = & 30 \\ 0\;0\;1\;1\;1\;1\;0\;0 & = & 60 \\ 0\;1\;1\;1\;1\;0\;0\;0 & = & 120 \end{array}\right\} 15$$



Note: not all patterns have 8 shifted variants (e.g. 11001100 has only 4)

Local Binary Patterns

- LBP can be multi-resolution and rotation-invariant
  - Rotation-invariant: varying the way of constructing the 8-digit binary number, e.g. performing bitwise shift to derive the smallest number => this reduces the LBP feature dimension from 256 to 36

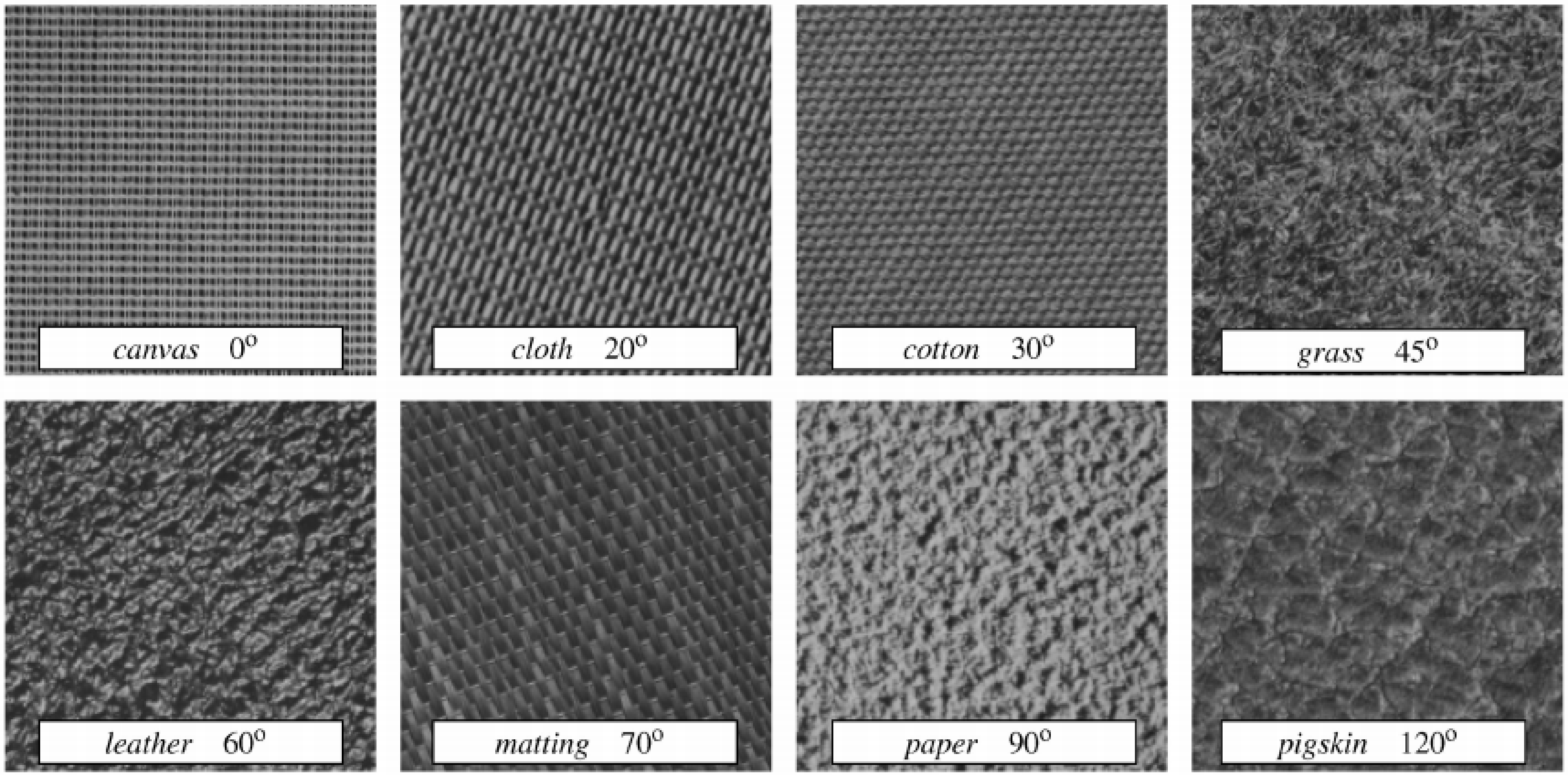


T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):971-987, 2002. https://doi.org/10.1109/TPAMI.2002.1017623

Application Example

- Texture classification

P,R	$LBP_{P,R}$ BINS	$LBP_{P,R}$ RESULT
8,1	10	88.2
16,2	18	98.5
24,3	26	99.1
8,1+16,2	10+18	99.0
8,1+24,3	10+26	99.6
16,2+24,3	18+26	99.0
8,1+16,2+24,3	10+18+26	99.1

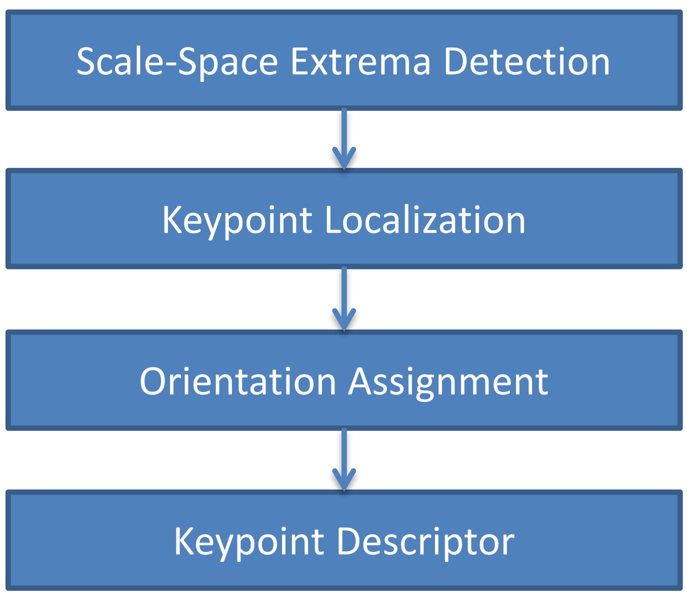


Scale-Invariant Feature Transform

- SIFT feature describes the texture features in a localised region around a **key points**

Distinctive image features from scale-invariant keypoints 70976 2004
DG Lowe
International journal of computer vision 60 (2), 91-110

- SIFT descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes



Find maxima/minima in DoG images across scales

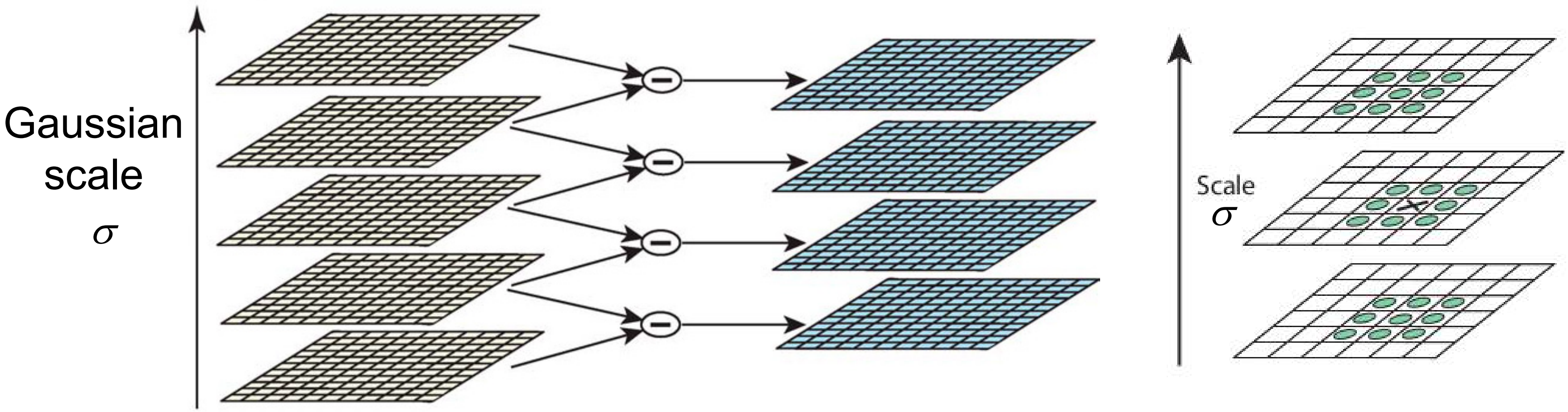
Discarding low-contrast keypoints
Eliminating edge responses

Achieve rotation invariance

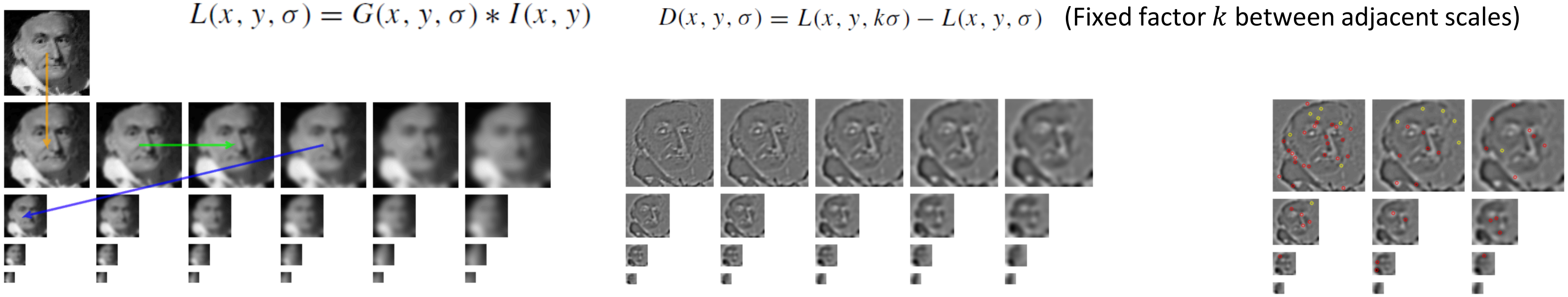
Compute gradient orientation histograms

SIFT Extrema Detection

- Difference of Gaussian (DoG) features at multiple scales
- Detect *maxima* and *minima* in the scale space of the image



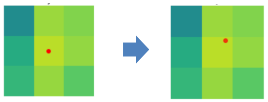
$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$ (Fixed factor k between adjacent scales)



http://weitz.de/sift/

D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis. 60(2):91-110, November 2004. https://doi.org/10.1023/B:VISI.0000029664.99615.94

SIFT Keypoint Localization



- Improve and reduce the set of found keypoints
  - Use 3D quadratic fitting in scale-space to get **subpixel** optima
    - Taylor expansion of the scale-space function up to quadratic term
  - Reject low-contrast and edge points using Hessian analysis

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1}\frac{\partial D}{\partial \mathbf{x}}.$$

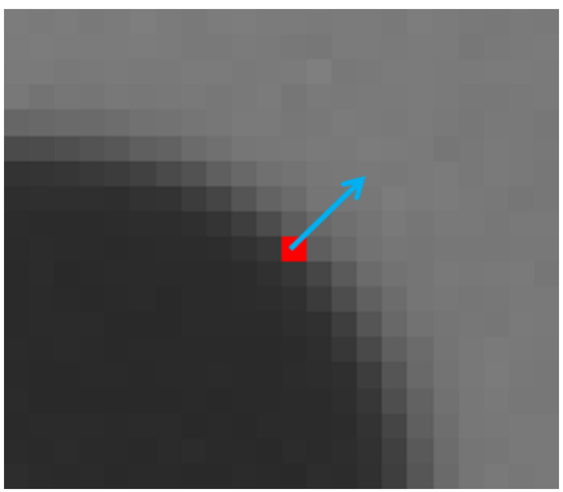
Initial keypoints from scale-space optima

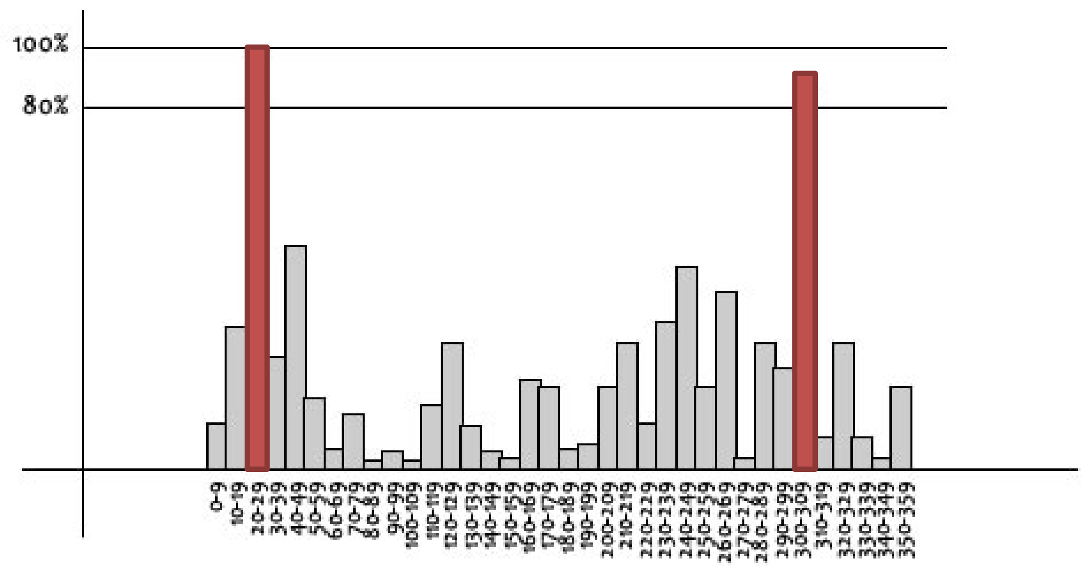
Keypoints after rejecting low-contrast points

Final keypoints after rejecting edge points

SIFT Orientation Assignment

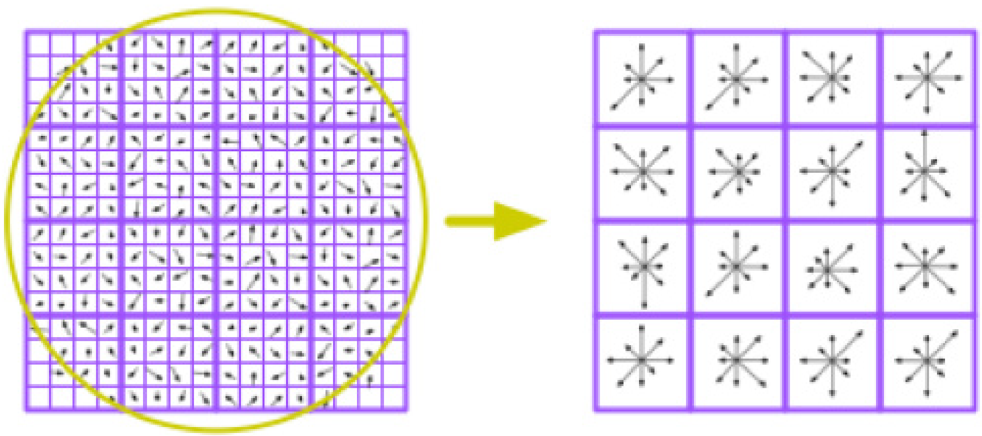
- Estimate keypoint orientation using local gradient vectors
  - For each keypoint, make an orientation histogram of local gradient vectors (pixels)
  - Find the dominant orientation from the main peak of the histogram
  - Create additional keypoint for second highest peak if >80%





SIFT Keypoint Descriptor

- Divide the 16x16 neighbor areas into 4x4 subareas (4x4 subwindow)
- Bin gradients within subwindow, get histogram
  - 8 bins in gradient orientation histogram
  - Rotating coordinates following orientation of keypoint -> orientation independence
  - Some clamping and normalization operations -> illumination change robustness
- Total 8 x 4 x 4 array = 128 dimensions
- Each keypoint represented by a 128D feature vector

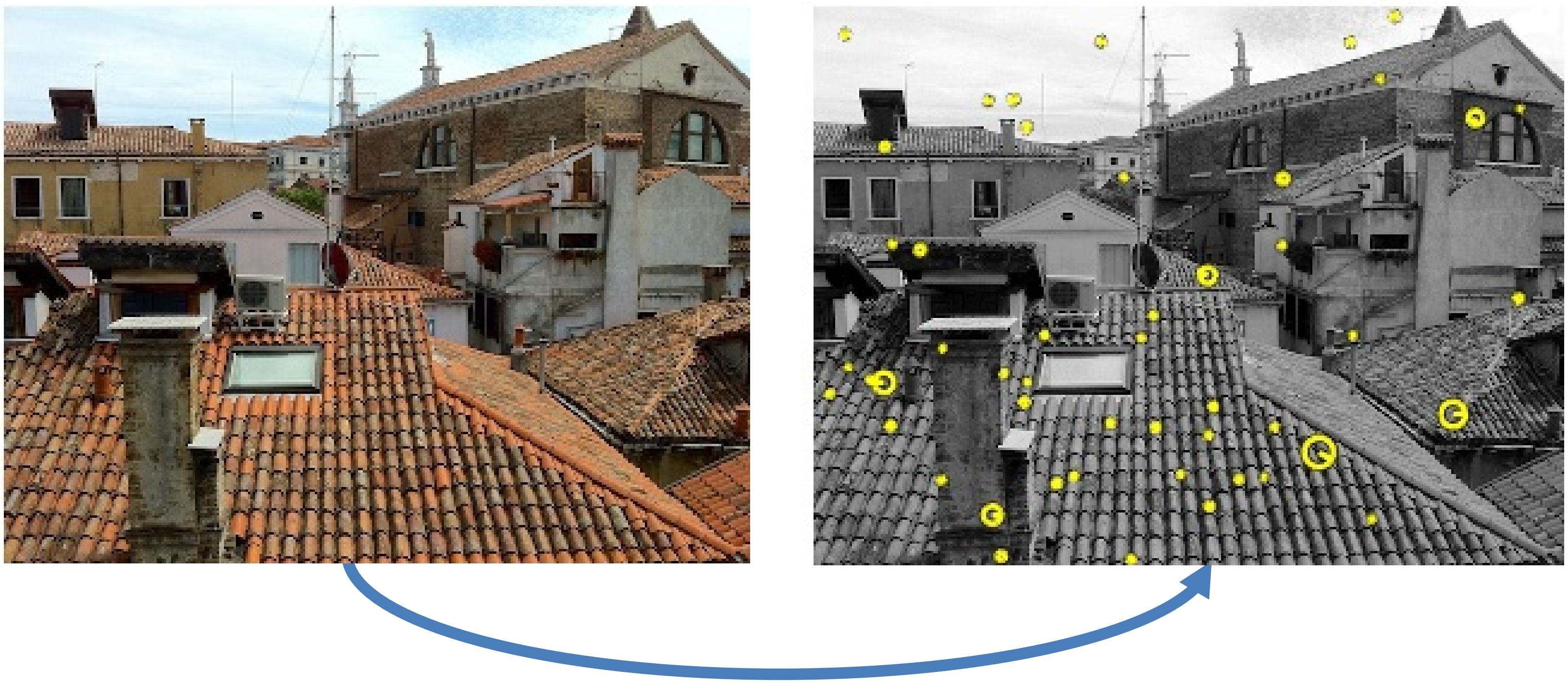


Application Example

- Image matching

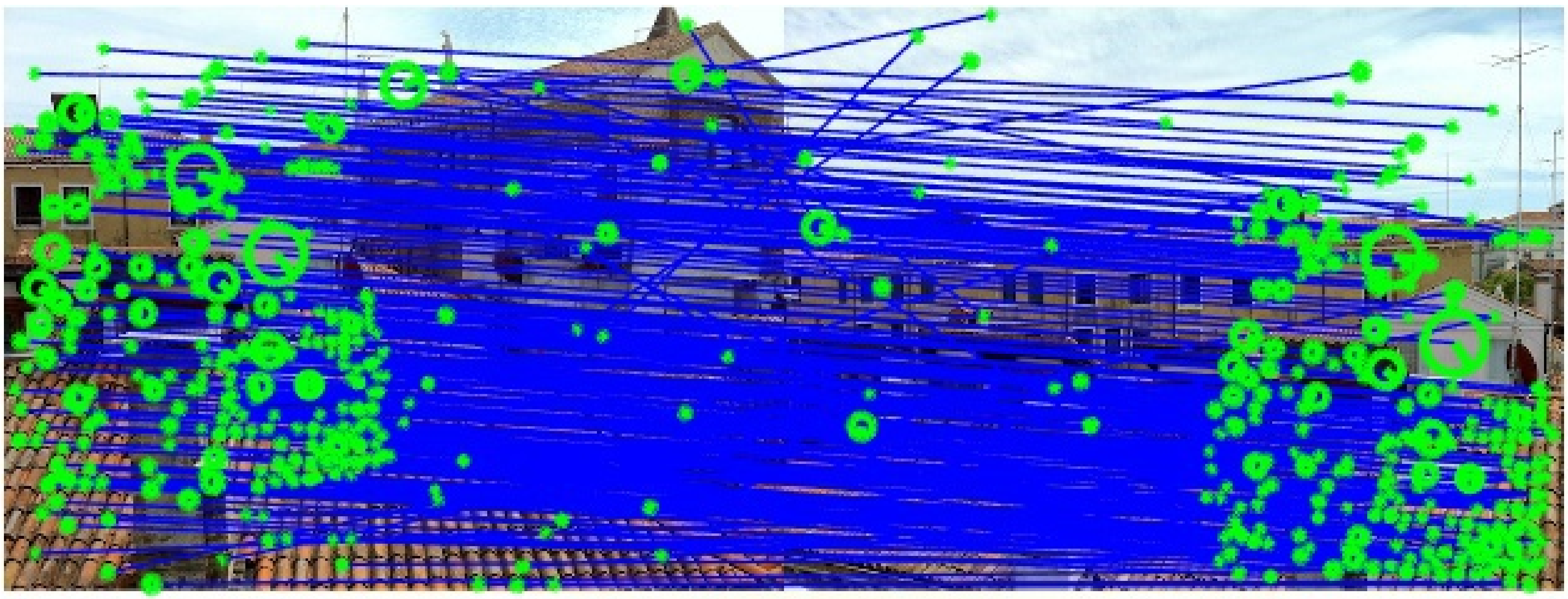
Application Example

- Image matching
  - Compute SIFT keypoints for each image



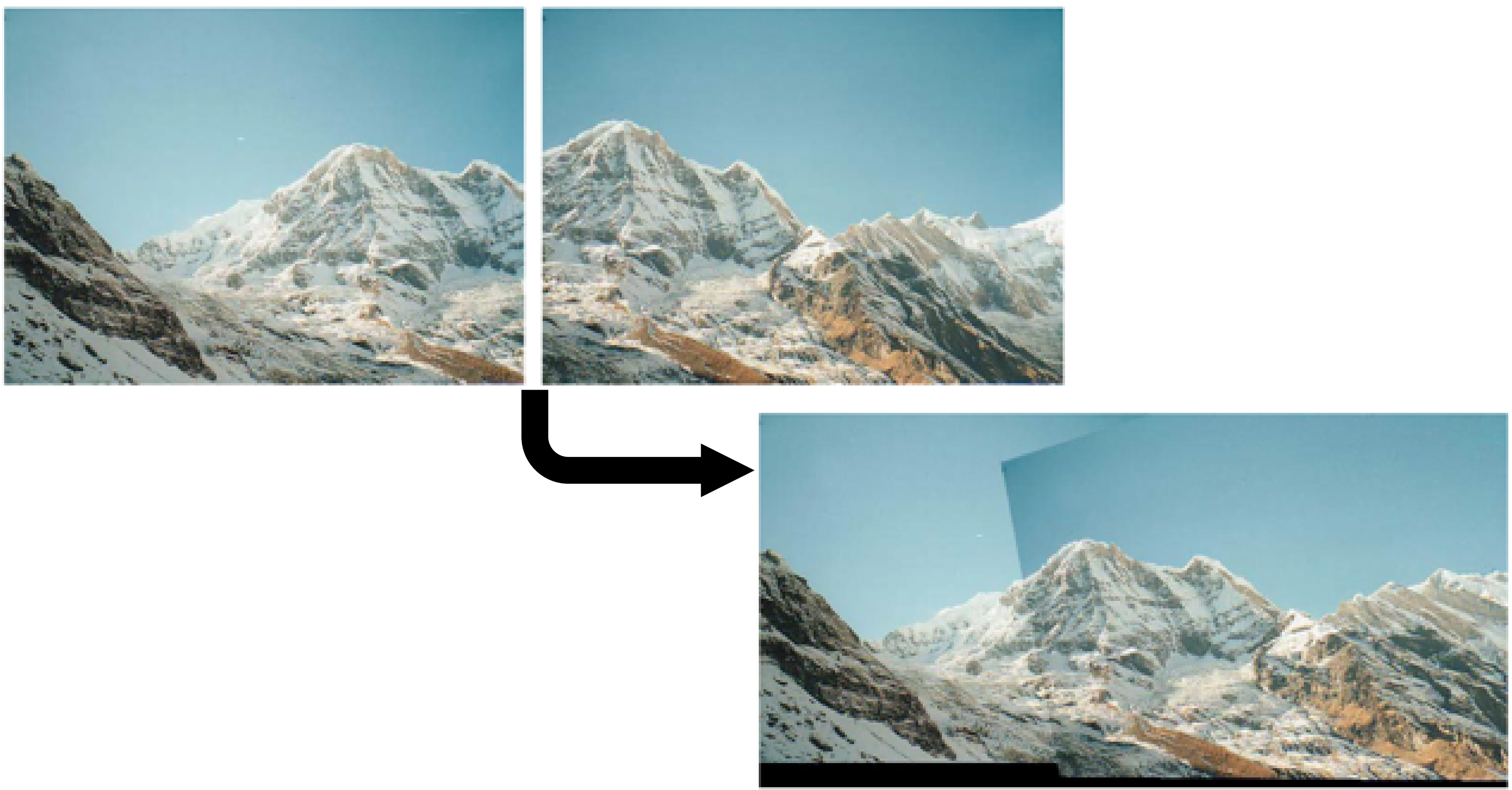
Application Example

- Image matching
  - Find best match between SIFT keypoints in 128D feature space



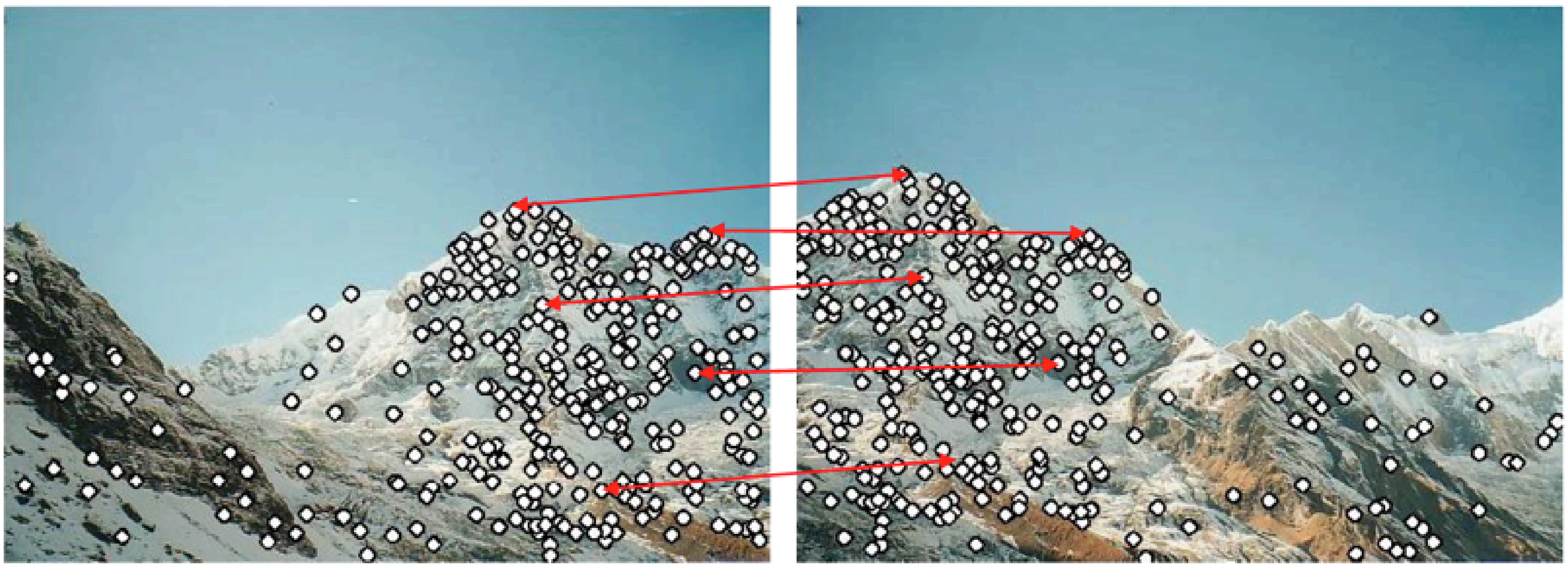
Application Example

- Image stitching



Application Example

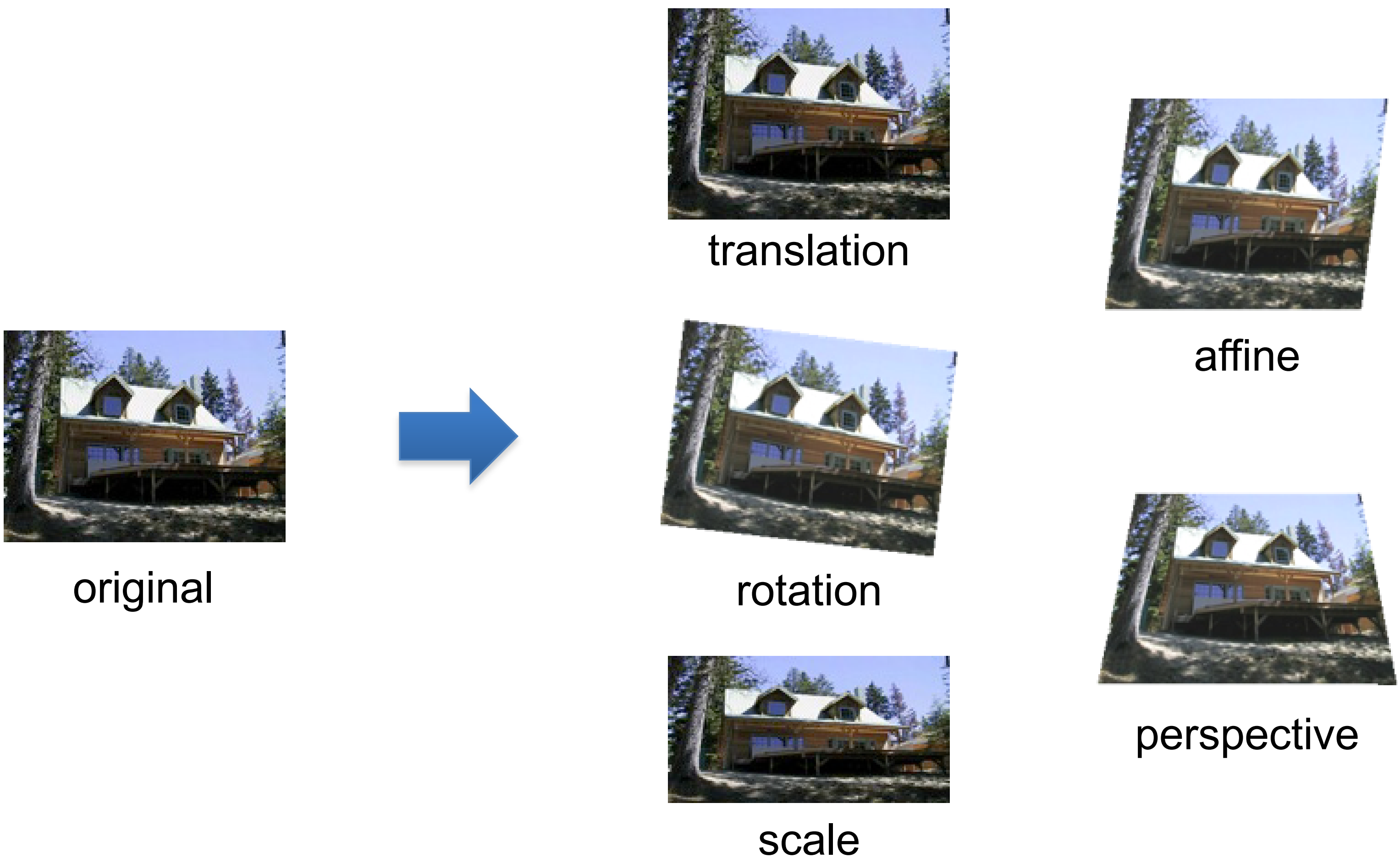
- Image stitching
  - Find SIFT keypoints and feature correspondences



Application Example

- Image stitching
  - Find the right spatial transformation

Transformations



Transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Projective

Fitting and Alignment

- Least-squares (LS) fitting of corresponding keypoints $(\mathbf{x}_i, \mathbf{x}_i^{'})$

$$E_{LS} = \sum_i \left\| \mathbf{r}_i \right\|^2 = \sum_i \left\| f(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}_i^{'} \right\|^2$$

where $\mathbf{p}$ are the parameters of the transformation in f

Example for affine transformation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ d \\ e \\ c \\ f \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ \vdots \end{bmatrix}$$

This calculation is on pixel coordinates.

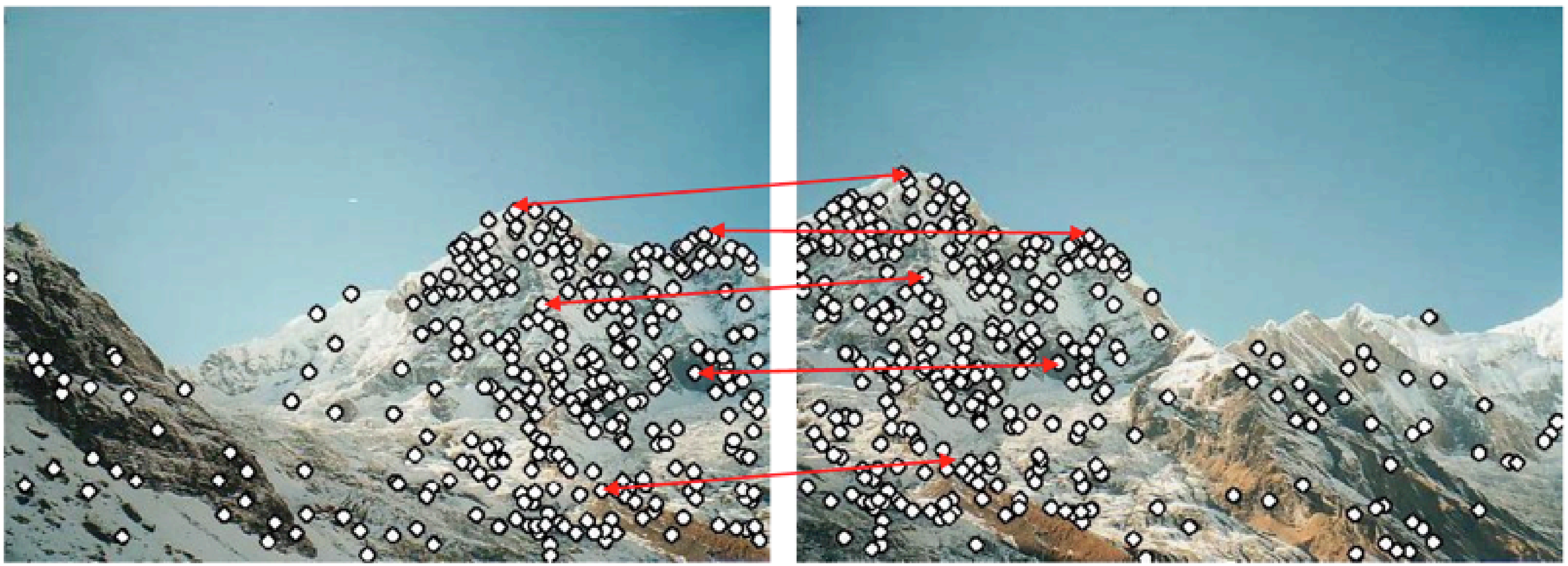
$$\Downarrow$$

$$\mathbf{p} = [\mathbf{A}^{\mathrm{T}} \mathbf{A}]^{-1} \mathbf{A}^{\mathrm{T}} \mathbf{b} \quad \Leftarrow \quad \mathbf{Ap} = \mathbf{b}$$

Fitting and Alignment

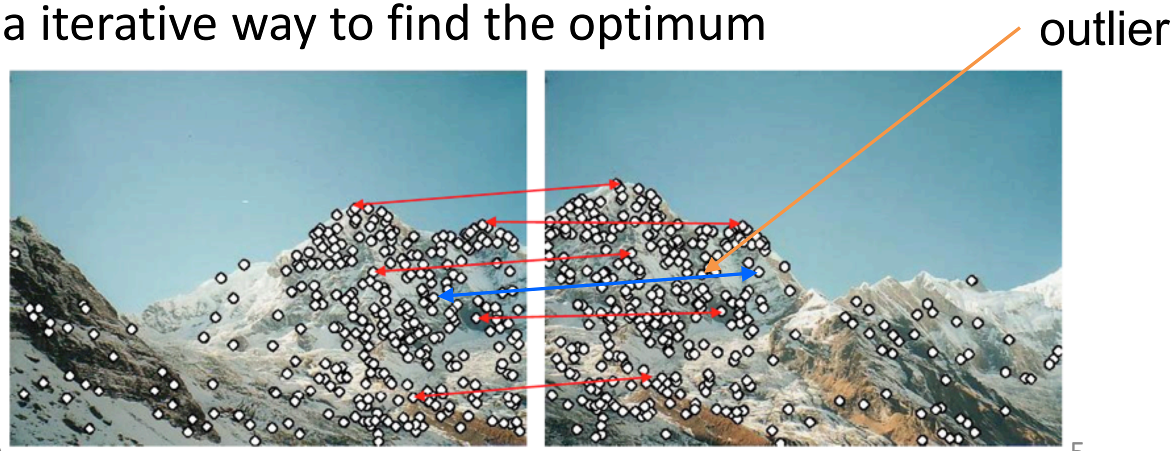
- Solving for the transformation, replying on the correspondence in the overlapping area

$$E_{LS} = \sum_i \|\mathbf{r}_i\|^2 = \sum_i \|f(\mathbf{x}_i;\mathbf{p}) - \mathbf{x}_i'\|^2$$



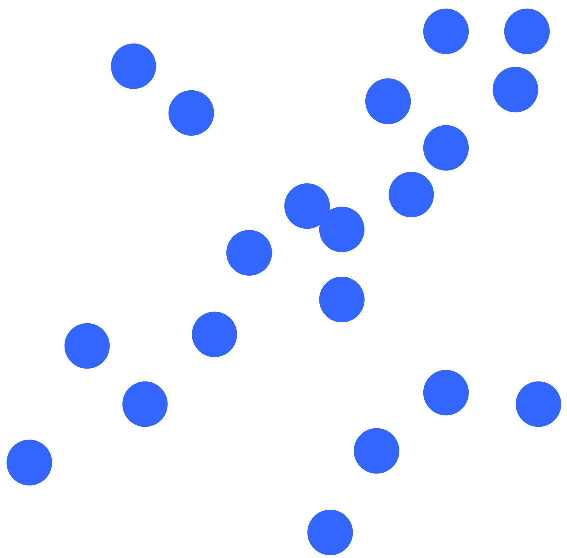
Fitting and Alignment

- Matching results are not always perfect
  - Containing outliers
  - But most (with a rate) of the matching relationship is correct
- RANdom SAmple Consensus (RANSAC) fitting
  - Least-squares fitting is hampered by outliers
  - Some kind of outlier detection and rejection is needed
  - Better use a subset of the data and check inlier agreement
  - RANSAC does this in a iterative way to find the optimum
  - Critical in 3D vision



Fitting and Alignment

- RANSAC



(line fitting example)

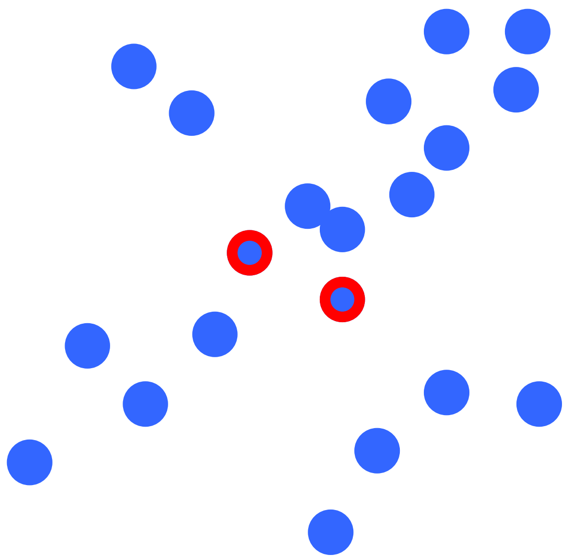
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting and Alignment

- RANSAC



(line fitting example)

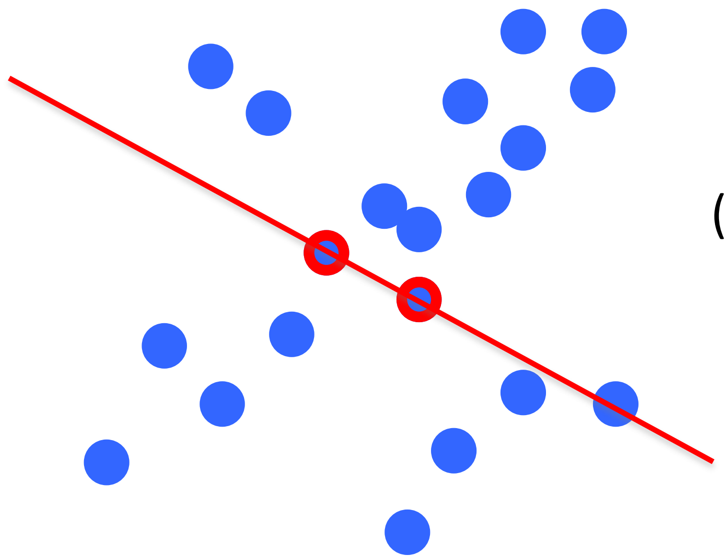
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting and Alignment

- RANSAC



(line fitting example)

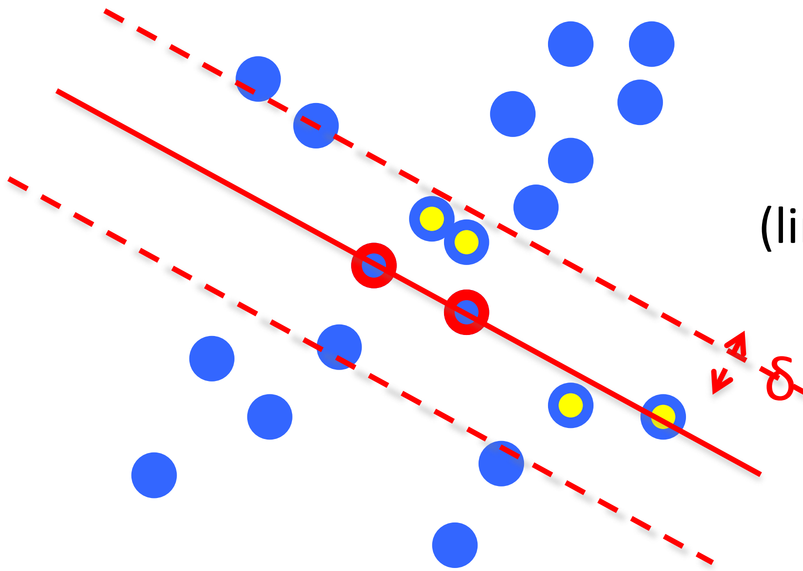
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting and Alignment

- RANSAC



(line fitting example)

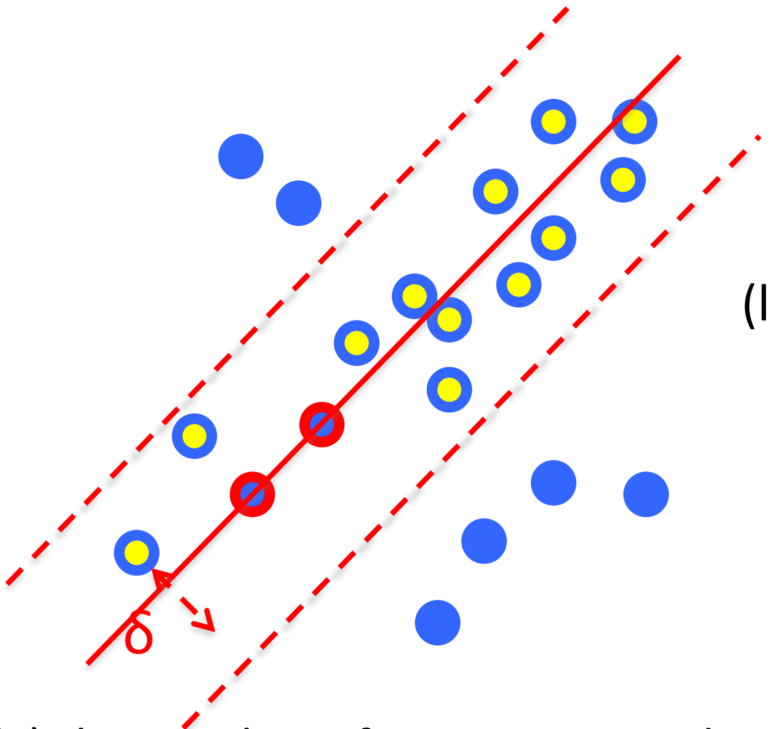
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting and Alignment

- RANSAC



(line fitting example)

Algorithm:

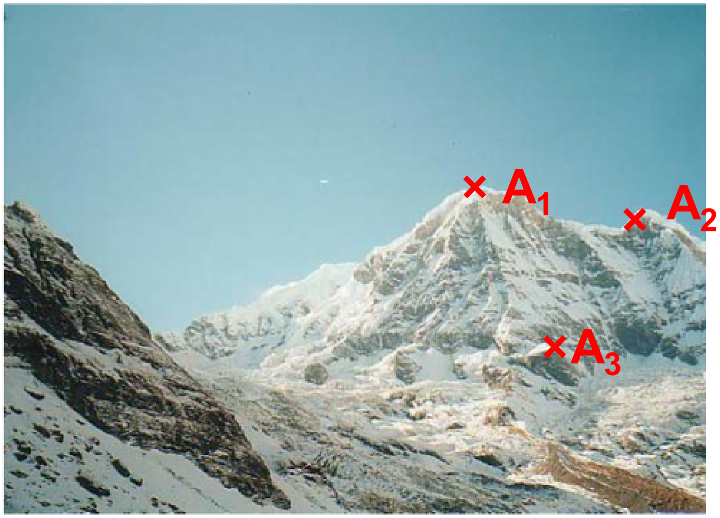
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

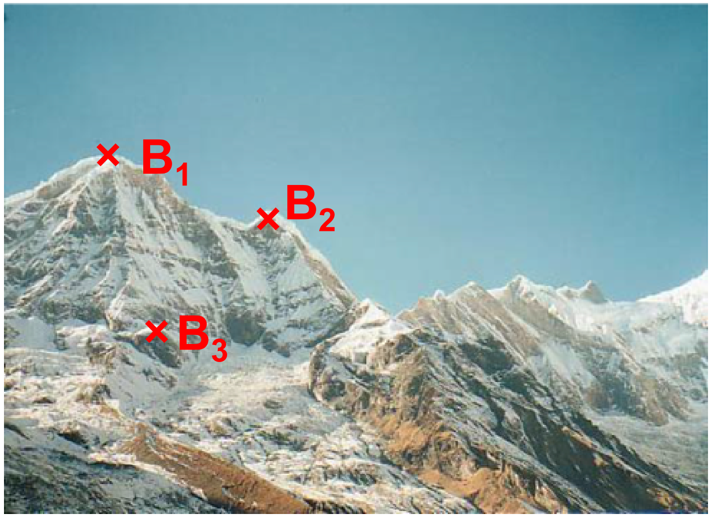
Repeat 1-3 until the best model is found with high confidence

Stereo Matching with RANSAC

1. Write down the objective function
2. Obtain the analytical solution
   a) Compute derivative
   b) Compute solution
3. Obtain computational solution
   a) Write in form **Ap** = **b**
   b) Solve using pseudo-inverse (or another solver)

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

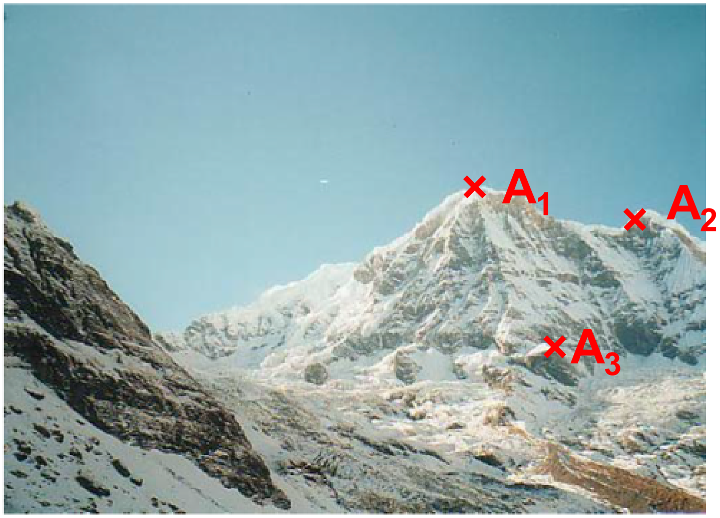


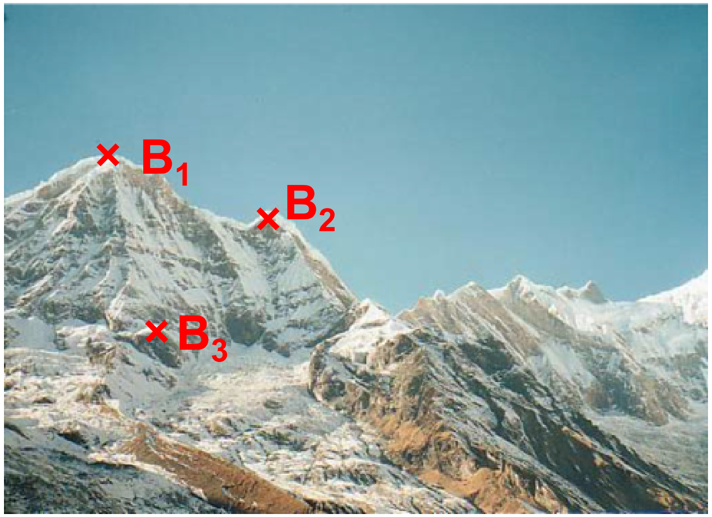


Stereo Matching with RANSAC

1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



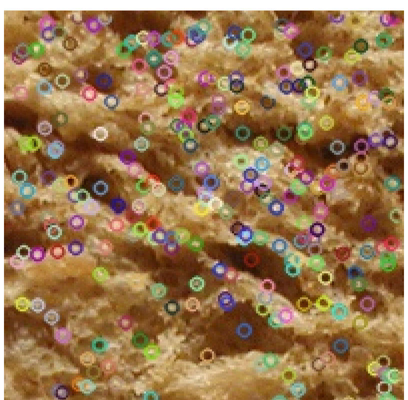


Feature for Classification

- SIFT-based texture classification – how to do this?

bread

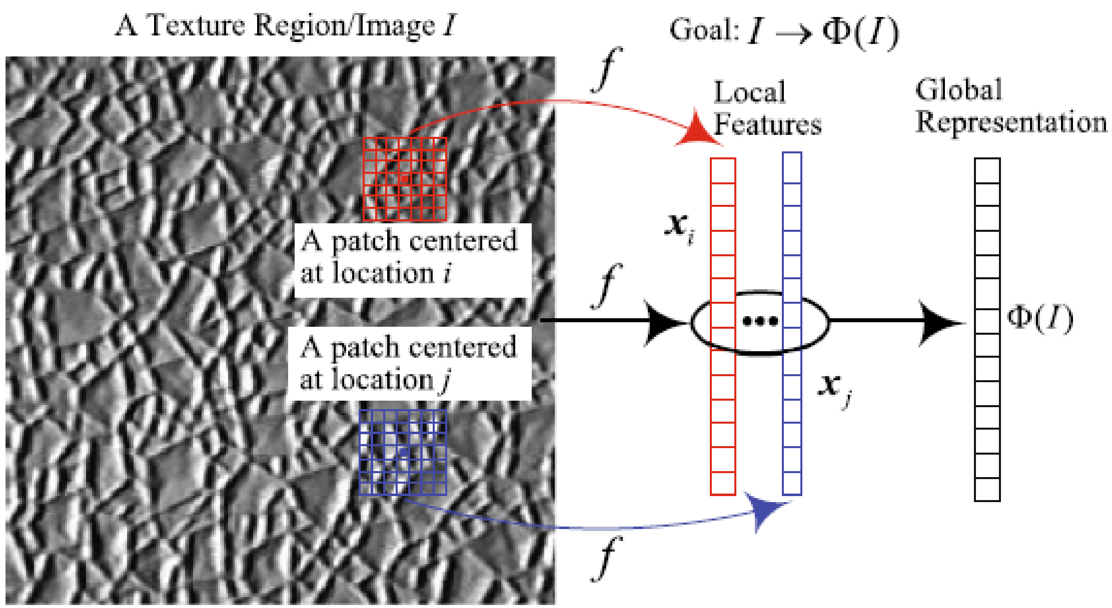
cracker



Problem: the number of SIFT keypoints (and thus the number of SIFT feature descriptors) may vary highly between images

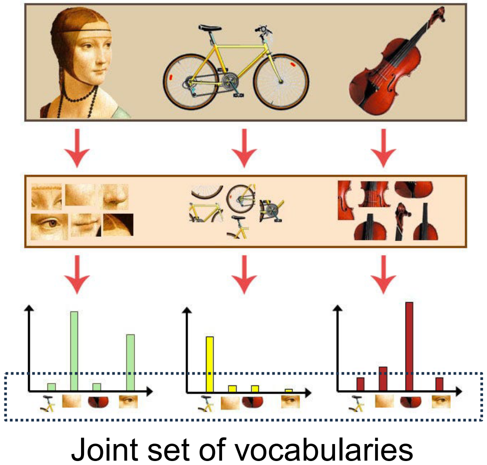
Feature Encoding

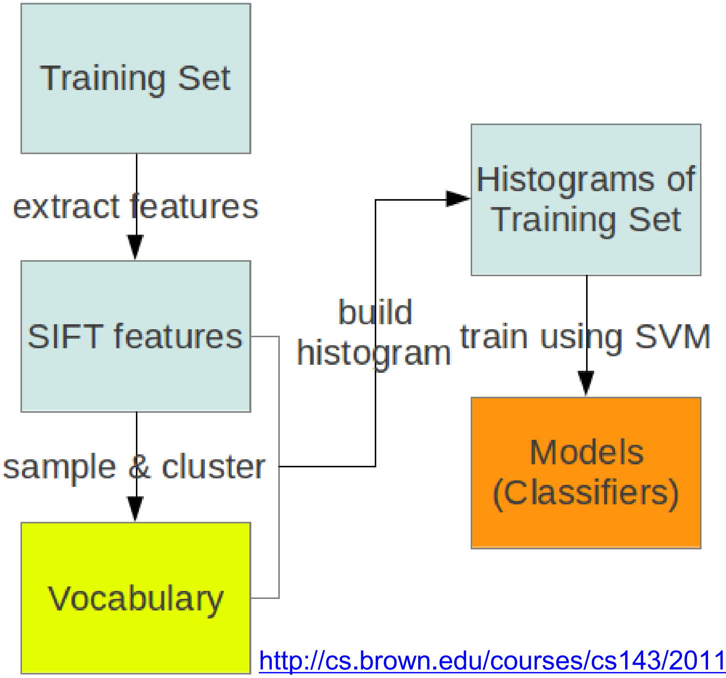
- Global encoding of local SIFT features
  - Integrate the local features (SIFT keypoint descriptors) of an image into a global vector to represent the whole image



Feature Encoding

- Most popular method: Bag-of-Words (BoW)
  - The variable number of local image features are encoded into a fixed-dimensional histogram to represent each image

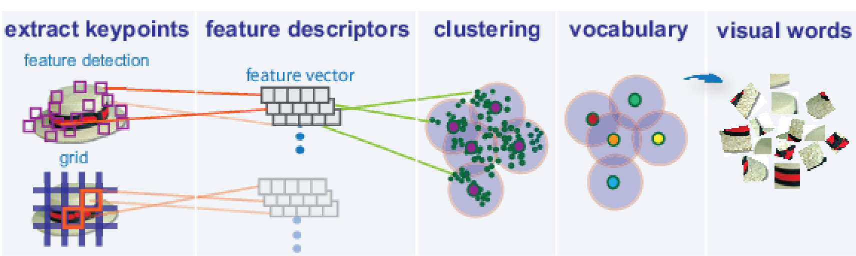


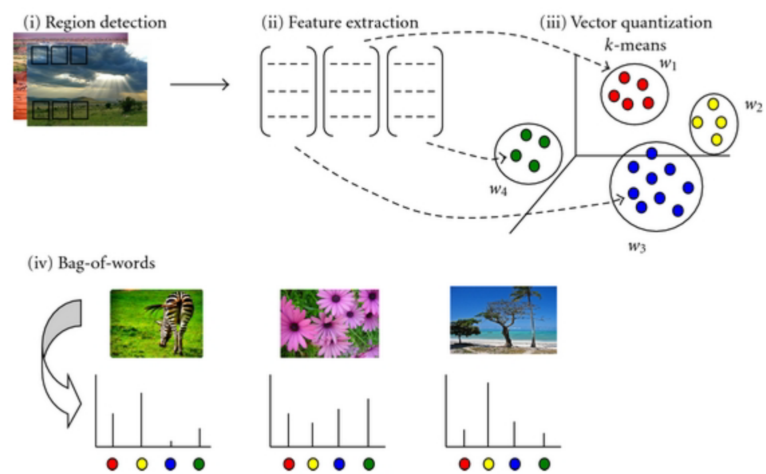


http://cs.brown.edu/courses/cs143/2011/results/proj3/hangsu/

Feature Encoding

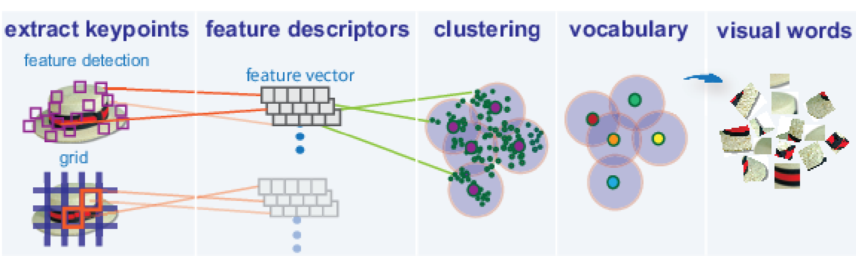
- Bag-of-Words (BoW) – step 1
  - Create the vocabulary from the set of local descriptors (SIFT keypoint descriptors) extracted from the training data
  - This vocabulary represents the categories of local descriptors

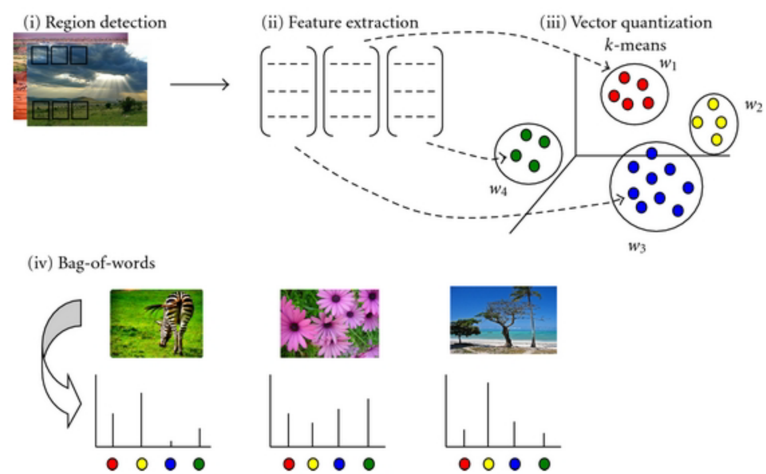




Feature Encoding

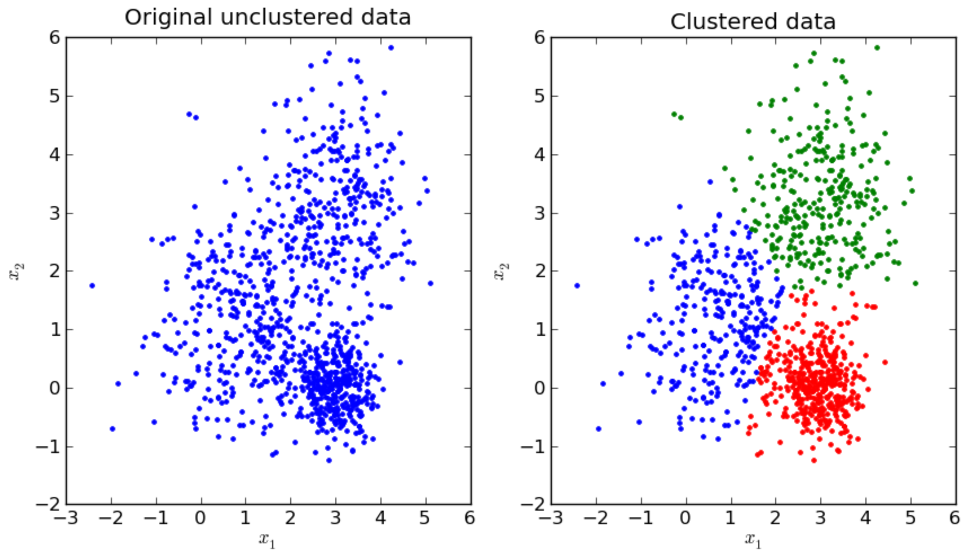
- Bag-of-Words (BoW) – step 1
  - Extracting local feature descriptors
  - Clustering
    - k-means clustering is one of the simplest and most popular unsupervised learning approaches that perform automatic clustering (partitioning) of the training data into multiple categories





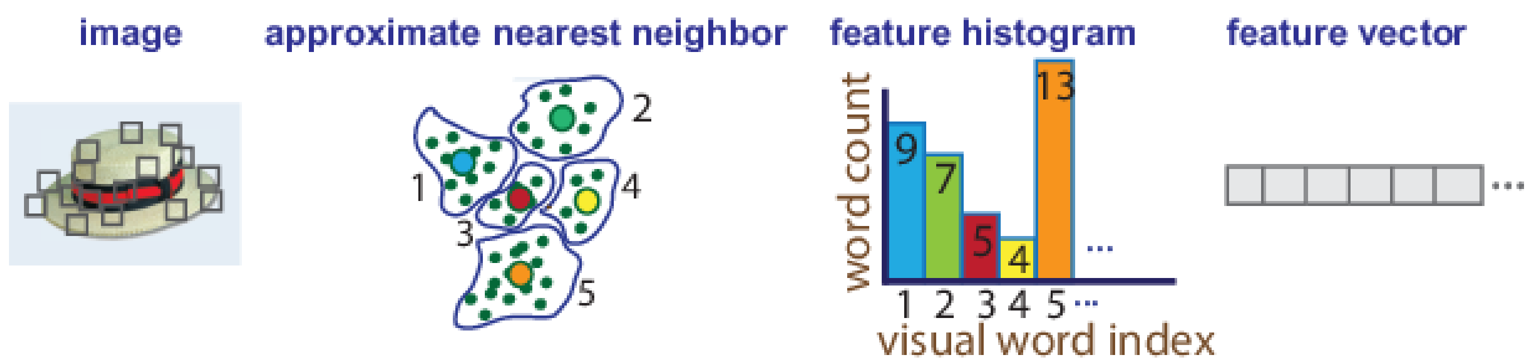
Feature Encoding

- Bag-of-Words (BoW) – step 1
  - K-means clustering:
    - Initialize: *k* cluster centres, typically randomly
    - Iterate: 1) Assign data (feature vectors) to the closest cluster (Euclidean distance)
      2) Update cluster centres as the mean of the data samples in each cluster
    - Terminate: When converged or the number of iterations reaches the maximum



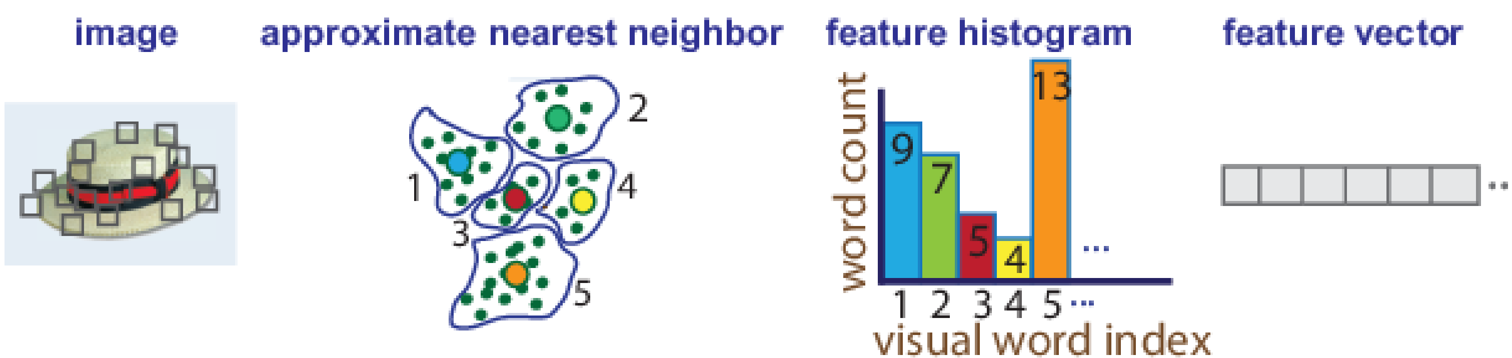
Feature Encoding

- Bag-of-Words (BoW) – step 2
  - The cluster centres are the “visual words” which form the “vocabulary” that is used to represent an image
  - An individual local feature descriptor (e.g. SIFT keypoint descriptor) is assigned to one visual word with the smallest distance



Feature Encoding

- Bag-of-Words (BoW) – step 2
  - For an image, the number of local feature descriptors assigned to each visual word is computed
  - The numbers are concatenated into a vector which forms the BoW representation of the image



Feature Encoding

- Example feature vectors of texture images

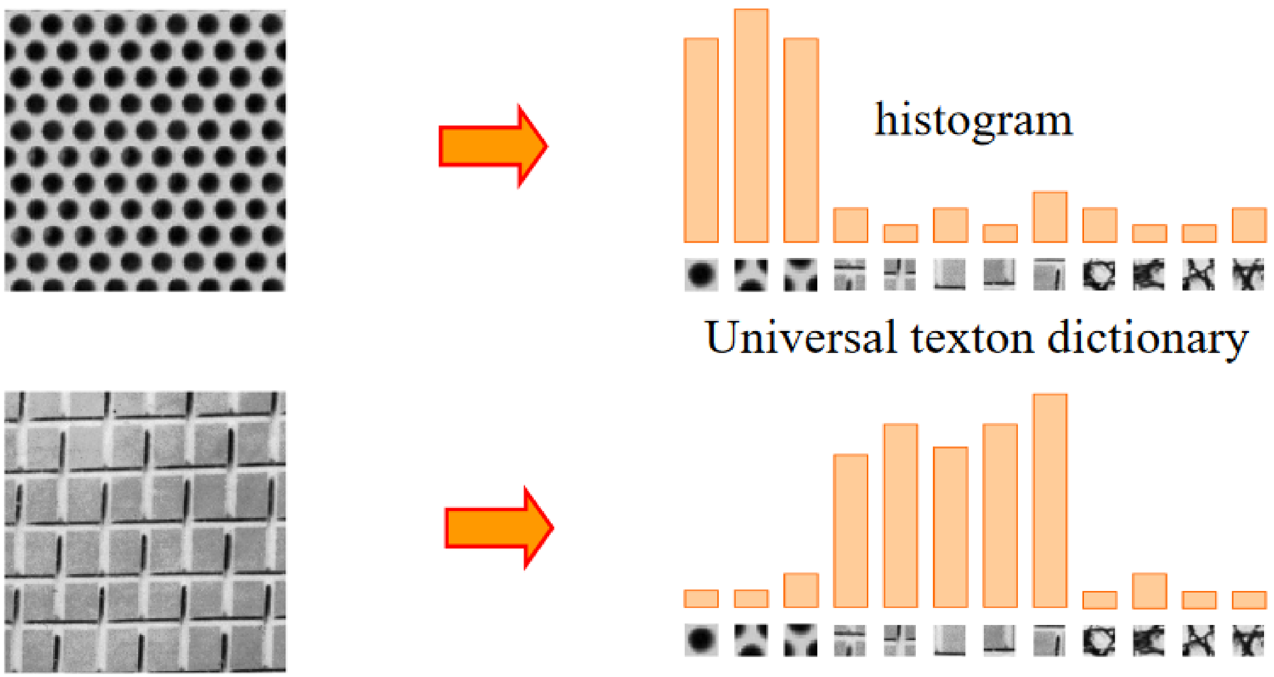
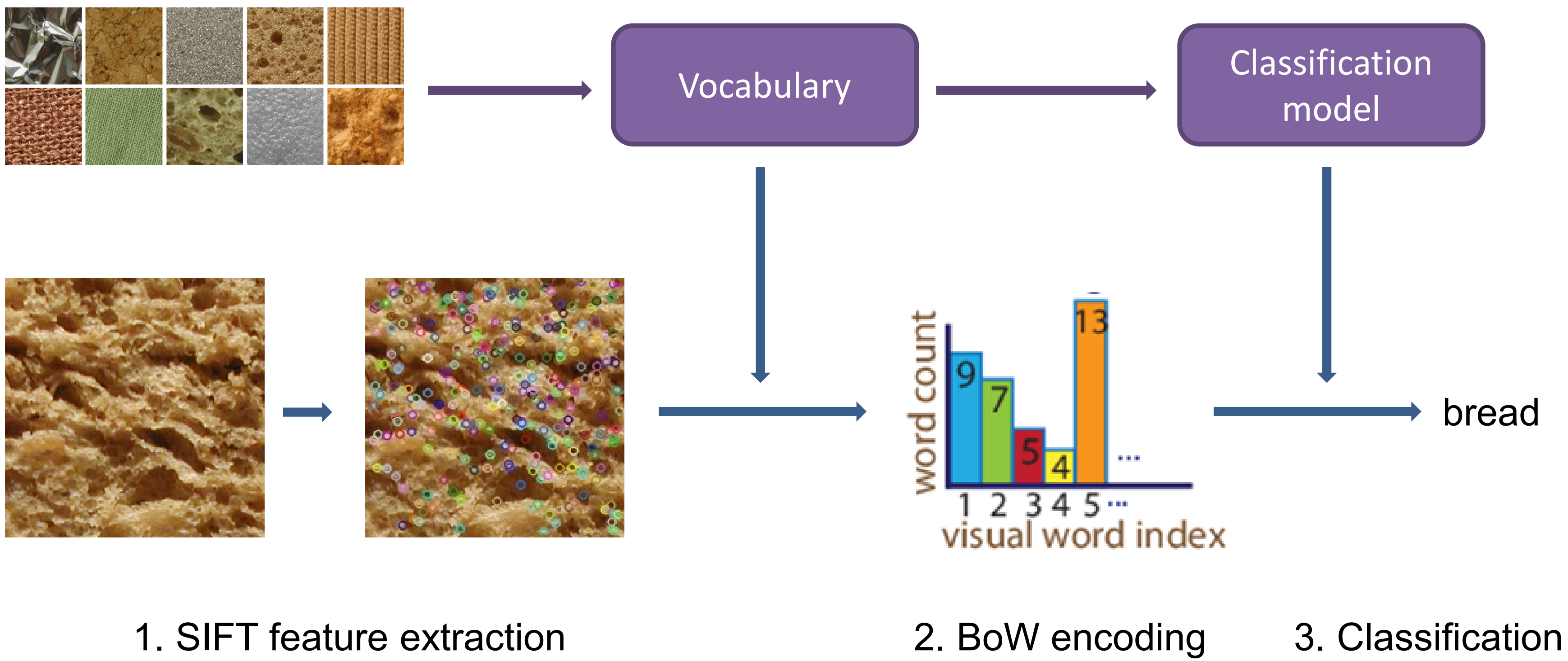


Image from Cordelia Schmit

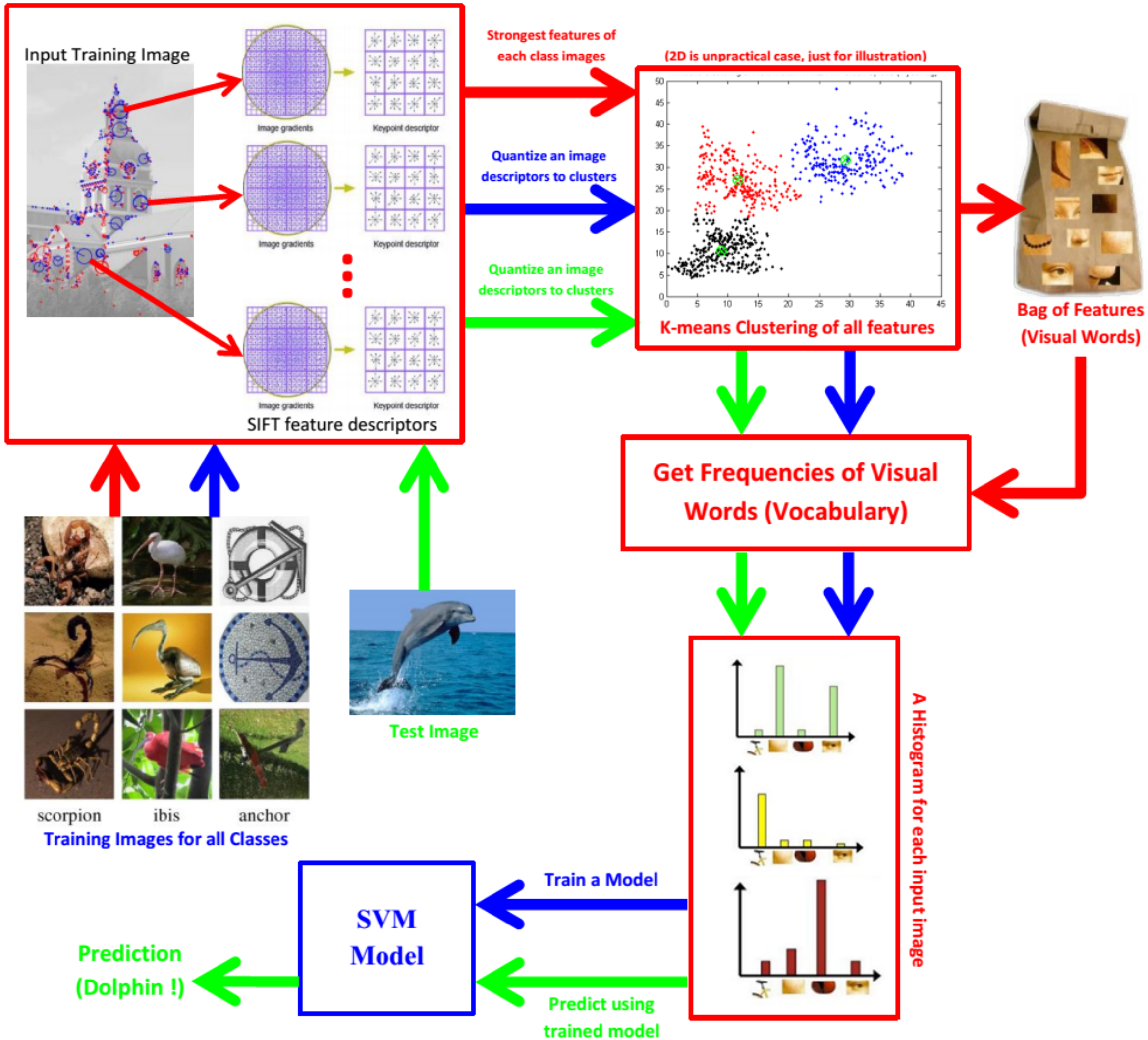
Application Example

- SIFT-based texture classification



Application Example

- SIFT-based classification



Build vocabulary

Train classifier

Classify image

http://heraqi.blogspot.com/2017/03/BoW.html

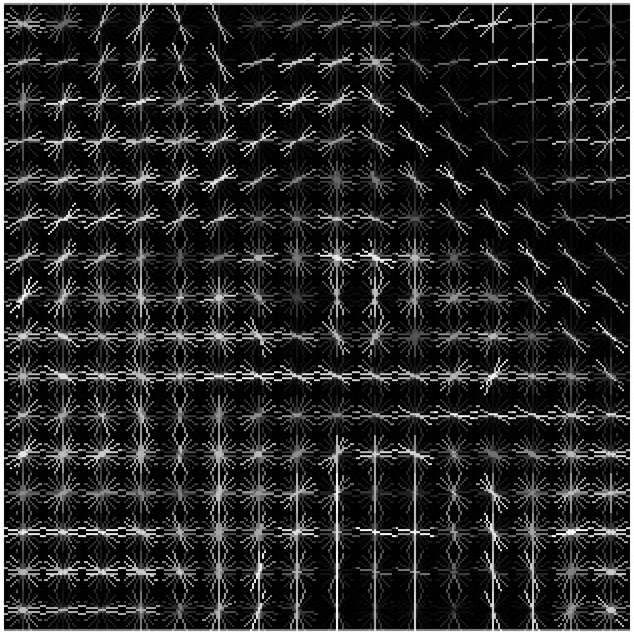
Feature Encoding

- Local features can be other types of features, not just SIFT
  - LBP, SURF, BRIEF, ORB
- There are also more advanced techniques than BoW
  - VLAD, Fisher Vector
- A very good source of additional information is VLFeat.org
  - http://www.vlfeat.org/

Histogram of Oriented Gradients

- HOG describes the distributions of gradient orientations in localized areas and does not require initial segmentation

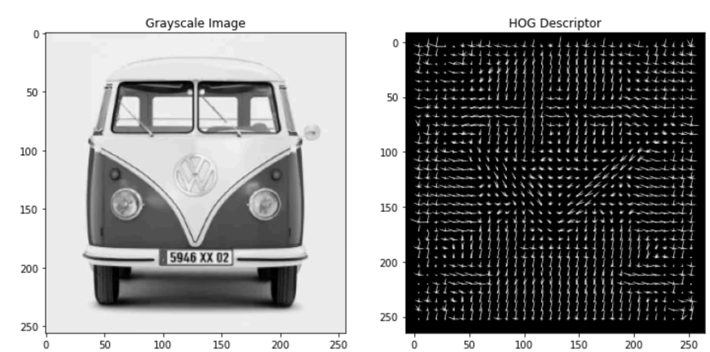
$\Rightarrow$



N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," Computer Vision and Pattern Recognition 2005. https://doi.org/10.1109/CVPR.2005.177

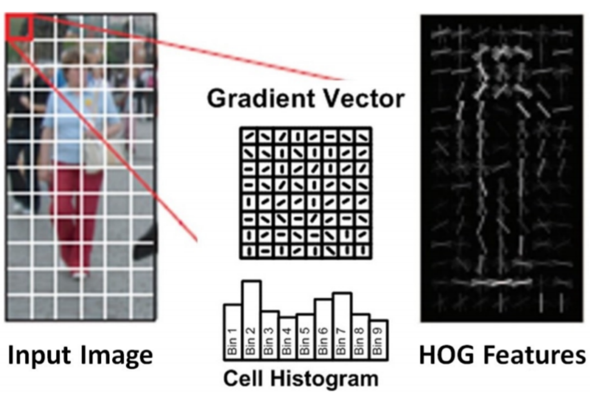
Histogram of Oriented Gradients

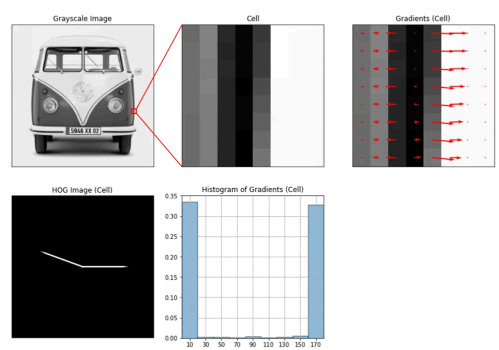
- HoG feature for image description
  - Compute gradients
  - Bin gradients
  - Aggregate blocks (4x4, 16x16 cells)
  - Normalize gradient magnitudes
- Not reliant on magnitude, just direction
  - Invariant to some lighting changes
- Dense on images
- Object detection

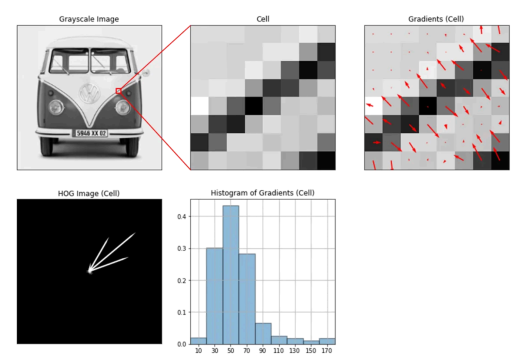


Histogram of Oriented Gradients

- Step 1: Calculate gradient magnitude and orientation at each pixel with a gradient operator => gradient vector
- Step 2: Divide orientations into N bins and assign the gradient magnitude of each pixel to the bin corresponding to its orientation => cell histogram
  - For example 9 bins evenly divided from 0 to 180 degrees



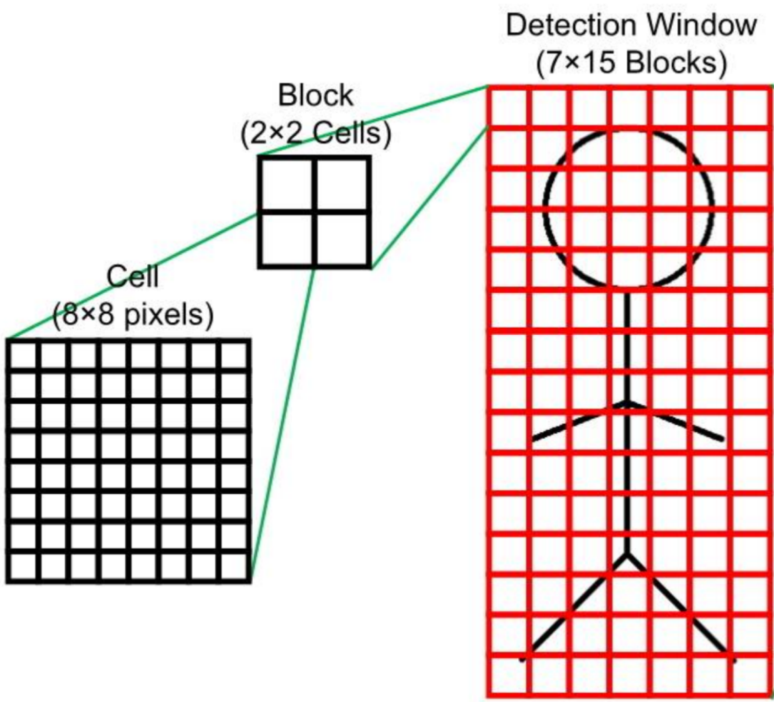




https://medium.com/@dnemutlu/hog-feature-descriptor-263313c3b40d

Histogram of Oriented Gradients

- Step 3: Concatenate and block-normalise cell histograms to generate detection-window level HOG descriptor



orientations/cell

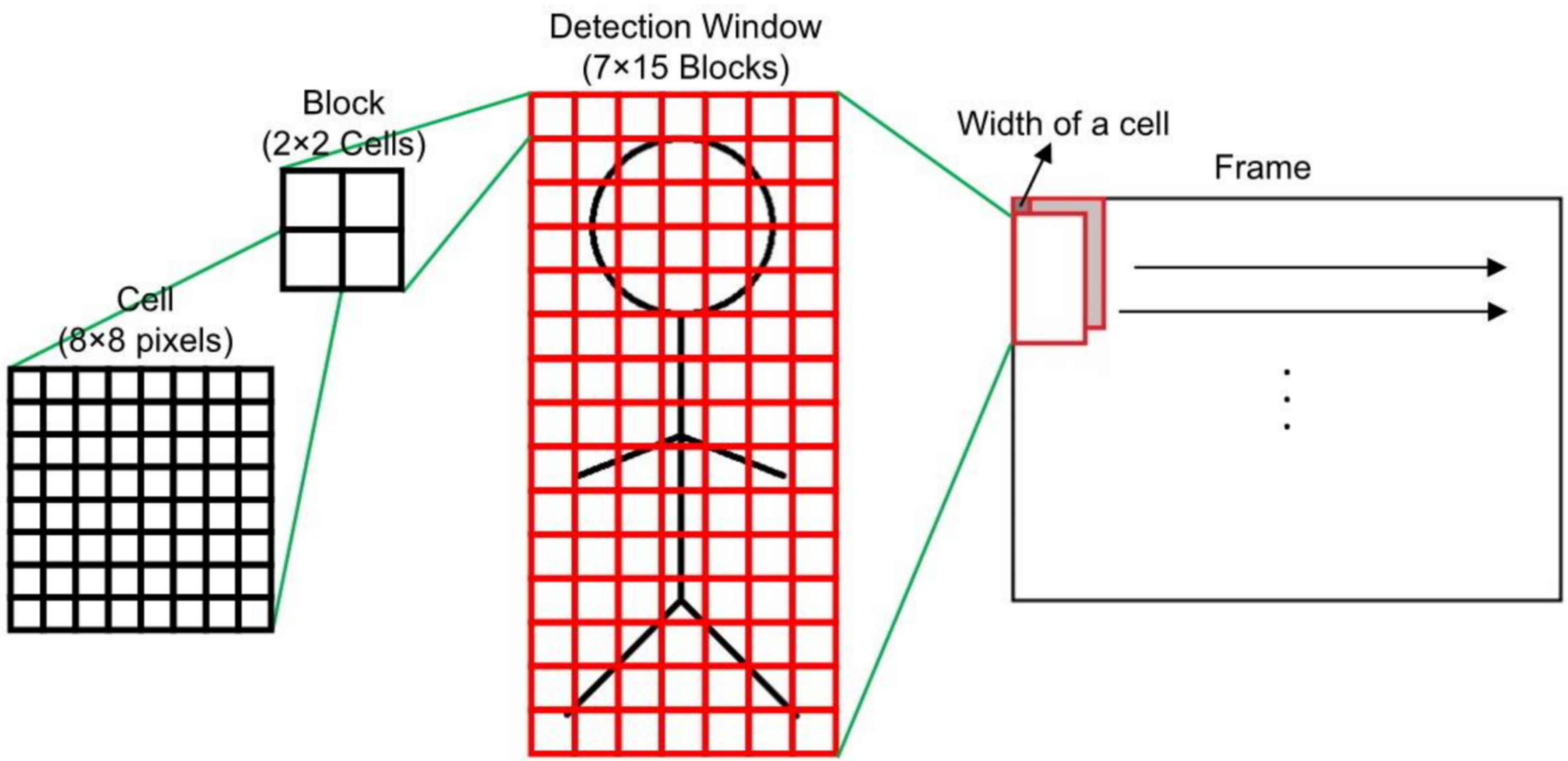
features = (7 x 15) x 9 x 4 = 3780

blocks

cells/block

Histogram of Oriented Gradients

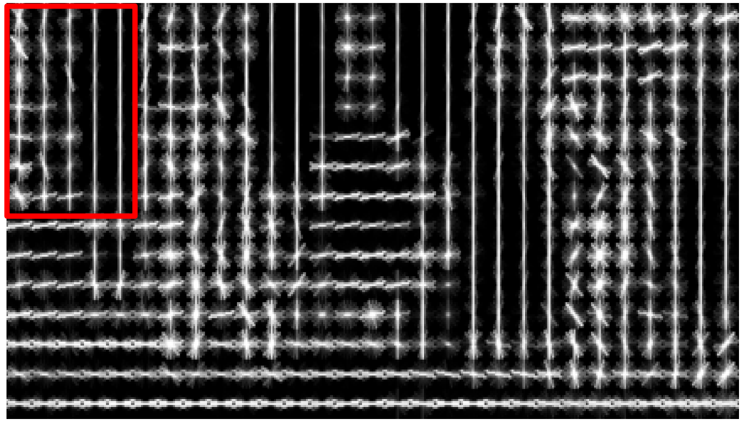
- Detection via sliding window on the image



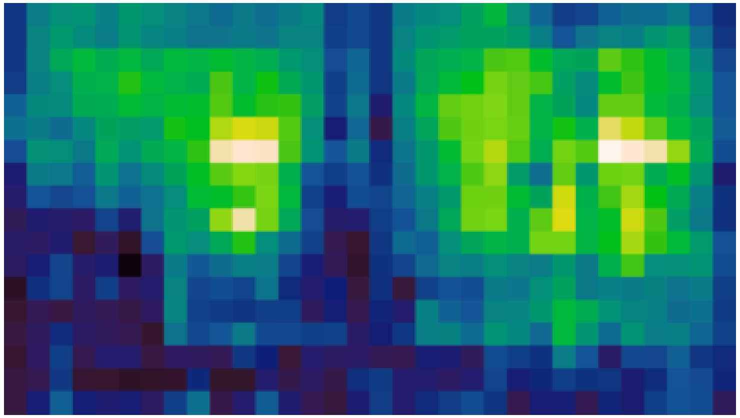
Histogram of Oriented Gradients

- Detection via sliding window on the image

HOG feature map



Detector response map

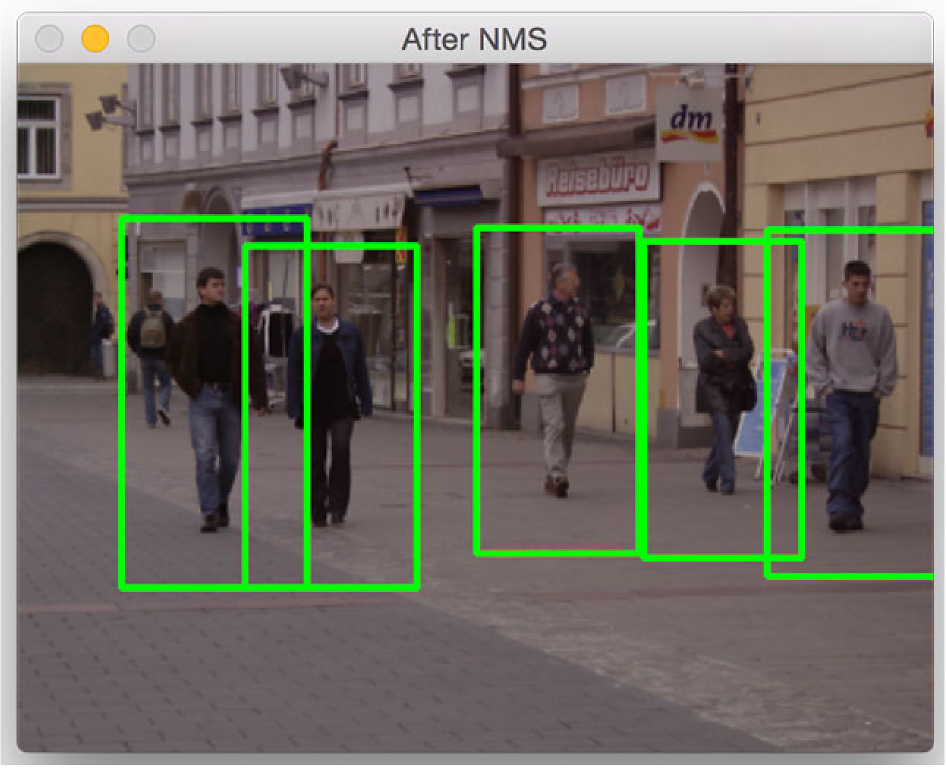


A response map could be computed for example as follows:

- Compute the HOG descriptor for many example windows from a training dataset
- Manually label each example window as “person” or “background”
- Train a classifier (such as a SVM) from these example windows and labels
- For each new (test) image, predict the label of each window using this classifier

Application Example

- Human detection

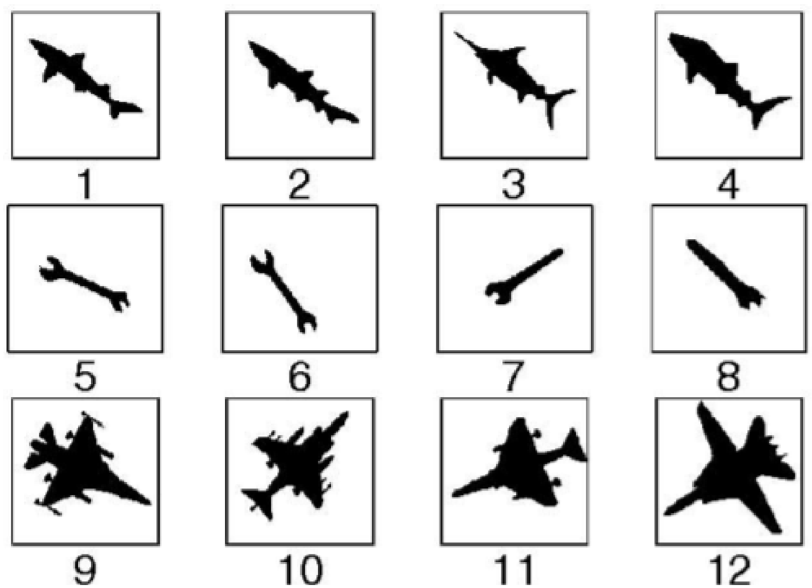


https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/

Shape Features

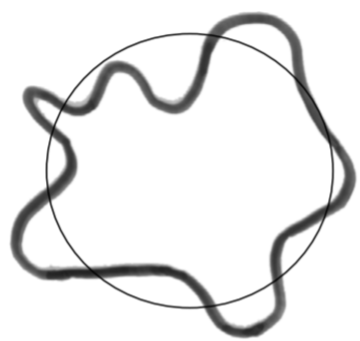
- **Shape** is an essential feature of material objects that can be used to identify and classify them

- Example: object recognition



Basic Shape Features

- Simple geometrical shape descriptors



Compactness:

Ratio of the area of an object to the area of a circle with the same perimeter

Circularity:

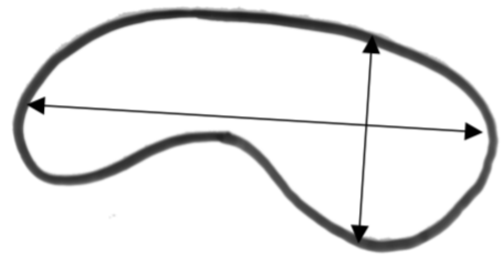
Ratio of 4π times the area of an object to the second power of its perimeter ($4\pi A/P^2$ equals 1 for a circle)

Basic Shape Features

- Simple geometrical shape descriptors

Elongation:

Ratio between the length and width of the object's bounding box

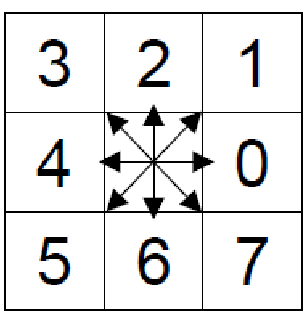


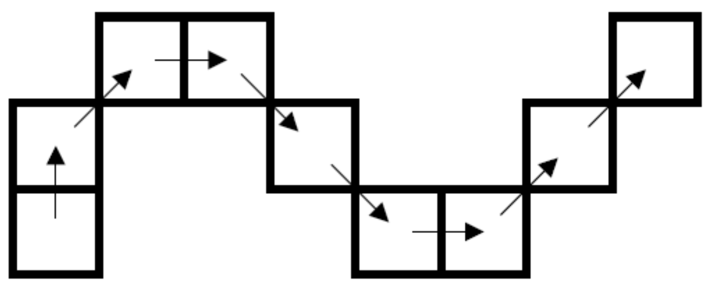
Eccentricity:

Ratio of the length of the minor axis to the length of the major axis

Boundary Descriptors

- Chain code descriptor
  - The shape of a region can be represented by labelling the relative position of consecutive points on its boundary
  - A chain code consists of a list of directions from a starting point and provides a compact boundary representation





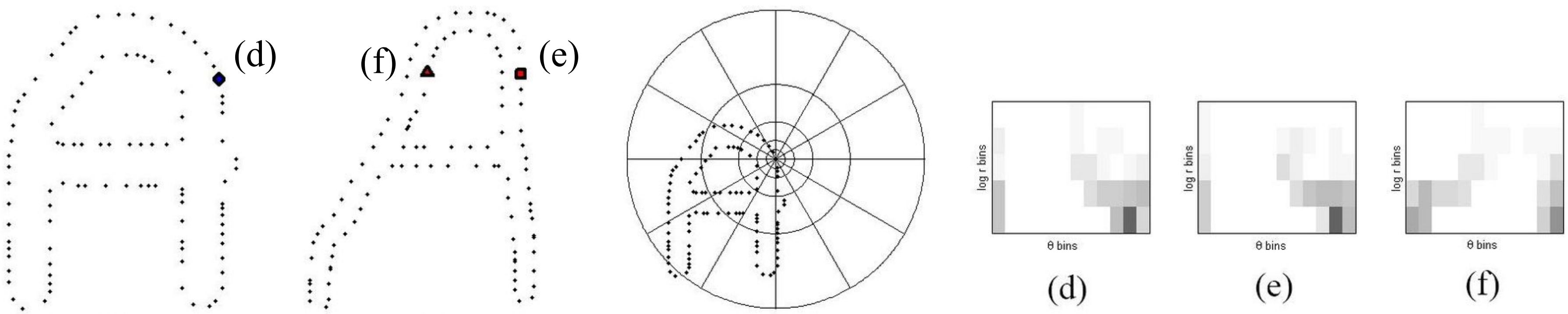
Example:

2,1,0,7,7,0,1,1

Shape Context

- **Shape context** is a point-wise local feature descriptor
  - Pick n points on the contour of a shape
  - For each point p_i construct a histogram h_i of the relative coordinates of the other $n-1$ points => this is the shape context of p_i

$$h_i(k) = \#\{q \neq p_i \ : \ (q - p_i) \in \mathrm{bin}(k)\}$$



(a) (b) (c) (d) (e) (f)

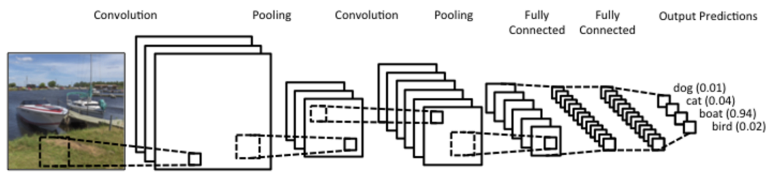
S. Belongie, J. Malik, J. Puzicha (2002), “Shape matching and object recognition using shape contexts,” IEEE Transactions on Pattern Analysis and Machine Intelligence 24(4):509-522. https://doi.org/10.1109/34.993558

Learning Representations

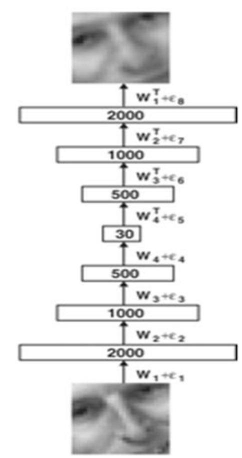
- Handcrafted features
  - LBP, SIFT, HoG, BoW ….
  - Used for very long time
  - Worked well for many applications
- Can the feature engineering process be automatic?
  - Finding discriminative signatures automatically and systematically
- Learning for reorientations
  - a brief overview

Learning Representations

- Supervised learning for representations
  - With some specific task



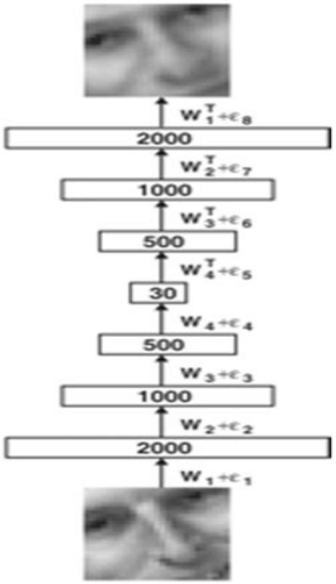
- Unsupervised learning for representations
  - Learning with some protext tasks
    - Image reconstruction



LeCun et al. 1998.
Hinton et al. 2006.

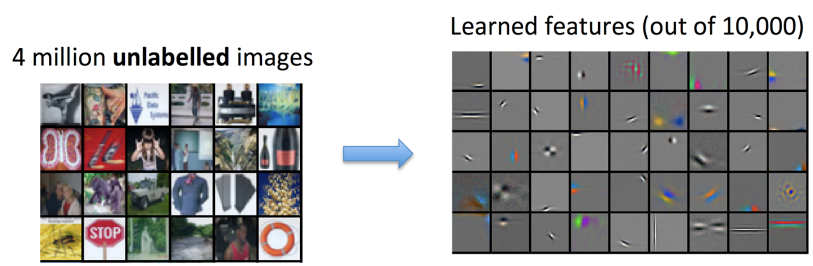
Unsupervised Representation Learning

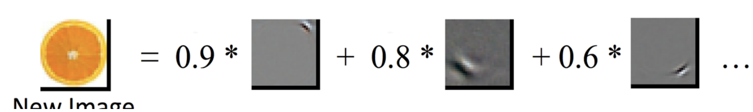
- Representation learning still needs a loss function to provide supervision signal
  - Reconstructing the input image
    - L2 / L1 / GAN loss
    - Learning representations that can reconstruct the image
- Different models/methods
  - Sparse coding
  - (Deep) autoencoders
  - Generative Adversarial Network based feature learning



Sparse Coding

- Learning to obtain a dictionary for representing each image/patch with a sparse vector
- The sparse vector can be used as descriptors for downstream tasks, such as classification

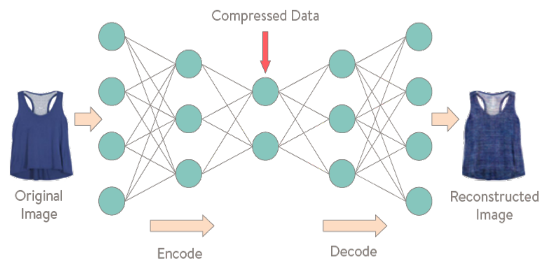


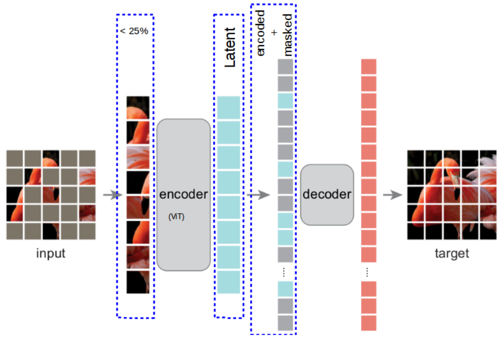


Lee et al. 2006.
Salakhutdinov. 2016.

Autoencoder

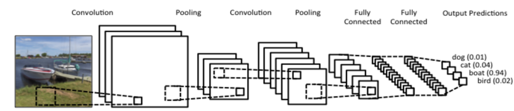
- (Deep) neural network trained with the task for reconstructing images
  - encoder, decoder
  - Extracted features (from encoder) can be used for downstream tasks
  - Can be very large-scale & powerful



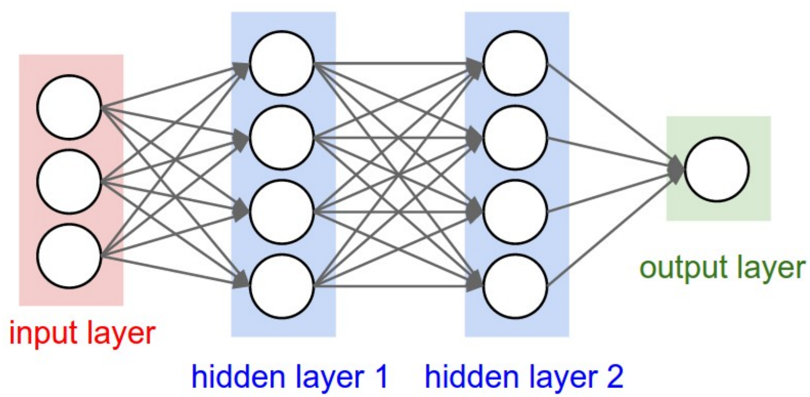


He, Kaiming, et al. "Masked autoencoders are scalable vision learners." *CVPR*. 2022.

Supervised Representation Learning



- Deep neural networks
  - End-to-end model
    - Image – features – output of the task
  - will be discussed more in week 7



Neural Net

Summary

- What and why of feature representation
  - Feature representation is essential in solving almost all types of computer vision problems
- How of feature representation
  - Different feature extractors/descriptors
    - Classical approaches
      - Color features
      - Haralick texture features, LBP, SIFT, BoW, HoG, Shape descriptors
    - Representation learning
      - Unsupervised/supervised representation learning
  - Application cases in various computer vision applications
    - Classification
    - RANSAC for stereo matching (robust fitting)
    - Detection
    - …

References and Acknowledgements

- Szeliski, Chapter 4 (in particular Sections 4.1.1 to 4.1.3 and 4.3.2), Chapter 6 (in particular Sections 6.1.1 to 6.1.4)
- Some content are extracted from the above resource, James Hays slides, slides from Michael A. Wirth, slides from Cordelia Schmit
- L. Liu et al., From BoW to CNN: two decades of texture representation for texture classification, International Journal of Computer Vision, 2019
- And other resources as indicated by the hyperlinks