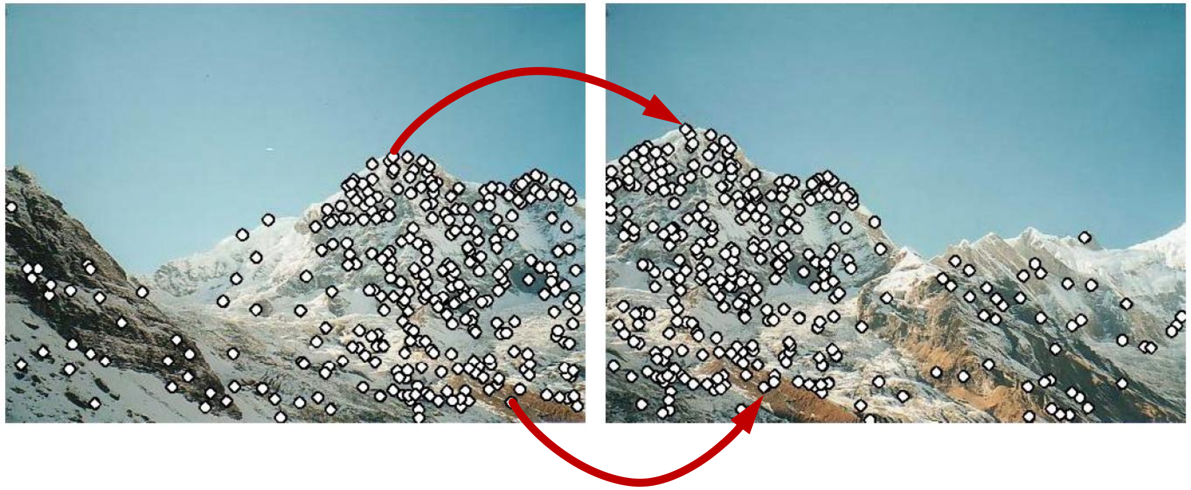
COMP9517

Computer Vision

2025 Term 1 Week 3

A/Prof Yang Song



Feature Representation

Part 1

Topics and learning goals

- Explain the need for feature representation

  Robustness, descriptiveness, efficiency

- Discuss major categories of image features

  Colour features, texture features, shape features

- Understand prominent feature descriptors

  Haralick features, local binary patterns, scale-invariant feature transform

- Show examples of use in computer vision applications

  Image matching and stitching

What are image features?

- Image features are **vectors** that are a **compact representation** of images
- They represent important information shown in an image
- Examples of image features:
  - Blobs
  - Edges
  - Corners
  - Ridges
  - Circles
  - Ellipses
  - Lines

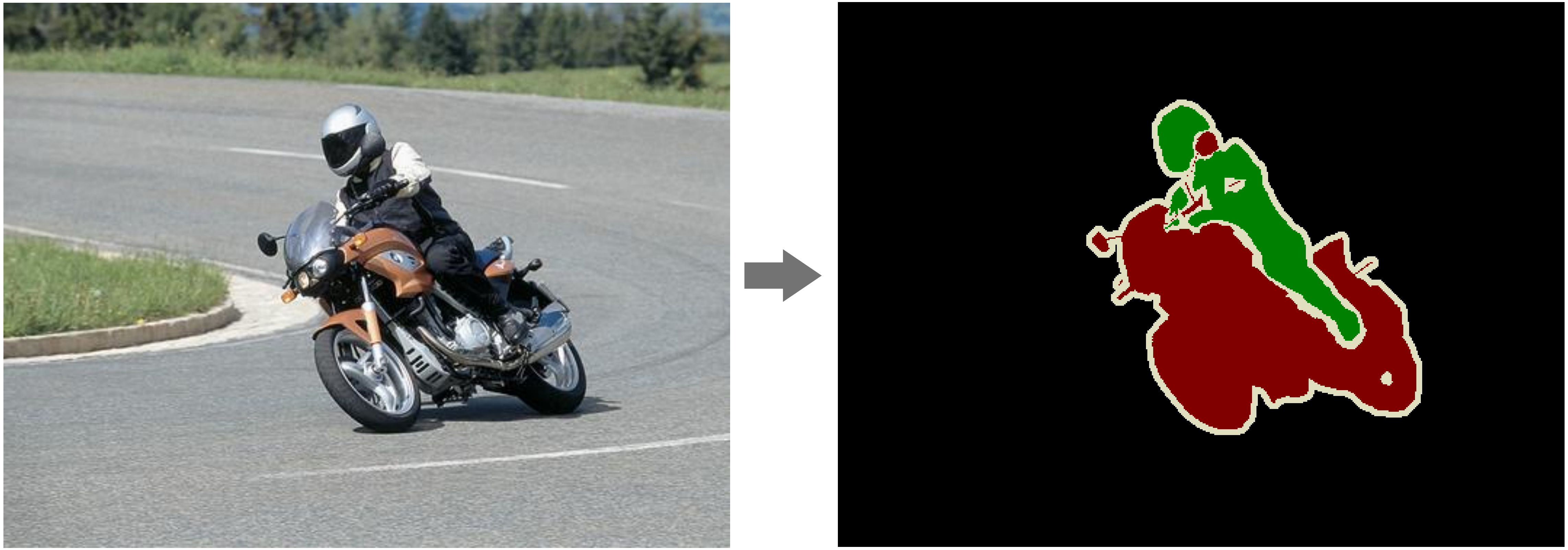
Example

Why do we need image features?

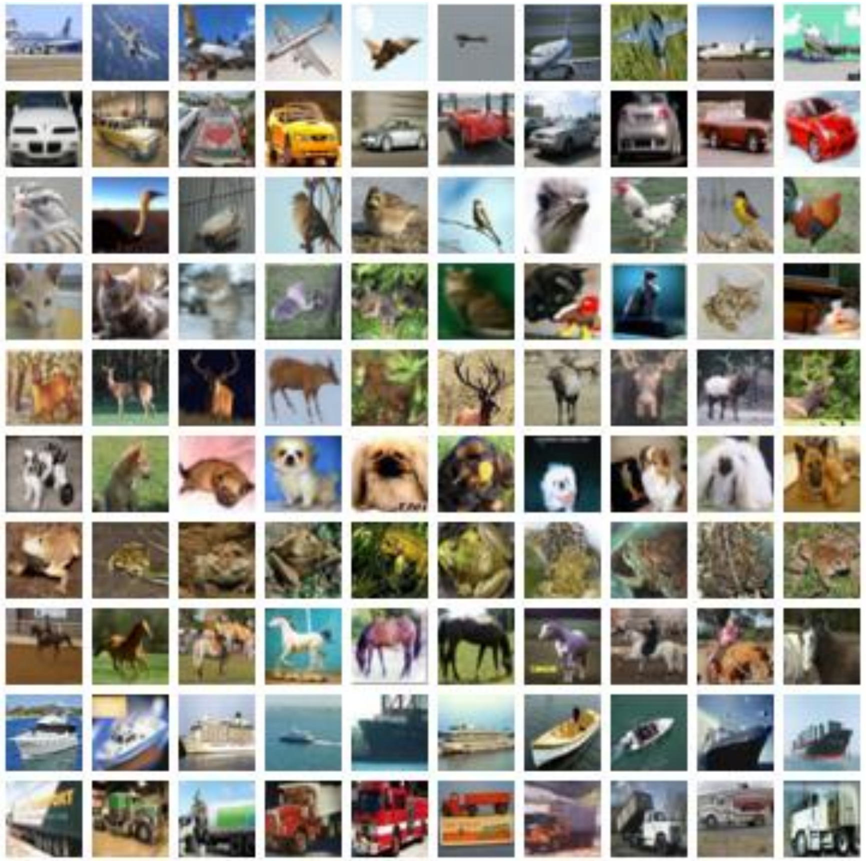
- We need to represent images as feature vectors for further processing in a more efficient and robust way
- Examples of further processing include:
  - Object detection
  - Image segmentation
  - Image classification
  - Image retrieval
  - Image stitching
  - Object tracking

Example: object detection

Example: image segmentation



Example: image classification



Airplane

Car

Bird

Cat

Deer

Dog

Frog

Horse

Boat

Truck

Training set

Unseen

test image

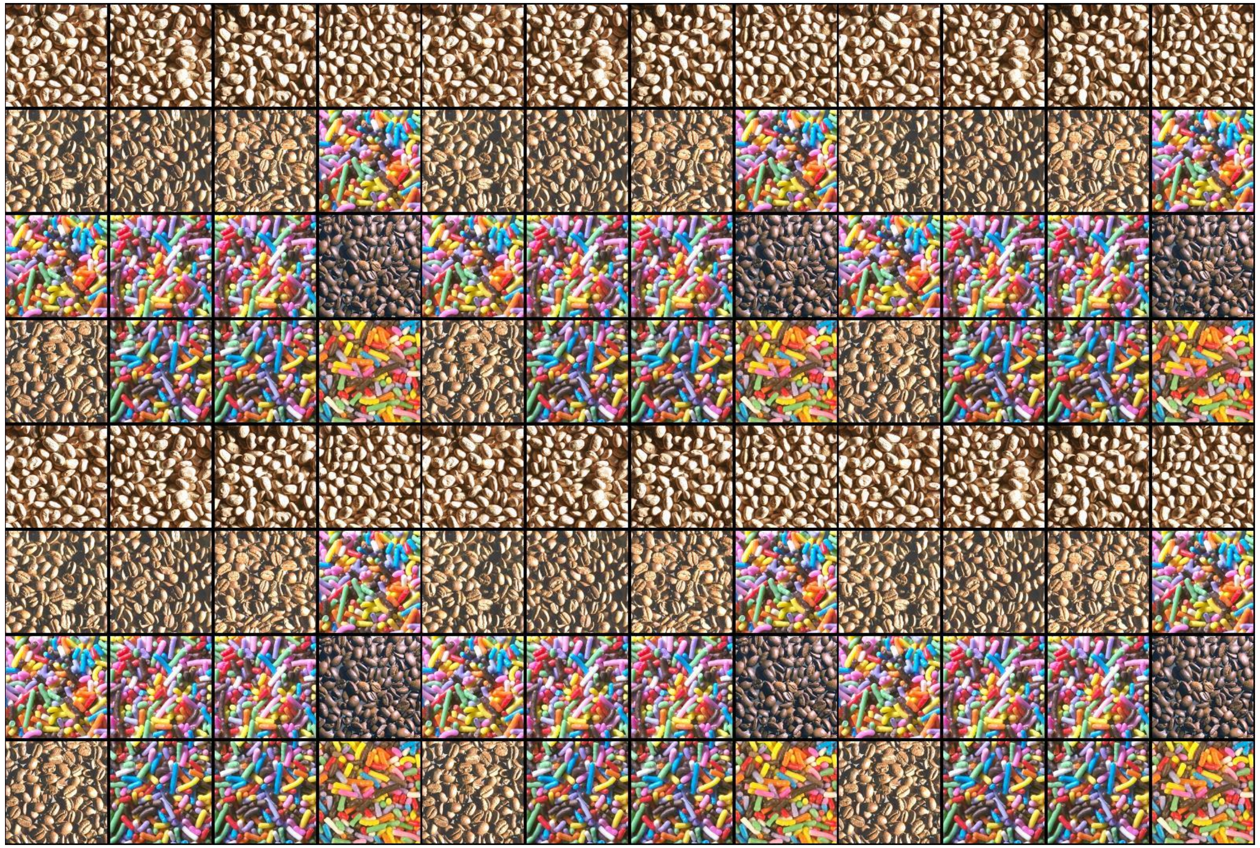
Class?

Example: image retrieval

Given image

Find most similar image

Database



Example: image stitching

Example: object tracking

Need for image features

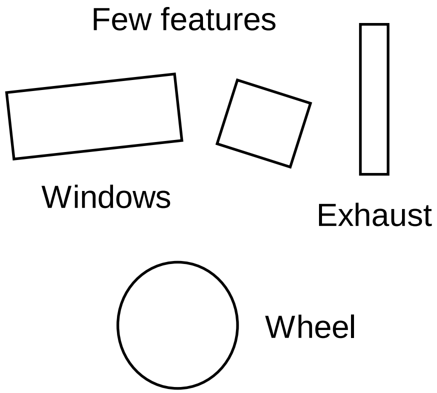
- Why not just use pixel values directly as features?
  - Pixel values change with light intensity, colour, angle
  - They also change with camera orientation
  - And they are highly redundant

1,000 x 1,000 pixels =

1,000,000 values

x 3 channels =

3,000,000 values



Truck!

Desirable properties of features

- Reproducibility (robustness)
  - Should be detectable at the same locations in different images despite changes in illumination and viewpoint
- Saliency (descriptiveness)
  - Similar salient points in different images should have similar features
- Compactness (efficiency)
  - Fewer features
  - Smaller features

General framework

Object detection

Image segmentation

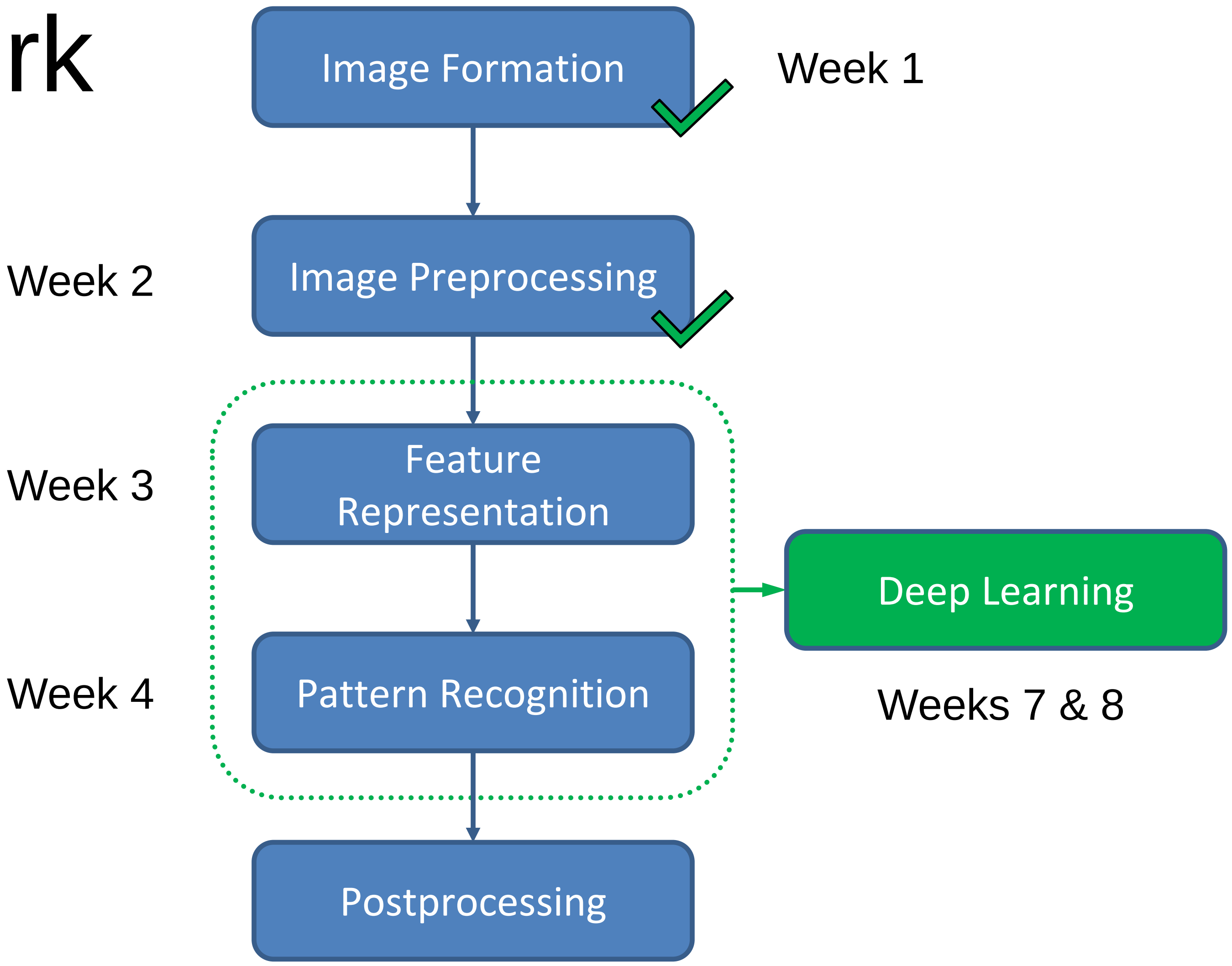
Image classification

Image retrieval

Image stitching

Object tracking

…



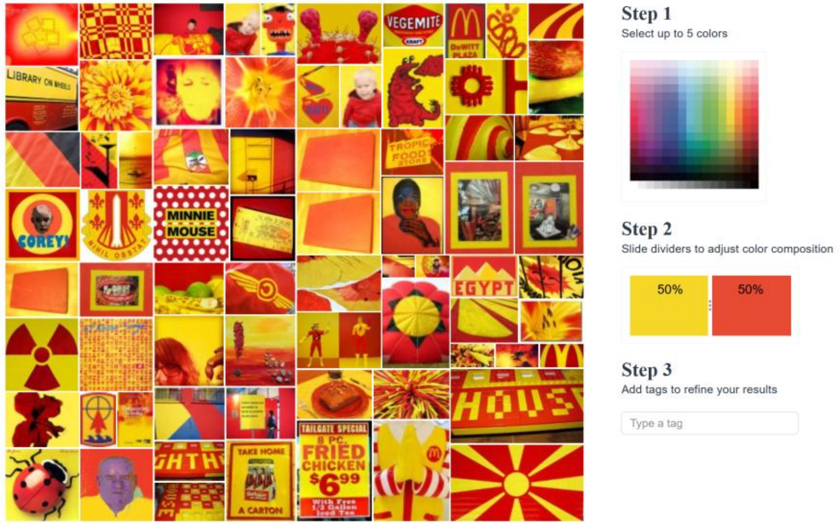
Types of image features

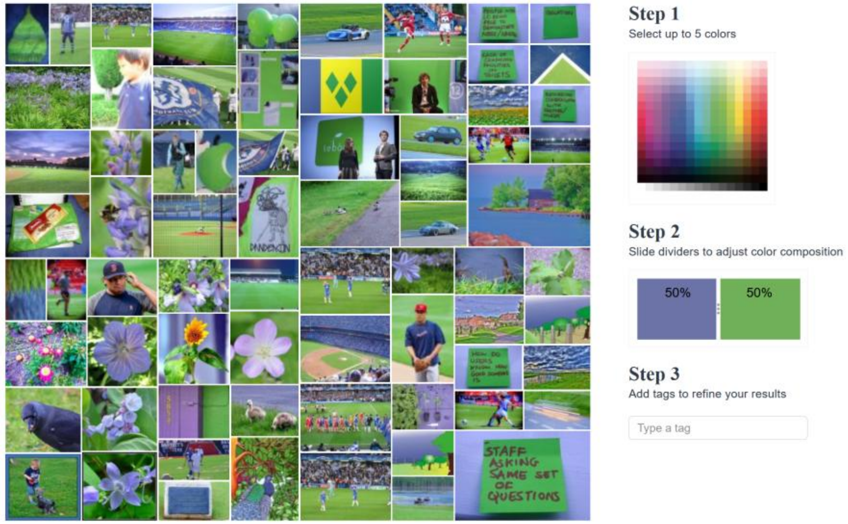
- Colour features **Part 1**
  - Colour histogram
  - Colour moments
- Texture features
  - Haralick texture features
  - Local binary patterns (LBP)
  - Scale-invariant feature transform (SIFT)

- Shape features **Part 2**
  - Basic shape features
  - Shape context
  - Histogram of oriented gradients (HOG)

Colour features

- Colour is the simplest feature to compute
- Invariant to image scaling, translation and rotation
- Example: colour-based image retrieval

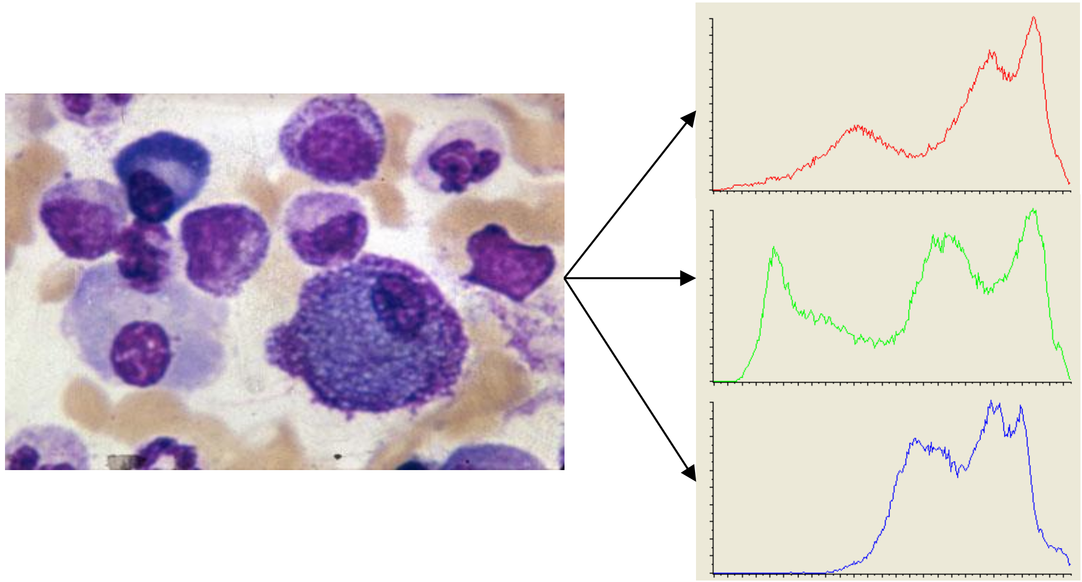




http://labs.tineye.com/multicolr/

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

$$\mathbf{v} = \begin{pmatrix} r_0 \\ \vdots \\ r_{255} \\ g_0 \\ \vdots \\ g_{255} \\ b_0 \\ \vdots \\ b_{255} \end{pmatrix}$$

Colour moments

f_{ij} = value of the ith colour component of pixel j
N = number of pixels in the image

- Another way of representing colour distributions

First-order moment

$$\mu_i = \frac{1}{N}\sum_{j=0}^{N-1} f_{ij}$$

(Mean)

Second-order moment

$$\sigma_i = \sqrt[2]{\frac{1}{N}\sum_{j=0}^{N-1}\left(f_{ij}-\mu_i\right)^2}$$

(Standard Deviation)

Third-order moment

$$s_i = \sqrt[3]{\frac{1}{N}\sum_{j=0}^{N-1}\left(f_{ij}-\mu_i\right)^3}$$

(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

Example application

- Colour-based image retrieval

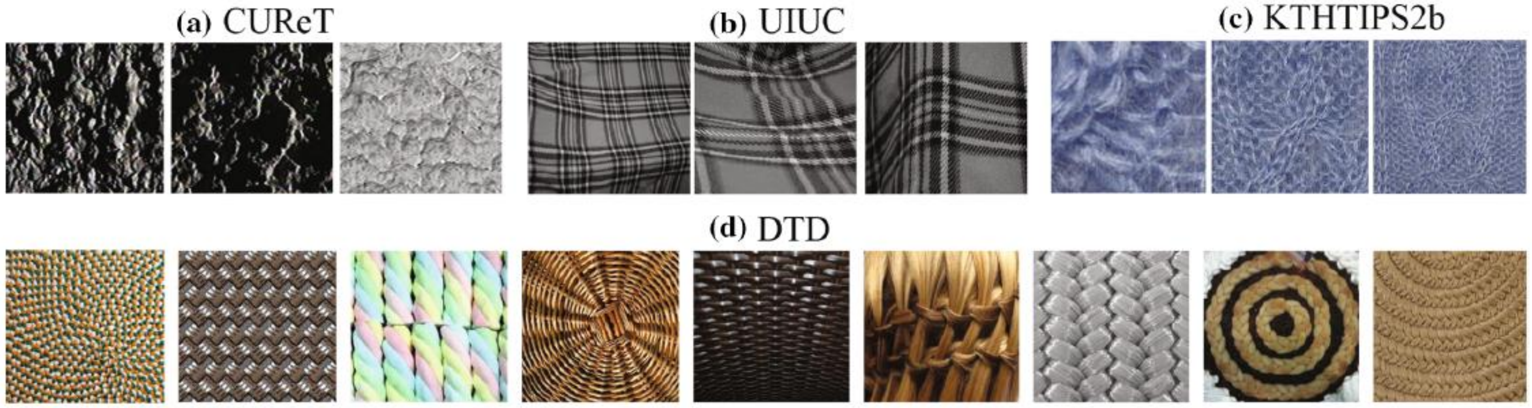
https://doi.org/10.1016/j.csi.2010.03.004

Using only colour histogram information

Using colour + texture + shape information

Texture features

- Visual characteristics and appearance of objects
- Powerful discriminating feature for identifying visual patterns
- Properties of structural homogeneity beyond colour or intensity
- Especially used for texture classification



https://arxiv.org/abs/1801.10324

Haralick texture features

- Array of statistical descriptors of image patterns
- Capture spatial relationship between neighbouring pixels
- Step 1: Construct the gray-level co-occurrence matrix (GLCM) – representing the frequency of pixel intensity pairs occurring at a specific offset and direction
- Step 2: Compute the Haralick feature descriptors from the GLCM – that summarises texture information (how pixel intensities are spatially related)

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

Haralick texture features

- Step 1: Construct the GLCMs
  - Given distance d and orientation angle a
  - Compute co-occurrence count $p_{(d,a)}(i_1, i_2)$ of going from gray level i_1 to i_2 at d and a
  - Construct matrix $\mathbf{P}_{(d,a)}(i_1, i_2)$ with elements (i_1, i_2) being $p_{(d,a)}(i_1, i_2)$
  - If an image has L distinct gray levels, the matrix size is $L \times L$

Example image:

$L = 4$

0	0	0	0	1	1	1	2
0	0	0	1	1	2	2	3
0	0	1	1	2	2	3	3
0	2	2	3	3	2	2	1
2	2	3	3	3	2	1	1
2	3	3	3	2	2	1	0
3	3	2	2	1	1	0	0
3	2	2	1	1	0	0	0

$p_{(d=1,a=0)}(i_1 = 0, i_2 = 0)$

$$\mathbf{P}_{(1,0°)} = \begin{bmatrix} 18 & 6 & 1 & 0 \\ 6 & 14 & 8 & 0 \\ 1 & 8 & 16 & 10 \\ 0 & 0 & 10 & 14 \end{bmatrix}$$

$p_{(d=1,a=90)}(i_1 = 0, i_2 = 2)$

$$\mathbf{P}_{(1,90°)} = \begin{bmatrix} 18 & 5 & 2 & 0 \\ 5 & 10 & 8 & 2 \\ 2 & 8 & 14 & 11 \\ 0 & 2 & 11 & 14 \end{bmatrix}$$

$(i_1 = 3, i_2 = 3)$

Haralick texture features

- Step 1: Construct the GLCMs
  - For computational efficiency L can be reduced by binning (similar to histogram binning) Example: $L = 256/n$ for a constant factor n
  - Different co-occurrence matrices can be constructed by using various combinations of distance d and angular orientation a
  - On their own these co-occurrence matrices do not provide any measure of texture that can be easily used as texture descriptors
  - The information in the co-occurrence matrices needs to be further extracted as a set of feature values such as the Haralick descriptors

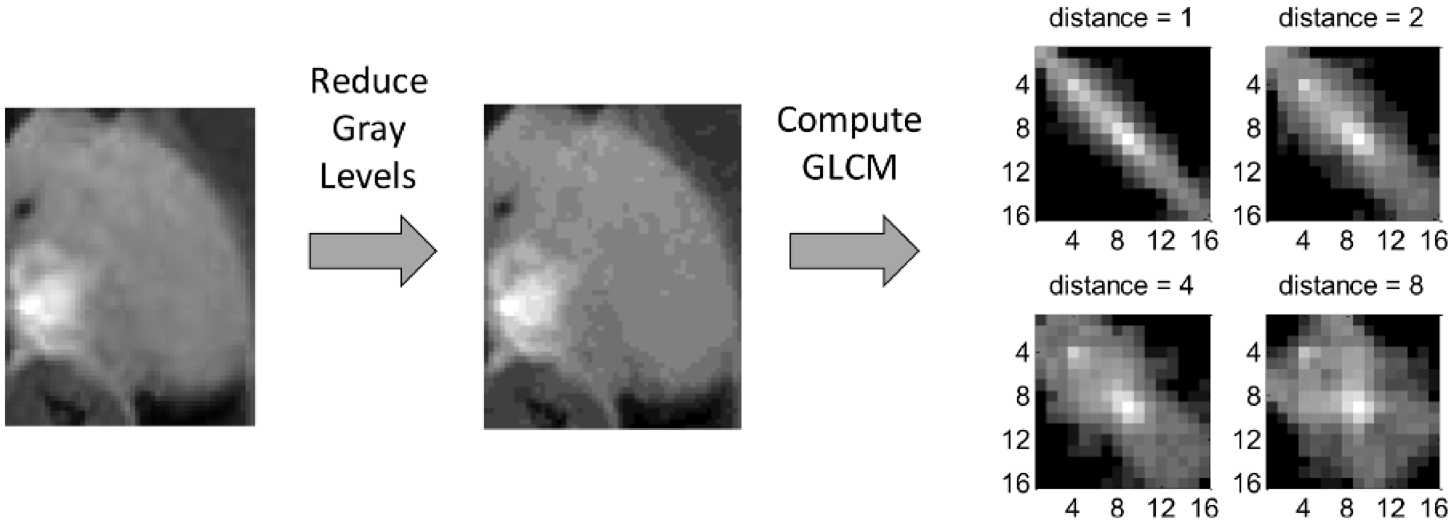
Haralick texture features

- Step 2: Compute the Haralick descriptors from the GLCMs
  - One set of Haralick descriptors for each GLCM for a given d and a

No	Features	Formula
1	Angular Second Moment	$\Sigma i \, \Sigma j \; p(i,j)^2$
2	Contrast	$\Sigma_{n=0}^{Ng-1} n^2 \{ \Sigma_{i=1}^{Ng} \Sigma_{j=1}^{Ng} p(i,j) \}, \lvert i-j \rvert = n$
3	Correlation	$\frac{\Sigma i \Sigma j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$
4	Sum of Squares: Variance	$\Sigma_i \Sigma_j (i-\mu)^2 p(i,j)$
5	Inverse Difference Moment	$\Sigma_i \Sigma_j \frac{1}{1+(i-j)^2} p(i,j)$
6	Sum Average	$\Sigma_{i=2}^{2N_g} i p_{x+y}(i)$
7	Sum Variance	$\Sigma_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i)$
8	Sum Entropy	$-\Sigma_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\} = f_8$
9	Entropy	$-\Sigma_i \Sigma_j \, p(i,j) \log(p(i,j))$
10	Difference Variance	$\Sigma_{n=0}^{Ng-1} i^2 p_{x-y}(i)$
11	Difference Entropy	$-\Sigma_{n=0}^{Ng-1} p_{x-y}(i) \log\{p_{x-y}(i)\}$
12	Info. Measure of Collection 1	$\frac{\mathrm{HXY} - \mathrm{HXY1}}{\max\{HX, HY\}}$
13	Info. Measure of Collection 2	$(1 - \exp[-2(HXY2 - HXY)])^{\frac{1}{2}}$
14	Max. Correlation Coefficient	*The square root of the second largest eigenvalue of Q, where* $Q(i,j) = \Sigma_k \frac{p(i,k)p(j,k)}{p_x(i)p_y(k)}$

Haralick texture features

- Example:
  - Often used in medical imaging studies due to their simplicity and interpretability



1. Preprocess the MRI images
2. Extract Haralick, run-length, and histogram features
3. Apply feature selection
4. Classify using machine learning algorithms

Yang et al., Evaluation of tumor-derived MRI-texture features for discrimination of molecular subtypes and prediction of 12-month survival status in glioblastoma, Medical Physics, 2015.

Local binary patterns

- Describe the spatial structure of local image texture
  - Divide the image into cells of $N \times N$ pixels (for example $N = 16$ or 32)
  - Compare each pixel in a given cell to each of its 8 neighbouring pixels
  - If the centre pixel value is greater than the neighbour value, write 0, otherwise write 1
  - This gives an 8-digit binary pattern per pixel, representing a value in the range 0…255

Example:

0	0	0	0	1	1	1	2
0	0	0	1	1	2	2	3
0	0	1	1	2	2	3	3
0	2	2	3	3	2	2	1
2	2	3	3	3	2	1	1
2	3	3	3	2	2	1	0
3	3	2	2	1	1	0	0
3	2	2	1	1	0	0	0

→ 1 1 1 1 0 0 0 0 → 240

Local binary patterns

- Describe the spatial structure of local image texture
  - Count the number of times each 8-digit binary number occurs in the cell
  - This gives a 256-bin histogram (also known as the LBP feature vector)
  - Combine the histograms of all cells of the given image
  - This gives the image-level LBP feature descriptor

Example:

0	0	0	0	1	1	1	2
0	0	0	1	1	2	2	3
0	0	1	1	2	2	3	3
0	2	2	3	3	2	2	1
2	2	3	3	3	2	1	1
2	3	3	3	2	2	1	0
3	3	2	2	1	1	0	0
3	2	2	1	1	0	0	0

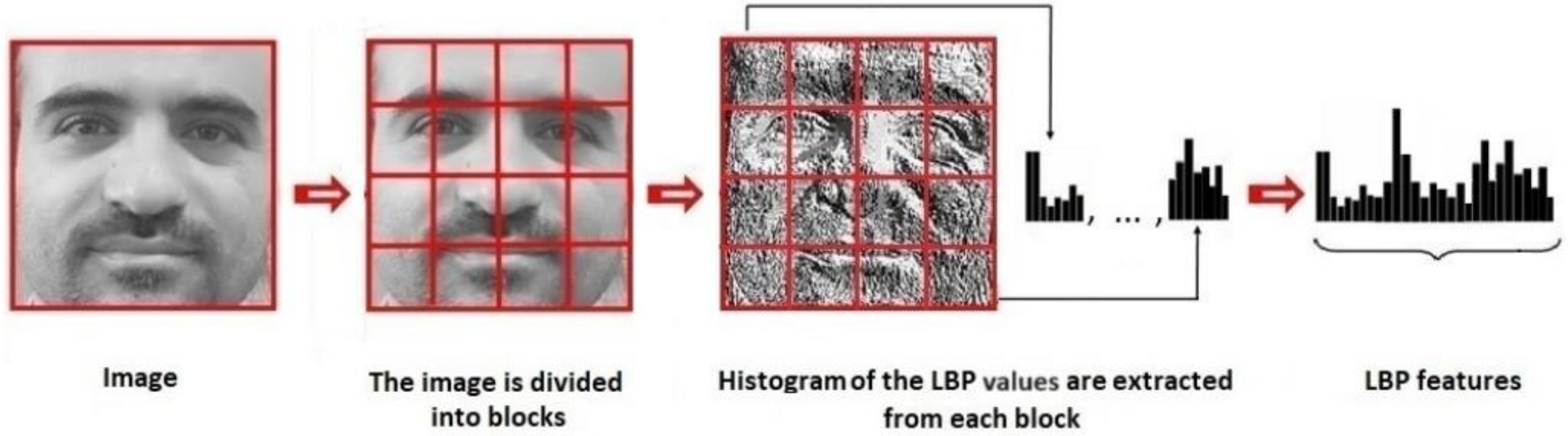
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 1
1 1 1 1 1 0 1 1
…

255
255
241
251
…

histogram

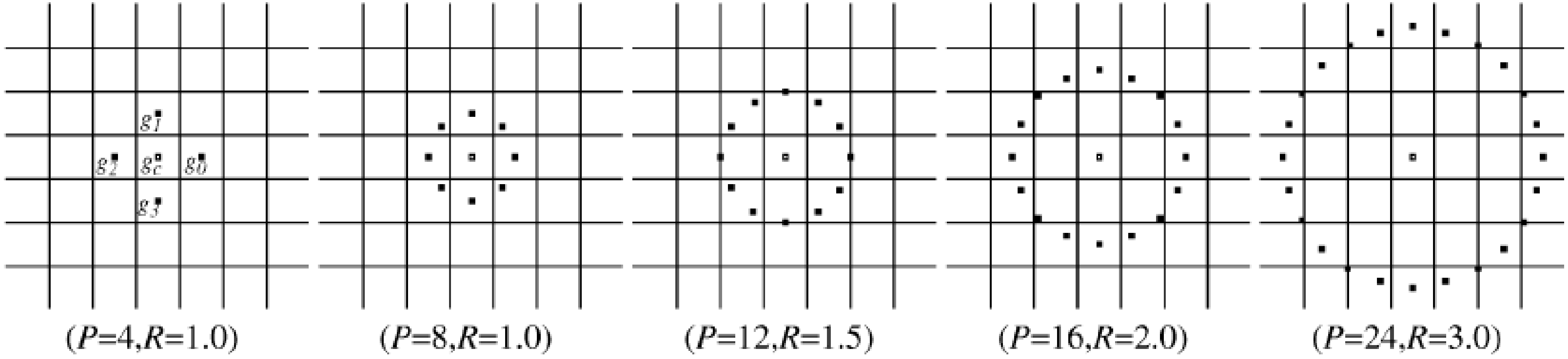
Local binary patterns

- Describe the spatial structure of local image texture
  - Count the number of times each 8-digit binary number occurs in the cell
  - This gives a 256-bin histogram (also known as the LBP feature vector)
  - Combine the histograms of all cells of the given image
  - This gives the image-level LBP feature descriptor



Local binary patterns

- LBP can be multiresolution and rotation-invariant
  - Multiresolution: vary the distance between the centre pixel and neighbouring pixels and vary the number of neighbouring pixels



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

Local binary patterns

- LBP can be multiresolution and rotation-invariant
  - Rotation-invariant: vary the way of constructing the 8-digit binary number by performing bitwise shift to derive the smallest number

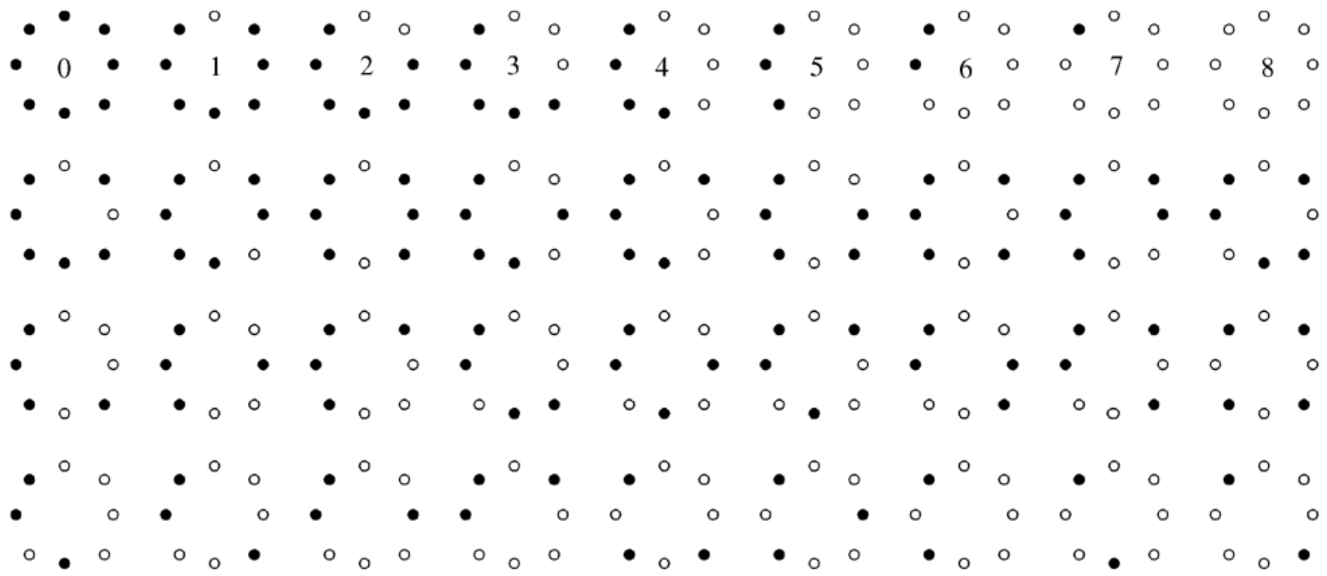
Example:

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

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

Local binary patterns

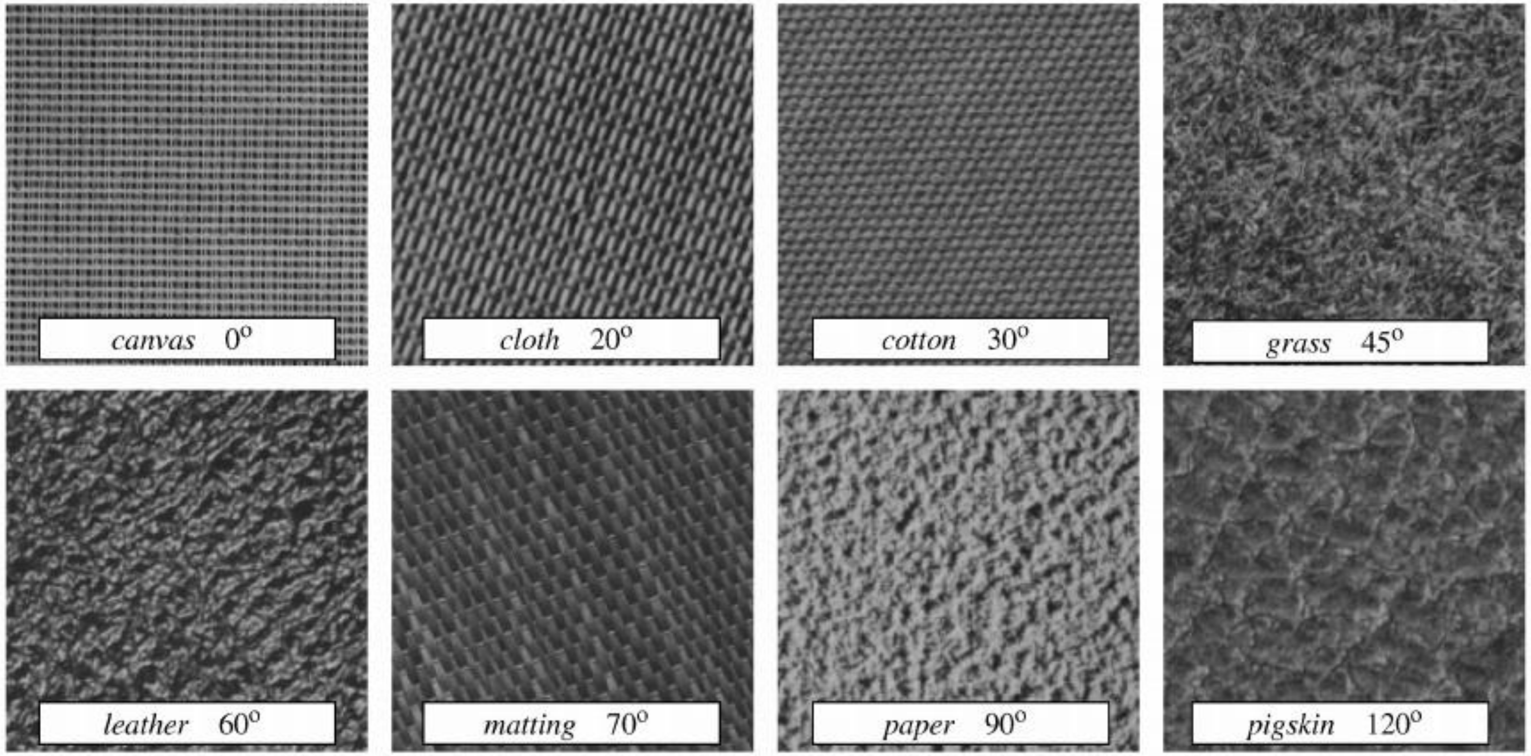
- LBP can be multiresolution and rotation-invariant
  - Rotation-invariant: vary the way of constructing the 8-digit binary number by performing bitwise shift to derive the smallest number



This reduces the LBP feature dimension from 256 to 36

Example application of LBP

- Texture classification



P,R	$LBP_{P,R}$	
	BINS	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

https://doi.org/10.1109/TPAMI.2002.1017623

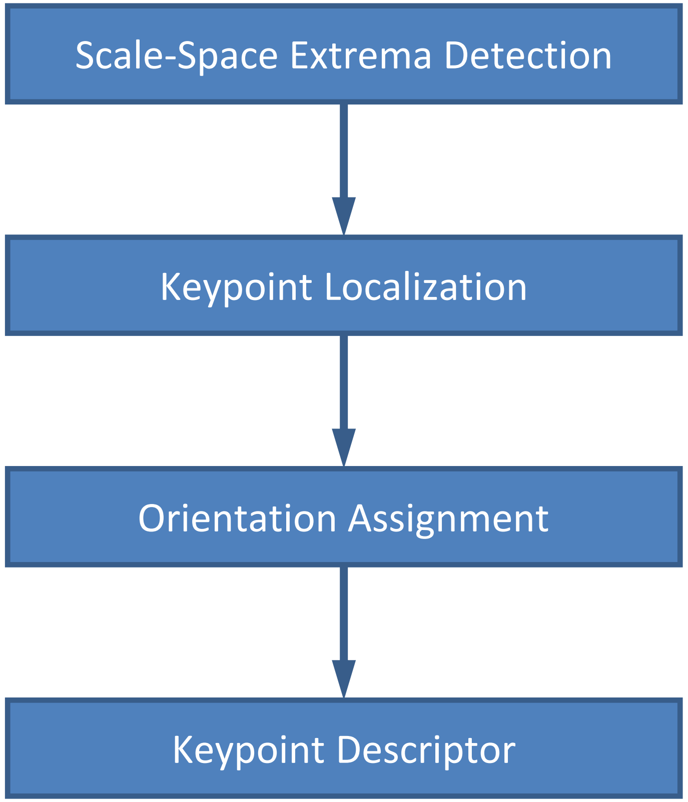
Scale-invariant feature transform

- SIFT feature describes texture in a localised region around a **keypoint**
- SIFT descriptor is **invariant** to various transformations

Recognising that this is the same object requires invariance to scaling, rotation, and shift, and robustness against affine distortion, illumination changes, …

https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/

SIFT algorithm overview



Find maxima/minima in DoG images across scales

Discard low-contrast keypoints and eliminate edge responses

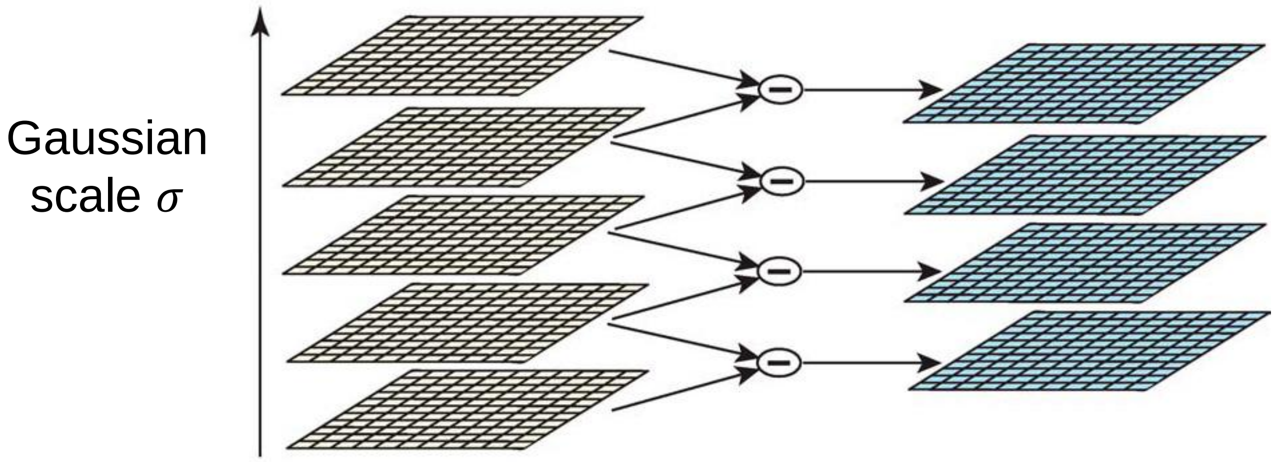
Achieve rotation invariance by orientation assignment

Compute gradient orientation histograms

D. G. Lowe (2004). *Distinctive image features from scale-invariant keypoints*. International Journal of Computer Vision 60(2):91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

SIFT extrema detection

- Detect maxima and minima in the scale space of the image



$L(x, y, \sigma) = I(x, y) * G(x, y, \sigma)$ $\quad\quad$ $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$

(Fixed factor k between adjacent scales)

SIFT keypoint localization

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

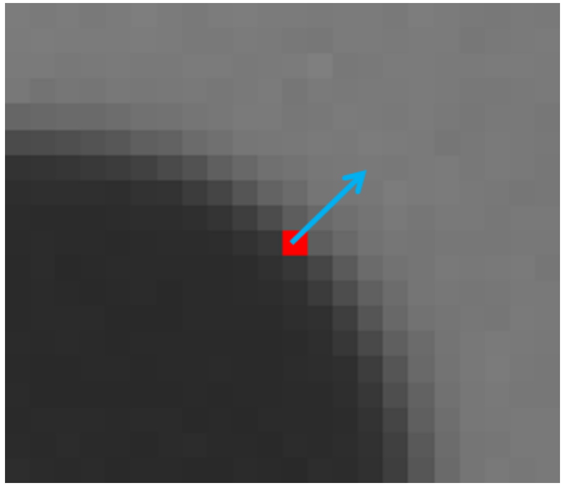
Initial keypoints from scale-space extrema

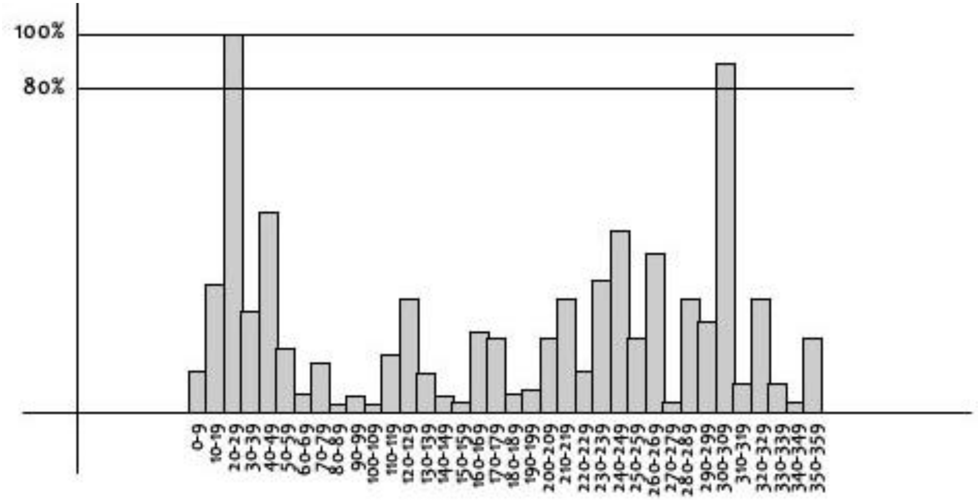
Keypoints after rejecting low-contrast points

Final keypoints after rejecting edge points

SIFT orientation assignment

- Estimate keypoint orientation using local gradient vectors
  - Make an orientation histogram of local gradient vectors
  - Find the dominant orientation from the main peak of the histogram
  - Create additional keypoint for second highest peak if >80%



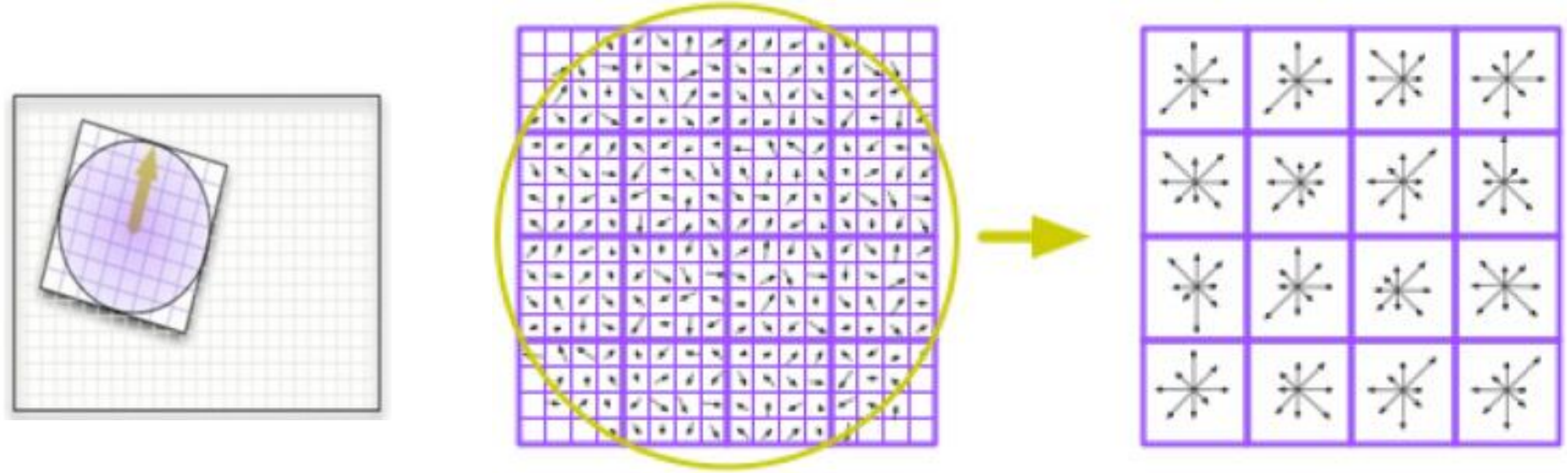


$$M(x,y) = \sqrt{G_x^2 + G_y^2}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

SIFT keypoint descriptor

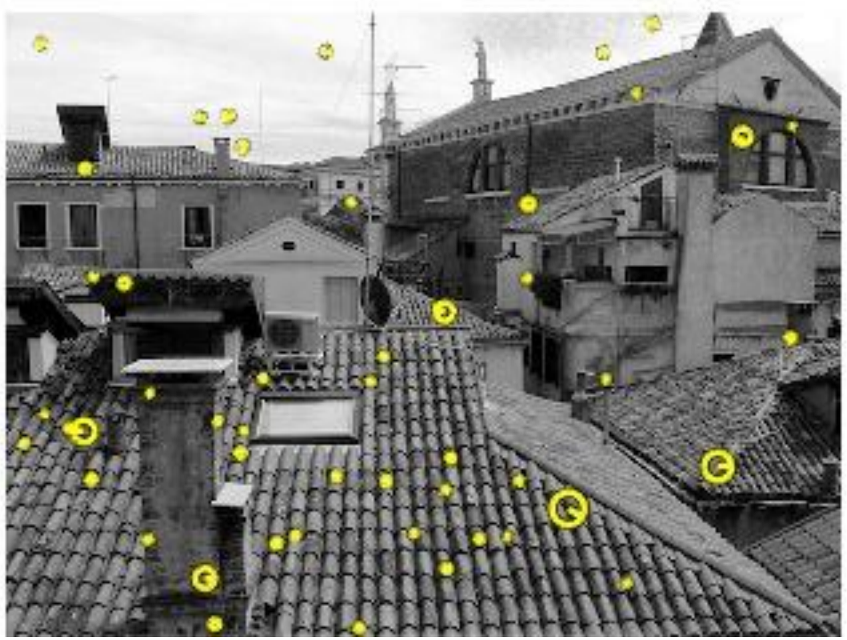
- Represent each keypoint by a 128D feature vector
  - 4 x 4 array of gradient histogram weighted by magnitude
  - 8 bins in gradient orientation histogram
  - Total 8 x 4 x 4 array = 128 dimensions



https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

Scale-invariant feature transform

- SIFT feature describes texture in a localised region around a **keypoint**
- SIFT descriptor is **invariant** to various transformations



Recognising that this is the same object requires invariance to scaling, rotation, and shift, and robustness against affine distortion, illumination changes, …

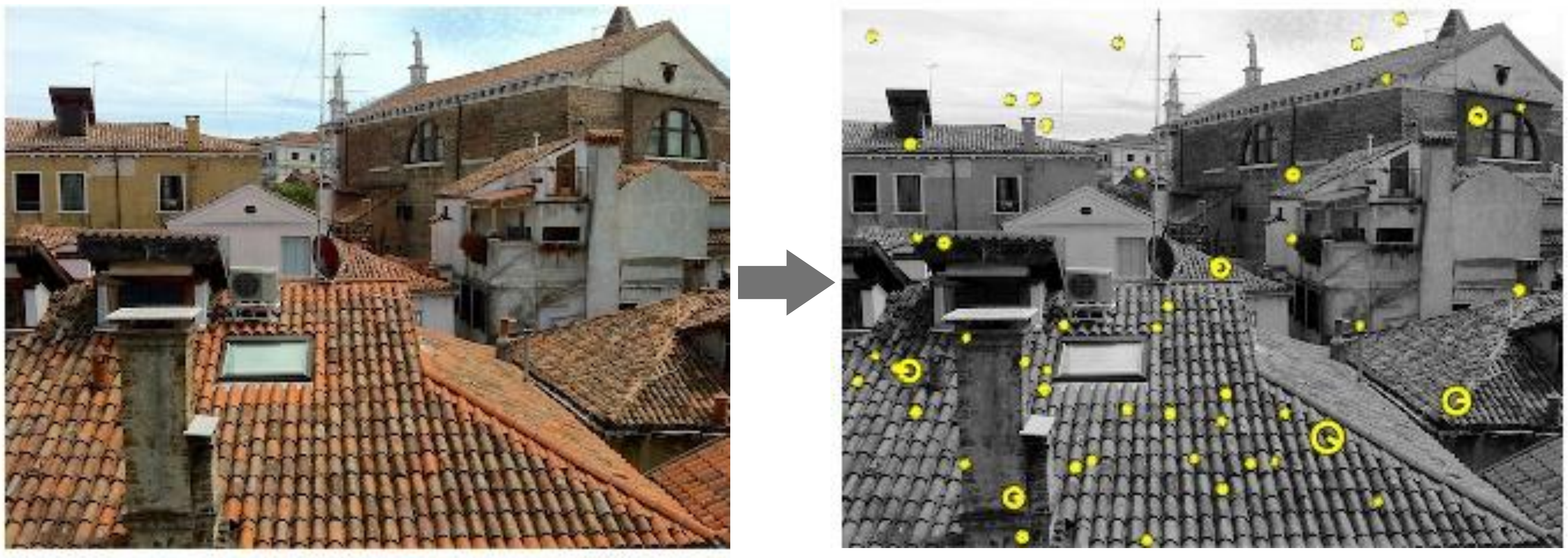
https://ics.uci.edu/~majumder/VC/211HW3/vlfeat/doc/overview/sift.html

Example application of SIFT

- Matching two partially overlapping images

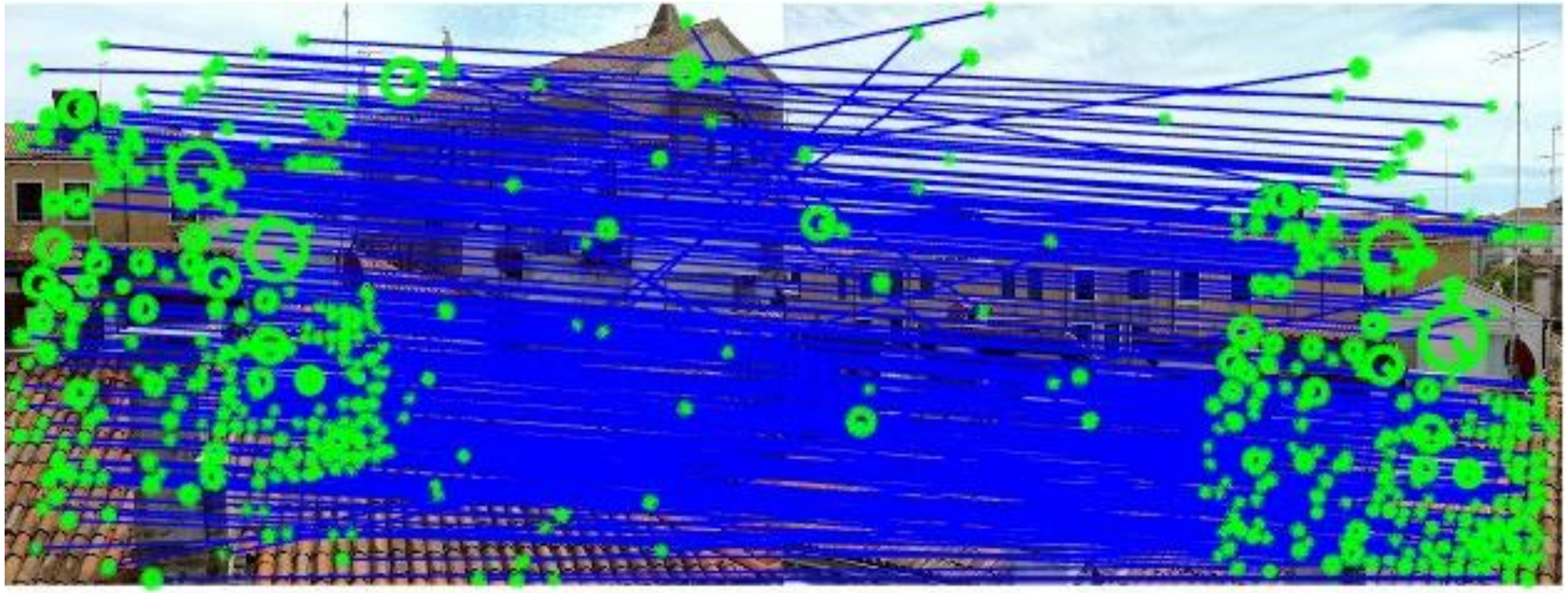
Example application of SIFT

- Matching two partially overlapping images
  - Compute SIFT keypoints for each image



Example application of SIFT

- Matching two partially overlapping images
  - Find best match between SIFT keypoints in 128D feature space

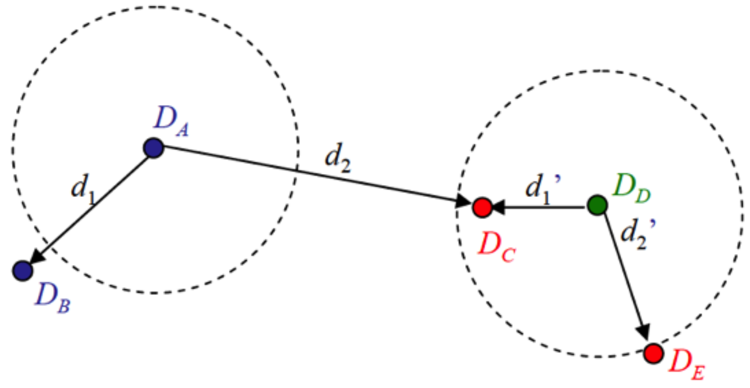


Descriptor matching

- Using the nearest neighbour distance ratio (NNDR)

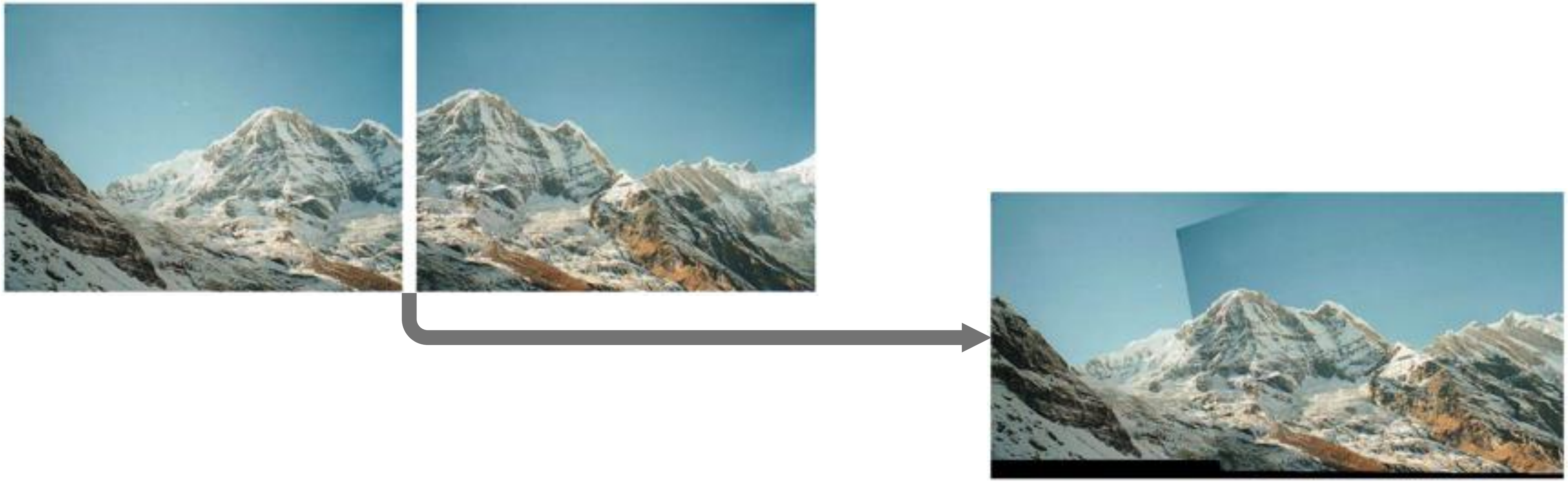
$$\mathrm{NNDR} = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|}$$

  - Distance d_1 is to the first nearest neighbour
  - Distance d_2 is to the second nearest neighbour
  - Nearest neighbours in 128D feature space
  - Reject matches with NNDR > 0.8



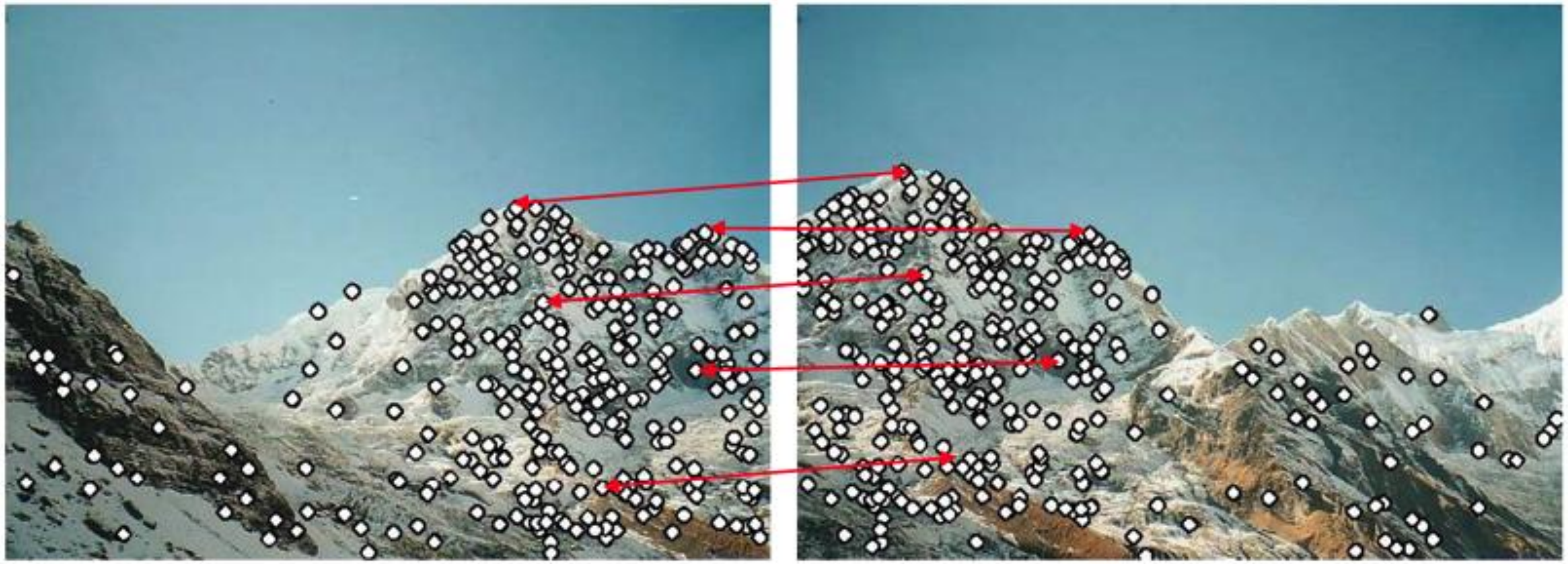
Example application of SIFT

- Stitching two partially overlapping images



Example application of SIFT

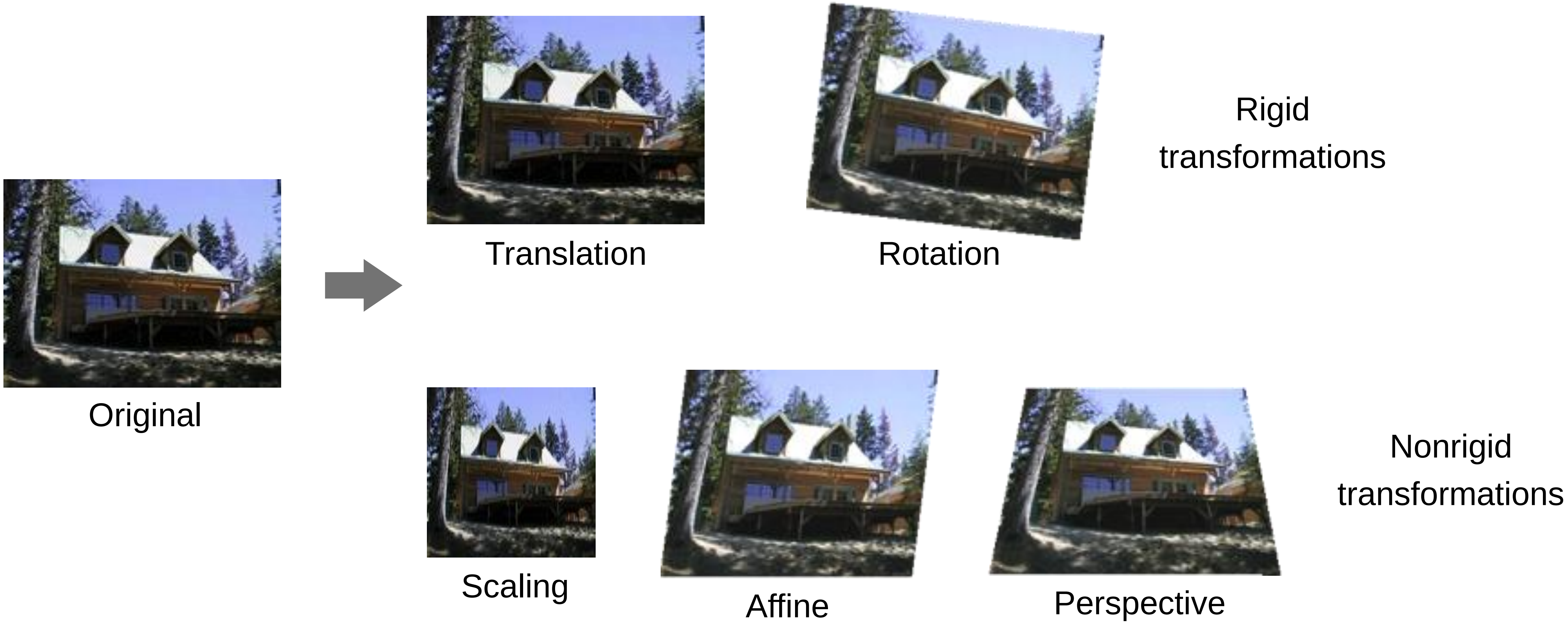
- Stitching two partially overlapping images
  - Find SIFT keypoints and feature correspondences



Example application of SIFT

- Stitching two partially overlapping images
  - Find the right spatial transformation

Types of spatial transformation



Spatial coordinate transformation

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

Shear: $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & r_x \\ r_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate: $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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

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

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

Fitting and alignment

- Least-squares (LS) fitting of corresponding keypoints $(\mathbf{x}_i, \mathbf{x}_i')$
  - Find the parameters $\mathbf{p}$ of the transformation T that minimize the squared error E

$$E = \sum_i \|T(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}_i'\|^2$$

  - Example for affine transformation of coordinates $\mathbf{x}_i = (x_i, y_i)$ into $\mathbf{x}_i' = (x_i', y_i')$:

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

$$\Downarrow$$

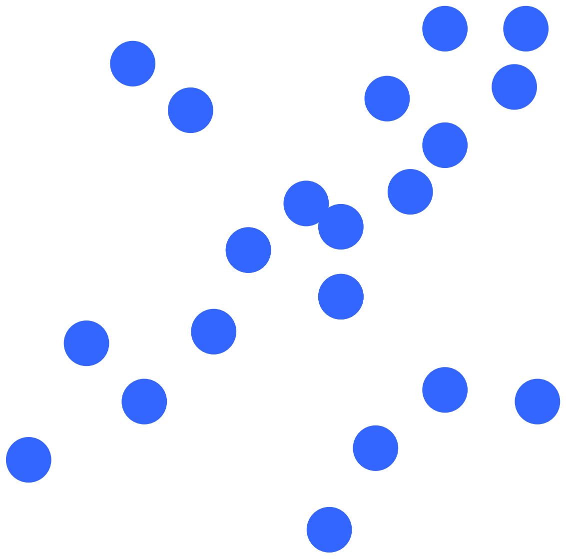
$$\mathbf{p} = [\mathbf{A}^T\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{b} \quad \Leftarrow \quad \mathbf{A}\mathbf{p} = \mathbf{b}$$

Fitting and alignment

- 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 an iterative way to find the optimum

Fitting and alignment

- RANSAC example (line fitting model)

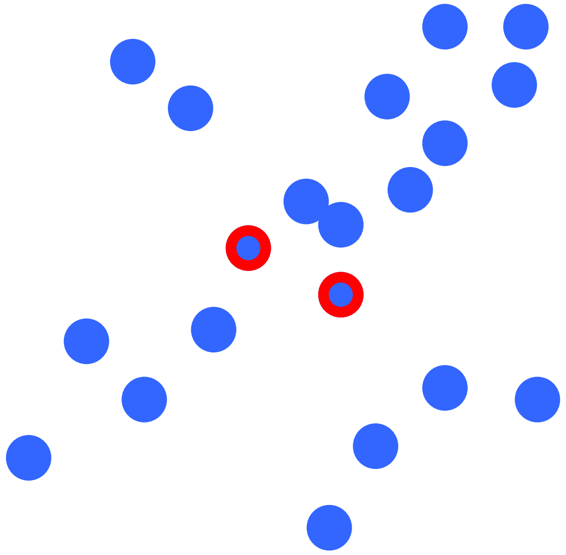


1. Sample (randomly) the number of points required to fit the model
2. Solve for the model parameters using the samples
3. Score by the fraction of inliers within a preset threshold of the model

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

Fitting and alignment

- RANSAC example (line fitting model)



1. Sample (randomly) the number of points required to fit the model

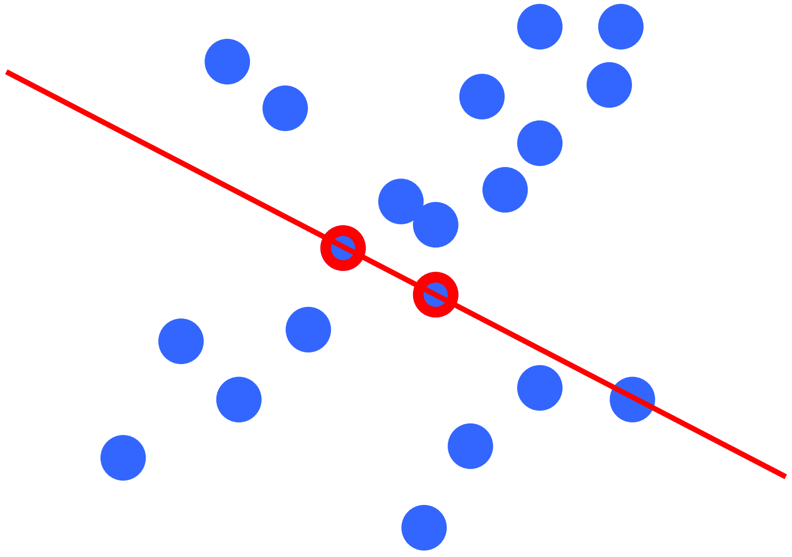
2. Solve for the model parameters using the samples

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 example (line fitting model)

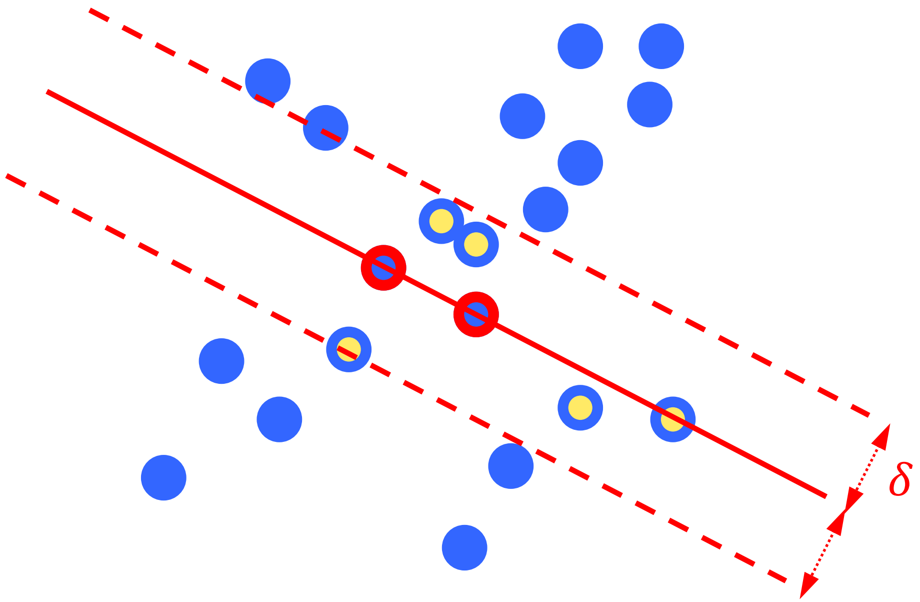


1. Sample (randomly) the number of points required to fit the model
2. Solve for the model parameters using the samples
3. Score by the fraction of inliers within a preset threshold of the model

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

Fitting and alignment

- RANSAC example (line fitting model)

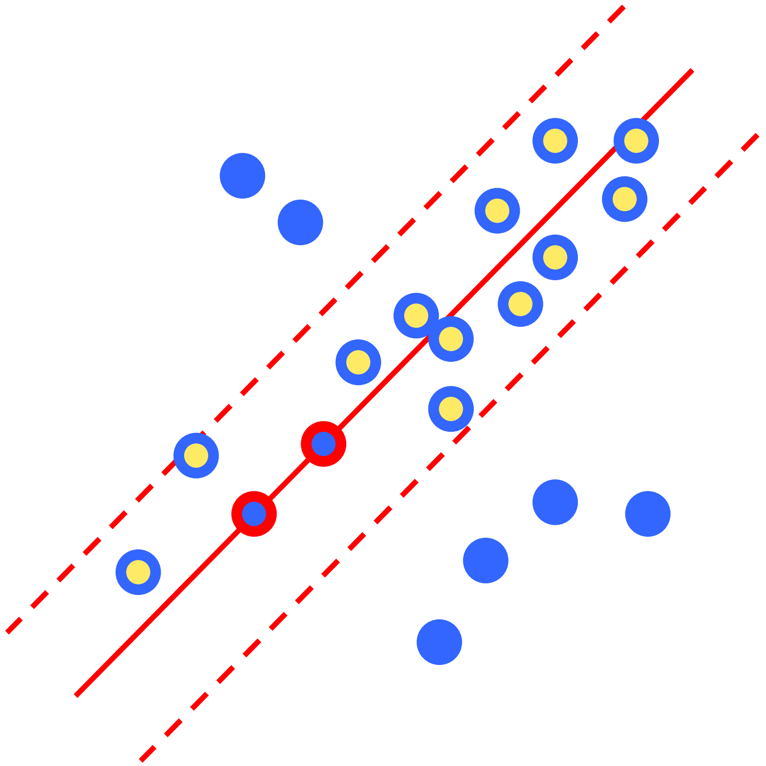


1. Sample (randomly) the number of points required to fit the model
2. Solve for the model parameters using the samples
3. Score by the fraction of inliers within a preset threshold of the model

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

Fitting and alignment

- RANSAC example (line fitting model)



1. Sample (randomly) the number of points required to fit the model

2. Solve for the model parameters using the samples

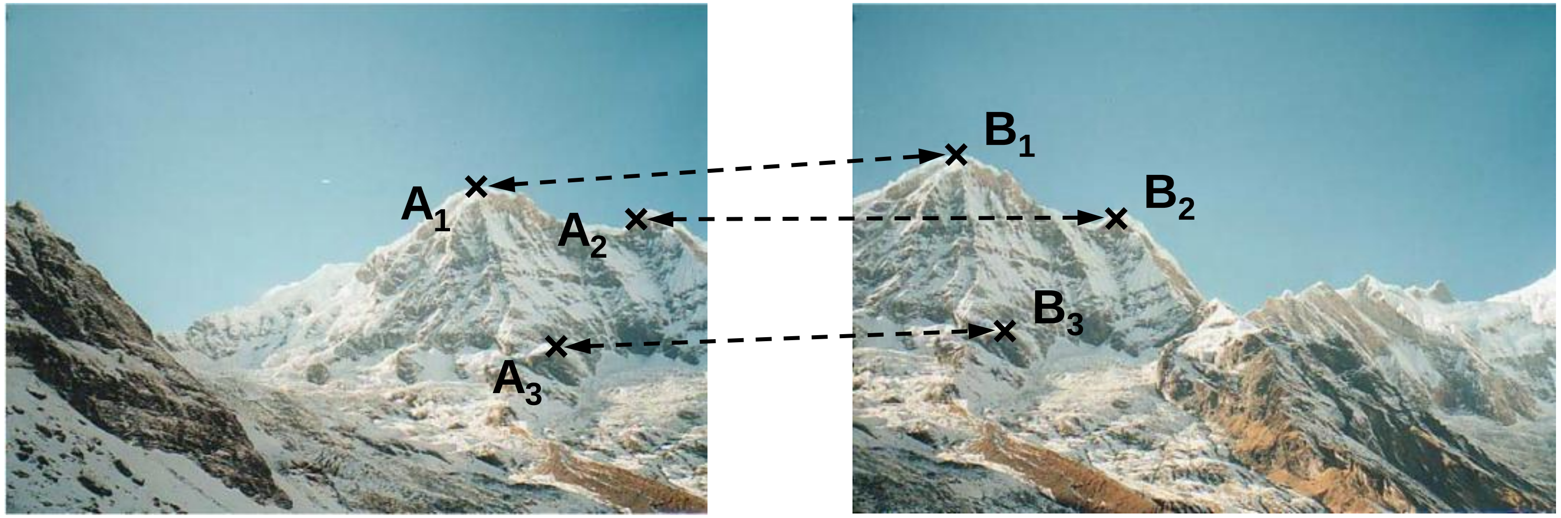
3. Score by the fraction of inliers within a preset threshold of the model

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

Fitting and alignment summary

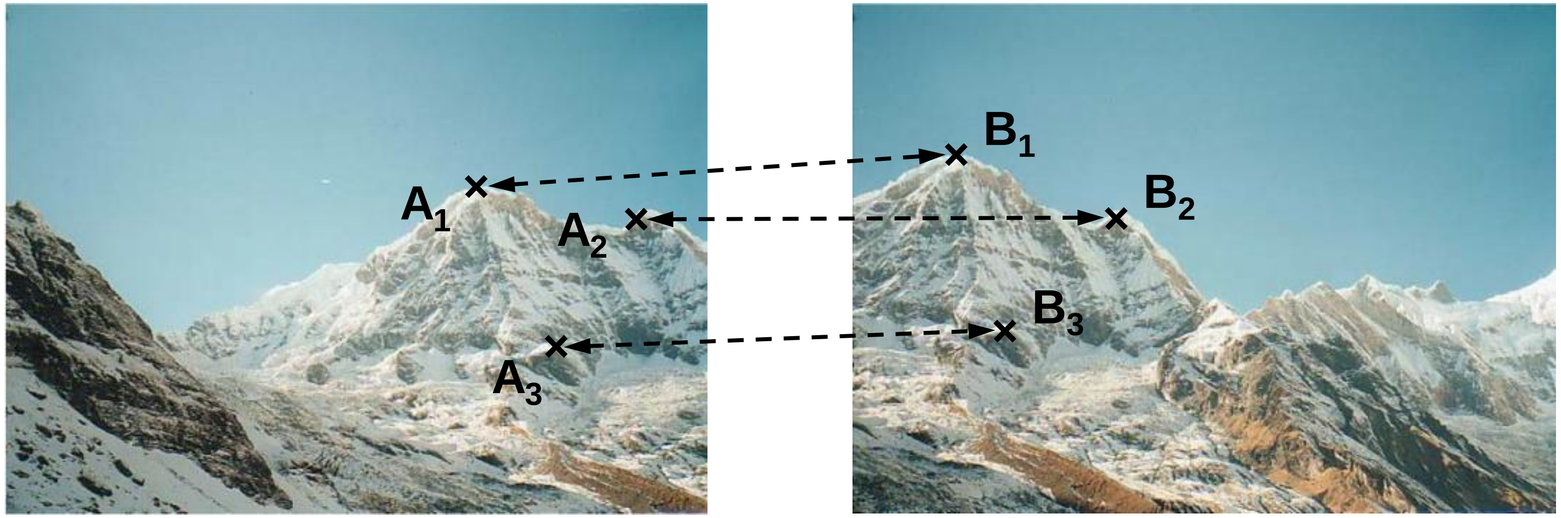
- Estimate the transformation given matched points A and B
  - Example for translation:

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



Alignment by least squares

- Estimate the transformation given matched points A and B
  - Write down the system of equations: $\mathbf{Ap} = \mathbf{b}$
  - Solve for the parameters: $\mathbf{p} = [\mathbf{A}^T\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{b}$



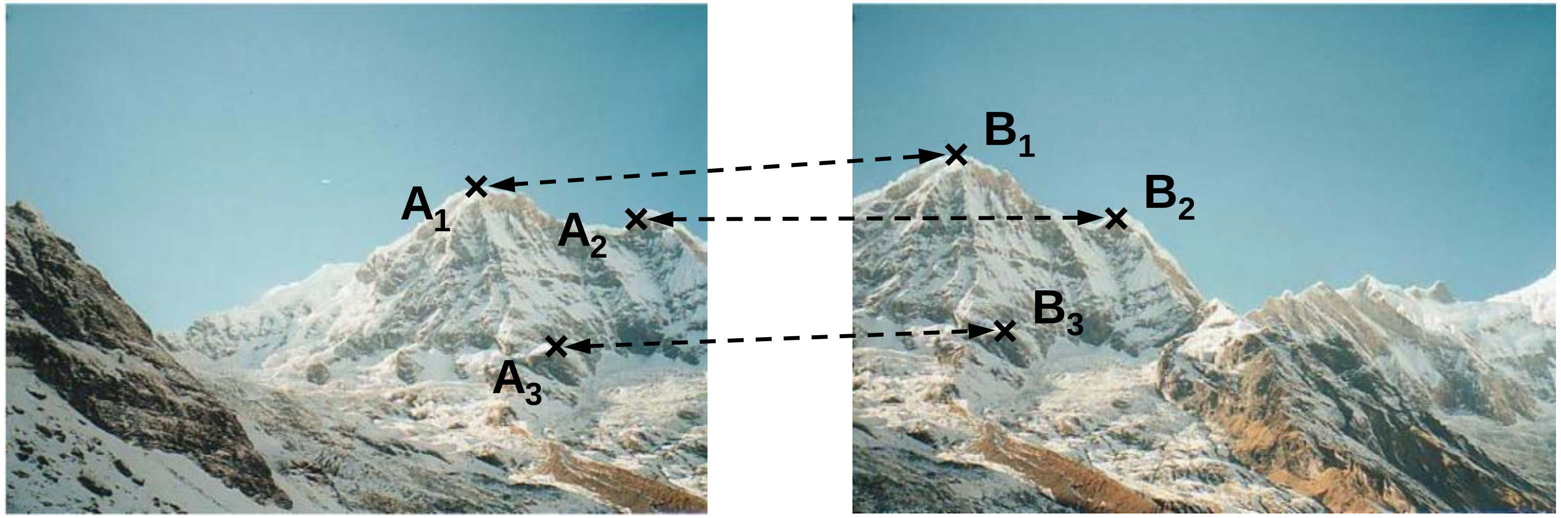
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ x_2^B - x_2^A \\ y_2^B - y_2^A \\ x_3^B - x_3^A \\ y_3^B - y_3^A \end{bmatrix}$$

Alignment by random sample consensus

- Estimate the transformation given matched points A and B

1. Sample a set of matching points (one pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers

Repeat N times



Summary

- Feature representation is essential in solving many computer vision problems
- Most commonly used image features:
  - Colour features (Part 1)

    Colour moments and histogram
  - Texture features (Part 1)

    Haralick, LBP, SIFT
  - Shape features (Part 2)

    Basic, shape context, HOG

Summary

- Other techniques discussed (Part 1)
  - Descriptor matching
  - Least squares and RANSAC
  - Spatial transformations
- Techniques to be discussed (Part 2)
  - Feature encoding (Bag-of-Words)
  - K-means clustering
  - Shape matching
  - Sliding window detection

Further reading on discussed topics

- Chapters 4 and 6 of Szeliski

Acknowledgements

- Some content from slides of James Hays, Michael A. Wirth, Cordelia Schmit
- From BoW to CNN: Two decades of texture representation for texture classification
- And other resources as indicated by the hyperlinks

Example exam question

Which one of the following statements about feature descriptors is incorrect?

A. Haralick features are derived from gray-level co-occurrence matrices.

B. SIFT achieves rotation invariance by computing gradient histograms at multiple scales.

C. LBP describes local image texture and can be multiresolution and rotation-invariant.

D. Colour moments have lower representation capability than the colour histogram.