Computational Vision

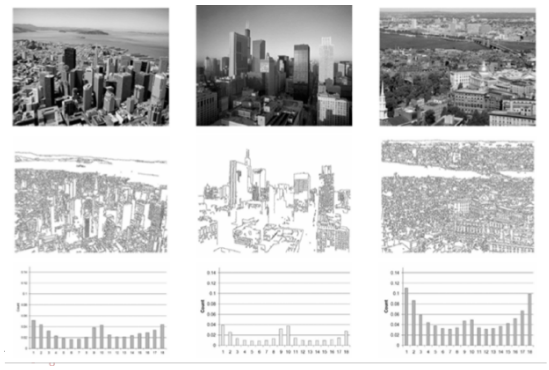
Riccardo Caprile

March 2022

1 Image Processing

1.1 Describing the the whole image

Histograms can be used as a way of obtaining an overall description of the image content. It is easy to compute , robust to geometrical variations and to scale changes. But you loose spatial information such as pixels' position , i know the quantity of a value but i don't know where it is.



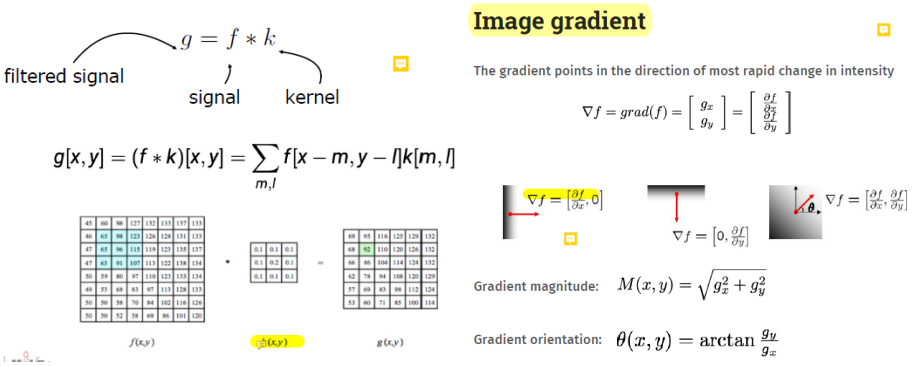
1.2 Image Filters

A basic tool in image processing used for a variety of tasks , included noise reduction and signal enhancement.

We will consider linear filters in the space domain.

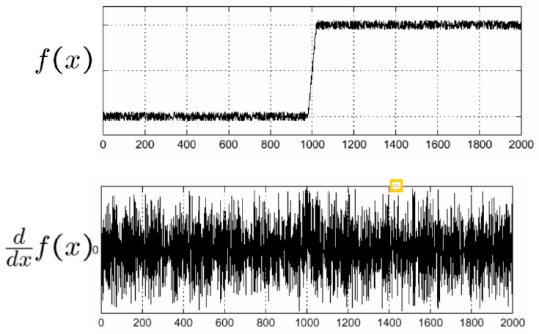
1.2.1 Linear filtering in space - Convolution

In linear cases , filtering corresponds to applying to a signal f a convolution operator.



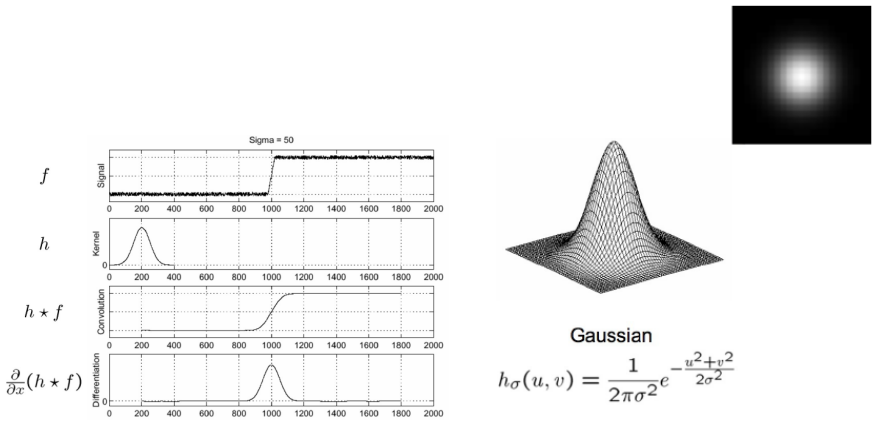
In the first example , along y i should be see 0 whereas along x some changes. It is the opposite for the example after. There is a change in intensity level only in the horizontal direction in the first case and no change in the y direction. Gradient magnitude represents the strength of the change in the intensity level of the image. Higher Gradient magnitude , stronger the change in the image intensity.

1.2.2 The effect noise



With the noisy signal derivative , it clearly loose the significant change of frequency that we have in the original signal.

1.2.3 Smoothing



Using directly the derivative of the Kernel will save us one operation.

1.3 Local features : edges

Edges are pixels at which the image values undergo a sharp variation.

Edge detection : given an image locate edges most likely to be generated by scene elements and not by noise.

1.3.1 Edges or objects boundaries?

Edges and boundaries are two different concepts (texture of the animal is not a boundary but could be an edge).

1.3.2 Corners : Good features to match

We observe how patches with gradients in at least two different orientations are the easier to localize. This can be formalized by analysing a simple matching criterium.

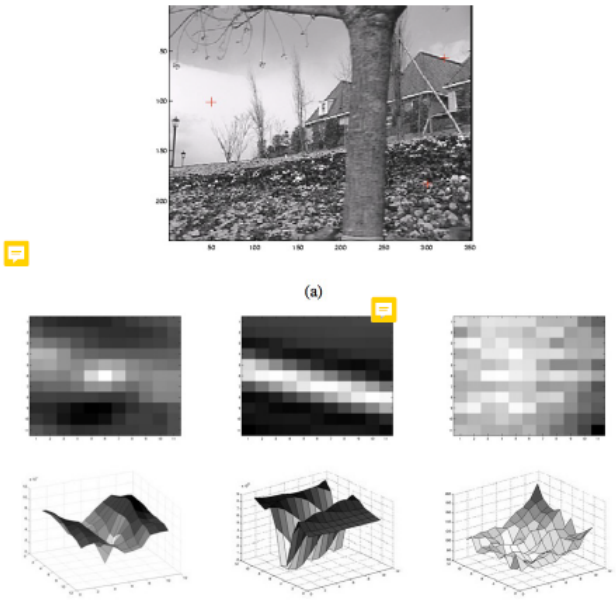
Take a portion of an image , and i look to the similarities between that portion and its shifted version. $I(x_i)$ is the portion of the image centered in that point.

Low autocorrelation : the two portion are similar

The **gradient** can be defined as the change in the direction of the intensity level of an image.

These intutitions can be formalized by looking at the simplest possible matching criterion for comparing two image patches (their summed sqaure difference).

When performing feature detection , we do not know which other image locations the feature will end up being matched against. Therefore , we can only compute how stable this metric is with respect to small variations in position Delta u , comparing an image patch against itself



It is interesting to see in the image c lower values are near the edge and higher when we move away from it. In the image b , there is a strong minimum indicating that it can be well localized.

1.4 Local features : corners

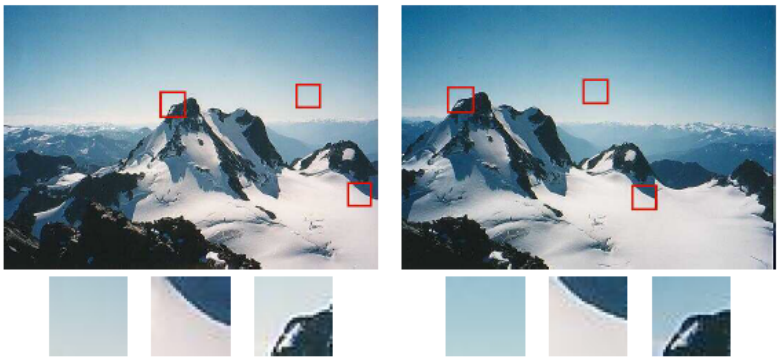


Figure 7.3 *Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.*

How can we find image locations where we can reliably find correspondences with other images? As you may notice , textureless patches are nearly impossible to localize.Patches with a large contrast changes (gradients) are easier to localize. Patches with gradients in at least two significantly different orientations are the easiest to localize.

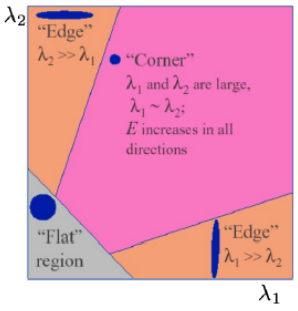
Corners correspond to points where the image gradient varies in at least two directions.

They can be detected by computing and analyzing the autocorrleation matrix A of a patch around each point. (Given an image point q we consider a neighborhood Nq and compute its autocorrelation matrix) Autocorrelation consider a patch and the entire image.

$$\nabla I = [I_x, I_y]^\top$$

$$A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

$$eig(A) = [\lambda_2, \lambda_1] \lambda_2 > \lambda_1$$



2 Image Matching

2.1 Introduction

The concept of estimating the similarity between image pairs is very broad and is usually guided by the application.

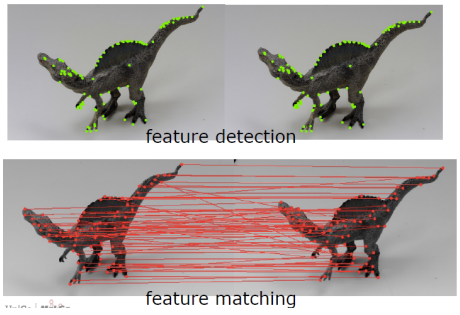
Similarity is usually estimated with an image matching process where we try to match one image (or parts of it) with another.

- **Global matching** : Compute a global descriptor for each image (eg, brightness/color mean and variance , color histograms) and estimate the similarity between descriptors.
- **Local matching** : Extract local features and solve a feature correspondence problem

2.2 Local matching as a correspondence problem

It involves two decisions :

1. Which image elements to match (all the pixels from an image (**dense correspondences**) , or a subset of pixels meeting some requirements (**sparse correspondences**)
2. Which feature description + similarity to adopt

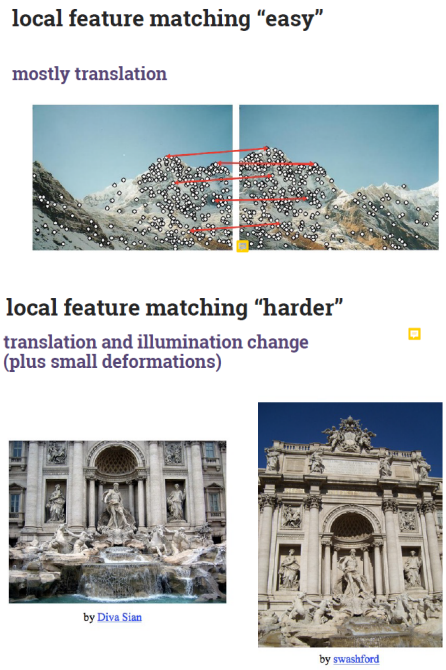


2.3 Our goal : local matching

Match elements in different images. Objects may appear at different scales , rotation or viewpoint.

Pipeline :

1. **Feature Detection** : Find interest points on each image
2. **Feature Description** : Compute a vector description for each point



3. **Feature Matching**

We simply obtained the image by translating the camera.

Images taken from different devices and part of the day , because of the light we need some kind of normalization.

2.4 Local Feature Matching

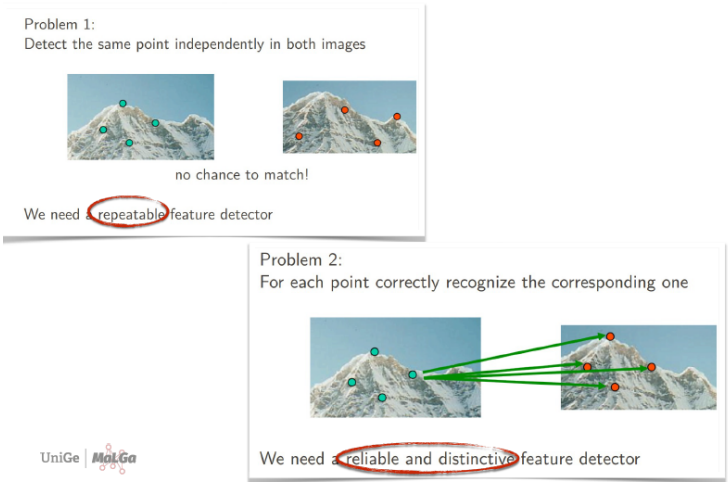
If image pairs which are similar enough , then local features undergo a quasi translation transformation.

Interest points may be **corners**.

Image patches are an appropriate feature description.

In the case of scale , rotation and more severe viewpoint changes we may need **scale invariant** interest points and better feature descriptors.

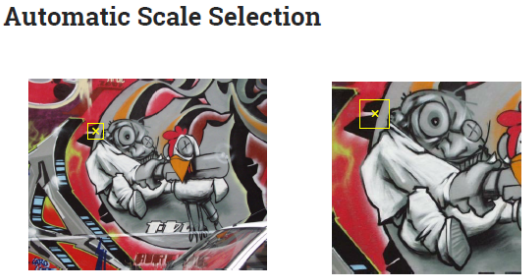
2.4.1 Local Feature Matching : Detection Problems



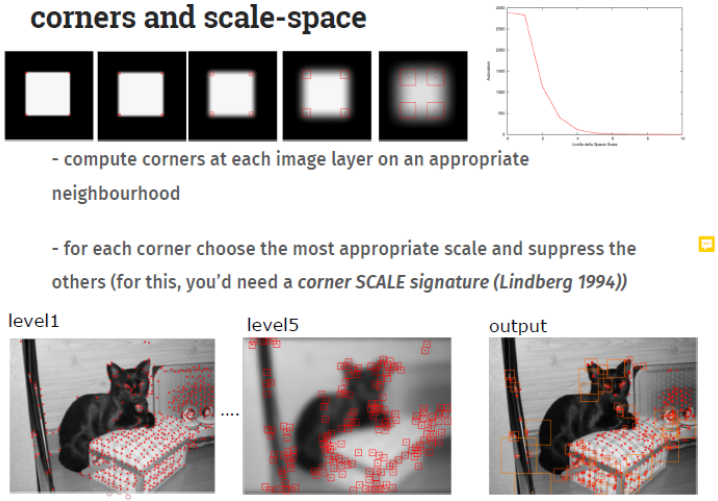
2.5 Scale invariant Feature detectors

In many situations , detecting features at the finest stable scale possible may not be appropriate. For example , when matching images with little high-frequency detail such as clouds. One solution to the problem is to extract features at a variety of scales , for example by performing the same operations at multiple resolutions in a pyramid and then matching features at the same level.

In the scaled image we need a larger square because we need the same amount of information.

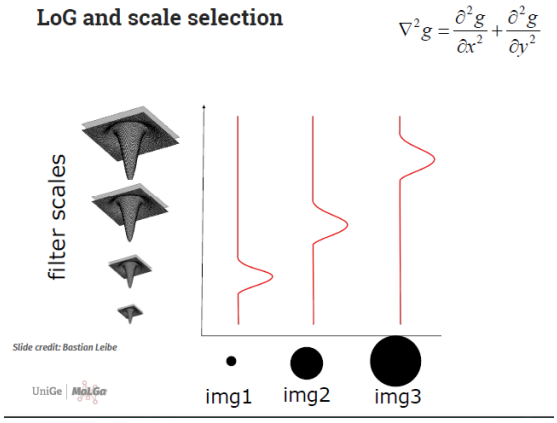


In the next slide level5 is a the image filtered 5 times. In the first image we want to take a point in a very small neighbourhood. Going on with the other filtered images we want to have more general details and similarities. Standard deviation of the Gaussian grows and so grows the number of points in the neighbourhod we are considering.



However , for most object recognition applications , the scale of the object in the image is unknown. Instead of extracting features at many different scales and then matching all of them , it is more efficient to extract features that are stable in both location and scale.

Lindeberg proposed using extrema (max and min) in the Laplacian of Gaussian (LoG) function as interest point locations. It is the filter used before that allows you to implement blob detection.

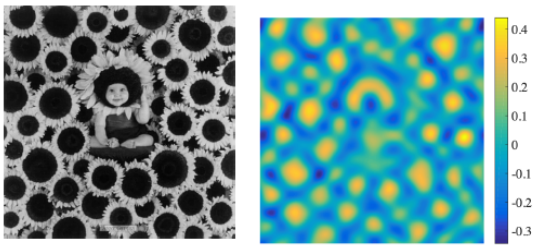


2.6 Blob-like features

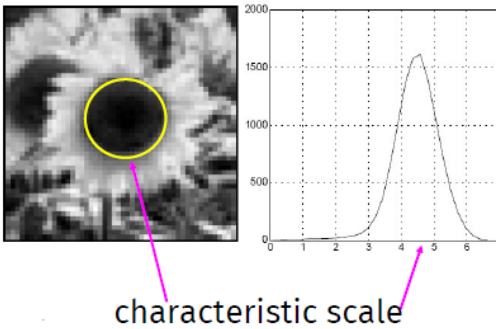
Another type of local key point quite naturally associated with the concept of scale.

Blobs : uniform areas surrounded by a sharp variation of the signal.

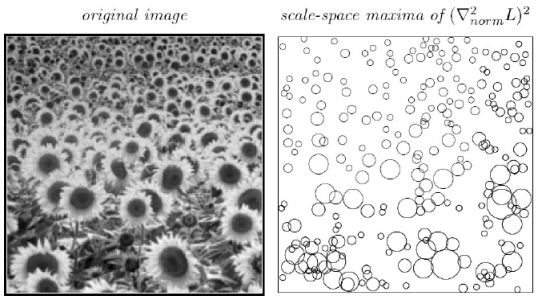
Blob enhancement may be carried out by looking for extrema of the image Laplacian.



We define the **characteristics scale** as the scale that produces peak of Laplacian response.



2.6.1 Scale-space blob detector : Example



2.7 SIFT Explanation in Details

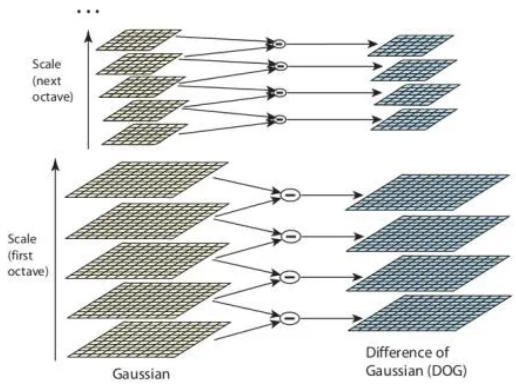
2.7.1 Scale-Space Peak Selection

Scale space : Real world objects are meaningful only at a certain scale. You might see a sugar cube perfectly on a table. But if looking at the entire milky way, then it simply does not exist. This multi-scale nature of objects is quite common in nature. And a scale space attempts to replicate this concept on digital images.

The scale space of an image is a function $L(x, y, \sigma)$ that is produced from the convolution of a Gaussian kernel(Blurring) at different scales with the input image. Scale-space is separated into octaves and the number of octaves and scale depends on the size of the original image. So we generate several octaves of the original image. Each octave's image size is half the previous one. G is the Gaussian Blur operator and I is an image. While x,y are the location coordinates and σ is the "scale" parameter. Think of it as the amount of blur. Greater the value, greater the blur.

2.7.2 DoG , Difference of Gaussian kernel

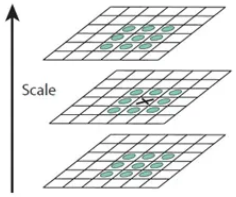
Now we use those blurred images to generate another set of images, the Difference of Gaussians (DoG). These DoG images are great for finding out interesting keypoint in the image. The difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different σ, let it be σ and σ. This process is done for different octaves of the image in the Gaussian Pyramid. It is represented in below image :



2.7.3 Finding keypoints

Up till now, we have generated a scale space and used the scale space to calculate the Difference of Gaussians. Those are then used to calculate Laplacian of Gaussian approximations that are scale invariant. One pixel in an image is compared with its 8 neighbors as well as 9 pixels in the next scale and 9 pixels

in previous scales. This way, a total of 26 checks are made. If it is a local extrema, it is a potential keypoint. It basically means that keypoint is best represented in that scale.



2.7.4 Keypoint localization

Key0points generated in the previous step produce a lot of keypoints. Some of them lie along an edge, or they don't have enough contrast. In both cases, they are not as useful as features. So we get rid of them. The approach is similar to the one used in the Harris Corner Detector for removing edge features. For low contrast features, we simply check their intensities.

2.7.5 Orientation Assignment

Now we have legitimate keypoints. They've been tested to be stable. We already know the scale at which the keypoint was detected (it's the same as the scale of the blurred image). So we have scale invariance. The next thing is to assign an orientation to each keypoint to make it rotation invariance. A neighborhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created. Let's say the gradient direction at a certain point (in the "orientation collection region") is 18.759 degrees, then it will go into the 10–19-degree bin. And the "amount" that is added to the bin is proportional to the magnitude of the gradient at that point. Once you've done this for all pixels around the keypoint, the histogram will have a peak at some point.The highest peak in the histogram is taken and any peak above 80percent of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions. It contributes to the stability of matching.

2.7.6 Keypoint descriptor

At this point, each keypoint has a location, scale, orientation. Next is to compute a descriptor for the local image region about each keypoint that is highly distinctive and invariant as possible to variations such as changes in viewpoint and illumination.

To do this, a 16x16 window around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size.

For each sub-block, 8 bin orientation histogram is created.

So 4 X 4 descriptors over 16 X 16 sample array were used in practice. 4 X 4 X 8 directions give 128 bin values. It is represented as a feature vector to form keypoint descriptor. This feature vector introduces a few complications. We need to get rid of them before finalizing the fingerprint.

2.7.7 Keypoint Matching

Keypoints between two images are matched by identifying their nearest neighbors. But in some cases, the second closest-match may be very near to the first. It may happen due to noise or some other reasons. In that case, the ratio of closest-distance to second-closest distance is taken. If it is greater than 0.8, they are rejected. It eliminates around 90percent of false matches while discards only 5percent correct matches, as per the paper.

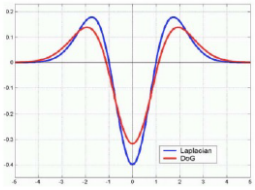
2.8 DoG Features

It can be shown that the Laplacian operator may be approximated by the difference of the image smoothed with two different Gaussian filters

$$\nabla^2 L(x, y; t) \sim c * (L(x, y; t + \Delta t) - L(x, y; t))$$

normalization factor

L(x,y;t) is an image I filtered with a gaussian kernel of variance t

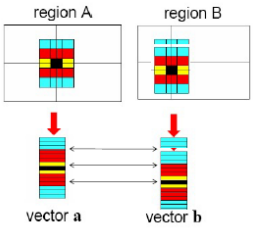


2.9 Feature Descriptors

After detecting keypoint features , we must match them. We must determine which features come from corresponding locations in different images. In some cases the sum of squared difference can be used to directly compare the intensities in small patches around each feature point. Even after compensating for these changes , the local appearance of image patches will usually still vary from image to image. How can we make image descriptors more invariant to such changes, while still preserving disciminability between different patches?

2.9.1 Raw patches

The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a vector . But this is very sensitive to even small shifts , rotations.



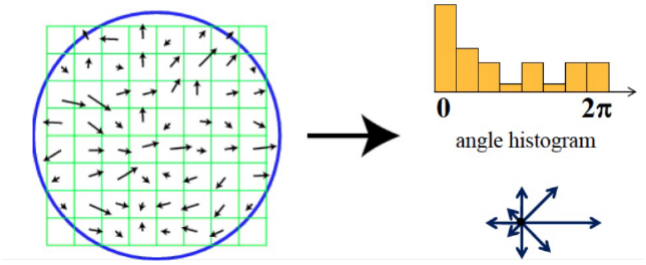
2.9.2 Invariant Feature Descriptors : Scale Invariant Feature Transform SIFT

The basic idea is to take a 16x16 square window around detected interest point (in the case of this image is 8x8).

Compute edge orientation (angle of the gradient minus 90) for each pixel.

Throw out weak edges (threshold gradient magnitude).

Create histogram of surviving edge orientation.



2.9.3 Invariant Feature Descriptors : Multiscale Oriented PatcheS Descriptors MOPS

Take a 40x40 square window around detected feature.

Scale to 1/5 size to get 8x8 square window.

Rotate to horizontal.

Normalize the window values by subtracting the mean and dividing bt the standard deviation in the window.

2.10 SIFT by David Lowe

The idea is to detect interesting points such as corners or blobs , on a multi-scale image representation , and then associate with them a vectorial description which is invariant to translation , scale and rotation changes.

This idea is very robust , fast , efficient and tolerant to illumination changes.

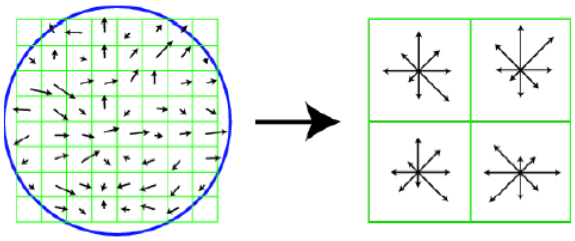
2.11 DoG detection and SIFT description

Main steps :

1. Scale-space image construction
2. Scale-space extrema detection (DoG keypoints)
3. Accurate key point location (sub-pixels analysis , remove low contrast and edges)
4. Keypoint orientation assignment
5. Keypoint descriptor

This is a schematic representation of Lowe SIFT. (a) Gradient orientations and magnitude are computed at each pixel and weighted by a Gaussian window (blue circle). (b) A weighted gradient orientation histogram is then computed in each subregion. The highest peak in the histogram gives the main orientation of the feature. 8 bins for the 8 different directions in the keypoint descriptor

Each local feature is represented collecting the information of the gradient around its location , considering a patch of size corresponding to its scale and rotated according to its orientation



(a) image gradients (b) keypoint descriptor

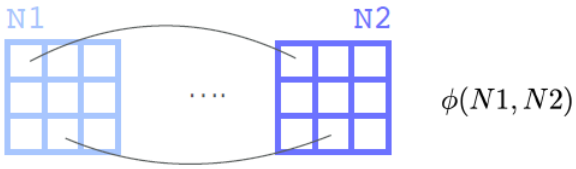
2.12 Feature Matching

Once we have extracted features and their descriptors from two or more images, the next step is to establish some preliminary feature matches between these images. Determining which feature matches are reasonable to process further depends on the context in which the matching is being performed. Say we are

given two images that overlap to a fair amount (e.g., for image stitching or for tracking objects in a video). We know that most features in one image are likely to match the other image, although some may not match because they are occluded or their appearance has changed too much.

2.12.1 Similarity measures for image patches

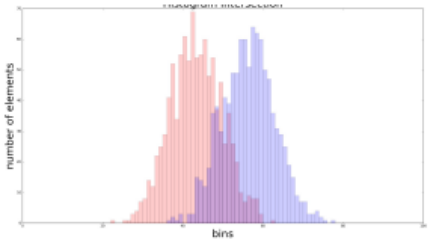
Let N1 and N2 be two square image patches of size WxW.



How to measure similarity? Using Sum of squared differences or Normalized cross correlation.

2.12.2 Similarity measures for histograms

Given a Euclidean distance metric , the simplest matching strategy is to set a threshold (maximum distance) and to return all matches form other images within this threshold. Setting the threshold too high results in too many false positive (incorrect matches being returned). Setting the threshold too low results in too many false negatives , too many correct matches being missed. We can evaluate the performance of a matching algorithm at a particular threshold by first counting the number of true and false matches and match failures .



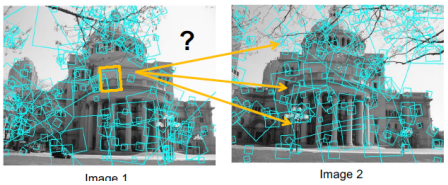
2.13 Matching Strategy

To generate candidate matches , find features with the most familiar appearance. Brute force approach : compare them all , take the closest (or closest k , or within a tresholded distance).

To add robustness to matching , consider ratio : dist to best match / dist to second best match

If low , first match looks good.

If high , could be ambiguous match.

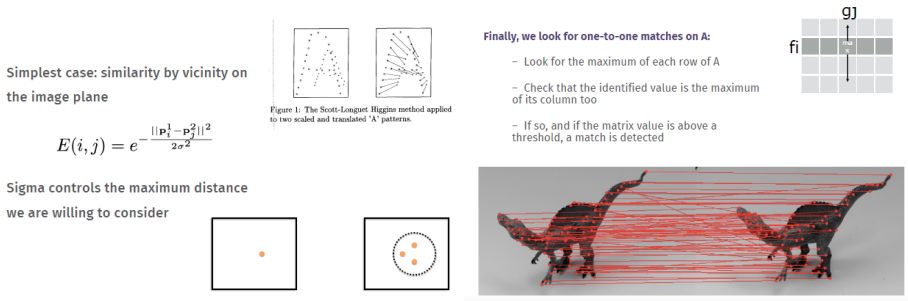


2.13.1 Matching through an affinity matrix

In graph theory , an affinity matrix is similar to an adjacency matrix E but its entries take values $[0, 1]$.

It describes the similarity between graph nodes and between features.

Each entry measures how similar two keypoints are.



3 Motion Analysis

3.1 Introduction

What do we gain if we analyse videos instead of images?

We focus on motion information.

We are observing a scene with one camera acquiring a set of images "close in time".

Image sequence : Series of N images , or frames , acquired at discrete time instants

$$t_k = t_0 + k\Delta t$$ *fixed time interval (small)*

3.1.1 Brightness Constancy assumption

An important assumption to make when analysing image sequences is the following: **Illumination does not vary in the observation interval** , but what does this mean?

As i'm analyzing multiple frames in my video , i'm assuming that something is changing due to motion , not to lightning.

It this case the image changes from t1 to t2 are caused by the relative motion between scene and camera.

3.1.2 The problems of motion

We may identify different types of questions related with motion analysis.

- **Correspondence**: Which elements of a frame correspond to which element of the next frame? We have further knowledge than feature matching because we have more images to use
- **Reconstruction** : What was the 3D position and 3D velocity of the observed elements?
- **Motion segmentation** : What regions of the image plane correspond to different moving parts? In the case the camera is still , motion segmentation is called change detection. A frame is divided , for example , in 4 subregion , and every element of a subregion should move in the same direction of all the element that belong to the same subregion.
- **Tracking** : Estimate the trajectories of points or objects from image sequences

3.2 Motion Segmentation : Change Detection

In this case the camera is stil.

The analysis is pixel based.

Black : elements that are not moving

White : elements that are moving

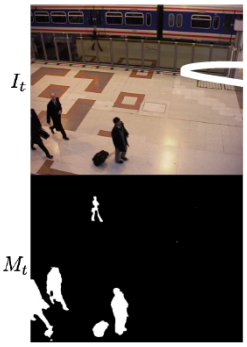
I_t is my image , I_ref is the reference image (image of the empty room) for detecting image changes.

x,y is the position of the pixel i am considering.

Easier example : I_t is a black image with a white circle in the middle , I_ref is just a black image.

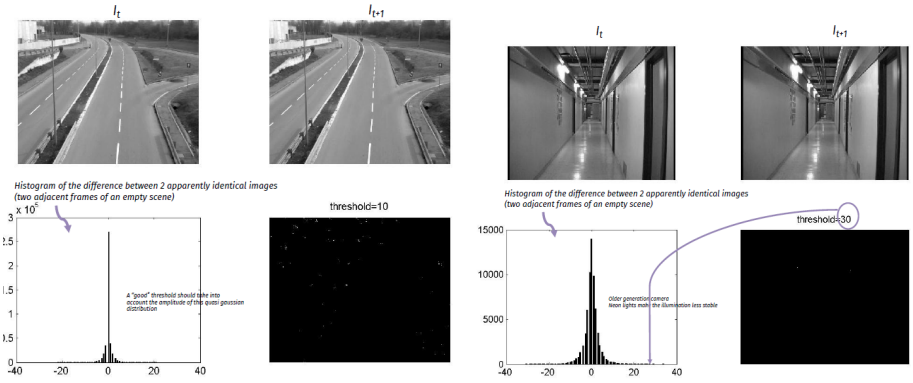
Why do we need absolute value ? We need values that are not minor than 0.

The threshold τ depends on the acquisition device and the acquisition conditions. It must be chosen considering a trade-off between false positives and false negatives (noise)



$$M_t(x,y) = \begin{cases} 1 & \text{if } |I_t(x,y) - I_{ref}(x,y)| > \tau \\ 0 & \text{otherwise} \end{cases}$$

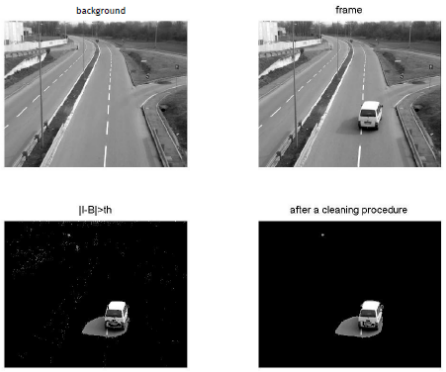
3.2.1 The threshold



3.2.2 The reference image

The simplest way of detecting moving objects is to compare consecutive frames. In general is not a good choice , because it is difficult to detect slow motion . Noisy localization

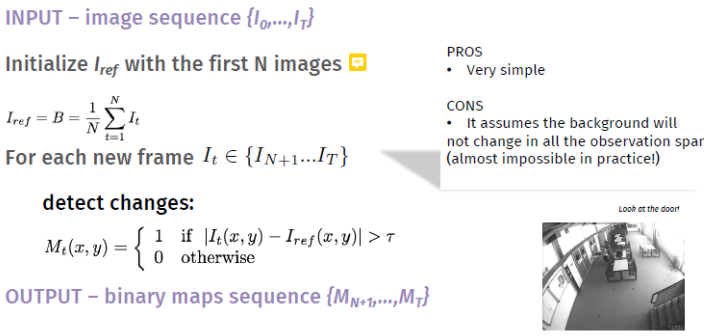
The reference image or , more in general the background model , is a picture of the empty scene , containing all the parts which are not moving



The reference image - Version 1

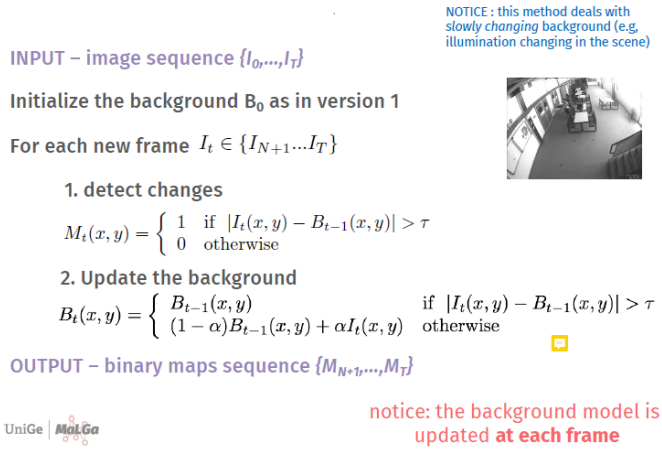
The reference image is a picture of the empty scene , containing all the parts which are not moving.

The simplest way of computing it is to average the first N frames (assuming that at the beginning the scene is stationary)



I am assuming that the first frames are empty.

The reference image - Version 2 (Running Average)



Alpha is a learning factor. What we are doing here , is basically refreshing the background.

This method it is quite robust to moving objects in the scene. It incorporates smooth stable changes and it is simple and efficient.

The problem is that it does not deal with repetitive motion.

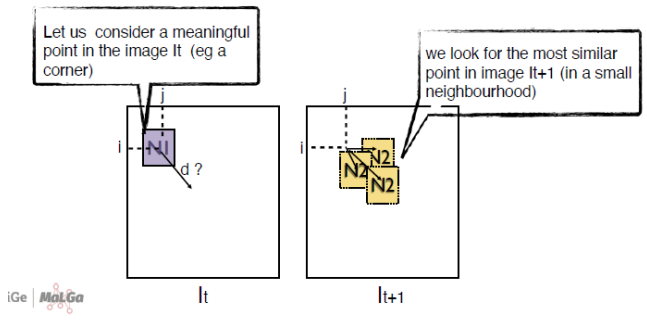
3.3 Correspondence : Optical Flow

Correspondence : Which elements of a frame correspond to which elements of the next frame?

We take advantage of the high similarity between adjacent frames.

I want to understand how things are moving in an image plane.

Correlation-based approaches and Gradient-based approaches



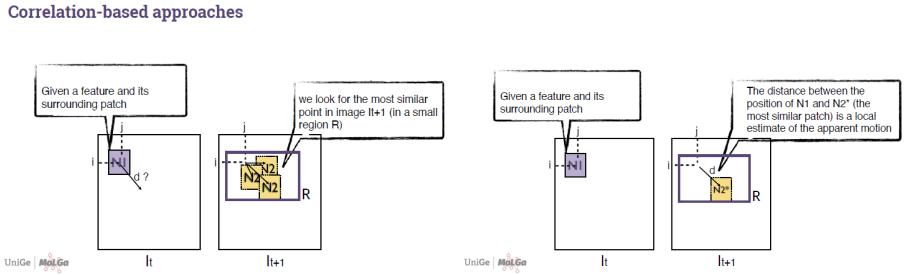
3.3.1 Motion analysis as a correspondence problem

In the case of motion , correspondence may also be seen as a problem of estimating the apparent motion of the brightness pattern (the so called optical flow , which is the estimation of the apparent motion)

Why apparent? Because it is not real motion , but the one we perceive.

Ex1 : a ball bouncing in a dark room , what do we perceive? Nothing

Ex2 : The barber pole



Gradient-based Derivation

It is based on the image brightness constancy equation : from one frame to the next image appearance does not change. We are basing our analysis based on estimating derivatives. For every x,y pixel acquired at time t ,there will be at time t+1 something identical but in a different position.

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

If the motion is small, we can use the following Taylor approximation

$$I(x + u, y + v, t + 1) = I(x, y, t) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial x}v + \frac{\partial I}{\partial t} \quad + H.O$$

We thus obtain $\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$

Or more compactly

$$I_x u + I_y v + I_t = 0$$

$$(\nabla I)^\top \mathbf{u} + I_t = 0$$

The underlined parts are the partial derivatives and our objective is to estimate u and v.

The optical flow is a vector field subject to the constraint above.

This constraint gives us 1 equation per each image point.

3.3.2 Optical Flows Algorithms : Lucas Kanade Algorithm

Many algorithms start from the idea of adding constrints to the undetermined system obtained by the brightness constancy equation,

The assumption of this algorithm is : **u is constant in a small neighbourhood of a point** , the neighbour points are moving in the same way , so with the same vector u.

This assumption allows us to obtain a system of equations with one equation for each point in the neighbourhood.

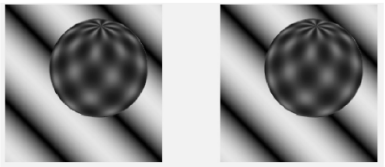
$$(\nabla I(\mathbf{x}_i, t))^\top \mathbf{u} + I_t(\mathbf{x}_i, t) = 0 \quad \mathbf{x}_i \in N$$

we then obtain a linear system Au=b with

$$A = \begin{bmatrix} \nabla I(\mathbf{x}_1, t)^\top \\ \nabla I(\mathbf{x}_2, t)^\top \\ \cdot \\ \cdot \\ \cdot \\ \nabla I(\mathbf{x}_m, t)^\top \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} -I_t(\mathbf{x}_1, t) \\ -I_t(\mathbf{x}_2, t) \\ \cdot \\ \cdot \\ \cdot \\ -I_t(\mathbf{x}_m, t) \end{bmatrix}$$

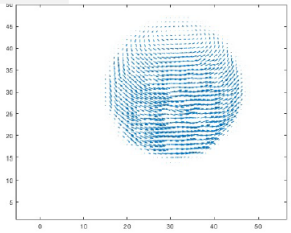
m elements in the N neighbour.

Lucas Kanade example



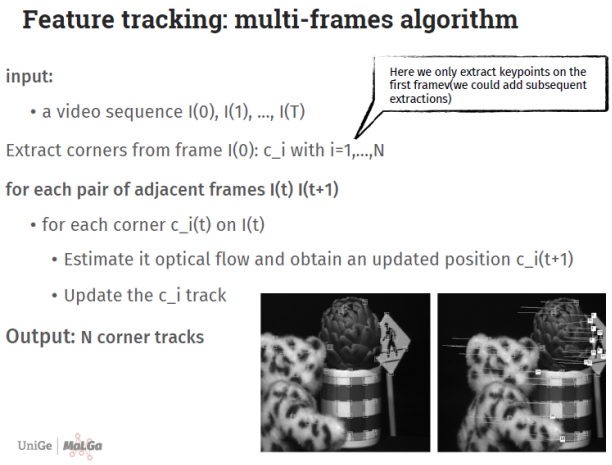
Lucas Kanade on a synthetic highly textured sequence of a slowly rotating sphere

Small displacements!



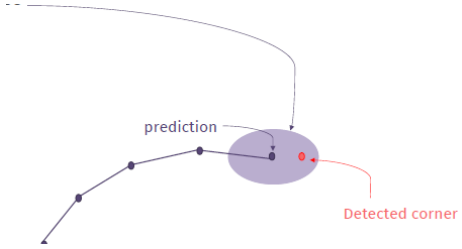
UniGe | MaLGa

3.4 Correspondence over time : Feature Tracking

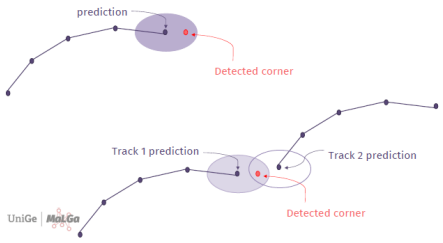


3.4.1 The role of Kalman Filter

Kalman filter allows us to smooth the noisy trajectories Further, it can be used to fill trajectory gaps: we may use the estimated state as a hypothesis of the missing measurement The Pt state covariance matrix provides us with a measure of the state uncertainty ellipse



When we are tracking multiple corners , we often face association problems.

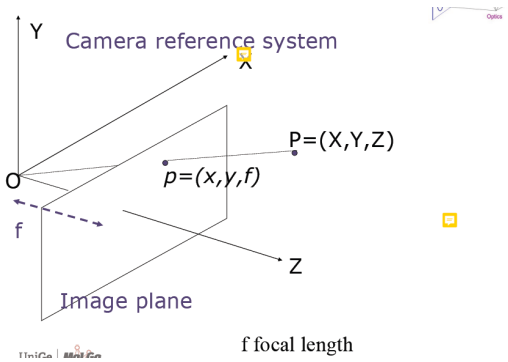


4 Principles of 3D computer vision

4.1 Active 3D Sensors

Time-of-Flight active sensors to measure distance : LIDAR (laser based , short to medium range) , RADAR(RF based , medium to long range).

4.1.1 The Geometry of image formation



When i take a photo what is happening is easy , i'm taking points from 3D world and projective on the image plane 2D.

The focal length is the distance between the camera center and the sensor.

4.1.2 Scale Ambiguity

3D information is lost during the projection to the image plane.

It is not possible (without a proper language) to distinguish small objects from far objects.

4.2 Stereopsis

Stereo vision refers to the ability to infer information on the 3D structure and distance of a scene from two or more images taken different viewpoints.

The correspondence problem : which parts of the left and right images are projections of the same scene element?

The reconstruction problem : Given a number of corresponding parts of the left and right image , and possibly information on the geometry of the stereo system , what can we say about the 3D location and structure of the observed objects?

4.2.1 A simple stereo system

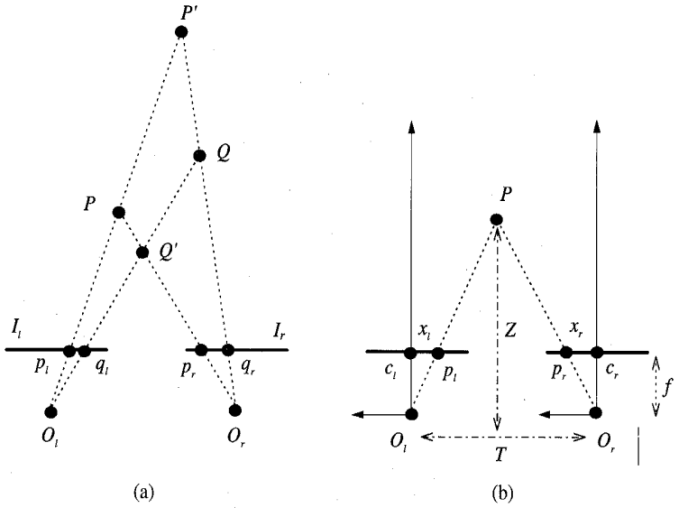


Figure 7.4 A simple stereo system. 3-D reconstruction depends on the solution of the correspondence problem (a); depth is estimated from the disparity of corresponding points (b).

The diagram on the left shows the top view of a stereo system composed of two pinhole cameras. The left and right image planes are coplanar , represented by the segments I_l and I_r, O_l and O_r are the centers of the projection. the optical axes are parallel , for this reason the fixation point , defined as the point of intersection of the optical axes, lies infinitely far from the camera. The way in which stereo determines the position in space of P and Q is triangulation.

Let us now assume that the correspondence problem has been solved , and turn to reconstruction.

It is instructive to write the equation underlying the triangulation .

We concentrate on the recovery of the position of a single point P , from its projections p_l and p_r.

The distance T , between the centers of projection O_l and O_r , is called baseline of the stereo system.

Let x_l and x_r be the coordinates of p_l and p_r , wrt to principal points c_l and c_r , f the common focal length , and Z the distance between P and the baseline.

From the similar triangles (p_l, P , P_r) and (O_l, P , O_r) we have :

$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}.$$

Solving (7.1) for Z we obtain

$$Z = f\frac{T}{d},$$

Whre $d = x_r - x_l$ the disparity , measures the difference in retinal position between the corresponding points in the two images.

From the last formula we can see that depth is inversely proportional to disparity (distant objects seem to move more slowly than close ones).

4.2.2 The problems of stereopsis

Stereo covers two main problems : finding correspondences between image pairs (p , p') and reconstructing the 3D position of a point P given its corresponding projections on the images

4.3 Disparity

The correspondence problem involves two decisions :

1. Which image element to match
2. Which similarity measure to adopt

The first problem can be solve in two ways : use all pixels in the image (obtaining dense correspondences or disparity maps) or use subsets of pixels meeting some requirements(obtaining sparse correspondences)

4.3.1 Dense correspondences

We consider the so calle correlation methods , the elements to match are image windows of fixed size , and the similarity criterion is a measure of the correlation between windows in the two images. The corresponding element is given by the window that maximizes the similarity criterion within a search region.

We assum we have two rectified images , where conjugate points lie on corresponding scanlines of the image("rows").

Our goal is to obtain a disparity map giving the relative displacement for each pixel.

Assuming a fixation point at infinity , disparity is proportional to the inverse of the distance.

In a standard color coding bright areas correspond to high disparities.
In the next there is a summary of the algorithm

input:

- a stereo pair of rectified images *Il* and *Ir*
- size of a correlation window W
- a search range [dmin, dmax]

for each pixel *pl* of *(i,j)* coordinates in *Il*

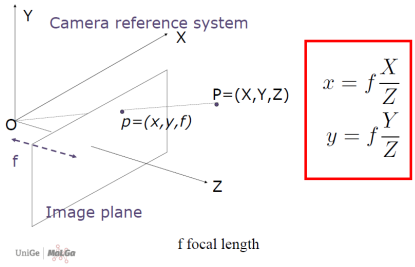
- for each disparity d in the search range
  - estimate the similarity $c(d) = \phi(N1(i,j), N2(i,j+d))$ — *For instance SSD or NCC (see image matching class)*
  - the disparity of the pixel is $\overline{d} = \text{argmax}_{d \in [d_{min}, d_{max}]}\{c(d)\}$

Dense correspondences: left-right consistency

- correspondences are made more difficult by occlusions (points with no counterpart on the other image)
- let us compute
  - *Dlr*: disparity map from *Il* to *Ir*
  - *Drl*: disparity map from *Ir* to *Il*
- then D(i,j)=d iff Dlr(i,j)= -Drl (i,j+d)= d

4.4 3D Reconstruction

Geometry of image formation: perspective or pin-hole model



$$x = f\frac{X}{Z}$$
$$y = f\frac{Y}{Z}$$

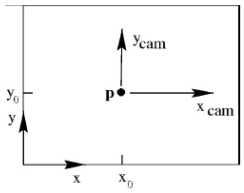
4.4.1 Parameters of a stereo system

The **intrisic** parameters characterize the transformation mapping an image point point from camera to pixel coordinates in each camera.

The **extrinsic** parameters describe the relative position and orientation of the two cameras

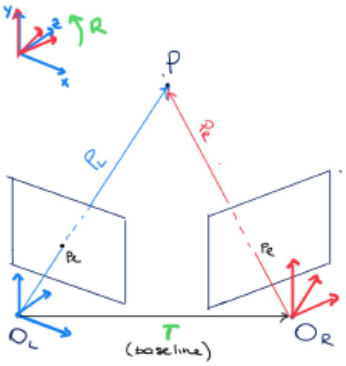
The intrinsic parameters, comprehend a minimal set for each camera that include the coordinates of the principal point and the focal length in pixel.

The extrinsic parameters , instead , describe the rigid transformation (translation and rotation) that brings the reference of the two cameras onto each other.



4.4.2 How to relate points in the world with pixels in the image

The reference frames of the left and right cameras are related via the extrinsic parameters. These define a rigid transformation in 3D space, defined by a translation vector $T = (O_r - O_l)$ and a rotation matrix R. Given a point P in space, the relation between P_l and P_r , is therefore $P_r = R(P_l - T)$



Internal parameters can be known a priori or estimated by camera calibration procedures.

External parameters can be obtained starting from point correspondences.

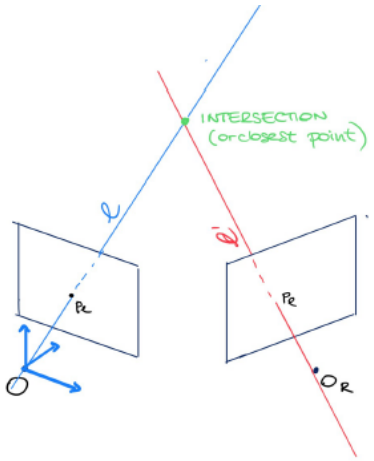
4.4.3 2D to 3D : points triangualation

Assuming we know the internal and external parameters.

Given a pair of corresponding in 2D (metric coordinates on a common reference frame) (p_l, p_r).

Estimate the equation of the two projection rays l , l'.

Compute the intersection it will be the 3D reconstruction P



4.5 Quick detour on geometry background

4.5.1 Homogeneous Coordinates

Points on a plane : (x,y) $\in R^2$

Lines -: Equation of a line on a plane : ax + by + c = 0

We may also represent it as $(a, b, c)^T$

Notice that $(ka, kb, kc)^T$. $k \neq 0$ it is the same line (they belong to the same equivalence class).

This equivalence class is called a **homogeneous vector**

The line is fixed with the coefficient a,b,c. If i multiply the coefficients with a scalar i got the same line as before.

Projective plane : The projective plane is formed by any $(a, b, c)^T$ representative of an equivalence class : $R^3 - (0, 0, 0)$

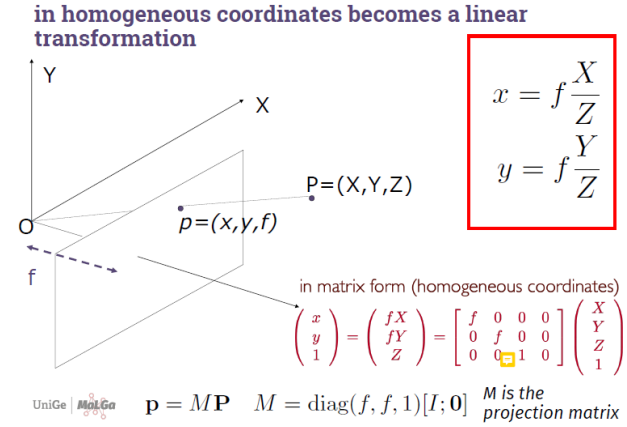
The triplet (0,0,0) could never be a line on the plane.

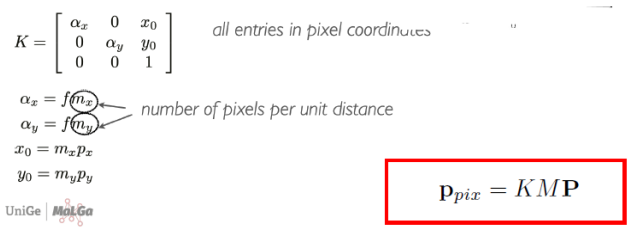
A point and a line : A point $(x, y)^T$ lies on a line $l = (a, b, c)^T$ IFF ax+by+c = 0

If we define $x = (x, y, 1)^T$ then $x^T l = 0$

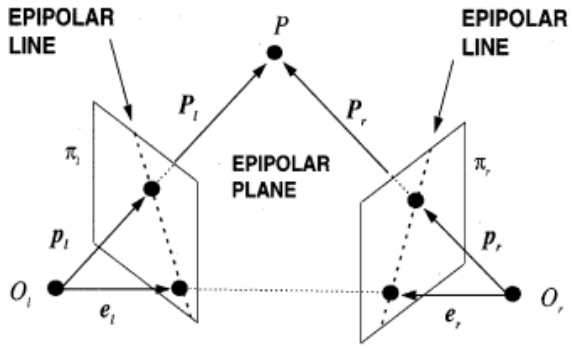
A point on the Euclidean plane may be represented as a point in the projective plane (as a 3D vector with a final 1).

4.6 Perspective model





4.7 Epipolar Geometry



Given a stereo pair of cameras , any point in 3D space . defines a plane π_p, going through P and the centers of projection of the two cameras. The plane π_p is called **epipolar plane**, and the lines where π_p intersects te image planes conjugated epipolar planes. The image in one camera of the projection center of the other is called **epipole**.

The figure shows two pinhole cameras, their projection centers O_l, O_r and image planes $\pi_l.\pi_r$

The vectors $P_l = [X_l, Y_l, Z_l]^T$ and $P_r = [X_r, Y_r, Z_r]^T$ refer to the same 3D point , P thought as a vector in the left and right cameras reference frames respectively.

The vectors $p_l = [x_l, y_l, z_l]^T$ and $p_l = [x_l, y_l, z_l]^T$ refers to the projections of P onto the left and right image plane respectively , and are expressed in the corresponding reference frame.

Let x and x' two point that intersect the two reference frame.

How is x' constrained?

x' lies on the line P_r intersection between the epipolar plane and the right image plane

P_r is the projection on the image plane of the ray passing through O_l and x.

In practice , when we look for the corresponding point x' we can limit our search to a line , P_l.

From a 2D to a 1D problem!

The obvious question at this point is , can we estimate the epipolar geometry? How do we determine the mapping between points in one image and epipolar lines in the other?

The Essential Matrix E

The equation of the epipolar plane through P can be written as the coplanarity condition of the vectors $P_l, T, P_l - T$ or $(P_l - T)^T T x P_l = 0$.

Using $P_r = R(P_l - T)$ we obtain $(R^T P_r)^T x P_l = 0$.

Recalling that a vector product can be written as a multiplication by a rank-deficient matrix , we can write $TxP_l = SP_l$

Then we got , $P_r^T E P_l = 0$ with E = RS.

By construction , S has always rank 2. The matrix E is called the Essential matrix and establishes a natural link between the epipolar constraint and the extrinsic parameters of the stereo system.

The Fundamental Matrix

We now show that the mapping between points and epipolar lines can be obtained from corresponding points only , with no prior information on the stereo system

We can now derive an equivalent equation relating points in pixel coordinates.

$x_m = K_l^- 1x$, $x'_m = K_r^- 1x'$.

Then $x'^T K_r^T E K_l^- 1x = 0$

The fundamental matrix satisfies the equation $x'^T Fx = 0$, for each pair of (x,x') of corresponding points in pixel coordinates.

Suppose we have two images acquired by cameras with no coinciding centres : $x'^T Fx = 0$.

The the fundamental matrix F is the unique 3x3 rank 2 matrix so that, for each corresponding pair (x,x')

Fundamental matrix is a map between point and lines.

$l' = Fx$

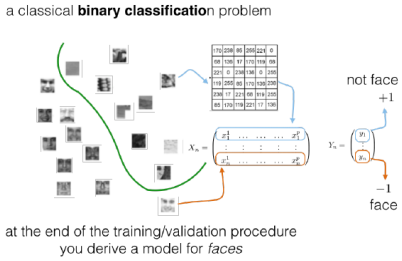
5 Image Representations

5.1 Image and Semantics

Image Classification : Associate a label to an image.

$$(x_1, y_1), \ldots, (x_n, y_n)$$
$$x_i \in \mathbb{R}^p \quad \text{and} \quad y_i \in Y, \quad i = 1, \ldots, n$$

$$X_n = \begin{pmatrix} x_1^1 & \cdots & \cdots & \cdots & x_1^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^1 & \cdots & \cdots & \cdots & x_n^p \end{pmatrix} \qquad Y_n = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$



X is p-dimensional input vector and Y is the output.

Multiclass classification : The set of possible labels grows

5.2 Object recognition vs Instance recognition

Notice the difference. Instance recognition : "his car" , "that bicycle" , "my mug"

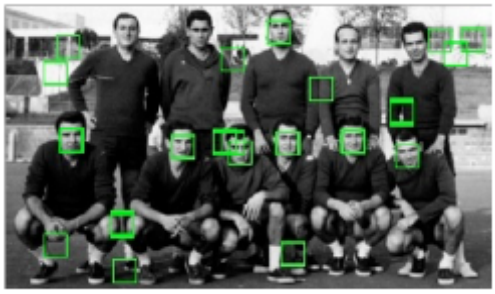
Category recognition : generic object recognition , "cars"

5.3 Object detection in its classicial formulation

Object detection is in essence a classification problem. Image regions of variable size are classified : is it an instance of the object or not?

A special case occurs when we have only a class of interest.

Unbalanced classes : In this 380x220 px image we perform millions of tests and we should find only 11 positives.



5.4 Global Image Representations

A list incorporating what we have seen so far :

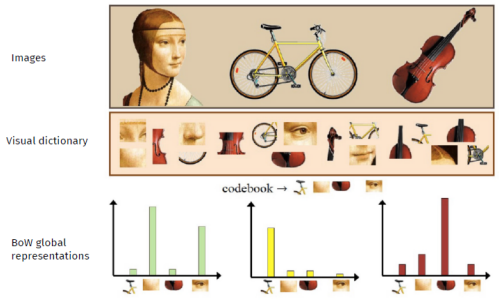
- Pixels (not so robust)
- Graylevels/color/gradient histograms(not too descriptive)
- Local keypoints (in Bag of keypoints)

5.5 Finding a global feature vector from local keypoints

5.5.1 Bag of words : the inspiration comes from text analysis

One of the simplest algorithm for category recognition is the bag of words.

This algorithm simply computes the distribution of visual words found in the query image and compares the distribution to those found in the training images. In Bags of words there are no geometric relationships between parts or features.



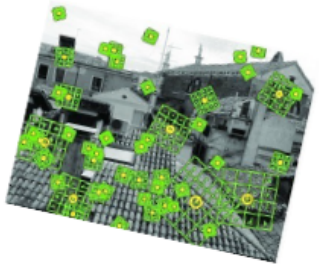
1. Build a visual dictionary from data
2. Represent all images with terms frequencies , or more specifically by quantizing its key points with respect to the dictionary

5.5.2 BOW - Visual Dictionary

Take 1 - Sparse

Start considering an appropriate training set of images G.

- Extract local features or keypoints in each image
- Compute descriptors (SIFT) from each key point and gather all of them in a common set V
- Group coherent descriptors in k groups , for instance with K-means
- Compute visual words (eg K-means centroids) and collect them in a dictionary D of size k



Take 2 - Dense

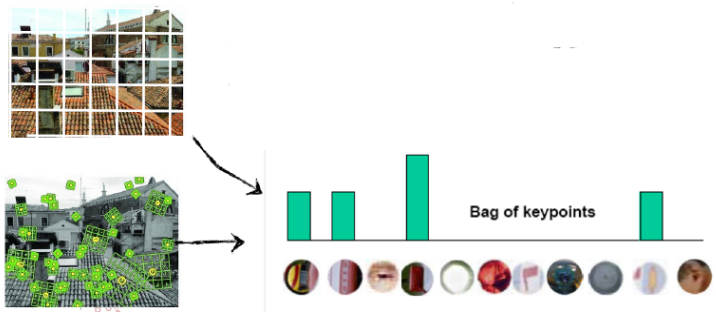
Compute SIFT on a dense set of image patches.
Start considering an appropriate training set of images G

1. Compute descriptors from each patch and gather all of them in a common set V
2. Group coherent descriptors in k groups , for instance with K-means
3. Compute visual words and collect them in a dictionary D of size K

Compute a histogram of the frequencies of all visual words appearing in it.

Depending on the type of images the sparse approach can produce very sparse feature vectors..

For this dense approach can be a good alternative.



5.5.3 Coding-Pooling Image representation

We compute a set of local descriptors $\mathcal{X}$ associated with an image, so that card($\mathcal{X}$)=n $\quad \mathcal{X} \subset \mathbb{R}^d$

Coding or Embedding step $\quad \phi : \mathbb{R}^d \to \mathbb{R}^D$

encodes each key point $\quad x \in \mathcal{X} \quad x \to \phi(x)$

Pooling or Aggregating step computes a single vector from a set of through an aggregating function $\quad \{\phi(x_1), \ldots, \phi(x_n)\}$

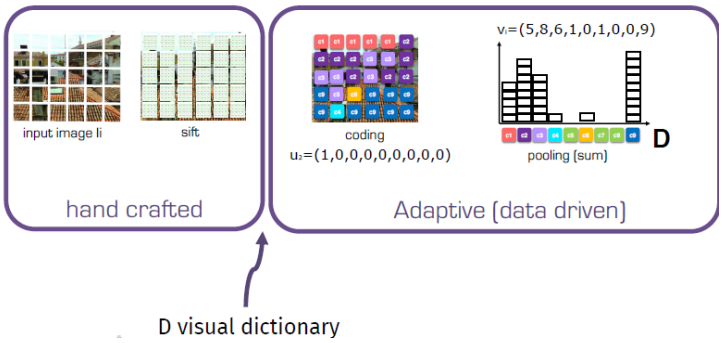
for instance a sum $\quad \psi(\mathcal{X}) = \sum_{x \in \mathcal{X}} \phi(x)$

In the BoW case , given a dictionary D of size $|D| = D$

Each feature x is encoded to a D-dimensional vector :

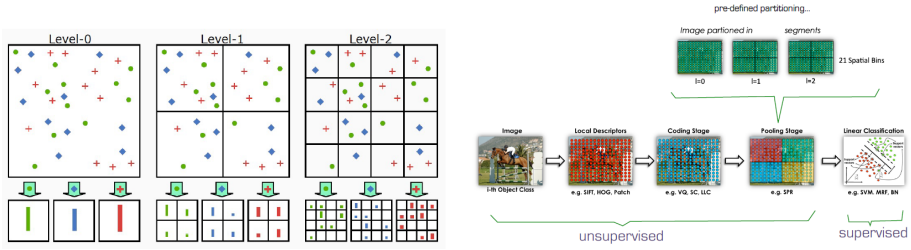
$\phi_B OW(x) = [0, ..., 0, 1, 0, ...0]^T$, the non-zero position is determined based on a nearest neighbor assignment rule.

The pooling step represents the histogram computation.



5.5.4 Pooling over a spatial pyramid

Goal : Restore some locality (we loose invariance)

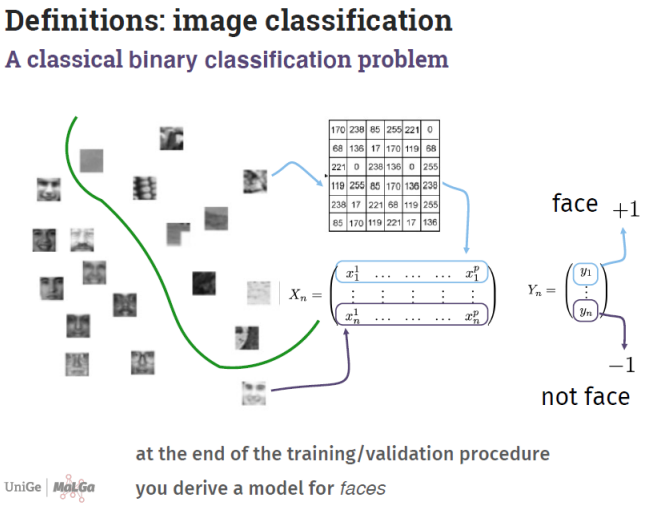


6 Object Detection : The Classical Way

6.1 Problem Definition

6.1.1 Definitions : Object Detection

Object detection is in essence a combination of binary classification problems. Image regions of variable size are classified.



Classification : associate a global label to the image (kitten or not , the first image is not an image of the kittens, because there are multiple kittens and contains other stuff).

Detection : It is the association of a label to a portion of the image.

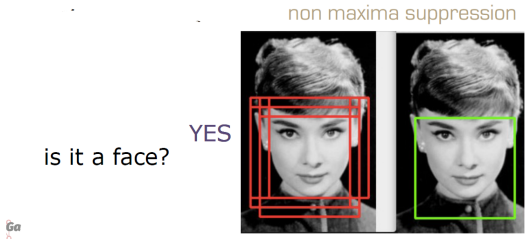
With unbalanced class , the second image is 380 x 220 px image we perform 6.4 x 10 to 5 tests and we should find only 11 positives.

The training set contains (third image) , images of positive examples and negative examples (fourth images).

6.1.2 Object detection : Basic idea

Slide a window across image and evaluate an object model at every location.

I'll have multiple square that will say that inside that square there is a face , this is a requirement. Because we cannot have only one yes.



In the past object detection has usually been adopted for very "meaningful" and very specific classes (faces , people , cars).

We have to remember that humans can recognize low-resolution faces of familiar people.

6.1.3 Object Detection and Data Representation

The complexity of the task requires we find appropriate data representations.

Before the DL era , most of the methods started off from local elements and later derived intermediate and then global information.

Local information may be :

- Encoded by means of pre-defined general purpose dictionaries
- Learnt from data

Many effective object detection methods resort to over-complete general purpose sets of features as they are effective for modelling "standardized" visual information.

Today we explore two alternatives , building on methods we know of : face detection with wavelet-based features and Pedestrian detection with gradient-based features.

6.2 Face Detection

6.2.1 2D Wavelets

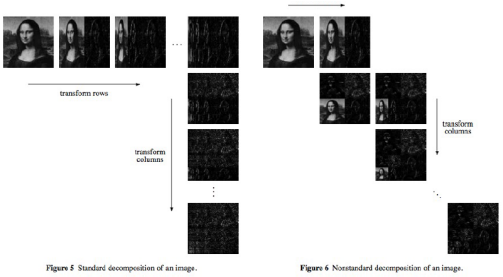


Figure 5 Standard decomposition of an image.

Figure 6 Nonstandard decomposition of an image.

6.2.2 2d Haar Basis

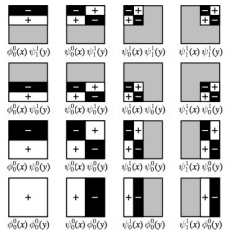


Figure 7 Standard construction of a two-dimensional Haar wavelet basis for V^2. In the unnormalized case, functions are +1 where plus signs appear, −1 where minus signs appear, and 0 in gray regions.

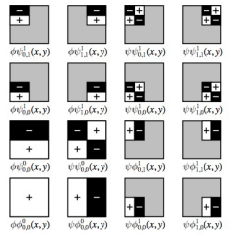
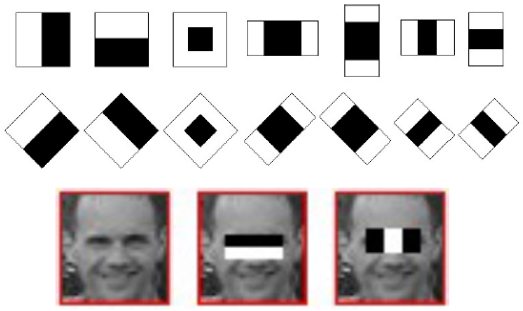


Figure 8 Nonstandard construction of a two-dimensional Haar wavelet basis for V^2.

Haar features or rectangle features : +1 and -1 filter coefficients .



Feature value is sum of the pixels in the white region minus the sum of the pixels in the black region.

Value = $\sum$ (pixels in the white area) - $\sum(pixelsinblackarea)$

Rectangle features can be computed very rapidly using an intermediate representation for the image which we call the **Integral Image**.

The integral image at location x,y contains the sum of the pixels above and to the left of x,y , inclusive :

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

where ii(x,y) is the integral image and i(x,y) is the original

Using the integral image any rectangular sum can be computed in four array references.

We have a lots of features , at least with integral images we may learn to compute them faster but they are still many (lots of redundancy)

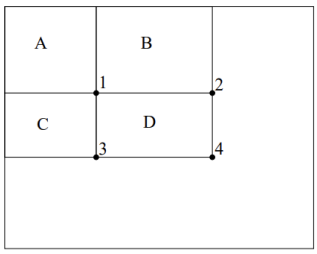
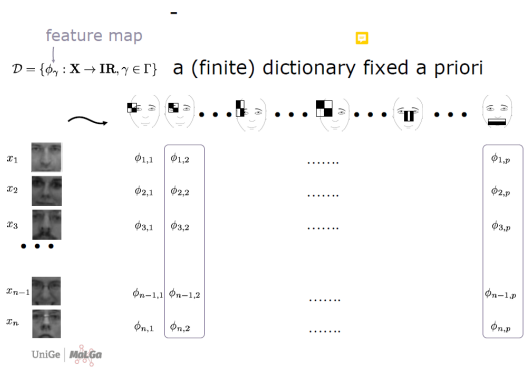


Figure 2: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.



6.2.3 Features Map

We are going to generate , for every image , a set based on those pattern.

Features & high dimensional spaces

$$X_n = \begin{pmatrix} x_1^1 & \dots & \dots & \dots & x_1^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^1 & \dots & \dots & \dots & x_n^p \end{pmatrix}$$

p>>n

under-determined system

Large amount of features, these features are also very redundant and correlated (because of the properties of images)... maybe we can just compute a smaller subset.

6.2.4 Feature Selection Procedures

Set up a learning procedure to select only a subset of meaningful features. These features will be the only one computed at run time.

Feature selection methods for face detection we will discuss : Regularization with a sparsity penalty and a reference face detector (Viola and Jones).

6.3 Viola and Jones face Detector

Effective and Efficient.

The key elements of this approach are : Haar features , embedded variable selection with a boosting procedure and optimization with a classifiers cascade.

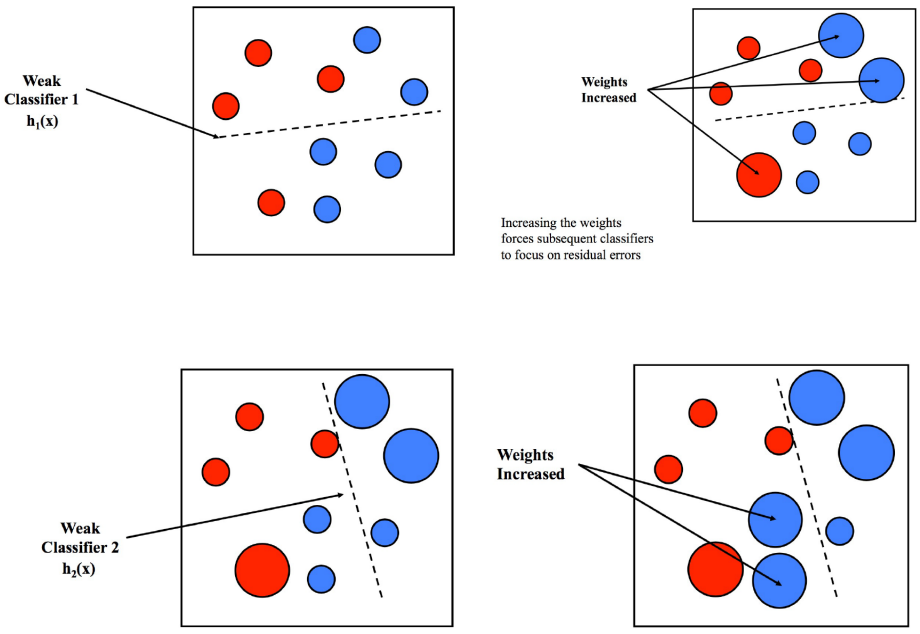
6.3.1 Boosting (overview)

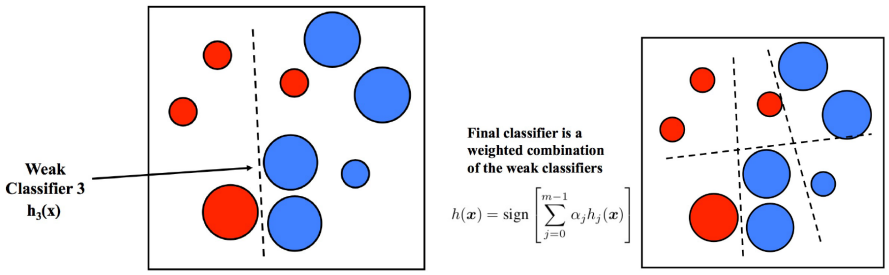
Boosting is a classification scheme that works by combining weak learners into a more accurate strong ensemble classifier (a weak learner needs only do better than chance).

Training consists of multiple boosting rounds.

During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners.

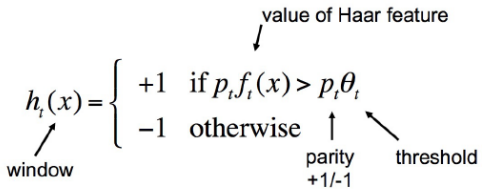
"Hardness" is captured by weights attached to training examples





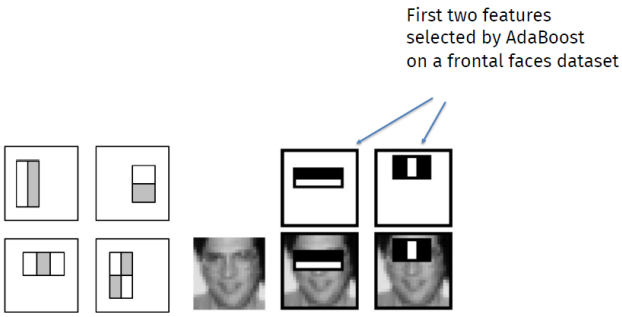
6.3.2 VJ : ADABOOST and Haar Features

The weak learning algorithm is designed to select the single rectangle features which best separate the positive and negative examples. For each feature , the weak learner determines the optimal threshold classification function , such that the minimum number of examples are misclassified.



For each round of boosting :

- Evaluate each rectangle filter on each example
- Select best threshold for each filter
- Select best filter and threshold combination as the weak learner
- Reweight examples



This feature combination can yield 100% detection rate

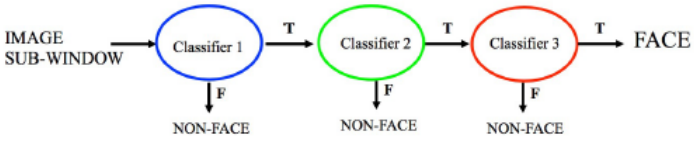
and 50% false positive rate

6.3.3 VJ : Classifiers Cascade

Start with simple classifiers which reject many of the negative subwindows while detecting almost all positive sub-windows

Positive response from the first classifier triggers the evaluation of a second classifier , and so on.

A negative outcome at any point leads to the immediate rejection of the sub-window.



Solves several problems : improves speed by early rejection of non-face regions by simple classifiers, reduces false positive rates.

The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages.

Training the cascade :

- Set target detection and false positive rates for each stage.
- Keep adding features to the current stage until its target rates have met.
- Test on a validation set : if the overall false positive rate is not low enough , then add another stage.
- The classifiers in the cascade are trained using AdaBoost on the remaining set of example images

- Thus, if the first stage classifier rejects a number of images then these images are not included when training the second stage classifier.
- But use false positive from current stage as the negative training examples for the next stage

The sub-images which do not contain the object are gradually removed from consideration leaving just the objects which are sought.

6.4 Pedestrian Detection

Another example of object detection.

Notice that the underlying problem is quite different.

We could set up the same procedure described for face detection.

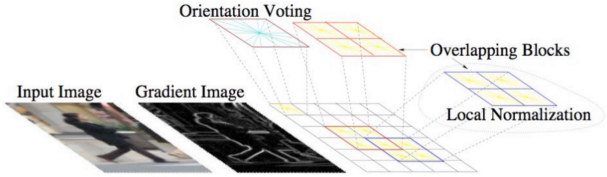
Among the better known methods : Histograms of Oriented Gradients (HoGs)

6.4.1 Hog - Computational sketch

Given an image region of appropriate aspect ration , we have to compute the image gradient.

Cell histogram : each pixel in a cell casts a weighted vote for an orientation (quantized).

Blocks : Histograms in a block are concatenated and then normalized.

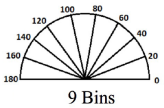


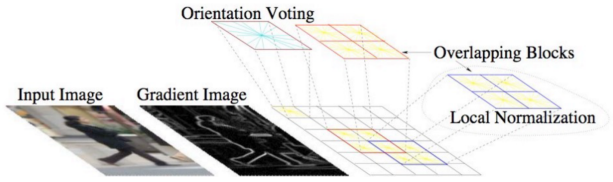
Descriptors belonging to the different blocks are finally concatenated

UniGe | MoLGa

Cell histogram - Each pixel in a cell casts a weighted vote for an orientation (quantized)

cell 8x8 pixels
weight - gradient magnitude
bins - 9 orientation angles [0-180]





6.5 Conclusions

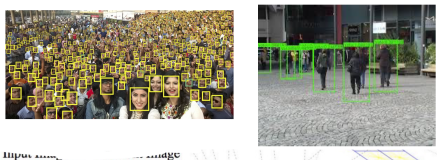
Object detection can be seen as a combination of binary image classification tasks (object vs non objects).

It has been long addressed for very specific classes (faces, pedestrians, cars).

Data representation has been the main focus of research, together with computational efficiency.

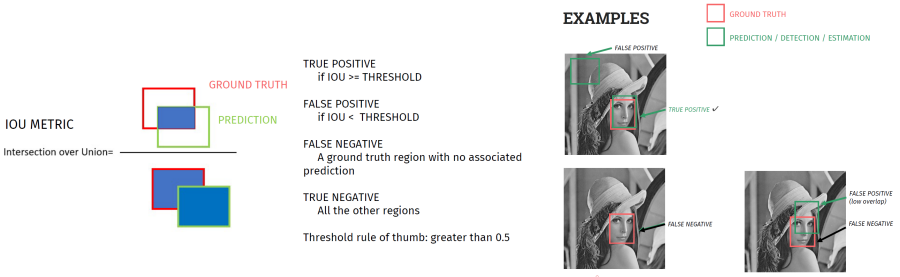
7 Object Detection and Deep Learning

Given a class of objects of interest (face , pedestrian) and given an input image I which contains N instances of the object , locate all the instances that is , find (x_i, y_i, w_i, h_i); i = 1..N the center and dimensions of boxes that best localize the objects in image I.



7.1 Detection evaluation

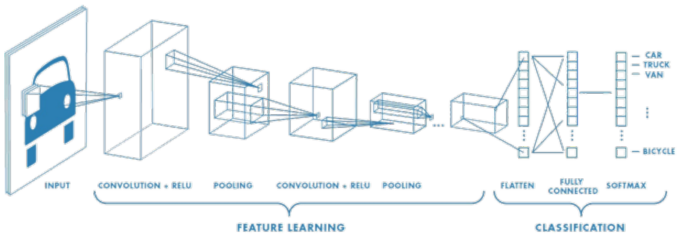
7.1.1 How are object detectors evaluated



7.2 Problem (re)definition

Given a set of classes of interest C (e.g. Street objects , office objects) and given an input image I that contains N object instances : find the center , the class and the dimensions of boxe (aligned with the coordinate system) that best localize the objects in image I , formally : $(c_i, x_i, y_i, w_i, h_i); c_i \in C, i = 1..N$

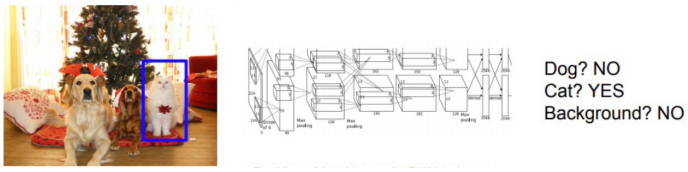
CNN and image classification (review)



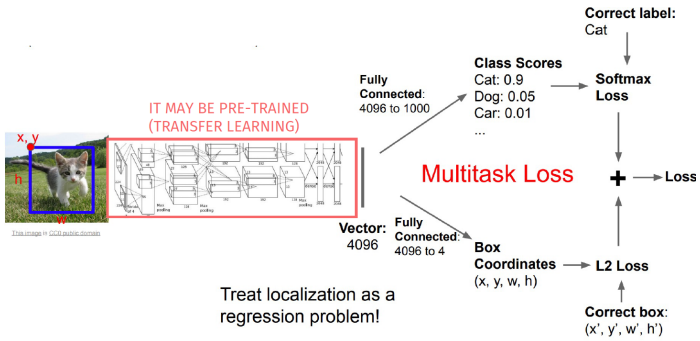
7.2.1 Sliding Window

Apply a CNN to many different parts of the image , CNN classifies each crop as object or background.

The problem is that it needs to apply CNN to huge number of locations and scales and so it's very compuationally expensive.



7.2.2 Classification + Localization



7.3 Region-based CNNs

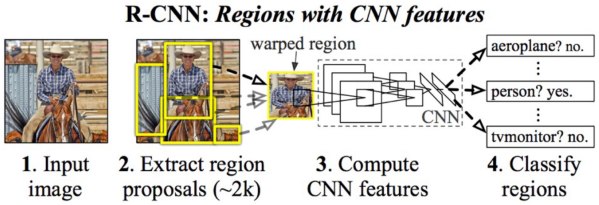
7.3.1 Region proposals

Find image regions that are likely to contain objects.

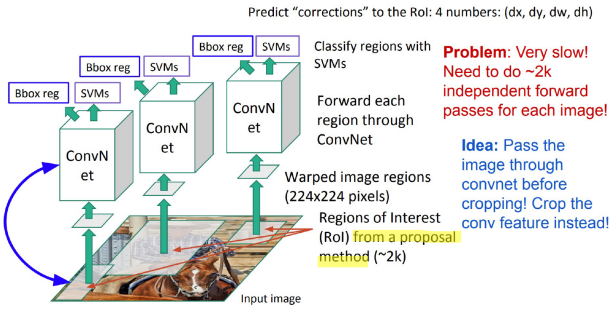
7.3.2 R-CNN

Two steps :

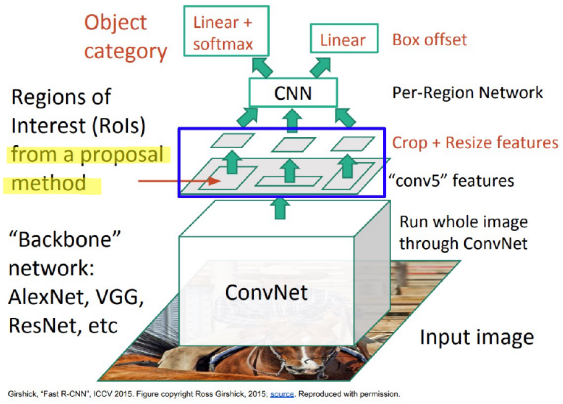
1. Selective search : identification of region proposals
2. CNN



"Slow R-CNN"



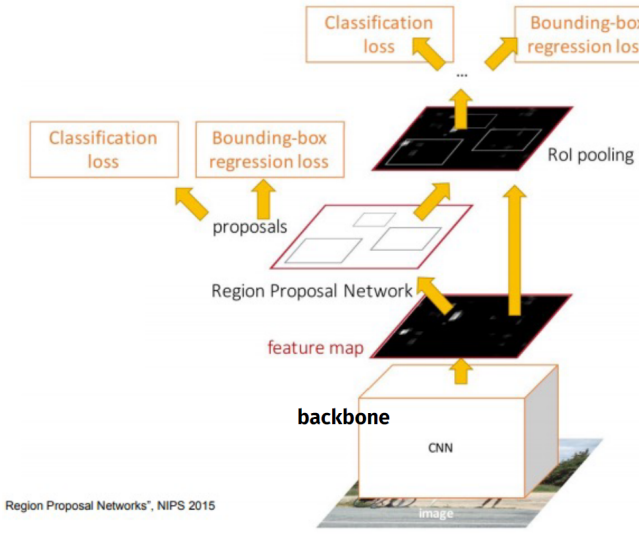
"Fast" R-CNN



The **backbone** is the "internal" network meant for feature extraction

7.3.3 Faster R-CNN

The network learns its proposals : insert a **Region Proposal Network (RPN)** to predict proposals from features



A convolutional backbone is used to extract features from the input image.

A RPN is used to generate proposals (regions with high probability of containing object. The network follows a brute-force approach to classify regions

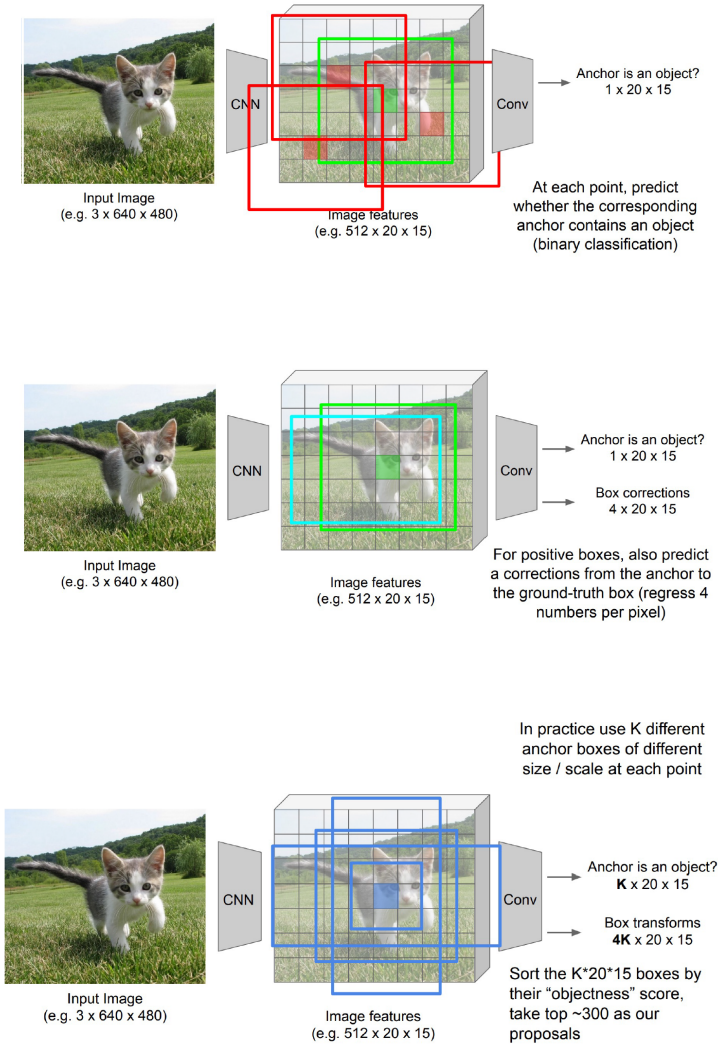
of the image to one of the two classes : background/object).

These proposals are used to perform RoI pooling of the feature map and extract a region-based features.

These features are then used to predict a class probability and a box offset.

7.3.4 Region Proposal Network

Red and green line forms the anchor box of fixed size centered on each point of the feature map.



8 Image Segmentation

Segmentation is a difficult task to solve and to formulate.

Images can also be divided in somewhat uniform image regions.

Uniformity is based on a quality (e.g. brightness , color or motion).

- Image segmentation (regions that can be further analyzed)
- Superpixels computation (features)

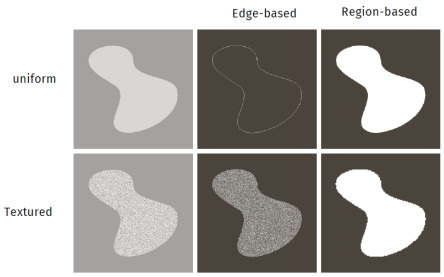
8.1 Image Segmentation

Unsupervised segmentation: is based on pixels and appearance information.

Supervised segmentation: is guided by some semantic concept

8.1.1 Unsupervised image segmentation methods overview

- Edge-based methods: look for discontinuities
- Region-based methods: look for similarity regions according to some criteria



8.1.2 Region based methods

Thresholding (followed by connected components computation). Region growing. Morphological watersheds.

Clustering methods(K-means,mean shift).

8.1.3 Image segmentation : definition

Let R be the spatial region occupied by an image. Image segmentation may defined as the process of partitioning R into n subregions.

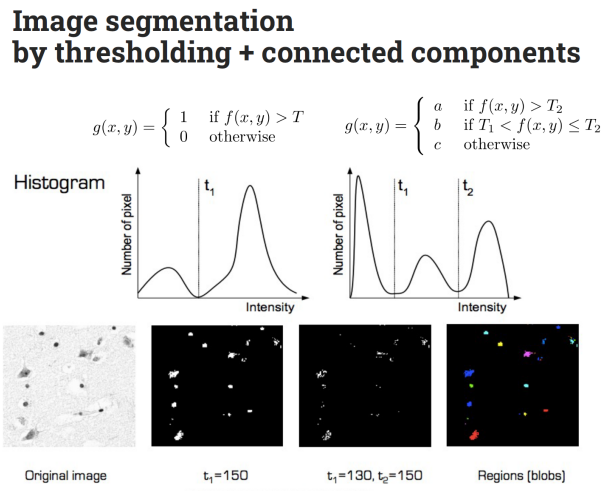
We already know a thresholding mehtod which is : **motion segmentation)**.

How to choose T automatically????

1. Select an initial estimate of T

$\cup_{i=1}^{n} R_i = R$
R_i is a connected set $i = 1, \ldots, n$
$R_i \cap R_j = \emptyset$ for all i and $j, i \neq j$
$Q(R_i) =$ TRUE $i = 1, \ldots, n$
$Q(R_i \cup R_j) =$ FALSE for any adjacent region R_i and R_j

Q is a logical predicate defined over the points in a set, for instance: Q(A)=TRUE if all pixels in A have the same intensity level



2. Segment the image using T (G1 are pixels so that f(x,y) ¿ T, G2 the others)
3. Compute the mean intensity values in G1 and G2, let them be m1 and m2
4. Compute a new threshold : $T = 1/2(m_1 + m_2)$
5. Repeat steps 2 to 4 until T does not change any more

8.1.4 Otsu's method for global thresholding

Objective : Segment foreground/background by maximizing the between class variance

All computations are performed on the normalized histogram of the image (the value of each bin is pi).

Let us consider a threshold $T(k) 0 < k < L - 1$ (L is the maximum value in the range, eg 255) and use it to segment image pixels in two classes : C1 pixels in the range [0,k] and C2 pixels in the range [k+1,L-1].

– The probability that a pixel is in C_1 is $P_1(k) = \sum_{i=0}^{k} p_i$

– Similarly, the probability that a pixel is in C_2 is

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$$

– The mean intensity value of pixels assigned to class C_1 is

$$m_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} i p_i$$

– Similarly

$$m_2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} i p_i$$

$$m_G = \text{ global mean } = \sum_{i=0}^{L-1} i p_i$$

– Verify by substitution that

$$P_1 m_1 + P_2 m_2 = m_G$$
$$P_1 + P_2 = 1$$

– The optimal threshold can be found as follows

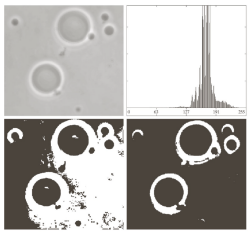
$$k^* = \arg \max_{0 \le k \le L-1} \eta(k)$$

with

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$$

where

$$\sigma_G^2 \text{ (global variance)} = \sum_{i=0}^{L-1} (i - m_G) p_i$$
$$\sigma_B^2 \text{ (between class variance)} = P_1(m_1 - m_G) + P_2(m_2 - m_G)$$



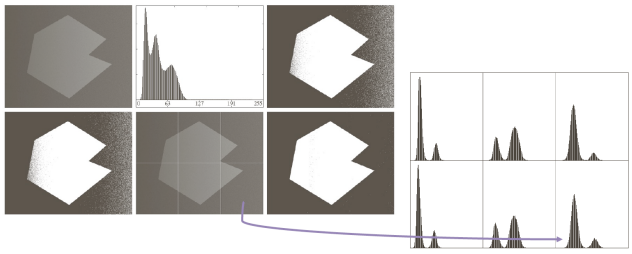
a | b
c | d

(a) original image;
(b) histogram of (a);
(c) global threshold: $T = 169$, $\eta = 0.467$;
(d) Otsu's method: $T = 181$, $\eta = 0.944$.

8.1.5 Variable Thresholding

Objective: choosing different thresholds for different image parts

Simple idea: subdivide an image into non overlapping rectangles.



Variable Thresholding from local image properties

Compute a threshold T_{xy} in each point (x,y) based on its neighbourhood properties.

A simple example of this approach is the following : $T_{xy} = a\sigma_{xy} + bm_{xy}$ $a, b > 0$

Variable Thresholding by moving average

This approach has been widely adopted for text segmentation.

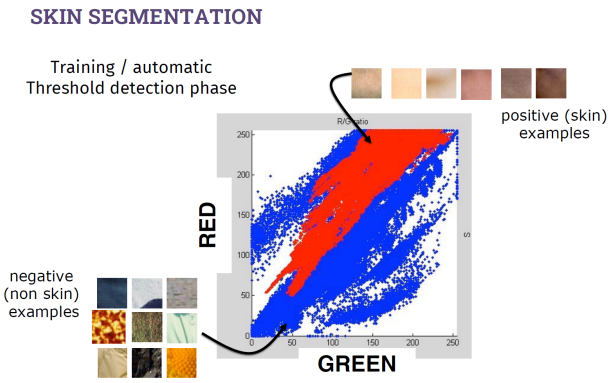
There is a uniform distribution of text and background over space.

8.2 Color Segmentation

8.2.1 Color-based Thresholding : one possibility

1. Smooth the image
2. Transform to HSV space
3. Discard V
4. Choose thresholds for H and S appropriate for the task you are trying to address.

Examples



8.3 Clustering Methods

What if you don't have a specific color in mind?

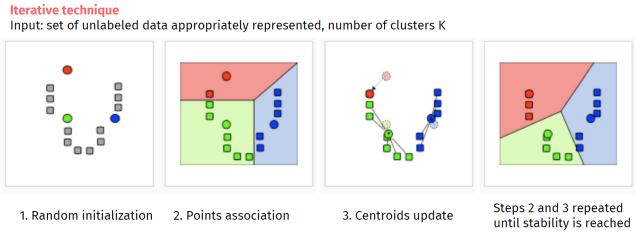
For more generic color segmentation: image pixels can be grouped in clusters of elements with similar colors.

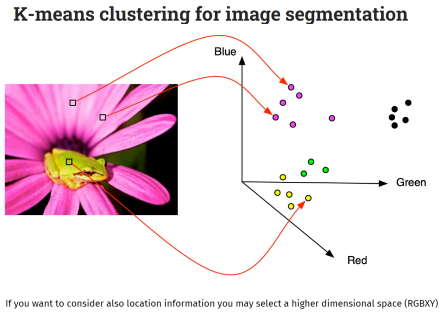
A classical choice is K-means, you already met several times.

8.3.1 K-means clustering reminder

Given a set of data $(x_1, ..., x_n)$ the goal is to partition it into K sets $S = (S_1, ..., S_K)$ so to minimize the within-cluster variance :

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{K} \sum_{\mathbf{x} \in S_i} ||\mathbf{x} - \mu_i||^2$$





The challenge is : how to choose K

Try multiple K and search the ones that give the highest confidence by using a cluster quality indicator.

This approach does not take into account the cluster size.

Pros: simplicity mainly

Cons: if we do not have priors on the choice of K we may obtain non deterministic results. The method does not prevent the formation of unbalanced clusters.

8.4 Graph-based Image Segmentation

The goal of this approach is to devise a segmentation method that "extracts a global impression of an image".

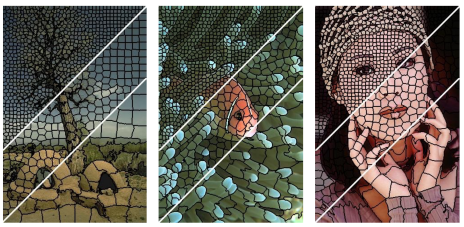
The segmentation process here becomes a graph partitioning problem.

This involved the following questions : what is a precise criterion for a good partition? How can we make it computationally efficient?

8.5 Superpixels computation

Small image parts characterized by internal uniformity.

Superpixels should be evenly distributed and should follow very well objects boundaries.

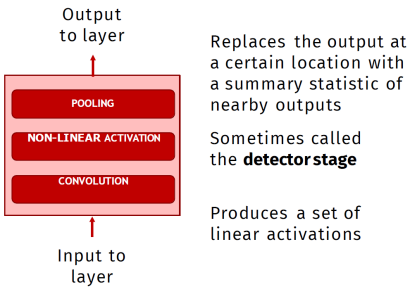


SLIC (Simple linear iterative clustering)
Simple and fast algorithm , based on Kmeans,
Search space limited to a region proportional to the desired super pixel size.
A weighted distance measure combines color and spatial proximity.

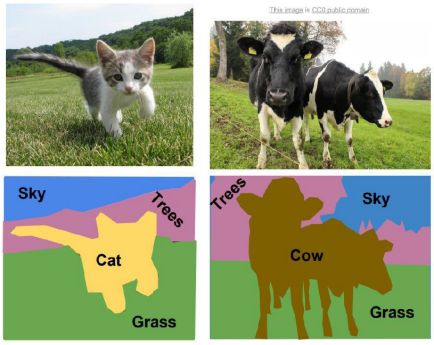
9 Instance Segmentation

9.1 Semantic Segmentation

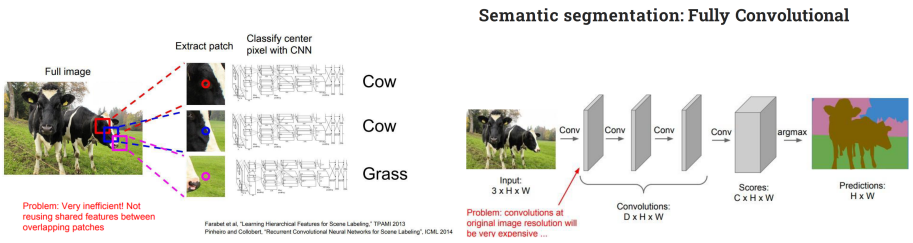
9.1.1 A typical CNN layer



Semantic segmentation labels each pixel in the image with a category labels. Do not differentiate instances , only care about pixels.

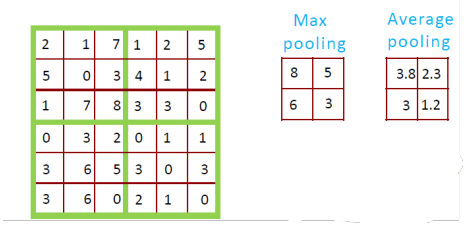


We can do it with sliding window technique

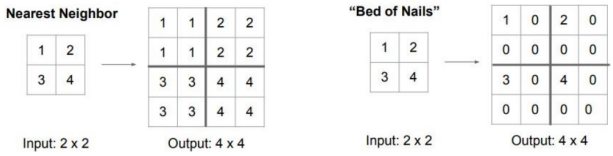


9.1.2 Downsampling : Pooling

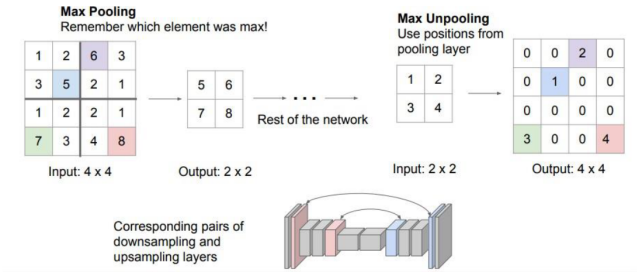
Pooling can help with local invariance although some information is lost. No parameter to be estimated here.



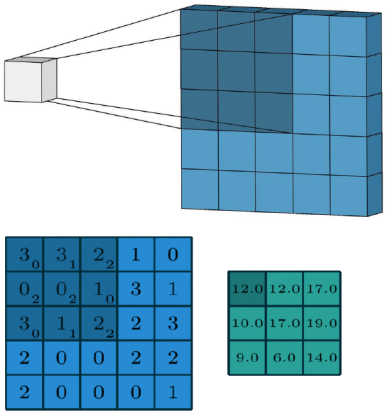
9.1.3 Upsampling : Unpooling



9.1.4 Upsampling : Max Unpooling

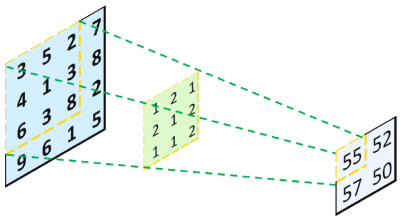


9.1.5 Downsampling : Convolution



9.1.6 Understanding upsampling convolution

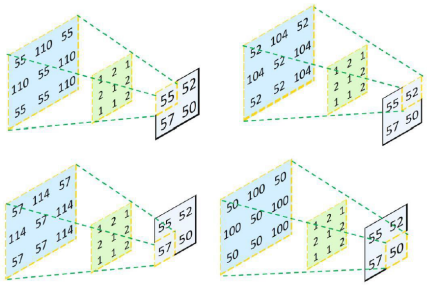
Let us consider this example where a 4x4 image is filtered with 3x3 convolution filters resulting in 2x2 matrix



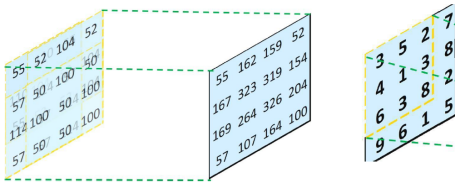
Now we want to go back to the original image. We do the opposite.

9.1.7 Upsampling : Transposed Convolution

For each pixel value we use the conv filter to transpose the convolution. So, we have one output map for each pixel.



We overlap the transposed maps (according to position , and we sum overlapping values obtaining the final output).



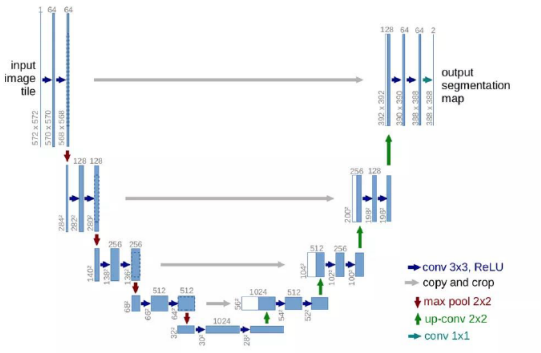
Values are different! Transpose convolution kernels are learnable (training)

9.2 Basic Neural Networks architectures for segmentation

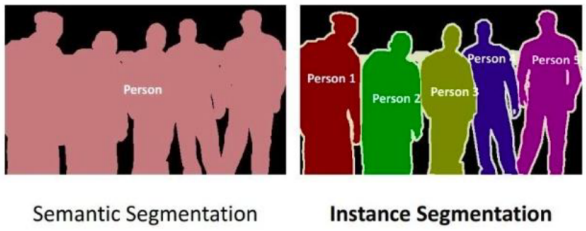
9.2.1 A basic network for semantic segmentation: the Unet

It is an Encoder-decoder network

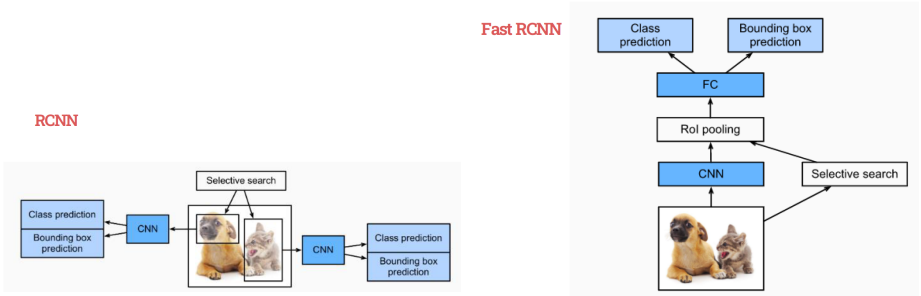
This is the architecture summary.



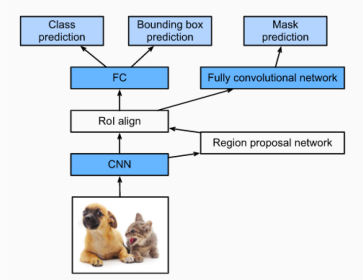
9.3 Instance Segmentation



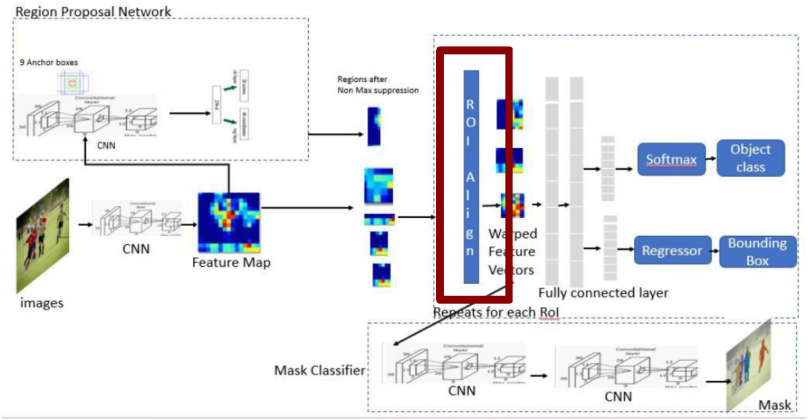
9.3.1 Previously in Object Detection



9.4 Mask RCNN



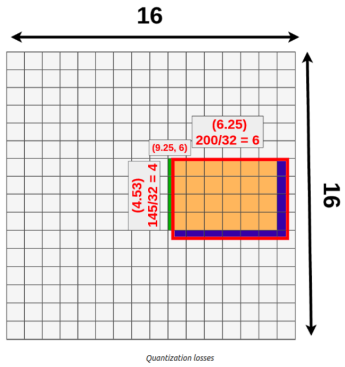
9.4.1 Architecture



9.4.2 Motivation and main features

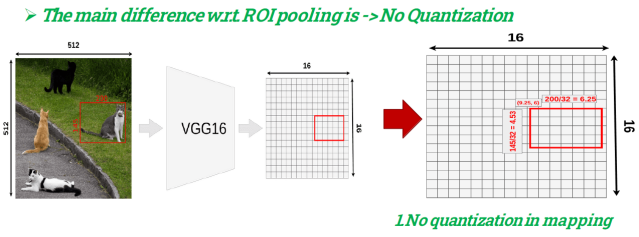
1. We get a fixed-size feature map obtained from a deep convolutional network with several convolutions and max pooling layers.
2. Region proposal provide in the first stage , many different ROI candidate.
3. ROI-Poooling : for every ROI , the corresponding features map region is scaled always in the same size (e.g 3x3).
4. So the corresponding region in feature map is divided in sections , and the max value is considered.
5. The resulting mapped and pooled ROI is fed to a FCN $->$ mask

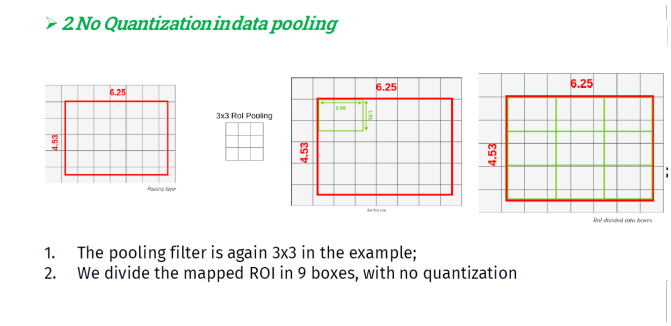
9.4.3 Faster RCNN - Region of Interest

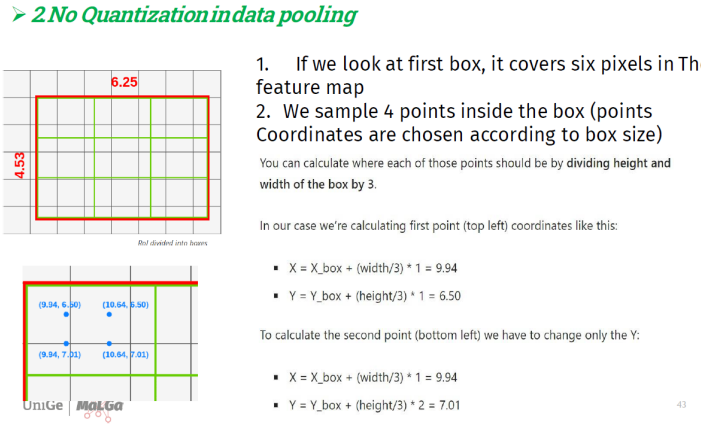


9.4.4 Faster RCNN - Region of Interest - Pooling (RoI Pooling)

9.4.5 Mask RCNN : Region of Interest Alignment

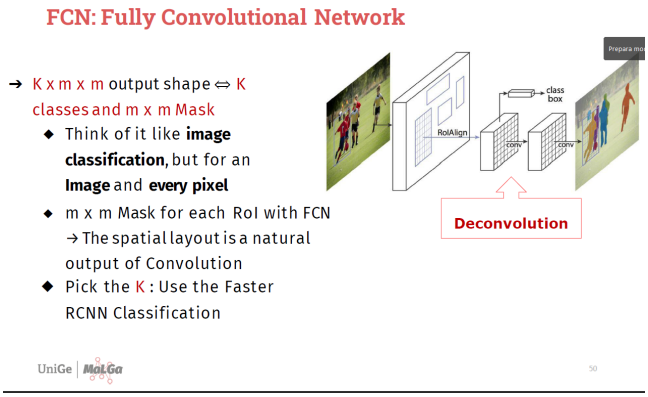






Now we apply bilinear interpolation for each of the sampling points. Then, max pooling on the interpolated data and iterate for each block.

9.4.6 Mask RCNN: The Mask Branch



10 Pose Estimation

10.1 Problem Definition

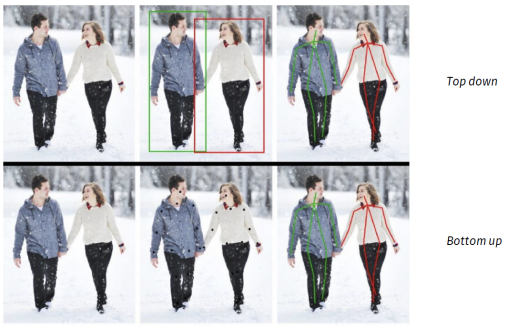
Problem : estimating the configuration of the body (pose).

While classical methods adopt wearable sensors or combinations of markers and motion capture (MoCap) systems , in Computer Vision this task is addressed using images as Inputs.

Pose can be estimated in 2D (apparent joints configuration on the image plane) or in 3D (estimating the pose on a 3D space)

Different methods are based on different assumptions on how many people can be jointly analyzed : single person or multi-persons algorithms exist.

10.1.1 Top-down and bottom-up approaches



Bottom-up approaches can be used as a richer alternative to people detectors.

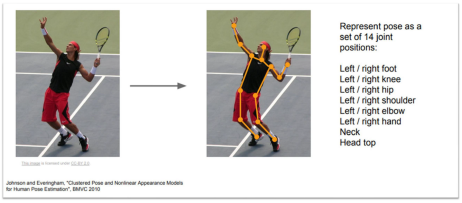
10.2 Computer Vision Algorithms

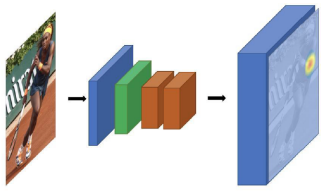
10.2.1 HPE following detection (top-down)

Top down approaches start from the output of a person detector.

A possibly strategy to address the task is to define the localization of each feature as a regression task.

An alternative approach is the estimation of heat maps.





Some of the challenges for HPE are : the unknown number of people , people can appear at any pose or scale , people contact and overlapping and runtime complexity grows with the number of people.

10.2.2 Bottom-up detection : OpenPose

It first detects parts (keypoints) belonging every person in the image , followed by assigning parts to distinct individuals. Shown below is the architecture of the OpenPose model.

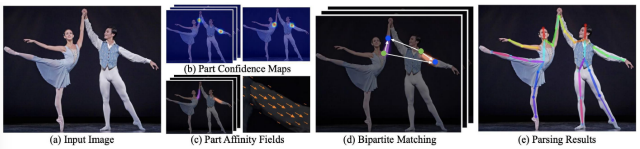


Figure 2. Overall pipeline. Our method takes the entire image as the input for a two-branch CNN to jointly predict confidence maps for body part detection, shown in (b), and part affinity fields for parts association, shown in (c). The parsing step performs a set of bipartite matchings to associate body parts candidates (d). We finally assemble them into full body poses for all people in the image (e).

1. Take the entire image as the input for a CNN
2. Predict confidence maps for body parts detection
3. Predict PAFs for part association
4. Perform a set of bipartite matching
5. Assemble into a full body pose

10.2.3 OpenPose : Schematic Architecture

An input RGB image is fed to a two branch multi-stage CNN.

Two-branches: The top branch , shown in beige, predicts the confidence maps of different body parts location such as the right eye , left eye, right elbow

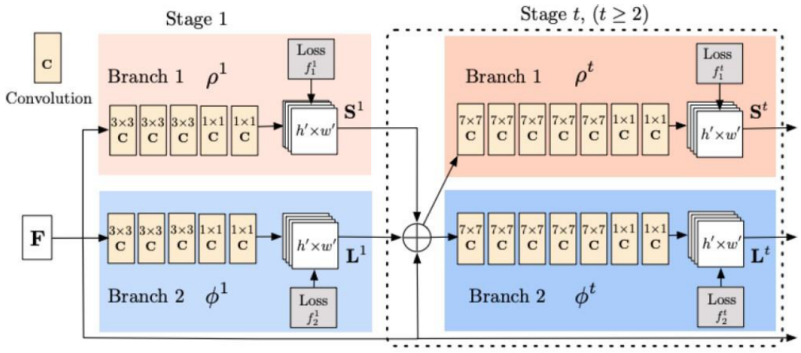


Fig 2. Architecture of the two-branch multi-stage CNN. Image taken from "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields".

and others. The bottom branch, shown in blue , predicts the part affinity fields, which represents a degree of association between different body parts.

Multi-stage: At step 0 an initial estimation of confidence maps and part affinity fields. In the next steps , the image is concatenated with the initial estimations to obtain refined estimations.

Finally , a greedy inference algorithm produces the final output, with the identification of the pose.

10.2.4 OpenPose: confidence map

Confidence map is the 2D representation of the belief that a particular body part can be located. A single body part will be represented on a single map. So , the number of maps is the same as the total number of the body parts.

Examples : Computer-generated imagery, medical application , musical applications.

10.3 Evaluation Metrics

Defining with x_p the predicted joint and with x_{GT} the ground truth :

Percentage of Correct Parts - PCP: the distance between two predicted join locations (x_p^1 and x_p^2) and the true limb joint location is less than half of the limb length (L) : $|(x_p^1 - x_p^2) - (x_{GT}^1 - x_{GT}^2)| < L/2$

Percentage of CorrectKey-points - PCK: A detected joint is considered correct if the distance between the predicted (x_p) and the true joint (x_{GT}) is within a certain threshold (thr) : $|x_p - x_{GT}| < thr$

10.3.1 Semantic Feature Detection

We focus here on bottom-up approaches.

Sometimes instead than a full pose estimation , one may stop at the feature detection step.

Why? Depending on the application we may require several/different types of features. Training the feature detector only is significantly more efficient.

10.3.2 Semantic Feature Detection: a semantic segmentation problem

Semantic features detection can be regarded as a semantic segmentation problem.

Classes are the key-points of interest, plus the background.

This is typically an imbalanced problem, since the number of pixels belonging to the background is significantly higher than the ones belonging to the key-points.

Weighted version of the loss function , with a meaningful strategy for weights computation.

We can use the Fully Convolutional Network (FCN) we have seen for semantic segmentation : Unet,LinkNet

A set of annotated data is necessary to train the deep architectures.

Usually, a set of 15 key-points is used for pose-estimation. Annotating 15 points for a high number of frames can be expensive and time-consuming.

The solution is using transfer-learning: ImageNet pre-trained models as a backbone(encoder) for the FCNs