

Appunti Computer Vision and Cognitive Systems

rosario lissandrello

March 2021

Contents

1	Introduction	6
1.1	il progetto	6
1.2	Computer Vision Overview	6
2	ABC of Images and Operators	9
2.1	Working with images	9
2.2	Histograms	10
2.3	Image processing operators	11
2.4	Point operators	11
2.5	Local operators	12
2.6	Linear filtering	13
2.7	Linear filters	16
2.8	Convolutive Neural Networks	16
2.9	Non-Linear filters	17
3	Edges in vision and cognitive systems	20
3.1	Border detection	22
3.2	Methods based on first derivate	23
3.2.1	Regolarization techniques using filtering	24
3.3	Canny Algorithm	25
3.4	Border Tracing	27
4	Templates and shapes	29
4.1	Template with the MSE	30
4.2	Geometry	32
4.2.1	2D Transformations	34
4.2.2	3D Transformations	36
4.3	Geometric traformations Applications	37
4.3.1	Homography and mosaik	37
4.4	Geometric shapes detection	38
4.4.1	RANSAC Random Sample consensus	39
4.4.2	Hough transform	39
5	From Image processing to AI-Based vision	44
5.1	Vision as a cognitive process	44
5.2	Data and Datasets	45
5.3	Datasets: second generation of Pedestrian detection	46
6	Classification and Retrieval	47
6.1	Measure of similarities	49
6.2	Feature vectors	50
6.3	EMBEDDINGS Bag-of words	51
6.4	Local features	52
6.5	Harris Detector for corners	52

6.6	SIFT Scale Invariant Feature Transform	57
6.7	SIFT detector algorithm	57
6.7.1	Scale-space extrema detection	57
6.7.2	Keypoints Localization	60
6.7.3	Orientation Assignment	61
6.7.4	Keypoint descriptor	63
7	The camera model	64
7.1	Image sensors technologies	64
7.2	The physics of light model	64
7.3	The pinhole model	65
7.4	Camera Calibration	69
7.4.1	Parametri Intrinseci	69
7.4.2	Parametri Estrinseci	70
7.4.3	Parametri intrinseci ed estrinseci	71
7.4.4	Methods of calibration	71
7.4.5	Zhang method of calibration	72
7.5	Structure from Motion	73
7.6	Distortion	73
7.7	Vanishing Point	75
8	Computer Vision and Neuroscience	76
8.1	Segmentazione	78
8.1.1	Metodo di segmentazione - Gray level Thresholding	80
8.2	Labeling	83
8.2.1	Pixels distance in images	83
8.2.2	Connessione tra i pixel	84
8.3	Categorie di Labeling	84
8.3.1	Labels Propagation Algorithms	85
8.4	Algoritmi Multiscan	85
8.4.1	Classic Labeling Rosenfeld (1966)	85
8.4.2	Classic Iterative Multiscan Labeling (Haralick 1981)	86
8.4.3	Iterative multiscan Algorithm	86
8.5	Algoritmi 2-step	87
8.5.1	Two scan algorithm + Equivalence table	87
8.5.2	Altri algoritmi 2-Steps	87
8.5.3	Conclusioni	88
8.6	Segmentation of foreground and background (Scontornamento delle Immagini)	89
8.6.1	Mean shift	89
8.6.2	Segmentation by graph-based methods	90
8.6.3	Min Cut	91
8.6.4	Normalized cut graph	92
8.6.5	GraphCut	93
8.6.6	GrabCut	94
8.6.7	Superpixels segmentation	96

8.6.8	SLIC Simple Linear Iterative Clustering	96
8.7	Saliency	98
9	Vision and Motion	99
9.1	Computer Motion	99
9.2	Tracking	100
9.2.1	Tracking Measures	100
9.3	Detection and Tracking	102
9.4	NCC Normalized Cross Correlation	102
9.5	Motion Analysis	103
9.6	Motion Field (descrizione del movimento)	104
9.7	Motion parallax	106
9.8	Motion estimation methods: Direct and Sparse	106
9.9	Optical flow (quantificazione del movimento)	106
9.9.1	Lucas Kanade alorithm	108
9.9.2	Sparse optical flow: Feature based methods (KLT method)	109
9.9.3	Dense optical flow by deep learning: Flownet	110
9.10	Motion estimation on moving objects	111
9.10.1	Segmentation by motion with fixed cameras	111
9.11	Tracking	114
9.12	kalman Tracking	115
10	Data Manipulation	117
10.1	Basic Tensor properties	117
10.2	How to create a tensor	118
10.3	Mapping from logical to phisical rapresentation	119
10.4	Operations	119
10.5	Reduction	119
10.6	Dot products	120
10.7	Tensor Concatenation	121
10.8	Broadcasting mechanism	121
10.9	Indexing and Slicing	122
11	2D Convolution	123
11.1	Dilatation	124
11.2	Grouping	124
12	Pooling, edge detection, template matching	126
13	Morphology, Hough Transform	127
13.1	Morphology	127
14	CNN Convolutional Neural Networks	128
14.1	AlexNet	128
14.2	VGGNet (2014)	129
14.3	GoogleNet (2014)	130

14.4 ResNet (2015)	132
14.5 Complexity Comparison	134
14.6 Other architectures to know	134
15 Automatic Differentiation	135
15.1 Derivatives and Differentiation	135
15.2 Automatic Differentiation	135
16 Detection Architectures	137
16.1 Classification + Localization	137
16.2 Detection as Classification	138
17 Semantic Segmentation	142
17.1 Fully Convolutional	142
17.1.1 Upsampling - Transpose Convolution	143
18 Geometrical Transformations	144
19 Deep Neural Network architectures for Video Understanding	145
19.1 Come fare Video Classification	145
19.2 Operatore per il video classification	146
20 Face Recognition	148
20.1 DeepFace	148
21 Appunti	149
21.1 Lab	149
21.2 Teoria	149

1 Introduction

strumenti utilizzati Python, Pytorch, OpenCV

Research in Computer Vision

- *IEEE Transactions on Pattern Analysis and Machine Learning*
- *IEEE Transactions on multimedia*
- *IEEE Transactions on image processing*

1.1 il progetto

il progetto si consegna almeno 10 giorni prima dell'esame, deve essere fatto da 3 persone e vale il 40 percento del voto finale. Oltre al codice e al materiale bisogna consegnare anche la documentazione

obblighi da rispettare

- nel progetto devono essere presenti gli elementi decisi decisi a priori dalla prof.
- prima di cominciare, l'idea del progetto deve essere accettata dalla prof.

elementi decisi a priori

- usare un operatore di processamento immagini classico
- utilizzare almeno un algoritmo geometric-based (correzione della distorsione, ecc.)
- utilizzare un retrieval algorithm
- utilizzare un componente Deep learning-based definito e trainato da noi

1.2 Computer Vision Overview

Datasets datasets are very important in computer vision to train networks but also to measure & increase performances (most important thing). We will use only RGB cameras, not lidar or sensor. We will discuss images and the concept of cognitive.

Accountability (responsabilità). Who's the accountant of some automated machine fault?

IEEE is interested in transactions on Pattern Analysis and Machine Intelligence, transactions on multimedia and transactions on image processing. CVPR, ICCV, ACC and ECCV are four of the most important CV foundations.

Computer Vision uses Deep Learning but is much more.

SEER is a billion-parameter self-supervision computer vision model that can learn from any group of images on the internet, is an internal library (only for facebook not open-source), ma quando vogliono rilasciano parte della libreria (VISSL vision library for state-of-the-art- Self-Supervised Learning) open-source per la ricerca.

Artificial Intelligence Definition Artificial Intelligence is the study of how make computers do things at which, at the moment, people are better. (per la prof non è vero al momento: l'IA non soltanto copia ma fa meglio degli umani)

European definition of AI AI refers to systems that do 3 things:

1. They can analyse the environment
2. They can display the intelligent behaviour
3. They can take actions (physical and software)

with some degree of autonomy (non imita gli umani ma è sotto il controllo degli umani*) to achieve specific goals. *nel g20 di quest'anno si è discusso di centralizzare l'AI sotto il controllo umano

Computer Vision Definition is the science (technology) providing computers with VISION capability

New Computer Vision Definition Computer Vision is becoming the science will provide computers with VISUAL INTELLIGENCE

Vision use intelligence to understand the world from visual stimuli

Visual Intelligence it is a cognitive capability to think and reason through mental images. *La funzione principale di un cervello visivo è attivare i processi per acquisire la conoscenza del mondo intorno a noi*

Vision vs. Visual Intelligence vision: due persone e un pallone, visual intelligence: due persone e un pallone -due persone che giocano a calcio

AI The scientific field which studies how to create computers and computer software that are capable of intelligent behavior, using Sensing, Perception, Knowledge, Reasoning and Learning.

Machine Learning The scientific discipline studying how to construct algorithms that can learn from and make predictions on data , for getting computers to act without being explicitly programmed.

Deep Learning A branch of Machine Learning for modeling and implementing deep neural network architectures and algorithms.

Pattern Recognition older term (too much generic), is an applicative filter that have a scope: recognize patterns, pattern can be a face, a song or an action, using a priori knowledge, statistical information and learning

Computer Vision the scientific discipline studying how computers can perceive and understand the world through visual data (and provide a visual intelligence). Computer vision use ML, DL; pattern Recognition, ecc...

Cognitive Computing important definition, better term than AI because *AI significa tutto e non significa niente*. describes technology platforms that, broadly speaking, are based on the scientific disciplines of Artificial Intelligence and Signal Processing . These platforms encompass machine learning reasoning natural language processing speech and vision human computer interaction dialog and narrative generation and more (By IBM 2011) (*rec. 10/03 10:34*)

Image (Video) analysis The scientific discipline studying how to automatically process images and videos to extract visual information.

Image Processing Is not computer vision, modify pixel that have some properties and study how to modify images (es. filtering, denoising, compressing)

Imaging a differenza dell' Image Processing c'è una modifica manuale da parte degli umani (es. photoshop)

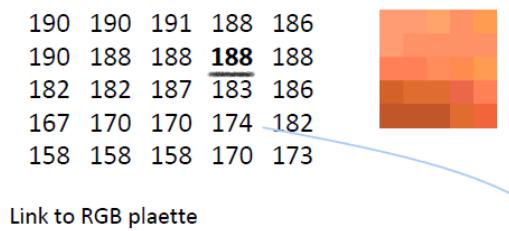
Machine Vision more related to industry, the engineering field studying how to build computer vision based systems.

2 ABC of Images and Operators

2.1 Working with images

We can consider an image like a **discrete representation of a 2D continue function $I(x,y)$** , thus image processing is a 2D extension of the 1D theory of **signal processing**. Or we can consider an image like a 2D matrix of digital pixels. According to definition that we do, we can apply different algorithms depending of the situation.

RGB and Palette If we see the image like a matrix of numer, each number is a pointer to the correspond row of the palette table, each row have 3 column: RGB, Be careful not to loosen the continuity of the signal. do not use palettes in image processing: we have to work with the RGB values (the palette table) and not with the pointers



	R	G	B
...			
Palette:	186	1.0000	0.6118 0.3216
	187	1.0000	0.5490 0.3529
	<u>188</u>	<u>1.0000</u>	<u>0.5686</u> 0.4000
	189	1.0000	0.6353 0.3255
	190	1.0000	0.6118 0.4510
	191	1.0000	0.6471 0.4196

Figure 1: palette pointers and palette table

Image resolution If it's not specified, in image processing we consider the **spatial resolution**. Resolution in pixel (the number of pixels) is not related with the dimension of the object.

Intensitive (brightness) images the dimension of the possible color (variation of the number) that can rappresent pixels; each pixel is a measure of the luminance intensity of the point in the 3d scene

Photometry is the discipline that study the brightness.

Monochromatic images they represent only brightness values.

Color or multispectral images they represent more spectral components: N=3 images in color spaces or M images in multispectral dimensions.

Colorimetry studies the wavelength and the color emission.

2.2 Histograms

The histogram at gray level is a vector with as many elements (bins) as the number of gray levels; the value of each bin is the accumulation of the number of pixels which, in that image, assume the correspondent gray level. In color images, there are 3 histograms, or a combined histogram cv2.

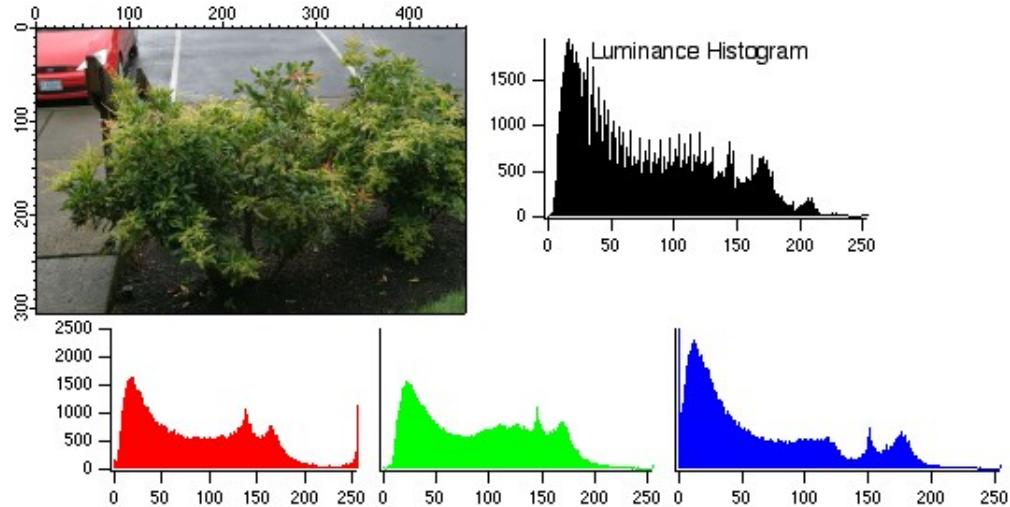


Figure 2: histogram example

Just watching the histogram we can extract many informations of brightness, for this, **histogram can be used as a feature vector**, but sometimes not discriminant enough.

Normalized Histogram Histogram can be viewed as a discrete approximation of a probability density function PDF (if the data could be associated with a random distributed variables i.i.d. independent and identically distributed). In order to be a pdf, must be used the Normalized version of histogram (with

the sum of each components equal to 1). The normalized histogram is the probability distribution. If the histogram is a probability curve, we can compute average and standard deviation.

Cumulative Histogram Cumulative histograms are useful and represent the percentage of pixels that are lighter than each value (the probability of having a pixel's value less of i). *Percentuale di pixels sotto la soglia*

Entropy The image entropy specifies the uncertainty in the image values. It measures the averaged amount of information required to encode the image values. Entropy is a measure of histogram dispersion

$$H(X) = - \sum p(X) \log p(X)$$

Adaptive Histogram In many cases histograms are needed for local areas in an image (e.g. pattern detection), for this adaptive histogram take in account only some parts of the image and for each of them construct the histogram.

Saturated arithmetic Saturated arithmetic is a version of arithmetic in which all operations such as addition and multiplication are limited to a fixed range between a minimum and maximum value. Saturated arithmetic is used especially in imaging when the result of the image processing should be another image , visible for human (e.g. the Instagram, Snapchat etc effects)

2.3 Image processing operators

Image processing uses some operators on image pixels which transform images into other images

- **Point operators** A task of image processing where each pixel is processed separately: the value of each pixel of the resulting image depends only on the original pixel in the same image's spatial position (e.g. Thresholding)
- **Local (neighborhood operators)** When the value of each pixel depends only on the original pixel in the same image's position and in a local Neighborhood (e.g. filters)
- **Global operators** When the value of each pixel depends on all the pixels of the original image (e.g. Fourier transform)

These 3 operators are implemented and parallelized in 3 different ways.

2.4 Point operators

Linear Point Operator

$$g(x) = h(f(x))$$

x is the D Dimensional Domain of the function ($D=2$ for images); $h()$ is the operator which transforms an image to another image after an image processing operation. If the $h()$ transformation is linear it can be written as:

$$g(x) = h(f(x)) = sf(x) + k$$

- s is the scale factor, often called also gain or contrast.
- k is the offset constant often called also bias or brightness.

Sometimes s and k could be dependent on the position $s(x)$ and $k(x)$. They are linear operators since they obey the superposition principle:

$$h(f_0 + f_1) = h(0) + h(f_1)$$

NB: cambiando contrasto e luminanza stai cambiando i dati dell'immagine! Per questo solitamente è utilizzato per modificare le foto con lo scopo di migliorarle a livello visivo, o alcune volte quando si vuole abbassare la luminosità o avere più contrasto per rendere più efficienti gli algoritmi che si andranno ad applicare successivamente, bisogna tener conto però che l'immagine finale ha un contenuto semantico diverso rispetto all'immagine iniziale e che non è possibile tornare allo stato precedente.

Contrast-stretching Is a simple operator that take the histogram and expand him, in order to do this the simplest way is use a combination of linear point operators working with saturated mathematics.

Equalization (isn't useful for CV), change images for better perception. we apply a non linear operation to the image, computers don't like it :(It tries to linearize *cumulative histogram*. Be careful: equalization maximizes entropy, gives more "information" but change information.

2.5 Local operators

Filtering It's a typical local operation that work not only with the single pixel but also with the neighborhood pixels (3x3,5x5,...)

Noisy In CV, there is more definition of noise; It is everything that is not a signal or It is everything is not an useful information. A general definition can be: **Noisy is a random variation of brightness or color information**. We can define 3 types of Noise:

- **Signal Noise** In general, but not necessarily, noise is an *additive noise*
- **Computational Noise** It is an error produced by computational tasks with approximations or computational limitation
- **Perceptual Noise** Everything in the image which is NOT the target of the image itself; sometime called semantic noise.

Signal Noise

- **Salt and Pepper Noise** Random occurrence of black and white pixels saturation.
- **Impulse Noise** Random occurrence of white pixels saturation.
- **Gaussian Noise** Variations in intensity drawn from a Gaussian normal distribution

Reduce Noise The simplest idea is to use smoothing process such as the average of the values, let's replace each pixel with an average of all the values in its neighborhood. This type of operations is a linear filtering.

2.6 Linear filtering

Definition 2.1 (Linear Filtering) Given an initial image F , **linear filtering** consist in a process which gives in output a new image G , where each location is a weighted sum of the original pixel values from the locations surrounding the corresponding location in the image, **using the same set of weights each time**

the result is:

- **Shift invariant** The value depends on the pattern in an image neighborhood but it is not depending of the position
- **Linear** the output for the sum of two images is the same as the sum of the outputs obtained for the images separately

The pattern of weights used for a linear filter is usually referred to as the **kernel** of the filter. The process of applying the filter is usually referred to as **correlation** or **convolution**.

Convolution Convolution was not invented in ML, it comes from mathematics and signal theory and it is an operator from 2 functions. It is defined as the integral of the product of the two functions after one is reversed and shifted. Is also a *commutative operator*.

$$f(t) * g(t) := \int_{-\infty}^{+\infty} f(\tau)g(t - \tau) d\tau$$

It can be applied with a discrete sum of a function $g(n)$ and a filter $f(m)$, this is called **discrete convolution**

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$$

when we use the sign + it is called **discrete correlation**

$$F \circledast I(x) = \sum_{i=-N}^N F(i)I(x+i)$$

Given a signal $I(x)$, where x is a variable value, e.g. time or the space, the application of a Filter F that has a variability between $-N$ and N is given by the Dot product or (cross)correlation. This is valid in 2D too:

$$F \circledast I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j)I(x+i, y+j)$$

Local operators or neighborhood operators can be used as a **linear filter**. Given an image $f(i,j)$ from F a linear filter produces the output G where the $g(i,j)$ values as the weighted sum of the input pixels $f(i,j)$, weighted with a **kernel H** (*kernel, mask or filter have the same meaning*), having $h(k,l)$ as **filter coefficients**

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$$

It is called **correlation**, borrowed by math. Often we uses its variant with the instead of + that is called **convolution**, borrowed by signal processing.

Border problem What can we do with the incorrect data in the borders? Many possibilities of padding where the correct information is not available:

- Zero padding: insert pixels with value 0
- Constant padding: insert a specific color in the border
- Clamp to edge: repeat the edge value
- Wrap: loop around in a toroidal configuration
- Mirror: reflect the edge

Mean filter The averaging operator is extremely basic. It averages the values of pixels based on their neighbouring pixels. It corresponds to convolving the image with a kernel of 1 values and then scaling (normalizing to sum 1). Averaging removes noise from images, but fine details are also lost in the process. With a large enough template size all the detail is lost. Lo smoothing non funziona quando l'errore da correggere è di tipo salt and pepper.

Gaussian filter The Gaussian Smoothing Operator performs a weighted average of surrounding pixels based on the Gaussian distribution. It is used to remove Gaussian noise and is a realistic model of defocused lens. It is an isotropic mask given by a Gaussian function with **zero average** (*tanto bianco quanto nero*) value and a **given standard deviation**, convolved with the image. This filter must be convoluted to smooth the image and filter away noise in a low-pass wise. **Sigma defines the amount of blurring:** lower sigma apply limited smoothing, larger sigma apply stronger smoothing. High sigma values require significantly more calculations per pixel and blurring ruin the original image but make it easily managed by the computer.

	1	2	1
$\frac{1}{16}$	2	4	2
	1	2	1

Figure 3: Il filtro gaussiano più utilizzato

in 1D:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}$$

in 2D:

$$G(x, y) = G(x)G(y) = \frac{1}{\sigma^2 2\pi} e^{-(x^2+y^2)/2\sigma^2}$$

Ovviamente il filtro deve essere discretizzato scegliendo la sua dimensione k e la sua deviazione standard. Avremo così una maschera kxk dove solitamente k e σ sono scelte in modo tale che k sia circa 5σ . In questo modo si coprono circa il 99% dei valori della distribuzione che stiamo considerando. Solitamente, per ragioni computazionali, si utilizzano $\sigma=1$ e $k=5$.

Eg $\sigma=1$ $k=5$

$h =$	0.0029	0.0131	0.0215	0.0131	0.0029
	0.0131	0.0585	0.0965	0.0585	0.0131
	0.0215	0.0965	0.1592	0.0965	0.0215
	0.0131	0.0585	0.0965	0.0585	0.0131
	0.0029	0.0131	0.0215	0.0131	0.0029

Best values	
$\sigma=1$	7×7
$\sigma=2$	13×13
$\sigma=3$	19×19

Nell'esempio, i valori del filtro h sono i valori per x e y compresi fra -2 e +2 (essendo k=5) della distribuzione gaussiana avente $\sigma=1$. I pesi che vanno a

comporre il kernel sono campionati dalla funzione gaussiana calcolando tale valore per diversi valori assegnati a x e y. Ad esempio se la maschera è 3x3 x e y si suppongono compresi fra 1 e -1.

Mean vs Gaussian filter Avarage operator limits the information with the same spatial frequency (**blurring**) and does not work on salt-pepper noise, also is not isotropic, instead Gaussian operator is Isotropic (so it is better). (*l'isotropia è la proprietà dell'indipendenza dalla direzione*)

2.7 Linear filters

Properties of linear convolution

- **Commutativity** source image and filter can be inverted in the order
- **Linearity** it remains valid if we add a scale factor
- **Associativity** we can use many filters in a cascade or a composition of filter
- **Separability** the kernel H can be represented as the convolution of multiple kernels and can separated in a pair dimensional kernel x and y.

Sharpering Is a linear filter used for accentuates differences with local average subtracting values in the border of the local image and adding values on the pixel in the center of the filter.

Technique that consist in applying 2 filters, one that emphasizes the central pixel and the second one that apply an average and then subtract them.

$$\begin{bmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & 1 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{bmatrix}$$

Figure 4: sharpering filter example

Unsharpering Useful for emphasizing transitions in image intensity (e.g., edges).

2.8 Convulsive Neural Networks

The 5 (important) rules of CNN

1. **Receptive field** Neurons do not receive all inputs, but only the ones in a given neighborhood

2. Each neuron uses the same set of parameters, the same kernel, as filtering to create an output image in this way:

$$a(X) = \sum_{i=1}^n w_i x_i$$

3. Many different kernel can be used in parallel to create different output images, often called **feature maps** (o feature vectors quando sono compressi). Possiamo dire che i feature maps sono i risultati dell'applicazione di ognuno dei diversi filtri sull'immagine.
4. Feature maps are processed in a non-linear manner (by non linear functions) poi compresse da pooling per lavorare in multirisoluzione o per allargare il receptive field (la porzione di immagine processata dal neurone). Le funzioni non lineari servono per prendere le decisioni.
5. Feature vectors can be used for **final inference** in order to take a decision. I feature vectors sono usati per fare inferenza finale, ovvero per mettere insieme le decisioni prese fin ora utilizzando un layer fully connected che si occupa della classificazione. Questo viene fatto anche per ridurre l'overfitting.

The 4 important features of a good CNN I filtri che usiamo nella rete possono essere impostati manualmente quando sappiamo su cosa stiamo lavorando e sappiamo cosa è meglio usare per il nostro obiettivo, oppure possono essere imparati se non sappiamo nulla. Se i parametri w sono imparati gli aspetti che possono influire sull'apprendimento sono:

1. Architecture NN
2. Loss function (*la funzione che decide se un risultato è ottimo o no*)
3. Annotated data
4. GPU

2.9 Non-Linear filters

Ci sono certe occasioni in cui bisogna prendere decisioni non lineari (come si/no, sostituire un dato o usare una soglia) e in questi casi si utilizzano filtri non lineari. I principali filtri non lineari che vedremo sono **min max filter** e **median filter**.

Min-Max filter è un filtro non lineare che restituisce rispettivamente il valore minimo e il valore massimo tra i valori dei pixels della porzione R di immagine che stiamo considerando. Un filtro di questo tipo è ad esempio il max pooling.

Median Filter It is useful for impulsive noise, It replaces the pixels with the median value of the neighborhood (receptive field). *Se ho un punto nero in uno spazio bianco, con il filtro gaussiano andrei a distribuire il punto nero nell'intorno, invece con il filtro mediano viene sostituito applicando la mediana.*

1. Consider a 2D neighborhood
2. order it
3. choose the central value
4. substitute pixel with the median one

It is possible to compute the **Weighted median** per dare più peso a certi pixels piuttosto che ad altri. I pesi w possono essere fissi o possono dipendere dalla distanza dal pixel centrale della porzione.

$$g(i, j) = \min_{k,l} \sum_{k,l} w(k, l) |f(i + k, j + l) - g(i, j)|^p$$

Variable-valued filters in Variable-value filters kernel value are **not** fixed. Non applico lo stesso filtro a tutta l'immagine ma cambia in base alla posizione in cui sono. Il principale filtro di questo tipo è il **bilateral filter**

Bilateral filter Bilateral filter is a variable-valued filter and it is used in Photoshop because avoid the noise but preserve the borders (*rimuove la texture ma mantiene i bordi*).

$$g(i, j) = \frac{\sum_{k,l} f(k, l) w(i, j, k, l)}{\sum_{k,l} w(i, j, k, l)}$$

$$w(i, j, k, l) = \exp \left\{ \left(-\frac{(i - k)^2 + (j - l)^2}{2\sigma_d^2} - \frac{|f(i, j) - f(k, l)|^2}{2\sigma_r^2} \right) \right\}$$

- The domain kernel take into account the position of the pixels.
- The derivative (or range) kernel the distance between the pixels.

Il bilateral filter è un filtro in grado di ridurre il rumore mantenendo però i bordi nitidi e non sfocati. Questo è ottenuto grazie al fatto che ogni pixels è sostituito dalla media pesata dei pixels nel suo intorno e i pesi sono scelti sia in funzione della vicinanza con il pixel centrale di riferimento, secondo l'idea che il peso sarà più alto per i pixels più vicini e più basso per quelli più lontani, che in funzione della similarità con il pixel centrale secondo l'idea che più i pixel sono simili più sarà probabile che l'informazione che mi forniscono sia rilevante (un bordo). Più i pixels sono simili più il peso è alto, più sono diversi più è basso. Si può dire che il bilateral filter è un low pass filter in grado di rimuovere il rumore e mantiene i bordi, ovvero mantiene la derivata, che si ottiene combinando un kernel contenente valori (pesi) in base alla distanza spaziale e un kernel contenente valori (pesi) sulla base della similarità cioè della distanza in termini di colore.

Deblurring Il deblurring è l'elaborazione che consente di ricostruire l'immagine originale partendo da un'immagine sfocata. La sfocatura (blur) è la degradazione della nitidezza e dei contrasti nell'immagine dovuta alla perdita delle alte frequenze e può essere dovuta sia alla convoluzione con un filtro per rimuovere il rumore che a instabilità o perdita di messa a fuoco della telecamera. When we apply a filter to an image, we can not return to the original image because we adding noise, motivo per cui è molto difficile ricostruire l'immagine di partenza. Se il noise fosse uguale a zero potremmo semplicemente fare la convoluzione con il filtro inverso e otterremmo l'immagine di partenza: Tuttavia in generale il nuovo noise non è mai uguale a zero per cui se si applica il filtro inverso all'immagine sfocata senza eliminare l'additive noise presente si rischia di propagarlo notevolmente un'immagine disastrosa.

$$Y = x * k + n$$

Definition 2.2 (Blur) *Degradation of sharpness and contrast of the image, causing loss of high frequencies. Technically- convolution with certain kernel during the imaging process*

Super resolution is an image processing task that produce an image. It uses the same network of deblurring.

Impainting leave the foreground in order to have the reconstruction of background

Visualizing intermediate results EX-AI EX-Ai means explainable AI, and it is the possibility to understand the results of...

3 Edges in vision and cognitive systems

What is Computer Vision Computer vision is the discipline studying

- how to create models and algorithms to allow computers to see
- how to create computing systems capable to see
- how to create intelligent systems capable to see

2.5D its the ability of a human being to recognize what is in foreground and in background from a 2D image. Use images to do feature extraction and then use it to clustering the objects.

Image processing pipeline

1. **Rielaborazione dell'immagine al fine di ottenere qualcosa di più facilmente analizzabile:**
 - (a) Source
 - (b) Filtering
 - (c) Edge detection and selection
 - (d) Segmentation
2. **Image analysis** Labeling e feature extraction
3. **Geometria** Camera calibration, estrazione della posizione, distanza tra gli oggetti
4. **Classificazione** Solitamente la classificazione avviene in modo non supervisionato raggruppando gli oggetti sulla base del loro feature vector.

Visual choice criteria To design a visual systems, you first analyze **what** do you want to see and **which** are the visual descriptors you need. Feature is useful to summarize the visual content with an n-dimensional vector of data. A good criteria to choose the feature is take into account 4 properties:

- **Discriminant properties** Features must assume values that are significantly different for objects belonging to different classes.
- **Reliability property** Features must assume values that are similar for objects belonging to the same class.
- **Independent property** Features must be independent. If two features appear always together and in the same context, then one is enough.
- **Minimum cardinality property** Features must be as few as possible

Gestalt psychology or Gestaltism Is a theory of mind of the Berlin School. Gestalt psychology tries to understand the laws of our ability to acquire and maintain meaningful perceptions in an apparently chaotic world. La Gestalt-Forma rappresenta l'attitudine a organizzare le sensazioni elementari in figure emergenti da uno sfondo. Si ottiene, in questo modo, una figura dai contorni dettagliati, che affiora in maniera netta rispetto a uno sfondo indifferenziato, che in alcuni casi appare impercettibile. Gli psicologi della Gestalt hanno identificato specifici principi di organizzazione percettiva:

- **Prossimità, vicinanza o similarità spaziale** All'interno di una stessa scena o immagine, gli elementi vicini tra loro vengono percepiti come un elemento unitario.
- **Similarità** Il cervello umano tende a mettere in relazione oggetti simili tra loro in modo automatico. Li può classificare per forma, per colore o per dimensione.
- **Continuità** : Elementi simili per forma, colore o dimensione, posti uno dietro l'altro (anche secondo schema casuale) danno un'idea di unitarietà molto forte.
- **Proprietà invariantiva rispetto alle operazioni geometriche** Il valore della feature non deve cambiare in seguito a scaling (traslazione in scala), contrazione geometrica, espansione, dilatazione, riflessione, rotazione, taglio, trasformazioni di similarità e tutte le loro combinazioni.
- **Proprietà di rilevanza soggettiva** esprime l'indipendenza della feature rispetto la variazione di luminosità, rispetto allo sfondo, rispetto al campo visivo, ecc.

The importance of EDGES We can recognize just only using simple line drawings without shading or color. Computing the border is very difficult.

Edge Is a local property of a pixel and its neighborhood to have a *Rapid intensity variation*. **Edge it is a VECTOR with a magnitude (il modulo) and a direction**. It depends on the luminance variation. We can compute this luminance variation as a gradient (L'idea è quella di verificare se c'è o meno una continuità dei valori dei pixels nell'intorno derivando la funzione e studiando il comportamento della derivata/gradiante). The edge has the direction perpendicular to the gradient direction. La derivata di un segnale denota la sua variabilità. A fronte di forti variazioni locali (i.e. contorni e altri bruschi cambiamenti di intensità) la derivata assume valori elevati mentre se il segnale è costante la derivata è zero. Nel caso di segnali bidimensionali (come le immagini), si devono considerare le derivate parziali lungo asse x e y. Il gradiente è il vettore le cui componenti sono le derivate parziali nelle diverse direzioni (2 nel caso di immagini). **The border is a property of a Region while Edge is a local property**. We can compute borders by selecting the high/strong edges. There are many parameters to detect an edge:

- Surface color discontinuity
- Illumination discontinuity
- Surface normal discontinuity
- depth discontinuity

3.1 Border detection

To detect a border, the most simple algorithm is:

1. Use of an edge detection operator (edge detector) to find the discontinuity
2. Selection of strong edges with some given criteria
3. Linking the edges (labelling)

Edge Detection In order to find an edge, we have to find the local with most variation (of luminosity in this example). The point with most variation is given by the **highest gradient**, another method is considering where the first derivate is equal to zero or also we can compute the second derivative (the Laplacian) and the point with most variation is the point where the secondary derivative changes from positive to negative (or vice versa).

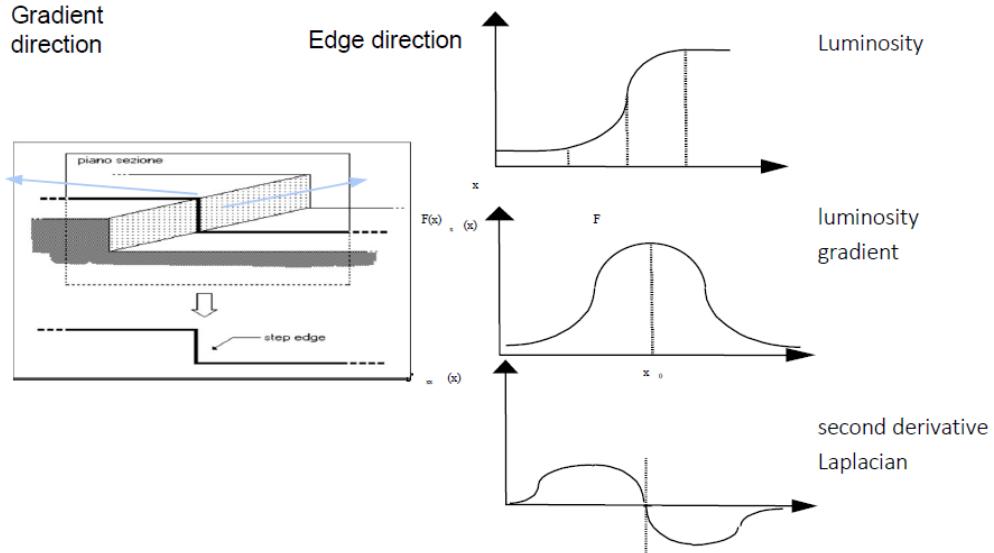


Figure 5: edge ideal

There are many algorithms of edge detection:

- **Methods based on first derivate** In genere l'individuazione dei bordi si avvale di filtri derivativi. Es: Sobel.
- **Regolarization techniques using filtering** Tecniche sempre basate sul primo metodo con in aggiunta tecniche di regolarizzazione che usano filtri e maschere ottimali. Es: Canny.
- **Border following** doesn't use the concept of derivative.
- **Classification** using deep learning

3.2 Methods based on first derivate

- **Image gradient:** Per prima cosa si calcola il gradiente e dopodichè si ricavano direzione e modulo (magnitude).

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- **Gradient direction:** ricordiamo che la direzione dell'edge è perpendicolare al gradiente

$$\Theta = \arctan\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

- **Gradient magnitude:**

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Discrete Derivative In realtà le immagini sono funzioni discrete, non continue, quindi dobbiamo trovare il modo per calcolare la derivata di una funzione discreta. Ci sono tre possibili approssimazioni che possono esse scelte:

- Forward difference:

$$\Delta_h[f](x) = f(x + h) - f(x)$$

- Backward difference:

$$\nabla_h[f](x) = f(x) - f(x - h)$$

- Central difference:

$$\delta_h[f](x) = f(x + \frac{1}{2}h) - f(x - \frac{1}{2}h)$$

It's possible compute the central different using a kernel with the 2 filters 3x3 below, this is equal to compute the derivative.

$$\downarrow y \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow x \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Actually this mask is never used because is too sensitive to the noise. A solution can be use the mask called **Prewitt mask**:

$$\downarrow y \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow x \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Then you can normalize dividing by 1/6. This is better than the original because in the same time while it compute the central derivative in one direction it compute the average in the other

The Sobel Operator However the solution most used is the Sobel operator:

$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

It is similar to the Prewitt mask but instead of apply the average it apply a sort of gaussian filter. (*Un'altra soluzione è il filtro Frei and Chen: è come il sobel ma utilizza la $\sqrt{2}$ per simulare la gaussiana, è meglio del sobel ma nessuno la usa comunque perché computare ogni volta la radice è computazionalmente un'operazione pesante*). Per ogni pixel viene eseguita la convoluzione sia con la maschera Sx che con Sy, dopo di che viene calcolata la magnitudo del gradiente e vengono selezionati come edges soltanto i pixels che hanno gradiente di magnitudo superiore alla soglia. In generale la soglia si impara, è difficile impostarla a mano perché dipende molto dal tipo di immagini su cui stiamo lavorando.

3.2.1 Regolarization techniques using filtering

Ora analizziamo algoritmi di edge detection della seconda tipologia elencata, ovvero quelli che aggiungono tecniche di regolarizzazione e ottimizzazione.

Canny Criteria To find a good edge operator I have to satisfy the 3 criteria:

- **Good Detection** search for low error probability , this criteria is equal to try to **maximize the signal-to-noise ratio**. The signal is everything is in the edge and noise is everything isn't in the edge.
- **Good Localization** I want to have high precion and not found the false edge
- **One Response to Single Edge** I want to have low false positives

3.3 Canny Algorithm

Smoothing In order to reduce the noise and satisfy the first criteria, we can apply a Gaussian and then use the **DOG Derivative of Gaussian**. NB:

$$D * (G * I) = (D * G) * I$$

Quando nell'immagine è presente rumore la derivata prima non è sufficiente per computare i bordi perché è troppo sensibile al rumore visto che il rumore produce pixels che sembrano molto diversi dai loro vicini. Una possibile soluzione sarebbe quella di effettuare uno smoothing dell'immagine prima di calcolare la derivata in modo da ridurre il rumore forzando i pixels differenti dal loro intorno ad assomigliare di più ai pixels circostanti. Tuttavia invece di applicare un filtro di smoothing e poi un filtro derivativo è possibile applicare un solo filtro che esegue entrambe le operazioni. Tale filtro è chiamato DOG e corrisponde alla derivata di un filtro gaussiano. (derivo un filtro gaussiano e poi lo applico). Tuttavia questo non è sufficiente a soddisfare tutti e tre i criteri esposti prima perché comunque non restituisce un solo edge per ogni bordo reale.

1. **Smooth** image I with 2D Gaussian

$$G * I$$

2. **Compute edge magnitude** because i want only the strongest edges

$$|\Delta(G * I)|$$

3. Apply the derivative filter and **Find the normal direction** of each pixel of the edge

$$n = \frac{\Delta(G * I)}{|\Delta(G * I)|}$$

4. **Locate edges by finding the zero-crossings** placing the second derivative equal to 0

$$\frac{\partial(G * I)}{\partial n^2} = 0$$

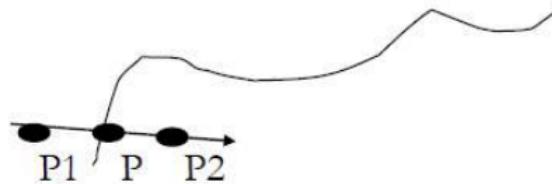
Il punto in cui si annulla la derivata seconda calcolata nella stessa direzione della direzione del gradiente corrisponde al punto in cui la derivata prima ha un massimo.

5. **Non-Maximum Suppression** Tra tutti i punti in cui la derivata seconda è zero, si escludono tutti gli edge tranne quello in corrispondenza del punto massimo della derivata prima.

6. Use the **Hysteresis-thresholding** In order to find only the strongest edges

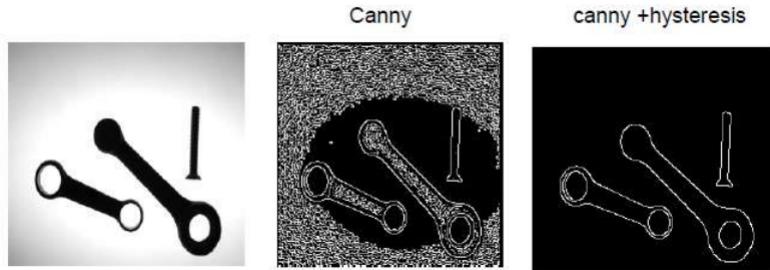
NB: the hyper-parameters in the Canny algorithm are: the dimension of the filter and the strength of the edge that you want to keep.

Non-Maximum suppression Il valore di sigma del gaussian filter definito per lo smoothing è chiamato scala di smoothing. Aumentare la scala di smoothing consente di rimuovere il rumore tuttavia sfoca i bordi. Bordi sfocati portano alla rilevazione di bordi spessi. Poichè quello che vogliamo ottenere noi è un bordo sottile, per ogni pixel candidato ad essere edge compare il suo gradiente con il gradiente degli edge nell'intorno, se il suo è il punto di massimo (ha il valore più alto), allora quello sarà l'edge effettivo, altrimenti viene scartato. Ripeto questo ragionamento per ogni punto candidato ad essere edge.



$$\begin{cases} \text{Grad } (P) \geq \text{Grad } (P_1) \\ \text{Grad } (P) \geq \text{Grad } (P_2) \end{cases} \Rightarrow P \text{ is a valid edge}$$

Threshold with hysteresis Dopo aver effettuato la soppressione dei non massimi gli edges che sono stati selezionati come validi hanno la proprietà di generare delle curve chiuse. Tuttavia fra questi dovrebbero essere selezionati solo i bordi forti. Canny ha proposto l'uso di una sogliatura basata sull'isteresi. Si prendono in considerazione due soglie, una alta T_H e una bassa T_L



$$T_H = 2/3T_L$$

Si seleziona il primo edge che ha un gradiente con un valore maggiore della soglia alta e che siamo sicuri far parte del bordo e poi iterativamente si selezionano gli

edge che hanno un valore maggiore della soglia bassa e che sono collegati (hanno nell'intorno) almeno un punto che ha gradiente con valore maggiore della soglia alta. If you take in an edge point only the points that are higher than the

$$E^{T_L T_H}(i, j) = \begin{cases} 1 & \text{if } G(i, j) > T_L \wedge \exists (k, l) \in N(i, j) \mid G(k, l) > T_H \\ \text{otherwise} & \end{cases}$$

low threshold but also that are in the neighborhood of the gradient of the high threshold you will find only the strongest edges. *The method is iterative.*

Log and Zero crossing It consist in the Laplacian of the Gaussian, it isn't very sensitive to threshold but it is very sensitive to small closed edges and not working well in the place where there is strong discontinuity.

3.4 Border Tracing

Ora andiamo ad analizzare un algoritmo della terza tipologia. Se l'immagine è stata segmentata e binarizzata, dobbiamo solamente codificare i bordi. Andiamo a vedere un algoritmo che lavora su immagini già segmentate e binarizzate.

Chain code A chain code is a lossless compression algorithm for monochrome images.

Freeman chain code The Freeman code is used for edges and for connected shapes. Given a binary image with background =0 (after thresholding, edge detection etc) we take a pixel $p(i, j)$ called **seed point** p_i and we assign a consecutive number for each neighboring pixels:

$$\begin{bmatrix} 0 & 7 & 6 \\ 1 & p & 5 \\ 2 & 3 & 4 \end{bmatrix}$$

If the boundary has at least 3 points each point p_i should have a previous one and a following one p_{i-1} and p_{i+1} connected in a 3×3 region, we can indicate their position with a relative chain code with values between 0 and 7 that are previous and post direction 'pre' 'post'.

1. Given p_i and its pre (initially we can fix an arbitrary pre)
2. To find the following (post) we can explore the 8-connected region starting in anti clockwise from the point after pre. The first point to analyze is 1°
 $\text{pixel} = (\text{pre}(p_i) + 1) \bmod 8$

example:

$$\begin{bmatrix} 0 & 7 & 6 \\ 1 & p & 5 \\ 2 & 3 & 4 \end{bmatrix}$$

- Given $p_i \rightarrow \text{pre}=0, \text{post}=5$.

We surely analyze the external boundary even more pixels are connected by maintaining the counter clock wise search. In general:

- $\text{post}(p_i) = \text{freeman code}(p_i)$
- $\text{pre}(p_{i+1}) = (\text{post}(p_i) + 4) \bmod 8$
- $p_i \leftarrow p_{i+1}$
- Continue until we find the initial point or terminator

Spline There are many splines that taking into account a neighborhood of points and the **Catmull Rom Spline** try to find the best curve that interpolate the curve with a 3 order polynomial function, more points we use, more complex and irregular is the border.

Active contours Active contours are used to find the border and follow it across the time (e.g. in a video)

4 Templates and shapes

Pattern Is something that is very recognizable and understandable. Il pattern è un particolare set di pixels facilmente riconoscibile (più semplice della shape), può essere ad esempio una linea o un cerchio (prima individuo i pattern, sulla base di essi costruisco la shape e poi sulla base della shape ricavo l'oggetto 3D).

Pattern Recognition Techniques Le possibili tecniche utilizzate per il riconoscimento e l'estrazione di pattern all'interno dell'immagine (Pattern recognition) sono:

- **Fitting** Consiste nel trovare il miglior **modello parametrico** che si allinea ai punti dell'immagine. I principali algoritmi usati per fare fitting sono: RANSAC, Linear Regression e Hough Transform.
- **Matching** Utilizzo di un **modello non parametrico** che corrisponde a un pattern. Finding the best correspondence between the points of the model and the reference image.
- **Measuring Similarities** to find and measure the similarities between the objects
- **Asking trained neural network to do it for us**

Template Matching

- **Template** is a precise shape described by pixels, it is generically a windows of pixels agnostic on its semantic
- **Matching** is a generic operation consisting in similarity comparison between entities, regions, vectors of the same type
- **Template matching** is a simple process of image inspection for verifying the presence of a given sub-image that is the template

Recognition is provided by measuring the similarity between template and current image. Il riconoscimento di templates nel concreto può risultare difficile perché si può incontrare nell'immagine lo stesso template ma con colore diverso, dimensione diversa, prospettiva diversa, rotazione diversa, ecc.. A good template matching should take into account **Invariance** to:

- scale
- rotation
- translation
- luminance, etc...



There are different types of template matching:

- **Global template matching** the complete object
- **Local template matching** is a specific

Template Matching vs Object Detection C'è molta differenza tra il matching e la detection. Detection significa trovare qualcosa simile al modello che abbiamo o che abbiamo imparato, matching significa trovare l'esatta (a meno delle trasformazioni sopra) corrispondenza tra modello e immagine.

Mean Square Error and Cross-correlation The most easiest form of template matching is using a template like a sliding window on the image and calculate the similarity. Ci sono due modi per calcolare la similarità, il primo è l'utilizzo del **mean square error** (distanza euclidea tra immagine e template), il secondo metodo è applicare una **cross-correlation**. If the results of the cross-correlation is high, means that the template match with this portion of the image.

4.1 Template with the MSE

If we would like to find the best matching we have to use the MSE. Il MSE misura la distanza euclidea tra il punto dell'immagine e il corrispondente punto del template. Per ogni punto (m,n) si calcola:

$$D_{ij}(m, n) = \sqrt{\sum_i \sum_j (g(i + m, j + n) - t(i, j))^2}$$

Ovviamente più il mean square error è piccolo più alta sarà la similarità, il matching migliore è quello in cui il MSE è minimo. The MSE is 0 only if the pixels are exactly the same (but this never happen because image have noise and luminance difference with the luminance of the template).

MAD and SAD Sometimes to reduce the computational burden are used the simplified versions of the MSE: the MAD and the SAD.

- *Mean Absolute Difference (MAD):*

$$MAD_{ij}(m, n) = \frac{1}{N} \sum_i \sum_j |g(i + m, j + n) - t(i, j)|$$

- and faster *Sum of Absolute Difference (SAD):*

$$SAD_{ij}(m, n) = \sum_i \sum_j |g(i + m, j + n) - t(i, j)|$$

From MSE to Cross-Correlation Possiamo dimostrare che massimizzare la cross-correlation significa minimizzare il mean square error. Starting from the MSE formula we compute an approximation (removing the square root) and we compute the square.

$$E2(m, n) = \sum_i \sum_j g^2(i + m, j + n) - 2g(i + m, j + n)t(i, j) - t^2(i, j)$$

the first term is the summation of the gray level in the image window and the third term is the same but for the template window. **If we suppose that the average of the grey level in the image don't change** we can eliminate the first term and we obtain the cross-correlation formula.

$$R(m, n) = \sqrt{\sum_i \sum_j g(i + m, j + n)t(i, j)}$$

This simple cross-correlation doesn't work so well because the assumption that the average gray level in all the image is a constant is a very strong assumption. So cross-correlation is used to divide the geometric mean of the luminance of image and template

$$N(m, n) = \frac{\sum_i \sum_j g(i + m, j + n)t(i, j)}{\sqrt{\sum_i \sum_j g(i + m, j + n)^2} \sqrt{\sum_i \sum_j t(i, j)^2}}$$

But it is not enough so we use the normalized cross correlation.

NCC Normalized Cross-Correlation

$$N(m, n) = \frac{\sum_i \sum_j (g(i + m, j + n) - \tilde{g})(t(i, j) - \tilde{t})}{\sqrt{\sum_i \sum_j (g(i + m, j + n) - \tilde{g})^2} \sqrt{\sum_i \sum_j (t(i, j) - \tilde{t})^2}}$$

In questo caso viene standardizzato il valore della cross- correlation in base al valore medio del livello di grigio nell'intorno considerato. Dove \tilde{g} e \tilde{t} sono i valori medi del livello di grigio nell'intorno

Summary Riassumendo il template matching è usato in immagini a livelli di grigio o colorate quando so che devo trovare esattamente lo stesso template. Tuttavia questo metodo, così definito, non funziona quando il template sull'immagine differisce dal template utilizzato nel colore, nella forma, nella dimensione, nella rotazione o in trasformazioni affini o prospettiche. In questo caso dobbiamo:

- Trasformare il pattern: ripetere il template per ogni sua possibile rotazione o scaling, oppure fare lo scaling dell'immagine o applicare altre trasformazioni all'immagine.
- Usare descrittori compatti

Ora andiamo a studiare la geometria per capire come fare le varie trasformazioni.

4.2 Geometry

Why using Homogeneous coordinates L'immagine rappresentata in coordinate cartesiane 2D perde informazioni relative al corrispondente oggetto 3D che l'immagine dovrebbe rappresentare. In particolare si perde la rappresentazione di alcune primitive 3D, come la definizione di quali rette parallele si incontrano in un punto all'infinito, come informazioni sulla prospettiva oppure le informazioni sul punto di vista ed un eventuale cambio di punto di vista. Per questo viene utilizzata la geometria proiettiva.

Capire la Geometria Proiettiva La geometria proiettiva è la parte della geometria che modellizza i concetti intuitivi di **prospettiva** e **orizzonte**. Può essere pensata informalmente come la geometria che nasce dal collocare il proprio occhio in un punto dello spazio, così che ogni linea che intersechi l'"occhio" appaia solo come un punto. Le grandezze degli oggetti non sono direttamente quantificabili (perché guardando il mondo con un occhio soltanto non abbiamo informazioni sulla profondità) e l'orizzonte è considerato parte integrante dello spazio. Come conseguenza, nella geometria piana proiettiva due rette si intersecano sempre, non esistono quindi due rette parallele e distinte che non hanno punti di intersezione.

Projective plane Wikipedia *In matematica il piano proiettivo è un'estensione del piano euclideo a cui viene aggiunta una "retta impropria" posizionata idealmente all'infinito e in modo da circoscriverlo. Esteso in questo modo il piano diventa uno spazio compatto in cui anche le rette tra loro parallele si incontrano in un unico punto e tale punto di intersezione è idealmente collocato sulla "retta impropria".*

Homogeneous coordinates Homogeneous coordinates or projective coordinates are a system of coordinates used in projective geometry; The points are projected from a space of dimension n to a space of dimension $n+1$ without the center of the coordination.

- In the 2D case $x = (x, y) \in R^2 \rightarrow \tilde{x} = (\tilde{x}, \tilde{y}, \tilde{w}) \in P^2$
- In the 3D case $x = (x, y, z) \in R^3 \rightarrow \tilde{x} = (\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w}) \in P^3$

Projective Plane Let's suppose to be in a plane in 3D space the points in this space have coordinates $(x, y, z = 0)$, now consider another **parallel** plane π distant in the third coordinate 1 and having the point coordinates $(x, y, z = 1)$. Let's consider to go from 2 to 3 dimensions

$$p = (x, y, 1) \in \pi \rightarrow ((wx, wy, w), w \in R) \in S$$

the point p represent all possible point belonging to the prospective line passing to the center.

Conversion from euclidean to homogeneous coordinates

- from 2D to homogeneous

$$(x, y) \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- from 3D to homogeneous

$$(x, y, z) \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Conversion from homogeneous to euclidean coordinates

- if **w non zero:**

- from homogeneous to 2D

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \rightarrow (x/w, y/w)$$

- from homogeneous to 3D

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow (x/w, y/w, z/w)$$

- if **w=0** no conversion (point to infinity)

Why homogeneous coordinates Because it is easier working on matrix product and makes simpler any computation. Points in homogeneous coordinates are invariant to the scale.

4.2.1 2D Transformations

Translation in 2D a translation is given by:

- $X_2 = X_1 + dx$
- $Y_2 = Y_1 + dy$

In inhomogeneous coordinates there is no matrix T such that $p_2 = T * p_1$. In homogeneous coordinates, translation can be obtained by a matrix product.

$$\begin{bmatrix} X_2 \\ Y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ 1 \end{bmatrix}$$

Translation transformation preserves the orientation and area

Translation and Rotation The transformation that considers both rotation and translation is called the "2D rigid body motion".

$$x' = [R \quad T]\tilde{x}$$

where T is the translation matrix and R is the rotation matrix. R must to be:

$$RR^T = I, \quad |R| = 1, \quad R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Euclidean transformation preserves the length and the euclidean distance

Gradi di Libertà Il numero di gradi di libertà corrisponde al numero di parametri indipendenti necessari per specificare la posizione o il movimento di ciascun corpo nel sistema.

Class I: Isometries

- There are 3DOF (1 rotation, 2 translation)
- All Euclidean transformations are isometries
- Invariants: lenght, angle, area

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \epsilon\cos\theta & -\sin\theta & t_x \\ \epsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \epsilon = \pm 1$$

orientation preserving: $\epsilon = 1$ orientation reversing: $\epsilon = -1$

$$x' = H_E x = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} x$$

2D Scaling In 2D a scaling is given by a factor s

$$\begin{aligned}\overline{x_2} &= \begin{bmatrix} sI & 0 \\ 0^T & 1 \end{bmatrix} \overline{x_1} \\ \begin{bmatrix} X_2 \\ Y_2 \\ 1 \end{bmatrix} &= \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = S * \begin{bmatrix} X_1 \\ Y_1 \\ 1 \end{bmatrix}\end{aligned}$$

Se $s > 0$ l'oggetto viene ingrandito, se $s < 0$ l'oggetto viene rimpicciolito. Lo scaling preserva l'orientazione ma non l'area.

Class II: Similarity transformation 2D scaling + rotation + translation is called **Similarity transformation**.

- There are 4DOF (1 scale, 1 rotation, 2 translation)
- Is equi-form (shape preserving)
- Invariants: ratios of length, angle, ratios of areas, parallel lines

$$\begin{aligned}\overline{x_2} &= \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \overline{x_1} \\ x_2 &= [sR \quad t] \overline{x_1} \\ \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} &= \begin{bmatrix} \epsilon s \cos \theta & -s \sin \theta & t_x \\ \epsilon s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \epsilon = \pm 1\end{aligned}$$

Class III: Affine Transformations A generic Affine transformation relaxes any constraint in the matrix. In 2D it can be written with a generic 2x3 A matrix.

- There are 6DOF (2 scale, 2 rotation, 2 translation)
- non-isotropic scaling
- Invariants: parallel lines, ratios of parallel lengths, ratios of areas

$$\begin{aligned}\overline{x_2} &= \begin{bmatrix} A \\ 0^T \end{bmatrix} \overline{x_1} \\ x_2 &= \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \overline{x_1}\end{aligned}$$

Affine transformation preserves only the parallel lines

$$\begin{aligned}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} &= \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ x' &= H_A x = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} x\end{aligned}$$

Class IV: Perspective Transformations It is the most general and also the most expensive transformation.

- There are 8DOF (2 scale, 2 rotation, 2 translation, 2 line at infinity)
- Invariants: cross-ratio of four points on a line ratio of ratio

$$x' = H_p x = \begin{bmatrix} A & t \\ v^T & v \end{bmatrix} x \quad v = (v_1, v_2)^T$$

$$\tilde{x}_2 = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tilde{x}_1$$

4.2.2 3D Transformations

3D Rotation In 3D a rotation around the Z direction [$P_1=(X_1, Y_1, Z_1)$ becomes P_2]:

$$\begin{cases} X_2 = X_1 \cos\theta - Y_1 \sin\theta \\ Y_2 = X_1 \sin\theta + Y_1 \cos\theta \\ Z_2 = Z_1 \end{cases}$$

The corresponding rotation matrix is:

$$R_Z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With the same method, we can generate the corresponding rotation matrix around X and Y:

$$R_X(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_Y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Scaling

$$\overline{x_2} = \begin{bmatrix} sI & 0 \\ 0^T & 1 \end{bmatrix} \overline{x_1}$$

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = S * \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}$$

Any class II Similarity transformation uses both Scaling, Rotation and Translation in a 3x4 matrix

3D affine transformation In class III affine, parallel lines remains parallel

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{bmatrix} \bar{x}$$

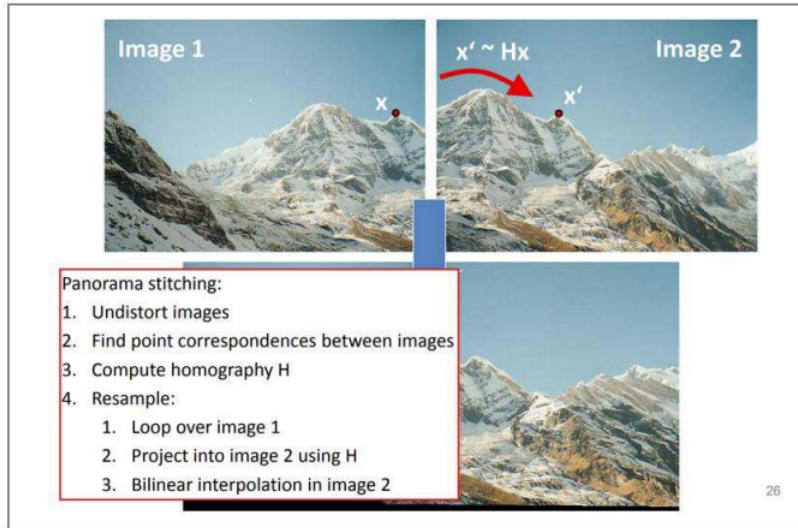
Perspective transformation Class IV Projective transformation (or homography or collineation). Where H is an arbitrary homogeneous matrix

$$\tilde{x}' = \tilde{H}\tilde{x}$$

4.3 Geometric transformations Applications

Principali applicazioni delle trasformazioni omografiche

1. Cucitura di più immagini fra loro



2. Estendere le operazioni di template matching Dati un template e un immagine è possibile forzare l'algoritmo di template matching creando tutte le possibili trasformazioni euclidiane, affini e omografiche e applicandole all'immagine per poi provare a fare il matching con il template

3. Realtà aumentata

4.3.1 Homography and mosaik

To create a mosaik of images, the first thing to do is take 4 points in the first image and the corispectives points in the second image (you can do this manually or using some algorithms), then create a correspondences matrix:

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$h_{00} \dots h_{21}$ are the variables to found

$$\begin{cases} x'_i = h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i = h_{10}x_i + h_{11}y_i + h_{12} \\ 1 = h_{20}x_i + h_{21}y_i + 1 \end{cases}$$

$$\begin{cases} x'_1 = \frac{h_{00}x_1 + h_{01}y_1 + h_{02}}{h_{20}x_1 + h_{21}y_1 + 1} \\ y'_1 = \frac{h_{10}x_1 + h_{11}y_1 + h_{12}}{h_{20}x_1 + h_{21}y_1 + 1} \end{cases}$$

Repeat for the 4 points.

$$\begin{aligned} h_{00}x_i + h_{01}y_i + h_{02} - h_{20}x_ix'_1 - h_{21}x_i &= x'_i \\ h_{10}x_i + h_{11}y_i + h_{12} - h_{20}x_iy'_1 - h_{21}y_i &= x'_i \end{aligned}$$

Caso con 4 punti:

$$\begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0x'_0 & -y_0x'_0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0y'_0 & -y_0y'_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y_1 \\ x_2 & . & . & . & . & . & . & -y_2x'_2 \\ 0. & . & . & . & . & . & . & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{bmatrix} = \begin{bmatrix} x'_0 \\ y'_0 \\ x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

Caso con N punti:

$$\begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0x'_0 & -y_0x'_0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0y'_0 & -y_0y'_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y_1 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ x_{N-1} & y_{N-1} & 1 & 0 & 0 & 0 & -x_{N-1}x'_{N-1} & -y_{N-1}x'_{N-1} \\ 0 & 0 & 0 & x_{N-1} & y_{N-1} & 1 & -x_{N-1}y'_{N-1} & -y_{N-1}y'_{N-1} \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{bmatrix} = \begin{bmatrix} x'_0 \\ y'_0 \\ x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ . \\ . \\ . \\ x'_{N-1} \\ y'_{N-1} \end{bmatrix}$$

4.4 Geometric shapes detection

Fin ora abbiamo visto l'utilizzo di un modello basato su una finestra di pixels più eventuale applicazione di geometria per individuare una determinata shape. Tuttavia è possibile usare anche modelli basati sulla rilevazione di **forme geometriche** come linee, cerchi o ellissi.

4.4.1 RANSAC Random Sample consensus

bello ma pesante dal punto di vista computazionale $O(n^3)$. It is a non deterministic algorithm in the sense that it produces a reasonable result only with a certain probability. L'algoritmo consiste nel considerare due punti e definire la retta che passa per quei due punti. Dopo aver definito tale retta si conta quanti punti seguono quella retta (consenso). Ripeto questo per ogni possibile coppia di punti e la retta che avrà il consenso più alto corrisponderà al mio modello, cioè alla retta che ho trovato nell'immagine.

```

1 Mmax = 0;
2 For all pairs (e1,e2) in E:
3     - find line equation passing through e1 and e2;
4     - M=0;
5     - for all e points in E:
6         if e fits the line (as a thresholded distance):
7             - M++;
8             if M > Mmax:
9                 Mmax=M;
10                store e1, e2;
11 iterative methos; re-calculate the line

```

The set of inliers obtained for the fitting model is called **consensus set**.

4.4.2 Hough transform

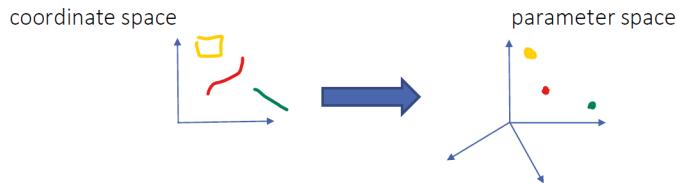
più leggero del RANSAC, solo $O(N^2)$. In general the algorithm is used only on edges, so it is applied after the Edge Detection and it is more lighter than RANSAC. **HT transforms the points on the coordinate space into points of the parameter space where the evidence of a curve is accumulated.**

- **PROS:**

- The Hough transform is tolerant of gaps in the edges
- It is relatively unaffected by noise
- It is also unaffected by occlusion in the image

- **CONS:**

- The HT is able to detect curve and lines but not to localize them.



Approfondimento Hough Transform Algorithm Da wikipedia:

Metodo di Hough [modifica | modifica wikitesto]

Si consideri un punto $\mathbf{a}' = (x', y')$ su un piano bidimensionale. Le rette passanti per tale punto sono $y' = m' \cdot x' + c'$ per ogni m' e c' . Tale equazione descrive una linea nello spazio parametrico m-c. Allo stesso modo, considerando un secondo punto $\mathbf{a}'' = (x'', y'')$, si ottiene che tutte le rette passanti per esso sono $y'' = m'' \cdot x'' + c''$. Ancora una volta, tale equazione descrive una linea nello spazio m-c. L'intersezione di tali linee nello spazio m-c identifica una retta nello spazio x-y che collega i punti \mathbf{a}' e \mathbf{a}'' .

Seguendo questo principio è possibile descrivere i passi per la determinazione delle linee:

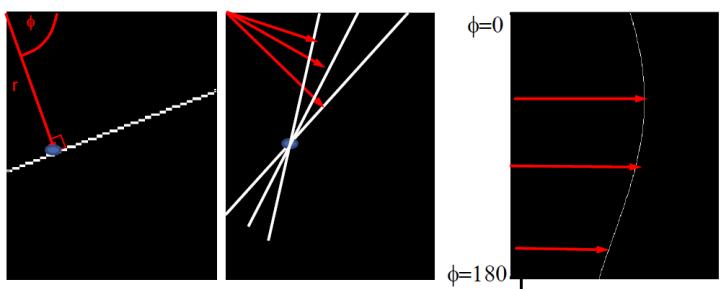
1. Quantizzazione dello spazio m-c sfruttando adeguati valori massimi e minimi per c ed m .
2. Definizione di un array di accumulazione $A(c, m)$ i cui elementi devono essere inizializzati a zero.
3. Per ogni punto (x, y) appartenente all'immagine si determina il gradiente.
4. Se tale gradiente supera un determinato livello si incrementa il corretto elemento dell'array di accumulazione: $A(c, m) := A(c, m) + 1$.

I massimi raggiunti dall'array di accumulazione indicano le linee nell'immagine analizzata.

EHT for straight lines All the points of a line are represented by a pairs of parameters (r , θ):

- the distance from the origin, r (algebraic distance due to the sign that can be negative)
- the angle (θ discretized into θ_n).

We can provide a transformation from the image space to the parametric space where each line is corresponding to a point (The parameter space, considered as a discrete accumulation space, is called **Hough space**). How can we find the point? Each point of the image space votes in the transformation space for all the set of lines passing through that points: **each point votes for a sinusoidal shape of points**



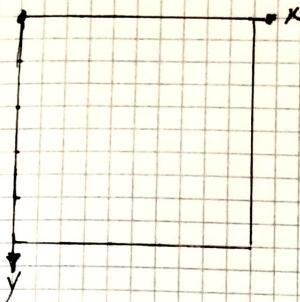
- **NB1:** For image shape detection, using only feature points is better, e.g. edges (thus Edge based HT).
- **NB2:** The $y = mx + c$ representation fails in case of vertical lines (m to infinite, difficult to represent in a discrete manner).

*Ogni edge point si trasforma in una **sinusoide** e se due punti che appartengono alla stessa retta, ho 2 **sinusoidi** che si intersecano.*

EHT Algorithm

1. Quantize the Hough Transform space: identify the maximum and minimum values of r and α (r_{\min} , r_{\max} , α_{\min} , α_{\max})
2. Discretize the Hough space sampling steps D_r , D_α
3. Generate an accumulator array $H(r,\alpha)=0$ for all r and
4. create the Hough space
 - For all edge points $E(x_i, y_i) = 1$ in the image
 - For each angle α in (α_{\min} , α_{\max})
 - Compute r from the equation $r = \text{quantize}(x_i \cos(\alpha) + y_i \sin(\alpha))$;
 - $H(r,\alpha)++$;
5. search the lines
 - For all cells in $A(r,\alpha)$
 - Search for the maximum value of $A(r,\alpha)$ or over a threshold
 - Calculate the equation of the line

EHT - Edge based Hough transform

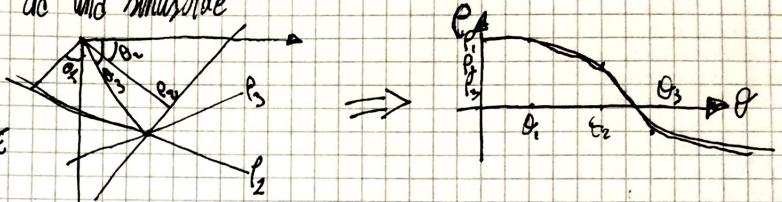


dobbiamo trovare i FEATURE POINTS (i punti più importanti) come ad esempio gli edges

Sappiamo che per un punto passano infinite rette e che l'equazione parametrica di una retta è:

$$\rho = x \cos \theta + y \sin \theta$$

Un punto nello spazio parametrico è rappresentato da una curva

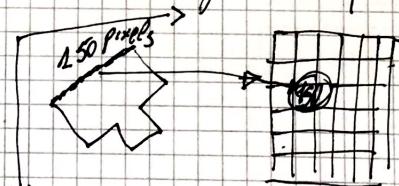


MATRICE DI ACCUMULAZIONE

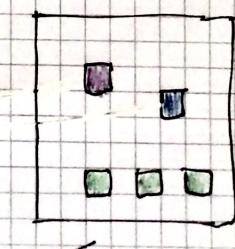
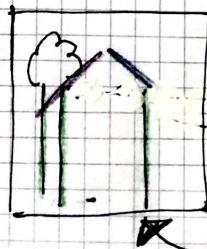
	0°	90°	180°	270°
0°	1	1	1	1
90°	1	1	1	1
180°	1	1	1	1
270°	1	1	1	1

Per ogni pixel consideriamo i tre angoli con cui incontra le rette. Se la sua retta passa per il punto aggiungiamo +1 alla matrice di accumulazione.

* prendiamo in considerazione solo i pixel dei feature points



allo fine del ciclo, la matrice di accumulazione a di 70°, quindi sono le rette più importanti rispetto agli angoli più notati



Gradient-based Hough transform GHT To go faster instead of compute ht for each angle, we can use gradient direction for an (thus a single angle) and increment for a value proportional to the gradient weight (thus higher edges are more important).

HT for circles

Parameters for analytic curves recap:

Analytic Form	Parameters	Equation
Line	ρ, θ	$x\cos\theta + y\sin\theta = \rho$
Circle	x_0, y_0, ρ	$(x - x_0)^2 + (y - y_0)^2 = r^2$
Parabole	x_0, y_0, ρ, θ	$(y - y_0)^2 = 4\rho(x - x_0)$
Ellipse	x_0, y_0, a, b, θ	$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1$

Generalized HT For each possible gradient direction of the points, we list the possible distance to the center. For a given gradient direction, if the objects has a whichever shape, We can have many edges points (with different r).

ALGORITHM



Given the edges (x_i, y_i, ϕ_i) find the object center (x_c, y_c)

1. Form an Accumulator array to hold the candidate locations of the reference point, initialized to 0 for each possible center
 $A(x_c, y_c)$
2. For each point on the edge (x_i, y_i, ϕ_i)
 1. Compute the gradient direction and determine the row of the R-Table it corresponds to, use the correspondent r
 2. For each entry on the row calculate the candidate location of the reference point $x_c = x_i + r \cos \theta$
 3. Increment the Accumulator value for that point $y_c = y_i + r \sin \theta$
3. The reference point location is given by the highest value in the accumulator array

5 From Image processing to AI-Based vision

Nella prima parte di questo capitolo andiamo ad analizzare quali sono state le varie pipeline utilizzate per fare pattern recognition su immagini negli anni e quali sono gli approcci usati anche oggi.

Pipeline di Pattern Recognition classica

1. **Selection of DATA** Dataset, real-time data and if necessary the DATA ANNOTATION (human/machine made, redundancy...)
2. **Data Preparation** pre-processing (Human crafted or AI based)
3. **Feature Extraction** to understand the model and define the inference engine (hand-crafted features or deep learning based)
4. **Model for classification / inference** es: classification, definition of the measure and the Loss
5. **Decision**

Ora “pattern recognition” (che teoricamente sfrutta machine learning ed euristiche definite dall’uomo) e “machine learning” sono sinonimi.

5.1 Vision as a cognitive process

Image retrieval Image retrieval is a large subarea of Computer Vision

- **Statement 1** Vision requires to extract what is characteristic of a target
 - **Features** the characteristics of a target (scene, object, event, key-points, details)
 - **Descriptors** the way we represent them (es: colors and RGB tensor)
- **Statement 2** Vision requires an inference engine to provide reasoning (association, memory, logic deduction...)
 - classification
 - retrieval
 - detection
 - understanding

Vision as a Cognitive process Vision is the task that has its semantics only in the visual stimulus.

- **Predicative construction** It is a verb that induces the generation of knowledge in the subject who sees.
- **Object construction**

5.2 Data and Datasets

PASCAL VOC (2008) Dataset must Know:

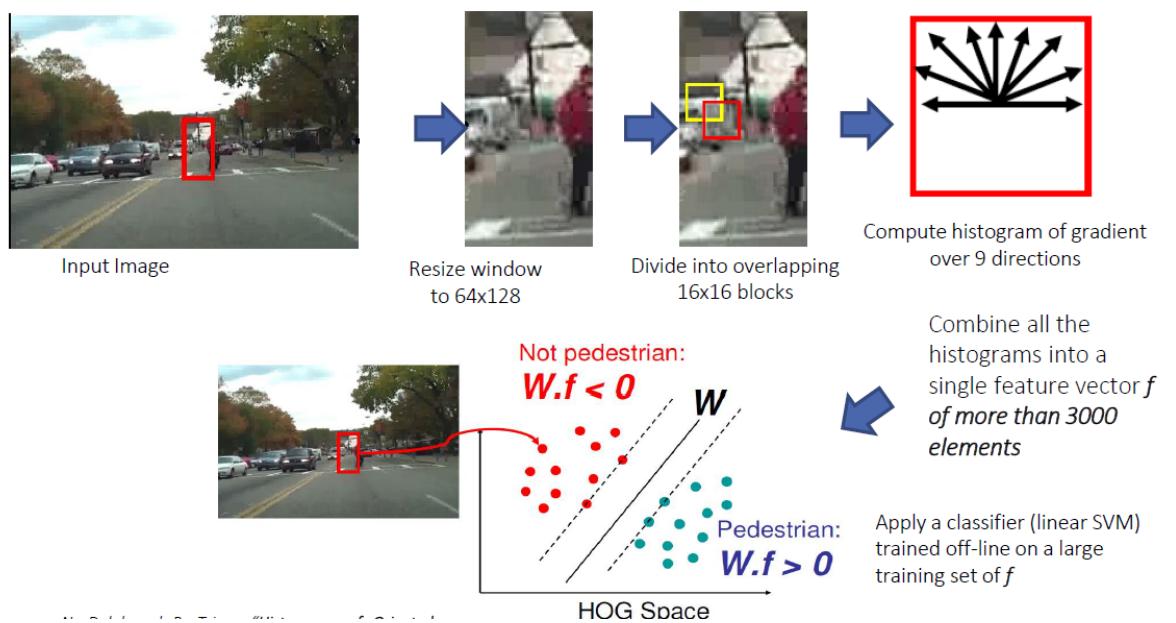
- 20 classes
- 11.000 images
- 27.000 annotation
- A large European Effort

OpenVisor dataset dell'aimagelab per la videosorveglianza

INRIA People dataset (2005) Dataset for people (pedestrian) detection:

- 1237 bounding-boxes
- 614 positive images and 1218 negative images
- know issues with mis-labeled annotation, but still made significant contributions to computer vision

HOG+svm with the first pedestrian dataset (INRIA), the first robust solution.



5.3 Datasets: second generation of Pedestrian detection

Caltech Pedestrian Dataset

- Genesis in 2009
- Approximately 10 hours of 600x400 video taken at 30 frames per second
- Provides two bounding-boxes: full and visible
- 346621 bounding-box labels
- 250000 frames

ImageNet The birth of Deep Learning ERA: The Fei Fei proposal of IMA-GENET.

- Genesis in 2010
- Used in the creation of "Alexnet"
- 485688 object detection training and validation images
- 200 classes for object detection

SUN It contains 4.479 object categories and 313.884 instance segmentation labels in 131,067 images. For people alone, SUN has 6.202 instances of people in 2.062 images.

KITTI

- Genesis in 2012
- Variety of labelling such as tracking, scene flow, odometry and more
- Images from a video, so images share a temporal relationship
- 4500 people labeled in 7500 images

COCO Introduced by Microsoft in 2015, Microsoft Common Object in Context (COCO) is a dataset containing instance segmentation of 80 common objects in their natural context. The term "common" refers to the objects that can be "easily recognizable by a four-year-old.

- Genesis in 2015
- 80 common object classes
- 2.5 million labeled instances
- 380000 images

Summary Datasets for Classification, retrieval detection:

	Pascal VOC	ImageNet	COCO	INRIA	Caltech	SUN	KITTI
Number Images	11,530	485,688	382,000	1,237	346,621	313,884	7,480
Number Annotations	27,450	369,440	2,500,000	1,832	250,00	131,067	4,487 (people)
Anno Type	BBox	BBox	Instance Seg	BBox	BBox	Instance Seg	BBox

6 Classification and Retrieval

Image retrieval è diverso da image classification. L'image retrieval consiste nel trovare gli oggetti simili all'immagine che sto considerando ed è molto complesso perché per fare ciò è necessario definire un modo per rappresentare l'immagine, definire una misura di similarità e calcolare tale similarità.

Retrieval vs Classification

- **Image Retrieval** Given a query image find the same *similar* images in the set
- **Image Classification** Recognize the particular general class to which the image content belongs from a set of possible classes

Performance measures

- **TP: true positives** Number of correct matches
- **FN: false negatives** Matches that were not correctly detected
- **FP: false positives** Proposed matches that are incorrect
- **TN: true negatives** Non matches that were correctly rejected

Precision

$$Precision = \frac{TP}{TP + FP}$$

E.g.: if our model has a precision of 0.5, when it predicts a tumor is malignant, it is correct 50% of the time.

		Real Label		
		Positive	Negative	
Predicted Label	Positive	True Positive (TP)	False Positive (FP)	Precision = $\frac{\sum TP}{\sum TP + FP}$
	Negative	False Negative (FN)	True Negative (TN)	
		Recall = $\frac{\sum TP}{\sum TP + FN}$		Accuracy = $\frac{\sum TP + TN}{\sum TP + FP + FN + TN}$
		$err = \frac{fp + fn}{tp + tn + fn + fp}$		

Recall specific object \leftarrow recall

$$Recall = \frac{TP}{TP + FN}$$

E.g: if our model has a recall of 0.50, it correctly identifies 50% of all malignant tumors.

Precision Vs Recall Ex: If you have to deploy an e-commerce, in you e-commerce your goal is sell something, so it is important that you haven't low precision, because we want optimize user experience. Instead when we have to deploy for example a security application and we have to recognize a specific object it's more important have a high recall.

F-Measure F-measure take into account both measures (precision and recall) in a single number and it is defined like *weighted harmonic mean of precision and recall*

$$F = \frac{2PR}{P + R}$$

E-Measure Is an improvement of F-measure, there is a weight β and you can chose allow more emphasis on precision or on recall

$$E = \frac{(1 + \beta^2)PR}{\beta^2 P + R}$$

Rank List The rank list is the list of the result (the best result is the first) returned by a retrieval algorithm and it is used to understand if the result is the ideal.

R-precision R-precision is the precision at a given relevant cardinality; e.g. if there are 6 relevant element you must look after a rank given to 6. *Se il*

n	doc #	relevant
1	588	x
2	589	x
3	576	
4	590	x
5	986	
6	592	x
7	984	
8	988	
9	578	
10	985	
11	103	
12	591	
13	772	x
14	990	

$$R = \# \text{ of relevant docs} = 6$$

$$\text{R-Precision} = 4/6 = 0.67$$

numero di elementi rilevanti è 6, la r-precision è data da: (numero di elementi rilevanti trovati nei primi 6 tentativi)/6)

Average precision (AP) Is a typical performance measure used for ranked sets. Average Precision is defined as the average of the precision scores after each relevant item (true positive, TP) in the scope S (if not indicated, S by default is 11).

Mean Average Precision (MAP) Average of the average precision value for a set of queries in a given dataset.

6.1 Measure of similarities

The Euclidean distance fa schifo per tanti motivi, l'approssimazione è troppo grande e soprattutto attribuisce ad ogni feature lo stesso peso nonostante sappiamo che ci sono feature più importanti di altre, inoltre computare la radice quadrata è sempre un'operazione pesante.

$$d_{euclidea} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Cosine similarity già meglio della distanza euclidea, anche se però anche qui tutte le feature hanno lo stesso peso. Non prende in considerazione la magnitude delle feature ma almeno considera la distanza tra le feature, se le feature sono

normalizzate è meglio utilizzare la cosine similarity, se non sono normalizzate potrebbe essere meglio la distanza euclidea.

$$s = \frac{x * y}{|x||y|}$$

Bhattacharya Distance Solitamente usata perchè facile da implementare, non perchè sia la migliore

$$\rho(p, q) = \int \sqrt{p(z)q(z)dz}$$

K-L Distance

$$D(A, B) = \sum_{i=0}^{b-1} P_i(B) \log \frac{P_i(B)}{P_i(A)}$$

Ground Distance (Moving earth) Non compara bin a bin ma direttamente gli istogrammi, consentendo così di mettere a paragone bins che sono shiftati nello spazio, da usare quindi quando i vettori sono simili ma sono shiftati. esempio: dati due istogrammi e due pixel identici ma uno più scuro dell'altro.

$$D(I, J) = \frac{\sum_{i,j} g_{ij} d_{ij}}{\sum_{i,j} g_{i,j}}$$

6.2 Feature vectors

Color Histogram the first most easier feature vector is the color histogram, non consigliabile

After histograms Descriptors tre tipi di feaure descriptors:

- **based on the gradient** (very simple)
- **key points vector** based on details (key points) is useful when we want do a detection based on small local features
- **feature vector from neural network** Si può trainare una rete per fare determinate classificazione poi utilizzare la stessa rete già trainata eliminando gli ultimi layer di classificazione per ottenere dei feature vector che verranno classificati in altro modo

Google Search la prima versione di google search usava: *colori, shape e image name.*

6.3 EMBEDDINGS Bag-of words

Three concepts very important:

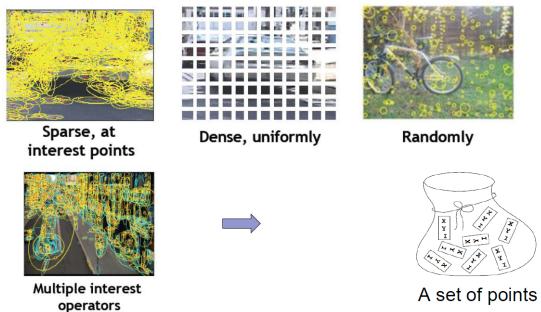
1. The concept of bag of word
2. The use of bag of words with SIFT
3. The use of bag of words with CNNs → Embedding

The concept of bag of word In the past, in order to find the similarity between two texts, there was no need to take all the words and compare them, but it was enough take only the more important, count the occurrences and compare it. An attempt was made to do the same with the images. The images do not have the concept of the word but have the keypoints.

Step Bag of word

1. **Scelta delle possibili visual words** The first step is create our vocabulary composed by the visual words and called **the code-book** through the search and the retrieval based on the content of the image. How to select the words:

- sparse, at interest points
- dense, uniformly
- randomly
- multiple interest operations



2. **Clusterizzare tutte le possibili visual words trovate per trovare le reali visual words** I Cluster Classifier trattano le words come fossero dei vettori numerici. Questi vettori sono chiamati descrittori di caratteristiche. Un buon descrittore dovrebbe avere la capacità di gestire in una certa misura intensità, rotazione, scala e variazioni affini. Uno dei descrittori più famosi è SIFT (Scale-invariant feature transform). SIFT

converte ogni patch in un vettore a 128 dimensioni. Dopo questo passaggio, ogni immagine è una raccolta di vettori della stessa dimensione (128 per SIFT), in cui l'ordine dei diversi vettori non ha importanza.

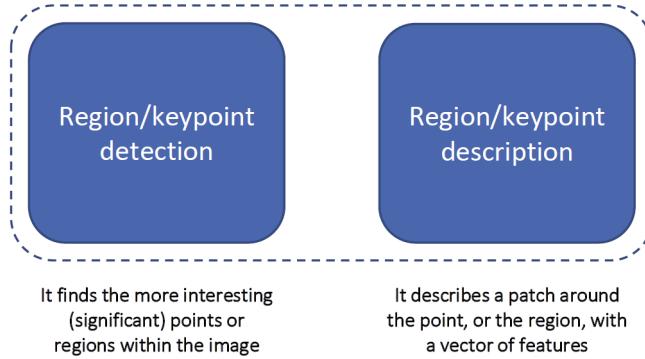
dopo aver creato il cluster of words, per ogni immagine si calcolano i key-points/visual words, si clusterizzano e si mettono a paragone il primo cluster con il secondo per vedere dove cadono i key points. Alla fine tutte le ricorrenze delle words vengono messe in un istogramma → bagging.

step 6: classification, search, recognition by embedding.

6.4 Local features

There are two main approaches to finding feature points and their correspondences. The first is to find features in one image that can be accurately tracked using a local search technique, such as correlation or least squares (Harris). The second is to independently detect features in all the images under consideration and then match features based on their local appearance (SIFT).

- **Region/keypoint detection** define an algorithm to detect keypoints
- **Region/keypoint description** define an algorithm to describe them

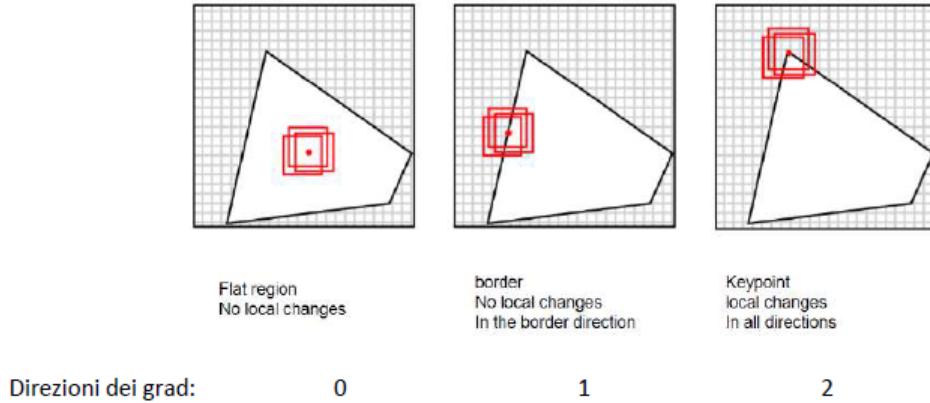


6.5 Harris Detector for corners

Harris Detector is an algorithm used to find the keypoints. The keypoints are characterized by:

- A formal definition (perceptual, algebraic,...)
- A precise position (localization)
- Possibly some invariance (eg to rotation, luminance, etc...)

Harris use the concept of auto-correlation: take a patch and use this patch to find others similar around (moving it around), if everything is uniform there is not more information (*se prendo un pezzo della mia maglietta, cerco e trovo lo stesso pezzo intorno a me significa che l'intorno è molto uniforme e non ci sono feature points*).



Le porzioni che hanno il gradiente in almeno due direzioni sono quelle più significative e devono essere localizzate. **Si ha un picco del gradiente in corrispondenza dell'angolo.** As you may notice, textureless patches are nearly impossible to localize. Patches with large contrast changes (gradients) are easier to localize, although straight line segments at a single orientation suffer from the *aperture problem*. Patches with gradients in at least two (significantly) different orientations are the easiest to localize. Formalizzando il concetto esposto a grandi linee sopra, dato un pixel $q(x,y)$ e una sua finestra di intorno $\Omega(q)$ facciamo il confronto (come nel template matching) di tale finestra con la finestra che si ottiene traslando la finestra stessa di una lunghezza δ in direzione d (dove d è un vettore unitario).

$$D_q(d) = \sum_{r \in \Omega(q)} [d^\top \nabla I(r)]^2$$

↑

D is the information content associated with
 q in the d direction, relative to the window
 D can be expressed as a (u,v) vector

L'informazione che stiamo cercando, cioè quanto la finestra iniziale sia saliente, è chiamato contenuto informativo della porzione di immagine che stiamo con-

siderando ed è calcolato come la differenza di intensità $E(u,v)$ tra la finestra originale e la finestra traslata moltiplicata per una funzione w :

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2$$

- $w(x, y)$ is the weight, it can be uniform or gaussian
- $I(x + u, y + v)$ is the shifted intensity
- $I(x, y)$ is the intensity

Interpretando nel modo corretto questo valore possiamo capire se l'area considerata inizialmente è saliente o meno. Approssimiamo l'espressione di $E(u,v)$ ottenuta sopra utilizzando le serie di taylor:

$$\begin{aligned} E(u, v) &\approx \sum_{x,y} w(x, y) [I(x, y) + uI_x + vI_y - I(x, y)]^2 \\ &= \sum_{x,y} w(x, y) [uI_x + vI_y]^2 \\ &= \sum_{x,y} w(x, y) \begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \end{aligned}$$

Possiamo usare la forma matriciale dell'approssimazione bilineare:

$$E(u, v) \cong [u, v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

M è chiamata structure tensor (tensore di struttura) o anche matrice di autocorrelazione o anche matrice hessiana:

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Il punto p nell'immagine è saliente (quindi per il ragionamento visto nell'immagine prima è un angolo) se il contenuto informativo è sufficientemente significativo. Questo equivale a dire che il contenuto informativo è maggiore di una soglia scelta, ovvero:

$$\min \{ d^T M d \mid |d| = 1 \} > \tau$$

Questo equivale a dire che, per essere rilevante il punto considerato, il minimo autovalore di M deve essere maggiore della soglia:

$$\min \lambda > \tau$$

Nella realtà non vengono sottoposti a soglia gli autovalori ma dei valori calcolati sulla base di essi. Ci sono diverse metriche che si possono calcolare applicando diversi metodi, noi e vedremo due:

1. Il valore f viene sottoposto a soglia:

$$f = \frac{\text{Det}(M)}{\text{trace}(M)}, f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

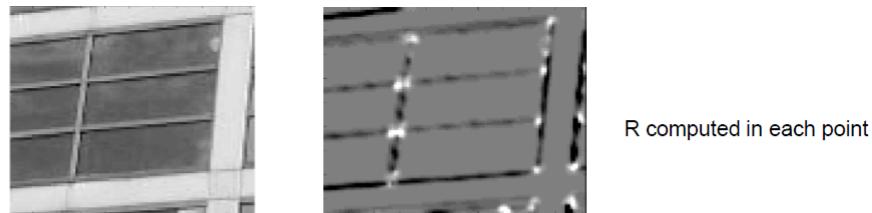
2. Corner response: il valore R viene sottoposto a soglia:

$$R = \det M - k(\text{trace } M)^2$$

$$\det M = \lambda_1 \lambda_2$$

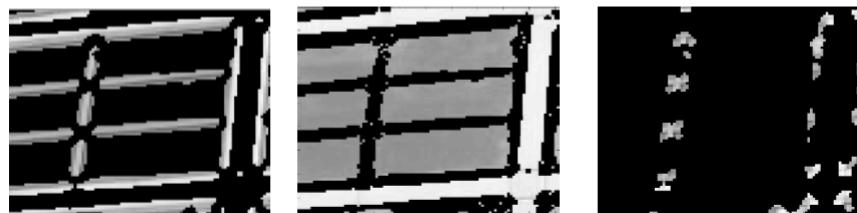
$$\text{trace } M = \lambda_1 + \lambda_2$$

Dopo aver trovato quali sono le porzioni dell' immagine in cui ci sono potenziali angoli si applica un metodo per rilevare il massimo locale (es. non maxima suppression, oppure posso tenere solo i keypoints che sono un 10% maggiori rispetto ai punti nell'intorno) per tenere un solo punto per porzione: quello è un keypoint.



R computed in each point

Derivatives are computed with sobel, w is a gaussian with sigma=1



R<1000, borders

[R]<10000 flat regions

R>10.000 keypoints

HARRIS

Summary of harris detector algorithm

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x \cdot I_x \quad I_{y2} = I_y \cdot I_y \quad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma t} * I_{x2} \quad S_{y2} = G_{\sigma t} * I_{y2} \quad S_{xy} = G_{\sigma t} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

$$H(x, y) = \begin{bmatrix} S_{x2}(x, y) & S_{xy}(x, y) \\ S_{xy}(x, y) & S_{y2}(x, y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

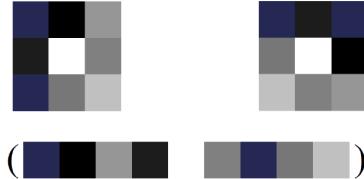
$$R = \text{Det}(H) - k(\text{Trace}(H))^2$$

6. Threshold on value of R. Compute nonmax suppression.

Spiegazione Harris Wikipedia L'algoritmo di Harris può essere riassunto come:

1. conversione dell'immagine in scala di grigi;
2. riduzione del rumore tramite un filtro Gaussiano;
3. approssimazione del gradiente tramite l'operatore di Sobel;
4. per ogni pixel dell'immagine, calcolo della funzione di risposta R in un intorno di dimensione 3x3
5. determinazione dei punti di massimo locale di R , che rappresentano gli angoli rilevati nell'immagine.

Harris Descriptor It's difficult to find a descriptor for the feature matrix extracted with Harris. A possible solution could be a vector of the pixel values, but this is not invariant to rotation and scale. Harris is also a good detector of keypoints, and in this case they are invariant to rotation. So it's possible to say that keypoints are invariant to rotation while the descriptor isn't. Perché se io ruoto una stessa immagine rilevo gli stessi keypoints ma i descrittori di quei keypoints sono diversi quindi lo stesso angolo risulterà come un angolo diverso nelle due immagini. Inoltre sia il rilevamento di keypoints con Harris che il loro descrittore non sono invarianti allo scaling. Infatti un punto ad una certa risoluzione può essere visto come un angolo mentre ad un'altra solo come un bordo.



6.6 SIFT Scale Invariant Feature Transform

SIFT like harris is an algorithm used to find the keypoints, it is robust to luminance variations and invariant to scale and translations.

6.7 SIFT detector algorithm

The algorithm is composed by 2 parts:

- **DETECTOR** Find the extremes (space invariant) and localize the keypoints
- **DESCRIPTOR** Find the good orientation and describe the keypoints

The algorithm

1. **Scale-space extrema detection (Detect the feature points)** *A point is strong when it is detected in 3 different scales*
 - (a) Build a Gaussian-blurred image pyramid (image pyramid = take the image at different scales)
 - (b) Subtract adjacent levels to obtain a Difference of Gaussians (DoG) pyramid (so approximating the Laplacian of Gaussians)
 - (c) Take local extrema of the DoG filters at different scales as keypoints
2. **Keypoint localization** Find the correct localization of this points
3. **Orientation assignment** Find the good orientation in order to have a standardization
4. **keypoint descriptor**

6.7.1 Scale-space extrema detection

We need to identify the most distinct features in a given image while ignoring any noise. Additionally, we need to ensure that the features are not scale-dependent. (We use the *Gaussian Blurring technique* to reduce the noise in an image.)

Come specificato nelle righe precedenti, il primo punto si compone di due step: trovare le key features (quei punti che hanno particolare significato nell'immagine come ad esempio gli angoli) e assicurarsi che le key features siano indipendenti

dalla scalabilità dell’immagine in modo da ottenere l’invarianza rispetto lo scaling.

- **Step 1: Creazione delle ottave** Per prima cosa dobbiamo definire lo space scale di un’immagine che è una funzione $L(x, y, \sigma)$ ottenuta facendo la convoluzione fra kernel gaussiani (blurring) con σ differenti e l’immagine iniziale a scale differenti. Lo scale-Space viene organizzato in ottave e il numero di tali ottave e le loro diverse scale dipendono dalla dimensione dell’immagine originale. Quindi generiamo diverse ottave partendo dall’immagine originale ognuna delle quali contiene immagini aventi la stessa dimensione ma sfuocate con kernel gaussiani aventi σ diversi. La dimensione delle immagini cambia solo fra ottave diverse mentre rimane la stessa dentro la stessa ottava. La dimensione dell’immagine in un’ottava è pari alla metà della dimensione delle immagini nell’ottava precedente. Mentre i diversi σ dei diversi kernel applicati dentro la stessa ottava varianno nel seguente modo:

$$\sigma_n = k^n \sigma_0 \quad (k = 2)$$

(ottave piramidali)

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

The ideal number of octaves should be four, and for each octave, the number of blur images should be five.

- **Step 2: Creazione dei DoG** Arrivati a questo punto utilizziamo le ottave che abbiamo ottenuto per costruire un altro insieme di immagini il Derivative of Gaussian (DoG) definito da $D(x, y, \sigma)$. Quello che viene fatto all’interno di ogni ottava è quello di sottrarre tra loro due immagini sfuocate con filtri aventi σ differenti. In questo modo otterremo un nuovo insieme. Questa sottrazione ci consente di ottenere un’approssimazione del gradiente che ci consente di trovare i picchi della funzione dell’immagine.

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

$$= L(x, y, k\sigma) - L(x, y, \sigma)$$

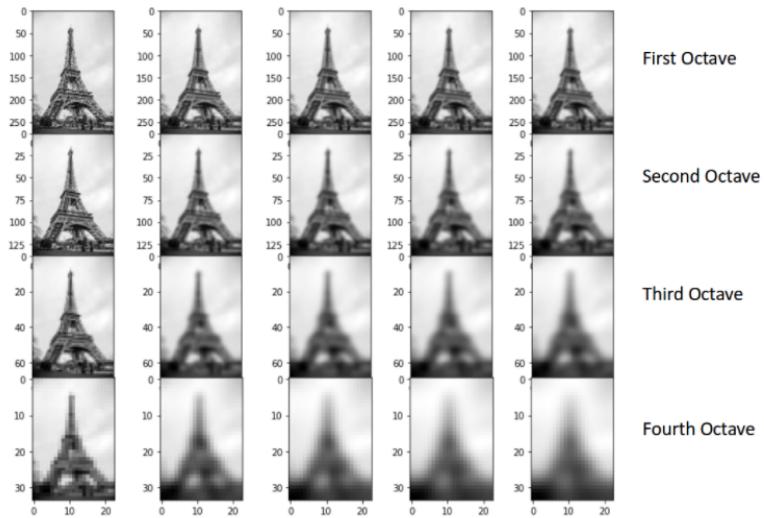


Figure 6: step 1: Creazione delle ottave

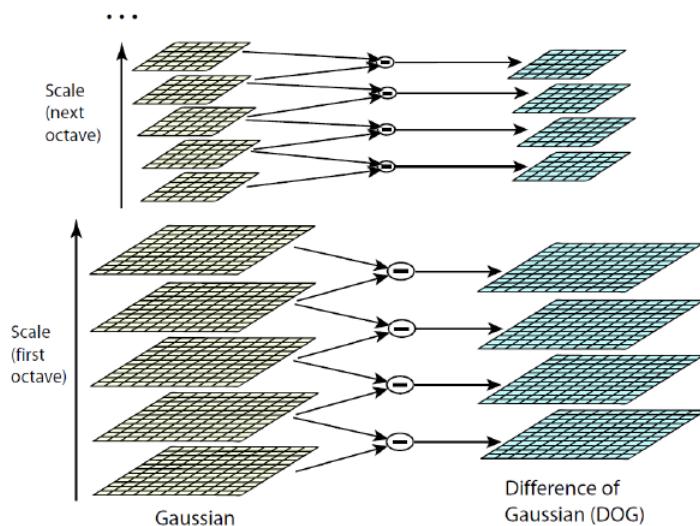
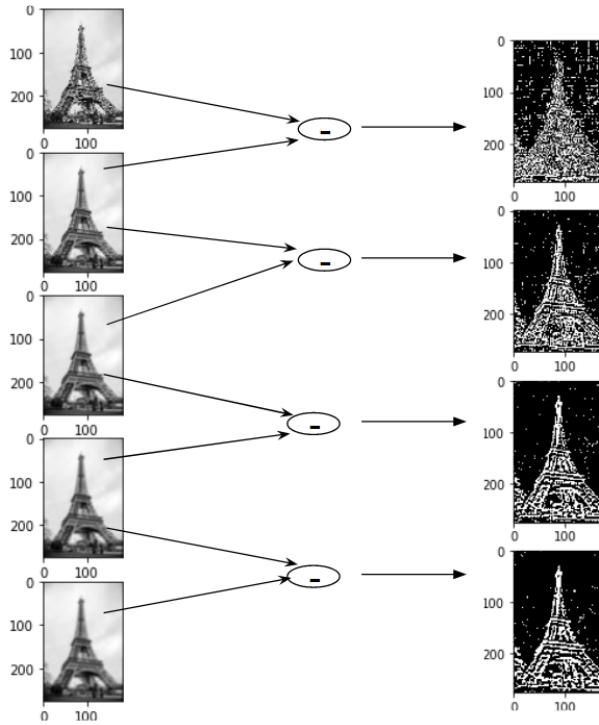


Figure 7: Step 2: Creazione dei DoG

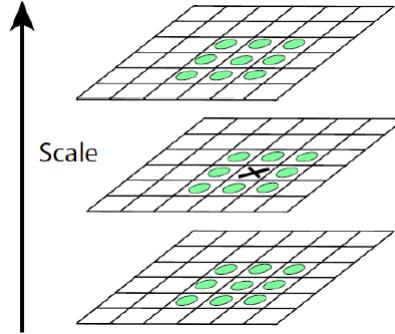


6.7.2 Keypoints Localization

Once the images have been created, the next step is to find the important keypoints from the image that can be used for feature matching. The idea is to find the local maxima and minima for the images. This part is divided into two steps:

- Find the local maxima and minima
- Remove low contrast keypoints (keypoint selection)

Find the local maxima and minima To locate the local maxima and minima, we go through every pixel in the image and compare it with its neighboring pixels. When we say ‘neighboring’, this not only includes the surrounding pixels of that image (in which the pixel lies), but also the nine pixels for the previous and next image in the octave. This means that every pixel value is compared with 26 other pixel values to find whether it is the local maxima/minima



Se il punto p è un estremo locale (minimo o massimo locale) nei tre livelli allora viene selezionato come keypoint. I keypoints generati in questo modo sono troppi e molti di essi sono posizionati lungo un edge oppure non hanno abbastanza contrasto per essere definiti keypoints. In entrambi i casi non sono sufficientemente interessanti per essere rilevati come feature quindi dobbiamo eliminarli.

Remove low contrast keypoints (keypoint selection) Il metodo utilizzato per eliminare quelli che non hanno sufficiente contrasto è semplice, basta confrontare l'intensità del contrasto con una soglia minima e trascurare il punto se tale contrasto non è sufficiente.

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Lowe suggests keep points with $r=10$.

6.7.3 Orientation Assignment

At this stage, we have a set of stable keypoints for the images. We will now assign an orientation to each of these keypoints so that they are invariant to rotation. We can again divide this step into two smaller steps:

- Calculate the magnitude and orientation
- Create a histogram for magnitude and orientation

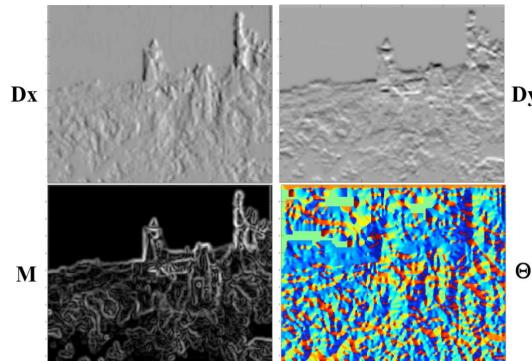
Calculate Magnitude and Orientation Consideriamo un intorno del keypoint in ogni immagine con σ (nelle immagini di L non di DoG) differenti e calcoliamo magnitudo e direzione del gradiente in questo intorno.

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

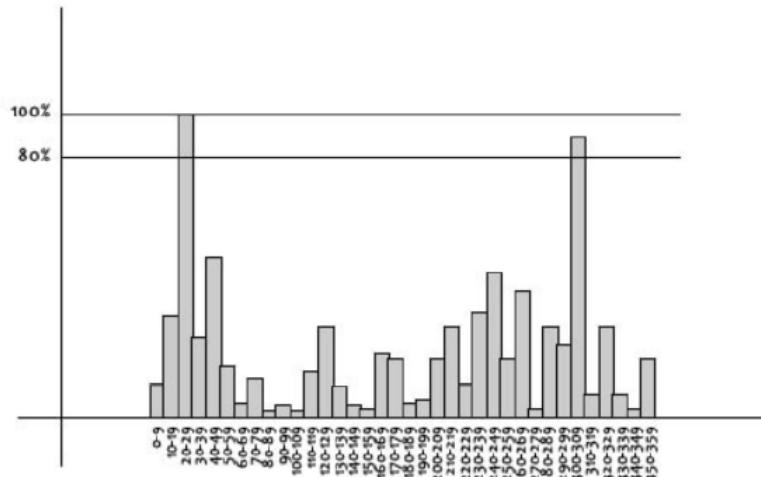
$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$

The magnitude represents the intensity of the pixel and the orientation gives the direction for the same.

Creating a Histogram for Magnitude and Orientation Dopo di che si crea un istogramma con 36 bins che coprono tutte le 360 direzioni (intervallo di 10 direzioni per ogni bin) e si va ad aggiungere in ogni bin un valore proporzionale alla magnitudine del gradiente che ha direzione compresa nell'intervallo di direzioni associato a quel bin. Questa operazione si fa per ogni magnitudine e direzione del gradiente calcolata nell'intorno del keypoint.



Una volta ripetuta questa operazione per ogni punto nell'intorno del keypoint l'istogramma avrà un picco da qualche parte.

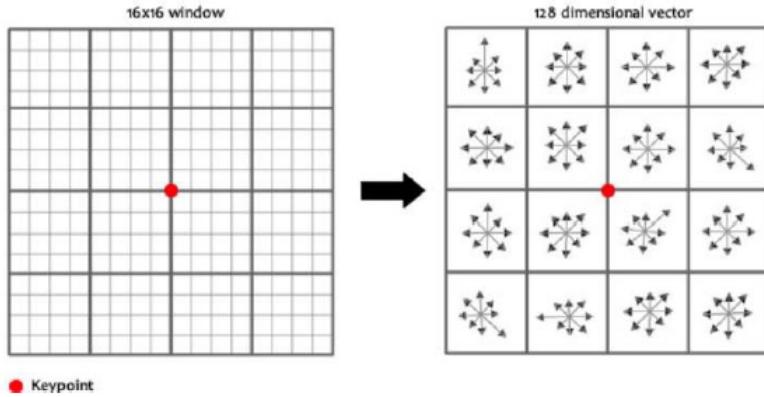


Viene considerata come orientazione la direzione corrispondente al picco più alto e anche ogni direzione che ha valore superiore all'80% di tale picco è tenuta in considerazione. In questo modo si crea un keypoint che ha stessa location

e stesso scale ma diverse orientazioni. Questo rende in keypoint invariante all'orientazione.

6.7.4 Keypoint descriptor

This is the final step for SIFT. So far, we have stable keypoints that are scale-invariant and rotation invariant. In this section, we will use the neighboring pixels, their orientations, and magnitude, to generate a unique fingerprint for this keypoint called a ‘descriptor’. We will first take a 16×16 neighborhood around the keypoint. This 16×16 block is further divided into 4×4 sub-blocks and for each of these sub-blocks, we generate the histogram using magnitude and orientation.



At this stage, the bin size is increased and we take only 8 bins (not 36). Each of these arrows represents the 8 bins and the length of the arrows define the magnitude. So, we will have a total of 128 bin values for every keypoint.

- **Rotation dependence** The feature vector uses gradient orientations. Clearly, if you rotate the image, everything changes. All gradient orientations also change. To achieve rotation independence, the keypoint’s rotation is subtracted from each orientation. Thus each gradient orientation is relative to the keypoint’s orientation.
- **Illumination dependence** If we threshold numbers that are big, we can achieve illumination independence. So, any number (of the 128) greater than 0.2 is changed to 0.2. This resultant feature vector is normalized again. And now you have an illumination independent feature vector!

7 The camera model

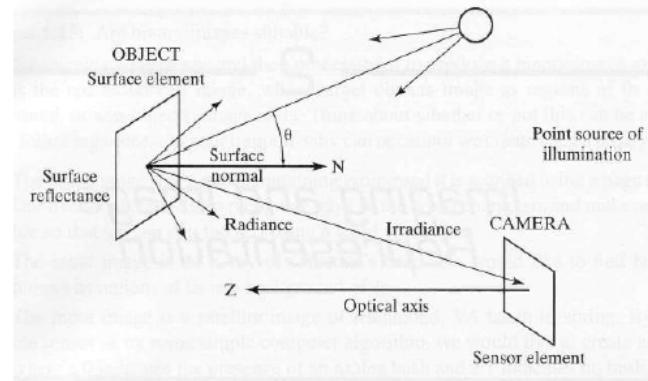
7.1 Image sensors technologies

The basic operating principle is as follows: the image is focused on a grid composed of a myriad of small point sensors which individually convert the detected brightness. The two most common digital technologies used to acquire the images are the CCD and the CMOS.

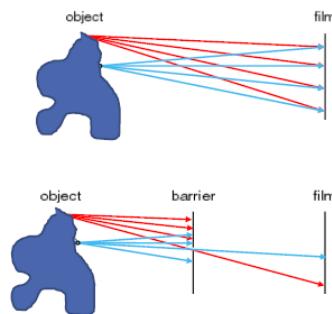
- **CCD** Move photogenerated charge from pixel to pixel and convert it to voltage at the output node
- **CMOS** Imagers convert charge to voltage inside each pixel

7.2 The physics of light model

The scene reflects radiation towards the camera and the camera senses it via chemicals on film or by electronic devices.



Per semplificare il modello di geometria dell'immagine si assume che la scena sia illuminata da una singola sorgente. Because each point irradiates light everywhere, we add a barrier to block off most of the rays, this reduces the blurring. The barrier's small opening is known as the **aperture**.



The simplest device to form an image of a 3D scene on a 2D surface is the "pinhole" camera: the rays of light pass through a "pinhole" and form an inverted image of the object on the image plane.

7.3 The pinhole model

Il pinhole camera model sfrutta il principio della camera oscura. La camera oscura è un dispositivo ottico composto da una scatola oscurata con un foro (pinhole) sul fronte e un piano di proiezione dell'immagine sul resto. La camera oscura tramite il foro raccoglie i raggi proiettati su qualsiasi oggetto e ottiene dall'altra parte della camera l'immagine capovolta. Il funzionamento è intuitivo dalla foto:

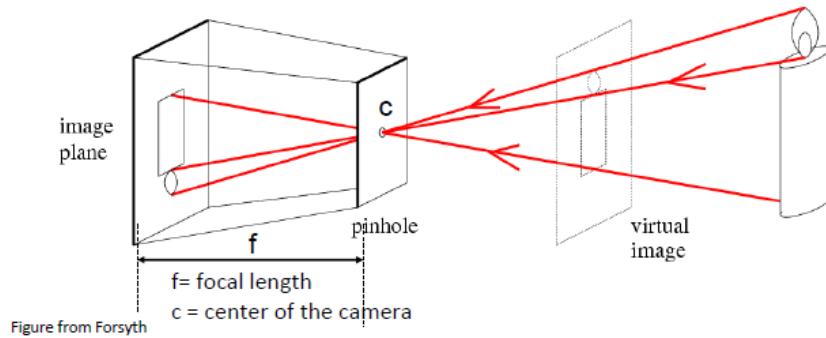


Immagine reale e Immagine virtuale I raggi di luce che emergono da un oggetto, dopo la riflessione o la rifrazione da uno specchio o una lente, si incontrano in un punto specifico e formano una riproduzione dell'oggetto noto come immagine. Esistono due tipi di immagini che si formano, ovvero **Immagine reale** e **Immagine virtuale**. Un'immagine si dice virtuale quando non esiste nella realtà ma si forma sul prolungamento dei raggi riflessi prodotti dal nostro cervello (immagine allo specchio). Un'immagine si dice invece reale quando si forma al punto di intersezione di raggi luminosi reali.

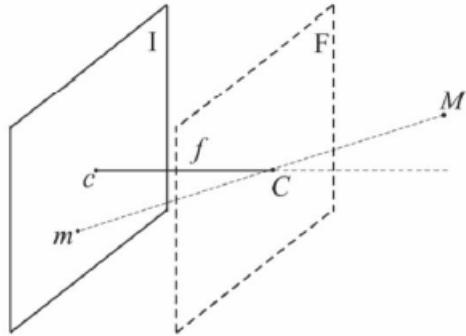
Pinhole camera con la proiezione prospettica I parametri essenziali di un sistema pin-hole sono due:

- Il centro ottico **C**
- Un piano immagine o retina **I**

Fissati questi due parametri si derivano altri elementi notevoli:

- La distanza del centro ottico dal piano immagine, detta distanza focale, **f**
- L'asse ottico, ovvero la retta passante per **C** e ortogonale ad **I**
- Il punto principale **c** intersezione tra il piano immagine **I** e l'asse ottico

- Il piano focale \mathbf{F} parallelo ad \mathbf{I} e contenente il centro ottico.



Per derivare un modello analitico di una pin-hole camera, è necessario introdurre due sistemi di riferimento cartesiani:

1. Uno 3D per i punti della scena: **S.d.R standard (x,y,z)** centrato nel centro ottico C , con l'asse Z coincidente con l'asse ottico
2. Uno 2D per i punti dell'immagine: **S.d.R (u,v)** centrato nel punto principale e con assi u e v orientati come x e y rispettivamente.

Riprendendo dei concetti di proiezione prospettica consideriamo:

- un punto W di coordinate $w = [x, y, z]^T$ nel S.d.R. standard
- la sua proiezione M su I attraverso C , di coordinate $m = [u, v]^T$

Semplici considerazioni di similitudine tra triangoli portano alla relazione:

$$\frac{f}{z} = \frac{-u}{x} = \frac{-v}{y}$$

da cui la relazione non lineare:

$$\begin{cases} u = -\frac{f}{z}x \\ v = -\frac{f}{z}y \end{cases}$$

È possibile esprimere le coordinate euclidiene di m e w in coordinate omogenee:

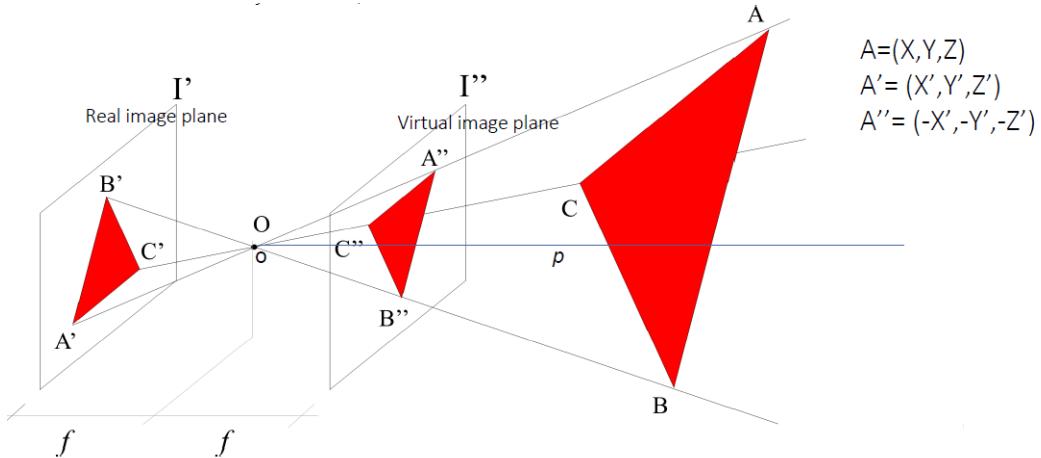
$$\tilde{\mathbf{m}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \tilde{\mathbf{w}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Usando le coordinate omogenee (e quindi considerando la trasformazione come tra spazi proeittivi), la proiezione prospettiva diviene una trasformazione lineare:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

linear projection in homogeneous coordinates!

$$\begin{aligned} X' &= -f \frac{X}{Z} \\ Y' &= -f \frac{Y}{Z} \\ Z' &= -f \end{aligned}$$



Proiezione Ortografica La prospettiva ortografica viene applicata nel caso in cui un’immagine sia lontano dalla camera. La scena è talmente lontana al tal punto che Z è troppo grande rispetto a f e la distanza tra gli oggetti può considerata come costante nel caso della componente z .

$$\begin{aligned} Z_0 \pm \Delta Z \quad X' &= -f \frac{X}{Z_0} \\ \frac{\Delta Z}{Z_0} \ll 1 \quad Y' &= -f \frac{Y}{Z_0} \\ Z' &= -f \end{aligned}$$

Quindi una proiezione ortografica elimina la componente z delle coordinate 3D:

$$\mathbf{x} = [\mathbf{I}_{2 \times 2} \mid \mathbf{0}] \mathbf{p}$$

Se usiamo le coordinate omogenee possiamo scrivere:

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{p}}$$

L'ultima colonna della matrice presenta comunque un 1 per evitare che il punto possa essere un punto all'infinito.

Trasformazione Ortografica vs Prospettica Proiezione Ortogonale o ortografica, vuol dire letteralmente proiettare un oggetto da rappresentare su di un piano ad esso perpendicolare (ortogonale). E' come se illuminassimo un oggetto con una torcia e ne osservassimo la sua ombra proiettata su una parete (piano) posta alle sue spalle.

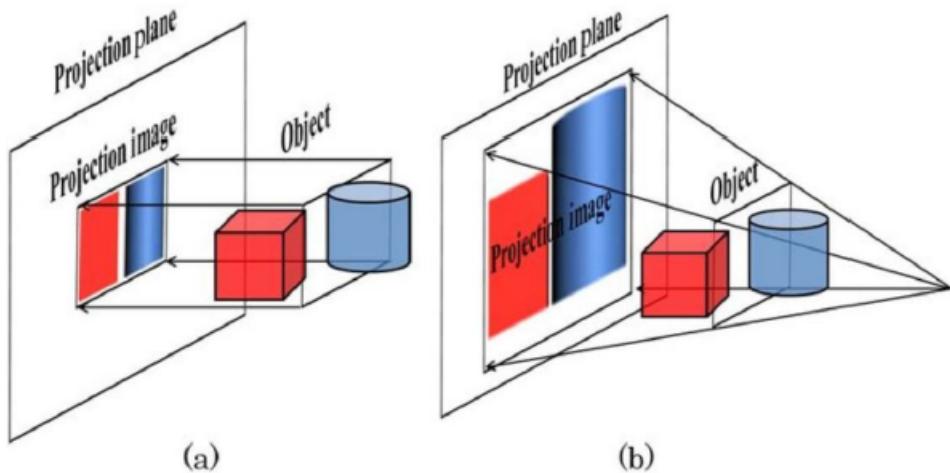


Figure 8: proiezione ortografica (a) e prospettica (b)

La proiezione ortografica viene usato perlopiù in immagini satellitari ecc. La perdita della terza dimensione di oggetti in una scena comporta che non si ha più informazione in merito alla distanza relativa fra gli oggetti e alle loro dimensioni. Con la proiezione prospettica invece è possibile notare che due oggetti anche se sembrano avere dimensioni diverse (uno più alto un più basso) in realtà sono semplicemente posizionati in prospettiva.

Perspective effects Parallel lines in the scene intersect in the image and converge in image on horizon line. Many to one: any points along same ray map to same point in image.

- **Vanishing point** the point at which receding parallel lines viewed in perspective appear to converge

7.4 Camera Calibration

La visione artificiale crea un modello approssimato del mondo reale 3D partendo da immagini bidimensionali 2D utilizzando delle equazioni che legano il mondo reale alle immagini. Queste equazioni sono scritte nel sistema di riferimento della telecamera e molto spesso si assume che:

- Il sistema di riferimento della telecamera è posizionato rispetto a un altro sistema di riferimento, conosciuto come sistema di riferimento mondo
- Le coordinate dei punti dell'immagine (o delle immagini) nel sistema di riferimento della telecamera possono essere ottenute dalle coordinate pixel, le uniche direttamente disponibili dall'immagine.

A questo scopo abbiamo bisogno di conoscere le caratteristiche della telecamera (o telecamere, se si parla di visione stereoscopica), conosciute come **parametri intrinseci** e **parametri estrinseci**. Il processo che determina i parametri sia intrinseci che estrinseci è detto **camera calibration**.

7.4.1 Parametri Intrinseci

Sono parametri necessari a collegare le coordinate pixel di un'immagine con le corrispondenti coordinate nel sistema di riferimento della telecamera; dipendono infatti dalla telecamera stessa. I parametri intrinseci sono:

1. **Lunghezza focale f** Assumes that the camera is positioned in the origin of the axis. The origin of the world coordinates is the camera center C, and the origin of the (virtual)image plane is the principal point of coordinates (0,0,f). So I need something that gives me the consciousness of f, a sort of given data that brings more information about spatial distances. (NB: z in the image plane does not exist, is 0 if the transformation is from 3D to 2D)

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

2. **Coordinate (x_0, y_0)** del punto principale dell'immagine, questo punto infatti non coincide sempre con l'origine del piano dell'immagine. If the origin in the image plane is translated with respect the principal point, and the principal point has coordinates x_0, y_0 in the image plane ((x_0, y_0, f) in the space) we must add a **Translation**

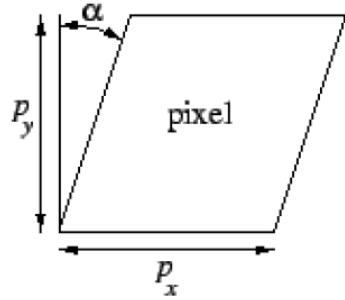
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

3. **Pixelizzazione** Nel caso in cui i pixel non siano quadrati ma per esempio rettangolari (può succedere nei video), la distanza focale deve tenere conto

di una distorsione della proiezione sia sull'asse x che sull'asse y. Quindi cambiano le formule di proiezione.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & x_0 & 0 \\ 0 & fy & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

4. **Skew** Lo skew è un parametro che serve alla calibrazione nel caso in cui l'angolo compreso fra i due assi all'origine del image plain non sia retto.



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & s & x_0 & 0 \\ 0 & fy & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Normally s is fixed to 0 (especially in new digital CMOS cameras)

Nel caso più generale possibile possiamo quindi identificare la matrice dei parametri intrinseci composta da 5 parametri:

- x_c, y_c traslazione dall'origine
- f_x, f_y lunghezza focale e pixellizzazione
- S skew

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & s & xc & 0 \\ 0 & fy & yc & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

7.4.2 Parametri Estrinseci

I parametri estrinseci sono parametri che definiscono la posizione e l'orientamento del sistema di riferimento della telecamera rispetto al sistema di riferimento

mondiale. Il sistema di riferimento della telecamera è molto spesso sconosciuto. Un problema comune è determinare la posizione e l'orientamento rispetto a un sistema di riferimento sconosciuto, utilizzando esclusivamente informazioni ottenute attraverso le immagini catturate. I parametri estrinseci sono definiti come un set di parametri geometrici che identifica univocamente la trasformazione tra il sistema di riferimento della telecamera a noi sconosciuto e il sistema di riferimento mondo a noi noto. Una tipica scelta per descrivere la trasformazione tra questi due sistemi di riferimento consiste nell'usare:

- Un vettore di traslazione \mathbf{T} a tre dimensioni che descrive la relativa posizione delle origini nei due sistemi di riferimento.
- Una matrice di rotazione \mathbf{R} 3×3 ortogonale, che porta gli assi corrispondenti dei due sistemi di riferimento l'uno sopra l'altro.

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \mathbf{R}_{3 \times 3} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{t}$$

7.4.3 Parametri intrinseci ed estrinseci

Considerando entrambe le matrici (intrinseca ed estrinseca) abbiamo:

$$x = k[R \ t]X$$

$$p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & s & x_0 \\ 0 & fy & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & tx \\ r_{21} & r_{22} & r_{23} & ty \\ r_{31} & r_{32} & r_{33} & tz \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- intrinsic parameters: parameters: focal length, optical center, skew and distortion
- extrinsic parameters: (pose): rotation and translation

7.4.4 Methods of calibration

1. **Multiple images** The chessboard method; extract corners. Very precise; require multiple image manual calibration, difficult to use for wide DoG
2. **Using different geometric structure of the scene** lines, vanishing points, selecting some existing lines.
3. **Camera self-calibration** To estimate intrinsic parameters with a moving camera on a video; it solve calibration also with distortion parameters using bipolar lines. (*Stima i parametri intrinseci muovendo la telecamera*)
4. **Deep Learning**
 - **DeepFocal** [Workman et al. 2015] predicts only the focal length (no distortion estimation).

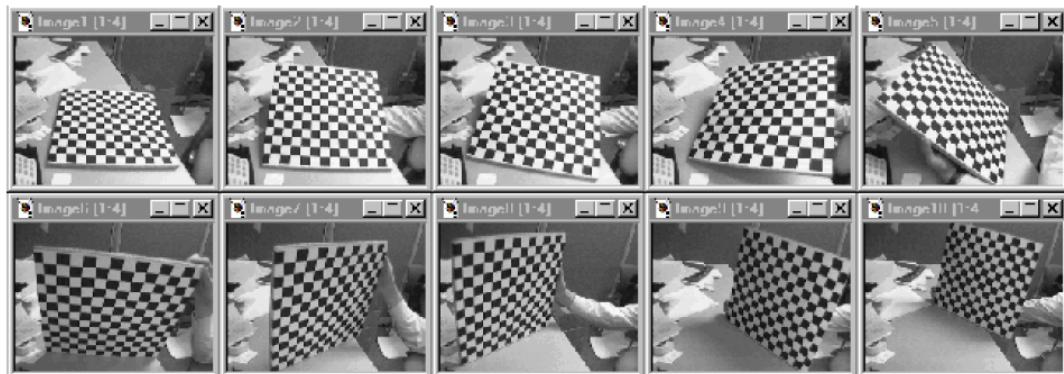
- **Rong et al.** et al. [2016] estimate only the radial distortion (no focal length).
- **Hold-Geoffroy et al.** et al. [2018] estimate the focal length and camera orientation (no distortion parameter).

(non esistono ad oggi tecniche di calibrazione con deep learning efficaci)

7.4.5 Zhang method of calibration

Il metodo di Zhang è un metodo di calibrazione semplice e prevede l'acquisizione, nella stessa immagine o in immagini successive, di pattern piani caratterizzati da giaciture diverse (traslati o ruotati). Questa tecnica sfrutta il calcolo di diverse matrici omografiche H ottenute dall'osservazione di un piano (per esempio una griglia di calibrazione con marker equispaziati - chessboard). Il grosso vantaggio risiede nel fatto che non è necessario conoscere la posizione relativa di tali piani, quindi risulta essere un metodo estremamente pratico. Vantaggi:

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online (Intel's OpenCV library)



Quante immagini servono per poter calcolare i parametri ? Considerando che lo skew $s=0$, quindi ci sono 4 parametri intrisici (costanti) e 6 parametri estrinseci 3 angoli e 3 rotazioni, variabili. Avendo N_i immagini con M angoli ciascuna:

- $4+6*N_i$ parametri
- $2*N_i*M$ costanti ($2*M$ coordinate per ogni immagine)

deve essere:

$$2 * N_i * M \geq 4 + 6 * N_i \rightarrow N_i \geq \frac{2}{M - 3}$$

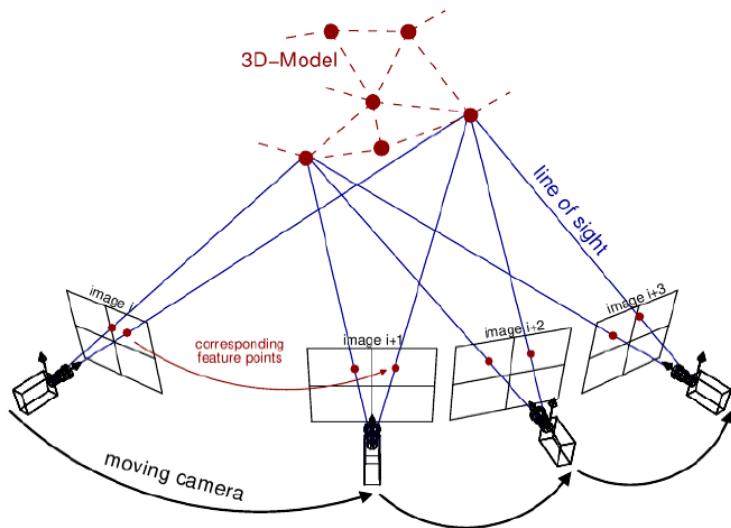
quindi 2 immagini con 4 punti è abbastanza.

Algoritmo metodo Zhang

1. **Data acquisition**
2. **Specify corner order**
3. **Corner extraction**
4. **Minimize projection error**
5. **Camera calibration**

7.5 Structure from Motion

SfM è una tecnica di fotogrammetria che permette di stimare un modello 3D, sovrapponendo sequenze di immagini 2D che vengono scattate dalla stessa camera da diverse angolazioni.



7.6 Distortion

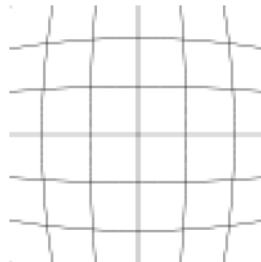
La totalità delle telecamere commerciali devia dal modello della pin-hole camera e tale deviazione è generalmente tanto maggiore quanto grande è il campo visivo della camera: siccome ogni ottica è composta da un certo numero di lenti, la distorsione deriva dalle non idealità nella fase di produzione e di assemblaggio dell'ottica. Ottenere infatti una lente non distortante è un processo estremamente costoso e soprattutto nelle applicazioni a basso costo dove bisogna fare affidamento a ottiche economiche risulta un problema molto evidente. Queste non idealità generano una distorsione non lineare difficilmente modellizzabile e, anche per il fatto che tale distorsione dipende dall'interazione tra la lente e il sensore, i produttori di lenti normalmente non forniscono, o non riescono

a fornire, informazioni geometriche su come rappresentare tale distorsione. È importante osservare che il modello della pin-hole camera è valido solamente se l'immagine su cui si lavora è non distorta pertanto calibrare, ovvero correggere la distorsione geometrica, è un prerequisito per ricostruire in maniera accurata la tridimensionalità della scena osservata. In generale i contributi distorcenti della lente si dividono in:

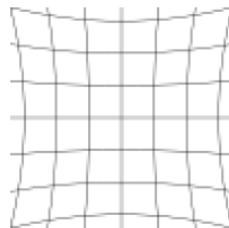
- **Radiali** diretti lungo la direttrice che unisce il punto al centro di distorsione
- **Tangenziali** che sono perpendicolari alla direttrice

I contributi di distorsione tangenziali (e altri contributi qui non citati) sono normalmente piccoli mentre la distorsione radiale è sempre rilevabile e, man mano che la distanza focale diventa corta, in generale aumenta di intensità. Le distorsioni radiali possono essere suddivise a loro volta in:

- **Barrel distortion (distorsione a botte)**, di solito causata dalle lenti grandangolari. L'effetto della barrel distortion è quello di un'immagine mappata attorno ad una sfera (barile). Questo tipo di distorsione viene utilizzata per esempio negli obiettivi fisheye. Si verifica quando l'ingrandimento della lente diminuisce con l'aumentare della distanza dal punto principale. In presenza di questo effetto, linee rette verticali che si trovano vicino al bordo dell'immagine appariranno curve verso l'esterno.



- **Pin Cushion distortion** di solito causate dai teleobiettivi. Si verifica quando, al contrario, l'ingrandimento aumenta con l'aumentare della distanza dal punto principale quindi, in presenza di questo effetto, linee rette verticali che si trovano vicino al bordo dell'immagine appariranno curve verso l'interno.



Tangetial distortion

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = L(r) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + \begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix}$$

- $L(r)$ is the radial distortion
- $\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$ is the original point
- $\begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix}$ is the tangential distortion

There is a non linear relation depending to the r distance to the center

$$r = \sqrt{(\tilde{x} - \tilde{x}_c)^2 + (\tilde{y} - \tilde{y}_c)^2}$$

approximation using Taylor $L(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots$

- Radial distortion of a point x_c, y_c :

$$\begin{aligned} \hat{x}_c &= x_c (1 + \kappa_1 r_c^2 + \kappa_2 r_c^4) \\ \hat{y}_c &= y_c (1 + \kappa_1 r_c^2 + \kappa_2 r_c^4) \end{aligned}$$

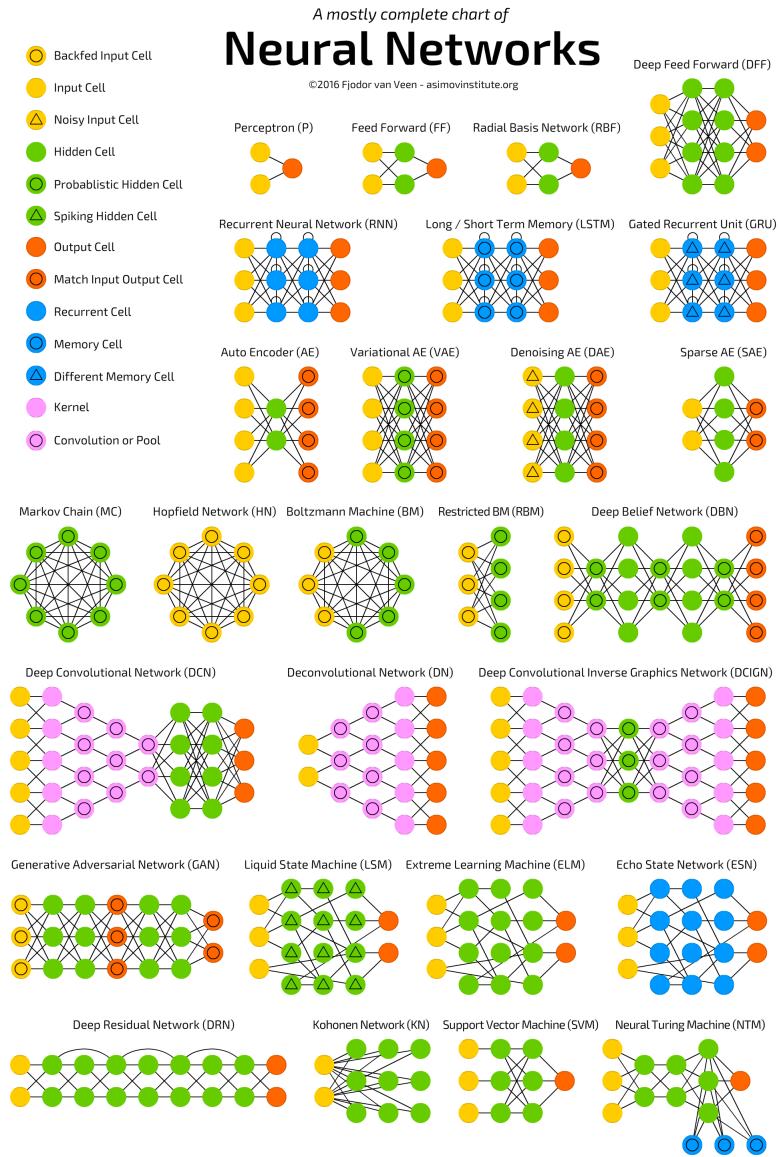
- Tangential distortion:

$$\begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix} = \begin{pmatrix} 2p_1 \tilde{x} \tilde{y} + p_2 (r^2 + 2\tilde{x}^2) \\ p_1 (r^2 + 2\tilde{y}^2) + 2p_2 \tilde{x} \tilde{y} \end{pmatrix}$$

7.7 Vanishing Point

I vanishing point o punti all'infinito, sono i punti in cui in una proiezione prospettica le rette convergono. Le immagini possono avere dei punti all'infinito anche al di fuori dell'immagine stessa, infatti i punti all'infinito di un'immagine sono 3, rispettivamente per gli assi x, y e z. Ogni insieme di rette parallele su un piano definisce un punto all'infinito. L'unione di tutti i punti all'infinito sullo stesso piano è detta vanishing line/retta proiettiva, che nell'immagine corrisponde all'orizzonte. È possibile calcolare i vanishing point usando Hough Transform.

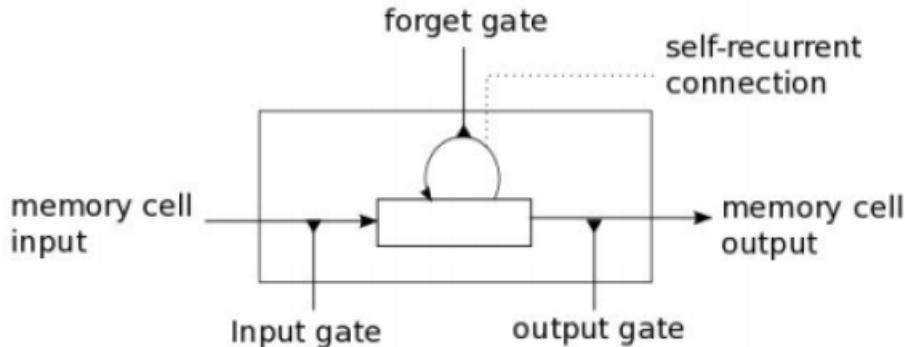
8 Computer Vision and Neuroscience



Recurrent Neural Network Una rete neurale ricorrente è una rete in cui alcuni neuroni sono collegati in loop tra loro. Tipicamente i valori di uscita di uno strato di un livello superiore sono utilizzati in ingresso di uno strato di livello inferiore. Quest'interconnessione tra strati permette l'utilizzo di uno degli

strati come memoria di stato, e consente, in ingresso di una sequenza temporale di valori, modellarne un comportamento dinamico temporale dalle informazioni ricevute all'istante di tempo precedente.

Long short-term memory Si tratta di una rete neurale ricorrente che differentemente dalle standard feedforward reti neurali presenta delle cosiddette **connessioni di feedback**. LSTM presenta un'architettura in grado di elaborare sia singoli dati che una sequenza, come ad esempio un video completo. Può essere utilizzata per il riconoscimento vocale e il riconoscimento della calligrafia. Il processo di trasferimento dei dati è lo stesso delle reti neurali ricorrenti standard. Tuttavia, l'operazione di propagazione delle informazioni è diversa. Durante il propagarsi dei dati nella rete, si effettua un'operazione di decisione in merito a quale informazioni/dati mantenere e quali invece scartare. La decisione sulle informazioni da mantenere viene effettuata all'interno di cells e i responsabili sono i gates. LSTM presenta appunto dei gates, che si aprono e si chiudono in base al flusso di informazioni che scorre nella rete. Solo i dati rilevanti vengono selezionati in sequenza per effettuare una previsione efficace. LSTM non è più tanto utilizzata come in passato, perché non si applica in maniera performante con architetture parallele.



The study of memory in Kahneman è uno psicologo, vincitore del premio Nobel 2002, ha studiato i meccanismi della memoria nella psicologia. Le sue considerazioni sono raccolte nel libro “Thinking, fast and slow”. Khaneman ha individuato due sistemi di memorizzazione nella mente umana:

- **Sistema 1**

- È veloce
- Si definiscono le caratteristiche di quello che viene memorizzato, a livello di inconscio e in maniera automatica
- Non si ha il controllo della memorizzazione e a volte nemmeno la coscienza di stare memorizzando

- Il ruolo di questo sistema è valutare la situazione
- Costituisce il 98% del nostro pensiero

- **Sistema 2**

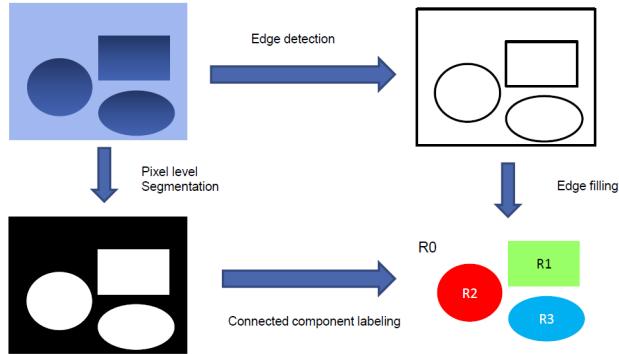
- È lento
- Si definiscono tutte le caratteristiche, con coscienza, si tratta di un processo mentale controllato e razionale
- Si è coscienti di stare memorizzando
- Cerca di trovare informazioni nuove o mancanti ed effettuare decisioni
- Costituisce il 2% del nostro pensiero

Eric Kendal Eric Kendal ha vinto il premio Nobel nel 2000 in psicologia e in medicina per le sue ricerche sulla psicologia alla base dello spazio di memorizzazione nei neuroni. Le sue sperimentazioni sono state effettuate sulla Aplysia, una lumaca di mare con 20mila cellule nervose, molto simili alla mente umana ma di numero decisamente inferiore. Kandel voleva studiare cosa succede a livello molecolare e nervoso nel momento in cui si crea una short-term o long-term memory. Kendal ha individuato due tipi di memoria:

- Il primo più complesso, richiede l'uso dell'ippocampo e viene detta memoria esplicita.
- Mentre la seconda più semplice, che permette di effettuare operazioni in maniera automatica, come dopo che si è imparato a guidare, è detta memoria implicita.

8.1 Segmentazione

Data un'immagine che presenta una regione di interesse in primo piano e un background, il compito della segmentazione è quello di individuare la regione o il segmento dell'immagine nella regione di interesse. Questo tipo di operazione si può fare partendo da una semplice edge detection per poi andare a riempire gli edge che sono stati individuati, oppure facendo una segmentazione a livello di pixel. La segmentazione è l'operazione da eseguire prima del riconoscimento dell'oggetto. Tramite la segmentazione, infatti, è possibile distinguere tra l'oggetto e il background. Dopo aver determinato la forma e la posizione dell'oggetto possono essere calcolati la dimensione, il centroide, l'orientamento. È importante distingnere tra la segmentazione percettiva e quella semantica, a livello percettivo si estrae semplicemente l'oggetto dal background in cui si trova nell'immagine ma non si effettua alcuna considerazione semantica. Per questo motivo la segmentazione si può intendere come un problema di raggruppamento, trova diverse applicazioni nella cluster analysis. Infatti quello che si effettua con la segmentazione è il raggruppamento di pixels simili. Sia la segmentazione che il raggruppamento sono problemi indefiniti (illdefined).



La segmentazione consiste nel dividere le immagini in insieme di punti omogenei chiamati regioni dell'immagine (che spesso corrispondono a oggetti reali nel mondo 3D) date features visive. Spesso una regione può essere chiamata oggetto se viene effettuata una associazione semantica ad essa.

Def. Segmentation Data un immagine I e un predicato P (criterio di omogeneità basato su una feature visuale), la image segmentation significa dividere I in un set di regioni $R_1 \dots R_n$ tale che le regioni in cui l'immagine viene suddivisa devono soddisfare alcune proprietà:

- **Distinte** Nessun pixel deve essere condiviso da due regioni
- **Complete** Tutti i pixel dell'immagine sono assegnati ad almeno una regione della partizione
- **Connesse** Tutti i pixel appartenenti alla stessa regione sono connessi
- **Omogenee** Tutte le regioni sono omogenee rispetto ad un criterio fissato (intensità, colore, texture...)

Metodi di segmentazione

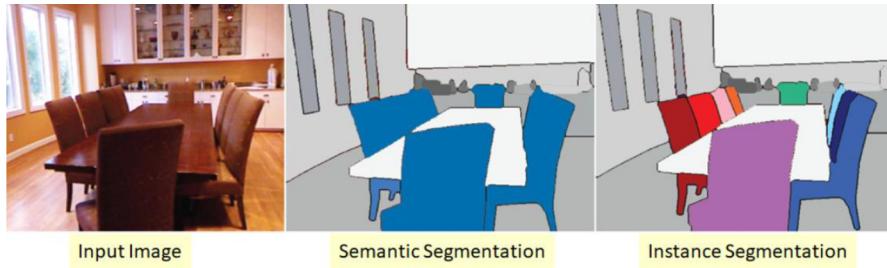
- Gray level Thresholding
- Border detection
- Color clustering
- Motion-based Segmentation
- Texture grouping

Criteri di suddivisione delle regioni

- Semantic segmentation (pixel-level classification)
- Regions and super-pixel
- Instance segmentation

Semantic Segmentation La segmentazione semantica è un problema di classificazione, ogni pixel è classificato considerando un'etichetta e dalla classificazione di ogni pixel si determinano le regioni dell'immagine. Si tratta di un problema di supervised learning, oggi è un approccio molto usato grazie all'uso di grandi database di immagini annotati, vedi ImageNet.

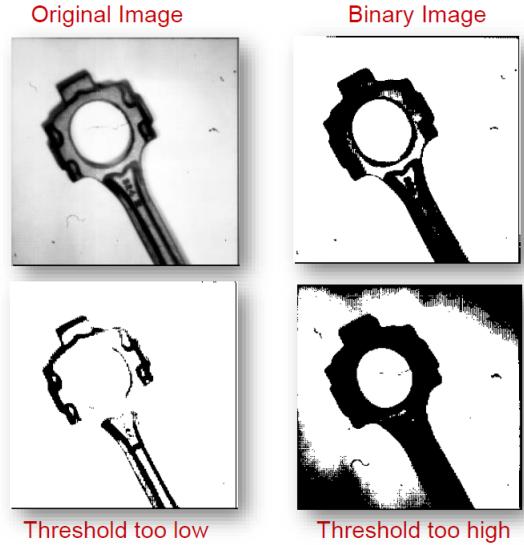
Instance Segmentation Un approccio più evoluto della semantic segmentation è l'**instance segmentation**. Con l'instance segmentation le cose si complicano perché non basta individuare regioni che sottostanno ad uno stesso criterio semantico, ma ogni oggetto che fa parte di queste regioni viene a sua volta segmentato. Necessita per questo motivo di una conoscenza a priori del contesto.



8.1.1 Metodo di segmentazione - Gray level Thresholding

Il thresholding o sogliatura è il metodo più antico di segmentazione dell'immagine. Da un'immagine a livelli di grigio, la sogliatura restituisce un'immagine binaria, a cui a sua volta viene associato il cosiddetto **bimodal histogram**. Durante un processo di sogliatura, i singoli pixel dell'immagine sono catalogati come "pixel oggetto" se il loro valore è maggiore di una certa soglia e come "pixel di sfondo" se il valore è sotto la soglia. Solitamente l'immagine binaria in uscita ha valore pari a "1" dove è presente l'oggetto e pari a "0" per lo sfondo, ottenendo quindi un'immagine in bianco e nero. Il problema di un metodo come la sogliatura è la scelta della soglia, perché è necessario individuare un valore T di intensità capace di dividere l'immagine in regioni di pixels. La soglia può essere:

- **Globale** se $T = T(f)$, quindi applicata in maniera uniforme su tutta l'immagine
- **Adattiva** se $T = T(f, W(i,j))$, se dipende dalla regione o window W dell'immagine in cui è calcolata.



Otsu thresholding Il metodo Otsu è un metodo di sogliatura automatica dell’istogramma nelle immagini digitali. L’algoritmo presume che nell’immagine da sogliare siano presenti due sole classi e quindi calcola la soglia ottima per separare queste due classi minimizzando la varianza intra classe. Algoritmo step-by-step:

1. Calcola per una data soglia t : $q_1(t), q_2(t)$, le probabilità a priori che un pixel appartenga o meno al gruppo 1 o al gruppo 2

$$q_1(t) = \sum_{i=1}^t P(i) \quad q_2(t) = \sum_{i=t+1}^L P(i)$$

Dove $P(i)$ è la probabilità a posteriori calcolata sull’istogramma (normalizzato).

2. Calcola la media e la varianza

$$\begin{aligned} \mu_1(t) &= \sum_{i=1}^t \frac{iP(i)}{q_1(t)} & \sigma_1^2(t) &= \sum_{i=1}^t \frac{[i - \mu_1(t)]^2 P(i)}{q_1(t)} \\ \mu_2(t) &= \sum_{i=t+1}^L \frac{iP(i)}{q_2(t)} & \sigma_2^2(t) &= \sum_{i=t+1}^L \frac{[i - \mu_2(t)]^2 P(i)}{q_2(t)} \end{aligned}$$

3. Calcola la varianza all’interno del gruppo

$$\sigma_W^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Varianza all’interno del gruppo: più piccola è la wgv, allora sono più compatti al valore medio i due gruppi

4. minimization: For each $t = 1, \dots, L$ cerca il valore minimo calcolato al punto 3

Sogliatura Adattiva (Linda Shapiro) Invece di calcolare una soglia univoca per tutta l'immagine, si calcola T solo su una finestra W(i, j). Le dimensioni della finestra dipendono dal problema e dalla variabilità dei dati. T può essere:

- $T = \text{mean}(W)$
- $T = \text{median}(W)$
- $T = (\text{Max}(W) - \text{Min}(W)) / 2$
- $T = \text{mean}(W) - C$ con C costante globale (metodo più efficace)

Algoritmo step-by-step:

1. An initial threshold T is chosen, randomly or according to any other method.
2. The image is segmented into object and background pixels as described above, creating two sets; given $f(p)$ the pixel value:

$$\begin{aligned} G_1 &= \{f(p) : f(p) > T\} \text{ (object pixels)} \\ G_2 &= \{f(p) : f(p) \leq T\} \text{ (background pixels)} \end{aligned}$$

3. The average of each set is computed.
4. A new threshold is created that is the average of m_1 and m_2

$$\begin{aligned} m_1 &= \text{mean } G_1 \quad m_2 = \text{mean } G_2 \\ T' &= (m_1 + m_2) / 2 \end{aligned}$$

5. Go back to step two, now using the new threshold computed in step four, keep repeating until the new threshold matches the one before it (i.e. until convergence has been reached).

L'algoritmo iterativo introdotto da Shapiro negli anni 80 è un caso speciale unidimensionale del k-means, il quale ha provato che si può convergere ad un minimo locale. Cerca di ottenere il partizionamento dei dati minimizzando il valore della somma dei quadrati delle distanze di ciascun pattern con il prototipo più vicino (criterio del mean square error). Genera K cluster C1, C2,...,CK tali che il pattern ni appartiene esattamente ad un unico cluster e quel cluster sia Ci.

- Repeat:

1. Compute the centers of classes m1 and m2

$$\begin{aligned} m_1 &= \frac{1}{N_1} \sum_{i=1}^{N_1} g_1(i) \\ m_2 &= \frac{1}{N_2} \sum_{i=1}^{N_2} g_2(i) \end{aligned}$$

2. Redefine the two groups so that

$$\begin{aligned}|g_1(k) - m_1| &< |g_1(k) - m_2| \quad k = 1..N_1 \\ |g_2(k) - m_2| &< |g_2(k) - m_1| \quad k = 1..N_2\end{aligned}$$

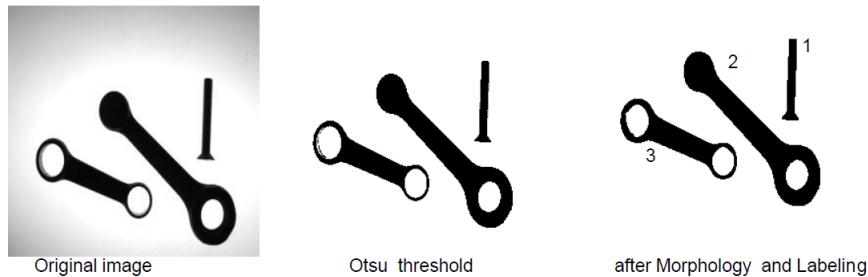
- All grey levels in set 1 are nearer to cluster centre m1 and all grey levels in set 2 are nearer to cluster centre m2
- Repeat it until none of the pixel labels changes anymore.

Esempio di K Means



8.2 Labeling

Labeling significa identificare segmenti di pixel che sono “connessi” o “adiacenti” con la stessa label. Come abbiamo visto fino ad ora, la segmentazione si occupa di definire il background e foreground, ma nella stessa immagine possono esserci oggetti diversi che quindi hanno bisogno di etichettature diverse.



8.2.1 Pixels distance in images

I problemi che sorgono con la labeling sono causati dal fatto che non si è ancora stabilito un concetto univoco di adiacenza e distanza tra due pixels. Le misure di distanza che possono essere applicate sono:

- **Distanza Euclidea** (computazionalmente più complessa)

$$\text{De } (i, j)(h, k) = \sqrt{(i - h)^2 + (j - k)^2}$$

- **Distanza CityBlock** the minimum number of steps in a grid to reach a point from another one

$$D4(i,j)(h,k) = |i - h| + |j - k|$$

- **Distanza Chessboard** The number of steps for a king in a chessboard (è in grado di muoversi in diagonale sui pixel)

$$D8(i,j)(h,k) = \max\{|i - h|, |j - k|\}$$

Connected component labeling (CCL) Trasforma un'immagine binaria in input (già effettuata la segmentazione background foreground) in una simbolica in cui tutti i pixels che appartengono allo stesso componente hanno la stessa label

8.2.2 Connessione tra i pixel

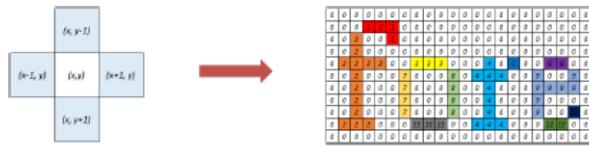
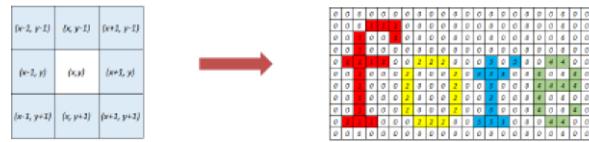
E' possibile definire la connessione tra i pixel in due modi:

- 4-neighborhood

$$N_4(p) = \{q \in L \mid |p_x - q_x| + |p_y - q_y| \leq 1\}$$

- 8-neighborhood

$$N_8(p) = \{q \in L \mid \max(|p_x - q_x|, |p_y - q_y|) \leq 1\}$$



8.3 Categorie di Labeling

Gli algoritmi di labeling possono dividersi in 3 macro categorie. Ricordiamo la definizione di CCL Labeling: algoritmi che associano un'unica etichetta ad ogni regione connessa dell'immagine.

1. Multiscan

- Algoritmi iterativi (lenti ma con una buona allocazione delle risorse)
- Algoritmi classici

2. 2-steps

- Algoritmi classici
- Run-length
- Decision table e algoritmi con Tree

3. labels-propagation algorithm

- Contour tracing and filling (openCV)

8.3.1 Labels Propagation Algorithms

Uno degli approcci più utilizzati per effettuare la labelling è la labels propagation. Si tratta dell'approccio più utilizzato negli ultimi anni, presenta due passi fondamentali: nella prima fase viene assegnata una label ad ogni pixel e nella seconda viene propagata la label ai pixel adiacenti (l'adiacenza può essere 4-neighborhood o 8-neighborhood), ricapitolando:

1. Ricerca con raster scan e etichettare i pixels senza label; (inizializzazione)
2. Propagare la stessa label sui pixels che sono connessi fra di loro.

Il punto due, cioè la propagazione delle label, può essere implementato tramite più algoritmi, i due principali sono l'iterative multiscan e il 2-steps.

8.4 Algoritmi Multiscan

8.4.1 Classic Labeling Rosenfeld (1966)

- È l'approccio classico al problema del labeling
- Utilizza un **raster scan** sull'immagine (Raster scan = linee di scansione, sono le linee in cui è suddivisa un immagine sottoposta a scansione elettronica).
- In output fornisce l'immagine con le sue label calcolate dall'algoritmo e fornisce anche un **equivalence table** in cui sono contenute le “ridondanze” (equivalenze) dei pixels.
- Le ridondanze vengono rimosse tramite algoritmi di ordinamento. Dalla tabella ordinata si aggiornano le label sull'immagine

Debolezze dell'algoritmo Il problema di questo algoritmo è il suo costo computazionale, bisogna infatti memorizzare sia l'immagine che l'equivalence table, inoltre bisogna applicare un algoritmo di sort.

8.4.2 Classic Iterative Multiscan Labeling (Haralick 1981)

Partendo dall'immagine già segmentizzata e quindi binarizzata, in questo approccio **non viene usata l'equivalence table**, ma si effettuano iterativamente dei passaggi di forward e di backward con il raster scan sull'immagine di output per risolvere le ridondanze (equivalenze). Nei passaggi di risoluzione si lavora sui pixels di neighborhood. Questo approccio è meno costoso a livello di memoria, ma è comunque computazionalmente complesso mano a mano che la dimensione dell'immagine binaria aumenta.

8.4.3 Iterative multiscan Algorithm

L'algoritmo consiste in:

1. **Inizializzazione** ogni pixel dell'immagine è numerato con una label consecutiva

0 1 1 0 1 1 0	0 1 2 0 3 4 0
0 1 1 0 1 1 0	0 5 6 0 7 8 0
0 1 1 1 1 1 0	0 9 10 11 12 13 0

Figure 9: Prima e dopo l'inizializzazione

2. **Top down** si scannerizza dall'alto verso il basso da sinistra a destra per dare la stessa label a pixels connessi
3. **Bottom up** si scannerizza dal basso verso l'alto da destra a sinistra per dare la stessa label a pixel connessi
4. **Ripetere step II e III** tante volte quante servono per far smettere alle label di cambiare

top-down:	bottom-up:
0 1 1 0 3 3 0	0 1 1 0 1 1 0
0 1 1 0 3 3 0	0 1 1 0 1 1 0
0 1 1 1 1 1 0	0 1 . 1 1 1 1 0

Figure 10: Applicazione dei punti 2 e 3

Debolezze dell'algoritmo Molto lento per via del fatto che il numero di iterazioni da fare può essere molto grande.

8.5 Algoritmi 2-step

8.5.1 Two scan algorithm + Equivalence table

L'algoritmo fa largo uso dell'**equivalence table**

1. Quando ad un pixel possono essere assegnati più di una label, gli viene assegnata la label con valore più basso e le altre label vengono memorizzate in una tabella detta **equivalence table**
2. La tabella viene scansionata e ordinata assegnando ogni volta ai pixel la label più bassa fino a che tutte le equivalenze non sono risolte. NB: si scelgono le label più basse perché meno ne ho più il labelling è uniforme

image	First step	Equivalence
0110110	0 1 1 0 2 2 0	
0110110	0 1 1 0 2 2 0	
0111101	0 1 1 1 1 0 3	
		1=2
		2=3
	0 1 1 0 1 1 0	
	0 1 1 0 1 1 0	
	0 1 1 1 1 1 0	

Debolezze dell'algoritmo Potrebbe volerci molto tempo per processare la tabella di equivalenza

8.5.2 Altri algoritmi 2-Steps

Run Length E' un approccio nato per evitare di memorizzare l'immagine di output e quindi risparmiare memoria. L'algoritmo Run-length è il primo algoritmo di compressione per le immagini. Si tratta di un metodo di tipo lossless (senza perdita di dati) che riduce la ripetizione di elementi uguali all'interno del file. Per ogni elemento trovato uguale, l'algoritmo sostituisce un RUN, ovvero un insieme di 3 caratteri: il primo indica il codice identificativo del RUN, il secondo il carattere sostituito, ed infine il terzo il numero di volte che il carattere deve essere ripetuto.

Decision tree La struttura dati utilizzata per gestire la risoluzione della label è implementata utilizzando tre array per collegare gli insiemi di classi equivalenti senza l'uso di un puntatore. Adottando questa struttura dati, sono stati quindi proposti due algoritmi:

- Viene utilizzata una prima scansione run-based, in cui un **decision tree** ottimizza l'esplorazione sui neighborhood per applicare il merging solo quando necessario.
- Grana et al. 2010: migliorarono l'approccio al punto 1 con l'utilizzo della **decision table** per implementare un decision tree applicabile anche su grandi neighborhood

NB: il primo metodo utilizza un decision tree mentre il secondo un decision table

Approccio generale:

1. Nel primo passo, si assegna una label provvisoria per ogni “oggetto” nell’immagine
2. Nel secondo passo, si integrano le label provvisorie ad una sola

Pro:

- Sono cache friendly
- Possibilità di implementazioni parallele dato che sono algoritmi di tipo raster-scan

Cons:

- Il calcolo delle features shape richiede una spesa computazionale abbondanza onerosa
- Necessitano di memoria per memorizzare l’equivalenza tra le label

Decision Table Si tratta di un approccio importante. L’approccio con decision table è tipico dell’intelligenza artificiale. Permette la costruzione di un algoritmo che permette di selezionare automaticamente la scelta migliore tra tutte.

1. Model the neighborhood exploration problem with decision tables
2. Construct the decision table
3. Convert decision tables to optimal decision trees
4. Automatic code generation from the decision tree

Il decision table può essere di tipo AND o OR:

- AND Decision Table: si devono effettuare tutte le azioni
- OR Decision Table: si possono effettuare una delle azioni equivalenti

8.5.3 Conclusioni

È importante quindi sapere che con segmentation non si effettua solo un’operazione di classificazione dei pixels, ma anche di aggregazione. Labeling e segmentation permettono di distinguere un oggetto dall’altro in un’immagine.

8.6 Segmentation of foreground and background (Scontornamento delle Immagini)

L'approccio che vedremo da questo momento in poi non assume più di avere immagini semplici e binarie. Si chiama infatti **interactive segmentation**. Possiamo dividere gli algoritmi di FB segmentation in 3 macrocatergorie:

1. CLUSTERING METHODS IN TWO CLASSES

- **Region splitting**
- **Region merging**, clustering agglomerativo
- **Superpixel method**

2. K-means based

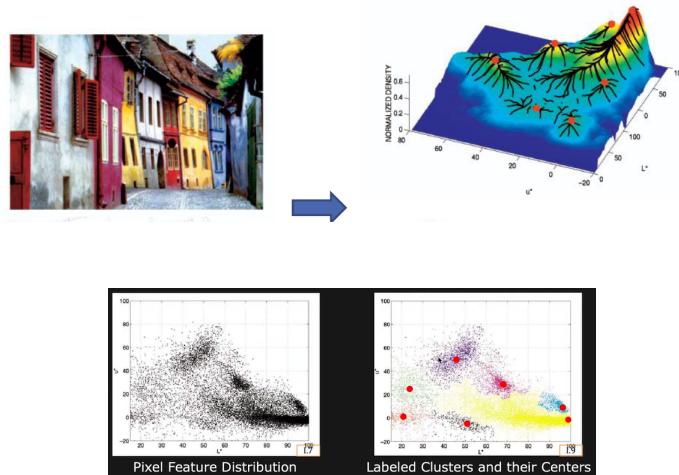
- **K-means clustering + Gaussian** (modello parametrico)
- **Mean-shift**

3. GRAPH BASED METHODS (importanti)

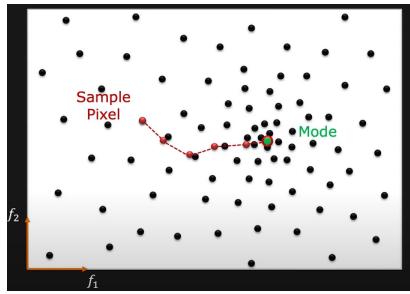
- **Normalized cut**
- **Grabcut e graphcut**

8.6.1 Mean shift

Utile per F/B Clustering e per multiple region segmentation, sfrutta i feature vectors dei pixel dell'immagine (es: colore e posizione) considerandoli come una densità di probabilità in maniera tale da trovare i clusters (le mode) all'interno della distribuzione (l'immagine).



Spiegazione del Mean Shift Consideriamo l'immagine come un insieme di pixel sparsi in base al colore, il nostro obiettivo è quello di suddividere l'immagine in regioni (o clusters) dove ogni regione contiene i pixel simili a livello di colore. Partiamo scegliendo a caso un primo pixel e considerando una finestra di dimensione W centrata intorno al punto calcoliamo la media di tutti i pixel all'interno della finestra, dopodichè selezioniamo il pixel più vicino alla media e spostiamo la finestra centrandola sul nuovo pixel, ripetiamo quindi il processo fino alla convergenza. Ripetiamo il processo per tutti i pixel e i pixel che convergono nello stesso punto appartengono allo stesso cluster.



Algoritmo Mean shift

1. Choose kernel and bandwidth h
2. For each point:
 - (a) Center a window on that point
 - (b) Compute the mean of the data in the search window
 - (c) Center the search window at the new mean location
 - (d) Repeat (b,c) until convergence
3. Assign points that lead to nearby modes to the same cluster

Questo algoritmo può essere utilizzato anche per semplice tracking, se applicato su frames di immagini successive.

8.6.2 Segmentation by graph-based methods

Quando le immagini sono complesse quindi non riconducibili a binarie si utilizzano dei grafi per poter segmentarle, si tratta di metodi interattivi. L'obiettivo è quello di avere label binarie per il background e il foreground.

Approccio Ottimizzazione di un funzione di Energia (Labeling) = data + smoothness, la funzione deve essere minimizzata. dove:

- **Data** per ogni pixel, la probabilità che appartenga a B o a F

- **Smoothness** per coppie di pixel nello stesso neighborhood, si associa una penalità (aumentando i pesi) nel caso in cui i pixel siano di colori molto diversi

Dall'immagine al grafico

- Ad ogni pixel corrisponde un nodo (vertice)
- Arco tra ogni coppia di pixel
- Ogni arco che connette una coppia di vertici è pesato dalla sua affinità o similarità (w_{ij})

La similarità può essere sia a livello di colore che a livello di spazio all'interno dell'immagine. Con un algoritmo di tipo Graph Cuts è possibile dividere il grafico in segmenti e quindi l'immagine in regioni. Si tagliano gli archi che hanno bassa similarità, pixels simili dovrebbero essere nello stesso segmento di grafico.

Algoritmi di segmentazione basati sui grafici

- Min Cut
- Normalized Cut
- Graphcut
- Grabcut con Graphcut

8.6.3 Min Cut

Dato un grafo $G = \{V, E, W\}$, dove:

- **V** è il nodo
- **E** i suoi edges
- **W** la matrice di affinità

La matrice di affinità associa un peso ad ogni edge in E. Il taglio (cut) di un grafo è la partizione di V in due sotto-insiemi A e B tali che:

$$A \cup B = V, A \cap B = \emptyset$$

Il min cut di un grafico è il taglio che partiziona G in due segmenti disgiunti tali che la somma dei pesi associati agli edges tra i due segmenti è minima.

$$C_{\min}(A, B) = \sum_{u \in A, v \in B} W_{uv}$$

Si tratta di un problema NP-Hard. Min cut algorithm selects to cut where the edge have a small weight. It has the drawback to separate singleton or small groups

8.6.4 Normalized cut graph

Il taglio del normalized cut sopperisce al problema del min cut che penalizza i segmenti larghi.

$$N\text{cut}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)}$$

$$\begin{aligned}\text{cut}(A, B) &= \sum_{u \in A, v \in B} w(u, v) \\ \text{assoc}(A, V) &= \sum_{u \in A, t \in V} w(u, t)\end{aligned}$$

- **assoc(A/B, V)** somma dei pesi di tutti gli edges che toccano A o B, dove w è la similarità tra i pixels.
- **cut(A,B)** somma di tutti i pesi dal nodo A a quello B.

Calcolo del peso tra due pixel I pesi possono essere ottenuti tramite un metodo supervised oppure definiti in maniera unsupervised come il prodotto di due fattori:

- **L'affinità in termini di colore** preso un kernel gaussiano, considerando la distanza del vettore colore $F(i)$ e $F(j)$;
- **L'affinità in termini di prossimità spaziale** se i pixels sono distanti più di una certa soglia definita r , per evitare calcoli inutili allora la prossimità è 0, oppure si definire un kernel Gaussiano per la misura della distanza

$$\begin{cases} w_{ij} = e^{\frac{-\|F(i) - F(j)\|_2^2}{\sigma_i^2}} & \text{if } \|X(i) - X(j)\|_2 < r \\ w_{ij} = 0 & \text{otherwise} \end{cases}$$

Note that small sigma grop only very near points and Large sigma accept affinity of far away points

Pro e Contro

- **Pro** si tratta di un framework generico, può essere usato con diverse features e formule di affinità, permette di avere dei segmenti regolari
- **Contro** bisogna scegliere il numero di segmenti, richiede un alto livello di storage e complessità di tipo NP Hard, bias per la partizione in segmenti uguali
- **Utilizzo** ottimo nel caso in cui si vogliano segmenti regolari.

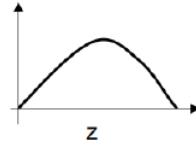
8.6.5 GraphCut

GraphCut è una generalizzazione della normalized cut che utilizza istogrammi.

- The image is an array $z = (z_1, \dots, z_N)$ of grey values indexed by the single index n .
- The segmentation of the image is an alphachannel, or, a series of opacity values $\alpha = (\alpha_1, \dots, \alpha_N)$ at each pixel with $0 \leq \alpha_n \leq 1$
- The parameter θ describes the foreground/background grey-level distributions. i.e. a pair of histogram of gray values:

$$\theta = \{h(z; \alpha), \alpha = 0, 1\}$$

\iff



- Note that these histograms are directly assembled from the trimaps T_B and T_F
- Re-pose the segmentation task: The segmentation task is to infer the unknown opacity values α from image z and the model θ .

Segmentazione come la minimizzazione di energia una funzione di energia E è definita così che il suo minimo corrisponda a una buona segmentazione. Si utilizza la formula della Gibbs Energy

$$E(\alpha, \theta, z) = U(\alpha, \theta, z) + V(\alpha, z)$$

Dove U valuta l'adattamento dell'opacità α ai dati z (cioè dà un buon punteggio (punteggio basso) se α sembra che sia coerente con l'istogramma).

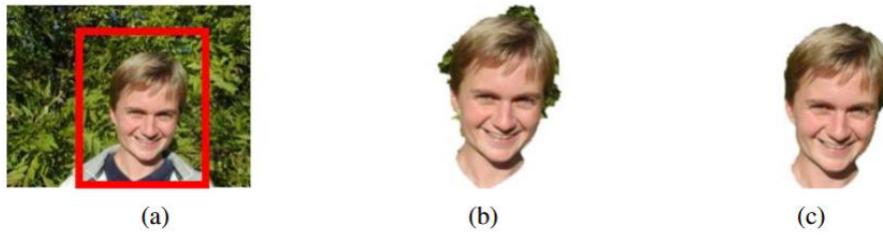
$$U(\alpha, \theta, z) = \sum_n -\log h(z_n; \alpha_n)$$

V è una termine di smoothness che penalizza nel caso in cui ci sia troppa disparità tra pixels dello stesso neighborhood.

8.6.6 GrabCut

L'idea di GrabCut è la seguente:

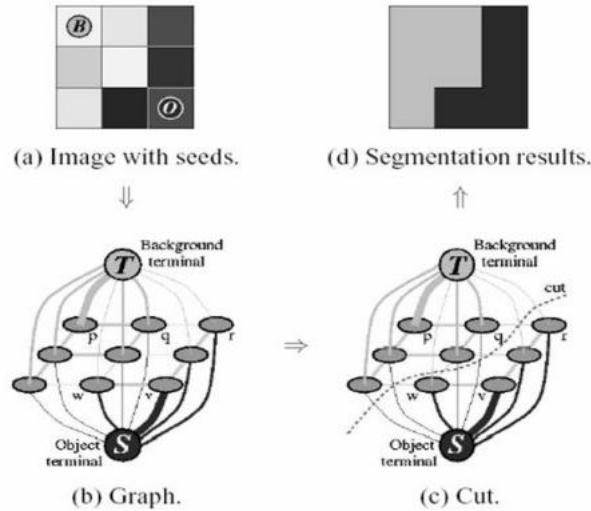
1. Inizialmente l'utente applica una bounding box intorno al foreground, il foreground deve essere completamente all'interno del rettangolo poiché tutto ciò che è al di fuori viene inizializzato come background. Quello che è presente all'interno della bounding box invece deve essere classificato tra background e foreground pixel per pixel.
2. L'algoritmo determina la distribuzione di colori per il foreground (l'oggetto) e per il background.
3. L'algoritmo applica una binary segmentation;
4. Il processo è ripetuto finché la distribuzione di colori migliora.



- GrabCut usa GraphCut.
- Per descrivere la distribuzione del colore dei pixel nel foreground e nel background viene utilizzato il Gaussian Mixture Model (GMM), poiché ogni pixel è in RGB ed è quindi preferibile all'uso degli istogrammi.
NB: A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.
In questo caso si utilizzano due distribuzioni Gaussiane con full-covariance e k componenti, due perché uno è per il background e l'altro per il foreground.
- In base ai dati che gli diamo (oltre al posizionamento dei bounding box, l'utente può anche selezionare parti di immagine come se fosse lo strumento pennarello di paint e specificare se si tratta di background o foreground), il GMM impara e crea nuove distribuzioni dei pixel (per ogni pixel gli assegna una probabilità che appartenga al foreground o al background basandosi sulla similarità di colore con gli altri pixel già etichettati).
- Da questa distribuzione di pixel viene quindi creato un grafo dove ogni nodo corrisponde un pixel. Inoltre sono presenti due nodi chiamati **Source Node** e **Sink Node**: ogni pixel di Foreground è connesso con il Source

node e ogni pixel di Backgorund è connesso con il Sink node e il peso di questi archi dipende dalla probabilità di appartenere a queste due categorie; se c'è una grande differenza di colore allora il peso dell'arco sarà basso.

- Viene utilizzato il mincut algorithm per segmentare il grafo, l'algoritmo taglia il grafo in due separando il nodo source e il nodo sink con la funzione di minimum cost (che ricordiamo essere la somma di tutti i pesi dei archi tagliati). Dopo il taglio, tutti i pixel collegati al nodo Source diventano foreground e quelli collegati al nodo Sink diventano backgorund.
- Il processo è iterato fino alla convergenza



Di solito Grabcut itera dalle 5 alle 10 volte.

- pro
 - È molto veloce a livello di inferenza
 - Può essere usato per recognition o high-level priors
 - Si applica ad un largo range di problemi (stereo, image labeling, recognition)
- contro
 - Non si può applicare in tutti i casi (solo quelli associativi)
 - Ha bisogno di termini unari (non utilizzabile per segmentazione generica)
 -

Purtroppo nel caso di molte immagini anche metodi efficienti possono risultare lenti. Una soluzione è Superpixels.

8.6.7 Superpixels segmentation

A superpixel could be defined as a group of **pixels that share common characteristics** (intensity, position, color).



Superpixels are useful in many Computer Vision and Image processing algorithms, because:

- They carry **more information** than pixels.
- They have a **perceptual meaning** (pixels belonging to a given superpixel share similar visual properties).
- They provide a convenient and **compact representation of images** (very useful for computationally demanding problems).

Gli aspetti negativi dell'uso di questo metodo di segmentazione sono:

- il calcolo dei superpixels è computazionalmente pesante
- c'è il rischio di perdere informazioni importanti in merito agli edges.

8.6.8 SLIC Simple Linear Iterative Clustering

L'algoritmo SLIC utilizza il concetto di superpixels.

Definizioni

- **N** Numero di pixel nell'immagine
- **K** Numero di superpixels
- $\frac{N}{K}$ Area di un superpixel (mediamente)
- $S = \sqrt{\frac{N}{K}}$ Distanza tra il centro di due clusters

Premesse

- Nella prima versione dell'algoritmo, i K superpixel sono inizialmente disposti a griglia sull'immagine con distanze prefissate, un'altra versione prevede che i superpixel si dispongano sul piano nei punti a gradiente minimo (più basso), questo passaggio viene fatto per evitare di posizionare il centro del cluster sugli edge o su del rumore.

- Un singolo super pixel è definito da 5 valori:

$$C_k = [L_k, a_k, b_k, x_k, y_k]$$

- L, a, b sono i valori del CIELAB colorspace: L per la luminosità a e b per le dimensioni colore-opponente.
- x, y le coordinate dei pixels

Calcolo della distanza tra Superpixels

$$d_{Lab} = \sqrt{(L_k - L_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

Il calcolo della distanza tra due pixel tiene conto della similarità del colore più la distanza euclidea.

$$D_s = d_{Lab} + \frac{m}{S} d_{xy}$$

NB: la distanza tra due colori di pixels nello spazio CIELAB è limitata, mentre la distanza xy spaziale tra due pixels dipende dalla dimensione dell'immagine. Per questa ragione prima di calcolare la distanza euclidea è necessario normalizzare la distanza spaziale.

Algoritmo

1. Si prendono K cluster centers con dimensione approssimativamente uguale.
2. Si dispongono sul piano nei punti a gradiente minimo
3. Si ripete fino alla convergenza:
 - Per ogni C_k :
 - Si trovano i pixel simili in un intorno $2S \times 2S$: ogni pixel viene associato al center cluster più vicino.
 - Ogni volta che un pixel viene associato ad un cluster center il centro viene di nuovo ricalcolato come la media dei vettori Labxy di tutti i pixels associati a quel cluster.

Superpixel sampling network L’evoluzione di SLICN è connessa alla volontà di usare deep learning in questo tipo di approcci. Superpixel sampling network, definisce le features associano una Deep Network ad ogni pixels (invece di CIELABxy). La differenziabilità non è più ottenuta usando nearest neighborhood nel clustering, ma ottenendo una loss differenziabile con SLIC.

8.7 Saliency

La visual saliency è una qualità percettiva e soggettiva che data una scena del mondo reale, fa risaltare alcuni elementi rispetto a quelli ad altri, cioè gli elementi salienti sono quelli che attirano l’attenzione visiva di chi guarda. L’elaborazione neurale delle informazioni visive è influenzata da fattori cognitivi topdown, come l’aspettativa, la memoria o l’attenzione selettiva. Questi fattori cognitivi creano una motivational salience map che una volta integrata con la stimulus-salience map forma la priority map che guida il comportamento visual-motorio.

- **Saliency map** calcola dei punti sull’immagine evidenziandoli, il calcolo avviene basandosi su features visive di basso livello come i colori più vivaci, gli edges orientati e la motion;
- **Priority map** in questa mappa vengono integrate le informazioni ottenuto con bottom-up nella saliency map con quelle relative al task e all’obiettivo iniziali;

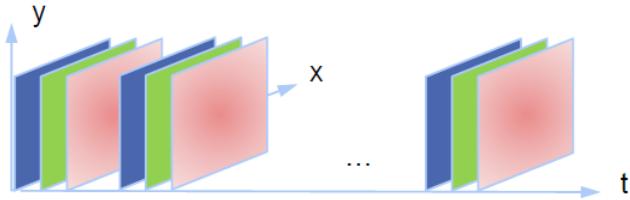
Saliency by deep learning SAM approach Il Saliency Attentive Model (SAM) è un modello per la predizione della salienza su immagini basato interamente su tecniche di Deep Learning. È composto da una rete convolutiva ricorrente che rifinisce la mappa di salienza predetta in modo iterativo grazie a meccanismi neurali attentivi che identificano le aree più importanti della scena. Inoltre, l’architettura introdotta tiene conto di un’importante proprietà dello sguardo degli esseri umani: quando l’immagine non contiene regioni particolarmente rilevanti, gli esseri umani tendono a focalizzarsi maggiormente sul centro della scena portando così ad avere un forte bias centrale nelle mappe di salienza. Per incorporare questo bias centrale, SAM integra una serie di “prior map”, imparate durante la fase di addestramento, che permettono di pesare maggiormente il centro della scena durante la predizione delle mappe di salienza.

9 Vision and Motion

Un video è una sequenza ordinata di immagini (frames) acquisite in una serie di instanti di tempo t_k in cui la differenza tra un tempo di acquisizione è il successivo è costante $\rightarrow D_t = t_{k+1} - t_k$ per $k = 0 \dots n - 1$. Un video è quindi una lista di frame definibile tramite l'equazione:

$$V(x, y) = [f_0(x, y), f_1(x, y), \dots, f_{n-1}(x, y),]$$

che definisce uno spazio tridimensionale $[(x, y), t]$. Ogni frame che compone il video viene al solito codificato in un formato video come ad esempio RGB e processato come abbiamo visto per le singole immagini. Una cosa da attenzionare è sicuramente la dimensione dei file video che rispetto alle singole immagini vedono un consumo di memoria molto più considerevole.



9.1 Computer Motion

È impossibile comprendere completamente il movimento attraverso delle immagini, è possibile solamente interpretare ciò che è deducibile dalle immagini. La deduzione del movimento può avvenire sia attraverso il cambiamento di posizione di un oggetto osservato in frame consecutivi ma anche osservando un'unica immagine in cui la posizione instabile dei soggetti fa comprendere la possibilità di un movimento. Un passo successivo alla comprensione del movimento è il **tracking**, ovvero il processo con cui si cerca di localizzare un determinato oggetto dello spazio e nel tempo, per fare ciò ovviamente la comprensione del movimento dello stesso è fondamentale. Lavorando sul movimento, la conoscenza del sistema di acquisizione con cui lavoriamo è fondamentale. Ed in base al setting possiamo definire quattro livelli di difficoltà crescente:

1. Analisi del movimento da una camera fissa
2. Analisi del movimento da una camera con movimenti ben definiti
3. Analisi del movimento da camere il cui i movimenti possibili sono ignoti
4. Analisi del movimento attraverso immagini presi dal punto di vista di un soggetto o oggetto i cui movimenti sono ignoti a priori (egocentric motion)

9.2 Tracking

Il tracking è definito come l'analisi di sequenze video con lo scopo di stabilire la posizione di un target attraverso una sequenza di frame, è quindi necessario dare una coerenza temporale all'oggetto considerato. Per fare ciò il tracking guarda al passato in modo da poter predire il futuro. Questo problema racchiude in sé la detection dell'oggetto, la sua localizzazione spaziale e la predizione del suo movimento. Esistono più tipi di tracking:

- **SO-T** tracking di un singolo oggetto
- **MO-T** tracking di oggetti multipli

9.2.1 Tracking Measures

definiamo:

- n_t^i il numero di true positives nel frame i-esimo, quindi il numero di **istanze corrette**
- n_{fp}^i il numero di falsi positivi nel frame i-esimo
- n_{fn}^i il numero di falsi negativi
- a TP, FP e FN si aggiunge anche il concetto di **Deviazione** che identifica, appunto, la deviazione dalla posizione reale della finestra di tracking proposta dal mio algoritmo.

nel caso di SO-T: $n_t^i, n_{fp}^i, n_{fn}^i = (0, 1)$ perché avendo un singolo oggetto e quindi una singola istanza di tracking questa può essere binariamente o corretta o non corretta, ovviamente nel caso di tracking multiplo i valori possono essere maggiori di 1.

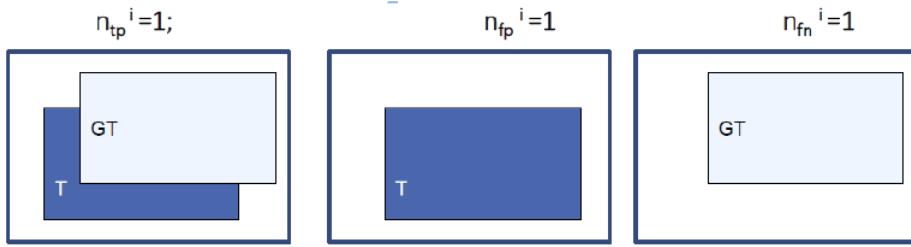
Standard IoU Una misura molto comune è l'intersection over union.

$$\frac{|T^i \cap GT^i|}{|T^i \cup GT^i|} \geq Th$$

dove:

- GT^i è il ground truth dell'immagine (verità assoluta) al frame i-esimo.
- T^i il target da noi rilevato sempre al frame i-esimo.

Attraverso questi due valori è possibile definire il grado di match a livello di pixel e fissando una soglia distinguere anche i casi di True positive, False negative e False positive, questa soglia viene per esempio considerata a 0.5 nel caso della misura **PASCAL** (se non consideriamo lo 0.5, la soglia viene detta **DICE**). Attraverso l'uso di una soglia possiamo andare a definire il numero di casi positivi e negativi (Es. Se sovrapposizione maggiore 0.5 allora $n_t^i = 1$).



A livello di oggetto, in una sequenza di n frame:

- **TP, FP, FN**

$$n_{tp} = \sum_{i=1}^{N \text{ frame}} n_{tp}^i \quad n_{fp} = \sum_{i=1}^{N \text{ frame}} n_p^{if} \quad n_{fn} = \sum_{i=1}^{N \text{ frame}} n_{fn}^i$$

- **Precision e Recall**

$$\text{Precision} = (n_{tp}) / (n_{tp} + n_{fp}) \quad \text{Recall} = (n_{tp}) / (n_{tp} + n_{fn})$$

- **F-score**

$$F = 2 \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **OTA** simile all'F-score OTA misura l'accuratezza in una frequenza:

$$OTA = 1 - \frac{\sum_{i=1}^{N \text{ frame}} (n_{fp}^i n_{fn}^i)}{\sum_{i=1}^{N \text{ frame}} g^i}$$

- **OTP** another similar to F-scores

$$OTP = \frac{1}{|Mi|} \sum_i^{\text{in } Mi} \frac{|T^i \cap GT^i|}{|T^i \cup GT^i|}$$

- **Deviazione** La deviazione rappresenta semplicemente una distanza normalizzata tra i centroidi dell'istanza reale e quella generata da me.

$$\text{Deviation} = 1 \frac{\sum_{i \in Mi} d(CTi - CGTi)}{|Mi|}$$

- **MOTA**

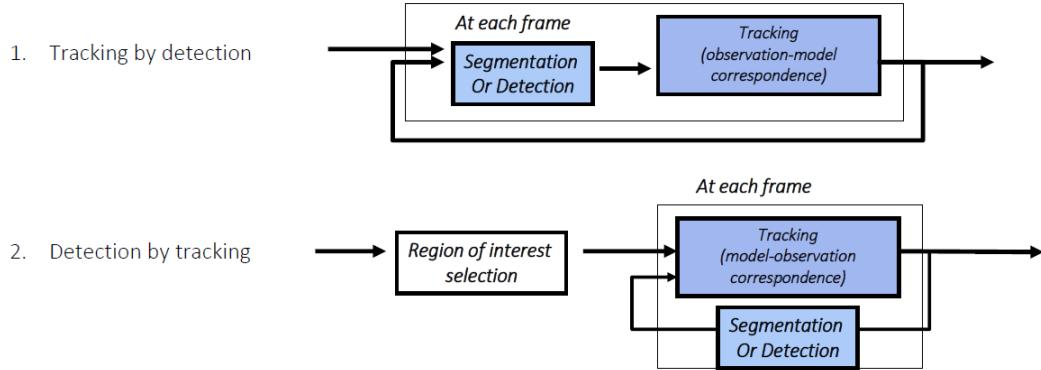
$$MOTA = 1 - \frac{\sum_{i=1}^{N \text{ frame}} (n_{fp}^i + n_{fn}^i + n_{fa}^i)}{\sum_{i=1}^{N \text{ frame}} g^i}$$

- **MOTP**

$$MOTP = 1 \frac{\sum_{i \in Mi} d(CTi - CGTi)}{|Mi|}$$

9.3 Detection and Tracking

Il tracking cerca di risolvere il problema di seguire un qualcosa attraverso il tempo e lo spazio. Ma la domanda nasce spontanea, conviene prima fare la detection degli oggetti e poi iniziare il tracking o il contrario? Dipende.



tracking-by-detection Nel tracking-by-detection paradigm per prima cosa un detector indipendente è applicato a tutti i frame dell'immagine per ottenere la detection. In seguito, un tracker viene fatto eseguire sulle detections per trovare un'associazione tra i dati così da collegare le varie detection e ottenere le traiettorie. Questo primo approccio va bene quando la detection è semplice.

detection-by-tracking Per quanto riguarda il detection-by-tracking paradigm è invece consigliato in contesti, come la robotica, in cui ci è nota una certa configurazione iniziale di ciò che vogliamo rilevare, oppure quando la detection non è semplice. In questo caso identificazione spazio-temporale dei soggetti è effettuata secondo una predizione dei suoi spostamenti ed in base a questi riusciamo a distinguere i vari soggetti della nostra attenzione.

9.4 NCC Normalized Cross Correlation

Il modo più semplice di fare tracking è attraverso la Normalized Cross Correlation. Questa è una tecnica, come abbiamo imparato nel capitolo 6, di template matching valutata con uno score che è appunto una cross correlation normalizzata rispetto al valore medio del vicinato di pixel che sto considerando. Questo approccio, definita un'area di ricerca, cerca di predire la nuova posizione con un template matching attorno alla precedente posizione nota, selezionando il candidato migliore attraverso la misura di similarità NCC. Ovviamente tale metodo soffre di un costo computazionale alto dovuto alla ricerca brute force, eseguita quindi senza un criterio su tutta l'area di ricerca, inoltre è importante

la limitazione dovuta all'assunzione di piccoli movimenti tra un frame e l'altro. Nonostante i limiti evidenziati, le prestazioni del tracking tramite NCC sono confrontabili ed a volte superiori rispetto ad una rete neurale addestrata per svolgere lo stesso compito. Questo perché anche reti "state of the art" spesso non riescono a risolvere il problema del tracking.

Considerazioni NCC

- **Assunzioni da fare:**

- La forma dell'oggetto non deve cambiare
- l'oggetto si muove di poco da un frame all'altro

- **Contro:**

- Costo computazionalmente alto dovuto alla ricerca brute-force

- **Pro:**

- Facile da implementare
- In alcuni casi sotto le dovute assunzioni torna ottimi risultati

Ncc algorithm

1. **Define a search area**
2. **Prediction of the new position** everywhere around the previous one:
Place the template defined from the previous frame at each position of the search area and compute a similarity measure between the template and the candidate
3. **Select the best candidate** with the maximal similarity measure
4. **No update** of the model

9.5 Motion Analysis

- **Motion Field** Il motion field è una proiezione del movimento in una scena 3D nell'immagine, si può facilmente dedurre che è difficile da calcolare e nel contesto di una singola camera necessita di una precisa calibrazione, che nel mondo reale non è sempre fattibile.

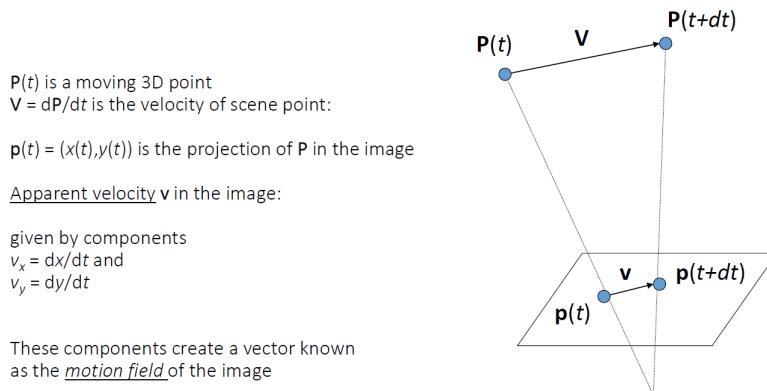


- **Optical Flow** L'optical flow può essere invece calcolato a partire dai soli dati visivi che riusciamo a ricavare da un video, il movimento è calcolato come un vettore nel punto. È un qualcosa di molto più impreciso che prende in considerazione solo l'apparente direzione del movimento. Introducendo ancora l'optical flow è opportuno dire che anche la valutazione della sua correttezza è molto complessa, questo perché non è possibile produrre annotazioni prese a mano in quanto gli umani non riescono naturalmente a computare questa misura. Per questo è nata l'esigenza di produrre dataset "sintetici", composti quindi di "cartoni animati" in cui il movimento di ogni punto è noto a priori e perciò l'optical flow può essere calcolato.



9.6 Motion Field (descrizione del movimento)

The motion field is the motion projection in the image plane. Considerando un punto $P(t)$ che si muove in uno spazio 3D e $V = \frac{dP}{dt}$ la sua velocità, dal punto di vista di un piano che rappresenta la nostra immagine quello che osserveremo sarà un punto $p(t) = (x(t), y(t))$ che presenta la proiezione del punto P sul piano. La velocità apparente rispetto al piano dell'immagine è v , data dalle componenti $v_x = \frac{dx}{dt}$ e $v_y = \frac{dy}{dt}$. Queste componenti creano un vettore conosciuto come **motion field**. Il motion field è quindi la proiezione del movimento reale sul piano dell'immagine.



Volendo calcolare la velocità v nell'immagine ed osservare come dipenda dalla velocità V nel piano reale, poiché sappiamo che p dipende dalla distanza

focale f e dall'asse Z ($p = f \frac{P}{Z}$) e che v è la derivata di p rispetto al tempo, differenziamo p e otteniamo $v = f \frac{ZV - V_z P}{Z^2}$.

$$\mathbf{V} = (V_x, V_y, V_z)$$

$$\mathbf{p} = f \frac{\mathbf{P}}{Z}$$

To find image velocity \mathbf{v} , differentiate \mathbf{p} with respect to t (using quotient rule):

$$\mathbf{v} = f \frac{Z\mathbf{V} - V_z \mathbf{P}}{Z^2}$$

Quotient rule:
 $D(f/g) = (g f' - g' f)/g^2$

$$v_x = \frac{fV_x - V_z x}{Z} \quad v_y = \frac{fV_y - V_z y}{Z}$$

Possiamo concludere notando che il movimento nell'immagine è funzione del movimento 3D del punto ma anche della profondità, espressa dalla coordinata Z , a cui avviene questo movimento, in particolare la lunghezza dei vettori che rappresentano il movimento è inversamente proporzionale alla profondità Z a cui avviene tale movimento.

• Pure translation: \mathbf{V} is constant everywhere in a plane

$$v_x = \frac{fV_x - V_z x}{Z} \quad v_y = \frac{fV_y - V_z y}{Z}$$

$$\mathbf{v}_0 = (fV_x, fV_y)$$

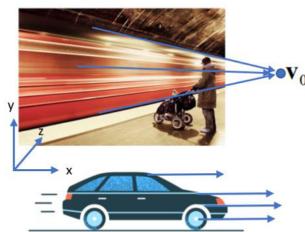
$$\mathbf{v} = \frac{1}{Z}(\mathbf{v}_0 - V_z \mathbf{p}),$$



It depends of the motion in the image plane and in the distance.

Casi Notevoli

- $V_z \neq 0$ il movimento del piano 3D ha una componente nella direzione della profondità, questo comporta che tutti i vettori che rappresentano il movimento nella nostra immagine punteranno nel verso o nel verso opposto a v_0 che per questo viene definito punto focale del movimento.
- $V_z = 0$ il movimento è parallelo al piano dell'immagine, allora tutti i vettori che rappresentano il movimento saranno paralleli tra loro



9.7 Motion parallax

Il fatto che la velocità sul piano dell'immagine è inversamente proporzionale alla distanza a cui il moto avviene crea quel fenomeno che ognuno sperimenta ogni giorno per cui oggetti a distanza minore vengono percepiti come più veloci, mentre quelli lontani come più lenti (Esempio: visti dal lontano gli aerei sembrano muoversi lentissimi). Ciò è dovuto ad un effetto inverso del fenomeno della parallasse noto come **motion parallax**

9.8 Motion estimation methods: Direct and Sparse

Metodi per il calcolo del movimento dei punti dell'immagine:

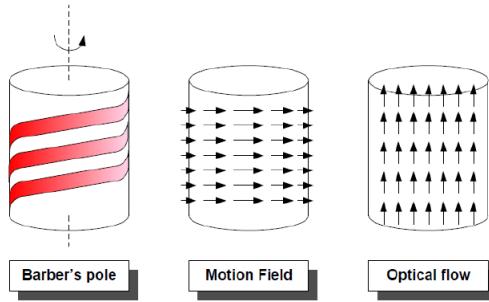
- **Diretti** I metodi “diretti” di cui fa parte il dense optical flow, cercano di stimare il movimento su tutti i punti osservando le variazioni di luminosità. Questo metodo da buoni risultati quando il movimento tra i vari frame è piccolo. La variazione di luminosità viene calcolata tramite il gradiente e la sua variazione nello spazio e tempo. Questo è il vero metodo per il calcolo dell'optical flow per ogni pixel delle nostre immagini.
- **Sparsi o Feature Based** I metodi feature-based, sparse optical flow, si basano sull'estrazione di features visuali (angoli o aree con particolari texture) e nel tentativo di seguire (tracking) nel corso di più frames. Questi metodi producono motion field sparsi, e non densi come i metodi precedenti, ma il tracking risultante è più robusto perché non basato semplicemente sulla variazione di luminosità dei pixel ma su caratteristiche visive invarianti nei nostri frame come gli angoli. Ovviamente come per tutto quello che abbiamo visto fin'ora per la produzione di un optical flow sparso esistono anche metodi deep basati sull'estrazione di features globali dall'immagine e addestrati con metodi supervisionati grazie ai dati ricavati dai dataset sintetici di cui avevamo parlato in precedenza.

9.9 Optical flow (quantificazione del movimento)

L'**optical flow** è la stima del moto apparente di un pattern con una data luminosità. Il risultato è un'approssimazione del motion field se la brightness constancy è soddisfatta. Le assunzioni chiave di questo metodo sono:

- **Color constancy** Un punto ha lo stesso aspetto nel corso di una sequenza di frame. Nel caso di immagini in scala di grigi la costanza del colore corrisponde con la costanza della luminosità (brightness constancy)
- **Small motion** I punti non si muovono molto lontano in un breve arco di tempo

Poichè l'optical flow è un'approssimazione del motion field calcolata a partire dai soli dati visivi disponibili e sfruttando il concetto di variazione di luminosità dei pixel, non sempre questo corrisponde al reale movimento che l'oggetto produce.



Generalmente possiamo definire cinque assunzioni da fare per calcolare l'optical flow:

1. **Small motion** I punti possono muoversi ad una velocità massima a cui corrisponde un massimo movimento tra un frame all'altro.
2. **Small acceleration** Tra due frame l'accelerazione è considerata trascurabile poichè è considerato trascurabile anche il tempo Δt che intercorre tra i due.
3. **Uniform movement** Tutti i punti degli oggetti si muovono rigidamente nella stessa direzione (no rotazioni).
4. **Spacial coherence** La luminosità di un punto dipende solo dalle coordinate del punto.
5. **Rigid body** Si suppone che la forma dell'oggetto in movimento non cambi nel tempo.

Un'assunzione semplificativa che faremo per permettere il calcolo dell'optical flow è quella della **brightness constancy**, questa dice che l'intensità luminosa di un pixel è costante nel tempo: Se vediamo uno stesso punto in posizioni diverse questo può essere dovuto solo al movimento.

Brightness constancy In caso di luminosità costante abbiamo:

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

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt + \dots$$

Quindi stiamo dicendo che la variazione della luminosità è uguale al suo valore iniziale più una serie di termini. Ora volendo garantire la costanza noi vorremmo annullare la variazione di questi termini rispetto al tempo:

$$\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} \frac{dt}{dt} = 0$$

Chiamando $u = \frac{dx}{dt}$ e $v = \frac{dy}{dt}$ abbiamo che l'equazione che garantisce la costanza della luminosità diventa:

$$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

Aperture problem Un movimento di un oggetto parzialmente occluso ha una direzione indistinguibile. Questo viene chiamato problema dell'apertura e ci dice che solo la velocità nella direzione del gradiente può essere definita. La velocità nella direzione perpendicolare al gradiente è invece ignota. Questa limitazione, come preannunciato, è dovuta al fatto che stiamo cercando di risolvere un problema con due incognite utilizzando una sola equazione che non basta per ricavare le due incognite u e v , ma solo per trovare una loro possibile combinazione, in particolare quella nella direzione del gradiente.

9.9.1 Lucas Kanade algorithm

è il più vecchio e più conosciuto algoritmo per l'optical flow. Si compone di 5 passi:

1. **Ipotesi:** la velocità è costante in un dato intorno di pixel, questo rappresenta il quinto vincolo (Spatial coherence) che si traduce nel dire che u e v devono essere costanti in un intorno (Per esempio 5x5).
2. Il gradiente rispetto al tempo può essere calcolato tramite la derivata all'indietro:
$$\frac{\partial I(x, y, t)}{\partial t} \cong I_n(x, y) - I_{n-1}(x, y)$$
3. Il gradiente spaziale viene invece calcolato con Sobel:

$$\begin{aligned}\frac{\partial I(x, y)}{\partial x} &\cong \sum_{i=-1}^1 \sum_{j=-1}^1 I(x+i, y+j) \cdot S_x(x+i, y+j) \\ \frac{\partial I(x, y)}{\partial y} &\cong \sum_{i=-1}^1 \sum_{j=-1}^1 I(x+i, y+j) \cdot S_y(x+i, y+j)\end{aligned}$$

4. A questo punto rappresentiamo il tutto in forma vettoriale costituendo un sistema lineare:

$$\begin{aligned}\frac{\partial I(x, y, t)}{\partial x} u(x, y, t) + \frac{\partial I(x, y, t)}{\partial y} v(x, y, t) &= -\frac{\partial I(x, y, t)}{\partial t} \\ A = \left(\frac{\partial I(x, y, t)}{\partial x} \frac{\partial I(x, y, t)}{\partial y} \right) &\quad B = \left(-\frac{\partial I(x, y, t)}{\partial t} \right)\end{aligned}$$

le equazioni considerando il vincolo di costanza in un intorno di 5x5 pixels la matrice A diventa 25x2, la matrice B diventa 25x1 mentre le mie

incognite u e v vengono rappresentati in una matrice 2x1:

$$A = \begin{pmatrix} \frac{\partial I_1}{\partial x} & \frac{\partial I_1}{\partial y} \\ \frac{\partial I_2}{\partial x} & \frac{\partial I_2}{\partial y} \\ \vdots & \vdots \\ \frac{\partial I_{25}}{\partial x} & \frac{\partial I_{25}}{\partial y} \end{pmatrix} \quad B = - \begin{pmatrix} \frac{\partial I_1}{\partial t} \\ \frac{\partial I_2}{\partial t} \\ \vdots \\ \frac{\partial I_{25}}{\partial t} \end{pmatrix} \quad A \begin{pmatrix} u(x, y, t) \\ v(x, y, t) \end{pmatrix} = B$$

5. A questo punto calcoliamo $d=(u,v)$ che minimizza $\|Ad-b\|^2$. La soluzione a tale problema di minimezzazione se l'ipotesi che $\det(A^T A) \neq 0$ è data in d dalla soluzione di $(A^T A) d = A^T b$. E la soluzione è data da:

$$\begin{pmatrix} u \\ v \end{pmatrix} = (A^T A)^{-1} A^T B$$

Questo tentativo di creare un optical flow denso ha però molti punti in cui la soluzione non è definita, in quanto in caso di costanza del colore il determinante è vicino allo zero proprio perché le derivate rispetto a x e y quasi nulle. Questo ci indica operativamente che possiamo “vedere” solo il movimento quando questo è caratterizzato da un contrasto nei colori, se non c’è contrasto, come un movimento di qualcosa di nero su di uno sfondo nero, attraverso i dati visivi a nostra disposizione non possiamo calcolarlo. Un altro problema è che l’assunzione della costanza in un intorno nei casi reali non è sempre verificata, anche per la presenza di rumore e corruzione nei frame. Questo ci porta a due interessanti conclusioni:

- Forse non è necessario calcolare l’optical flow su tutti i punti dell’immagine
→ Metodi sparsi
- Le informazioni ricavate dall’optical flow potrebbero essere usate per segmentare l’oggetto in movimento rispetto allo sfondo → Segmentation by optical flow

9.9.2 Sparse optical flow: Feature based methods (KLT method)

L’idea degli approcci sparsi è che l’optical flow non dovrebbe essere compiuto su tutti i punti ma solo su quello in cui la feature della luminosità è significativa. L’implementazione che vedremo di tali approcci è il metodo KLT (Kanade, Luca, Tomasi) per il tracking di features. Il KLT è semplicemente un algoritmo iterativo che lavora a diverse risoluzioni e che nel primo step trova i punti migliori su cui applicare l’optical flow e poi lo calcola sempre col metodo Lukas Kanade visto nel paragrafo precedente. La natura iterativa e il lavorare a diverse risoluzioni di questa soluzione ci richiama un po’ quello visto per SIFT e serve anche qui per garantire una soluzione robusta. Esiste anche un’implementazione di Sparse Optical Flow che usa i punti SIFT con ottimi risultati.

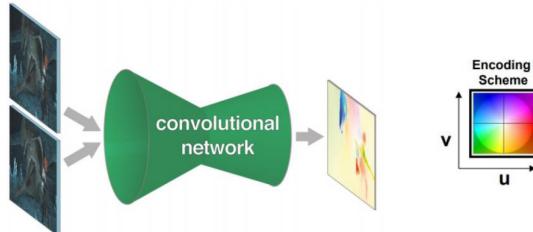
Segmentation by Optical Flow Grazie all'optical flow di ogni punto della mia immagine riesco a conoscere oltre ai soliti valori delle coordinate e del colore anche i valori del vettore di movimento associato.

$$[\underbrace{v_x, v_y}_{\text{motion vectors}}, \underbrace{x, y}_{\text{pixel coordinates}}, \underbrace{R, G, B}_{\text{color intensities}}]$$

Queste info posso essere usate in un algoritmo di clustering in modo da segmentare l'oggetto in movimento rispetto allo sfondo fermo. L'algoritmo però non funziona benissimo, questo perché in un algoritmo di clustering come il k-means si considerano far parte del cluster anche punti non in movimento poiché la vicinanza in termini di coordinate e di colore è grande. Nonostante ciò questo ci permette di individuare la zona di movimento per poi applicargli algoritmi più precisi.

9.9.3 Dense optical flow by deep learning: Flownet

Come avevamo già accennato l'uso di tecniche deep per il calcolo dell'optical flow è reso possibile da dataset sintetici, poiché annotare tale dato per noi umani sarebbe impossibile.



Come vediamo è un approccio generativo in cui prendiamo in ingresso due frame consecutivi ed in uscita tentiamo di produrre un'immagine che codifica l'optical flow attraverso i colori. La rete presenta le tipiche caratteristiche in una rete generativa con una parte contrattiva che si occupa di estrarre una rappresentazione delle immagini attraverso features (encoding) e la parte espansiva che crea a partire da queste feature l'optical flow (decoder).

FlownetSimple Flownetsimple prende in input due immagini poste una sopra all'altra (stacked) quindi un input con 6 canali. A partire da questo usa convoluzioni e pooling per fare l'encoding ed una rete simile ma specchiata per la fase di decoding che genera l'output. L'addestramento è totalmente supervisionato.

FlownetCorr Nel metodo simple mettiamo tutti i dati insieme e proviamo da questi a estrarre features significative, mentre in questo caso partiamo da

un'estrazione di features separate per le due immagini e la correlazione (unione) tra le due immagini viene fatta in seguito. In questo caso dopo tre layer uniamo le features estratte.

Flownet 2.0 Meglio delle due architetture flownet precedenti poichè capace di cogliere meglio i dettagli ma anche perchè è un'architettura molto veloce. Questa seconda versione unisce flownetsimple e flownetcorr, nei primi layer viene calcolato l'errore e viene migliorato l'output tenendo conto dell'errore commesso precedentemente. La rivoluzione importante da ricordare è il concetto di imparare dagli errori commessi ai passi precedenti espresso dal termine brightness error.

9.10 Motion estimation on moving objects

Branca della motion estimation che si occupa di segmentare gli oggetti in primo piano che si muovono assumendo uno sfondo più o meno fisso. Questi metodi trovano applicazione, per esempio, nella videosorveglianza dove le camere sono per lo più fisse e lo sfondo poco variabile.

9.10.1 Segmentation by motion with fixed cameras

Nel caso di una camera fissa possiamo semplificare la detection di oggetti in movimento, non servirà più il calcolo dell'optical flow, perché assumendo che l'equazione di costanza della luminosità sia valida possiamo considerare come punti in movimento quelli in cui osserviamo una variazione di luminosità. Possiamo esprimere ciò dicendo che consideriamo in movimento quei punti per cui osserviamo una variazione nel tempo e quindi la cui derivata rispetto al tempo è diversa da zero. Nota che qui stiamo solo detectando punti in movimento, mentre prima nell'optical flow cercavamo anche di misurare la velocità, in termini di componenti u e v, a cui questo movimento avveniva. Le metodologie utilizzate per detectare i pixel sono due:

- **Differential Methods** Guarda se la derivata rispetto al tempo dei pixel è meggiore di una soglia per dividere i punti in movimento da quelli fermi.

$$D(x, y) = \frac{dI(x, y)}{dt} > T$$

- Single difference
- Double difference
- Double difference with edge closure

- **Difference with reference data** Il secondo non guarda le differenze tra un frame e l'altro ma tra il background e l'immagine attuale.

$$D(x, y) = \Delta I(x, y) = \Delta(I(x, y), B(x, y)) > T$$

- Background suppression

Absolute Difference (Single Difference) Questo è il metodo più semplice che approssima il calcolo della derivata rispetto il tempo come la differenza in valore assoluto tra il valore del pixel in due frame consecutivi. Se questa differenza è maggiore di una soglia allora il pixel riguarda qualcosa in movimento.

$$D_n(x, y) = |I_n(x, y) - I_{n-1}(x, y)|$$

Computazionalmente è davvero poco pesante ma nei casi reali è quasi inutile, non riesce ad avere risultati precisi ed è inoltre molto dipendente dalla soglia scelta. Può essere utile solo per detectare che ci sia o meno movimento nel mio video. Trovare la soglia corretta per avere una segmentazione efficace è complesso.

Double Difference Questo secondo metodo aggiunge uno step di filtraggio e utilizza tre frame consecutivi invece che due.

$$DD_n(x, y) = (D_n(x, y) > Th) \ \& \ (D_{n+1}(x, y) > Th)$$

Quindi adesso sia la soglia tra i primi due frame, che quella tra il secondo ed il terzo deve superare la soglia T. A questo punto produco le due immagini con single difference e di queste faccio un AND, in questo modo riesco ad eliminare i due ghost che avrei prima e dopo dell'oggetto in movimento reale. Il problema è che se la distanza tra i frame non è sufficiente o comunque l'oggetto si muove più lentamente di quanto mi aspettavo i risultati possono essere pessimi.

Background suppression Come preannunciato un secondo approccio al problema è quello di vedere le differenze, non tra un frame e l'altro ma tra l'immagine corrente ed un background che prendiamo come riferimento. Il metodo si compone di tre punti principali:

- Scelta dell'immagine di sfondo
- Background suppression (può essere fatta anche con una semplice differenza tra immagine e immagine di sfondo)
- Segmentation e labelling

Il problema di questo approccio è come modellare il background, cioè come scegliere l'immagine di sfondo in maniera tale che non muti nel corso del tempo. La differenza tra un frame e lo sfondo può anche essere fatto attraverso una rete neurale.

Background Modelling Come abbiamo capito prima il vero problema nella background suppression è la modellazione dello sfondo che deve in qualche modo adattarsi all'evoluzione del contesto. Il background ideale dovrebbe quindi essere aggiornato dinamicamente, robusto rispetto alle variazioni di illuminazione, robusto rispetto ai falsi oggetti (ghosts) e con inizializzazione veloce. Ovviamente le soluzioni a questo problema sono molteplici. Per quanto riguarda

l'inizializzazione questa può essere davvero utile se sappiamo che nel nostro contesto di lavoro c'è un momento in cui nessuno sarà nella scena. Altrimenti potrei interpolare diversi frame per trovare le differenze e modellare così il mio sfondo eliminando gli oggetti superflui. Parleremo di due metodi di modelling: in primis del padre di tutti i metodi di questo genere, l'adaptive background, per poi muoversi verso un qualcosa di più raffinato, rappresentato dal metodo statistico mixture of gaussian.

Adaptive Background Questo metodo prevede l'implementazione di un semplice filtro adattivo. Quindi il background al tempo $t+1$ è una somma pesata tra il background al tempo t e l'immagine al tempo t .

$$B_{s,t+1} = (1 - \alpha)B_{s,t} + \alpha I_{s,t}$$

Il parametro alpha regola quanto mi voglio fidare del background precedente e quanto dell'immagine corrente. Con alpha uguale a zero il background è fisso e non cambia mai, mentre se è 1 vuole dire che prendo come background l'immagine corrente. Questa formula va bene se tutti gli oggetti in primo piano sono in continuo movimento, fu infatti sviluppata per le auto in movimento sull'autostrada. Il problema è che se un oggetto si ferma un attimo questo inizia a far parte dello sfondo e quindi lo perdo. Ma se gli oggetti si muovono sempre va bene.

Adaptive and Gaussian Background Il metodo considera una gaussiana per ogni pixel, ogni pixel quindi ha una media ed una matrice di covarianza. In questo modo quando arriva un nuovo frame ogni pixel viene confrontato con media e covarianza dei pixel precedenti, se il nuovo pixel sta sotto la curva della gaussiana allora fa parte del background, mentre se è abbastanza diverso dalla gaussiana allora possiamo dire che fa parte di un oggetto in primo piano. Il formulare il pixel con una distribuzione di probabilità mi permette di tenere in considerazione che i colori possono lievemente cambiare e non sono esattamente fissi su di un preciso valore, e nel caso reale è così, e mi permette anche di seguire i cambiamenti naturali e lenti come il cambio di illuminazione tra giorno e notte adattando la formula e solo di fronte a cambiamenti repentini e insoliti, un valore fuori dalla campana, di segnalare che quel pixel fa parte di un oggetto in primo piano e differente dalla mia idea di background. Quindi quando un oggetto in movimento entra ed il suo colore è in contrasto lo rilevo come oggetto in primo piano e se continua a muoversi il suo colore modificherà di poco le gaussiane dei pixel che incontrerà e ciò lascerà il mio modello dello sfondo con dei buoni valori. È un metodo molto semplice in fin dei conti ma è molto funzionale.

Mixture of gaussian - MOG E' il metodo più famoso per modellare lo sfondo ed è accessibile con semplicità da openCV. L'idea è quella di rappresentare uno sfondo con più gaussiane in cui ognuna rappresenta un possibile range di colori dello sfondo. Avrà quindi K gaussiane e la probabilità di un pixel di appartenere ad uno dei suoi possibili sfondi è data da una sommatoria

da 1 a K delle sue gaussiane calcolate nel punto X, che sarebbe il nuovo valore del pixel che sto considerando, pesata da un fattore w che indica quanto faccio affidamento su di una particolare gaussiana. Se non ho particolari informazioni tutti i modelli dello sfondo sono equiprobabili ed allora i pesi w saranno tutti uguali. Se il mio sfondo è molto statico i diversi modelli coincideranno, ma in ambienti dinamici come il mare avrò diverse rappresentazioni. Se qualcosa passa di fronte alla mia scena questo creerà a sua volta una gaussiana ma con una probabilità molto bassa, capirà quindi che si tratta di un oggetto in primo piano, ma nel caso questo si fermi e diventi parte del mio sfondo la sua staticità nel tempo in una posizione farà crescere la probabilità e quindi il peso della gaussiana che lo rappresenta facendo diventare quel colore parte dello sfondo. Questo ragionamento mi va bene perché se qualcosa fa una apparizione breve la voglio considerare un oggetto in primo piano mentre se si fissa nella mia scena voglio aggiornare il mio modello così che tenga conto anche di questo nuovo oggetto come parte dello sfondo. I passi principali che compongono l'algoritmo sono tre:

1. **Comparison** Ad ogni istante di tempo t ogni pixel viene confrontato con i K modelli ed associato al modello a lui più “vicino”. Un punto viene considerato appartenere ad un modello se è lontano dal valore medio della gaussiana meno di 2.5 volte la deviazione standard. Deciso come associare i punti possiamo adesso stabilire che un punto è considerato in primo piano se appartiene alla gaussiana meno probabile (con il peso w minore) oppure se non appartiene a nessuno dei modelli che ho.
2. **Update**
 - (a) Il peso delle gaussiane che hanno ottenuto una corrispondenza con il valore del pixel osservato vengono incrementate
 - (b) Il peso delle altre gaussiane che non hanno ottenuto un match viene decrementato
3. **New Models** In caso di non match invece la gaussiana meno probabile viene eliminata e se ne crea una nuova centrata nel valore osservato e con un peso inizializzato ad un valore basso w0.

9.11 Tracking

Esistono molti approcci per fare tracking diversi a seconda se il tracking riguarda uno o molti oggetti. Solitamente la pipeline prevede la detection, l'associazione e una fase di prediction per capire più o meno precisamente dove ci aspettiamo di trovare il nostro target nel frame successivo. La predizione può essere:

- “cieca”, nel senso che non avendo grossi indizi facciamo assunzioni semplici come l'aspettarci di trovare il target in un certo intorno della posizione precedente.
- Può sfruttare un semplice modello che descrive il movimento.

- Nel caso in cui conosciamo precisamente il tipo di movimento che ci aspettiamo è possibile anche costruire modelli complessi basati sulla memoria di ciò che abbiamo osservato.

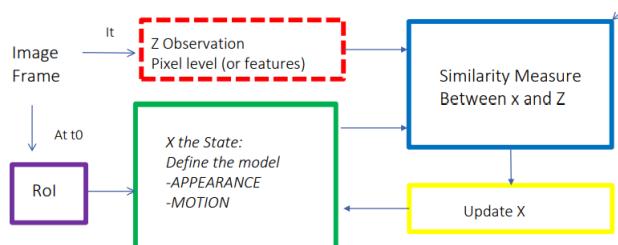
9.12 kalman Tracking

Il kalman tracking è un algoritmo di tracking basilare. Per funzionare bene l'algoritmo ha bisogno di soddisfare tre assunzioni:

1. L'oggetto si muove verso una direzione specifica senza cambiamenti random (es macchine in autostrada)
2. Non ci sono grosse sovrapposizioni tra gli oggetti
3. Non abbiamo informazioni e dobbiamo supporre il moto uniforme

Schema Generale

1. In una prima fase t0 vengono proposte delle ROI (Regions of Interest)
2. A partire dalle ROI viene creato un modello X $X_i = [x_i, y_i, z_i, V_i, V_j, A_i]$ dove A_i definisce l'**apparenza** (solitamente un istogramma che descrive i colori), x,y,z la posizione del pixel e V la sua **velocità**.
3. Parallelamente un modello Z estrae le features del frame
4. Viene predetto lo stato X_i rispetto a tutte le osservazioni precedenti $P(X_i|z_0, z_1, \dots, z_{i-1})$
5. avendo anche l'osservazione Z_i possiamo correggere il modello X_i



Laboratorio

10 Data Manipulation

Pytorch è una libreria che permette di lavorare sui tensori. E' stata sviluppata da facebook ma successivamente si sono aggiunte moltissime aziende, diventando così uno degli strumenti più utilizzati per l'IA.

Tensor un tensore è un array n-dimensionale con un numero arbitrario di assi, in poche parole è la generalizzazione dei vettori (tensore di primo grado) e delle matrici (tensore di secondo grado). In generale, possiamo pensare a un tensore sia la composizione di:

1. **dati**
2. **metadati** la dimensione, la grandezza di ogni elemento, il tipo di elemento
 - Shape (ES D,H,W)
 - Strides
 - Offset
3. **device** in cui è situato il tensore (*solo nel caso si utilizzi pytorch*)

NB: questi elementi non descrivono il tensore, ma sono il tensore (se si apre un tensore su pytorch si vedranno tutti i campi descritti)

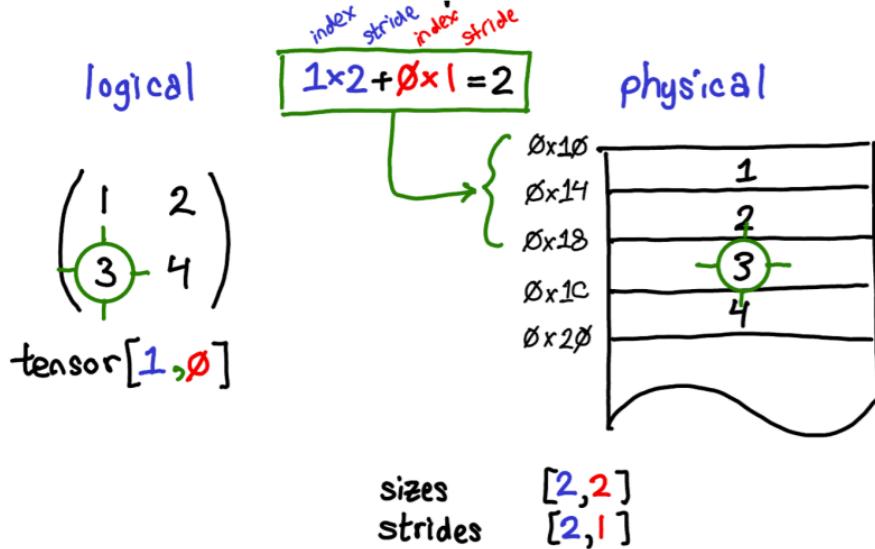
10.1 Basic Tensor properties

```
1 >>> import torch
2
3 >>> x = torch.arange(12)
4 >>> x
5 tensor([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
6
7 >>> type(x)
8 <class 'torch.Tensor'>
9
10 >>> x.shape
11 torch.Size([12])
12
13 #visualizzare il numero di valori
14 >>> x.numel()
15 12
16
17 #riarrangiare il numero di righe e colonne del tensore mantenendo
18 #lo stesso valore degli elementi
19 >>> x.reshape(4, 3)
20 tensor([[ 0,  1,  2],
21         [ 3,  4,  5],
22         [ 6,  7,  8],
23         [ 9, 10, 11]])
```

10.2 How to create a tensor

```
1 >>> import torch
2
3 #creare un tensore di 0
4 >>> torch.zeros((2, 3, 4))
5 tensor([[[0., 0., 0., 0.],
6         [0., 0., 0., 0.],
7         [0., 0., 0., 0.]], ,
8
9         [[0., 0., 0., 0.],
10        [0., 0., 0., 0.],
11        [0., 0., 0., 0.]]])
12
13 #creare un tensore di 1
14 >>> torch.ones((2, 3, 4))
15 tensor([[[1., 1., 1., 1.],
16         [1., 1., 1., 1.],
17         [1., 1., 1., 1.]], ,
18
19         [[1., 1., 1., 1.],
20         [1., 1., 1., 1.],
21         [1., 1., 1., 1.]]])
22
23 #creare un tensore contenente numeri random (rand campiona
24 #i valori da una gaussiana, esistono altre funzioni per
25 #campionare i numeri #da altre distribuzioni come poisson,
26 #bernoulli, binomiale, ecc.)
27 >>> torch.rand(3, 4)
28 tensor([[0.9314, 0.1139, 0.5396, 0.8389],
29         [0.6939, 0.0653, 0.5673, 0.8853],
30         [0.5372, 0.4634, 0.1357, 0.1339]])
31
32
33 #creare un tensore specificando i valori esatti
34 >>> torch.tensor([[2, 1, 4, 3], [1, 2, 3, 4]])
35 tensor([[2, 1, 4, 3],
36         [1, 2, 3, 4]])
37 #NB: utile quando si hanno pochi elementi ma da non fare
38 #mai quando se ne hanno molti poiche' i valori non sono
39 #allocati consecutivamente e le operazioni su questo
40 #tensore saranno lentissime
```

10.3 Mapping from logical to physical representation



Strides lo stride è "il numero di salti" che si devono fare per passare da un elemento al successivo, in questo esempio le colonne hanno stride 2 e le righe stride 1. Se ad esempio eseguiamo il comando:

```
1 >>> x[1, 0]
```

il calcolatore sa che il valore che abbiamo richiesto è archiviato in:

$$1 * 2 + 0 * 1 = 2$$

cioè (contando da 0) nella terza cella di memoria.

10.4 Operations

```
1 >>> x = torch.tensor([1.0, 2, 4, 8])
2 >>> y = torch.tensor([2, 2, 2, 2])
3 >>> x + y
4 >>> x - y
5 >>> x * y
6 >>> x / y
```

10.5 Reduction

`sum()` somma tutti gli elementi di un tensore

```

1 >>> import torch
2
3 >>> x = torch.rand(3, 4)
4 >>> x
5 tensor([[0.0368, 0.6315, 0.0140, 0.1983],
6         [0.6407, 0.2772, 0.1112, 0.6068],
7         [0.0171, 0.1459, 0.7999, 0.4062]])
8 >>> x.sum()
9 tensor(3.8858)
10
11 #possiamo anche specificare l'asse su cui far collassare il tensore
12 >>> x.sum(axis=0)
13 tensor([0.6946, 1.0546, 0.9251, 1.2114])
14 >>> x.sum(axis=1)
15 tensor([0.8807, 1.6358, 1.3692])

```

l'asse che si specifica è l'asse che scompare es: dato x con shape (3, 4), se collasso sull'asse 0 x risultante avrà shape (4); se collasso sull'asse 1 x risultante avrà shape (3).

```

1 >>> x = torch.zeros((125, 25, 200, 300))
2 >>> x.shape
3 torch.Size([125, 25, 200, 300])
4 >>> x.sum(axis = 0).shape
5 torch.Size([25, 200, 300])
6 >>> x.sum(axis = 2).shape
7 torch.Size([125, 25, 300])

```

Non-reduction se si vuole collassare l'asse di un tensore senza però eliminare quell'asse si può utilizzare la keyword `keepdim=True`

```

1 >>> x = torch.rand(2,2)
2 >>> x.sum(axis=0, keepdim=True).shape
3 torch.Size([1, 2])
4 >>> x.sum(axis=0).shape
5 torch.Size([2])

```

10.6 Dot products

Dati due vettori, il dot product è la somma dei prodotti degli elementi nella stessa posizione (ritorna quindi uno scalare)

```

1 >>> y = torch.ones(4, dtype = torch.float32)
2 >>> x = torch.ones(4, dtype = torch.float32)
3 >>> x, y, torch.dot(x, y)
4 (tensor([1., 1., 1., 1.]), tensor([1., 1., 1., 1.]), tensor(4.))
5
6 #analogo a:
7 >>> torch.sum(x * y)
8 tensor(4.)

```

se i due vettori sono normalizzati, il dot product esprime il coseno dell'angolo tra di essi.

10.7 Tensor Concatenation

I tensori si possono concatenare utilizzando il comando **cat**

```
1 >>> import torch
2 >>> X = torch.arange(12, dtype=torch.float32).reshape((3,4))
3 >>> Y = torch.tensor([[2.0, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]])
4 >>> torch.cat((X, Y), dim=0)
5 tensor([[ 0.,  1.,  2.,  3.],
6         [ 4.,  5.,  6.,  7.],
7         [ 8.,  9., 10., 11.],
8         [ 2.,  1.,  4.,  3.],
9         [ 1.,  2.,  3.,  4.],
10        [ 4.,  3.,  2.,  1.]])
11 >>> torch.cat((X, Y), dim=1)
12 tensor([[ 0.,  1.,  2.,  3.,  2.,  1.,  4.,  3.],
13         [ 4.,  5.,  6.,  7.,  1.,  2.,  3.,  4.],
14         [ 8.,  9., 10., 11.,  4.,  3.,  2.,  1.]])
```

10.8 Broadcasting mechanism

Under certain conditions, even when shapes differ, we can still perform elementwise operations by invoking the **broadcasting mechanism**. This mechanism works in the following way: First, expand one or both arrays by copying elements appropriately so that after this transformation, the two tensors have the same shape. Second, carry out the elementwise operations on the resulting arrays.

```
1 >>> a = torch.arange(3).reshape((3, 1))
2 >>> a
3 tensor([[0],
4         [1],
5         [2]])
6 >>> c = torch.arange(2).reshape((1, 2))
7 >>> c
8 tensor([[0, 1]])
9 >>> a+c
10 tensor([[0, 1],
11          [1, 2],
12          [2, 3]])
```

Broadcasting two arrays together follows these rules:

- If the arrays do not have the same rank, prepend the shape of the lower rank array with 1s until both shapes have the same length.
- The two arrays are said to be compatible in a dimension if they have the same size in the dimension, or if one of the arrays has size 1 in that dimension.
- The arrays can be broadcast together if they are compatible in all dimensions.
- After broadcasting, each array behaves as if it had shape equal to the elementwise maximum of shapes of the two input arrays.

- In any dimension where one array had size 1 and the other array had size greater than 1, the first array behaves as if it were copied along that dimension

10.9 Indexing and Slicing

11 2D Convolution

Quella che nella teoria chiamiamo convoluzione, nella pratica, per problemi implementativi, è una cross-correlazione, l'unica differenza sta nel segno dell'operazione (da - a +).

2D Cross-Correlation on a 2D input

- Shape of the input: (H, W),
- Shape of the kernel (kH, kW)
- Shape of the output: (H-(kH-1), W-(kW-1))
- Shape of the output with stride >1: (H-kH, W-kW)/S + 1
- **NB** Solitamente il Kernel è quadrato ma non è una regola fissa, potrebbe essere anche rettangolare.

In short: place the kernel in the right position, element wise multiplication, then sum and apply bias.

2D Cross-Correlation on a 3D input

- Shape of the input: (iC, H, W)
- Shape of the kernel: (iC, kH, kW)
- **Note 1:** this is still a **2D Convolution** because the kernel still moves over 2 axes.
- **Note 2:** the number of channels of the kernel **must be** the same number of channel of the input

Convolutional Layers in CNNs Nelle CNN, solitamente si applicano più kernel e non uno solo. oC kernel producono quindi oC output.

- Shape of the input: (iC, H, W)
- Shape of the kernel: (oC, iC, kH, kW)
- **Note 1:** bias shape is **oC** it's like having oC different kernels and biases
- Shape of the output: (oC, oH, oW)
- **Note 2:** it's like performing oC different cross-correlations

Plus, we do this for a batch of n images in parallel, so:

- The input shape is (n, iC, H, W)
- Kernel shape is (oC, iC, kH, kW), bias shape is (oC,) : same kernel and biases for all images!
- Output shape is (n, oC, oH, oW), where oH and oW are computed as before

Other Parameters

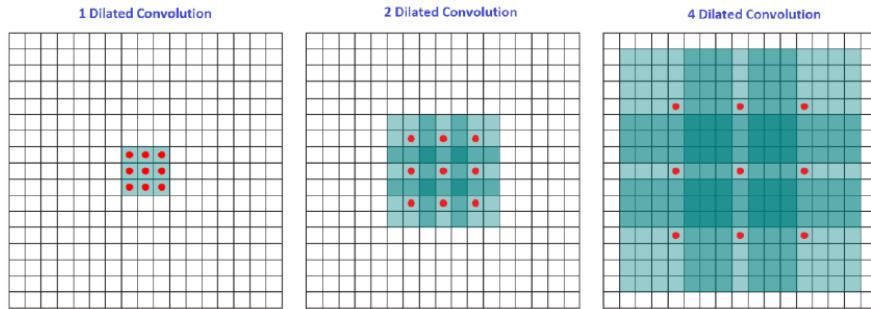
- **Stride** indica di quanti pixel si sposta il kernel, vale 1 se non specificato altrimenti
 - $H \times K$ dimensione immagine
 - $kH \times kW$ dimensione kernel
 - $(H-kH, W-kW)/S + 1$
- **Padding** evita che l'immagine di output sia più piccola dell'immagine iniziale, riempì il bordo di 0 se non specificato altrimenti.

Altre cose interessanti

- Ricordando che le immagini in input hanno iC canali e in output oC canali, un trucco per cambiare il numero di canali di un'immagine è quello di applicare una convoluzione con un kernel di dimensione $(oC, 1, 1)$ (1×1 con oC canali)

11.1 Dilatation

E' un'estensione della normale convoluzione in cui il kernel è dilatato.



Applichiamo la convoluzione solo nei punti rossi. Teoricamente è come avere un grosso kernel pieno di 0 tranne che nei punti rossi, nella pratica no perché sarebbe computazionalmente pesante e inutile fare la convoluzione di un kernel per lo più vuoto. Non è molto popolare ma ci sono casi specifici (come nella segmentation) che si usa molto.

11.2 Grouping

Soltanamente in una convoluzione senza grouping tutti i canali di output sono il risultato di tutti i canali di input correlati con il kernel. Questa all-to-all connectivity una volta era computazionalmente troppo pesante e il grouping è una maniera per alleggerirla, oggi non abbiamo più questo problema ma ci sono

alcuni casi in cui il grouping si usa ancora. Il grouping divide i canali di input in N gruppi di uguali dimensioni e fa lo stesso con i filtri. Ogni gruppo dei canali di input è connesso con il corrispettivo gruppo di filtri. Quindi se diciamo che il numero di gruppi è 1 è come dire che non applichiamo il grouping.

12 Pooling, edge detection, template matching

2D Pooling Nella maggior parte dei casi per ogni canale applichiamo una sliding window e selezioniamo il massimo.

- 3 iperparametri:
 - kernel size (k_H, k_W)
 - stride S
- output (n, i_C, o_H, o_W)
 - $o_H = (i_H - k_H)/S + 1$
 - $o_W = (i_W - k_W)/S + 1$
- come per la convoluzione, possiamo applicare:
 - padding
 - fattore di dilatazione

Pooling vs Convolution Non dire mai al prof che la differenza è che il pooling applica una riduzione, anche nella convoluzione se si applica una stride maggiore di 1 si ha una riduzione; the difference between convolution and the pooling is that 2d pooling **operating independently of the activation**, and we apply a different operation inside the sliding window.

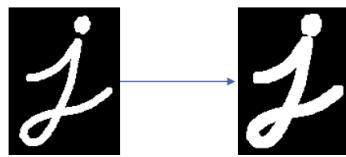
13 Morphology, Hough Transform

13.1 Morphology

Nel caso della morphology invece del kernel abbiamo una matrice binaria chiamata structuring element che viene utilizzata per espandere o "assottigliare" gli shapes dell'immagine.

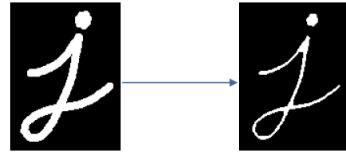
- L'operatore di espansione viene detto **Dilation**

$$result(x, y) = \max(p \text{ for } (p, k) \text{ in } \text{zip}(neighborhood, kernel) \text{ if } k > 0)$$



- L'operatore di diminuzione viene detto **Erosion**

$$result(x, y) = \min(p \text{ for } (p, k) \text{ in } \text{zip}(neighborhood, kernel) \text{ if } k > 0)$$



- **Closing** è una sequenza di dilatazione ed erosione utile per riempire i buchi all'interno degli edges



- **Opening** è una sequenza di erosione e dilatazione utile per eliminare il rumore e migliorare gli edge con difetti morfologici



14 CNN Convolutional Neural Networks

14.1 AlexNet

AlexNet (2012) è stata la prima CNN proposta

- First use of ReLU
- Used Norm layers (not common anymore)
- There are 3 full-connected layer (the layers where all the inputs from one layer are connected to every activation unit of the next layer) before the softmax

Architecture:

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

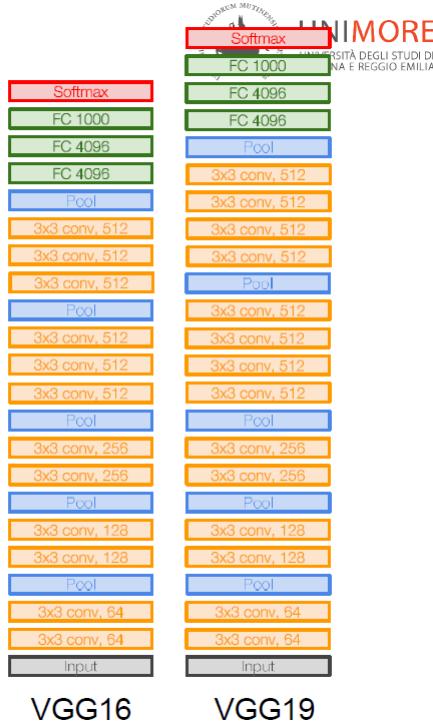
- The number of filters in each layer is very high ($96 \rightarrow 256 \rightarrow 384 \rightarrow 384 \rightarrow 256$)
- In the first layer the filters window is very strange (11x11 at stride 4)

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)



14.2 VGGNet (2014)

- 16 or 19 layers
- The number of filters start from 64 and *duplica* each layer ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512$)
- there are 3 full-connected layer before the softmax



A differenza di AlexNet nel VGGNet non si utilizzano più filtri con dimensioni "strane" come 11×11 ma vengono tutte standardizzate a 3×3 . Perchè usare dimensioni così piccole come 3×3 ?

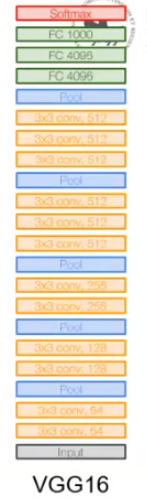
- usiamo solo 9 parametri per filtro
- dopo applichiamo il maxpooling e con una finestra troppo grande perdiamo troppi dati

The number of parameters is 138M and the memory occupied is 96MB for image
ad ogni layer aumenta il numero di parametri e diminuisce la memoria occupata dalle immagini

```

INPUT: [224x224x3]      memory: 224*224*3=150K  params: 0
CONV3-64: [224x224x64]   memory: 224*224*64=3.2M  params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]   memory: 224*224*64=3.2M  params: (3*3*64)*64 = 36,864
POOL2: [112x112x64]     memory: 112*112*64=800K  params: 0
CONV3-128: [112x112x128]  memory: 112*112*128=1.6M  params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]  memory: 112*112*128=1.6M  params: (3*3*128)*128 = 147,456
POOL2: [56x56x128]       memory: 56*56*128=400K  params: 0
CONV3-256: [56x56x256]   memory: 56*56*256=800K  params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]   memory: 56*56*256=800K  params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]   memory: 56*56*256=800K  params: (3*3*256)*256 = 589,824
POOL2: [28x28x256]       memory: 28*28*256=200K  params: 0
CONV3-512: [28x28x512]   memory: 28*28*512=400K  params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]   memory: 28*28*512=400K  params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]   memory: 28*28*512=400K  params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]       memory: 14*14*512=100K  params: 0
CONV3-512: [14x14x512]   memory: 14*14*512=100K  params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]   memory: 14*14*512=100K  params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]   memory: 14*14*512=100K  params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]          memory: 7*7*512=25K  params: 0
FC: [1x1x4096]            memory: 4096  params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]            memory: 4096  params: 4096*4096 = 16,777,216
FC: [1x1x1000]             memory: 1000  params: 4096*1000 = 4,096,000

```



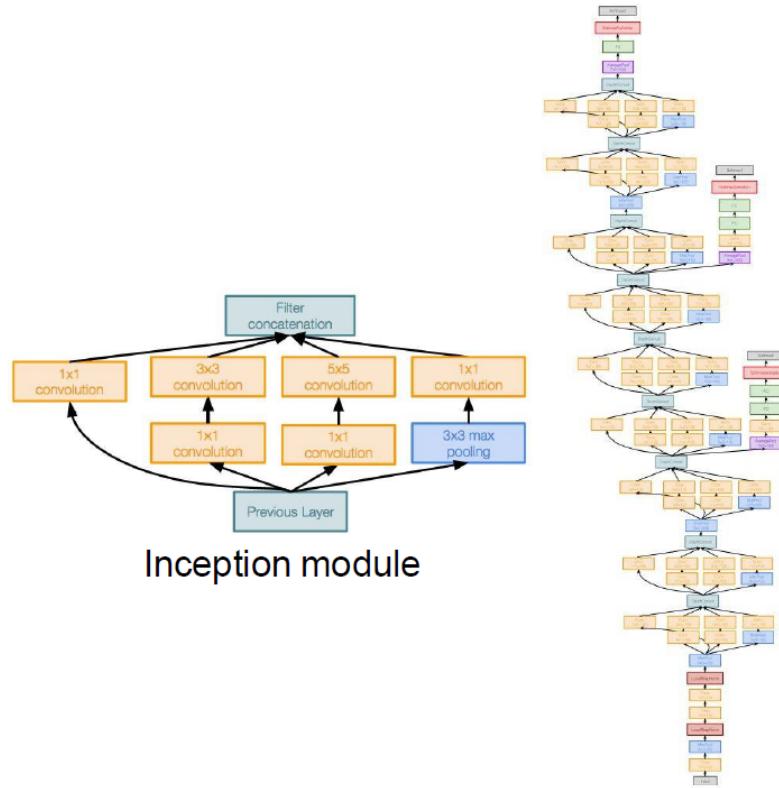
TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)

TOTAL params: 138M parameters

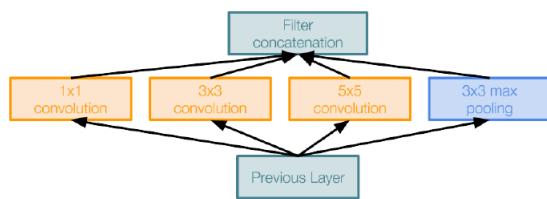
14.3 GoogleNet (2014)

The GoogLeNet architecture consists of 22 layers (27 layers including pooling layers), and part of these layers are a total of 9 inception modules.

- More complex than VGG
- The layers are connected in a particular way: **the inception module**
- It is without fully connected layer at the end of the architecture
- It has only 5M parameters (Much less than VGG)



Inception Module The Inception module is a neural network architecture that leverages feature detection at different scales through convolutions with different filters and reduced the computational cost of training an extensive network through dimensional reduction.



Naive Inception module

Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution ($1 \times 1, 3 \times 3, 5 \times 5$)

- Pooling operation (3×3), with stride 1

Nella versione dell'inception module finale vengono introdotti i filtri convolutivi 1x1 per ridurre il numero di canali (più canali significa più parametri e quindi più peso computazionale, il numero di canali deve quindi rimanere basso). The weakness of the inception module is the Computational complexity:

$$\begin{aligned} & [1 \times 1 \text{ conv}, 128] 28 \times 28 \times 128 \times 1 \times 1 \times 256 \\ & [3 \times 3 \text{ conv}, 192] 28 \times 28 \times 192 \times 3 \times 3 \times 256 \\ & [5 \times 5 \text{ conv}, 96] 28 \times 28 \times 96 \times 5 \times 5 \times 256 \end{aligned}$$

Total: $854M$ ops

Perchè ci sono tre output nell'architettura La ragione per cui ci sono tre output è dovuta che il gradiente nell'output finale finiva per diventare sempre più piccolo durante la back-propagation, questo perchè quando si devono aggiornare i pesi, bisogna applicare la formula dello **stochastic gradient descent**, che a livello teorico significa applicare la **derivata parziale della Loss rispetto ai pesi** ma a livello pratico viene eseguita da pytorch tramite la formula della **chain rule**, cioè il calcolo della derivata parziale viene effettuato come la moltiplicazione delle derivate parziali ai layer precedenti, che dipendono dai pesi, siccome i pesi sono compresi tra 0 e 1, alla fine la formula consiste nella moltiplicazione di un enorme numero pesi minori di uno, quindi più indietro si va nell'architettura più la formula fornisce un numero piccolo che comporta un calcolo errato dei pesi.

Perchè questo non accadeva su VGG perchè l'architettura (il numero di layer) di VGG era abbastanza piccola da non risentire di un gradiente troppo piccolo.

14.4 ResNet (2015)

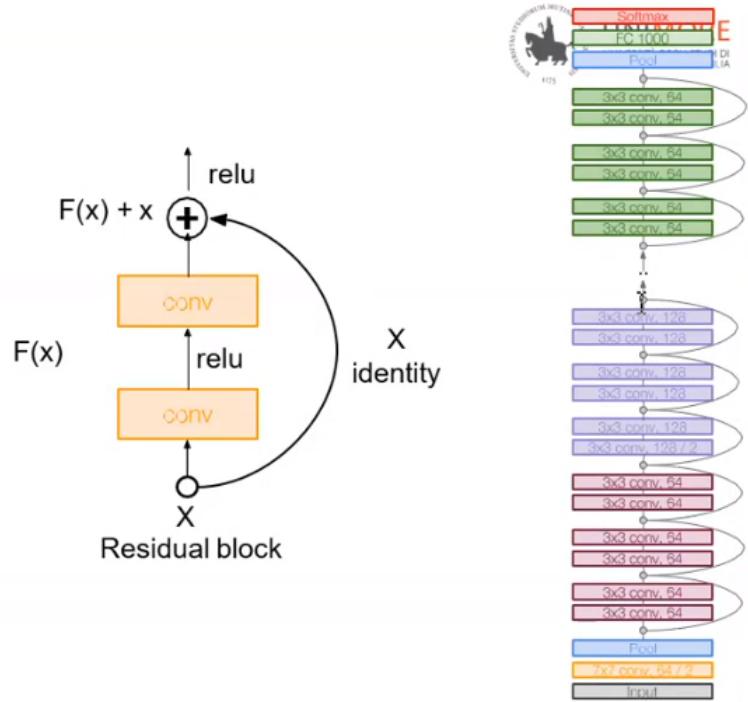
ResNet è il primo esempio di architettura creata dopo la **Revolution of Depth**, infatti mentre prima VGG e GoogleNet avevano rispettivamente 19 e 22 layers, ResNet ne ha 152.

Perchè si è scelto di aumentare il numero di layers Avere più layers significa avere più filtri che identificano più casi specifici, ogni layer migliora e allarga il feature maps dal layer precedente fornendo così una rappresentazione migliore e riducendo l'errore. Ogni volta che aggiungi un layer aggiungi una non-linearietà, creando quindi un modello che può apprendere funzioni più complesse e ritornare un modello più dettagliato.

Caratteristiche dell'Architettura

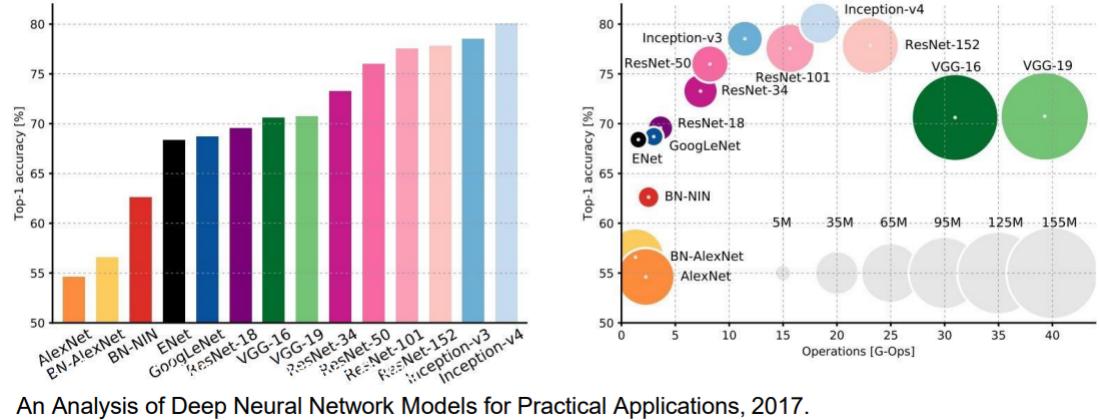
- come nel caso di VGGNet, anche in ResNet il numero di filtri inizia da 64 e duplica ad ogni layer ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512$)

- La principale caratteristica di ResNet è che ad ogni blocco convoluzionale c'è un collegamento tra l'input e l'output



Come hanno risolto il problema del Vanishing Gradient in ResNet

14.5 Complexity Comparison



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Il grafico sopra mostra la quantità di memoria e la difficoltà computazionale di ogni architettura, nel secondo grafico il raggio delle circonferenze delle architetture indica il numero di parametri necessari (e quindi è direttamente collegata alla quantità di memoria occupata), mentre l'asse delle ascisse rappresenta il numero di operazioni necessarie, quindi tutte le architetture che stanno a sinistra sono computazionalmente leggere mentre quelle che stanno a destra sono computazionalmente pesanti. L'asse delle ordinate invece indica il livello di accuracy.

14.6 Other architectures to know

Network in Network (NiN) Consisteva nel mettere un Multi-Layer Perceptron tra i vari layer convolutivi

identity Mappings in Deep Residual Networks Cerca di migliorare il concetto di Residual block della ResNet

Wide Residual Networks Cerca di migliorare ResNet, aumenta il numero di filtri invece di aumentare il numero di layers

ResNext Una fusione tra ResNet e Inception

15 Automatic Differentiation

15.1 Derivatives and Differentiation

If I have a multivariate function (a function with more variables), the gradient of the function $f(x)$ with respect to x is a vector of n partial derivatives:

$$\nabla_x f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right]^T$$

Chain Rule The way to computing the derivatives is the chain rule.

date due funzioni f e g con $y = f(u)u = g(x)$, la derivata di f rispetto ad x è data da:

$$\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$$

15.2 Automatic Differentiation

Al giorno d'oggi tutti i framework di DL hanno almeno una libreria che calcola automaticamente le derivate (Automatic Differentiation)

- Based on our designed model the system builds a computational graph, tracking which data combined through which operations to produce the output.
- Automatic differentiation enables the system to subsequently backpropagate gradients.
- Here, backpropagation simply means to trace through the computational graph, filling in the partial derivatives with respect to each parameter.

Recall that the computing the stochastic gradient descend is compute the partial derivative of the loss with respect all the weights.

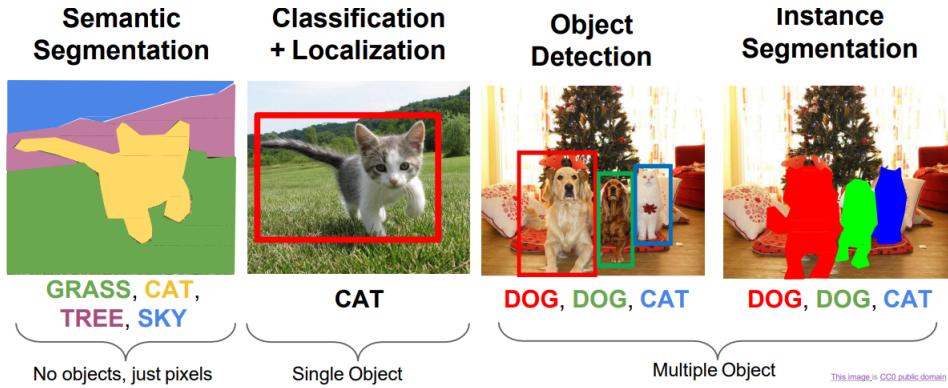
Esempio di implementazione l'obiettivo dell'esercizio è quello di differenziare la funzione $y = 2x^T x$ rispetto ad x

```
1 import torch
2 x = torch.arange(4.0)
3
4 #in pytorch il gradiente e' accessibile tramite la funzione .grad
5
6 x.requires_grad_(True)
7 # Il valore di default (come in questo caso) e' None
8 x.grad
9 # creiamo la funzione dell' esercizio
10 y = 2 * torch.dot(x, x)
11 # il gradiente si calcola tramite la funzione .backward()
12 # NB: y.backward calcola TUTTE le derivate rispetto a TUTTE le
     variabili presenti all'interno di y
```

```
13 y.backward()
14 # per accedere poi alla derivata parziale ddi y rispetto ad x (che
15 # e' quella che interessa a noi)
16 dobbiamo utilizzare x.grad
17 # grad torna in questo caso 4x (la derivata di 2xTx)
18 x.grad
```

16 Detection Architectures

Fino a questo momento abbiamo utilizzato le CNN per fare classificazione d'immagini, ci sono però molti altri campi che possono essere implementati tramite CNN, tra i più importanti abbiamo:



- **Semantic Segmentation** Capire per ogni pixel dell'immagine a che classe semantica appartiene
- **Classification + Localization**
- **Object Detection** Capire quanti oggetti sono presenti nella scena, a quale classe appartengono e dove sono posizionati
- **Instance Segmentation** E' la fusione tra semantic segmentation e l'object detection

Differenza tra semantic segmentation e instance segmentation

1. la semantic prende in considerazione tutti i pixel dell'immagine (ogni pixel dell'immagine avrà una label)
2. se nella semantic abbiamo un'immagine con due cani che si toccano non si riesce a capire dove finisce uno e comincia l'altro, perchè l'algoritmo li rileva entrambi come appartenenti alla classe cani, l'instance segmentation invece è chiamata istance proprio perchè riesce a distinguere le istanze all'interno delle classi.

16.1 Classification + Localization

Classifica gli oggetti presente nell'immagine e li localizza

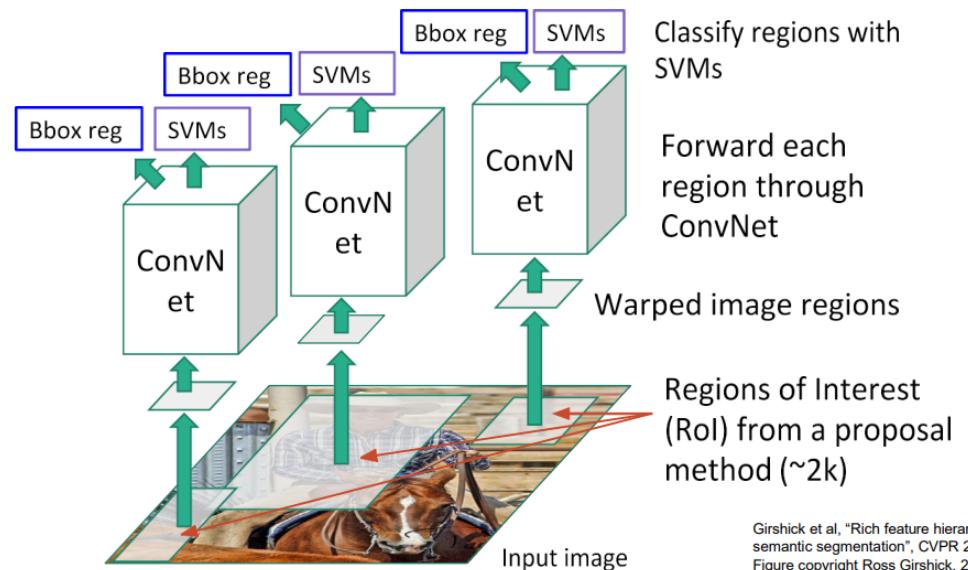
Object Detection as Regression Utilizza il concetto di Multitask Loss, viene applicata una CNN in cui vengono utilizzate due Loss, una **Softmax Loss** per individuare la classe dell'oggetto e una **L2 Loss** per trovare le coordinate dell'oggetto (x, y, w, h). Questo metodo ha un problema poiché per ogni oggetto bisogna trovare 4 parametri (x, y, w, h) e se sono presenti molti oggetti nell'immagine bisogna trovare moltissimi parametri

16.2 Detection as Classification

Sliding Window Si applica una CNN a diversi ritagli dell'immagine, la CNN classifica ogni ritaglio individuando se si tratta di un oggetto o dello sfondo

Problema della Sliding Window Bisogna applicare la CNN ad un numero enorme di posizioni, decidendo anche su che scala e che dimensione della finestra, diventa così costoso a livello computazionale.

R-CNN Poichè non possiamo applicare tutti i bounding boxing nell'immagine, l'idea è quella di trovare dei candidati, ridimensionarli ed applicargli la rete convolutiva. La CNN fornisce due output per ogni bounding boxing, il primo output è ottenuto da una SVM che si occupa di classificare, mentre il secondo output, poichè c'è il problema che con il ridimensionamento delle aree si rovina l'aspect ratio, fornisce informazioni del tipo: bisogna allargare o restringere la finestra, bisogna spostarla verso destra o sinistra perché non è perfettamente allineata con l'oggetto, ecc.

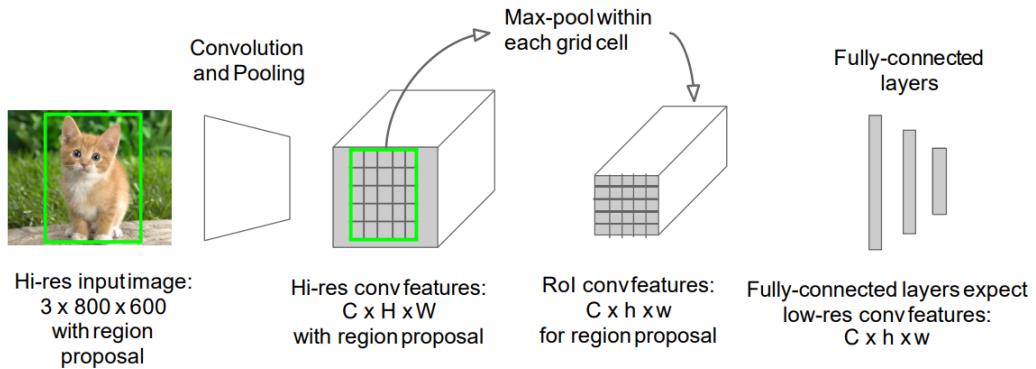


R-CNN Problems

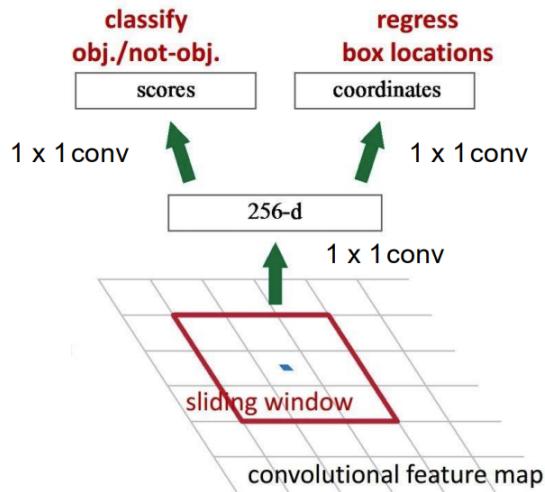
- E' molto lento, per ogni finestra bisogna applicare una CNN
- Bisogna trainare i due classifier SVM e Bbox reg

Fast R-CNN L'idea è quella di, invece di trovare le 2mila finestre e per ognuna ridimensionarla e applicare la CNN, applicare direttamente la CNN su tutta l'immagine come primo step, prendere l'output dell'ultimo layer convoluzivo e successivamente su di esso effettuare la region proposal, trovare le finestre candidate ed effettuare tutti gli step come nella R-CNN classica. Per effettuare il ridimensionamento viene utilizzato un nuovo layer chiamato **"RoI Pooling"**, il suo funzionamento è il seguente: prende varie immagini di varie dimensioni e le converte in un batch di immagini tutte della stessa dimensione senza rovinarle a livello semantico e lasciandole completamente differenziabili (posso applicare la back-propagation).

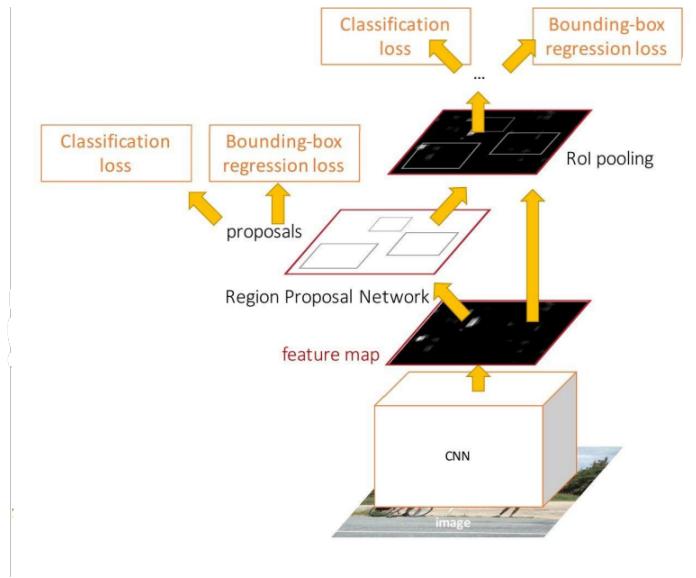
Region of Interest RoI Pooling Il seguente paragrafo spiega come fa il RoI a "standardizzare" i vari input applicando lo stesso ridimensionamento ad ogni bounding box. Dato un bounding box di dimensione $C \times M \times N$, lo divido in H colonne e W righe formando così una sorta di matrice composta da $H \times W$ celle. Poichè la dimensione del bounding box non è fissa, all'interno di ogni cella possono essere presenti uno o più pixel, per questo applico una max pooling all'interno di ogni cella ottenendo così una matrice $H \times W$ di dimensioni fisse in cui è poi possibile applicare layer convolutivo.



Faster R-CNN La region proposal è effettuata da un layer chiamato **Region Proposal Network RPN**. Si prende la convolutional feature map in uscita dalla CNN, si applica una convoluzione 1x1 e dopodichè altre due convoluzioni 1x1 in parallelo, la prima dice se c'è un oggetto in quella regione e la seconda dà le coordinate



L'architettura finale compresa di tutti i layers è la seguente:



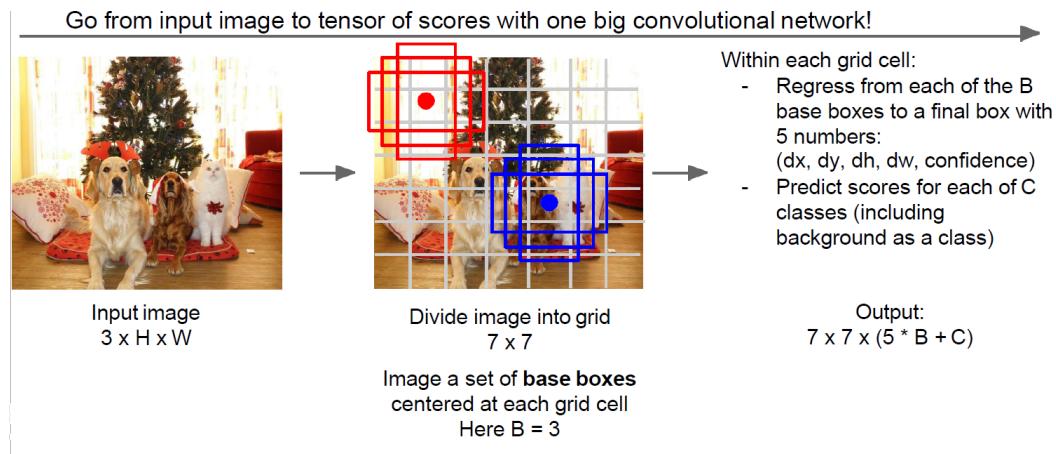
Mask R-CNN Evoluzione della faster oltre alla instance segmentation riesce ad effettuare anche la detection, anche lui ha un layer di RoI chiamato Roi Align, la differenza sta nell'applicare una maschera finale che applica la detection della classe.

- **CNN** la cnn fa una region proposal
- RoI Align

- per ogni region proposal viene creata una feature maps

RoIAlign Come il RoI pooling ma più preciso. Quello che fa è prendere il Roi, dividerlo in blocchi e applicare la bilinear interpolation. La differenza rispetto la Roi Pooling è quindi solo che invece di applicare una max pooling applichiamo una bilinear Interpolation, l'interpolation permette di non perdere la posizione delle features e quindi di non dover approssimare le rispettive coordinate.

Detection without Proposals: YOLO / SDD Meno preciso ma molto veloce, L'idea alla base di queste due architetture è molto semplice, si evita lo step di region proposal ma si applica una CNN e per ogni pixel dell'output si traina la rete che predice la classe e la posizione dell'oggetto

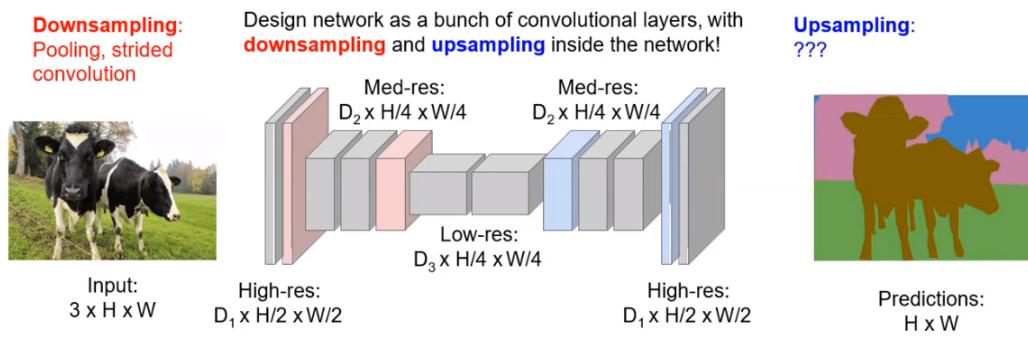


17 Semantic Segmentation

Per effettuare una semantic segmentation, una possibile soluzione sarebbe quella di implementare una Sliding Window, ma non è la soluzione migliore perché è molto pesante. Un'altra soluzione è quella di applicare una CNN su tutti i pixel dell'immagine, il problema di utilizzare una CNN è che però ritorna un'immagine di output più piccola di quella di input.

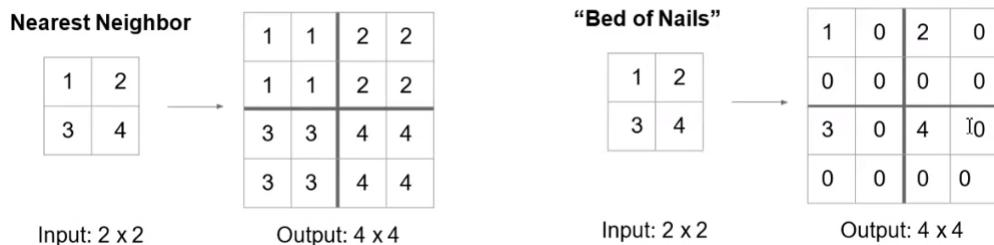
17.1 Fully Convolutional

Per risolvere il problema del rimpicciolimento dell'immagine, si implementa la rete come due CNN: una chiamata **DownSampling** (che contiene i layers di max pooling) e una chiamata **UpSampling** che si occupa di ingrandire l'immagine e ritornare l'output con la stessa dimensione dell'input.

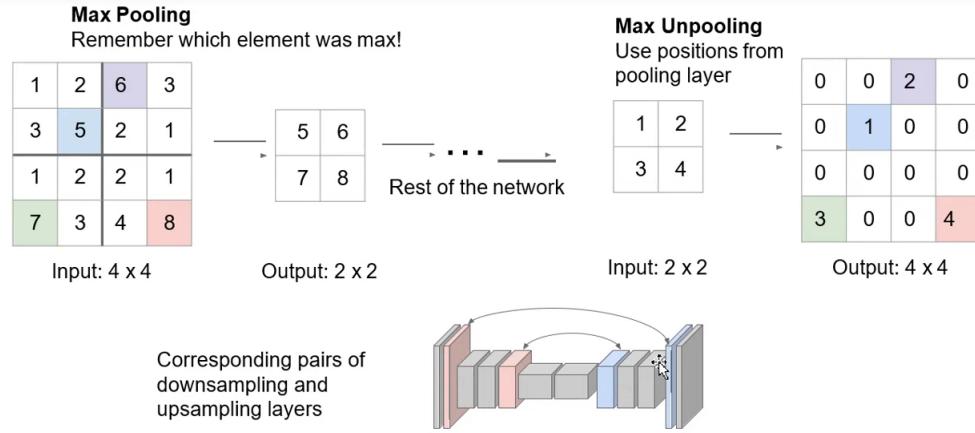


UpSampling Non esiste un solo metodo per implementare l'Upsampling, sono state proposte varie implementazioni tra cui:

- **Unpooling** Ne esistono due tipologie, il Nearest Neighbor e il Bed of Nails, entrambi espandono il blocco originale di pooling, nel primo caso ripetendo i valori e nel secondo inserendo degli zeri. In entrambi i casi il risultato non è molto bello da vedere per via dell'effetto "pixeloso".

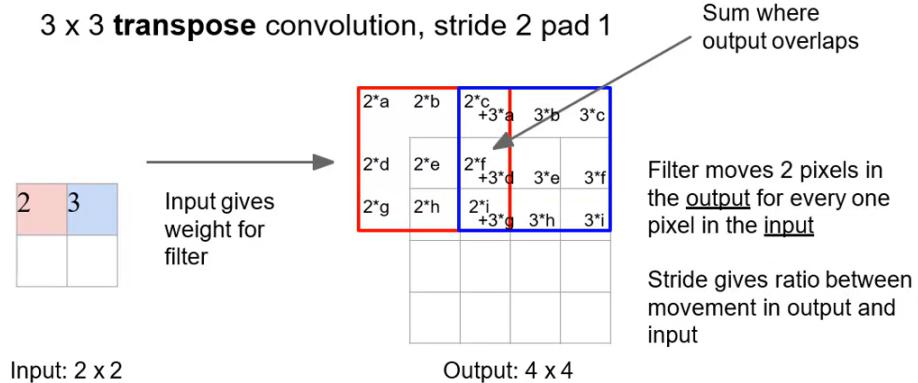


- **Max Unpooling** Durante il max pooling (nel downsampling), viene salvata la posizione del valore maggiore e nel upsampling si espande il blocco di pooling ponendo 0 nelle posizioni create e il valore nell'esatta posizione in cui si trovava prima del pooling



17.1.1 Upsampling - Transpose Convolution

E' l'implementazione utilizzata ai giorni nostri. L'idea di base è quella di effettuare una convoluzione al contrario



Ogni pixel di input viene moltiplicato per i valori del kernel (Nell'esempio sopra 3x3) e i valori di output vengono posizionati nella corrispettiva posizione nell'immagine di output, nel caso in cui due kernel si sovrappongono i valori vengono sommati.

Perchè è chiamata Transpose Sappiamo che è possibile implementare la convoluzione tramite una moltiplicazione matriciale tra l'immagine e il ker-

nel. Nel nostro caso il motivo per cui utilizziamo il nome transpose convolution è perchè per implementarlo dobbiamo effettuare una moltiplicazione tra l'immagine e la trasposta del kernel

Seg-Net Esiste ed utilizza la **Dilated Convolution**

U-Net Nato nell'ambito della segmentazione medica, è chiamato U-Net per via della forma della sua architettura.

DeepLab molto famoso, anche lui utilizza la dilated convolution e lo fa tramite un architettura piramidale

18 Geometrical Transformations

Dal punto di vista pratico le trasformazioni geometriche alla fine non sono altro che funzioni che cambiano la posizione dei pixel tramite la moltiplicazione dell'immagine con delle matrici di trasformazione

19 Deep Neural Network architectures for Video Understanding

Carrallata di dataset video

- **UCF 101** Classico, uno dei primi, non molto grande.
- **Sports-1M** Contiene 1 milione di video riguardanti lo sport, tutti i video provengono da youtube, bella paraculata perché i video di youtube sono già categorizzati e questo ci permette di non dover annotare tutto a mano
- **Youtube8M** Molto popolare in questo momento, come si capisce dal nome ha 8 milioni di video anch'essi presi da youtube ma questa volta appartenenti a tutte le categorie. Del pro di prendere i video da youtube ne abbiamo già parlato, il contro è che la maggior parte dei video riguardano alcune categorie come games, sport e arte.

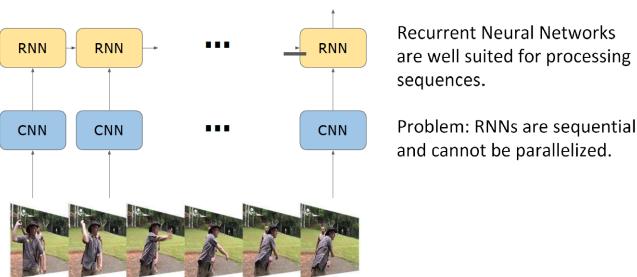
Problemi di lavorare con i video

- Il tempo di training è enormemente più grande, parliamo di giorni e giorni, che da un punto di vista scientifico significa che possiamo fare meno esperimenti e questo è il motivo per cui la CV applicata ai video è meno sviluppata di quella applicata alle immagini.

19.1 Come fare Video Classification

Combination method Abbiamo una sequenza di frames e li passiamo in parallelo nelle cnn per poi combinarle alla fine. Contro: il primo è che dividendo il video in frames la cnn non può estrarre la **motion feature**, secondo problema è che **permutation invariant**, se cambiamo l'ordine dei frames di input il risultato non cambia.

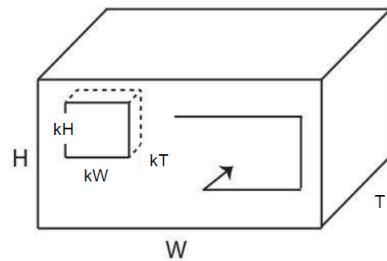
2D CNN + RNN Ogni frame passa da una CNN come nel combination method ma si aggiunge un livello di Recurrent Neural Network (che ovviamente non è permutational invariant). Contro: le RNN non possono essere parallelizzate (è abbastanza impossibile a livello di tempo trainare un video dataset con una sola GPU).



19.2 Operatore per il video classification

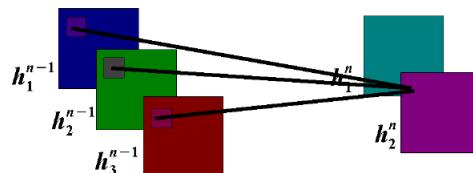
3D CNN Prima idea: 3D CNN, utilizzo di un kernel 3D che a differenza del 2D si muove in 3 dimensioni, altra differenza con la 2D è che è il kernel è tridimensionale. (NB: l'unica e sola ragione per cui è chiamato 3D è perchè si muove su 3 assi non per la dimensione del suo kernel o del suo input).

- We can add an extra dimension to standard CNNs:
 - An image is a (iC, H, W) tensor: (oC, iC, kH, kW) kernels
 - A video is a (iC, T, H, W) tensor: (oC, iC, kT, kH, kW) kernels



Visto che il Kernel è 3D con un un asse che possiamo considerare temporale, grazie ad esso possiamo estrarre la motion feature.

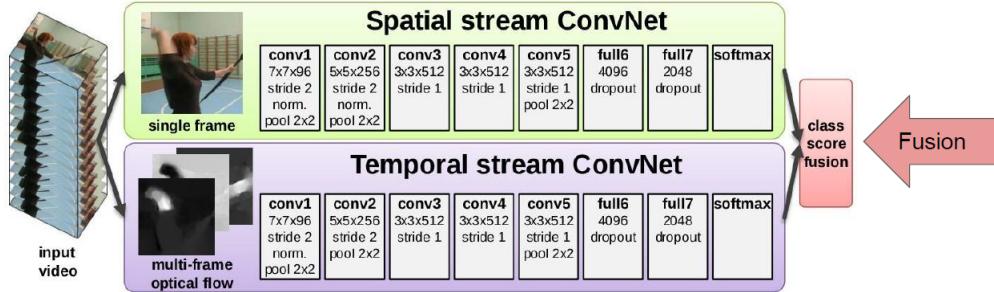
- The input shape is (iC, T, H, W)
- Kernel shape is (oC, iC, kT, kH, kW) , bias shape is $(oC,)$
 - it's like having oC different kernels and biases
 - each kernel moves over three axes: T, H, W
- Output shape is (oC, oT, oH, oW) → it's like performing oC different cross-correlations



- Plus, we do this for a batch of n images in parallel, so:
 - The input shape is (n, iC, T, H, W)
 - Kernel shape is (oC, iC, kT, kH, kW) , bias shape is $(oC,)$ → same kernel and biases for all videos! ☺
 - Output shape is (n, oC, oT, oH, oW) , where oT, oH and oW are computed as in 2D Convolution

Two Stream 2D CNN Secondo approccio totalmente differente rispetto alla convoluzione 3D. L'idea alla base è quella di utilizzare due 2D CNN che lavorano in parallelo, la prima si occupa di gestire la parte statica del video mentre la

seconda la parte dinamica (lavora sull'optical flow), dopodichè l'ultimo layer si occupa di fondere le due cnn e back-propagare il gradiente.

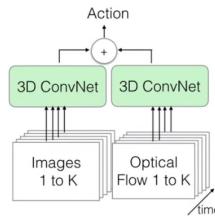


Comparazione tra 3D convolution e 2Stream Il 3D ha bisogno di un numero molto più grande di parametri e prende 16 frames di input alla volta (più o meno un video di 0.6 secondi alla volta). La 3D convolution ha bisogno di un dataset molto grande su cui essere allenata altrimenti la sua accuracy fa schifo (grosso difetto)

Altre Varianti Nel corso dei secoli sono state messe a punto altre varianti che mixano le reti 2Stream e 3D

Inflated 3D CNN Inflated si traduce con "gonfiato", l'idea alla base è quella di invece di avere un layer 3D con un kernel 3D è quello di prendere una cnn 2D con un kernel 2D e ripetere il kernel creando più copie di esso in maniera tale da avere un kernel 3D. Riassumendo l'idea è quella di trainare una 3D CNN inizializzando il kernel con un kernel 2D pre-trainato. L'algoritmo è molto utilizzato ai giorni nostri perchè mette una pezza al problema che la 3D CNN abbia bisogno di un sacco di dati di training.

Two Stream I3D Potentissimo, unisce l'Inflated 3D CNN con la TwoStream. Senza il pretraining ha un accuracy ottima ma con il pretraining sul dataset kinetics l'accuracy arriva fino al 98%.



20 Face Recognition

Detection vs Recognition La detection disegna una bounding box intorno all'oggetto senza sapere nient'altro, la recognition è più avanzata poichè data la bounding box riconosce l'oggetto (in questo caso la persona)

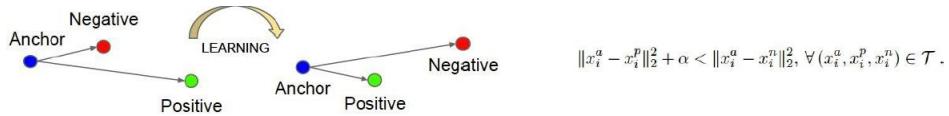
20.1 DeepFace

Training Dobbiamo allenare un modello che se per una faccia ci torna un vettore, allora date delle immagini contenenti la stessa faccia ci torna vettori simili.

Deep ID2

FaceNet Usa la triplet loss function

Triplet Loss Function Prende tre items, il primo è chiamato Anchor (possiamo pensarla come un item preso a caso dal dataset), il secondo è il Negative, sempre preso dal trainingset è importante che appartenga a una classe differente dell'Anchor e infine il Positive che è un item appartenente alla stessa classe dell'Anchor. Quello che vogliamo ottenere con il learning è che la distanza tra l'anchor e il negative sia maggiore della distanza tra l'anchor e il positive più un dato margine (anchor e positive vicini e anchor e negative lontani).



- **Triplet Loss Function**

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

21 Appunti

Cose che possono tornare utili

21.1 Lab

- Passare da pytorch a numpy e viceversa durante una convoluzione si può fare ma non è la migliore delle soluzioni visto che si svuota la memoria contenente il calcolo del gradiente
- la differenza tra una convoluzione e una pooling è che la pooling è indipendente dall'attivazione precedente (sarebbe da chiedere al prof cosa intende per indipendente dall'attivazione precedente) e che invece di eseguire una correlazione esegue un'operazione di massimo/minimo
- l'unica differenza tra una convoluzione 2D e una convoluzione 3D è che nel primo caso il kerner si muove lungo gli assi (x,y) e nel secondo lungo gli assi (x,y,z)

21.2 Teoria

- **Shift Invariance** proprietà dei sistemi che fa sì che l'output di un pixel dipendi solamente dal valore iniziale (input) di quel pixel stesso ed eventualmente dai valori nel suo intorno, non dalla posizione. Nei sistemi lineari in cui vale la proprietà di shift-invariance si può applicare la convoluzione.
- **Receptive field** la porzione dell'immagine in cui viene applicato il kernel
- **Feature maps** anche detti feature vectors, sono l'output della convoluzione, cioè i risultati dell'applicazione di ognuno dei diversi filtri sull'immagine.
- **Segmentation** Operation that divide the image in regions based on perceptive characteristics. Raggruppamento di pixel simili.
- **Pattern Recognition** insieme di tecniche utilizzate al fine di trovare un pattern in un set di dati, non si limita solo alla computer vision e più in generale è sinonimo di machine learning.
- The main difference between clustering and segmentation is that the former usually ignores pixel layout and neighborhoods, while the latter relies heavily on spatial cues and constraints.
- Il motion field è la proiezione del movimento reale sul piano dell'immagine.
- L'optical flow è la stima del moto apparente di un pattern con una data luminosità.
- **Detection vs Recognition** La detection disegna una bounding box intorno all'oggetto senza sapere nient'altro, la recognition è più avanzata poiché data la bounding box riconosce l'oggetto