Syllabus Computer Vision

Eros Bignardi

November 4, 2021

Contents

1 Introduction

Artificial Intelligence, presentation Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better. The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

A journey in the 10 years of research This past decade has continued to see a deepening interplay between the vision and graphics fields. In particular, many of the topics introduced under the rubric of image-based rendering, such as image stitching, light-field capture and rendering, and high dynamic range (HDR) image capture through exposure bracketing, were re-christened as computational photography to acknowledge the increased use of such techniques in everyday digital photography. A second notable trend during this past decade has been the emergence of feature-based techniques (combined with learning) for object recognition. Feature-based techniques also dominate other recognition tasks, such as scene recognition (Zhang, Marszalek, Lazebnik et al. 2007) and panorama and location recognition. Another significant trend from this past decade has been the development of more efficient algorithms for complex global optimization problems. The final trend, which now dominates a lot of the visual recognition research in our community, is the application of sophisticated machine learning techniques to computer vision problems (Freeman, Perona, and Schölkopf 2008). This trend coincides with the increased availability of immense quantities of partially labelled data on the Internet, which makes it more feasible to learn object categories without the use of careful human supervision.

Ai commercial potential Optical character recognition (OCR), Machine inspection, Medical imaging, Automotive safety, Surveillance, Fingerprint recognition and biometrics.

Definitions of Artificial Intelligence Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents": any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving".

- analysing their environment
- display intelligent behaviour
- taking actions
- with some degree of autonomy to achieve specific goals

Computer Vision and Visual Intelligence Definitions Computer Vision since the 80's is the science providing computers with VISION capability and is becoming the science will provide computers with VISUAL INTELLIGENCE, use intelligence to understand the world from visual stimuli. Visual intelligence it is a cognitive capability to think and reasoning through mental images.

Computer Vision Definition 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. The scientific discipline studying how computers can perceive and understand the world through visual data (and provide a visual intelligence). Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do. Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that make sense to thought processes and can elicit appropriate action.

Cognitive System Definition Technical systems capable of independently solving and developing strategies for human tasks. To accomplish this, these systems are equipped with cognitive capabilities for context comprehension, interaction, adaptation and learning. Cognitive systems can utilize artificial intelligence (AI) methods such as machine learning, neural networks and deep learning.

Computer Vision vs Computer Graphics Computer vision is focused on extracting information from the input such as images/videos to have a proper understanding to predict the visual input like human brain where as Computer Graphics is mainly focused on processing the input images/videos to enhance them or preparing them to do other tasks. Computer Vision is a superset of Computer Graphics and Computer Graphics is a subset of Computer Vision. Examples of some Computer Vision applications are- Object detection, Face detection, Hand writing recognition etc and some examples of Computer Graphics are- Correcting illumination, Changing tones, recycling images etc.

---

2 Computer Vision Applications

A brief history of Computer Vision

- 1966 Marvin Minsky, Seymour Papert a summer project, Understanding visual data with «computer».. an old idea, even before computer
- 1973 Leon Harmon The Recognition Of Faces Bell Labs
- 1975 Gala Contemplating the Mediterranean Sea (Salvator Dali)
- 1980 «Vision» David Marr
- 1990-2000 Pattern recognition on visual data
- 1998 SIRIO Bologna and a first idea of an Italian project with surveillance

Vision as 4 R definitions

- Recognition: understanding the world from images
- Reconstruction: create a vision (active process) of the world in 3D from 2D
- Registration: find correspondence in space in time predicting the future
- Reorganization: create unsupervised deductions

Example of processing Image processing, Signal processing, Video Processing, Image enhancement , Imaging, Computational Photography, Detection, Matching, Feature extraction, Measuring..

Example of reconstruction Geometric 3D reconstruction from stereo, video and single images, Measuring 3D distances by camera calibration, integrating RGB and depth images, Computing 3D anchor points for augmented reality; Reconstruction of 3D and motion (example GOOGLE EARTH PROJECT or UNIMORE VAEX Project 2014)

Example of spatial understanding Vision on image plane, segmentation, detection and localization, Document analysis, working on color, shape and deep learned visual descriptors (example Keypoint matching in 2D images, for transparent object grasping for robot vision)

Example of temporal understanding Tracking in time, Time series and trajectory analysis, anomaly detection, event detection, Video comprehension, Temporal video segmentation and alignment, tracking moving objects in the time by cameras with no, constrained or unconstrained motion, for surveillance and autonomous driving (example Tracking in the crowd with Structural SVMs UNIMORE 2016; Tracking multiple occluded poses of multiple people; LAMV Learning to Align and Match Video)

Example of visual intelligence Predicting of Spatio-temporal visual events, Imagining and Predictive behavior, Mental Images Generative networks and domain transfer knowledge (example Automatic video captioning: an open research area. NEURALSTORY 2017 Aimagelab, UNIMORE; Mental Images and Domain Transfer Learned by the POSEidon Net and GANs)

3 Data Manipulation in Pytorch

Tensor and their representation in Pytorch The tensor is the central data structure in PyTorch, just like in Numpy, are n-dimensional arrays with an arbitrary number of axes. Vectors, for example, are first-order tensors, and matrices are second-order tensors. Tensors are denoted with capital letters of a special font face (e.g., X, Y, and Z) and their indexing mechanism (e.g., x_{ijk} and $[X]_{1,2i1,3}$) is similar to that of matrices. It is an n-dimensional data structure containing some sort of scalar type, e.g., floats, ints, et ceteraz. Metadata

- The shape of the tensor
- The type of the elements it contains
- Stride
- Offset

Basic tensor properties

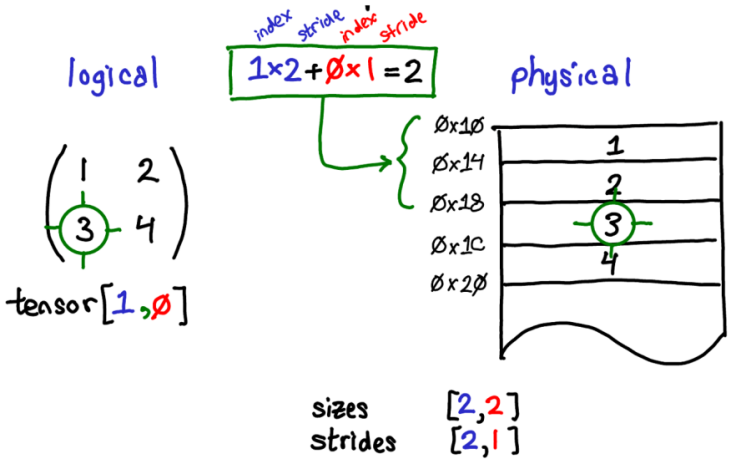
- import torch, to import PyTorch libraries
- torch.arange(10), create a row vector containing first 10 integer starting with 0
- x.shape, to access a tensor's shape
- x.numel(), to know the toal number of elements in a tensor
- x.reshape(3, 4), to change the shape of a tensor without altering the number of elements and their values
- torch.zeros((2, 3, 4)), to create a tensor representing a tensor with all elements set to 0 and a given shape
- torch.ones((2, 3, 4)), create tensor with each element set to 1
- torch.randn(3, 4), randomly sample the values for each element
- torch.tensor([[2, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]]), to specify the exact value for each element in the desired tensor

Strided representation To find out where any element for a tensor lives, I multiply each index with the respective stride for that dimension, and sum them all together.

Tensor operations Elementwise operations (tensor of the same shape), $x+y, x-y, x*y, x/y$

Reduction operators, non-reduction, concatenation

- Reduction: reduces a tensor along all its axes to a scalar, like A.shape or A.sum(), we can specify the axes along which the tensor is reduces (axis=0 for rows and axis=1 for columns)
- Non-reduction: keep the number of axes unchanged, insert "keepdims=True" like sum_A = A.sum(axis=1, keepdims=True)
- Concatenation: stacking them end-to-end to form a larger tensor, just need to provide a list of tensors and tell the system along which axis to concatenate.
  torch.cat((X, Y), dim=0), torch.cat((X, Y), dim=1)



Broadcasting When shapes differ, we can still perform element wise operations by invoking the broadcasting mechanism, 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 element wise operations on the resulting arrays.
a = torch.arange(3).reshape((3, 1))
c = torch.arange(2).reshape((1, 2))
a + c
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.

Indexing and slicing

- Indexing: As in any Python array, the first element has index 0 and ranges are specified to include the first but before the last element. As in standard Python lists, we can access elements according to their relative position to the end of the list by using negative indices.
- Slicing: If we want to select multiple elements, we simply index all of them. For instance, [0:2, :] accesses the first and second rows, where : takes all the elements along axis 1 (column).

Views Strides are also the basis on which we can provide views to the users, tensor[1, :] here's the important thing: when I do this, I don't create a new tensor; instead, I just return a tensor which is a different view on the underlying data. This means that if I, for example, edit the data in that view, it will be reflected in the original tensor.
torch.as_strided(input, size, stride, storage_offset=0), Create a view of an existing torch.Tensor input with specified size, stride and storage offset.
contiguous(), return a copy of the tensor (and of its data) in which the data is copied in a contiguous arrangement. If self tensor is already contiguous, this function returns the self tensor.
.view(*shape), returns a new tensor with the same data as the self tensor but of a different

shape. The returned tensor shares the same data and must have the same number of elements but may have a different size.

Numpy bridge The Torch Tensor and NumPy array will share their underlying memory locations, and changing one will change the other.
a = torch.ones(5)
c = a.numpy()

or We can also transfer from Numpy to PyTorch (again, a zero-memory copy)

import numpy as np
a = np.ones((5,))
c = torch.from_numpy(a)

---

4 ABC images and point operators

Image: a signal and a data matrix The image is 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. The image is a pixel matrix; I image is a 2D matrix of digital pixels thus image processing as application of mathematical function or algorithmic procedure working on images (algebra, geometry, computer science..).

Rgb and palette We use often Images in RGB space as a triplet of three values for the R, G, B dimensions. Numbers in an images can be also pointers in a palette (be careful not to loosen the continuity of the signal. do not use palettes in image processing).

Image a signal function For image processing, image is a bidimensional signal, sampled and quantized. The x point defined in the image plane has an associated function: I(x) value of image in x, The physical meaning of I(x) depends on the sensor.

$$I(x) = f(x,y) : R^2 \rightarrow R$$

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

Digital images and color, resolution An image is a sequence of values, in image processing we consider spatial resolution, the number of pixels (not related with the dimension of the objects), each pixel is a measure of the luminance intensity of the point in the 3D scene, projected in the 2D space, depending on the luminance of the object, the reflectivity, the background luminance, the sensor... Color or multispectral images represent more spectral components N=3 images in color spaces or M images in multispectral dimensions. Image is a set of matrixes one for each dimension of the color space

Multispectral images Multispectral imaging combines two to five spectral imaging bands of relatively large bandwidth into a single optical system. A multispectral system usually provides a combination of different bands, like visible, near infrared, short-mid-long wave infrared,into a single system.

Histogram, normalized histogram 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; it gives important information for image processing, especially for contrast enhancement and segmentation (for color images, 3 histograms, or a combined histogram). It is a global feature vector , but sometime not discriminant enough.
Histogram: $H(j) = \#x : I(x) = j$
Normalized Histogram: is the probability distribution $p(i) = \#\frac{x:I(x)=i}{n} \quad 0 \leq i < L$ where n be the number of occurrences of gray level i.

Cumulative Histogram Probability of having a pixel's value less of j

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

Entropy specifies the uncertainty in the image values. It measures the averaged amount of information required to encode the image values. An infrequent event provides more information than a frequent event. Entropy is a measure of histogram dispersion.

$$Entropy(I) = -\sum_k P(k) log P(k)$$

Image processing Science studying how to automatically process images with a computer to produce other images, modified or enhanced in some visual proprieties. Image processing is a first step to transform images into other images where the information and the process of knowledge extraction requested by the goal could be easier.

Saturated arithmetics is used expecially in imaging when the result of the image processing should be another image , visible for humans.

$$I'(x) = 0 \quad if \quad s \quad I(x) + k < 0$$

$$I'(x) = maxrange \quad if \quad s \quad I(x) + k > maxrange \quad \text{where} \quad maxrange = 255$$

Operator: Point, Neighborhood and Global operators

- Point Operator: the value of each pixel of the resulting image depends only on the original pixel in the same image's spatial position (ex. thresholding)

$$I'(x, y) = h(I(x, y))$$

- Local Operator: the value of each pixel depends only on the original pixel in the same image's position and in a local Neighborhood (ex. filters)

$$I'(x, y) = h(I(x, y)), \aleph(I(x, y))$$

- Global Operator: the value of each pixel depends on all the pixels of the original image (ex. Fourier trasform)

$$I'(x, y) = h(\iint I(x, y))$$

Linear operators

$$g(x) = h(f(x)) \quad or \quad g(x) = h(f_0(x), f_1(x), ..., f_n(x))$$

- x, is the D-Dimensional Domain of the function (D=2 for images); for the image, it is the pixel location, represented by the point coordinates in the 2D plane $x = (x, y)$ or the image coordinates $x = (i, j)$.
- 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 $g(x) = j(f(x)) = s \quad f(x) + k$
  - s, is the scale factor, often called also gain or contrast

– k is the offset constant often called also bias or brightness

Luminance variation Varying the luminance intensity or brightness means only shift the histogram values, by changing proportionally the pixel value in a linearly manner.

$$I'(x) = h(I(x)) = sI(x) + k$$

Linear blend Is a linear pixel-level operator combining two images. It is used in video processing for producing visual effects such as a linear transition or cross-dissolve

$$g(x) = h(f_0(x), f_1(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

Contrast stretching is the expansion of the gray (or color) level of pixels in a dynamic range given the histogram

$$I'(p) = (I(p) - min) * \text{ScaleFactor} + min1$$

Equalization Change the histogram to create ≪better≫ images for perception (introduces a non linear change in the data). Maximizes entropy, gives more "information" and change information

$$S_k = (L - 1)cdf(x)$$

---

5 Saturated arithmetic, color histogram

Image representation, reading/writing process

- Image representation: grayscale images are represented as (H, W) tensors, and color images are represented as (3, H, W) tensors, where the color channels (first axis) will be either in BGR or in RGB order. Pixels are usually represented with 8-bit unsigned integers (torch.uint8 in PyTorch), though they can be converted to other datatypes for processing (e.g., in a Deep Network they are usually casted to 32-bit floats for homogeneity with weights/activation tensors). In grayscale images, the pixel value represents the intensity of the pixel. Increasing its value increases its intensity.
- Reading process: images can be read using the cv2.imread function, which returns a (H, W) np.ndarray for grayscale images, and (H, W, 3) np.ndarray for color images, in BGR order.
- Writing process: images can be show using the plt.imshow(im, cmap='gray') or plt.imshow(im[::-1].swapaxes(0,2)) for reconvert to RGB and to (H, W, 3) for visualization, and finally call function plt.show().

Saturated arithmetics You must be very careful when using 8-bit integers, and pay attention to their numerical limits (0-255) and to the return datatype of each operation. In image processing we always clamp functions between 0 and 255: if a function would modify a pixel to 256, its value is clamped to 255; if a function would return a negative value for a pixel, the value is clamped to 0.

Pixel-wise linear transformation w/ saturated arithm Given as input a color image and two scalars a and b, this transform each pixel p into $a * p + b$ where a is called gain or contrast and b is the offset called bias or brightness.

Normalized image histogram w/ quantization The normalized histogram is the probability distribution (the probability of an occurrence of a pixel of level i in the image)

$$\sum_{i=0}^{255} p_i = 1$$

---

6 Filtering

Image Processing introduction

- Point Processing: same effect of NOISE, they change some pixel content.
- Neighborhood Processing: takes "spatial information" into account, images contain "spatial information"

Noise: types of Noise (signal, computational, perceptual)

- Signal Noise: everything that is not a ≪signal≫ or everything is not an ≪useful information≫, random (not present in the object image) variation of brightness or color information.
$$I'(i,j) = I(i,j) + n(i,j)$$
- Computational noise: an error, acting as a noise, produced by computational tasks with approximations or computational limitation
- Perceptual noise (distractors) everything in the image which is NOT the target of the image itself; sometime called semantic noise, a set of data (pixels) or a part of an image that is not useful and contributed to decrease the perception and understanding of the rest.

Types of Signal Noise

- Salt and pepper noise: random occurrences of black and white pixels
- Impulse noise: random occurrences of white pixels
- Gaussian noise: variations in intensity drawn from a Gaussian normal distribution

White noise A random signal (or process) with a flat power spectral density, it has equal power within a fixed bandwidth at any center frequency. In statistical sense, a time series r t is characterized as having weak white noise if r t is a sequence of serially uncorrelated random variables with zero mean and finite variance. White noise also has the quality of being iid independent and identically distributed, which implies no autocorrelation. Gaussian White Noise: if r t is normally distributed with zero mean and standard deviation

Image restoration, smoothing The image processing subfield which aims at improving the quality of the image by eliminating noise and artefacts. The simplest idea is to do a smoothing process such as the average of the values. Let's replace each pixel with an average of all the values in its neighborhood.

Linear filtering for noise Consists 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. There are 2 type of result:

- shift-invariant: value depends on the pattern in an image neighborhood, rather than the position of the neighborhood
- linear: the output for the sum of two images is the same as the sum of the outputs obtained for the images separately.

Convolution Denoting the operator with the symbol . It is defined as the integral of the product of the two functions after one is reversed and shifted. As such, it is a particular kind of integral transform:

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

Discrete convolution and correlation

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

Discrete Convolution: when use sign - in the function g
Discrete Correlation: when use sign + in the function g
Dot-product or Cross-correlation: when the application of a filter F has variability between -N and N

Local operator application Can be used as a linear filter.

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

$$\text{Convolution} \quad g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = f(k,l)h(i-k, j-l)$$

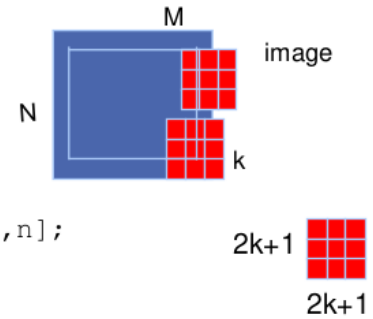
Convolution on an image The output value depends on the pixel value and maybe its neighborhood but not on its position in the image. Convolution is a local process over a window of an image, in an area of the size of the convolutional kernel.

$$g(i,j) = f * h = \int_{-\infty}^{+\infty} f(i-m, j-n)h(m,n)dmdn$$

$$g(i,j) = f * h = \sum_{m}\sum_{n} f(i-m, j-n)h(m,n)$$

```
h(m,n) m=-k..+k, n=-k..+k

for (i=1+k; i<=N-k, i++)
        for (j=1+k; j<=M-k, j++)
        {
        g[i,j]]=0;
         for (m=-k; m<=k, i++)
                for (n=-k; n<=k, n++)
                        g[i,j]=g[i,j]+f[i-m,j-n]*h[m,n];
        }
```



Padding Extension mode in an are where the correct information is not available.

- Zero padding: insert 0 pixels
- 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 Smoothing or blurring, given by averaging the pixel with the neighbor ones. It corresponds to convolving the image with a kernel of 1 values and then scaling . The size of the filter depends on the size of the noise frequency w.r.t. the signal spatial frequency. It limits also the information with the same spatial frequency (blurring) , does not work on saltpepper noise

Gaussian filter The best filter to smooth Gaussian noise is an isotropic mask given by a Gaussian function with zero average value and a given standard deviation, convolved with the image.

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

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

Smoothing with different sigma The filter must be discretized, choosing k, that is the size of the filter, and the standard deviation:
Mask k x k, k about 5 σ

Cognitive smoothing The filter values for smoothing can be defined and fixed forever or could depend on the contexts of from some a priori assumptions.

Sharpening Accentuates differences with local average.

Unsharp Useful for emphasizing transitions in image intensity (e.g., edges). Note that the response of high-pass filtering might be negative.

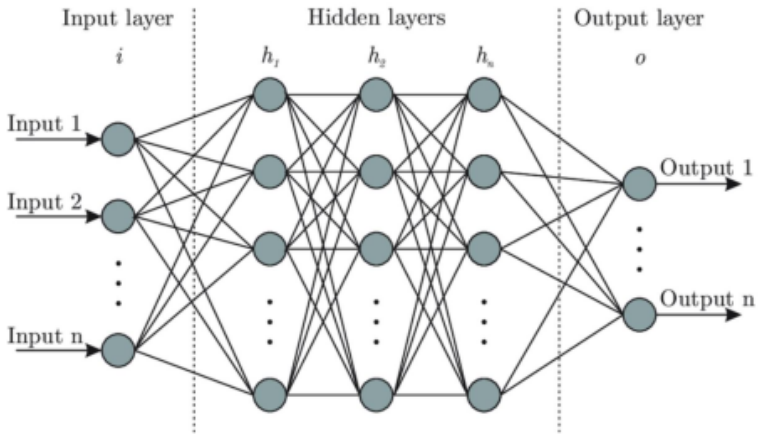
The Neuron as a convolutive operator The Neuron (Perceptron) is only a type of Linear filter.

$$a(x) = \sum_{i=1}^{n} wiAi$$

If the input (Ai) is the function and the set of wi is the filter, you can decide which filter use in dependence of the goal.

Parameters for standard neural networks in an image Consider an input given by an image 100x100 pixels (10.000 inputs), each neuron have 10.000 weights. If we have a number

of neurons as the pixels we have 10.000 x 10.000 parameters → 100.000.000 parameter. If we want to do a number n of filters e.g. n=50 → 5.000.000.000 of different weights to LEARN.



Convolutional Neural Networks and receptive fields

- Neurons do not receive all inputs, but only the ones in a given neighborhood: it is similar to what happen in our brain cortex, called receptive field.
- Each neuron uses the same set of parameters (i.e. the same convolutive kernel), as in filtering to create an output image.
- Many different kernel can be used in parallel to create different output images, often called feature maps.
- Feature maps are processed in a non linear manner (by non linear functions), and compressed by pooling to work in multiresolution or to enlarge the receptive fields.
- Feature vectors can be used for final inference, the reasoning part of the network by full connected layers that "take decisions".

Linear and not linear operators in CNNs Min Max Filters the minimum and maximum value in the moving region R of the original image is the result of the minimum and maximum filter respectively.

$$I'(u,v) \leftarrow \min I(u+i, v+j | (i,j) \in R$$
$$I'(u,v) \leftarrow \max I(u+i, v+j | (i,j) \in R$$

Median Filter Non linear filter useful for impulsive noise, the output pixel is the median value in the neighborhood. Consider a 2D neighborhood, Order it, Choose the central value and Substitute pixel with the median one.

Weighted Median The weight w can be fixed or depending on the distance to the center, median value is the g(i,j) value which minimize this function with p=1 and if p=2 is the weighted mean.

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

Bilateral filter Non linear filter, edge-preserving, and noise-reducing smoothing filter for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution. Crucially, the weights depend not only on Euclidean distance of pixels, but also on the radiometric differences (e.g., range differences, such as color intensity, depth distance, etc.). This preserves sharp edges. Improved weighted filter in a neighborhood of f(i,j) the result g(i,j) is a normalized weighted sum.

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

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

Domain Kernel $d(i,j,k,l) = \exp(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2})$

Range Kernel $r(i,j,k,l) = \exp(-\frac{||f(i,j)-f(k,l)||^2}{2\sigma_r^2})$

Measure Psnr Peak signal-to-noise ratio (PSNR) is computed and is used to measure the difference, where Imax is the max range (eg 255) and MSE the maximum squared error.

$$MSE = \frac{1}{n} \sum_x [I(x) - \widehat{I}(x)]^2$$

$$PSNR = 10 \log_{10} \frac{I_{max}^2}{MSE}$$

Denoising with cnns With Neural networks we can learn the best coefficients for denoising. Off course it works, but in some cases is useless to adopt a very complex procedure (a network to be trained) for a simple calculus.

Learning filter parameters, autoencoders Autoencoders is used to learn the correct reconstruction; useful if we do not know the type of noise. You can create many image processing algorithms . Very often is not necessary... e.g. you can learn by data how to colorize an image in different color. spaces.

Deblurring Blur is a Degradation of sharpness and contrast of the image, causing loss of high frequencies (due to camera motion or camera defocuss).

$$y = x \otimes k + n \Rightarrow y(f) = X(f) * K(f) + N(f) \Rightarrow X(f) = Y(f) * k(f)^{-1} + N(f) * K(f)^{-1}$$

Solutions to Deblurring with Autoencoders Introduced from Mao et al. "Convolutional Autoencoders for Image Restoration". They apply the same network for tasks such as image denoising, image super-resolution, JPEG deblocking, non-blind image deblurring and image inpainting. As in VGG, the kernel size for convolution and deconvolution is set to 3×3, which has shown excellent image recognition performance. They use 64 feature maps

for convolutional and deconvolutional layers with 20 layers.

Examples

---

7 2D Convolution

2D cross-correlation on 2D/3D inputs Application of a filter over an image. The value of each pixel of the image is replaced with a weighted average of its neighborhood. Weights are given, for each pixel of the neighborhood, by the filter (or kernel). In Deep Networks, Convolutional Layers typically apply a cross-correlation with biases. Application: place the kernel in the right position, element-wise multiplication, then sum and apply bias.

2D inputs:
Shape of the input: (H, W), shape of the kernel (kH, kW)
Shape of the output: (H – (kH-1), W – (kW – 1))

3D inputs:
Kernel will be (iC, kH, kW), bias will still be a scalar
A convolutional layer usually takes iC input feature maps, but also produces oC output maps. Each output feature map considers all input feature map, basically, equivalent to performing the operation of the previous slides oC times with different kernels and biases.

2D Convolution in CNNs The input shape is (iC, H, W)
Kernel shape is (oC, iC, kH, kW), bias shape is (oC,) (like having oC different kernels and biases)
Output shape is (oC, oH, oW) (like performing oC different cross-correlations)

Stride, padding

- Stride: controls the stride for the cross-correlation (might be different for the two axes on which the kernel slides). Assume 1, if we don't specify otherwise.
- Padding: the amount of implicit paddings on both sides for padding number of points for each dimension. We assume 0, if we don't specify otherwise. A typical choice is k-1 (where k is the kernel size), so that the output shape matches the input shape on spatial axes. Usually, we pad with 0s, even if other options are on the table (e.g. reflection, constant padding)

Input and output shapes

Input: $(N, C_{in}, D_{in}, H_{in}, W_{in})$
Output: $(N, C_{out}, D_{out}, H_{out}, W_{out})$ where

$$D_{out} = \left\lfloor \frac{D_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[2] - \text{dilation}[2] \times (\text{kernel\_size}[2] - 1) - 1}{\text{stride}[2]} + 1 \right\rfloor$$

Dilation, Grouping

- Dilation: an extension of the regular 2D convolution, in which the kernel is «dilated», Bigger receptive field, same number of parameters, given a dilation factor (l), it is defined

as follows:

$$(F *_l k)(p) = \sum_{s+lt=p} F(s)k(t)$$

If we apply a sequence of increasingly dilated Convolutional layers, we get an exponentially growing receptive field.

- Grouping: In the regular 2D Convolution, all input channels are convolved using all filters, Which implies that all output channels "have seen" all input channels, each through its own filter. Changing this "all-to-all" connectivity schema we could connect each input channel to a specific set of filters. For instance, we could say that the first half of the input channels is connected to the first half of the filters, while the second half of the input channels is connected to the second half of the filters. It's like having two convolutional layers in parallel, or, better to say, two groups running inside the same layer. Following this schema, we could define n groups: input channels are divided into n groups with equal size, and so are the filters. Each i-th group of input channels is connected to the i-th group of filters.

Implementation of a 2D Conv. layer w/ stride 1, no padding, no dilation, no grouping

---

8 Edges

Human and computer vision Connections between computer vision and perceptive psychologies and cognitive science: cognitive science → connected with the mind.
Connections between computer vision and neuroscience: Neuroscience → connected with the brain.

Marr and computational vision Computational vision has three levels:

- Computational level: what does the system do (e.g. what problems does it solve or overcome) and, equally importantly, why does it do these things
- Algorithmic/representational level: how does the system do what it does, specifically, what representations does it use and what processes does it employ to build and manipulate the representations
- Implementational level: how is the system physically realized (in the case of biological vision, what neural structures and neuronal activities implement the visual system)

A standard computer vision pipeline

- Image Processing: take source image, filtering, edge detection selection and segmentation
- Image Analysis: labeling and feature extraction
- Geometry: camera calibration
- Classification: Bayesian networks, Clustering, K-means, SVM and Neural networks

Criteria for feature extraction Is the task which performs the extraction of a n-dimensional vector representing some visual propriety. The feature has to be designed to effectively summarize the visual content: it's a quantization problem, or a compression problem. It's a necessary procedure: Data generalization (avoid overfitting) and Computational constraint.

- Discriminant Propriety: features must assume values that are significantly different for objects belonging to different classes.
- Reliability Propriety: features must assume values that are similar for objects belonging to the same class.
- Independent Propriety: feature must be independent each other.
- Minimum Cardinality Propriety: features must be as few as possible.

Gestalt The central principle of gestalt psychology is that the mind forms a global whole with self-organizing tendencies. Gestalt psychologists identified specific principles of perceptual organization.

- Grouping by similarity
- Grouping by proximity
- Grouping by good continuation

Stability of features Invariant propriety (scaling translation, geometric contraction, expansion, dilation, reflection, rotation, shear, similarity transformations, and spiral similarities plus their combinations)
Subjective relevance propriety: independence from luminance changes, background, occlusions, field of view..

Edge, contours, borders: types of edges

- Edge: is a local proprety of a pixel and its neighborhood to have a «rapid intensity variation» (or it is the location where the rapid intensity variation occurs). It is a vector with a magnitude and a direction, depends on the luminance variation. We can compute this luminance variation as a gradient. The edge has the direction perpendicular to the gradient direction.
- Borders: is a propriety of a Region while Edge is a local propriety. We can compute borders by selecting the high/strong edges
- Contours: occur at boundary between homogeneous regions, but often segmentation is difficult so that it is provided using local variation

Border detection Use of an edge detection operator (edge detector), Selection of strong edges with some given criteria and Linking the edges (labeling)

Methods for edge detection

- Methods based on first derivative computation gradient masks (Roberts, Sobel..)
- Regolarization techniques using filtering and optimal masks (Canny, Sarkar-Bowyler, Marr-Hildreth)
- Border following local techniques based on neighborhood operation of labeled edges and after segmentation
- Classification (using deep learning)

Image gradient, discrete operators Gradient:

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

Gradient direction:

$$\theta = \tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial y})$$

Gradient magnitude:

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

Forward difference:

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

Backward difference:

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

Central difference:

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

Prewitt and Sobel

- Prewitt: to limit noise this Mask do central derivative in one direction and average filter in the other. (Normalize dividing by 1/6)

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

- Sobel: it uses a first central derivative , smoothed in the opposite direction.

$\frac{1}{8}$

-1	0	1
-2	0	2
-1	0	1

s_x

$\frac{1}{8}$

1	2	1
0	0	0
-1	-2	-1

s_y

Canny's Criteria

- Good Detection: search for low error probability in recognizing true edges and recognizing false edges. If non- edge is considered noise, both probabilities are monotonic descent function with the SNR, this criterium corresponds to try to maximize the signal-to-noise ratio (good detection $\rightarrow$ high SNR, high recall)
- Good Localization: The selected edge points by the operator must be more closed as possible to the true edge , to have perfect localization (good localization $\rightarrow$ precise position)
- One Response to Single Edge: if there is a single edge the operator should return a single edge too and low false positives (one response $\rightarrow$ high precision) (partially in the first criterium since if we have two answers, one is a false one)

Canny edge detection
The advantage of hysteresis operator Hysteresis is uised to remove low-strength contour segments. The basic idea of hysteresis is to set two different thresholds and allow a curve

Algorithm

1.) **Smooth** image *I* with 2D Gaussian: $G * I$

2) **Find local edge** normal directions for each pixel $\overline{\mathbf{n}} = \dfrac{\nabla(G*I)}{|\nabla(G*I)|}$

3)Compute edge magnitudes Grad(I,j) $|\nabla(G*I)|$

4)Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**): search for points which cross zero with second derivative

5) **hysteresis-thresholding** $\dfrac{\partial^2(G*I)}{\partial \overline{\mathbf{n}}^2} = 0$

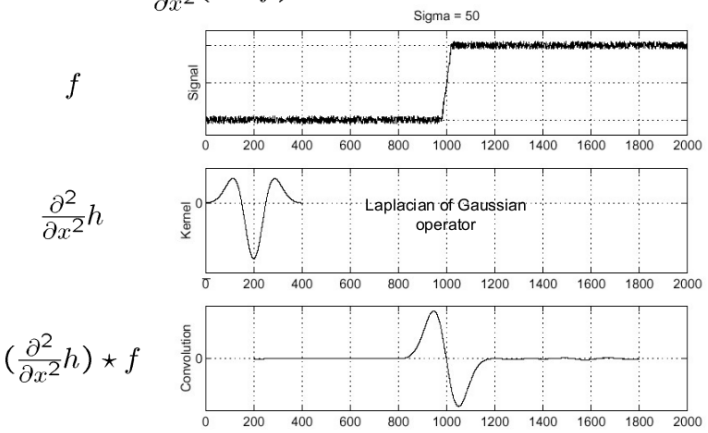
being tracked above the higher threshold to dip in strength down to the lower threshold. It's consist in Select a very strong Thh (strong edges) and Edges are the weak edges (highest than a low threshold Thl) but connected with strong edges.

Normally:

$$T_H \approxeq 2/3T_L$$

Log (Laplacian of the Gaussian) and zero crossing We can compute the second derivative robustly to noise with a smoothing without edge delocalizaion and 2D discretization of the second derivative. A pixel is a zero crossing if $(I(x)t)($ one of neighborhood $N(I(x)) > t)$ or viceversa; Pros: non very sensitive threshold, Cons: The LOG edge detectors does not work in case of discontinuity (e.g. with ANGLES), very sensitive to small closed edges.

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$



(**LOG**, Marr & Hildreth, 1980): Laplacian:

0 1 0 1 1 1
1 -4 1 1/3 1 -8 1
0 1 0 1 1 1

$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

- In general:

a b a
b e b
a b a

With e = -(4a + 4b) e 2a + b = 1

Log can combine in a single operator both Gaussian+Laplacian

2 -1 2
-1 -4 -1
2 -1 2

$\nabla^2(G * I)$

Example of applications

Deep learning edges Edges are very important in machine vision, automotive and in constrained environment, in natural images Edges should be selected according with their semantics. (DeepContour: A Deep Convolutional Feature Learned by Positive-sharing Loss for Contour Detection)

Border tracing If the image has been segmented or binarized we need only to code the borders. Chain code: is one of the possible reppresentations of a contour Freeman code (1966), used for boundary tracing, for compression etc. Given a seed initial point we can follow the bounday in clockwise (or anti clockwise) till we can back to the seed point. The chain code is a list which takes into account the changes of directions from a pixel to the next one the chain codes are the 8 possible directions given p the initial point.
Given a binary image with background = 0 (after thresholding, edge detection etc) and a seed point p_i, if the boundary has at least 3 points each point p_i should have a previous one and a following one p_{i-1} e p_{i+1} connected in a 3x3 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».

0	7	6
1	**p**	5
2	3	4

Splines and active contours B-spline: define the control points to represent a curve with a 3 degree polynomial function or a polynomial function of a higher degree.
Catmull Rom Spline: is the curve which better interpolates the points with a 3 order polynomial function.

Active contours: are boundary detectors which iteratively move towards their final solution under the combination of image and optional user-guidance forces.
Snakes: are an energy-minimizing, two-dimensional spline curve that evolves (moves) towards image features such as strong edges.

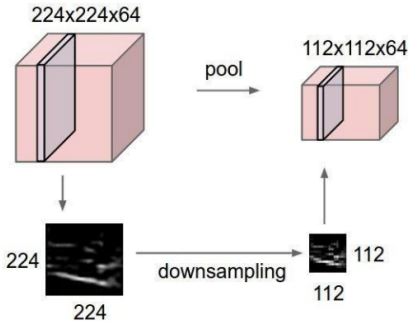
Level set techniques: evolve the curve as the zeroset of a characteristic function, which allows them to easily change topology and incorporate region-based statistics.

Examples of splines and edges
- Following weels boundaries in real time
- Lip Reading
- Find the skyline

---

9 Pooling, Edge Detection, Template Matching

2D Pooling layer Operates over each activation map independently, often used to make representations smaller and more manageable



Input, output sizes, stride

- Input: tensor of size (n, iC, iH, iW), Kernel sieze (kH, kW), Stride S
- Output: tensor output size (n, iC, oH, oW), where oH = (iH – kH)/s + 1 and oW = (iW – kW)/s + 1

Implementation of 2D Max Pooling
Implementation of Sobel Edge detection + visualiz.
Implementation of template matching, in tensorial form

---

10 Shapes

Object, shapes, patterns

- Object: from 3D world, objects, physical thing
- Shape: 2D view of the Object in the image, visual shape and geometrical/ideal shape
- Pattern: connected pixels and visual features

Pattern recognition approaches: fitting vs matching Fitting: a parametric or geometric model to a set of points, usually refers to finding the values of the model parameters giving rise to the best alignment of the corresponding instances of the model with (a part of) the reference set of points. (Linear Regression, The Hough Transform, RANSAC).
Matching: a nonparametric model to a set of point usually refers to finding the best correspondence between the points of the model/template/shape and the reference image/set of points. (Template Matching, Iterative Closest Point (ICP), Global Matching).
Shape Matching: Nonparametric model represented by Image (grey level, colour, depth), Set of 3D points, Global shape descriptors vector, Vector of local feature points with their descriptors, Learned compressed model.

Template Matching Is a simple process of image inspection for verifying the presence of a given subimage that is the template. Similarity is applied against a given template (2D image) or protoype which must be recognized Recognition is provided by measuring the similarity between template and current image. It is the search in all possible positions of the image (sliding window) and the evaluation of a measure about the degree of matching.
Template: shape described by pixels, it is generically a windows of pixels agnostic on its semantic, detection of templates in the real word is difficult: it can change color, size, perspective, rotation etc...
Matching: generic operation consisting in similarity comparison between entities, regions, vectors of the same type.

Template Matching by cross correlation The correlation approach uses the correlation coefficient as a measure of similarity between the reference (template) for each location (x,y) in the target image. The result will be maximum for locations where the template have correspondence (pixel by pixel) to the subimage located at (x,y). Cross-correlation of an image g with a t template of (2k+1)x(2k+1) pixels:

$$R(m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} g(m+i, n+j) * t(i,j)$$

Normalized Cross Correlation (NCC) Is the cross correlation , normalized with respect to the average value in the neighbourhood. It is used for searching a template in an image or for searching a given image windows in successive frames. (For image-processing applications in which the brightness of the image and template can vary due to lighting and exposure conditions, the images can be first normalized)

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

Block matching Is used to find motion vector in consecutive frames. Given It e It+1 search for the motion of the block k x k to compute the motion vector. In block matching the SSD is normally used, more than cross correlation. Acceptable if not rotation is considered. (Application: Machine vision, Multimedia and Surveillance)

Extended and Fast Affine template matching Extended: NCC is useless when the template changes in color, shape, rotate, size, affine or perspective transformations, for this reason we can Transform the pattern and Use compact descriptors.
Fast Affine: is a fast algorithm for approximate template matching under 2D affine transformations that minimizes the Sum-of-Absolute-Differences (SAD) error measure. Very useful in frames when the shape/object is the same.

Geometry: homogeneous coordinates A system of coordinates used in projective geometry. Homogeneous coordinates have a range of applications, including computer graphics and 3D computer vision, since they allow affine transformations and, in general, projective transformations. The points are projected from a space of dimension n to a space of dimension n+1 without the center of the coordination.

Projective plane Is an ordinary Cartesian plane augmented with a point at the infinite. [Study in deep...]

2D transformations: Isometries (Translation and Rotation) Isometries: all Euclidean transformations are isometries.

Translation transformation preserves the orientation and area. Translations can be written as $x' = x + t$ or

$$x' = [I \quad t]\bar{x}$$

where I is (2 x 2) identity matrix or

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

where 0 is the zero vector.

Rotation + Translation is also known as 2D rigid body motion or the 2D Euclidean transformation (since Euclidean distances are preserved). It can be written as $x' = Rx + t$ or

$$x' = [R \quad t]\bar{x}$$

where

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \bar{x}$$

is an orthonormal rotation matrix with $RR^T = I and |R| = 1$.

2D transformations: Similarities The similarity transform preserves angles between lines. This transformation can be ex-pressed as $x' = sRx + t$ where s is an arbitrary scale factor. It can also be written as

$$x' = [sR \quad t]\bar{x} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{x}$$

2D transformations: Affine and Projective The affine transformation is written as $x' = A\bar{x}$, where A is an arbitrary 2×3 matrix

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{x}$$

Parallel lines, ratios of parallel lengths, ratios of areas remain invariants.

Projective: or also known as perspective transform or homography, operates on homogeneous coordinates, $\tilde{x}' = \tilde{H}\tilde{x}$, where $\tilde{H}$ is an arbitrary 3×3 matrix and is homogeneous, i.e., it is only defined up to a scale, and that two $\tilde{H}$ matrices that differ only by scale are equivalent. The resulting homogeneous $\tilde{x}'$ must be normalized in order to obtain an inhomogeneous result x, i.e.,

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \quad and \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

Perspective transformations preserve straight lines (i.e., they remain straight after the transformation).

3D transformation The set of three-dimensional coordinate transformations is very similar to that available for 2D transformations. As in 2D, these transformations form a nested set of groups.

Parametric transformation algorithms Given a transformation $x' = Hx$ and given initial image $I = f(x)$ we can create a new image $I' = g(x)$ with two type of algorithm:

- Forward algorithm: which for every pixel x in f(x) compute the destination location x'=h(x) and copy the pixel f(x) to g(x'). This is the straightforward algorithm but has some limitations, x' can be not integer and then a further transformation or blending is needed to distribute it in the neighborhood (in computer graphics called splatting), some x' point can be not defined
- Inverse algorithm: which for every pixel x' in g(x') compute the source location $x = \hat{h}(x')$ resample f(x) at location x and copy to g(x')

It is defined in each point (no holes), if the initial location is not an integer coordinates some interpolation methods can be adopted (neighbour, bilinear, bicubic etc), the inverse kernel in this case is the inverse matrix.

Perspective parameters estimation

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

- Select 4 points: in the same plane but not collinear (4 points = 8 coordinates and 1 is scale factor)
- Divide with the third equation: $1 = h_{20}x_i + h_{21}y_i + 1$
- Repeat this process for the 4 points

Example of applications

- Stiching Panorama

- Extended Template matching
- Augmented Reality

Ransac Random sample consensus) is one of the most famous iterative math methods for fitting parameters of a mathematical model from a set of observed data which contains inliers (the points of the model) and outliers. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability it is used for many applications, and possibly for lines too RANSAC is a stochastic algorithm for fitting a model to a dataset that contains outliers, i.e., points that cannot be explained by a model instance.

Hough transform Hough transform transforms,the points in the coordinate space into points of the parameter space where the evidence of a curve is accumulated.
Classical Hough Transform can locate regular curves like straight lines, circles, parabolas, ellipses, etc. It is a technique to isolate the curves of a given shape in a given image. It requires that the curve is specified in some parametric form. Generalized Hough Transform can be used where a simple analytic description of feature is not possible.
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 but it can be used to detect but not to localize.

EHT and HT

- Edge based HT: using only feature points because in case of vertical lines classical representation $y = mx + c$ fails, a more useful representation in this case is the polar representation $x\cos(\theta) + y\sin(\theta) = r$. All the points of a line are represented by a pairs of parameters, the distance from the origin, r (algebric distance due to the sign that can be negative), and the angle (theta discretized into an). Any line in the image space , can be represented by the pairs (r, an).
- HT: transforms points (feature points) from the image space to the discrete Hough space where points belonging to the same parametric shape accumulate "votes" in the same point of the Hough space. Each line in the Cartesian space (x,y) corresponds to a point in the (m,c) parameter space.

HT for circles Parametric equation of the circle:

$$(x - x0)^2 + (y - y0)^2 = r0^2$$

The equation has three parameters: a, b, r. A circle corresponds to a point in the a, b and r space. The curve obtained in the Hough Transform space for each edge point will be a right circular cone Point of intersection of the cones gives the parameters a, b, r.
Gradient at each edge point is known, we know the line on which the center will lie and if the radius R is also known then center of the circle can be located. The HT search the peaks of accumulations as (x0, y0), it is robust to partial view.
Gradient HT and Generalized HT We can use a whichever shape (boundary shape) that can be described in a parametric way. Given a point of origin xc yc, the shape is a list of points represented by a pair (ri, ai). We can create an hash table to collect the points. 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). It can be very useful also for finding partial shapes. Also used in tracking. It is independent from rotation and translation: The size and orientation of the shape can be found out by simply manipulating the R-Table, For scaling by factor S multiply

the R-Table vectors by S, For rotation by angle θ, rotate the vectors in the R-Table by angle θ.

Examples

- Lane detection for the streets
- Delineation of Road Networks
- Fruit Detection

---

11 Morphology, Hough Transform

Morphology operators Morphology operators are neighborhood operators used to modify the shape of binary patterns using a sliding-window protocol (like convolution).

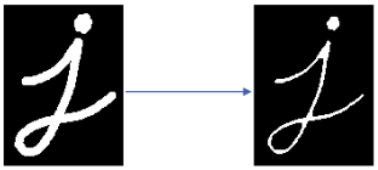
Dilation, erosion, opening, closure

- Dilation: "expands" A with B. As the structuring element B is scanned over the image, we take the maximum pixel value of the neighborhood which is overlapped with a positive value of B.

$$result(x, y) = \max(p \quad for(p, k) \quad in \quad zip(neighborhood, kernel) \quad if \quad k > 0)$$

- Erosion: "reduces" A with B. As the structuring element B is scanned over the image, we take the minimum pixel value of the neighborhood which is overlapped with a positive value of B.

$$result(x, y) = \min(p \quad for(p, k) \quad in \quad zip(neighborhood, kernel) \quad if \quad k > 0)$$



- Opening: is a sequence of an erosion and then a dilation, useful for removing noise, and separating shapes with small morphological defects.

- Closing: is a sequence of a dilation and then an erosion, useful for closing small holes.

Hough transform (w/ OpenCV)
- Lines: HoughLines(image,rho, theta, threshold, ...)
- Circles: HoughCircles(image, method, dp, minDist, circles, param1, param2, minRadius, maxRadius)

12 From image processing to AI-based Vision

From Pattern Recognition to machine learning approaches Machine learning techniques have always played an important and often central role in the development of computer vision algorithms. Computer vision in the 1970s grew out of the fields of artificial intelligence, digital image processing, pattern recognition (now called machine learning) and one of the premiere journals in our field still bears testament to this heritage.

Example. fingerprint In the past pattern matching, now (since decades) minutiae matching. A good fingerprint contains between 20 to 80 minutiaes depending on the sensor and the position. Some false minutiaes arise for insufficient ink or poor quality.

- Segmentation (binarization E.g. Otsu)
- Minutiae detection
- Minutiae classification
- Minutiae matching
- Find total minutia points
- Find their location

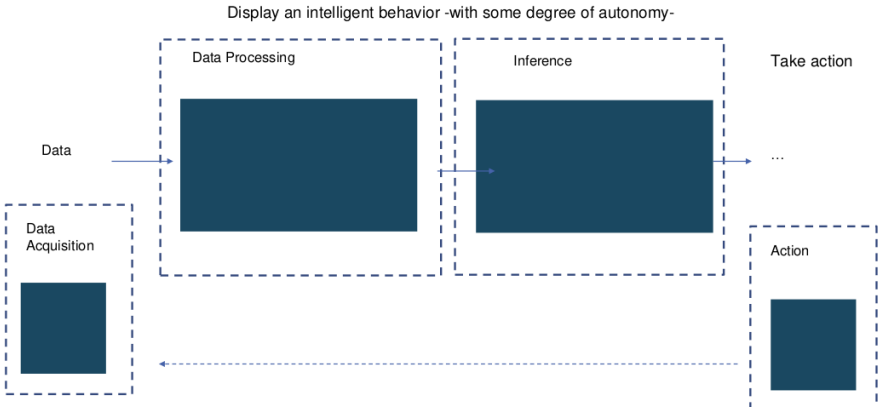
Biometric stability

- Universality: Only very few people miss all 10 fingers. Most fingerprint recognition software allows to enroll multiple fingers which avoids that an individual is no longer granted access after injury.
- Uniqueness: It is generally accepted that fingerprints are unique to an individual. However, there is a risk that fingerprints of two different individuals match if the fingerprint image is of insufficient quality. Therefore the False Acceptance Rate (FAR) is highly dependent on the quality of the fingerprint reader.
- Permanence: Fingerprints do not change with ageing, but as people age they lose collagen which makes their fingerprint harder to read and this can lead to significantly more false rejects with elderly people. Injuries, such as fire wounds, can damage a fingerprint but if multiple fingers are enrolled the likelihood of an authorized individual being denied access is reduced.
- Collectability: Fingerprints are easy to acquisition, the cheapest fingerprint readers available use a digital camera. Fingerprint readers that are more difficult to fool, such as CMOS readers, are even not overly expensive. In some environments, where for example people are unable to wash their hands, more expensive means might be necessary to acquire a useable fingerprint image.
- Acceptability: Fingerprints are easily accepted as soon as people reflect that they leave their fingerprints everywhere and that no sensitive information, such as medical conditions, can be derived from fingerprints.
- Circumvention: There are a number of concerns when using fingerprint recognition.

Statements in vision: vision as a cognitive process

- Vision requires to extract what is characteristic of a target (scene, object, event.. Or keypoints, details): Features and Descriptors (can be feature vectors, tensors but also complex structure like graphs)
- Vision requires an inference engine to provide reasoning (association, memory, logic deduction..): Classification, Retrieval, Detection, Understanding...
- Visual Features: They should be representative, accurate, discriminative, independent, invariant and with the minimum cardinality, to speed up computation and to provide a better reasoning by focusing on the necessary only.
- Challenges: select the good (best) features for the target, select the adequate (best) description for the understanding process and select the minimum (best) cardinality of features.

The change of the pipeline Classical Pipeline:



Explainability and trustworthy Guidelines:

- Human agency and oversight
- Technical robustness and safety
- Privacy and data governance
- Transparency
- Diversity, non discrimination and fairness
- Societal and environmental well being
- Accountability

Supervised vs unsupervised classification In Supervised Learning, paired training inputs x_i (which may be features derived from input images ore previously computes) and labels/targets t_i are used to estimate the model parameters that best predict the labels from their corresponding inputs. At run time the model parameters are frozen and the model is applied to new inputs to generate the desired outputs.
In Unsupervised Learning there are no labels or outputs, e.g. to find similarities as well as semi-supervised learning, in which labels or targets are only provided for a subset of the data. Then we will have self-supervised learning when we have dataset of possible input and

output as in generative vision.

Learning and loss The loss function L measures the "cost" of predicting an output f(xi; w) for input xi and model parameters w when the corresponding target is yi. In classification tasks, it is common to minimize the miss-classification rate, penalizing all class prediction errors equally using a class-agnostic delta function.

Binary classification Use Logistic classification, where:

$$t_i \in 0, 1 \quad p_i = p(C_o|x_i) = \sigma(l_i) = \sigma(w^T x_i + b)$$

We can maximize the likelihood of the correct labels being predicted by minimizing the negative log likelihood, the cross-entropy loss or error function:

$$E_{CE}(w, b) = -\sum_i t_i \log p_i + (1 - t_i) \log(1 - p_i)$$

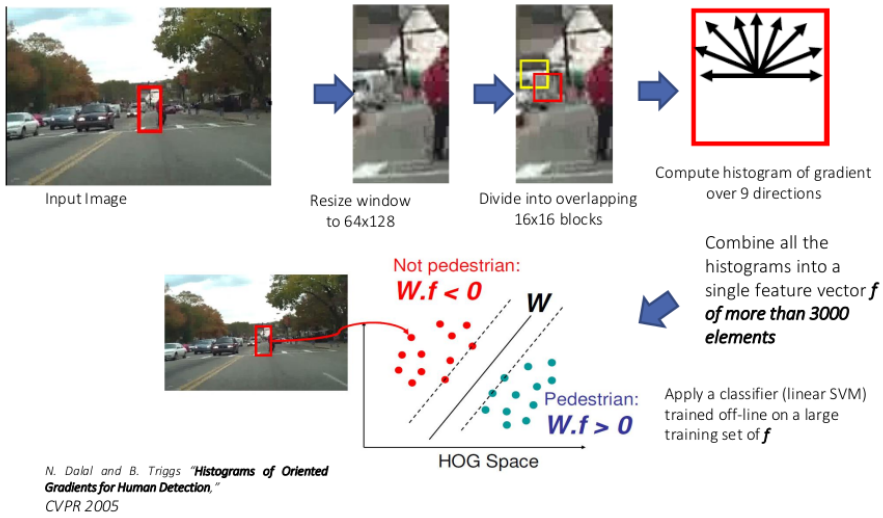
Semi-supervised learning Often in Vision we have a modest amount of accurately labeled data and a far larger set of unlabeled or less accurate data. There are two varieties:
- Transductive learning: the goal is to classify all of the unlabeled inputs that are given as one batch at the same time as the labeled examples; - Inductive learning: we train a machine learning system that will classify all future inputs, all the regions in the input space. E.g. for online applications such as autonomous driving or new content classification.

Data and datasets, data preprocessing Computer Vision need Examples of Visual Data, sometimes (often) need a Dataset.
Data preprocessing consist in:
- Centering the feature vectors means subtracting their mean value
- Standardizing the feature vectors means re-scaling each component so that its variance (average squared distance from the mean) is 1.
- Whitening is a more computationally expensive process, which involves computing the covariance matrix of the inputs, taking its SVD, and then rotating the coordinate system so that the final dimensions are uncorrelated and have unit variance (under a Gaussian model).

Datasets first generation

- Pascal VOC (2008) for objects detection
- OpenVisor (2008-2010) for surveillance
- INRIA (2005) for recognition of people

HOG for people detection

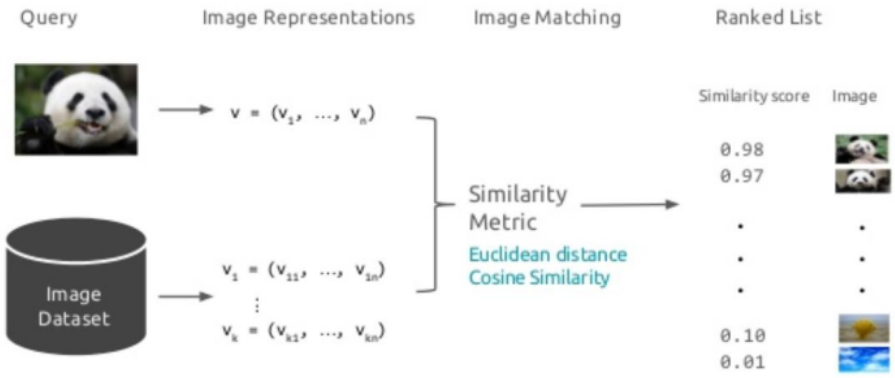


Datasets second generation

- Caltech: video taken from vehicle driving through regular urban traffic, is used for Pedestrian detection and is good also for learning occlusions
- ImageNET (2010): for object detection, used in the creation of Alexnet
- AlexNET (2012)
- SUN: motivated to cover a wide variety of scenes, wide range of object categories
- KITTI (2012): first large Dataset for autonomous driving
- Mapillary Vistas (2017): Dataset in Transport
- COCO (2015): for instance segmentation of objects in natural context
- DUKE-MCMT (2019): datasets used in multicamera surveillance

Retrieval, features and embedding

Image retrieval: the pipeline



Biometrics: identification vs authentication

- Identification: Biometric identification can be applied to digital and physical scenarios, and it's a solution that is used in defense, law enforcement, and border control. With identification, there is a database that contains physical characteristics of a vast number of people (FBI's repository stores the height, hair color, weight, eye color, scars, and tattoos of over 70,000,000 criminal records)
- Authentication: Biometric authentication's aim is to verify that you are who you are supposed to be. With such systems, a computer will scan a person for inherent attributes (face recognition template, and will then compare the individual's characteristics to a template stored within a database, if the scanned attributes match the template, the person is allowed into the system)
- Verification: Biometric verification indicates that a person has the same biometric features as somebody who is in the presented online data, verification can conclusively prove that their online identity is linked with their real-life identity.

Performance Measure in classification

- 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: $\frac{\sum TP}{\sum TP+FP}$
- Recall: $\frac{\sum TP}{\sum TP+FN}$
- Accuracy: $\frac{\sum TP+TN}{\sum TP+FP+FN+TN}$
- Error: $\frac{\sum FP+FN}{\sum TP+TN+FP+FN}$
- Intersection over Union/Jaccard index: $\frac{B_{pr} \bigcap B_{gt}}{B_{pr} \bigcup B_{gt}}$

Performance Measure in medicine

- Sensitivity = True Positive Rate = Recall: $\frac{TP}{TP+FN}$
- Specificity = True Negative Rate: $\frac{TN}{TN+FP}$

Performance Measure in biometrics

- True Acceptance Rate (TAR): $\frac{TP}{TP+FN}$ (ideally close to 1)
- False Acceptance Rate (FAR): $\frac{FP}{FP+TN}$ (ideally close to 0)

Measures for relevant retrieval, MAP

- Recall: $\frac{Retrieved \quad Relevant}{Relevant \quad in \quad collection}$
- Precision: $\frac{Retrieved \quad Relevant}{Retrieved \quad in \quad collection}$

- F-Measure: is a single measure that that takes into account both recall and precision. It is the the weighted harmonic mean of precision and recall. Compared to arithmetic mean, both precision and recall must be high for harmonic mean to be high.

$$F = \frac{2PR}{P+R} = \frac{2}{\frac{1}{R}+\frac{1}{P}}$$

- E-Measure: a variant of F-measure that trades off precision versus recall. Allows weighting emphasis on precision over recall (Beta controls trade-off).

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

- R-Precision: is the precision at a given relevant cardinality $\frac{Relevant \quad docs}{Position}$
- 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.

$$AP = \frac{1}{GTP}\sum \frac{TP_{seen}}{i}$$

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

$$MAP = \frac{1}{N}\sum AP$$

Distance: Euclidean vs Cosine

- Euclidean distance: suppose two feature vectors x and y in R^n, then the Euclidean distance is:

$$d_{euclid} = ||x-y||_2 = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} = \sqrt{||x||^2+||y||^2-2xy}$$

- Cosine distance/similarity: $s = \frac{xy}{||x||||y||}$

Other distances in retrieval

- Distance between functions: $D^r(I,J) = \frac{|\mu_r(I)-\mu_r(J)|}{|\sigma(\mu_r|} + \frac{|\sigma_r(I)-\sigma_r(J)|}{|\sigma(\sigma_r|}$ or $\rho(p,q) = \int \sqrt{p(z)q(z)}dz$
- Distance between histograms: $D(I,J) = (\sum_i |f(i,I)-f(i,J)|^P)^{1/p}$
- K-L distance: $D(A,B) = \sum_{i=0}^{b-1} P_i(B) log \frac{P_i(B)}{P_i(A)}$
- Bhattacharyya distance: $D(A,B) = -\log \sum_{i=0}^{b-1} \sqrt{P_i(A)P_i(B)}$

Feature vector descriptor: histogram Color histogram is a representation of the distribution of colors in an image and can be build for any kind of color space (RGB or HSV). Color information is faster to compute compared to other invariants. Histograms for classification is that the representation is only dependent of the color, ignoring its shape and texture and have high sensitivity to lighting intensity changes.

Feature vector descriptor: SIFT and BoW
- Scale-invariant feature transform (SIFT): a feature detection algorithm in computer vision to detect and describe local features in images. It was published by David Lowe in 1999. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving. SIFT Feature Vectors : n SIFT = n x 128 bin.
- SIFT + BoW: still very used, consist in extract a number of keypoints (e.g. SIFT in the image), Embedding with Bag of Word and Find for similarity with NN.

Feature vector descriptor: CNN One of the first experiment is in CVPR 2014 using CNN for retrieval. AlexNET is the first fully connected layer (layer 22) of the network; with a feature vector of 4026 dimensions, the feature vector is then L2 normalized + SVM for classification.

Datasets for retrieval Supervised training: training the network for classification over 1000 classes Imagenet. After evaluation of the Layer to use the Best result are: FC7 layer (feature of 4096D), L2 norm + PCA and Euclidean Distance.

Bag of word An image can be treated as a document, and features extracted from the image are considered as the visual words. An image is represented in the same way, as an unordered collection of visual words. Bag of Visual Words have been successfully used to object categorization in images (e.g. faces, car, airplanes...) and, more recently, for action recognition in video sequences (e.g. running, walking, clapping...).
This method requires:

- Codebook (vocabulary): training images of objects are selected and visual words are obtained from local descriptors (offline process)
- Classifier training: a classifier is trained for each object class from the visual words (offline process)
- Recognition: the trained classifier classifies the input image according to the learned model (online process)

Local features keypoints: detector and descriptors
- Detector: finds the more interesting (significant) points or regions within the image.
- Description: describes a patch around the point, or the region, with a vector of features.

Harris detector The first detector of SALIENT points, or KEYPOINTS. They are characterized by A formal definition (perecptual, algebric), a precise position (localization) and possibly some invariance (eg to rotation, luminance etc). Harris detector uses the concept of autocorrelation (again). Patches with gradients in at least two (significantly) different orientations are the easiest to localize.

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

Where:

- D, is the information content associated with p, relative to the window
- $\Omega(q)$, is the neighborhood window

- d, is the direction which is the unitary vector

The information content D is the value computed as the intensity difference E(u,v) after the shift (u,v):

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

Where:

- $w(x,y)$ is the window function and can be Gaussian or (1 in windows and 0 outside)
- $I(x+u, y+v)$ is the Shifted Intensity
- $I(x,y)$ is the Intensity

Taylor series approximation to shifted image is:

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

$$= \sum_{x,y} w(x,y)[uI_x + uI_y]^2 = \sum_{x,y} w(x,y)(u,v) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Considering the Taylor expansion valid, i.e for small shifts [u,v], we can use the maticial form of the bilinar approximation, being M the 2x2 image derivatrive matrix.

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

Where M is the structure tensor/auto-correlation matrix/Hessian matrix:

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

The point p in the image is «salient» (as a corner) if the information content is meaningful enough, Thus if a (meaningful) threshold value exhists so that this equivalent to say:

$$\min d^T M d | d \in R^2 . ||d|| = 1 > \tau$$

Where the minimum eigenvalue λ is higher than the threshold τ

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

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

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

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

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

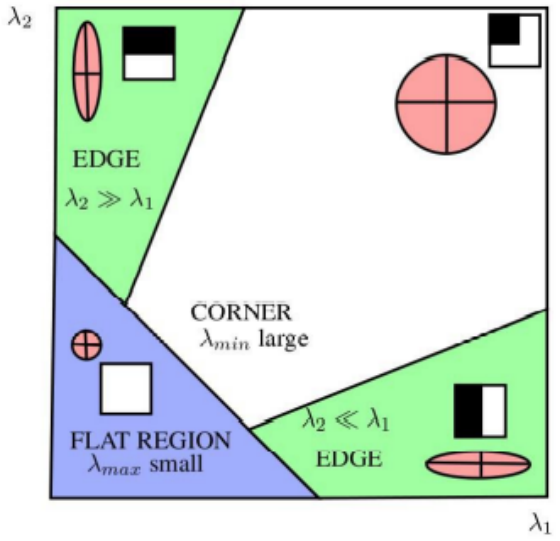
$$R = Det(H) - k(Trace(H))^2$$

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

Harris for geometrical point of view The easiest way to visualize and reason about this uncertainty is to perform an eigenvalue analysis of the auto-correlation matrix A, which produces two eigenvalues and two eigenvector directions. Since the larger uncertainty depends on the smaller eigenvalue, it makes sense to find maxima in the smaller eigenvalue to locate good features to track (Shi and Tomasi 1994).

$$M = (v_{max} v_{min}) \begin{pmatrix} \lambda_{max} & 0 \\ 0 & \lambda_{min} \end{pmatrix} \begin{pmatrix} v_{max}^T \\ v_{min}^T \end{pmatrix}$$

Where v_{min} and v_{max} are the the eigenvectors. Corners and edges can be distinguished by analyzing the Hessian eigenvalues computed on a sliding window $w(x, y)$. Eigenvalues λ_1 and λ_2 of M reveal the intensity of the gradient change within the two most significant orthogonal direction.



Sift: Scale Invariant Feature Transform It has been introduced by D. Lowe in 2004 to represent visual entities according to their properties with empirically determined parameters. The method employs local features taken in correspondence of salient points (referred to as keypoint of SIFT point). Image points selected as keypoints and their SIFT descriptors are robust under: Luminance change (based on the gradient analysis), Scale change (due to scale-scape), Rotation (due to local orientations wrt the keypoint canonical). They have been VERY IMPORTANT IN CV because Stressed in the importance of keypoints Demostrate the powers of descriptors for searching recognizing.

1. Scale-space extrema detection
   1. Build a **Gaussian-blurred image pyramid**
   2. Subtract adjacent levels to obtain a **Difference of Gaussians (DoG) pyramid (so approximating the Laplacian of Gaussians)**
   3. Take **local extrema of the DoG filters at different scales as keypoints**
2. Keypoint localization
3. **Orientation assignment**
4. **Keypoint descriptor**

Other descriptors

- FAST and FASTER
- BRISK
- ORB
- LIFT
- SuperPoint
- LF-Net
- Key-Net
- D2D

Example of applications

- Object recognition
- Image retrieval
- Robot location
- Pick and Place
- Structure from Motion
- Augmented reality
- Automatic image stitching

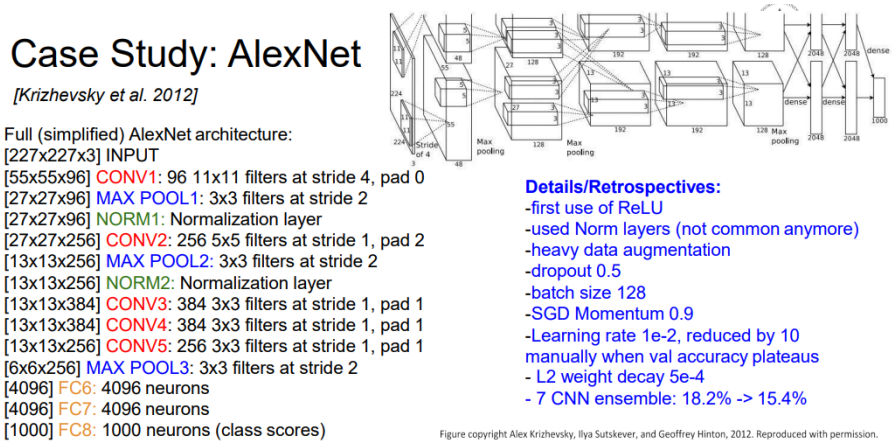
Bow with Deep Learning

Vlad,Fisher Gold, Neuralstory, examples Vector of Locally Aggregated Descriptors (VLAD) differs from the classical BoW by recording the difference from the cluster center, rather than the number of descriptors assigned to the cluster. VLAD records aspects of the distribution of descriptors assigned to a cluster center, it includes the mean of local descriptors.
Fisher Vector encoding provides a more efficient representation than BoW including more statistical information at similar costs.

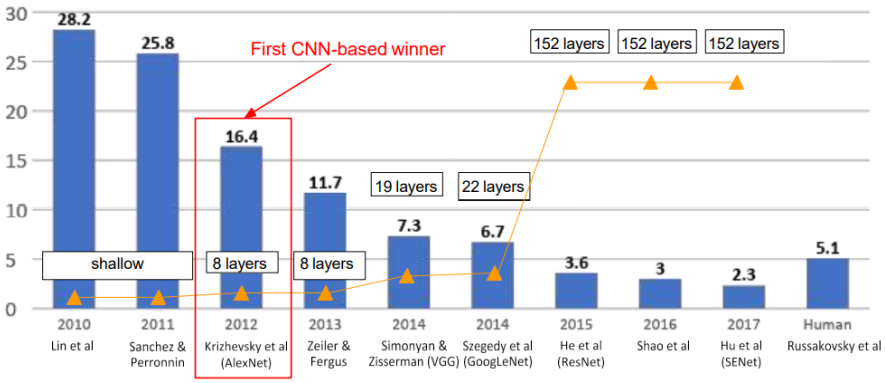
---

13 SoTA CNN Models

AlexNet: architecture



The ILSVRC Challenge



VGGNets: architecture, kernel sizes, channels, memory occupation, parameters Stack of three 3x3 conv (stride 1) layers has same effective receptive field as one 7x7 conv layer. Total memory occupation is 96 MB/image and total parameters are 138M.

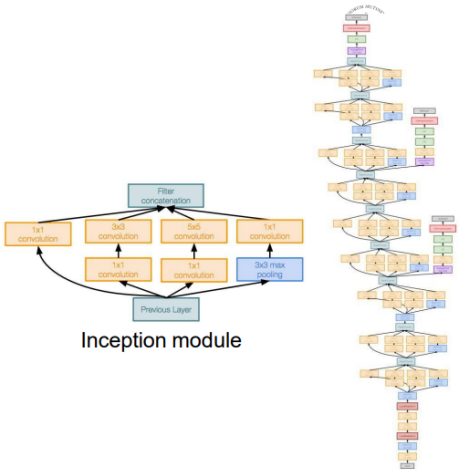
GoogleNet: inception layer (naive and final), role of 1x1 convolutions, auxiliary classifier outputs, gradient vanishing issues "bottleneck" layers that use 1x1 convolutions to reduce feature depth, auxiliary classifier outputs is used to inject additional gradient at lower layers.

Case Study: GoogLeNet

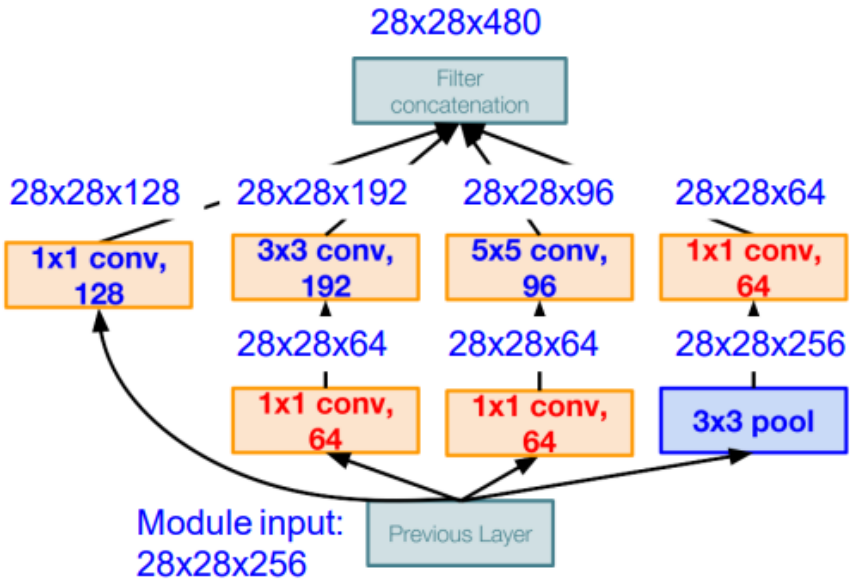
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient "Inception" module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC'14 classification winner
  (6.7% top 5 error)



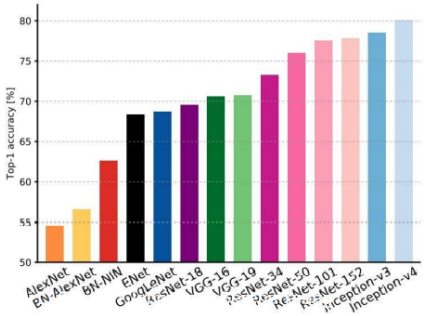
Inception module

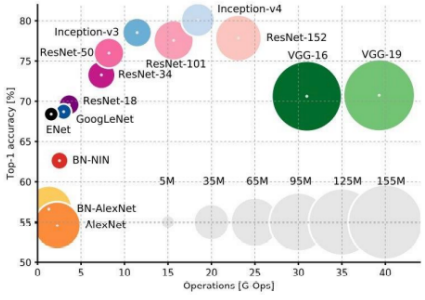


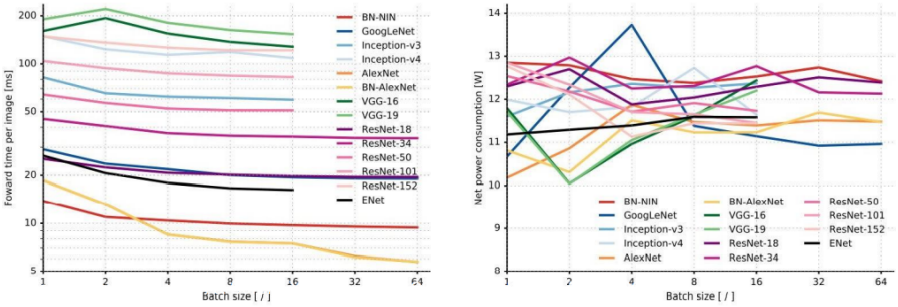
ResNet: motivation, residual block, gradient of residual block, bottleneck layers, architecture of ResNet-152

Implementation of a residual layer

Complexity/Accuracy comparison of CNNs. Forward pass times, memory occupation, power consumption

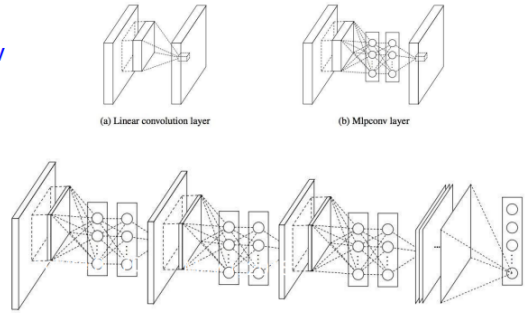






Network-in-network

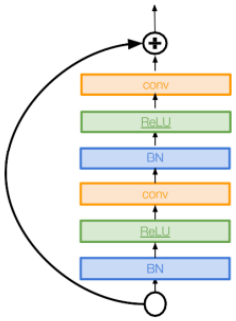
- Mlpconv layer with "micronetwork" within each conv layer to compute more abstract features for local patches
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)
- Precursor to GoogLeNet and ResNet "bottleneck" layers
- Philosophical inspiration for GoogLeNet



Figures copyright Lin et al., 2014. Reproduced with permission

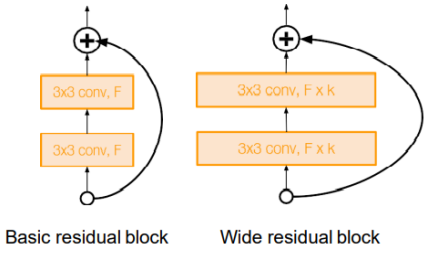
Identity mappings in Residual Networks

- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout network (moves activation to residual mapping pathway)
- Gives better performance



Wide Residual Networks

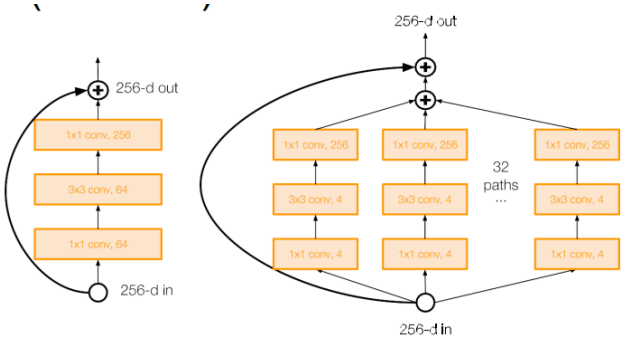
- Argues that residuals are the important factor, not depth
- User wider residual blocks (F x k filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)



ResNeXt

[Xie et al. 2016]

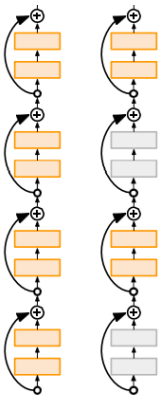
- Also from creators of ResNet
- Increases width of residual block through multiple parallel pathways ("cardinality")
- Parallel pathways similar in spirit to Inception module



Deep Networks with Stochastic Depth

[Huang et al. 2016]

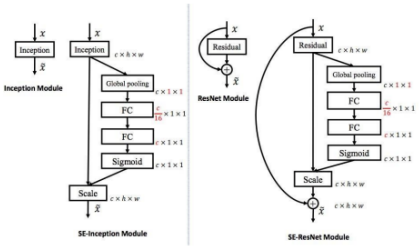
- Motivation: reduce vanishing gradients and training time through short networks during training
- Randomly drop a subset of layers during each training pass
- Bypass with identity function
- Use full deep network at test time

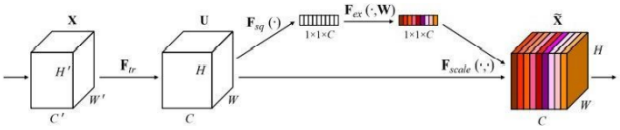


Squeeze-and-Excitation block

[Hu et al. 2017]

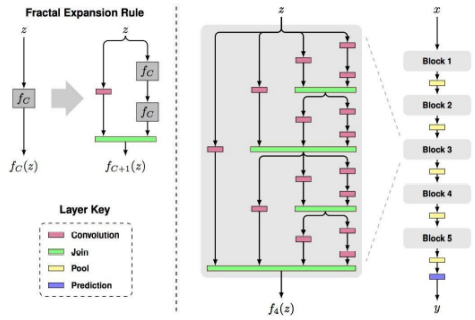
- Add a "feature recalibration" module that learns to adaptively reweight feature maps
- Global information (global avg. pooling layer) + 2 FC layers used to determine feature map weights
- ILSVRC'17 classification winner (using ResNeXt-152 as a base architecture)





FractalNets

- Argues that key is transitioning effectively from shallow to deep and residual representations are not necessary
- Fractal architecture with both shallow and deep paths to output
- Trained with dropping out sub-paths
- Full network at test time



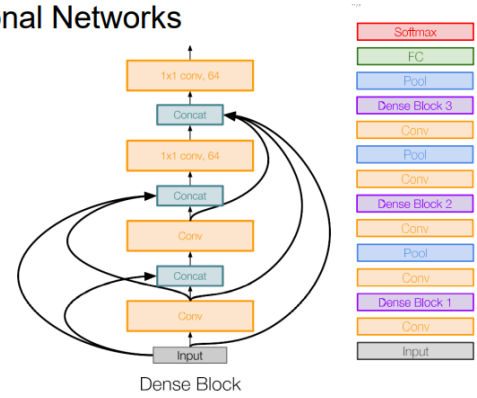
Figures copyright Larsson et al., 2017. Reproduced with permission.

Densely Connected CNNs

Densely Connected Convolutional Networks

[Huang et al. 2017]

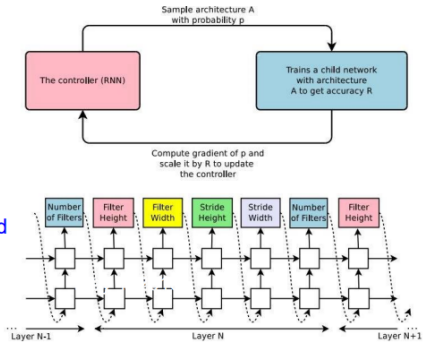
- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse



Neural Architecture Search w/ RNN-based controllers

[Zoph et al. 2016]

- "Controller" network that learns to design a good network architecture (output a string corresponding to network design)
- Iterate:
  1) Sample an architecture from search space
  2) Train the architecture to get a "reward" R corresponding to accuracy
  3) Compute gradient of sample probability, and scale by R to perform controller parameter update (i.e. increase likelihood of good architecture being sampled, decrease likelihood of bad architecture)



14 Autograd, developing and training NNs in PyTorch

Chain Rule recap Suppose that functions $y = f(u)$ and $u = g(x)$ are both differentiable, then the chain rule states that $\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$

Computational graph construction for automatic differentiation Tracking which data combined through which operations to produce the output. Automatic differentiation enables the system to subsequently backpropagate gradients, backpropagate simply means to trace through the computational graph, filling in the partial derivatives with respect to each parameter.

PyTorch's requires grad and backward Requires grad (True), attribute of a Tensor tells the autograd module to begin recording operations on the tensor, and creates the .grad attribute where the gradient will be stored.
y.Backward() calculate the gradient of y and with x.grad we can visualize it.

Computing the gradient in presence of Python control flow statements

Detaching computation

Building a NN and training loop

```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)

model = torch.nn.Sequential(
          torch.nn.Linear(D_in, H),
          torch.nn.ReLU(),
          torch.nn.Linear(H, D_out))

learning_rate = 1e-2
for t in range(500):
    y_pred = model(x)
    loss = torch.nn.functional.mse_loss(y_pred, y)

    loss.backward()

    with torch.no_grad():
        for param in model.parameters():
            param -= learning_rate * param.grad
    model.zero_grad()                                    10
```

Sequential block, nn.Module The following code generates a network with one fully-connected hidden layer with 256 units and ReLU activation, followed by a fully-connected output layer with 10 units (no activation function).

```
import torch
from torch import nn
from torch.nn import functional as F

net = nn.Sequential(nn.Linear(20, 256), nn.ReLU(), nn.Linear(256, 10))

X = torch.rand(2, 20)
net(X)
```

In the following snippet, we code up a block from scratch corresponding to an MLP with one hidden layer with 256 hidden units, and a 10-dimensional output layer.

```
class MLP(nn.Module):
    # Declare a layer with model parameters. Here, we declare two fully
    # connected layers
    def __init__(self):
        # Call the constructor of the MLP parent class Module to perform
        # the necessary initialization. In this way, other function arguments
        # can also be specified during class instantiation, such as the model
        # parameters.
        super().__init__()
        self.hidden = nn.Linear(20, 256)  # Hidden layer
        self.out = nn.Linear(256, 10)  # Output layer

    # Define the forward propagation of the model, that is, how to return the
    # required model output based on the input X
    def forward(self, X):
        # Note here we use the funtional version of ReLU defined in the
        # nn.functional module.
        return self.out(F.relu(self.hidden(X)))
```

15 Detection architectures

Detection as classification+localization Detection as regression We have 4 numbers (x, y, w, h) for each object detected, in case we have a lot of these we need variable sized outputs because each image needs a different number of outputs (not standard number of object in each image). In case of Detection as Classification we have to answer only yes or not.

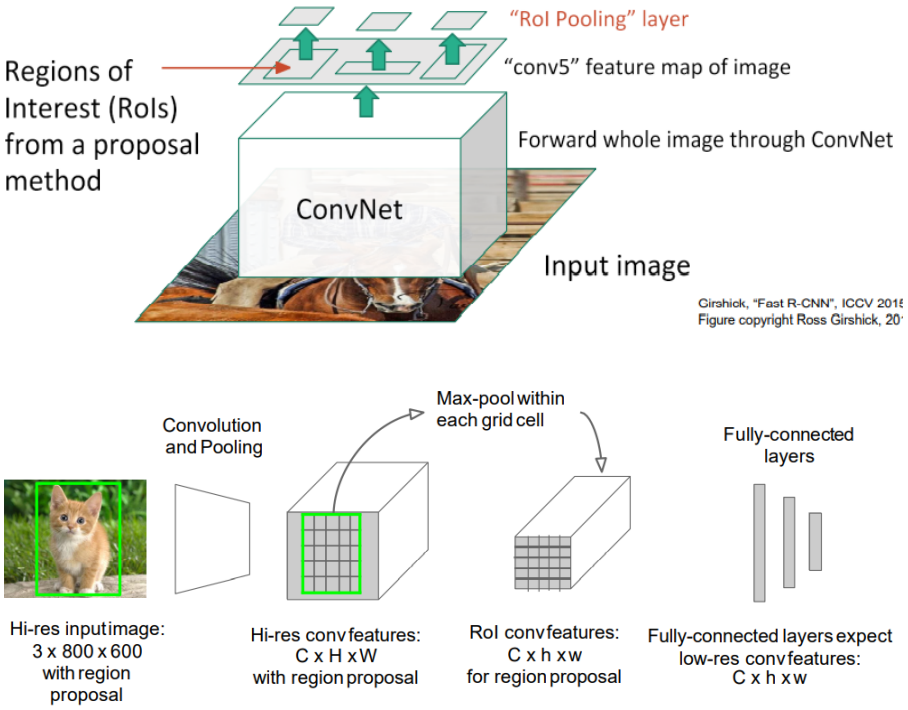
Sliding-window approach Apply a CNN to many different crops of the image, CNN classifies each crop as object or background. The problem is Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region proposals Find "blobby" image regions that are likely to contain objects, Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU. Bottom-up segmentation, merging regions at multiple scales

Region-CNN: architecture and training

- Train/download a classification model for ImageNet
- Fine-Tune model for detection, Throw away final fully-connected layer, reinitialize from scratch for object classes we want
- Extract features with region proposals for all images, For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Train one binary SVM per class to classify region features
- Bbox regression: For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for "slightly wrong" proposals

Fast-RCNN: architecture



RoI Pooling layer Given a feature map obtained from an image, with shape (C, H, W), the coordinates of a bounding box (y1, x1, y2, x2), and a desired output shape (oH, oW), divide the bounding box into a (oH, oW) grid of cells, The (i, j)-th cell covers the range $[\lfloor y1 + i * (y2 - y1 + 1)/oH \rfloor, \lceil y1 + (i + 1) * (y2 - y1 + 1)/oH \rceil]$ over the y axis
$[\lfloor x1 + i * (x2 - x1 + 1)/oH \rfloor, \lceil x1 + (i + 1) * (x2 - x1 + 1)/oW \rceil]$ over the x axis
Finally, we max pool over the (oH, oW) grid of cells, As in a normal max pooling, we take the maximum over each channel independently Therefore, we obtain a (C, oH, oW) output.

Role of bounding-box regressors Da riguardare...

Faster-RCNN and Region Proposal Network Da riguardare...

Instance segmentation: Mask-RCNN Da riguardare...

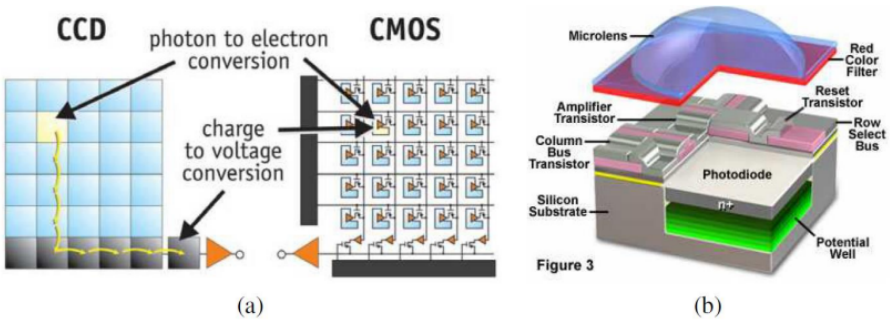
RoiAlign vs RoIPooling Da riguardare...

Detection without proposals: YOLO and SSD Da riguardare...

---

16 The camera model

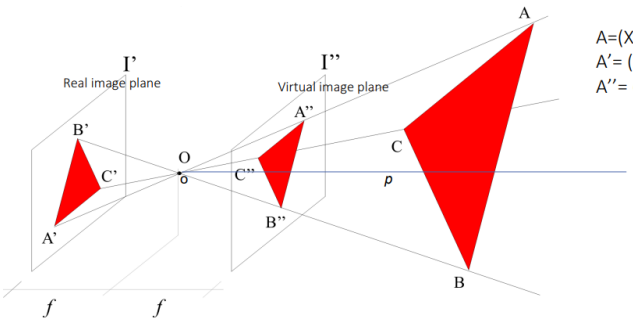
Camera ccd, cmos CCD move photogenerated charge form pixel to pixel and convert it to voltage at the output node. CMOS imagers convert charge to voltage inside each pixel.



The physics of the lights Determines the brightness of a point in the image plane as a function of illumination and surface properties (and sensor capabilities). The scene reflects radiation towards the camera and the camera senses it via chemicals on film or by electronic devices.

The pinhole model

Projective perspective in pinhole



A=(X,Y,Z)
A'= (X',Y',Z')
A''= (-X',-Y',-Z')

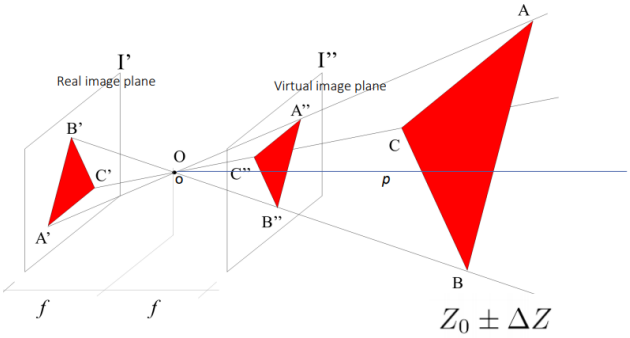
$$X' = -f\frac{X}{Z}$$
$$Y' = -f\frac{Y}{Z}$$
$$Z' = -f$$

homogeneous coordinates

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

linear projection in homogeneous coordinates!

Orthographic perspective



IF the depth of the object is negligible w.r. the distance of the object, the distance z is a constant
A=(X,Y,Z)
A'= (X',Y',Z')

$$X' = -f\frac{X}{Z_0}$$
$$Y' = -f\frac{Y}{Z_0}$$
$$Z' = -f$$

$$Z_0 \pm \Delta Z$$
$$\frac{\Delta Z}{Z_0} \ll 1$$

Perspective effects: the vanishing points The point at which receding parallel lines viewed in perspective appear to converge.

The role of the lens A lens focuses light onto the film, Rays passing through the center are not deviated, All parallel rays converge to one point on a plane located at the focal length f.

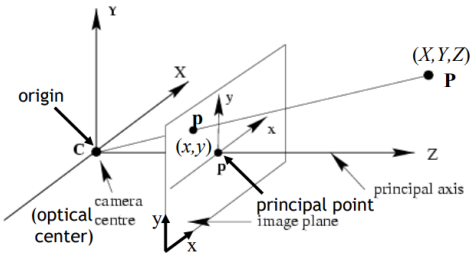
Why using thin lens Thin lens: scene points at distinct depths come in focus at different image planes.

Focus and depth of field Is the distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. The depth of field depends on the circle of confusion as a function of both the focus distance and the aperture diameter d. Depth of field depends on the lens width. Depth of field is the distance between image planes where blur is tolerable.

Field of view Angular measure of portion of 3d space seen by the camera. As f gets smaller, image becomes more wide angle more world points project onto the finite image plane, on the other hand As f gets larger, image becomes more telescopic smaller part of the world projects onto the finite image plane. Field of view depends on focal length.

Pinhole model from model 1 to complete model 4

- Model 1: 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) In homogeneous coordinates, the perspective or homographic transformation becomes linear. In the very ideal case: the H matrix is simple.



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h00 & h01 & h02 & h03 \\ h10 & h11 & h12 & h04 \\ h20 & h21 & h23 & h05 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

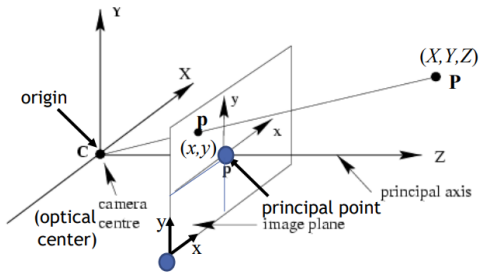
Intrinsic matrix K $\quad [\, K \mid 0 \,]$

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

(z in the image plane does not exist is 0 if the transformation is from 3D to 2D)

1. First intrinsic parameter: f

- Model 2: If the origin in the image plane is translated with respect the principal point, and the principal point has coordinates (x0, y0) in the image plane (i.e. (x0, y0, f) in the space) we must add a TRANSLATION



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & x0 & 0 \\ 0 & f & y0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2. Second pair of intrinsic parameters: x0, y0

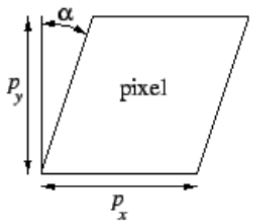
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

- Model 3: PIXELIZATION In case of non-square pixels (digital video) the focal distance has a distortion pixels are not squared and the focal length is different among x and y. two form factors of pixelization. Using in homogeneous coordinate the INTRINSIC MATRIX IS GIVEN BY FOUR PARAMETERS (which could be given by the camera, or learned by CALIBRATION)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & x0 & 0 \\ 0 & fy & y0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$fx = f\alpha x$$
$$fy = f\alpha y$$

- Model 4: The most general form considers also a 5° parameter: the skew which is the real angle between X and Y axes. Normally s is fixed to 0 (especially in new digital CMOS cameras)

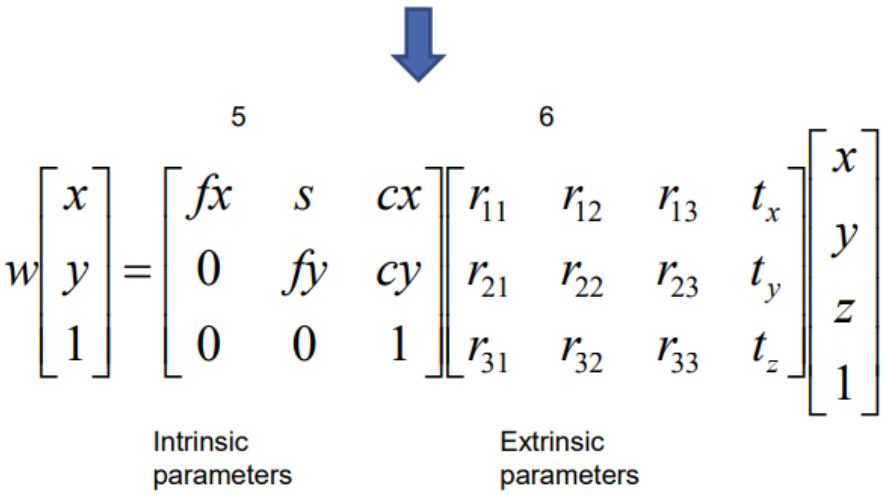


$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & s & x0 & 0 \\ 0 & fy & y0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

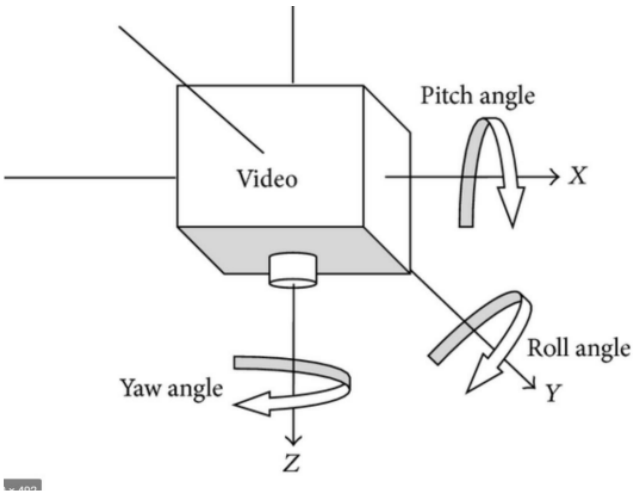
Intrinsic and Extrinsic matrix Intrinsic parameters are 5 and depending on the camera: translation of origin $(x_c y_c)$, focal length and pixelization $(f_x f_y)$ and skew s. Extrinsic parameters due to the position of the camera with reference to the real world coordinates.

Measuring real measures of the world

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$



$$w\begin{bmatrix}x\\y\\1\end{bmatrix} = \begin{bmatrix}fx & s & cx\\0 & fy & cy\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$



$$\begin{pmatrix}r_{11} & r_{12} & r_{13}\\r_{21} & r_{22} & r_{23}\\r_{31} & r_{32} & r_{33}\end{pmatrix} = \begin{pmatrix}\cos\phi & \sin\phi & 0\\-\sin\phi & \cos\phi & 0\\0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\varphi & 0 & \sin\varphi\\ & 1 & 0\\-\sin\varphi & 0 & \cos\varphi\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\0 & \cos\psi & \sin\psi\\0 & -\sin\psi & \cos\psi\end{pmatrix}$$

Camera calibration Important when we need to Reconstruct a world model: Virtual world, creating 3D reconstruction etc and Interact with the world : Robot, hand-eye coordination. Knowing the real world dimension by single images we can calibrate cameras. In mobile we need also the IMU inertial measurement unit calibration.

Methods of camera calibration

- Multiple images: the chessboard method extract corners. Very precise require multiple image manual calibration, difficult to use for wide DoG
- Using different geometric structure of the scene: lines, vanishing points, selecting some existing lines, only for man-made scenes (single image metrology)

- Camera self-calibration to extimate intrinsic parameters with a moving camera on a video; it solve calibration also with distortion parameters using epipolar lines (see Firzgibbon Simultaneous Linear Estimation of Multiple View Geometry and Lens Distortion. In CVPR. 2001)
- Deep learning (DeepFocal [Workman et al. 2015] predicts only the focal length)

Zhang method of calibration We need Ni images with M corners each [M ¿ 3 and Ni ¿= 2/(M-3)]

- Step 1: data acquisition
- Step 2: specify corner order
- Step 3: corner extraction
- Step 4: minimize projection error and camera calibration

Structure from motion Given many points in correspondence across several images, (u_{ij}, v_{ij}), simultaneously compute the 3D location x_i and camera (or motion) parameters (K, R_j, t_j)

Lent distortion radial and tangential definitions Radial distortion of the image Caused by imperfect lenses Deviations are most noticeable for rays that pass through the edge of the lens. Barrel distortion is from Wide Angle Lens and Pin cushion form telephoto lens (inverse). Radial distortion L(r) depending on radius r.Tangential distortion dx, dy.

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

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

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

Radial Distortion:

$$\hat{x}_c = x_c(1 + k_1 r_c^2 + k_2 r_c^4) \hat{y}_c = y_c(1 + k_1 r_c^2 + k_2 r_c^4)$$

Tangential Distortion:

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

Distortion parameter computation

- Project (X, Y, Z) to normalized image coordinates

$$x_n = \frac{X}{Z} \quad y_n = \frac{Y}{Z}$$

- Apply radial distortion

$$r^2 = x_n^2 + y_n^2 x_d = x_n(1 + k_1 r^2 + k_2 r^4) y_d = y_n(1 + k_1 r^2 + k_2 r^4)$$

- Apply focal length translate image center

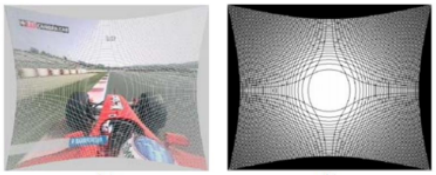
$$x = f x_d + x_c y = f y_d + y_c$$

Simple correction: given a pd(xd, yd) distorted pixel and pu=(xu, yu) undistorted pixel

$$x_u = c_x + (x_d - c_x)(1 + k_1 r_d^2 + k_2 r_d^4 + \ldots)$$
$$y_u = c_y + (y_d - c_y)(1 + k_1 r_d^2 + k_2 r_d^4 + \ldots)$$

It corrects the 90%

$$x_u = x_d + (x_d - c_x)(k_1 r_d^2)$$
$$y_u = y_d + (y_d - c_y)(k_1 r_d^2)$$



(a) (b)

Fig. 1. Example of incomplete mapping on the undistorted image. (b) shows the locus of points for which the coordinates are defined.

And using some approximations ru computed from the center distance of (xu, yu)

$$x_d = c_x + (x_u - c_x)\frac{r_d}{r_u}$$
$$y_d = c_y + (y_u - c_y)\frac{r_d}{r_u}$$

Spherical method for camera modeling

Vanishing points An image may have more than one vanishing point in fact every pixel is a potential vanishing point, each Different directions correspond to different vanishing points. Vanishing points can be also outsides the image. Any set of parallel lines on the plane defines a vanishing point. The union of all of vanishing points from lines on the same plane is the vanishing line For the lines in the ground plane, the vanishing line is called the horizon

Computing distance from single image To compute the distance in the real 3D wold we must know the perspective transformation We need to know the distance between image plane and camera center . i.e. focal length. For instance we have to do the Lens Distortion correction: lens correction (see in the following) then We must select 4 points that are in the ground plane [X,Y,Z]. If vanishing points are extracted some correspondence in lengths can be found and some measures can be computed.

---

17 Geometrical transforms and Camera Calibration

Geometrical transforms recap Act by changing the spatial location of points. Given a point x from the source image, a geometric transform is a function which transforms it into another point, $f(x) = Tx$, where T is the transformation matrix and x is in homogeneous coordinates. The two "most important" classes of transforms: affine (6 DoF), perspective (8 DoF). Images are discrete and quantized: the forward mapping is not enough. We need to use the reverse mapping, either by inverting the transformation, or by employing an interpolation approach.

Estimating transforms in OpenCV

- Affine transforms (2x3 matrix): cv2.getAffineTransform(src, dst)
- Perspective trasforms (3x3 matrix): cv2.getPerspectiveTransform(src, dst)
- Applying trasforms: cv2.warpAffine(), cv2.warpPerspective()

Applying geom. transforms in OpenCV, warping

- Affine transforms: cv2.estimateRigidTransform(src, dst, fullAffine) Finds the optimal affine transform given two point sets, using a least square objective
- Perspective transforms: cv2.findHomography(...) Does the same in the case of a perspective transform.

Implementation of Image Stitching Algorithm

Camera Calibration Camera calibration is the process of estimating the camera intrinsic and extrinsic parameters. To calibrate a camera you must Using a checkboard, acquire multiple images under different orientations and distances. Be careful to attach the checkboard to a rigid surface: folds or deformations in the checkboard may compromise the calibration process.

Camera Calibration in OpenCV

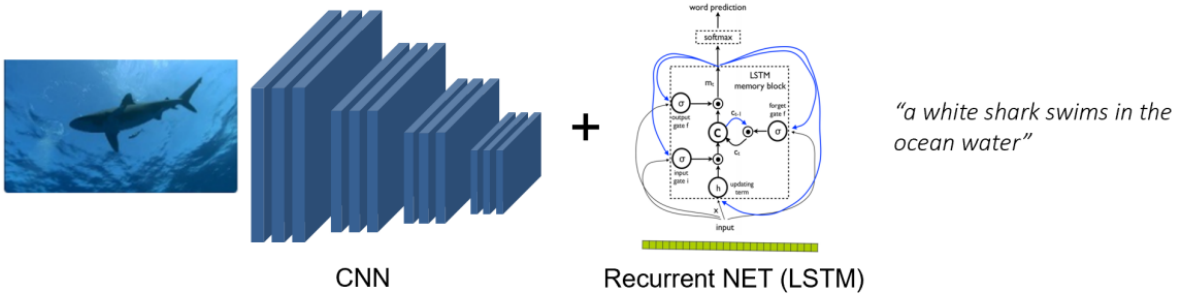
- Given different views of the board, detect the internal corners using cv2.findChessboardCorners(). At this point, we have the corners coordinates on the image plane for each view
- Ignoring the size of the board cells, we could say those points are (0,0), (1,0), (2,0), ... $\rightarrow$ in this case, the results we get will be in the scale of size of chess board square. But if we know the square size, (say 30 mm), and we can pass the values as (0,0),(30,0),(60,0),..., we get the results in mm.
- Now we have our object points and image points and we are ready to go for calibration. Use the function, cv2.calibrateCamera(). It returns the camera matrix, distortion coefficients, rotation and translation vectors.

---

18 Computer Vision and Neuroscience, segmentation, saliency

Vision in humans and AI Connections between computer vision and perceptive psychologies and cognitive science (we, as scientists, learned a lot from the human visual behavior) Cognitive science = connected with the mind. Connections between computer vision and neuroscience (we, as scientists, learned a lot from the structure of the neuron, the cortex and the inner brain) Neuroscience = connected with the brain. Both cognitive psychologists and neurosciencies analyze and try to understand our vision process and the visual descriptors we use for specific cognitive tasks. The neuroscience aims at understanding how information sensed by eyes become vision, how thoughts become memory, how behavior comes from biology

Example of comparison for memory (lstms) RNNs (LSTMs) are perfectly adopted for image captioning and generate textual description. A complex cognitive system: perceives, creates a neural representation, uses memory and generates



Base technical idea:

- Use CNN as feature extractor and RNNs as language models

The study of memory in Kahneman

The study of memory in perceptual psychology

- Daniel Kahneman, Nobel Prize 2002

System 1

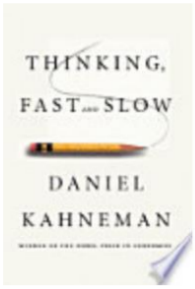
- FAST
- DEFINING CHARACTERISTICS: unconscious, automatic, effortless
- WITHOUT self-awareness or control "What you see is all there is."
- ROLE: Assesses the situation, delivers updates
- Makes **98%** of all our thinking

System 2

- SLOW
- DEFINING CHARACTERISTICS: deliberate and conscious, effortful, controlled mental process, rational thinking
- WITH self-awareness or control, logical and skeptical
- ROLE: seeks new/missing information, makes decisions
- Makes **2%** if all our thinking

https://www.youtube.com/watch?v=PirFrDVRBo4&t=148s

https://www.youtube.com/watch?v=uqXVAo7dVRU



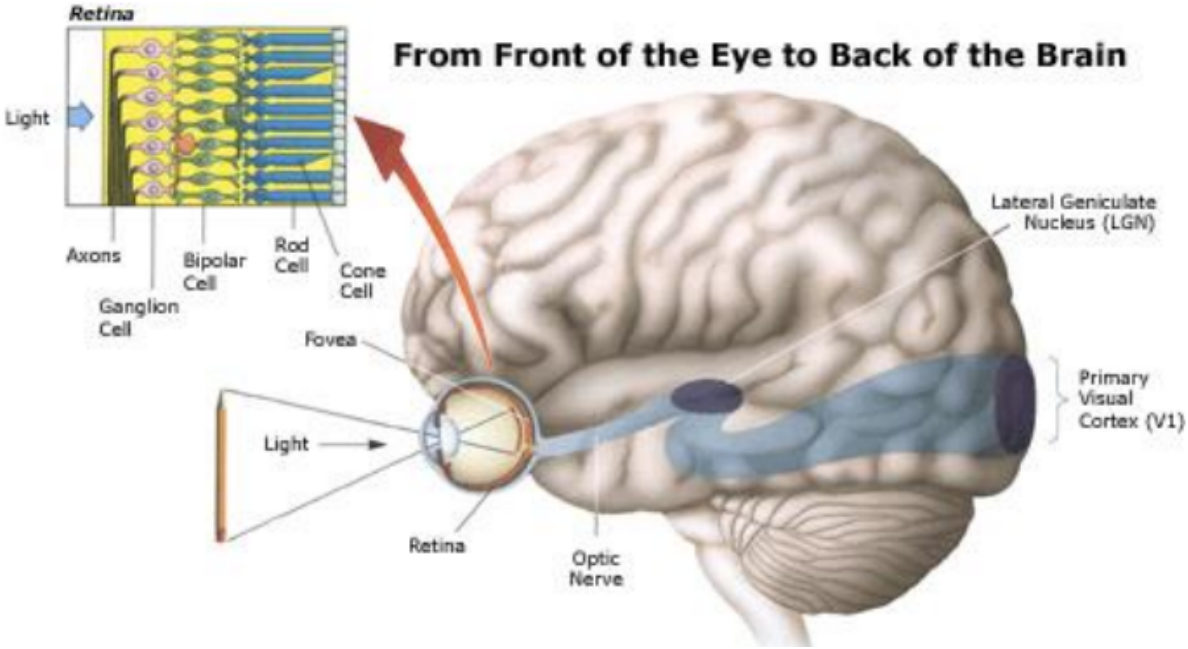
The study of reductionism in Kandel Kendal studies two types of memories. One is complex form of memory, which require the hippocampus and (are called) explicit memory storage. One is simple: the very simple form like driving a car – once you know how to do it, you do it automatically – we call that implicit memory storage. And the two involve different systems in the brain. Reductionism: studies the simplest cases.

The eye, the retina, the optical nerve, the cortex

- Eyes are two sensors
- Retina is the image plane
- Optical nerve is a wired connection
- Cortex is the low and intermediate vision system, the inference module and the actuator control

Visual activities in the cortex The frontal lobe is largely concerned with planning future action and with the control of movement; the parietal lobe with somatic sensation, with forming a body image, and with relating one's body image with extrapersonal space; (the where). The occipital lobe with vision; the temporal lobe with hearing; (and the what) and through its deep structures—the hippocampus and the amygdaloid nuclei—with aspects of learning, memory, and emotion.

Human vision pathway



Top down vision and recognition

- Neglect details that are not relevant in a context
- Search for the constancy
- Try to abstract the essential features that are constant and discriminant between objects
- Compare images and information of the present with "images" and information of the past

The problem of segmentation: definition that is the task of distinguish between the "object" (in this case the portion or region of the image) and the background. After detecting the precise shape and position of the object many measure in 2D and 2D can be computed, e.g. dimensions, centroid, orientation etc. This is perceptual segmentation without semantics.

Perceptual vs Semantic vs Instance segmentation Perceptual Segmentation is a problem of grouping. It could be a direct application of cluster analysis, where the homogeneity predicate is given by perceptual, cognitive cues.
Semantic Segmentation is a classification problem, where each pixel is classified according with a label then regions are merged together. Clustering, grouping, classifying.. Now a work for deep learning (with enough labelled data). It is a supervised learning.
Instance segmentation is a semantic segmentation where regions of difference instances, although adjacent are separated. It can be achieved by semantic segmentation followed by labeling, either by an end-to end network. It needs a prior knowledge.

Approaches for segmentation Segmentation consists in partitioning images into sets of homogeneous points called Regions in the image (often correspondent to real objects in 3D world), w.r.t. a given visual feature. Often a Region is called object when semantics can be associated with it. Segmentation: Given an image I and a predicate P (homogeneity

criterion, based on a visual feature) image segmentation means finding a partition S of I in a set of regions R1... Rn.
Thresholding: the first «segmentation operator», is the task to find (or give) a solution of a problem of transforming a gray level image into a binary image. Thresholding becomes a segmentation operator is the semantic associated with the perception is in such a way preserved or enhanced. The fist image has many gray levels buy we would be able to see only two levels: black (the target object) and white (the background).

Otsu thresholding The optimum threshold is that threshold which minimizes the within group variance thus maximizing the homogeneity of each group. Algotihm:

- Compute for a given threshold t: the prior probabilities that pixels belongs to group 1 and 2 where P(i) is the posterior probability computed by normalized histogram.

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

- Compute mean and variance for each group.

$$\mu_1(t) = \sum_{i=1}^{t} iP(i)/q_1(t) \quad \sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 P(i)/q_1(t)$$

$$\mu_2(t) = \sum_{i=1}^{t} iP(i)/q_2(t) \quad \sigma_2^2(t) = \sum_{i=1}^{t} [i - \mu_2(t)]^2 P(i)/q_2(t)$$

- Compute the within group variance

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

- Minimization: for each t = 1... L, search the minimum of within group variance

Adaptive thresholding Instead of computing T on the whole image, compute T only in a window W(i,j) ; the dimension of the windows depends on the problem and the variability of data. Algorithm:

- Initial threshold T is chosen, randomly or according to any other method
- The image is segmented into object and background pixels as described above, creating two sets (object pixels and background pixels)
- The average of each set is computed.
- A new threshold is created that is the average of m1 and m2
- 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)

Segmentation by Clustering This iterative algorithm is a special one-dimensional case of the k-means clustering algorithm, which has been proven to converge at a local minimum—meaning that a different initial threshold may give a different final result. A nearest neighbour clustering algorithm allows us to perform a grey level segmentation using clustering, A simple case of a more general and widely used K-means clustering, A simple iterative

algorithm which has known convergence properties. Repeat: Compute the centers of classes m1 and m2 and Redefine the two groups so that 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.

Adjacency, definition of 4/8 connection The Connection (adjacency) propriety is related with the concept of distance, which is defined in 2D image space but we can extend it to the 3D space and to the 4D space in the time too.

- Distance Euclidean: $De(i,j)(h,k) = \sqrt{(i-h)^2 + (j-k)^2}$
- Distance City Block: the minimum number of steps in a grid to reach a point from another one $D4(i,j(h,k) = |i-h| + |j-k|$
- Distance Chessboard: number of steps for a king in a chessboard $D8(i,j)(h,k) = max|i-h|, |j-k|$

Connected component labeling: methods

- Connected Components Labeling (CCL): transforms a binary input image into a symbolic one in which all pixels belonging to the same connected component are given the same label
- Pixel adjacency (connection in 2D) : Two pixels are adjacent if their distance is equal to the discretization range of the image
- Connection Between Pixels: 4-neighborhood and 8-neighborhood

$$N_4(p) = q \in L|p_x - q_x| + |p_y - q_y| \leqslant 1$$

$$N_8(p) = q \in L|max(|p_x - q_x|, |p_y - q_y|) \leqslant 1$$

Iterative algorithm (Multiscan) Algorithm

- Initialization: each pixel of the image (where pixel is 1) is numbered with a consecutive label
- Top-down: a scan from high to low from left to rigth to give the same label to connected pixels
- Botton-up: a scan from low to high from rigth to left to give the same label to connected pixels
- Repeat steps 2 and 3 while at least a label changing is done.

2 Step labeling Algorithm

- Whenever more than a label can be assigned to a pixel the lowest is assigned and the others are collected in an equivalence table.
- The table is scanned and ordered until all equivalences are solved. [Riguardare le Equivalence table nel dettaglio]

Decision trees and decision tables for labeling The Block-Based with Decision Trees (BBDT) algorithm employs a new scanning technique that moves on a 2×2 pixel grid over the image which is optimized by an automatically generated decision tree.

Decision tables are a concise visual representation for specifying which actions to perform depending on given conditions. They are algorithms whose output is a set of actions. The information expressed in decision tables could also be represented as decision trees or in a programming language as a series of if-then-else and switch-case statements.

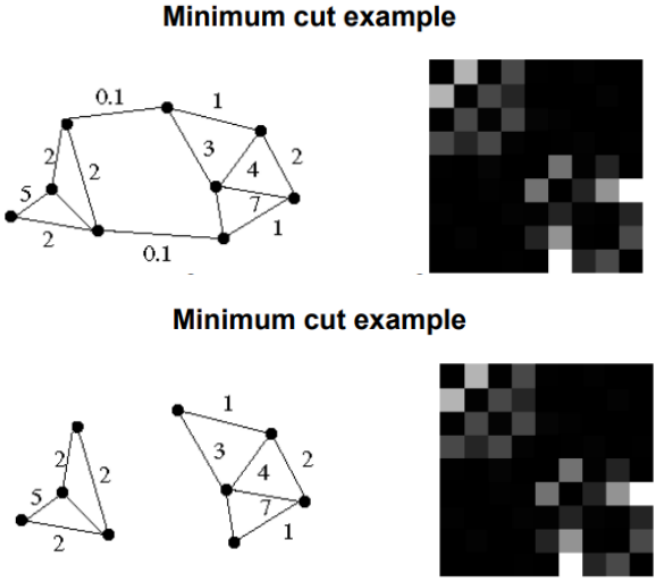
Segmentation by foreground and background: methods

- Clustering methods into two classes: Region splitting (divisive clustering), Region merging (agglomerative/recursive clustering), Superpixel based methods semantic agnostic
- K-means based: Pure k-means clustering and Mixture of Gaussian (parametric model) and Mean-shift (non-parametric model)
- Graph based methods: normalized cut and grabcut

Mean shift Useful for F/B clustering but also for multiple region segmentation, Used also for tracking in the time.

- Choose kernel and bandwidth
- For each point center a window on that point, compute the mean of the data in the search window, center the search windows at the new mean location and repeat pass b and c until convergence
- Assign points that lead to nearby modes to the same cluster

Graph based methods: mincut and normalized cut Fully-connected graph: node for every pixel, link between every pair of pixels, p,q, similarity w_{ij} for each link.
- Min cut has the drawback to separate singleton or small groups.



- Normalized Cut (NP complete): a cut penalizes large segments, fix by normalizing for size of segments.

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

where Association is the sum of costs of all edges that touch A or B; Cut(A, B) is the sum of all weights from A nodes to B node.

Grab cut with graph cut [Da riguardare]

- Define graph (usually 4-connected, sometime 8-connected)
- Define unary potentials (color histogram or mixture of Gaussian's for background and foreground)
- Define pairwise potentials
- Apply graph cuts
- Return to 2, using current labels to compute foreground, background models

Often grabcut uses 5-10 iterations, Then there is a refinement in the edges.

Superpixels by SLIC Superpixel could be defined as a group of pixels that share common/similar characteristics. It is generally color based segmentation. 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.
SLIC Simple Linear Iterative Clustering is one of the most famous algorithm to segment superpixels which doesn't require much computational power. The algorithm clusters pixels in the combined five-dimensional color and image plane space to efficiently generate compact, nearly uniform superpixels. 5 dimensions are defined by: the L, a, b values of the CIELAB colorspace and x, y coordinates of the pixels. SLIC takes a desired number of approximately equally-sized superpixels K as input. So each superpixels will have approximately N/K pixels. Hence, for equally sized superpixels, there would be a superpixel center at every grid interval $S = \sqrt{(N/K)}$. K superpixel cluster centers $C_k = [I_k, a_k, b_k, x_k, y_k]$ with k = [1, K] at regular grid intervals S are chosen.

Distance measure D_s is defined as follows.

$$d_{lab} = \sqrt{((l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2)}$$

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

$$D_s = d_{lab} + (m / S)* d_{xy}$$

Algorithm:

Algorithm 1 Efficient superpixel segmentation

1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
3: **repeat**
4: **for** each cluster center C_k **do**
5: Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure (Eq. 1).
6: **end for**
7: Compute new cluster centers and residual error E {$L1$ distance between previous centers and recomputed centers}
8: **until** $E \leq$ threshold
9: Enforce connectivity.

Pixel-Superpixel association: Associate each pixel to the nearest superpixel center in the five-dimensional space, i.e., compute the new superpixel assignment at each pixel p,

$$H_p^t = \underset{i \in \{0,\ldots,m-1\}}{\arg\min} \; D(I_p, S_i^{t-1}), \tag{1}$$

$S_i^t = \frac{1}{Z_i^t} \sum_{p|H_p^t=i} I_p,$ Zit is K at the iteration t
Sit is the centroid Ck at the iteration t

Superpixel by deep learning

Morphology

Saliency definition Saliency detection in humans is a key attentional mechanism that facilitates learning and survival by enabling organisms to focus their limited perceptual and cognitive resources on the most pertinent subset of the available sensory data. Two forms of attention: Initial, Bottom-up purely data driven, guided by Saliency and Refined, Task-driven and purposive.
Saliency map: computes visually conspicuous points based on low-level visual features such as brightness, colour, oriented edges and motion.
Priority map: the goal-directed 'map which integrates information from the bottom-up saliency map with task- and goal-relevant information.
A salience computational model describes how low-level exogenous visual features such as colour, orientation, luminance and motion are combined into a single global map representing the relative 'salience' of each point on the map.

Saliency by deep learning SAM approach To reduce the number of parameters and facilitate the learning, we constraint each prior to be a 2d Gaussian function, whose mean and covariance matrix are instead freely learnable. This lets the network learn its own priors purely from data, without relying on assumptions from biological studies.The priori central bias is integrated in a single end-to-end pipeline: 512 channel + 16 channel of Gaussian learned – a priori and 528 channels than convolved in a single Saliency map. The loss function is a combination of three measures:
- L1 Normalized Scanpath Saliency NSS
- L2 Linear Pearson's Correlation Coefficient CC
- L3 Kullback-Leiber divergence, KLD

Comparison of saliency and task based attention

- Botton Up Saliency: Bottom-up Data driven + knowledge based (high level features):
  ML-NET: VGG convolutive structure for images with multy-layer and bias features
  SAM: Improvement with LSTM , emulating scan-path refinement
  ML-NET and SAM useful for video too if motion is not too strong

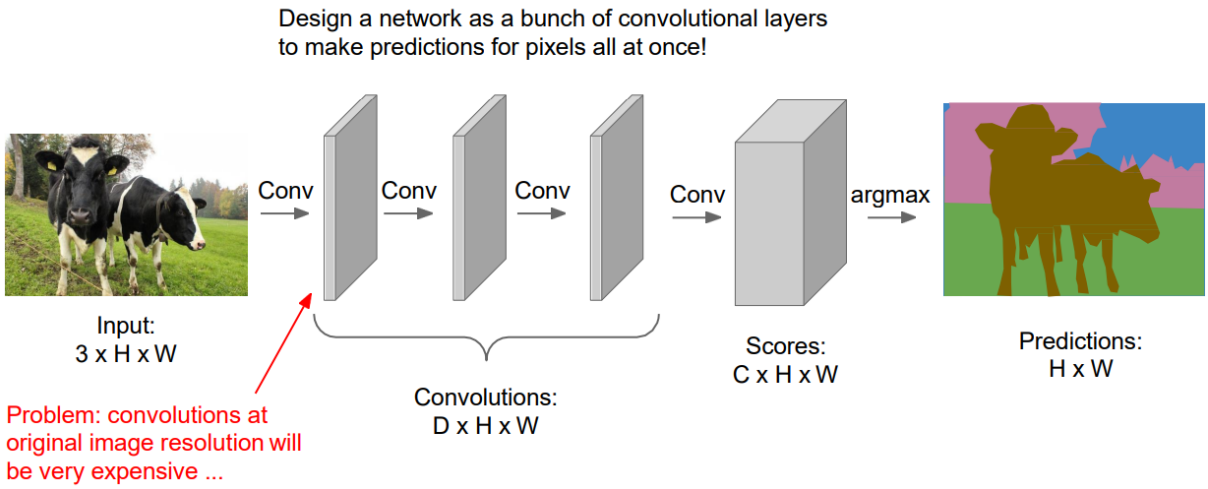
- Top-down Task-driven for driving: DR(EYE)VE C3D autoencoder with cropping areas

to observe every-where
Refinements convolutive to fit the learned models driving
Architecture extremely efficient in term of human replication; very large and time consuming

---

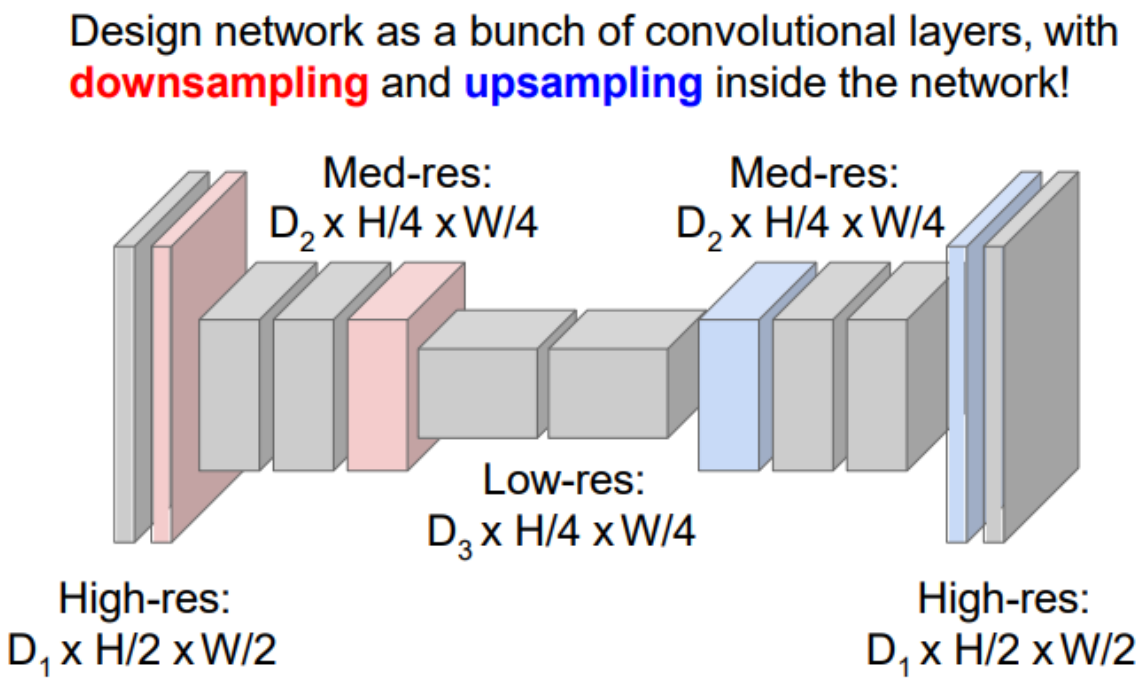
19 Segmentation architectures

Fully Convolutional Networks

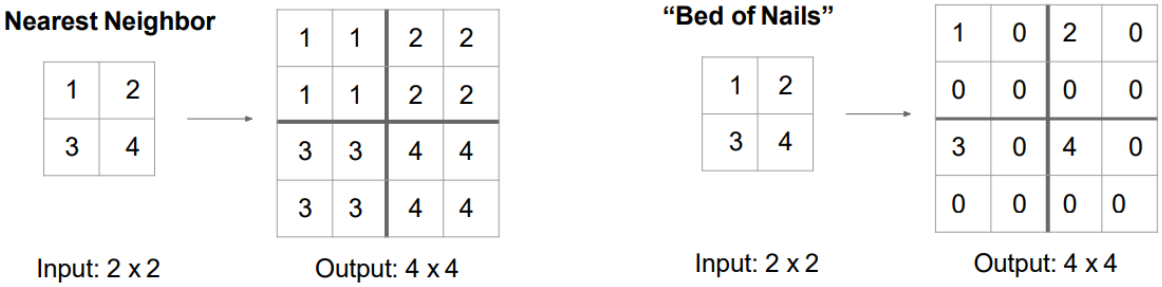


Downsampling/Upsampling protocol

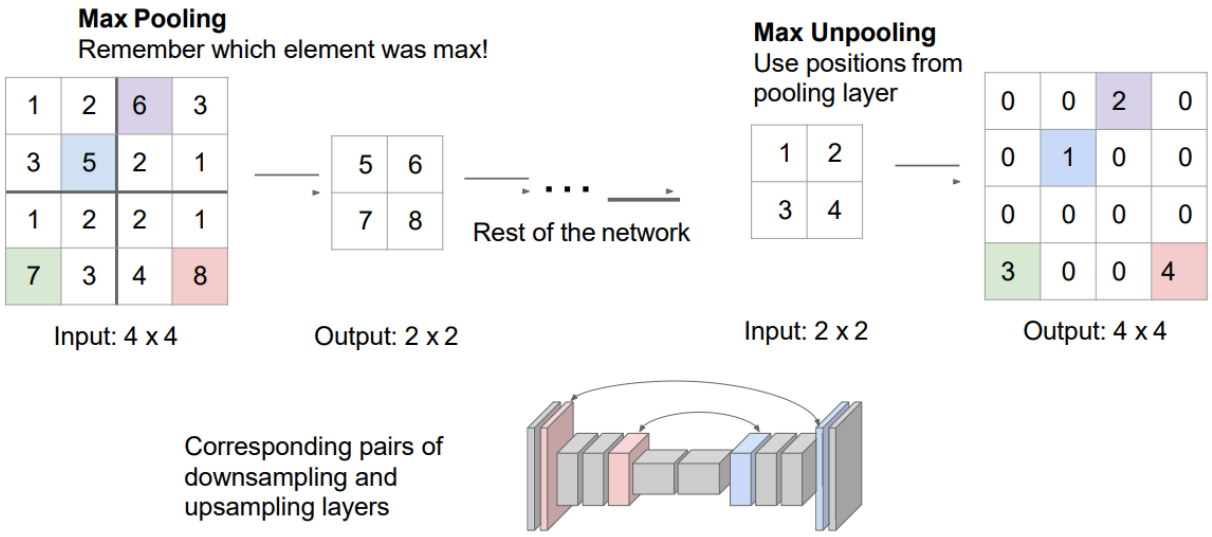
- Downsampling: Pooling, strided convolution
- Upsampling



Upsampling techniques: Unpooling (NN + Bed of Nails)



Upsampling techniques: Max Unpooling



Transpose Convolution: input/output shapes

- Input shape: n (images), iC, H, W
- Kernel shape: iC, oC, kH, kW (oC is bias shape)
- Output shape: n, oC, oH, oW
  With: $oH = (H-1) * stride[0] - 2 * padding[0] + dilation[0] * (kH-1) + 1$
  $oW = (W-1) * stride[1] - 2 * padding[1] + dilation[1] * (kW-1) + 1$

SegNet, UNet

- Seg-Net: Encoder-Decoder framework, Use dilated convolutions, a convolutional layer for dense predictions. Propose 'context module' which uses dilated convolutions for multi scale aggregation. Uses a novel technique to upsample encoder output which involves storing the max-pooling indices used in pooling layer. This gives reasonably good performance and is space efficient (versus FCN)
- UNet: Convolutional Networks for Biomedical Image Segmentation Encoder-decoder architecture. When desired output should include localization, i.e., a class label is supposed to be assigned to each pixel Training in patches helps with lack of data.

20 Motion

Definition of Video (dimensions of raw video) VIDEO IS NOT ONLY A SET OF IMAGES, Video is a SEQUENCE of images (frames) that is an ordered set of frames acquired in a sequence of instants of time t_K. With $D_t = t_{k+1} - t_k$ costant for k=0...n-1. Dt is not always constant; frames can be skipped or adjusted or added the acquisition or the network can modify the sequence. We suppose that Dt is constant and a priori known.

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

Motion sensors (examples)

- Passive Infrared (PIR): small, low power, easy to use, and inexpensive. The way it senses movement is by sensing the change in temperature between the background and a warm body. IRs have a pyroelectric sensor that detects levels of infrared radiation – everything emits some low-level radiation, but a human body emits a good amount of heat.
- Microwave sensors: uses continuous waves of microwave radiation to detect motion, similar to how a radar speed gun works. It sends out high radio frequency and measures the reflection off an object by sensing for a frequency shift. If it does detect a frequency shift, the motion detector is activated.
- Dual Tech/Hybrid: use both

Types of motion analysis

- From fixed cameras
- From cameras with constrained motion
- From moving or unknown cameras
- From constrained but unknown egocentric motion

There are 3 type of application (motion in computer vision):

- Moving point estimation: sparse and dense motion estimation, motion of undefined objects (e.g. cloud) point movement detection, shot detection speed of background, foreground points and camera motion SFM structure from motion
- Moving object detection: estimating motion of background and foreground regions detecting shapes in motion in 2D and 3D understanding objects, people, actions by motion cue
- Estimating motion of detected objects: spatio and temporal coherency analysis tracking single and multiple targets tracking objects with multiple point of views reidentifying moving objects (re-id)

Tracking Definitions and challenges Tracking is one of the most challenging computer vision problems, concerning the task of generating an inference about the motion of an object given a sequence of images. Tracking is the analysis of video sequences for the purpose of establishing the location of the target over a sequence of frames (time) give temporal coherence

of an object. Tracking looks at the past, prediction at the future. Tracking assumes (and includes) objected detection and localization and motion prediction. Challenges are: Single object tracking (SO-T on-line, off line), Multiple object tracking (MO-T or MTT: Multiple target tracking), Multiple Camera Multiple Target tracking MCMT T(with overlapped or not overlapped FOW) and Multitarget multisensory tracking.

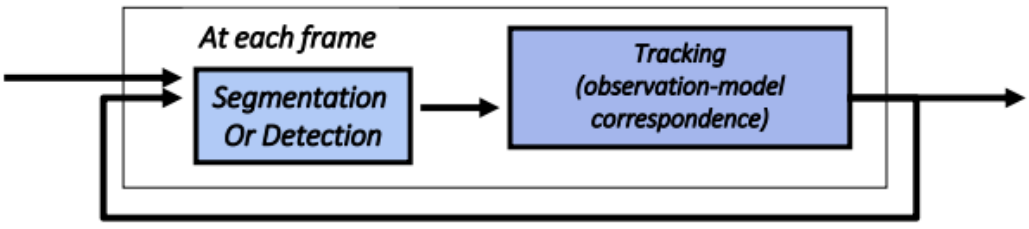
Measures of tracking (fscore, f1 score, ota, otp, deviation, mota, motp)

- F-score/Correct track ratio: $F = 2\frac{Precision \quad * \quad Recall}{Precision \quad + \quad Recall}$
- F1-score/Correct track ratio: $F1 = \frac{1}{Nframe}\sum_{i=1}^{N} frame 2\frac{P^i * R_i}{P^i + R^i}$
- OTA (object track accuracy): $OTA = 1 - \frac{\sum_{i=1}^{Nframe}(n_{fp}^i + n_{fn}^i)}{\sum_{i=1}^{Nframe} g^i}$
- OTP (precision at pixel level): $OTP = \frac{1}{|Mi|}\sum_{i}^{inMi}\frac{|T^i \cup GT^i|}{|T^i \cap GT^i|}$
- MOTA (Multiple object tracking accuracy): gives a very intuitive measure of the tracker's performance at detecting objects and keeping their trajectories, independent of the precision with which the object locations are estimated. $MOTA = 1 - \frac{\sum_{i=1}^{Nframe}(n_{fp}^i + n_{fn}^i + n_{fa}^i)}{\sum_{i=1}^{Nframe} g^i}$
- MOTP (Multiple object tracking precision): sum of the distances between GT and Tracks w.r.t. the correct match at time t (Ct); it is the error in the estimated positions; It shows the ability of the tracker to estimate precise object positions, independent of its skill at recognizing object configurations, keeping consistent trajectories, and so forth. $MOTP = 1 - \sum_{i \in Mi}\frac{|CT^i - CGT^i|}{|Mi|}$

Gi is the number of ground truth objects in the frame i (Gi = ntp + nfp) that is 1 in the frames where the object is present, to normalize OTA. Mi is the frame where there is a matching. n_{fa} is identity switches.

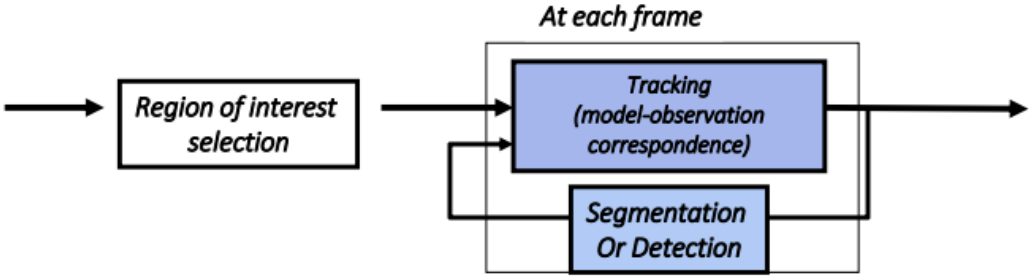
Tracking by detection, detection-by-tracking

- Tracking by detection: when detection is "easy"; it is simpler, well conditioned; use deterministic issues or more often Statistical model, Typical for multiple object. Detection by segmentation with motion features or (motion + x), with object detection or instance segmentation. Object Status computation Status, with position, speed in 2D or 3D, Appearance A. Tracking as Prediction – if needed, Measure (matching), Status correction or update.



- Detection-by-tracking: General and more challenging (and lower speed), often off-line, often for a single complex object. Initialization (manual or automatic computation of ROI), Search with or without prediction and status computation, Measure, matching

and Segmentation.



Schema of tracking

- Region of interest detected at the first slide or given
- Representation: how to observe invariant and variant features in the frame and how to hold them in an internal representation (motion and position)
- Inference Method (similarity measuring, matching and optimization)
- Model Update

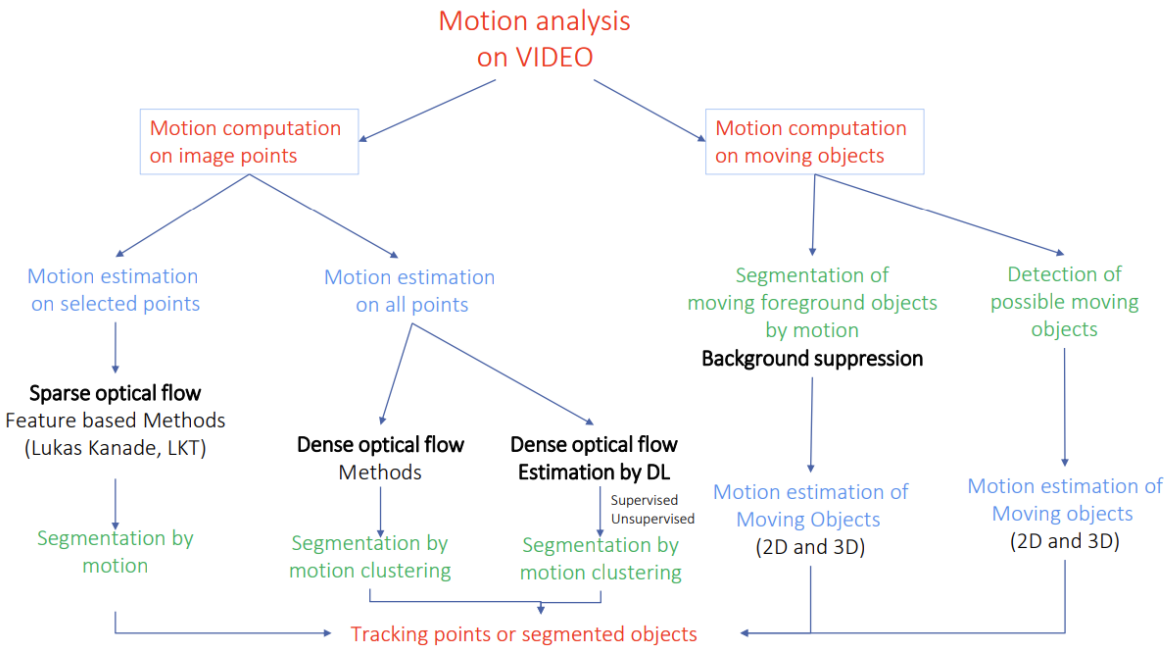
NCC (Normalized cross correlation) for tracking The only assumption is that: Shape does not change (enough), Motion is limited and a neighborhood analysis is enough. Template Matching: brute force method for tracking single objects and allow a MOTION ESTIMATION. Algorithm:

- Define search area
- 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.
- Select the best candidate with the maximal similarity measure, can be a direct template comparison or statistical measures between two probability densities

$$E_{NCC} = \frac{\sum_i [I_0(x_i) - \bar{I}_0][I_1(x_i + u) - \bar{I}_1]}{\sqrt{\sum_i [I_0(x_i) - \bar{I}_0]^2} \sqrt{\sum_i [I_1(x_i + u) - \bar{I}_1]^2}}$$

- No update of the model

Motion analysis on Video, a schema



Motion, motion field, optical flow

- Motion: of points, objects, cameras etc
- Motion Field: projection of 3D scene motion in a plane (the image plane)
- Optical Flow: what can be computed visually from images

Motion measures

- Angular Error: $AE = arccos((u_o, v_0) * (u_1, v_1))$
- Endpoint Error: $EPE = \sqrt{(u_0 - u_1)^2 + (v_0 - v_1)^2}$

Motion field and parallax Motion field: is the motion projection in the image plane. Is a vector create by $v_x = dx/dt \quad v_y = dy/dt$ where

- p(t) = (x(t), y(t)) is the projection of P in the image
- V = dP / dt is the velocity of scene point
- P(t) is a moving 3D point

We can also define $p = f\frac{P}{Z}$ and $v = f\frac{ZV - V_z P}{Z^2} = \frac{1}{Z}(v_0 - V_z P)$ and vector can be expressed as $v_x = \frac{fV_x - V_z x}{Z}$ $v_y = \frac{fV_y - V_z y}{Z}$ and $v_0 = (fV_x, fV_y)$
Motion parallax: is a depth cue that our minds associates with yjr movement. We see objects that are closer to us as moving faster than objects that are in the distance. We are able to see the full speed motion of the close items, but distant objects seem to be moving at a slower rate.

Motion estimation methods I (direct, sparse)

- Direct methods (dense OF): Directly recover image motion at each pixel from spatio-temporal image brightness variations Dense motion fields, but sensitive to appearance variations. Suitable for video and when image motion is small. The direct methods (or gradient based) compute the optical flow by analyzing the luminosity variation (gradient) in the space and in the time; some variatonal extensions are added.

- Feature-based methods (sparse OF or estimation of Dense): Extract visual features (corners, textured areas) and track them over multiple frames. Sparse motion fields, but more robust tracking, Suitable when image motion is large (10s of pixels). The second methods or discrete correspondences (matching) produce sparse maps and search for optical points only in those points that can be considered visually constant (EG a corner). Extract global compressed features and reconstruct the motion with supervised learning (by synthetic data) dense optical flow (Qualitative only).

Brightness constancy equations (and constraints) The intensity of the pixel is constant in the time, is the tendency for objects to maintain their perceived brightness.

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

and using Taylor of the first order with $\frac{dI}{dt} = 0$, we have $\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$ we have Brightness Constancy Equation:

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

Constraints:

- Small motion or Max speed: points can move of a given max speed which corresponds to a max shift $rmax = cmax\Delta t$
- Small acceleration: acceleration is considered negligible; this is acceptable if Δt is small enough
- Uniform movement: all points of the objects are moving in the same direction (no rotation)
- Spatial coherence: the brightness I(x,y,t) depends only on the coordinates (x,y) and points move like their neighbors
- Rigid body: the shape of the body is supposed not to change in the time (so that dx and dy are constant) (additional constraint)
- Smoothness: flow field should be smooth over the interior of objects in the scene
- Sharpness: Flow should be sharp at object boundaries
- Alignment: Flow map should be well aligned with object boundaries
- Stability: It should be temporally stable over frame sequences

Aperture problem Only the speed in the direction of the gradient can be defined, The speed in the direction perpendicular to the gradient is unknown.

$$-\frac{I_t}{\sqrt{I_x^2 + I_y^2}}$$

Lucas Kanade method for OF computation Hp: the speed is constant for a given neighborhood of the pixel; Spatial coherence constraint (the 5° constraint), u and v are constant in a neighborhood (es 5x5);

- The temporal gradient can be computed with the backward derivative.

- The spatial gradient can be computed with Sobel
- Representing it in the vector form in a linear system
- Compute d = (u,v) and minimize $||Ad - b||^2$, the solution is given by

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

Other methods for dense estimation Horn Schunck accepts a small variation of the motion in a neighborhood but with a smoothness constancy. Iterative method where the u and v components are computed with a regularization constant for the smoothness or continuity constraint. At each iteration you can use the average values obtained in The previous iteration.

OF methods for sparse estimation

Segmentation by optical-flow Consist in Extract regions of interest by thresholding magnitude of motion vectors, For each connected region, perform k-means clustering using feature vector: $[v_x, v_y, x, y, R, G, B]$, Color intensities (RGB) give information on object boundaries to counter the smoothing of motion vectors across edges in optic flow estimate, Remove small, isolated regions.

Extension to affine OF

OF by Deep Learning Flow net and Flownet2 Flownet: Contracting part extracts a rich feature representation, Expanding part creates the optical flow in color coordinates. Flownet2: It puts together the flownetcorr (First process the images separately, and then correlate their features at different locations and process further, Two Separate, yet identical processing stream for two images. Network first produces meaningful representation of two images separately and then combine at later stage. More like a matching approach) and the flownetsimple (It Consists only of convolutional layers Allow the network to decide itself how to process and extract the motion information Fully supervised) in a very complex but fast architecture

Moving visual objects Must be characterized by motion in a given time interval: thus we would like to identify also objects that have been moved in a recent past. Shouldn't confused with regions with motion which are not objects (eg regions due to shadows, changes in illumination, specularity) and with "objects" or shapes with apparent motion: i.e. set of pixels detected as moving objects but that does not correspond to any real object (eg "Ghosts").

Methods of segmentation by motion

Differential methods for MVO segmentation

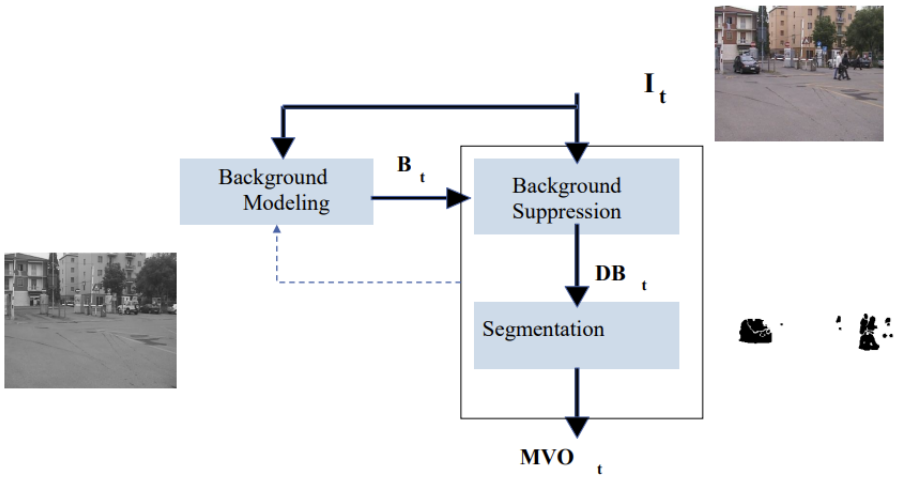
$$D(x, y) = \frac{dI(x, y)}{dt} > T$$

- Single difference: The derivative can be computed as the Absolute difference of luminance intensity (i.e., the grey levels) in two consecutive frames $D_n(x, y) = |I_n(x, y) - I_{n-1}(x, y)|$, the difference must be thresholded to obtain the moving points. Where

$I_n(x,y)$ is a moving point only if $D_n(x,y) > T$. This method is inexpensive and useful only to check the presence of motion, in dependency of the relative speed of the object with respect to the frame interval Dt.

- Double difference: $DD_n(x,y) = (D_n(x,y) > Th)$ & $(D_{n+1}(x,y) > Th)$, Introduces a form of filtering (on size), the sets of points detected by the Dn and Dn+1 do not correspond to the same physical parts of the objects, objects of small size are removed and the frame rate must be tuned to the objects' speed. Select only «strong points» Robust to noise and to small camera motions that are filtered out, The AND operator can avoid apparent motion and It depends on the relative speed.
- Double difference and Edge closure: Double Difference with blob validation, Edge detection, Double threshold (iterative check) and Filling for blob detection
- Moving edge closure: Moving points with the double difference DD(i,j), Edges with Sobel Gradient, MEC, Morphological closure and Filling

Background suppression for MVO detection: a schema



Methods for background suppressions Distance based method:

$$X_t(s) = \begin{cases} 1 & \text{if} \quad d(I_{s,t}, B_s) > \tau \\ 0 & \text{otherwise} \end{cases}$$

- Difference on pixel intensity (Koller 1994)
- Difference on the color space
- Malanobis distance (Wreng et al 1997)
- Multivalue distance (Stauffer et al 2000)
- Eigenbckg distance

Methods for background Modeling

- Adaptive background
- Statistical: : Mixture of Gaussian

- Statistical, point selective: Selective median
- Statistical and knowledge based : selective ad object level
- Layered background based on tracking
- Autoencoder for normality detection.. (a lot of proposal)

Adapting background modeling The simplest model takes into account the previous background and the current image with a parameter of adaptation α (To consider smoothed variable luminance conditions, alfa is between 0 and 1) . For each point s, at t+1 it is computed as:

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

and the distance is

$$d_\infty = \max |I^R_{s,t} - B^R_{s,t}|, |I^G_{s,t} - B^G_{s,t}|, |I^B_{s,t} - B^B_{s,t}|$$

MOG (Mixture of Gaussians) for background suppressions Each pixel is modelled differently and independently from the others, There are two (or more) models of background. The probability for a pixel to belong to a background model is given by a weighted sum of K (3-5) Gaussian models. Algorithm:

- Comparison: At each t, each pixel is compared to the k models and is associated with the closest model. A point «belongs» to a Gaussian model if is far from the mean value less than 2.5 the standard deviation. A point is a foreground point if it belongs to the less priority Gaussian or if it does not belong to anyone.
- Update: At each step the weights are updated. The weight of the Gaussian that obtains the match is incremented, The others are decremented and Mean and variance are adapted after each step.
- New models: If no match occurs, the Gaussian with minimum weight is canceled and a new one is created with the new foreground pixel. If the point has the same color for a long time, its weight can become higher than one of the K-1 weights of background models and can become background The parameter alfa modulates the time for a point to become a background from foreground and viceversa.

Sakbot for shadows Statistical and knowledge based object detector algorithm for detecting Moving Visual Objects in video taken by static cameras; Adopted in many single and multicamera surveillance systems features. Shadows are very critical in indoor surveillance (about same size of people shape). Problems in tracking (merging) and posture classification. A point is classified as SHADOW if: it is darker than the "shadowed" bkg (lower Value), small difference in Saturation and small absolute difference in Hue.

Background modeling by deep learning ...

Tracking: probabilistic model ...

Kalman filtering The Kalman filtering is a recursive linear estimation in the special case of Gaussian distribution. Algorithm:

- The initial uncertainly is given by the Gaussian Covariance with a given sigma (L). All equal since variable are not correlated

- State prediction: The first step is to predict the next state by using the motion model. The next state x(t—t-1) is obtained by multiplying the previous state by the state transition matrix - F.
- Covariance prediction: The covariance update is done by multiplying the covariance matrix from the previous iteration by the state transition matrix F (motion model) and by adding the process noise Q which can be constant. The update covariance of our prediction is wider because we are less sure about our estimate.
- Update: state update and uncertainty update (decreasing the uncertainty). When (if) we receive a noisy measurement (in our case position obtained form a detector), the update process begins. The noisy measurement z(t) is modeled as a single Gaussian, where the noise is modeled as covariance matrix R(t) which is usually constant. The uncertainty of the measurement appears as golden ellipse. Measurement update: A measurement has the same structure as a state or it just contains state parts; e.g. just the object position
- Correct: The Kalman gain is now used to update the state x and covariance matrix P. The result is the updated position which is denoted by the 4 yellow ellipse.

Kalman-based deep sort Extension to SORT (Simple Real time Tracker). It is based on Kalman, Kalman filter is a crucial component in deep SORT. The state contains 8 variables; (u,v,a,h,u',v',a',h') where (u,v) are centres of the bounding boxes, a is the aspect ratio and h, the height of the image. The other variables are the respective velocities of the variables. Distance metric (for z): to use the squared Mahalanobis distance (effective metric when dealing with distributions) to incorporate the uncertainties from the Kalman filter.

Other tracking methods

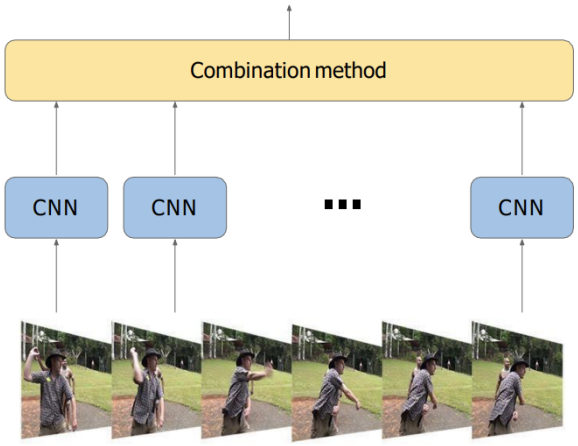
- Particle filtering
- Montecarlo
- Multiple target tracking

---

21 Video Architectures

Video Classification/Understanding datasets Like YouTube videos, Large variations in camera motion, object appearance and pose, viewpoint, background, illumination, etc. (UCF101 where there are 101 action categories, Sports-1M where there are sports labels, Youtube8M for audio-visual features, Kinectics, Atomic Visual Action [AVA] for pose and object interaction, Moments in Time [MIT], M-VAD and MPII-MD dataset for movie querying, DALY [INRIA], Large Scale Movie Description Challenge [LSMDC],

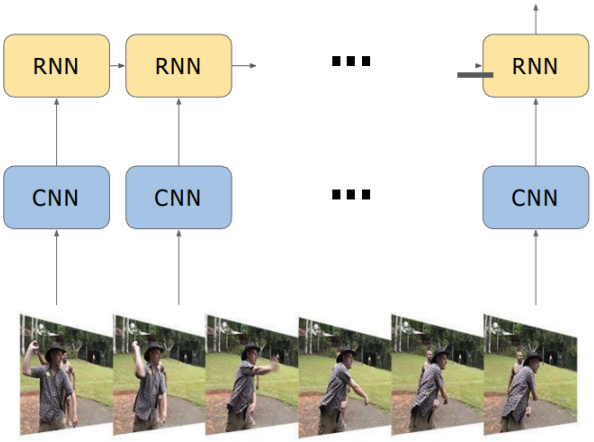
Single frame models



Combination is commonly implemented as a small NN on top of a pooling operation (e.g. max, sum, average).

Problem: pooling is not aware of the temporal order!

2DCNN+RNN

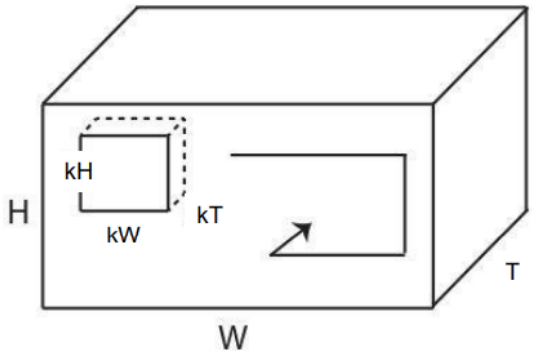


Recurrent Neural Networks are well suited for processing sequences.

Problem: RNNs are sequential and cannot be parallelized.

3D CNN (C3D) model

- 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



3D Convolutions

- Input Shape: n (images), iC, T, H, W
- Kernel Shape: oC, iC, kT, kH, kW where oC is bias shape and T, H, W are three axes
- Output Shape: n, oC, oT, oH, oW

Input: $(N, C_{in}, D_{in}, H_{in}, W_{in})$
Output: $(N, C_{out}, D_{out}, H_{out}, W_{out})$ where

$$D_{out} = \left\lfloor \frac{D_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[2] - \text{dilation}[2] \times (\text{kernel\_size}[2] - 1) - 1}{\text{stride}[2]} + 1 \right\rfloor$$

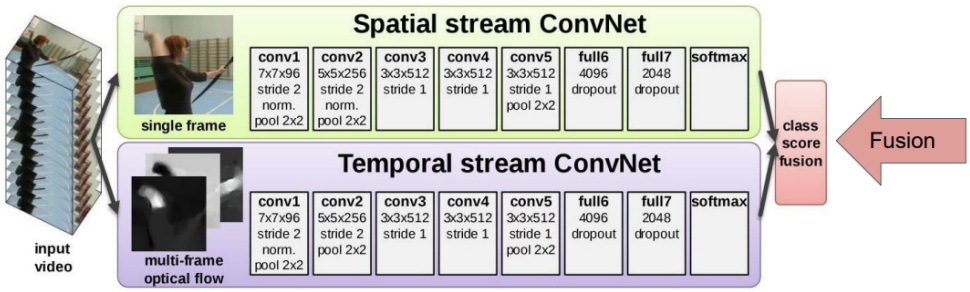
-

3D Pooling Performs an operation with zero parameters (e.g. max, mean) inside a neighbourhood, which slides over (T, H, W) with a given stride. Differently from convolution, we do not perform a reduction (i.e. sum) over the input channel axis; instead, the operation is applied independently on all input channels. Output size is (n, iC, oT, oH, oW) with:

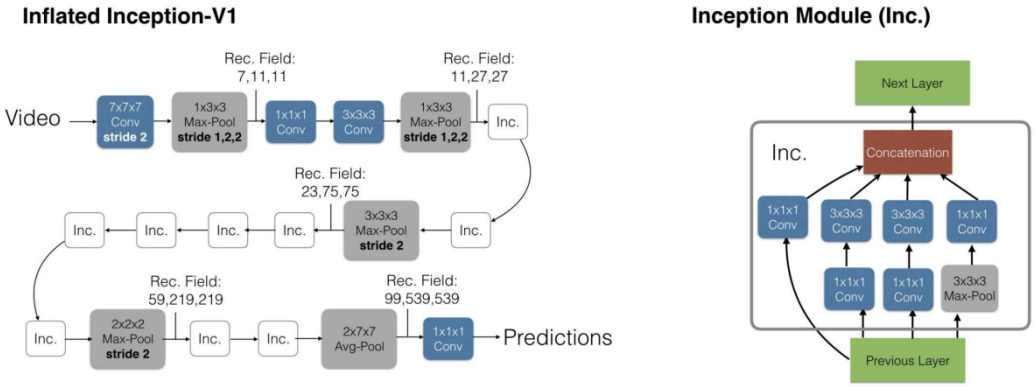
- $oT = (T – kT)/s + 1$
- $oH = (H – kH)/s + 1$
- $oW = (W – kW)/s + 1$

Two streams CNNs

Problem: Single frame models do not take into account motion in videos.
Solution: extract optical flow for a stack of frames and use it as an input to a CNN.



Inflated 3D CNNs (I3D) Adapt 2D CNNs found for ImageNet classification to 3D convolutions, 3D models are initialized with ImageNet images transformed into boring video sequences.

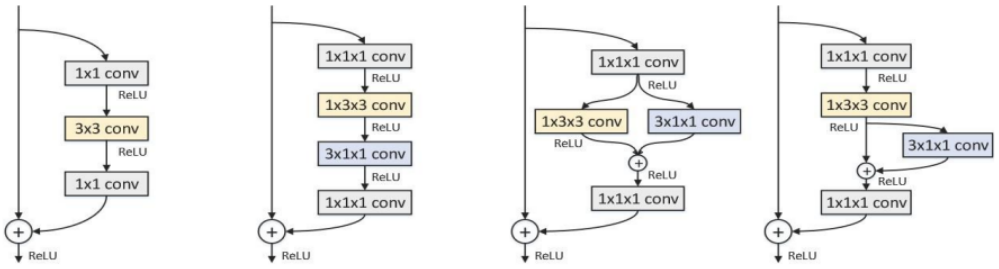


Pseudo 3D / ResNet(2+1)D

Pseudo-3D (P3D) Residual Net:

- 3 types of P3D blocks
- Interleaving Design for ResNet

ActivityNet	Top-1	Top-3	MAP
IDT [34]	64.70%	77.98%	68.69%
C3D [31]	65.80%	81.16%	67.68%
VGG_19 [26]	66.59%	82.70%	70.22%
ResNet-152 [7]	71.43%	86.45%	76.56%
P3D ResNet	**75.12%**	**87.71%**	**78.86%**



ResNet (2+1)D:

• Spatial-Temporal Decomposition but with similar number of parameters

Kinetics

method	pretraining dataset	top1	top5
I3D-RGB [4]	none	67.5	87.2
I3D-RGB [4]	ImageNet	72.1	90.3
I3D-Flow [4]	ImageNet	65.3	86.2
I3D-Two-Stream [4]	ImageNet	**75.7**	**92.0**
R(2+1)D-RGB	none	72.0	90.0
R(2+1)D-Flow	none	67.5	87.2
R(2+1)D-Two-Stream	none	73.9	90.9
R(2+1)D-RGB	Sports-1M	74.3	91.4
R(2+1)D-Flow	Sports-1M	68.5	88.1
R(2+1)D-Two-Stream	Sports-1M	**75.4**	**91.9**

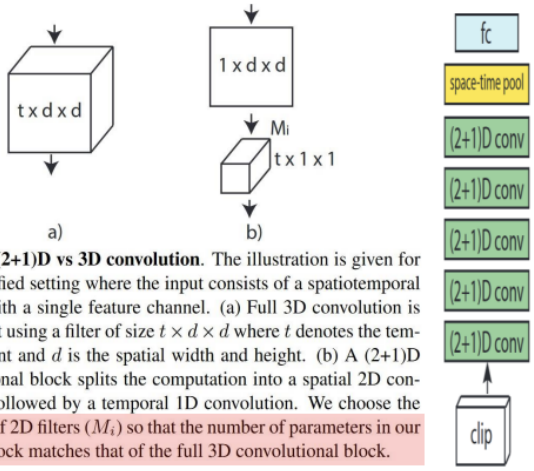
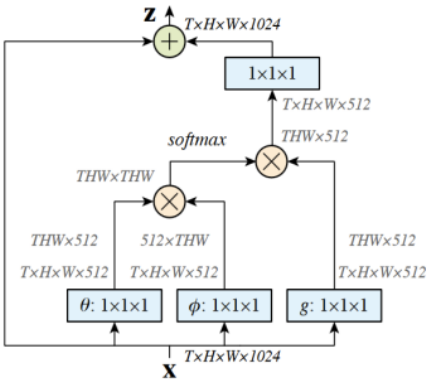
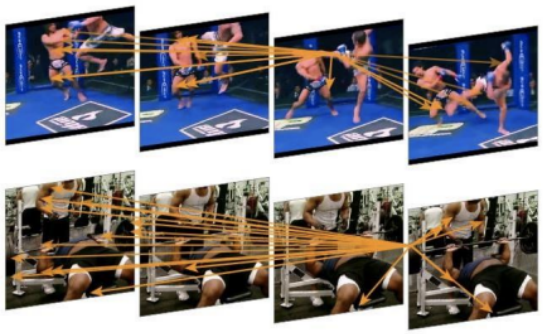


Figure 2. **(2+1)D vs 3D convolution**. The illustration is given for the simplified setting where the input consists of a spatiotemporal volume with a single feature channel. (a) Full 3D convolution is carried out using a filter of size $t \times d \times d$ where t denotes the temporal extent and d is the spatial width and height. (b) A (2+1)D convolutional block splits the computation into a spatial 2D convolution followed by a temporal 1D convolution. We choose the numbers of 2D filters (M_i) so that the number of parameters in our (2+1)D block matches that of the full 3D convolutional block.

Non local Neural Networks

Non-local Neural Networks: spatial-temporal attention among frames





Kinetics

	backbone	modality	top-1	top-5
I3D in [7]	Inception	RGB	71.1†	89.3†
2-Stream I3D in [7]	Inception	RGB + flow	74.2†	91.3†
RGB baseline in [3]	Inception-ResNet-v2	RGB	73.0	90.9
3-stream late fusion [3]	Inception-ResNet-v2	RGB + flow + audio	74.9	91.6
3-stream LSTM [3]	Inception-ResNet-v2	RGB + flow + audio	77.1	93.2
3-stream SATT [3]	Inception-ResNet-v2	RGB + flow + audio	77.7	93.2
NL I3D [ours]	ResNet-50	RGB	76.5	92.6
	ResNet-101	RGB	**77.7**	**93.3**

22 Face Recognition

DeepFace One of the first deeply-learned architectures for Face Verification. Two training modalities: Use of Siamese Networks, with the following distance function where α_i are the trainable parameters with standard cross-entropy loss and backward propagation

$$d(f_1, f_2) = \sum_i \alpha_i |f_1[i] - f_2[i]|$$

Or Using a weighted 2 distance where f1 and f2 are the DeepFace Representations. The weights parameters wi are learned using a linear SVM

$$X^2(f_1, f_2) = \sum_i w_i (f_1[i] - f_2[i])^2 / (f_1[i] + f_2[i])$$

FaceNet Map images to a compact Euclidean space, where distances correspond to face similarity. In 128-D space classical techniques can be applied for clustering and categorization, like k-means or KNN. Find $f(x) \in R^d$, for an image x, so that: $d^2(f(x_1), f(x_2))$, is small when x1 and x2 are in the same identity and are large when are in different identities. This face recognition/verification/clustering model learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity.

Triplet training/Selection Triplet Loss Function:

$$\sum_i^N [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha]_+$$

$$||f(x_i^a) - f(x_i^p)||_2^2 + \alpha < ||f(x_i^a) - f(x_i^n)||_2^2 \forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in T$$

where

- x_i^a is the anchor image
- x_i^p is the positive image
- x_i^n is the negative image
- T is the set of all possible triplets in the training set
- α margin between positive and negative pairs

Triplet Selection: generating all possible triplets would result in many that satisfy the equation. These triplets would not contribute to the training and would result in slower convergence. There are two possible solutions:
- Offline generation, every n steps, using the most recent network checkpoint and computing the argmin and argmax on a subset of the data.
- Online generation: done by selecting hard positive or negative examples from within a mini-batch. Use only hard triplets, i.e. triplets that violate the equation

$$_{x_i^p} ||f(x_i^a) - f(x_i^p)||_2^2$$

$$_{x_i^n} ||f(x_i^a) - f(x_i^n)||_2^2$$

Mini-batches of a few thousands face images, Minimal number of examples of each identity per mini-batch, Selecting only the semi-hard negatives and using all anchor-positive pairs of mini-batch

---