Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

INFORMATION ENGINEERING DEPARTMENT

COMPUTER ENGINEERING - AI & ROBOTICS

Computer Vision

Francesco Crisci 2076739

Low-level vision

Image coding

We can define an image as a 2-dimensional function f(x,y), where x an y are the spatial coordinates, and the amplitude of f is called intensity or gray level of the image at that point. f(x,y) must be nonnegative and finite. To digitize an image we need two operations: sampling, which is the operation through which we digitize the coordinate values, and quantization, which is the operation through which we digitize the amplitude values.
We mathematically define sampling as follows: f(m,n) := $f(m\delta x, n\delta y)$, where δx and δy are the sampling period along x an y axis.
We define quantization as: $f_d(m,n) = Q[f(m,n)]$.

Representing digital images

The coordinates (i,j)of the image are integers, where i = 0,...,M-1 and j = 0,...,N-1. Where M and N are the width and height of the image. We can express the f(x,y) as a matrix:

$$\begin{bmatrix} f(0,0) & f(0,1) & ... & f(0,N-1) \\ f(1,0) & f(0,1) & ... & f(1,N-1) \\ ... & ... & ... & ... \\ f(M-1,0) & f(M-1,1) & ... & f(M-1,N-1) \end{bmatrix}$$

Each element of the matrix is called pixel.
The origin of the image is at the top left corner, and x and y represent the rows and the columns of the matrix.
We have no restrictions on the values of M and N for an image other than they must be positive integers.
We have restrictions o the number of intensity levels L being an integer power of 2: $L = 2^k, k \in N$.
The number of bits required to store a gray scale image is b = MxNxk
Color images uses 3 channels instead of one, we then have 3 matrices(one per each channels) related to the images. We can define a color image as:

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

Spatial Resolution and Intensity Resolution

Spatial resolution is the number of pixels per unit distance. In other words, is the smallest detectable detail in the image.
Intensity resolution is the number of bits used per pixels. In other words the smallest detectable change in gray level. We usually use 8 bits to represent gray scale images, but there are some cases where it is useful to use 16 or 32.
Contrast is the difference between the highest and lowest intensity level in the image.

Geometric transformation

A geometric transform is a modification of the spatial relationship among pixels. We use two steps: coordinate transform $(x', y') = T\{(x,y)\}$ and image resampling.
The transformation coordinates may be expressed as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

where x an dy are the pixel coordinates of the original image and (x', y') are the corresponding pixels of the transform image.
To represent transformation in a more compact way we can use the homogeneous coordinates, which allows us to have matrix multiplication only.
We want to focus on affine transformations, which keep points, straight lined and planes.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This transform can scale, rotate, translate or sheer an image according to the values chosen for the elements of A.
It preserves point collinearity and the distance ratios along a line(given $p_1, p_2, P-3$ lying on a line, $\frac{|p_2-p_1|}{|p_3-p_2|} = k$(constant)).

Image mapping

We have two ways of doing mapping: forward or backwards mapping.
Forward mapping makes us compute a corresponding (x',y') for each(x,y), while the backwards mapping allows us to do the opposite. The second option is more reliable, since it finds each (x',y') only once and fills all the pixels.

Single pixel operations

We consider now grayscale images with L gray levels. Single pixel operations are function that change the gray levels of an image. We have two functions: I(x,y) which represents the image and the function T() which represent the gray level change.

Negative

It switches dark and light. Considering an image which has intensity levels in the range $[0, L-1]$. The negative image is then obtain by applying to each pixel the following transformation: s= L-1-r, where r is the intensity in the considered pixel.
This transform is used to enhance white or gray detail embedded in dark regions, especially when black areas are dominant in size.

Log

It highlights the differences among pixels in given conditions. To compute the new value of the pixel with the following relation: s = clog(1+r), where c is a constant, and $r \geq 0$ is the original intensity of the considered pixel.
It maps a narrow image of low intensity values in the input into a wider range of output levels.
$c = \frac{L-1}{log(L)}$

Gamma

More versatile than the log transform since it is tunable. The new value of the pixel is obtained through the following relation: s = cr^{γ}, where both c and γ are positive constant. the effect of $\gamma > 1$ are the opposite of those of $\gamma < 1$. Small gammas tends to make lighter the darker regions, which allows us to spot more visible details, while great values of gamma tends to darken the image.
$c = (L-1)^{1-\gamma}$
For gamma = 1 the transform equals the identity transform.
It was very popular when CRT monitors were used. The physics regulating the light intensity for CRT monitor is I = V^{γ}, while the compensation by the gamma transform s = $cr^{\frac{1}{\gamma}}$

Contrast stretching

Contrast is the difference between the highest and lowest intensity level in the image. Another way of measure it is the RMS contrast, or RMSC, which is defined as follows:

$$\text{RMSC} = \sqrt{\tfrac{1}{MN}\Sigma_{i=0}^{N-1}\Sigma_{j=0}^{M-1}(I(i,j)-\bar{I})^2}.$$

The contrast stretching expands the range of intensity levels in an image so that it spans the ideal full intensity range of the recording medium or display device.
Given a couple of points (r_1, s_1) and (r_2, s_2) we can control the the shape of the transformation function. if $r_1 = s_1$ and $r_2 = s_2$ we have a linear function which produces no changes in intensity. If $r_1 = r_2, s_1 = 0$ and $s_2 = L - 1$ we have a threshold function which creates a binary image. In general $r_1 \leq r_2$ and $s_1 \leq s_2$ so that we have a monotonically increasing function.

Intensity slicing

It enables to consider only a subset of the given image. We consider only a subset of the gray levels and we set the others either to 0 or to 255 in case of the intensity slicing. One approach is to display in one value all the values in the range of interest and in another all other intensities. Another approach is to brighten the desired range of intensities and leaving all the others unchanged.

Bitplane slicing

In the case of bitplane slicing we consider only a subset of the bits encoding the gray levels of the image. We can consider the image as composed of 8 planes, one per each bit.

Histogram Processing

Let r_k, for k = 0,1,2,...,L-1 denote the intensities on an L-level digital image. The unnormilized histogram is defined as $h(r_k) = n_k$, where n_k is the number of pixels with intensity r_k. The normilized histogram is defined as:

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}$$

The sum of p(r_k) for all its values is always 1.
The histogram is really useful for evaluating image statistics, compression, segmentation and image enhancement.

Histogram equalization

We can equalize the histogram tanks to an equalization function. This operation produces a flatten histogram and can be inverted if the function is strictly monotonic. We assume that T(r)is monotonic increasing in the interval $0 \leq r \leq L-1$ and that $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$. T(r) is continuous and differentiable.
The cumulative distribution function is the T(r) transformation that equalizes the histogram:

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$
$$s_k = T(r_k) = (L-1)\Sigma_{j=0}^{k} p_r(r_j)$$

Sometimes it can happen that the output is not completely flat due to the discrete nature of the data.

Histogram specification

We want to modify an histogram to a desired shape. This procedure is called histogram specification or histogram matching.
We equalize the histogram, then specify the desired output shape of the histogram(done by listing all the points in the histogram). We obtain the inverse transformation, then apply equalization and transformation to the desired shape. In the end we map the two together.(Ghidoni's slides).
Consider two continuous intensities: r (input) and z (output). We estimate $p_r(r)$ from the input image and $p_z(z)$ is the specified PDF we wish the output image to have. Let s be a random variable with the property:

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

where w is a dummy variable of integration. We now define a function G on variable z with the property:

$$G(z) = T(r) = (L-1)\int_0^r p_z(v)dv = s$$

where v is a dummy variable of integration. It follows that G(z)=s=T(r) and that z must satisfy the condition:

$$z = g^{-1}(s) = G^{-1}[T(r)]$$

To obtain the image whit the desired histogram we use the following procedure:

- obtain $p_r(r)$ from the input image
- use $p_z(z)$ to obtain function G(z)
- compute the inverse transform $z = G^{-1}(s)$; this is a mapping from s to z, the latter being the values that have the specified PDF
- Obtain the output image by first equalizing the input image; the pixel values in this image are the s values. For each pixel with values s in the equalized image, perform the inverse mapping $z = G^{-1}(s)$ to obtain the corresponding pixel in the output image. When all pixels have been processed with this transformation, the PDF of the output image will be equal to the specified PDF.

The procedure for discrete histogram specification is pretty much the same, just remember you have sum instead of integrals.

Spatial Filtering

Spatial filtering is based on a filter/kernel. A kernel defines a neighborhood (the surrounding pixels of the pixel we are actually considering) and a weight associated with each pixel involved in the computation.
A kernel is an array whose size defines the neighborhood operation, and whose coefficients determine the nature of the filter. At any point (x,y) in the image, the response g(x,y), of the filter is the sum of products of the kernel coefficients and the image pixels encompassed by the kernel. As coordinates x and y are varied, the center of the kernel moves from pixel to pixel, generating the filtered image, g, in the process.
We focus on kernel of odd size in booth coordinate directions.
Local operations are performed in the spatial domain of the image, implying spatial filtering and that the kernel is a spatial filter.
The filter weights can affect the brightness of the image. If $\Sigma_i w_i = 1$, then the brightness remains unchanged.

Correlation

The filter is superimposed on each pixel location. We have an evaluation of a weighted average.(Ghidoni)
Correlation consists of moving the center of a kernel over an image, and computing the sum of products at each location. The mechanism of convolution are pretty much the same , except that the kernel is rotated by 180 degrees. This implies that both, correlation and convolution, yield the same result.
Supposing the filter is mxn, with m = 2a+1 and n = 2b+1, we can define correlation as follows:

$$g(x,y) = \Sigma_{s=-a}^{s=a}\Sigma_{t=-b}^{t=b} w(s,t)I(x+s,y+t)$$

Convolution

The mathematical definition of convolution is the following(given the same premises as correlation):

$$g(x,y) = \Sigma_{s=-a}^{s=a}\Sigma_{t=-b}^{t=b} w(s,t)I(x-s,y-t)$$

In the CV context, convolution and correlation are often used as synonyms, even tho we are probably using correlation.
The filter obtained by applying convolution are called convolutional filters.

Linear filters

Averagin filter

We divide the kernel by the sum of the pixel weights. The size of the filter can be increased leading to a stronger smoothing.

Separable filters

We can apply a filter on rows and then on columns or the other way around. Is asymptotically faster: O(MN(a+b)) instead of O(MNab).

Non-linear filters

Derivative filters

Recall of first-order derivatives: is zero in flat segments; is non-zero on the onset of a ramp or step and along ramps.
Recall of second-order derivative: is zero in flat segments and along ramps of constant slope; is non-zero on the onset and at the end of a step or ramp.
We can evaluate the first order derivative as: $\frac{\delta f}{\delta x} = f(x+1) - f(x)$
We can also compute the second order derivative as: $\frac{\delta^2 f}{\delta x^2} = f(x+1) + f(x-1) - 2f(x)$.
Second order derivatives are easier to implement wrt first-order ones.

First order derivative filter

We can compute the Gradient module as $M = \sqrt{f_x^2 + f_y^2}$ which is a scalar, non-linear and ideally isotropic.
We can use Roberts kernel to compute first order derivative: $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$. We use it to see sudden changes in the image(kind of to see where there are borders or edges, but not quite yet.)

Second order derivative: Laplacian

The Laplacian filter implements the following equation:

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

which can be represented with the following 4 kernels:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

The Laplacian can be used to enhance the transitions in the image. Combination of the image and the Laplacian can be used to sharpen the image or to be subtracted if the center weight is negative: $g(x,y) = f(x,y) - \nabla^2 f(x,y)$.
The previous equation becomes more generally:$g(x,y) = f(x,y)c[\nabla^2 f(x,y)]$, where c = -1 if the center of the laplacian kernel are negative, c = 1 for the other 2 cases.

Image restoration through non-linear Filters

Examples of non-linear filters are min, max and median. They can be used to suppress components such as a spike(an element which strongly differs from another). Spikes can appear through the image as noise. Removing the noise is the main concept behind image restoration.
We can model the acquire image as the following:

$$f_{acq}(x,y) = f(x,y) + \eta(x,y)$$

which is the ideal image with noise. The noise component is usually unknown. Spatial filtering is the method of choice for estimating f(x,y) in situations when only additive random noise is present.
There are several classes of noise available: Gaussian, Rayleigh, Gamma, Exponential, Uniform and Impulse.

Smoothing filter

The smoothing filters we've seen are averaging and Gaussian.

Arithmetic mean

The simplest of the mean filters, can remove gaussian noise but not salt and pepper noises. Let S_{xy} denote the set of coordinates in a rectangular subimage window of size mxn, centered in (x,y). The arithmetic mean filter computes the avarage value of the corrupted image, g(x,y) in the area defined by S_{xy}. The value of the restored image at point (x,y) is:

$$\hat{f}(x,y) = \frac{1}{mn}\Sigma_{(r,c)\in S_{xy}} g(r,c)$$

where r and c are the row and column coordinates of the pixels contained in the neighborhood. This operation can be implemented using a spatial kernel of size mxn in which all coefficients have value $1/mn$.
It reduces noise as a result of blurring and smooths local variations in an image.

Geometric mean

It can remove Gaussian noise but not salt and pepper.It reaches the same result as the arithmetic mean but it tends to lose less image detail in the process.

$$\hat{f}(x,y) = [\Pi_{(r,c)\in S_{xy}} g(r,c)]^{\frac{1}{mn}}$$

Here, each restored pixel is given by the product of all pixels in the subimage area, raised to the power $\frac{1}{MN}$

Harmonic mean

It can remove Gaussian and salt, noise, but not pepper. The formula is here:

$$\hat{f}(x,y) = \frac{mn}{\Sigma_{(r,c)\in S_{xy}} \frac{1}{g(r,c)}}$$

Contraharmonic mean

Can basically remove Gaussian, salt and pepper noise according to the value of the parameter Q.

$$\hat{f}(x,y) = \frac{\Sigma_{(r,c)\in S_{xy}} g(r,c)^{Q+1}}{\Sigma_{(r,c)\in S_{xy}} g(r,c)^{Q}}$$

The parameter Q is called the order of the filter. It is ideal to remove the effect of salt or pepper noises. For positive values of Q it removes the pepper noise. For negative values it removes salt noise. If Q=0 it reduces to arithmetic mean filter. It cannot remove both salt and pepper noise simultaneously. For Q=-1 it reduces to the harmonic mean filter.

Median filters

It replaces the value of a pixel by the median of the intensity levels in a predefined neighborhood of that pixel:

$$g(x,y) = \frac{1}{mn-d}\Sigma_{s,t\in R_r} f(s,t)$$

where d is a parameter which tells us the window of values we consider.

Alpha-trimmed mean

Suppose we delete the d/2 lowest and highest values of g(r,c) in the neighborhood. Let $g_R(r,c)$ be the mn-d pixels in the neighborhood. A filter formed by averaging these remaining pixel is an alpha-trimmed mean filter:

$$\hat{f}(x,y) = \frac{1}{mn-d}\Sigma_{(r,c)\in S_{xy}} g_R(r,c)$$

where d goes from 0 to mn-1. When d = 0 we have an arithmetic mean filter. If d =mn-1 we obtain a median filter. It is really useful in situation where we have multiple type of noises, e.g., gausssian, salt and pepper.

Min and Max filters

The max filter highlights salt noise but removes pepper. The kin filter highlights the pepper and removes the salt. We take the max or the min in the neighborhood and we give the found value to the pixel we are considering.

Adaptive filters

This kind of filter are able to tune their effect comparing local image variance σ_L^2 and noise variance σ_η^2. If the local variance is way greater than the noise variance, the effect of the filter should be weak. An high local variance is typically associated with edges which should be preserved. When the values of the two variances are close the filter effect should be strong. In this case we want the filter to return the arithmetic mean value of the pixels in the neighborhood. This condition occurs when the local area has the same properties as the overall image, and local noise is to be reduced by averaging.

Image in frequency

Signals and sampling can be effectively described using a mathematical tool: the Fourier transform. This tool allows us to do frequency analysis and filtering. Due to this transform. we change domain: from space to space frequency: $x \to f_x$ and $x \to f_y$.
In the case of 1 dimension signal we have that the Fourier transform is the following:

$$\int_{-\infty}^{+\infty} f(t)e^{-2j\pi\mu t}dt$$

while the inverse Fourier transform is:

$$\int_{-\infty}^{+\infty} F(\mu)e^{2j\pi\mu t}d\mu$$

Sampling signals

Sampling is represented by a set of regular pulses, also called train of impulses. Each pulse is separated by a sampling interval ΔT. The sampled signal is the multiplication between the original signal and the train of impulses. A sampled signal is then expressed by:

$$\tilde{f}(t) = \Sigma_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T)$$

Notice that we need a uniform interval ΔT, of the independent variable t. Each component of this summation is an impulse weighted by the value of f(t) at the location of the impulse.
The value of each sample is given by:

$$f_k = \int_{-\infty}^{\infty} f(t)\delta(t - k\Delta T)dt = f(k\Delta T)$$

Notice how we used the sifting property of delta. This result holds for any value of $k \in Z^+$

Impulses and their shifting properties

A unit impulse of a continuous variable t, located at t = 0, and denoted $\delta(t)$, is defined as:

$$\delta(t) = \begin{cases} \infty \ if \ t = 0 \\ 0 \ otherwise \end{cases}$$

which is also called Dirac delta function. It has unit area since: $\int_{-\infty}^{+\infty} \delta(t)dt = 1$.
Another important property of the Dirac delta is the Sifting, which, with respect to integration, is the following:$\int_{-\infty}^{+\infty} f(t)\delta(t)dt = f(0)$, if f(t) is continuous at t=0. More generally, we can rewrite the sifting property in the following way:$\int_{-\infty}^{+\infty} f(t)\delta(t - t_0)dt = f(t_0)$, which gives us the value of the function at the location of the impulse.
In the case of a discrete variable, we have that the delta is the following:

$$\begin{cases} 1 \ if \ x = 0 \\ 0 \ otherwise \end{cases}$$

which clearly satisfies $\Sigma_{x=-\infty}^{\infty}\delta(x) = 1$.
We can now define the impulse train by a sum of delta function:

$$s_{\Delta T} = \Sigma_{k=-\infty}^{\infty}\delta(t - k\Delta T)$$

Discrete variables have the sift property as well: $\Sigma_{x=-\infty}^{\infty} f(x)\delta(x) = f(0)$, which can be rewritten in the more general form as before.

Effects of sampling

Let us denote $F(\mu)$ as the Fourier transform of continuous function f(t). We know that the product of two functions in the spatial domain is the convolution of the two functions in the frequency domanin. Thus the Fourier transform of the sampled function is: $\tilde{F}(\mu) = F(\mu) * S(\mu)$, where S is the transform of the delta function. The sampled signal then becomes:

$$\tilde{F}(\mu) = \int_{-\infty}^{\infty} F(\tau)S(\mu - \tau)d\tau$$

After some calculations, we obtain the following result:

$$\tilde{F}(\mu) = \frac{1}{\Delta T}\Sigma_{n=-\infty}^{\infty} F(\mu - \frac{n}{\Delta T})$$

which means that the transformed domain spectrum is replicated.

The sampling theorem

The spectrum is replicated, and each replicas can be at different distances according to the sampling period. In case some replicas overlap we have aliasing.
With this theorem we sate the conditions under which we can reconstruct a signal.
We can recover f(t) from its samples if we can isolate a single copy of $F(\mu)$from the periodic sequence of copies of this function contained in $\tilde{F}(\mu)$, the transform of the sampled function $\tilde{f}(t)$. To guarantee a sufficient separation we need to satisfy the following inequality: $\frac{1}{\Delta T} > 2\mu_{max}$, where μ_{max} is the Nyquist rate.
This equation indicates that a continuous, band-limited function can be recovered completely form a set of its samples if the samples are acquired at a rating exceeding twice the highest frequency content of the function.
To reconstruct the signal we need to isolate one repetition in frequency, which can be done by means of a rect function. When we multiply by the periodic sequence, the rect function isolates the period centered on the origin.

Fourier transform of images

Images have some differences with respect to commonly used signals, since they have 2 dimensions and only positive values for x and y. Let f(x,y) be a continuous function of two continuous variables. The Fourier transform and its inverse become:

$$F(u, v) = \int\int_{-\infty}^{\infty} f(x, y)e^{-2j\pi(ux+vy)}dxdy$$
$$f(x, y) = \int\int_{-\infty}^{\infty} F(u, v)e^{2j\pi(ux+vy)}dudv$$

and the discrete Fourier transform:

$$F(u, v) = \Sigma_{x=0}^{M-1}\Sigma_{y=0}^{N-1} f(x, y)e^{-2j\pi(\frac{ux}{M}+\frac{vy}{N})}$$
$$f(x, y) = \frac{1}{MN}\Sigma_{x=0}^{M-1}\Sigma_{y=0}^{N-1} F(u, v)e^{2j\pi(\frac{ux}{M}+\frac{vy}{N})}$$

To reconstruct the signal we use the rect in frequency. In case of the 2 dimensional case we use the Rect-sinc transform. The sampling is performed in both direction and the replicas are generated in both directions. The image is uniformly sampled over an orthogonal lattice with spacing ΔX and ΔY, leading tho the reformulation of the sampling theorem as follows:

$$F_X = \frac{1}{\Delta X} > 2\mu_{max} \text{ and } F_Y = \frac{1}{\Delta Y} > 2v_{max}$$

where μ_{max} and v_{max} are the maximum spatial frequencies of the signal along X and Y, respectively. Also in this case we are supposing that the signal is band-limited.

Aliasing in images

Since we cannot sample a function infinitely, aliasing is always present in digital images. We have 2 categories of aliasing in images: spatial or temporal aliasing. The first one is caused due to under-sampling the image, while the second kind is related to time intervals between images of a sequence of dynamic images. We'll mainly focus on spatial aliasing.
The higher the frequency of an image component, the higher the spatial aliasing is gonna be in the image. Low frequency elements can be rendered reasonably well. Properly sampled periodic images will be free of aliasing.
Sometimes aliasing can lead to misleading result, where the image seems normal, but in fact it is not what it is supposed to look like.

The Moirè effect

This effect is caused by beating of similar patters and aliasing is not involved. It comes into play because some patterns of this kind of effect are visually similar to aliasing.

Specturm and Phase angle

We can express the Discrete Fourier transform in polar form, which means decomposing the image in Spectrum and Phase:

$$F(u,v) = |F(u,v)|e^{j\phi(u,v)}$$

where the magnitude $|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2}$ is called the Fourier spectrum and $\phi(u,v) = arctan[\frac{I(u,v)}{R(u,v)}]$ is the phase angle or phase spectrum.(Remember that the arctan must be computed using a four-quadrant actangent function)
The Fourier spectrum is usually centered on the 0 value. A shift is typically needed to align the Fourier spectrum with the image representation space. The operation which does not affect the spectrum is the translation, while the rotation does. The phase encodes way more information than the spectrum.

Filtering in frequency

Filters can be defined in frequency as well. We will se Lowpass filters to smooth an image, Highpass filters to sharpening it and Selective filters.
To obtain a filter in the frequency domain, we need a spatial filter and we take the forward Fourier transform of the kernel. The filter obtain by this kernel is denoted as H(u,v).

Lowpass Filter

An ideal lowpass filter is expressed in terms of a distance D from the center, and has the zero-frequency in the middle of the figure.
Edges and other sharp intensity transitions contribute significantly to the high frequencies. Hence, smoothing is achieved by high-frequency attenuation.
Going back to an ideal low-pass filter, we have that $H(u,v) = \begin{cases} 1 \; if \; D(u,v) \leq D_0 \\ o \; otherwise \end{cases}$ where D_0 is a positive constant and D(u,v) is the distance between a point (u,v) in the frequency domain and the center PxQ in the frequency rectangle.

$$D(u,v) = [(u - P/2)^2 + (v - Q/2)^1]^{1/2}$$

Ideal refers to the assumption that that all frequencies on or inside the circle of radius D_0 are passed without attenuation, while all the others are attenuated.
The higher the radius value, the less noticeable is the ringing effect, which causes poor image quality. This kind of effect is typical of ideal filters due to their strong transition. Ringing effects can be reduced or eliminated by using filters showing smoother transitions.

Butterworth lowpass filter

The butterworth filter in frequency can be tuned by means of the parameter n. Lager values of n cause this filter to be similar to an ideal lowpass filter. The mathematical formulation of the filter is the following: $H(u,v) = \frac{1}{1+[D(u,v)D_0]^{2n}}$. For low values of n, we get closer to the Gaussian low pass filter. We can use this type of filter to approach the sharpness of an ILPF function with considerably less ringing.

Gaussian filter in frequency

This type of filter doesn't produce ringing. The mathematical formulation of the filter is the following: $H(u,v) = e^{-D^2(u,v)/2D_0^2}$.
It can be used to fill gaps or to do beautification. It is really useful to reduce noise.
Unlike other definitions, we do not use a multiplying constant in order to be consistent with the filters discussed. In this specific case, $\sigma = D_0$, leading to the mathematical formulation written above. In this case, D_0 is the cutoff frequency. When D(u,v)=D_0, the transfer function is down to 0.607 of its maximum value of 1.
Narrow Gaussian transfer functions i the frequency domain imply broader kernel functions in the spatial domain and vice versa.

Highpass filters

An highpass filter can be obtained from the corresponding low-pass filter: $H_{HP}(u,v) = 1 - H_{LP}(u,v)$. The features of LP and HP filter(like ringing) are similar for a given type of filter. In the case of high-pass filters we have a sharpening effect.
The formulation of an ideal high-pass filter is the following:

$$H_{HP}(u,v) = \begin{cases} 0 \; if \; D(u,v) \leq D_0 \\ 1 \; otherwise \end{cases}$$

It follows that the transfer function of a butterworth highpass filter is:

$$H(u,v) = \frac{1}{1+[D_0/D(u,v)]^{2n}}$$

The guassian becomes: $H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$

Selective filters

As in the spatial domain, also in frequency domain the noise can be present.
There are applications in which it is of interest to process specific bands of frequencies or small regions of the frequency rectangle. Filters in the first category are called band filters, while the filters in the second category are called notch filters.
This kind of filters operates on selected frequency bands, on selected frequency rectangles or area.

Band-reject and Band-pass filters

This type of filter are constructed on the basis of low-pass filters. A band-pass filter can be obtained from a band-reject filter by the following relation $H_{BP}(u,v) = 1 - H_{BR}(u,v)$.
When dealing with band-pass functions, the parameters of interest are the width, W, and the center D_0 from the origin.
The mathematical formulations of the filters are the following:

$$\text{Ideal: H(u,v)=}\begin{cases} 0 \; if \; D_0 - W/2 \leq D \leq D_0 + W/2 \\ 1 \; otherwise \end{cases}$$

$$\text{Buttherworth: H(u,v)=}\frac{1}{1+[\frac{DW}{D^2-D_0^2}]^{2n}}$$

$$\text{Gaussian: H(u,v)=}1 - e^{-[\frac{D^2-D_0^2}{DW}]^2}$$

Notch filters

This kind is the most useful of the selective filters. It rejects frequencies in a predefined neighborhood of the frequency rectangle. A notch filter with center at (u_0, v_0) must have a corresponding notch at location $(-u_0, -v_0)$. Notch reject filter transfer functions are constructed as products of highpass filter transfer functions whose centers have been translated to the center of the notches.

Vantages of filtering in frequency

Filtering in frequency means to apply convolution using filters that implement in space operations defined in frequency. The spatial mask can be obtained applying the inverse transform.

Bilateral filter

It is a very used spatial filter. It has many applications with high quality results. It is based on the gaussian filter but it can preserve the edges. The function consists in a weighted average of pixels with a new weight. The mathematical formulation is the following:

$$BF[I]_p = \frac{1}{W_p} \Sigma_{q \in S} G_{\sigma_s}(||p-q||) G_{\sigma_r}(|I_p - I_q|) I_q$$

where $\frac{1}{W_p}$ is the normalization factor, $G_{\sigma_s}(||p-q||)$ is the space weight and $G_{\sigma_r}(|I_p - I_q|)I_q$ is the range weight.
G_{σ_s} depends on geometrical distance, while G_{σ_r} depends on the gray level distance.
For G_{σ_r} that tend to infinity we get closer to a Gaussian blur, given that the other parameter is fixed. While the larger G_{σ_s}, the smoother becomes the larger features.
This kind of filter is really useful for denoising an image while preserving the edges. The stronger the filter, the more cartoonish the image becomes.

Mid-level vision

Edge detector algorithms

An edge can be caused by a variety of factors such as surface normal discontinuity, depth discontinuity, surface color discontinuity or illumination discontinuity. We can have several type of edges, due to different gradients. We have already analyzed several tools that can be useful to solve our problem such as the gradient, the Laplacian or smoothing filter. Let's try the gradient.

Gradient for edge detection

First of all, we need to formulate the discrete gradient formulas: $g_x = f(x+1,y) - f(x,y)$ and $g_y = f(x,y+1) - f(x,y)$. The gradient is a vector pointing towards the fastest varying direction. We get information about edge features considering the magnitude(edge strength) and the phase(fastest varying direction). We have already seen some masks for edge detection, such as Sobel operator.
We could also use the Laplacian, which can be used for isotropic detection or double line effect.
Derivative filters have a huge problem: they amplify noise, where the second derivative is noisier than the first.

First edge detection algorithm

A possible step of approaches to detect edges is the following: applying a low-pass filter for noise removal, which is really useful. Smoothing the image is usually useful prior to calculation. The second step would be calculating the gradient, while the last step would be applying a thresholding to detect only the strong edges.

Canny edge detector

It is a filter addressing the following targets:

- Low error rate. All edges should be found, and there should be no spurious responses.
- Edge points should be well localized. The edges located must be as close as possible to true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum.
- Single edge point response. The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. This means that the detector should not identify multiple edge pixels where only a singe edge point exists.

The first step in the canny edge detector algorithm is to smooth the image through a Gaussian filter. This step is really useful to remove the noise from the image. For the following step we need to compute the magnitude and the phase of the smoothed image, so that we can calculate the edge direction through the formula of the phase. Once we calculated the edge normal, we divide the ranges in bins of $45°$ in the interval $[0, 180]$. Afterwards we can formulate the nonmaxima suppression scheme at an arbitrary point (x,y) in α(it is the phase of the image):

- Find the direction d_k that is closest to $\alpha(x,y)$.

- Let K denote the value of $||\nabla f_s||$ at (x,y). If K is less that the value of the gradient at one or both neighbors of point (x,y) along d_k, let it become suppressed (its value becomes 0), otherwise let it be K.

The evaluated pixel considers a 3x3 neighborhood in the nonmaxima suppression.. Non-maxima suppression reduces the edge thickness. The last step is the hysteresis tresholding, which uses a low threshold and a high threshold. This operation allows to keep strong edges and week edges connected to the strong one. It eliminates isolated weak edges. From the histeresis thresholding we obtain two images according to the threshold: I_L and I_H. The pixels in the second image are considered part of an edge, while the pixels i the first are considered part of an edge if strong edger are around.
To understand better hysteresis: We create two images from the two different thresholds. The one that has been thresholded from the highest values has all it's edges contain in the one with a lower threshold. We then subtract the two images to obtain the final image with just strong edges and weak connected ones.
To make the edges longer we use the following procedure:

- Locate the next unvisited edge pixel, p, in the high-thresholded image.
- Mark as valid edge pixels all weak pixels in the low-threshold himage that are connected to p.
- if all nonzero pixels in the high-threshold image have been visited go to the next step or go back to the first one.
- Set to zero all pixels in the low-threshold image that were not marked ad valid edge pixels.

To recap, the canny edge algorithm has the following steps: smoothing, gradient computation, quantize the gradient, non-maxima suppression, hysteresis thresholding.

Resampling

It's the operation of changing sampling interval. Interpolation makes as going to a largr image, decimation does the opposite. It is usually made to go to a different lattice, which means to consider the pixel in different locations than the one they were sampled in.
One approach for interpolation is Nearest neighbor interpolation, which takes the pixel value of the pixel on it's top right.
We can also use bilinear interpolation, which uses a weighted average of the closest 4 samples. The coefficients depend on the distance to the samples and it's not linear with respect to x and y. IT is the best option to balance speed and accuracy. The formula is $p(x, y) = (1 - a)(1 - b)I_1 + a(1 - b)I_2 + (1 - a)bI_3 + abI_4$.
As a last option we have bicubic interpolation which consideres the 16 closest samples. It has 4 constraints on sample locations, 8 constraints on first order derivative(4 for x and 4 for y) and 4 constraints on x-y cross derivatives.

Finding lines

A possible approach to find lines is to compute the edges, considering all couples of edge points and evaluate the line passing through them an din the end, count the number of edge points for such line. It is not optimal since is quadratic in number of couples and it's cubic when we add comparisons.

Hough transform

It's a method to globally compute edges. Let us consider a general equation of a straight line which passes through the point (x_i, y_i) and which has equation $y_i = ax_i + b$. We have infinitely many lines that passes through these couple of points. If we rewrite the equation as $b = -ax_i + y_i$ and let us now consider the ab-plane we obtain the equation of a single line. A line that passes through 2 points allows us to find the intersection point in the parameter space. Vertical lines are a problem tho, since the slope of the line approaches to infinity. One way around this problem is to use the representation: $xcos\theta + ysin\theta = \rho$, which makes the parameter space be represented with sinusoidal curves. We have can rewrite the equation as follows $y = (\frac{cos\theta}{sin\theta})x + (\frac{\rho}{sin\theta})$. A horizontal line has $\theta = 0$, with ρ being equal to the positive x-intercept. Similarly a vertical line has $\theta = 90°$, with ρ being equal to the positive y-intercept, or $\theta = -90°$, with ρ being equal to the negative y-intercept. Each sinusoidal curve in the parameter space represents the family of lines that

passes trough a point (x_k, y_k)in the xy-plane. An intersection point in the parameter space represents a line that passes through 2 point in the xy-plane.
The true power of the Hough transform relies in dividing the $\rho\theta$ parameter space into so-called accumulator cells, where (ρ_{min}, ρ_{max}) and $(\theta_{min}, \theta_{max})$ are the expected ranges of the parameter values: $-90° \leq \theta \leq 90°$ and $-D \leq \rho \leq D$, where D is the maximum distance between opposite corners in an image.
For each edge pixel we compute (ρ, θ) pair values going along the sinusoidal curve and the we increment the value of the crossed cell. The counter present in each cell is the number of pixels on that line.(As i understood, we'll have the same sinusoidal shape in the accumulator cells, where the value of each cell is gonna tell us how many pixel are on that line).
The Hough transform can be used to find more complex shapes. We need to evaluate the general equation g(v,c)=0, where v is a vector of coordinates and c is a vector of coefficients. For example, the function which describes the points lying on a circle is $(x - c_1)^2 + (y - c_2)^2 = c_3^2$ and we can create a parameter space as before with an higher dimensionality(this case the parameter space would have 3 dimension).
The approach used to edge-linking problem using the Hough transform is the following:

- Obtain a binary image edge map using Canny
- specify subdivisions in the $\rho\theta$-plane
- Examine the counts of the accumulator cells for high pixel concentrations
- Examine the relationship between pixels in a chosen cell

Morphological image description

It's a low-level procedure but operates on the shapes found in an image. It generally works on binary images and it is based on set theory. The set-based description of an image is the following: A set is a vector of tuples, each tuple representing the (x,y) coordinates of a point belonging to the set (for example the set of all white pixels in an image).
It is possible to process an image working in this set-based description. An operator can add or remove pixel to or from a set. Such operators modify the image working on the shape.

Erosion

With A and B sets in Z^2, the erosion of A by be, denoted $A \ominus B$, is defined as:

$$A \ominus B = \{z | (B)_Z \subseteq A\}$$

where A is a set of foreground pixels. B is a structuring element and the z's are foreground values(1's). This means: translate B to point z, and keep z if and only if the whole structuring element is fully included in A. It is really useful to thin an image or to separate weakly connected components. The shape of the structuring element determines i which direction(s) the erosion operates.

Dilation

With A and B as sets in Z^2, the dilation of A by B, denoted as $A \oplus B$, is defined as:

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset\}$$

This equation is based on reflecting B about its origin and translating the reflection by z, as in erosion. The dilation of A by B then is the set of all displacements z, such that the foreground elements of $\hat{B}$ overlap at least one element of A. This means: translate B to point z, and keep z if and only if there is at least one pixel overlapping with A. It is really useful to thicken an image or merging close, unconnected components.As i n erosion ,the shape of the structuring element determines the direction(s) in which the dilation operates.

Opening

Morphological operations can be concatenated. Opening is the combination of erosion and dilatation and it is defined as follows: $A \circ B = (A \ominus B) \oplus B$ which allows us to do contour smoothing and to eliminate thin protrusions without reducing the element size.

Closing

Closing is the combination of dilation and erosion, and it is defined as follows $A \bullet B = (A \oplus B) \ominus B$ which allows us to fuse narrow breaks without increasing the element size.

Segmentation

The intuition behind segmentation is to partition an image into regions corresponding to different categories. Segmentation depends on the categories we have chosen. (Ghidoni's slides)
Let R represent the entire spatial region occupied by an image. WE may view the image segmentation ad a process that partitions R into n sub-regions $R_1, ..., R_n$ such that:

- $\cup_{i=1}^{n} R_i = R$
- $R_i \cap R_j = \emptyset \forall i, j (i \neq j)$
- R_i is a connected set, for all i

We have two main criteria: similarity between pixels in the same region and discontinuity between pixels in different regions.

Segmentation by similarity

Region Growing

Is a procedure that groups pixels or sub-regions into larger regions based on predefined criteria for growth. The basic approach is to start with a set of seed point, and from these grow regions similar to the seed. It is based on the concept of connectivity and local growing of regions. We need a stopping rule to make the grow stop. WE need to be able to define a region, or better, an interval in which we can cluster our pixels. It is not an easy task, but we'll see the following procedure:

- We first need an input image f(x,y), the seed array S(x,y) containing 1s at the locations of the seed points and 0 elsewhere (f and S have the same dimension) , and a predicate Q for controlling the growing of the region.
- We work on the seed points in S. We find all connected component in S and erode each component until one pixel is left; label all such pixels found as 1. All other pixels in S are labeled 0. (Ghidoni writes: Transform any S in a suitable set of seed points).
- Create an image $f_Q(x_i, y_i) = \begin{cases} 1 \; iff Q(x_i, y_i) \; is \; true \\ 0 \; otherwise \end{cases}$
- Create an image f_G by appending to each seed point in S the 1s in f_Q that are 8-connected to that seed point.
- Label each connected component in f_g with a different region label

Watershed

It is based on the 3D concept of an image (Intensity as the third value). It provides connected segmentation boundaries and connected components. We basically see the image as a topographic surface where the intensity is the height value. We consider 3 types of point: the one belonging to a regional minimum, points at which a drop of water, if placed at the location of any of those points, would fall with certainty to a single minimum, and point at which the drop of water could fall into two or more different minima (watershed lines). We want to find these watershed lines.
It is really useful to extract nearly uniform objects from the background. Regions characterized by small variations in intensity have small gradient values, thus we see the watershed segmentation applied to the gradient of an image. The minima are then correlated nicely with the small value of the gradient corresponding the objects of interest.
We need to fill the dam: Flood the surface from the minima, each flooded region is a catchment basin. We

need to prevent merging of water from different basins: when two catchment basins merge we build a dam between them, with the dam being taller than any pixel in the image. The dam points define the watershed lines which are the boundaries of the segments.
How do we build the dam? Dam construction is based on binary images. The simplest way to construct a dam is to use morphological dilation. When the dilation connects two different components we need to build a dam. The dilation has to be constrained to a connected component q and the dilation cannot be performed on points that would cause the sets being dilated to merge (in other words we cannot make two connected component a single connected component). When we construct the dam, we set the dam pixels to an higher value than the highest possible intensity one.
The watershed algorithm can be used on the gradient to extract uniform regions. It is really useful when we are dealing with small gradients, since small gradients means thick edges. The catchment basins correspond to homogeneous gray-level regions.

Over-segmentation

Direct application often leads to it due to noise and irregularities of the image. Enhancement: flood the topographic surface from a previously defined set of markers. Markers can drive the segmentation, where internal markers are associated with objects of interest and external markers are associated to the background. The marker selection can be done through pre-processing (that could be smoothing) or criteria definition.

Segmentation by thresholding

Segmentation needs one or more criteria which may be defined on the histogram. For example, we can apply the threshold and select the two resulting segments. This approach can be extended to multiple thresholds or ranges, where each segment is associated to a range.
Selecting the threshold may be trivial or tricky, and it is the only parameter in this kind of segmentation. According to the type of noise it might become really difficult to apply the threshold to the image, since the histogram of the image might look really sparse and without picks.
Also the illumination plays an important role in threshold segmentation. It can spread peaks in the image and make the histogram more homogeneous, making thresholding really hard since we do not have two perfectly separable regions. Controlling the illumination is then fundamental for image segmentation, then we need to be able to fix as much as possible before proceeding with segmentation.
Thresholding is effective depending on the distance between the peaks, image noise, relative size of the regions and illumination properties (we have similar effect when the reflectance properties of the object is not uniform).

Otsu's optimal threshold

It is a global thresholding method based on the histogram. It assumes that two classes are created by thresholding. It finds the optimal threhold and maximazes inter-class variance and minimizes intra-class variance.(Ghidoni's slides).
We can see thresholding as a statistical-decision theory problem whose objective is to minimize average error incurred in assigning pixels to two or more groups.
The basic idea is that properly thresholded classes should be distinct with respect to the intensity values of their pixels and that a threshold giving the best separation between classes in terms of their intensity values would be the best threshold.
Let $\{0, 1, ..., L-1\}$ denote the set of L distinct integer intensity levels in a digital image of size MxN. The normalized histogram has components $p_i = n_i/MN$, where n is a pixel, from wich it follows that:

$$\Sigma_{i=0}^{L-1} p_i = 1,\ p_i \geq 0$$

Now, suppose that we select a threshold T(k)=K, and use it to threshold the input image into two classes. Using this threshold, the probability, $P_1(k)$, that a pixel is assigned the the below threshold class is given by:

$$P_1(k) = \Sigma_{i=0}^{k} p_i$$

To get the probability of being above threshold we can use the complementary probability: $P_2(k) = 1 - P_1(k)$. We now compute the average intensity of the entire image (in other words the global mean) as $m_g = \Sigma_{i=0}^{L-1} i p_i$

and we compute the cumulative mean, or average intensity, up to level k as: $m(k) = \Sigma_{i=0}^{k} i p_i$.
To compute the mean intensity value of the pixels assigned to class 1 or 2 we use the following relations:

$$m_1(k) = \frac{1}{P_1(k)} \Sigma_{i=1}^{k} i p_i$$
$$m_2(k) = \frac{1}{P_2(k)} \Sigma_{i=k+1}^{L-1} i p_i$$

We now consider the global variance(related to all pixels in the image) and the inter-class variance as:

$$\sigma_G^2 = \Sigma_{i=0}^{L-1} (i - m_G)^2 p_i$$
$$\sigma_B^2(k) = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2$$

And we can define as well the quality of the threshold as follows: $\eta = \frac{\sigma_B^2(k)}{\sigma_G^2}$
We can now rewrite σ_B^2 as $\sigma_B^2(k) = P_1 P_2 (m_1 - m_2)^2 = \frac{(m_G P_1 - m)^2}{P_1(1-p_1)}$ for k=0,...,L-1. Then the optimal threshold is the value k^* that maximizes η such that $\sigma_B^2(k^*) = max_k(\sigma_B^2(k))$.
We define the intra-class variance and the global variance as:

$$\sigma_{in}^2 = P_1 \sigma_1^2 + P_2 \sigma_2^2$$
$$\sigma_G^2 = \sigma_{in}^2 + \sigma_B^2$$

.
To summarize it:

- Compute the normalized histogram of the input image. Denote the components of the histogram by p_i, i=0,...,L-1
- Compute the cumulative sums $P_1(k)$
- Compute the cumulative means, m(k)
- Compute the global mean m_G
- Compute the between-class variance term, σ_B^2
- Obtain the Otsu threshold, k^*, as the value of k for which σ_B^2 is maximum. If the maximum is not unique, obtain k^* by averaging the values of k corresponding to the various maxima detected
- Compute the global variance σ_G^2 and then obtain the separability measure η by evaluating $k = k^*$

The Otsu's method can be combined with other techniques such as edge detection. It can be generalized to Non-global thresholding and multiple categories.

Edge detection

Compute the histogram and the threshold on the edge image (or combination of edge with the normal image). Then apply the threshold to the original image.

Variable thresholding

Process different regions of the image using different thresholds

Multiple thresholding

Subdivision into multiple region. We need to maximize inter-class variance: $\sigma_B^2 = \Sigma_{j=1}^{N} P_j (m_j - m_G)^2$.
It is the extension of the Otsu's method to N regions.

Clustering

Clustering is the task of grouping a collection of heterogeneous elements into sets of similar elements and to put dissimilar objects into different sets. (Recall of unsupervised learning).
We represent each pixel with a vector of features. This representation depends on the goal of the image analysis process we are implementing. The vector has multiple dimensions. Each pixel has its own feature vector. This vector contains all the measurements that may be relevant to describe a pixel, such as spatial position, intensity or brightness, color information or even combinations of them.
Segmentation by clustering means segmenting an image using clustering techniques. We need to provide the vector of features and we need to apply a suitable clustering algorithm. Clustering techniques often evaluate how similar two pixels are, which means comparing the corresponding feature vectors. To do so, we need a distance function, which becomes critical when multiple types of data are involved. We can choose from different distance functions:

- Absolute value or Manhattan: $d_a(\bar{x_i}, \bar{x_j}) = \Sigma_{k=1}^{D} |x_{i,k} - x_{j,k}|$
- Euclidean: $d_e(\bar{x_i}, \bar{x_j}) = \sqrt{\Sigma_{k=1}^{D} (x_{i,k} - x_{j,k})^2}$
- Minkowski: $d_m(\bar{x_i}, \bar{x_j}) = [\Sigma_{k=1}^{D} (x_{i,k} - x_{j,k})^p]^{1/p}$

We have two different approaches to cluster elements: Divisive clustering, which it starts with considering the entire dataset as the first cluster and then recursively split each cluster to yield a good clustering and need some form of cluster quality measurement, or Agglomerative clustering which start considering every single pixel as a cluster and then recursively merges each cluster to yield a good clustering and need also some form of cluster quality measurement.

K-mean

It is a simple clustering algorithm and it is based on a fixed number of clusters. The number of clusters, k, shall be provided to the algorithm. After the process, each feature vector is associated with one of the k clusters. The objective of the k-means is to partition the set Q of observation into k disjoint cluster set $C = \{C_1, ..., C_k\}$, where each cluster C_i has a centroid μ_i. The k-means objective function measures the distance between each data point and the centroid of its cluster. The goal is to minimize the error made by approximating the points with the center of cluster it belongs to, where d is the distance function: $min(\Sigma_{i=1}^{k} \Sigma_{x \in C_i} d(x, \mu_i))$.
Exhaustive search is computationally unfeasible since there are too many combinations. We need an euristic. We commonly use the iterative algorithm and it can be applied to vectors containing any set of features.
The algorithm is the following:

- Get k initial centroids either randomly chosen among the data points or build k clusters randomly assigning all the points to clusters and then compute the centroids.
- Associate each point to the "closest" centroid $C_i = \{x : i = argmin_j d(x, \mu_j)\}$ for i =1,...,k
- Compute the new centroids as $\mu_i = \frac{1}{C_i} \Sigma_{x \in C_i} x$
- Repeat steps 2 and 3 until the centroid do not sensibly move or we reached the maximum number of steps.

The k-mean is light and simple, computational complexity can be reduced using euristics and it has fast convergence. On the other hand, optimality is not guaranteed and the solution found depends on the initialization. The number of clusters k needs to be known in advance and forces spherical symmetry of clusters.
We can do the k-mean clustering based on the histogram or the pixel vectors. We can have different distance measurements such as intensity level difference, color channel difference or combination of position, color, texture, descriptor and so on.

Density functions

It is a method to derive a deeper analysis on the dataset. We start from the set of samples, and then we can design a kernel density estimation to get the desired density function. We can create a density function by choosing a kernel, centering it on each sample and the summing up all the contribution. We can define the kernel as follows:

$$K(x) = c_k k(||\frac{x}{r}||^2) = c_k k(\frac{||x^2||}{r^2})$$

For point x in the n-dimensional feature space. The kernel K integrates to 1 and has radius r. k(•) is a 1-dimensional profile generating the kernel and it is applied to all dimensions of the feature space. We can use several kernels such has:

- Uniform: $k_U(x) = \begin{cases} c_u \; ||x|| < 1 \\ 0 \; otherwise \end{cases}$
- Normal: $K_N = c_n exp(-\frac{1}{2}||x||^2)$
- Epanechnikov: $k_E(x) = \begin{cases} c_e(1 - ||x||^2) \; ||x|| < 1 \\ 0 \; otherwise \end{cases}$

The convolution with a given kernel of width r is expressed as the sum of translated kernel for each data point. A function of the N data points:

$$f(x) = \frac{1}{Nr^n}\Sigma_{i=1}^N K(x - x_i) = \frac{c_k}{Nr^n}\Sigma_{i=1}^N k(\frac{||x-x_i||^2}{r^2})$$

where x_i are the input sample and $k(\bullet)$ is the kernel function. The factor r^n normalizes by the number of dimensions the vector $x \in R^n$.
After having derived a density function we can find the major peaks and identify the regions of the input that climb to the same peak (such regions belong to the same region or cluster).
It is computationally complex due to the n-dimensional space.

Mean shift

It's a tool to find stationary points, or peaks. The main idea is to find peaks in high-dimensional data distribution without computing the distribution function explicitly. It implicitly models the distribution using a smooth continuous non-parametric model. We assume that the data points are sampled from an underlying density function. The direct PDF estimation can be made by means of functions like: $f(x) = \Sigma_i c_i e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$.
Data point density is an implicit description of the PDF value, and is a non-parametric estimation. To avoid to evaluate the whole density function we evaluate the PDF gradient. The mean shift is a steepest-ascend method.
The kernel density gradient estimation: $\nabla f(x) = \frac{1}{Nr^n}\Sigma_{i=1}^N \nabla K(x - x_i)$, where $K(x - x_i) = c_k k(\frac{||x-x_i||^2}{r^2})$.
$\nabla f(x)$ is the multiplication of $c_k k'(\frac{||x-x_i||^2}{r^2})$, and the derivate of the inner function is $\frac{2}{r^2}||x - x_i|| = y_i$. We now define g(a)=-k'(a), where the minus sign is used to express in terms of $(x_i - x)$. The derivative of $\nabla f(x)$ can be written as:

$$\begin{aligned} &\frac{2c_k}{Nr^{n+2}}\Sigma_{i=1}^N[(x_i - x)g(y_i)] \\ &= \frac{2c_k}{Nr^{n+2}}(\Sigma_{i=1}^N[x_i g(y_i)] - x\Sigma_{i=1}^N g(y_i)) \\ &= \frac{2c_k}{r^2 c_g}[\frac{c_g}{Nr^n}\Sigma_{i=1}^N g(y_i)][\frac{\Sigma_{i=1}^N x_i g(y_i)}{\Sigma_{i=1}^N g(y_i)} - x] \end{aligned}$$

where c_g normalizes the integral in the feature space when g is used as a kernel. We can now rewrite the previous equation as $\nabla f_k(x) = Af_g(x)m_g(x)$, which can be rearranged as $m_g(x) = \frac{\nabla f_k(x)}{Af_g(x)}$. This shows that the mean shift vector proceeds in the direction of the gradient.
The mean shift vector is $m(x) = [\frac{\Sigma_{i=1}^N x_i g(y_i)}{\Sigma_{i=1}^N g(y_i)} - x]$. The mean shift vector is the direction to follow to find a mode. The iterative procedure is the following: compute the mean shift vector m(x), move the kernel window by m(x) and stops if the gradient is close to 0.
Saddle points have zero gradient, so they are unstable locations and a small perturbation on them can cause the process to move away. Then we need to prune the mode by perturbation.

The mean shift vector magnitude depends on the gradient. Steps are smaller towards a mode, and convergence depends on the step size. We can optimize the mean shift by dividing the space into windows and then running the procedure in parallel.
The good features of mean shift are that it doesn't assume clusters to be spherical, it has one single parameter, it finds a variable number of modes and has a robust outliers. The negative features are that it depends on window size, and most of the times in not easy to find the best value. A bad choice of the window size can cause modes to be merged. It is computationally expensive and does not scale with the dimension of the feature space.

Mean shift cluster

The clustering criterion is to define attraction basin as the region for which all trajectories lead to the same mode. All data points in the attraction basin are merged into a cluster. Mean shift can effectively separate complex modal structure. It is also possible to preserve discontinuities by using the joint domain of spatial and color: $K(x) = c_k k_s(\frac{x_s}{r_s} k_s(\frac{x_r}{r_r}))$, where x_s are the spatial coordinates and x_r are the color coordinates.

Markov Random Fields

We want to see the segmentation as a per-pixel labeling task, where labeling means defining a set of labels, an energy function and assign to each pixel a label in order to minimize the energy function. Usually the labeling set is discrete and it is defined as follows $L = \{l_1, ..., l_m\}$, where $|L| = m$, L is ordered. There is also the option to have the label set defined over a continuous set.
Labels in an image are associated with the pixel locations. The whole set of pixel is called Ω. Given a pixel location p, its label h is assigned by means of a function: $h = f_p = f(p)$. The labels shall be chosen in order to minimize the energy function or error function. The energy function is composed of two terms:

- The data term, which depends on specific image features at the pixel location $E_{data}(p, f_p)$. It evaluates how good f_p fits on pixel location p and define a match criterion based on the image feature vector. We can use the L_2 norm or even the L_1.
- A smoothing/contiuity/neighborhood term which depends on the label of the neighboring pixels, where the neighbor is defined as A(p) (commonly 4 -neighbors are considered). $\Sigma_{q \in A(p)} E_{smooth}(f_p, f_q)$. It is a prior that favors the same label at adjacent pixels.

The balance between the two terms is crucial since strong boundaries should be detected (so the smoothing term cannot penalize them too much) and weak boundaries should not affect the labeling (a smoothing term that is too week leads to fragmentation). The energy function evaluated over the whole image is defined as:

$$E(f) = \Sigma_{p \in \Omega}[E_{data}(p, f_p) + \Sigma_{q \in A(p)} E_{smooth}(f_p, f_q)]$$

and it is called Markov Random Field. The model considers local interactions between adjacent pixels.
Interactions are modeled considering a graph, where its connections represent the links of mutual influence and in the MRF it is undirected. A MRF is used to define a minimization problem where the goal is to select the labeling which minimizes the energy function. When the labeling is perfect for a given pixel the data term is zero. If there are no transition between adjacent pixels, the smooth component is zero.

Belief Propagation

The MRF problem can be solved using the belief propagation algorithm which is based on message passing between neighboring pixels. The messages conveys information about labels and related costs. The algorithm runs several iterations, and at each iteration messages are delivered.
Consider the message sent by pixel p to pixel q, it contains local information at p and info provided by p's neighbors in previous iterations. At each iteration every p in q's neighborhood passes its message to q. Throughout the process, each node acts as q in iteration i and as p in iteration i+1.
A Message is a function mapping the ordered discrete set of labels,L, into an array of length m having, for each position, the message value which are non-negatives and real. Such function is denoted as $m^t_{p \to q}(l)$, which is the message value for label l sent from p to q at time t, and is a function of the label $l \in L$. The message update equation is the following:

$$m^t_{p \to q}(l) = min_{h \in L}(E_{data}(p, h) + E_{smooth}(h - l) + \Sigma_{s \in A(p) \setminus q} m^{t-1}_{s \to p}(h))$$

The interpretation is the following: Node q inquires all its neighbors asking their opinion about selecting l at q. Node p gets the question and to reply, it scans all possible h and tells q about the lowest possible cost when q takes l. The reply of node p depends on the data term at p for label h (that is minimized), the smoothing term if the label l and h are different, the information brought by the neighbors of p (including q) in the previous iteration. Node q is excluded from the computation since it cannot influence the opinion of p about q itself.
So far, we have considered the influence of p at q. This process is extended to every neighbor of q. At q we should also take care of the local data term. The total cost at q is: $E_{data}(q,l) + \Sigma_{p \in A(q)} m^t_{p \to q}(l)$, where m embeds the smoothness costs.
The previously discussed cost is the driver for the belief propagation algorithm.
The messages are initialized as:

$$m^0_{p \to q}(l) = min_{h \in L}(E_{data}(p,h) + E_{smooth}(h-l))$$
$$\text{where the related costs are: } E_{data}(q,l) + \Sigma_{p \in A(q)} m^0_{p \to q}(l)$$

Times updates are controlled by the equations previously detailed. The termination of the algorithm is often based on pre-defined number of iterations, but also other option would be possible such as observing the mean of all changes, but there is no guarantee in the convergence of the algorithm.
Upon termination, the final label at each pixel location is determined as: $j = argmin_{1 \leq i \leq m} c_i^{t_0}$, and if more than one minimum is found, favor labels assigned to adjacent locations.
Belief Propagation can be used for image segmentation, where each label defines one or more segments (Remember that multiple connected regions can share the same label). The seed points are often updated at each iteration. This algorithm is usually applied at multiple resolutions and takes the name of pyramidal belief propagation.

Features

Detecting and matching elements of the image is a key task in computer vision. Such elements are called features, which are meaningful parts of the image. The feature has two components: feature detection, which means finding a stable point, or feature description, which is a description of the surrounding area. We have an image as input and a set of points and a description of the region as an output. An ideal keypoint shall be stable an repeatable, invariant to transformation, insensitive to illumination changes and accurate.
The stability of a keypoint is measured by means of repeatability score on a pair of images. Given two images, the stability is defined as the ratio between the number of point-to-point correspondences that can be established and the min number of keypoints in the two images.
The descriptor is a vector representation of the local patch, and it can be seen as ideal if it is robust to occlusion and clutter, to blur, noise, compression, discretization, if it is discriminative, stable over changes in viewing angle and illumination and computationally efficient.
Matching features means evaluating the feature in two images and find similar features. Similarity is applied to the descriptor using a distance.
Several CV systems are based on features, such as motion detection, stitching and 3D reconstruction. Matching can be done in instance matching or object localization, to stitch image mosaics, or, scene reconstruction and structure from motion, place recognition, object detection, tracking such as following the patterns in video flows.

Corner and blobs Detection

Salient points, AKA keypoints, can be detected in many different ways; the first and most famous one is the Harris corner detector. The intuition is to consider a patch in an image and a shifted version of the patch. If the region is uniform then the two patches will be similar, otherwise they'll be different and we found a salient points. A corner is a region producing a large difference if the patch is moved.

Harris corners

Let's consider a given position (x_i, y_i) and a displacement $(\Delta x, \Delta y)$. Similarity is measured by means of the auto-correlation, a function of the displacement. No assumption on $(\Delta x, \Delta y)$ is made. A keypoint can be matched in any portion of the image. The auto-correlation is defined as follows:

$$E(\Delta x, \Delta y) = \Sigma_i w(x_i, y_i)[I(x_i + \Delta x, y_i + \Delta y) - I(x_i, y_i)]^2$$

where I is the image, $\Delta x, \Delta y$ the displacements and i goes over all the pixels in the patch. Now we can apporximate E using the Taylor series:

$$I(x_i + \Delta x, y_i + \Delta y) \approx I(x_i, y_i) + I_x \Delta x + I_y \Delta y$$

where I_x is the partial derivative over x of $I(x_i, y_i)$ and I_y is the partial derivative over y of $I(x_i, y_i)$. This holds for small displacements. Now we can substitute it in the auto-correlation function, and by neglecting the weights it yields:

$$E(\Delta x, \Delta y) = \Sigma_i [I_x \Delta x + I_y \Delta y]^2$$

We can rewrite everything in matrix form:

$$E(\Delta x, \Delta y) = [\Delta x \ \Delta y] \begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_y I_x & \Sigma I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

The auto-correlation matrix describes how the region changes for a small displacement. The matrix A is real and symmetric. Under these conditions, the eigenvectors are orthogonal and point to the directions of max data spread. The corresponding eigenvalues are proportional to the amount of data spread in the direction of the eigenvectors. By studying the eigenvalues we get information about the type of patch: for example, if both eigenvalues are small the region is uniform, if only one of them is large we have an edge and in case they are both large we have a corner
The weigths of the auto-correlation function were so far neglected; they can be introduced into our formulation, expressed by means of convolution:

$$\text{A}{=}w * \begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_y I_x & \Sigma I_y^2 \end{bmatrix}$$

We have two main choices for the weights: Box has 1s inside the patch and 0 elsewhere; Gaussian puts more emphasis on changes around the center.
Given the A matrix and the eigenvalues/vectors, how is a keypoint selected? We have several options:

- Minimum eigenvalues $[Shi,\ Tomasi]$
- det(A)-$\alpha\bullet$ trace$(A)^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$ $[Harris]$
- $\lambda_0 - \alpha\lambda_1$ $[Trigs]$
- $\frac{det(A)}{trace(A)} = \frac{\lambda_0\lambda_1}{\lambda_0+\lambda_1}$ $[Brown,\ Szleski, Winder]$

The Harris corner detector is invariant to brightness offsets: $I(x, y) \rightarrow I(x, y) + c$. It's invariant to shift and rotations but not to scaling.
Outside the Harris corner we have several other detectors: USAN/SUSAN corner detector which analyzes a circular window around the point. It doesn't involve derivative and it is able to detect edges and corners. It is also robust to noise.

USAN: Univalue Segment Assimilating Nucleus

It does a comparison between the nucleus,i.e the central point, and the pixels in the mask. It computes the portion of the window with intensity difference from the nucleus within a given threshold. If the USAN us smaller than half mask area: corner; if it is similar to half mask area: edge; if it is similar to the mask area: uniform region. The SUSAN is the smallest USAN.

BLOB features

A blob is a region where considered properties are different form surrounding regions; a property is constant inside the region, or at least approximately. Blobs are used in feature detection.

MSER: Maximally Stable Extremal Regions

They are connected areas characterized by almost uniform intensity, surrounded by contrasting backgrounds(blobs). MSER feature detector can be used as a blob detector. The algorithm is the following: Apply a series of thresholds, for example one per gray level; compute the connected binary regions; Compute some statistics for each region, for example area, convexity, circularity and so on; analyze go persistent each blob is.
Extremal refers to the property that all pixls inside the MSER have either higher or lower intensity than all the pixels on its outer boundary.

Scale Space

The goal is to analyze the image at multiple scales. The scale space transforms the original signal into a family of derived signals. The parameter t controls the location in the scale space: $f(x, y) \to f(x, y, t)$. The higher we go more the fine-scale details are gradually suppressed. We want ti suppress image elements to simplify the image elements such that we preserve just the strongest elements in the image. In the SIFT algorithm, that we'll see afterwards, we use the N-dimensional Gaussian kernel to smooth, so that the scale parameter is the standard deviation. To produce the scale space we use convolution with a variable scale Gaussian kernel, defined as:

$$g(x, t) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-(x_1^2+\ldots+x_N^2)/2t}$$

where g is a function $R^N x R_+ \to R$, and the variance t of the kernel is the scale parameter. The scale space is then represented by:

$$L(x, t) = \int_{\xi \in R^N} f(x - \xi) g(\xi, t) d\xi$$

where g is the Gaussian kernel described above.
The scale space has a relation with biological vision. Photo receptors in the retina can be modeled as a superposition of Difference of Gaussians.
Edge detection can be run for each value of t. We shall be able to select the best value of t for each edge and combine edges found at multiple scales. A measure of edge strength can be defined, for example we can measure the strength of edge response normalized by t. Strong edges can be found at all scales. The finer the scale the more accurate and spurious the edges are. At coarse scales, the edges are inaccurate but only strong edges are found. The combination of the detection leads to the best result. By defining a metric for edge strength we can find the strongest edges at all scales.

SIFT algorithm

Algorithm developed to extract invariant features from an image. It transforms image data into scale-invariant coordinates relative to local image features. The algorithm has 4 main phases: Scale-space extrema detection, key-point localization, orientation measurement and descriptor calculation.

SIFT scale space

SIFT subdivides the scale space into octaves, with each octave corresponding to a doubling of σ (Gaussian transform). Then it sub-divides each octave into an integer number, s, of intervals, so that an interval of 1 consists of two images, an interval of 2 consists of three images and so forth. It then follows that the value used in the Gaussian kernel that generates the image corresponding to an octave is $K^s\sigma = 2\sigma$ which means that $k = 2^{1/s}$. For example, for s=2, k=$\sqrt{2}$, and the input image is successively smoothed using standard deviations of $\sigma, (\sqrt{2})\sigma, (\sqrt{2})^2\sigma$, so that the third image in the sequence is filtered using a Gaussian kernel with standard deviation of 2σ. The preceding discussion indicates that the number of smoothed images generated in an octave is s+1. However, the smoothed images in scale space are used to compute differences of Gaussians which, in order to cover a full octave, implies that an additional two images past the octave image are required, giving a total of s+3 images. Because the octave image is always the (s+1)th image in the stack, it follows that this image is the third image from the top in the expanded sequence of s+3 images. The first image in the second octave is formed by downsampling the original image, and then smoothing it using a kernel with twice the standard deviation used in the first octave. Subsequent images in the second octave are smoothed using σ_2, with the same sequence of values of k as in the first octave. We then apply

the same to the remaining octaves. The layers in the scale space are combined by subtracting consecutive layers. This requires two additional images in the scale space to process the first an last image.(that's why s+3). Subtracting consecutive layers means evaluating the difference between two smoothed images (same smoothing but different smoothing intensity). Such filtering is called Difference of Gaussians and is represented by the function $D(x, y, \sigma)$.
We need the Laplacian of a Gaussian which is obtained using the gaussian filter:

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

and considering its laplacian to filter the image, which means filtering the input image with G(x,y) and then compute the laplacian of the resulting image. This procedure allows us to smooth the image, to remove noise at scales smaller than σ. It has zero-crossing, it's isotropic and has filter size nxn s.t $n < 6\sigma$. An approximations of it is the difference of Gaussians, and it is defined as follows:

$$D(x, y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

Key-point localization

The meaning is to search for maxima and minima of the Difference of Gaussians. We habe a comparison with the 8-neighbors in the current, previous and next scale level. It is done at all scales, which means scale independence. Sub-pixel accuracy by means of interpolation, we use Taylor series expansion up to the quadratic term. We do interpolation along x,y, σ.
Remember that SIFT is based on the laplacian, the the Hessian matrix provides a more detailed description:

$$H = \begin{bmatrix} \delta^2 D/\delta x^2 & \delta^2 D/\delta x \delta y \\ \delta^2 D/\delta x \delta y & \delta^2 D/\delta y^2 \end{bmatrix} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

Further processing is based on H. The Hessian matrix is used to discard points with a low gradient $|D(\hat{x})| < 0.03$. It discards edge points- require that both eigenvalues of H are large. It means that require that both curvatures are high. And this is how we refine the key-points. More scales levels are considered to get more points. It tends to become more unstable. The best value so far is for s=3. The output is a set of key-points with associated scales.

Key-point orientation

Each key-point comes with its scale. The Gaussian smoothed image closest to that scale is selected-named L. This process is independent from the scale. We then compute the gradient magnitude and orientation in the key-point neighborhood using L.

$$\text{Magnitude: } M(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$
$$\text{Orientation: } \theta(x, y) = \tan^{-1}[\tfrac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}]$$

Top achieve rotation invariance, the dominant local direction shall be measured. We build the histogram of gradient orientations: 36 bins of 10 degrees each-region around the key-point. We then look for the peak. To refine the peak location we fit a parabola to the 3 bins close to the peak. In case we have also peaks that have a value > 80% of the highest peak we create a new key-point with the orientation set to the secondary peak (We are basically duplicating the key-point). SIFT assigns multiple orientations to only about 15% of points with multiple orientations, but these contribute significant to image matching.

Key-point descriptor

SIFT evaluates a descriptor for each key-point in the image L corresponding to the key-point scale and considering coordinates that are rotated based on the measured key-point orientation. A region of size $16x16$ pixels is centered on a key-point, and the gradient magnitude and direction are computed at each point in the region using pixel differences. A Gaussian weighting function with standard deviation equal to one-half the size of the region is then used to assign a weight that multiplies the magnitude of the gradient at each point. The purpose of this function is to reduce sudden changes in the descriptor with small changes in the position of the function. The $16x16$ region is then divided into regions of $4x4$ pixels, so that we can evaluate the histogram of each regions in 8 bins of 45 degrees each. We then save the values. The descriptor is composed

by 16 regions where each one have 8 histogram values. We have 128 elements in total, so 1 byte is sufficient to store the information.
Additional details are considered by the SIFT algorithm: we can normalize the feature vector to unit length to enhance invariance to illumination. We can threshold to 0.2 all the components to enhance the robustness to large gradient magnitude.

Summary of the SIFT

- Construct the scale space
- Obtain the initial key-point
- Improve the accuracy of the location of the key-points
- Delete unsuitable key-points
- Compute key-point orientations
- Compute key-point descriptors

SIFT variations

PCA-SIFT

PCA is a dimensionality reduction technique. Reducing dimensions can be useful to work on more compact representations, reduce computational workload and to highlight the most important information hiding the details. PCA is an orthogonal projection of data onto a lower dimensional linear space that maximizes variance of projected data and minimizes mean squared distance between original data points and their projection. Find a low dimensional space that preserves as much information as possible, for example find the best planar approximation for a 3D surface. It is based on the eigenvalue decomposition of the covariance matrix. The largest eigenvectors, called principal components, are the basis of the new space. The first principal component points in the direction of the largest variance. Each other principal component is orthogonal to the previous ones and points in the direction of the largest vvariance of the residual sub-space.
In the PCA-SIFT the steps 1 to 3 are the same. The onkly step that changes is the descriptor calculation. It refer to a 41x41 patch at the given scale, centered around the key-point, normalized to a canonical direction. It has SIFT weighted histograms. In this variant, we concatenate horizontal and vertical gradients (39x39) into a longer vector. The length is 2 directionx39x39=3042. We normalize the vector to unit norm. We reduce the vector using PCA to 3042 to 36.

SURF

It stands for Speeded Up Robust Features. It speeds up computations by fast approximation of hessian matrix and descriptors using integral images. We can denote the integral image as $I_\Sigma(x, y)$, where each pixel represents the sum of all pixels in the rectangle between (0,0) and that pixel $I_\Sigma(x_i, y_i) = \Sigma_{i=0}^{x_i-1}\Sigma_{j=0}^{y_i-1} I(i, j)$. We use box filters which are composed of black and white rectangles, where the whites represent the positive weight and the black represents the negative. Box filters exploit the integral image, and the calculation is constant-time irrespective of the filter size. To find the key-points we compute the hessian matrix in the scale space, and we find the maximum of the determinant of $H = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix}$ We approximate L using the box filters, instead of using the Difference of Gaussians like in SIFT.
An alternative to smoothing the image with a Gaussian and subsampled to move along the pyramid, we can use filters of increasing size. The computational time does not depend on the filter size using the integral image.
Let us now consider a neighborhood of size 6σ. We evaluate the gradients in x and y using the Haar wavelets, where the responses are smoothed using a Gaussian. We plot the responses in a 2-dimensional space, we sum horizontal and vertical response and determine their orientation. We then quantize everything in bins of 60 degrees. For the descriptor we consider 16 regions as in SIFT, and for each one of them we sum the response for d_x and d_y, and we sum the modules for d_x and d_y. We normalize the vector and we get 4 elements per region such that the feature size is of 64.

It is faster than SIFT but is less robust to illumination and viewpoint changes wrt SIFT. They are both patented.

GLOH

The gradient local-orientation histogram is obtained by modifying the 4th step of the SIFT algorithm. It computes the SIFT descriptor using a log-polar location grid. It has 3 bins in radial direction, 8 in angular directions. It presents 16 orientations such that it has 272 orientation bins. We use PCA to reduce dimensionality to 128.

Shape context

The shape context descriptor uses a 3D histogram of the edge point locations and orientations. The edges are often extracted using the canny edge detector algorithm. The locations are quantized using log-polar coordinate systems: we use 5 bins for distance, 12 for orientation. We get 60 different combinations.
We accumulate the distribution of edge points in the log-polar bins b_k s.t $h_i(k) = count[q \neq p_i; (q - p_i) \in b_k]$. It has translation invariance: positions are relative distances. It is scale-invariant, which is obtained by normalizing all radial distances by the mean distance by point pair. It's robust to deformations, noise, outliers, but it is rotation dependent.

Local Binary Pattern LBP

It was first proposed for texture recognition. It compares the central pixel with surrounding samples and build a binary sequence of signs of differences. It selects a pixel and compares it with its 8 neighbors. It follow the pixels along a circle: if it is greater than the center one, assign 1, otherwise assign 0. We compute the histogram of such values and then we normalize it. A whole image can be analyzed divided into sub-windows.

BRIEF descriptor

The binary robust independent elementary features is based on Gaussian smoothing and pairs of pixels compared inside a window:

$$\tau(p, x, y) = \begin{cases} 1 \; if \; p(x) < p(y) \\ 0 \; otherwise \end{cases}$$

We build a vector with comparison output: $f(p) = \Sigma_i 2^{i-1} \tau(p, x, y)$. Vectors are compared using the Hamming distance (XOR). The fixed sampling patterns are of 128, 256 or 512 pairs. Random pattern provides the best performance. The best solution is given by an isotropic Gaussian distribution.

ORB

The oriented FAST and Rotated Brief is a FAST corner detector, where we add the rotation invariance of BRIEF. The orientation assignment is based in the intensity centroid wrt the central pixel.

Feature matching

Features in an image are often matched. Matching strategy depends on the application at hand. similar features can be in similar positions or not. There can be many matches or not. The first strategy is the maximum distance. We match against all features within geometric distance, but it is difficult to set the threshold. The second strategy is the nearest neighbor, where we consider only the nearest neighbor in feature space. A threshold is also used. The third strategy uses the nearest neighbor distance ration NNDR=$\frac{d_1}{d_2}$, where d_1 is the nearest distance and d_2 the second nearest.
To measure the performance we use the following indicators: TP, true positives, which are the number of correct matches; TN, true negatives, which are the number of correct non-matches; FP: false positive, which are the number of non-matches that were wrongly matched; FN, false negative, which are the number of matches that were wrongly missed.
We can define the Presicion and the recall, respectively as follows: precision=$\frac{TP}{TP+FP}$ and recall=$\frac{TP}{TP+FN}$

Face detection

The challenges we face include huge number of pixels, multiple locations and scales (all combinations should be evaluated). Face are unlikely events. The key element for an optimal face detector is to fast process non-face candidates and a very low false positive rate $< 10^{-6}$ is required to avoid a false positive image.

Viola and Jones face detector

We use the Haar feature, which are given by rectangular filters. The local feature are obtained by subtracting the sum of pixels in the white areas from the sum of pixels in the black area: $f(x) = \Sigma_i p_b(i) - \Sigma_i p_x(i)$. We have 2-, 3- and 4- rectangle features. We also have a huge number of features. They are coarse features, and they are sensitive to edges, bars and other simple structure. They are computationally efficient since they allow us to compute a large number of features and compensates for the coarseness. If we consider already a 24x24 patch we have 160k possible features, which makes impossible to make an exhaustive analysis. We then use a weak learner to work on the number evaluated by a Haar feature f_j. It sets a threshold on a single feature, and separates positive and negative examples slightly better than casual guesses. It is evaluated as:

$$h_j(x) = \begin{cases} 1 \; if \; p_j f_j(x > p_j \theta_j) \\ -1 \; otherwise \end{cases}$$

We then go towards the boosting phase, where we build a strong classifier by combining several weak classifiers. It is obtained by a weighted sum of weak learners:

$$h(x) = sign[\Sigma_{j=0}^{m-1} a_j h_j(x)]$$

The weights are selected depending on the classifier accuracies. Usually it is used the AdaBoost to select the features and train the classifier. The AdaBoost starts with putting all equal weights, and for each round of boosting we evaluate each rectangle filter on each example, we select the best threshold for each filter, we select the best filter/threshold combination and in the end we reweight the examples. At each round the best weak classifier is found, which is the best classifier after the effect of the previously selected classifiers. It has a computational complexity of O(MNK), where M is the number of rounds, N the number of samples and K the number of features.
The last step is the cascade of classifiers. The classifier is a combination of weak learners, and it can become more efficient if weak learners are divided into different stages. Each stage is applied in sequence (cascade) and each stage acts as a filter, which means that the first stage that discard a sample prevents the subsequent stages to work. False negatives cause a failure while false positives are acceptable at high rate. First filters shall be as fast as possible. Training several classifier with an increasing number of features until the target decision rate is reached is the used approach. We re-weight training samples after each stage giving an higher weight to samples wrongly classified in previous stages. The classifiers are progressively more complex and have lower false positive rates.
The structure used by Viola and Jones had 32 stages and 4297 features. A monolithic classifier has similar accuracy bu there is a 10x difference in processing time.

High-level vision

Object recognition is a general term to describe a collection of related computer vision tasks that involve identifying objects in images or videos. Image classification is the task of associating one or more categories to a given image. We have as an input an image and as output one or more categories. Remember that the set of labels is already defined. Object localization is the task to localize an object in an image. We find the bounding boxes in an image and we associate a bounding box with a category. We know also the position of the object. Object classifications tells us what while object detection tells us what and where (this last one is really uncommon). Semantic segmentation is the task to assign a class to every pixel. We divide the image in multiple blobs. Each segment is associated with a label. Instance segmentation is the task to detect each object instance separately.
All the tasks discussed above shall cope with different camera positions, perspective deformations, illumination changes and intra-class variations.

Template matching

A template is something fashioned, shaped or designed to serve as a model, something formed after a model or a representative instance. We want to find instances of the template in the image, a similarity measure should be chosen. The hard part comes when we have to deal with template variability, deformable objects, image device properties, viewpoint changes, affine transforms, noise and illumination.
Simple differencing the images between each other doesn't always provide a reliable result.

Correlation-based

We have a template T: a rigid object, often a small image. T is placed in every possible position across the image: the basic approach doesn't involve rotation and scaling. We do a comparison with the pixel values, features and edges or gradient orientation. We use different similarity metrics:

$$\text{Sum of squared differences(SSD)}:\Phi(x,y) = \Sigma_{u,v\in T}(I(x+u,y+v)-T(u,v))^2$$
$$\text{Sum of absolute differences(SAD)}:\Phi(x,y) = \Sigma_{u,v\in T}|I(x+u,y+v)-T(u,v)|$$
$$\text{Zero-mean Normalized Cross-correlation(ZNCC)}:\Phi(x,y) = \frac{\Sigma_{u,v\in T}(I(x+u,y+v)-\bar{I}(x,y))(T(u,v)-\bar{T})}{\sigma_I(x,y)\sigma_T}$$

where $\bar{I}(x,y)$: average on window; $\bar{T}$: template average, $\sigma_I(x,y), \sigma_T$: standard deviation on image window and on template.
The Hough transform works for more complex shapes, and the general equation is g(v,c)=0, where v is a vector of coordinates and c a vector of coefficients. The parameter space might have high dimensionality (Remember the equation of the circle).
How can we cope with template matching weak points? We need to deal with illumination changes (we can use edge maps instead of images or use ZNCC to subtract th uniform illumination component), we need to deal with scale changes (We can use matching with several scaled versions of the template, or using the Hough transform, or work in the Scale space), and we need to deal with rotation (we can do matching with several rotated version of the template, or using the generalized Hough transform).

Bag of Words

It is an approach taken from document analysis and it allows us to manage slightly changes in the image such as perspective and so on. It is useful for image and object classification and it is designed to be invariant to

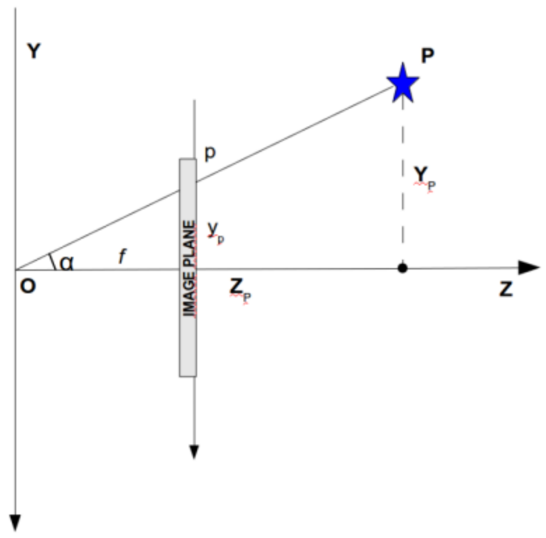
several factors such as viewpoint and deformation. It decomposes complex patterns into semi-independent features. We have a decomposition of the image into words/features. We then use the histogram representation to see which words appears the most in the image.
It can be used for image classification. The words can be represented using features which exploit discriminative and invariance properties, and reuses an efficient description.
We first extract the features such as key-points and descriptors, then we cluster in the feature space with a clustering algorithm,e.g k-means. We then use codebook generation, where each cluster generates a representative sample, which can be for example the centroid. We then evaluate the occurrence of each word in the codeword and we do a classification based on the histogram to do the image classification.

The pinhole camera model

We want to describe how an image is created. We need and object, a light source and a sensor. Since the light rays tend to propagate everywhere we use a pinhole to select just a few rays such that they are gonna be detected from the sensor. The selection of the rays depends on the size of the hole. We get a perfect definition if only one ray per point reaches the sensor. This happens just if the hole is a point, that's why we have the pinhole camera model.
We can construct a simplified model of the camera using the optical center, i.e the location of the pinhole, the image plane, and the optical axis which is perpendicular to the image plane. The intersection between the image plane and the optical axis is the principal point and the focal length is the distance between the optical center and the image plane. The system is right-handed and we have a reference system for describing the camera and on to describe the image plane.

Projection geometry

We need to describe the geometry of projection quantitatively. The first element is the relation between the two reference systems: 3D point in the world seen from the camera and the 2D point on the image plane. Let's consider a point P and its projection p. Now consider a plane that is parallel to the image plane, in front of the optical center and at the same distance f from the optical center. We do this so that we have the same geometrical relation and we avoid the upside down effect. We move to this plane for deriving the geometrical description. We use the similar triangle rule to derive the following relations: $\frac{Y_p}{y_p} = \frac{Z_p}{f}$, $\frac{X_p}{x_p} = \frac{Z_p}{f}$ and $tan(\alpha) = \frac{Y_p}{Z_p}$. Projecting points on a 2D surface causes the loss of the distance information. We can



rearrange the equations in matrix form using the homogeneous coordinates. The equations can be rearranged as:

$$x = fX/Z,\ y = fY/Z \Rightarrow Z\left[x//y//1\right] = Z\begin{bmatrix}\frac{fX}{Z}\\ \frac{fY}{Z}\\ 1\end{bmatrix} = \begin{bmatrix}fX\\ fY\\ Z\end{bmatrix} = \begin{bmatrix}f & 0 & 0 & 0\\ 0 & f & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}X\\ Y\\ Z\\ 1\end{bmatrix}$$

Which can be written as $\tilde{m} \simeq P\tilde{M}$, where P is the projection matrix. When f=1 we obtain the essential perspective projections. This represents the core of the projection process.
We need to map points projected onto the image plane i the coordinates used for pixels. From (x,y) to (u,v). The transformation can be defined considering the coordinates of the principal point to be (u_0, v_0). The origin is in the top-left corner. metric distances are converted to pixel using the pixel width w and height h. It evaluates the coordinates of the principal point in pixel. The conversion factors are usually defined as $k_u = 1/w$ and $k_v = 1/h$. Mapping form (x,y) to (u,v) is obtained by translation and scaling:

$$u = u_0 + \frac{x_p}{w} = u_0 + k_u x_p$$
$$v = v_0 + \frac{y_p}{h} = u_0 + k_v y_p$$

We can combine the mappings from 3D to 2D image plane and from image plane to pixels by substituting the last equation into the projection equation. We obtain the two following equations:

$$u = u_0 + fk_u\frac{X_p}{Z_p}$$
$$v = v_0 + fk_v\frac{Y_p}{Z_p}$$
where $fk_u = f_u$ and $fk_v = f_v$

and the projection is now expressed as $P = \begin{bmatrix}f_u & 0 & u_0 & 0\\ 0 & f_v & v_0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix} = K[I|0]$, where K is the camera matrix and $\tilde{m} \simeq P\tilde{M}$ still holds since it is just a different formulation for P. The camera matrix involves the following parameters: k_u, k_v, u_0, v_0, f, which are called the intrinsic parameter since they define the projection characteristics of the camera. Highlight: f_u, f_v embed three parameters. So far we have mapped world to image planed and image plane to pixels. However, a different reference frame can be defined on the world and it is always defined as roto-translation. A roto-translation in 3D homogeneous coordinates is expressed as $T = \begin{bmatrix}R & t\\ 0 & 1\end{bmatrix}$, and the correspondence becomes $\tilde{m} = PT\tilde{M}$. The roto-translation matrix T involves 3 parameters for translation and 3 for rotation. This parameter are called extrinsic parameters and they define the relation between the camera and the world. The whole projection process involves 4 reference systems and 3 transformations.
The inverse projection is kind of hard, since it is not possible to project from 2D back to 3D and the pixel quantization. We can invert the projection if we accept as a result the direction of the object, not the 3D position or we have additional constraints providing the location on the line. We can invert the projection if we neglect the quantization effect: the pixel location and the projected point are considered the same, which is acceptable for high-resolution sensors.

Camera and lenses

The major limit of the pinhole camera is the trade-off between sharp images and light intensity. A more complex system ca provide sharp images without needing a pinhole. We can then add a lens which has the main axis lying on the optical axis and a center lying in place of the pinhole. The lens is thin because we can neglect the lens width with respect to the curvature radius, we can ,model the lens as a 2D plane where every deviation occurs. The lens deviates the light and focuses the rays. The rays that passes through the center are not deviated, while the one that are parallel to the lens axis are deviated through the focal point. The focal point is at a distance defined as the focal length. By tracing the two rays, we can determine the location of the image point. The thin lens equation relates the distances between object and lens, d_0 and between lens and image, d_1, depending on the lens focal length, f: $\frac{1}{d_0} + \frac{1}{d_1} = \frac{1}{f}$. The points at a given distance are in focus, while all the points from other distances are not, generating a circle of confusion. We can then add a barrier, to reduce the circle of confusion, in front of the lens. We now have another meaning for the focal length: distance at which parallel rays intersect, for the thin lens model, and distance between pinhole and sensor for the pinhole camera model.

The field of view of a camera is the angle perceived by the camera. The angle α is the angle under which a point P is seen. The maximum value for α is half of the field of view (FoV). The field of view depends on the sensor size and the focal length: $\phi = arctan(d/2f)$.
Real lenses are affected by additional effects such as distortion, chromatic aberrations or other minor effects. A non-distortion lens is often complex to design and to build.
Distortion is a deviation from the ideal behavior described so far, and it can have two patterns: radial or tangential.

Radial distortion

The radial distortion depends in the distance of the distorted point from the image center. It can be pincushion or barrel according to the distance we have from the subject. The radial distortion can be analyzed by means of distortion patterns, which can be experimentally measured, analytically described if the lens structure is know in detail. The distortion chart shows the way the displacement acts according to an arrow based diagram. Camera models commonly consider a polynomial approximation for the radial distortion: it is just a mathematical model and not a physical one. It is often modeled as a correction: the corrected point depends on the distorted point:

$$x_{corr} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corr} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Tangential distortion

It is caused by the non-ideal alignment between lens and sensor. It looks similar to a perspective effect on the sensor and it is usually negligible. We can model it as:

$$x_{corr} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{corr} = x + [2p_1(r^2 + 2y^2) + 2p_2 xy]$$

Chromatic aberration

The dispersion is given by the refractive index depending on the wavelength. It modifies the ray blending and lens focal length: $f(\lambda)$. The color fringes near the image edges.

Real camera

A real camera makes us losing angles, distances and parallel lines, while straight lines are preserved. The length cannot be trusted since there might be perspective which can trick the perception we get through the image. According to the pinhole size we can get a more blurred image (bigger pinhole) or a sharper one (smaller pinhole). The lens allows us to gather more light but it need to be focused. The focal length is not the only element to change the subject size. We can compensate a change in focal length by moving the viewpoint, but a change in the background occurs. Telephoto makes it easier to select background, a small change in viewpoint is a big change in background.

Sensor

A sensor substitute the film. The photon get absorbed and then they are measured through conversion into electrons. We have two main types of sensor, the CCD and the CMOS. The CMOS are more used than the other since they are easier to integrate. The CCD have one line of pixels, but it allows to get high-quality pictures. CCD and CMOS do not differentiate on wavelength of photon energy, they measure only the total energy. They are essentially greyscale sensors. The color are sensed by: 3-chip color, which split light and with filters they separate the R,G and B components of the image; Single chip color which uses a single image with filters; or, chip penetration depending on the wavelength.
The Bayer Color Filter Array or mosaic was used to get colored images, but it needs interpolation to provide complete color info at each pixel. The 50% of the sensor is green since is the most perceived color. The Bayer was supposed to be replaced by the Direct image sensor which doesn't need interpolation and does not have information loss. It uses different wavelength to penetrates on different layers.

Light

The exposure tells us how much light enters the camera: we have two parameters, the aperture which regulates the con eof light that enters, and the shutter speed, which tells use the time frame in which the light can enter. The aperture is the diameter of lens opening, and it is expressed as a fraction of focal length, f-number: $f/2.0$ is the usual standard to express the aperture, it doesn't matter the diameter in this case cause two cameras with the same aperture receive the same amount of light, independently from the diameter of the aperture. When the focal length is doubled the amount of light gathered is divided by 4, and that's why f-stops are used instead of metric aperture. Small values of f means a large aperture. The typical number we have are expressed in function of the $\sqrt{2}$: f/4,f/5.6,f/8,f/11,f/16,f/22,f/32.

Shutter

It is different from the aperture. The sensor/film is usually protected from light. A picture is acquired when the shutter opens and closes, exposing the sensor/film. The exposure time is the time that the sensor is exposed to light. Typically they are: 1/60s,1/125s,...,1/4000s. Video cameras use electronic shutters. The effect of shielding and un-shielding is obtained by electronic control. We have 2 key-parameters: the exposure time, the time needed for acquiring an image, and the framerate, the time distance between two consecutive frames.

Reciprocity

Given a needed amount of energy(light), the same exposure is obtained with an exposure x2 and an aperture area /2: f-stops progression: $\sqrt{2}$, shutter speed progression: 2. Several pairs of shutter speed/aperture provide the same amount of light. An f/16 with an exposure time of 1/8s is closer to a pinhole, while an f/2 and an exposure time of 1/500s is closer to a freeze motion. The choice on the shutter speed changes the effect on the picture, if it has more motion blur or motion freeze, or camera shake effect. While the choice on the aperture influences the depth of field and the diffraction. Longer exposure means more light and more motion blur, while shorter time means less light and freeze motion. The smaller the aperture the less light we have and the more diffraction we get.

Depth of field

The aperture controls the depth of field: a smaller aperture increases the range in which the object is approximately in focus. The depth of field is the depth at which we can neglect the presence on the circle of confusion.

Camera calibration

How can we know the intrinsic and extrinsic parameters of the camera? we have to use calibration, which is the process of estimating the camera parameter. The intrinsic calibration only including the distortion parameters or the intrinsic+extrinsic are the two possible options. Calibration is needed if we want to measure the projection characteristics of a camera. The reference with world coordinate system is not needed, but the calibration process anyway links 3D object and projection. Only the intrinsic parameters are provided. A dedicated process can be used to evaluate the camera position and orientation in world coordinates. The camera orientation is expressed by means of three angles: Yaw, Pitch and Roll. Other applications include relating points in the image with the real world, for example, determining the distances with respect to the camera with additional constraints.
The general process to calibrate a camera is to take an object of known shape and appearance, the take pictures of it and in the end analyze the projection process. In principle, the calibration patter can be any object of known shape and dimension. The object shall be found in the image, for example a shape that is easily recognizable. Usually we use a checkerboard. The points used fo calibration are the square corners.
The process starts with collecting N images of the pattern, and for each image we list M 3D corner position in the pattern reference system, and then we find the corner position in the image reference system. We initialize the intrinsic calibration parameters to default values: for K: $f_u, f_v, u_0, v_o \rightarrow \theta_K$, and for the distortion: $k_1.k_2, k_3, p_1, p_2 \rightarrow \theta_d$. We then solve the non-linear least square problem:

$$min_{\theta_K,\theta_d}\Sigma_{i=0}^{N-1}\Sigma_{j=0}^{M-1}||K[I|0]T_i\tilde{P}_{i,j}-\tilde{p}_{i,j}||^2$$

where N is the number of images and M the number of points per image. Unfortunately the minimization process is doe over a large set of parameters, and can hardly provide a good result. Good initial guesses would be very desirable. To help the calibration process with a good initial guess we can exploit homography. The transformation from the object to the image plane can be represented using homography: it is possible only if the object is planar and if we neglect the distortion. Using a homography simplifies the mathematical description and allows us to have good initial guesses.
Each view of a planar object can be represented by a homography: the checkerboard corners are constrained by the homography: only four points per view are free: then we need 8 constraints, 4 per x and 4 per y, for each view to calibrate the camera. More points are useful for measuring the distortion.
Neglecting distortion we have 4 or 5 intrinsic parameters cause we can use just f_u and f_v neglecting f, and 6 extrinsic parameters. We need a minimum of 2 views. The calibration process is handled by a minimization and it is unstable. A larger number of views is needed in practise, like 10 or more.

Deep Learning

It is a field of computer science which investigates how it is possible to learn from data and being able to make predictions from them. The input is a dataset of input and target pairs. We want to find the best function which approximates the unknown function f. The best function, f^* is parametric and the optimal parameters are needed to be found. We have two categories of learning: supervised where the desired outcome is available or unsupervised where the desired outcome is not provided.
Non-linear phenomena can be linearized using a kernel function, which is easy to implement and relies on simple models. The kernel is often application-dependent and hard to generalize.
In computer vision we want to have as input a domain-specific representations generated through CV algorithms, and we make classification from these representation.
Machine learning build on image representation provided by some algorithms, while deep learning learns data representations, but is very effective in learning patterns, the data are represented by mean of a hierarchy of multiple layers, but need a huge amount of data to be trained. In deep learning the feature extraction is implemented in the model, while in machine learning we need to extract the features beforehand.
Handcrafted features might be incomplete and require a lot of human work for designing and validation. The learned features are adapted to the input data and the learned representations work for both supervised and non-supervised applications. The data representation and classification into one single network is one of the deep learning advantages.
A Deep Neural Network (DNN) is a composition of several simple functions, called layers:

$$f(x_i) = f_{\theta_l} \circ f \circ f_{\theta_{l-1}} \circ h \circ ... \circ f_2 \circ f_1 \circ h$$

where h is the activation function and f is a function depending on the specific layer. Each layer is made of a set of neurons and includes non-linear elements. If all the layers all linear then, we can collapse everything into simple matrix multiplications. The hierarchical topology generates a hierarchical abstraction.
The input to the neurons of one layer are the output of the neurons in the preceding layer in the case of feedforward networks. The output of a neuron is obtained through an activation function applied to a weighted average of the inputs plus a bias. There are several different types of activation functions. The value of a neuron, a_i is obtained through the following relation:

$$a_i(l) = h(\sum_{j=1}^{n_{l-1}} w_{ij}(l)a_j(l-1) + b_i(l))$$

Where h is the activation function, b the bias and l the layer the neuron we are considered is. There are the mathematical definitions of the activation function such as sigmoid, tanh and relu. I'm omitting them since we have seen them in 4 different courses :)
Training a network need the definition of a cost function. We need to propagate the data through the network and we have to measure the error between the desired and actual values using the cost function. In the end we need to update the wights using a backpropagation algorithm.
The output layer depends on the classification problem we are solving, for instance, if we have two classes, we have different ways of representations: one value describing the class or two values describing the score of each class. It can be generalized to N classes.

Fully connected network

A fully connected multilayer feedforward network ha severa layers, no loops and a high number of connections. It is typically used as a classification network, where the number of output neurons is the number of classes ad the max value in the outputs tells us which class should be chosen.

Convolutional Neural Network

It focuses on the input data structure. We see an image as a 2D matrix where the spatial neighborhood is important. Interesting features are local, shift-invariant and deformation-invariant. All of this should be taken into account in the formulation of our function f. A CNN has a local connectivity: receptive field for each pixel. It presents shared weights due to spatially invariant response. It presents multiple feature maps and we can do subsampling through pooling.
Due to local connectivity the number of weights to train is smaller since only neighboring nodes are connected. All the nodes in the next layer share the same parameters (tied weights), but operate on a different input window. Multiple feature maps are learned, and two different units can compute different function since they both operate on the same input data windows.
The pooling layer is used to reduce the resolution and initially was introduced to reduce the computational complexity. It adds some deformation invariance. The one mainly used is the maxppoling since is fast, efficient and shows strong results.

Regularization

It can be done through:

- Dropout: randomly drop units during training and each unit is retained with fixed probability p.
- Early stopping: we use the validation error to decide when training should be stopped. We stop when monitored quantity is not improved after n subsequent epochs, where n is called patience.

Transfer learning

The concept is to solve one problem and then apply part of the solution to a different but related problem. The motivation is to exploit the trained network since it doesn't need to train from scratch. This allows us to make things faster and allowing us to use small data sets. The simplest pattern is to take a pre-trained network, resetting the last layers, freeze the deeper layers and train the network. We can use several other patterns and combinations; for example we can exploit the output of a pretrained network as a feature vector, which is useful when the output stage provides multiple values; or we can exploit the features provided by the inner layers,but due to the high dimensionality of the feature some reduction might be needed.
Fine tuning is slightly different than what we have just seen. Optionally we can reset or remove part of the network or resume training on the new dataset. It needs a larger dataset with respect to transfer learning.