Business
Services

Technical studies and benchmark

A survey of multi-object detection algorithms

Table of contents

0. Introduction

0.A. Context

Detection of features in images has become something companies are eager to set up. Many tasks currently performed by humans are deconstructed and carefully studied in order to understand what makes it so complex to automatize them. Hopefully computers are increasingly becoming faster and more performant, enabling the release of algorithms and smart networks that can recognize multiple objects in images and videos from up to a thousand categories of objects.

0.B. Design brief

Orange Applications for Business plans to develop a solution for identifying features in images from different use-cases, e.g. counting humans in public transport, or identifying free lots, whether it is in warehouse shelves or in a parking. Although it could look simple at first sight to binarily classify features, many constraints add up to the classical image datasets on which these algorithms and networks train. Such constraints can be:

- Changes in the light exposure, contrast, or position of the camera. The designed solution shall be robust and generic enough to handle all those changes.
- The period at which the solution shall be executed. As this will depend on the use case, the goal is to conceive a solution as fast as possible.
- The classification duality. For each pixel of the image, it will be determined whether it belongs to the target class or doesn't.
- The mandated execution latency of the solution. This latter shall not run continuously because of the useless data and resources consumption.
- The initialization needs. Since most use-cases aim at rolling out the solution on a substantial number of devices, it is crucial to shrink the initialization needs as much as possible.
- Reuse of video equipment already deployed for other services and/or hardware resources of cameras, that do not come with GPUs. This prevents the processing of data on the spot and implies the usage of remote hardware resources (cloud computing).
- Data anonymization asked by the European referential for customer privacy. The solution shall either ensure the complete anonymization of data, such that it is impossible to identify anyone from the data, or not store any sensitive information.

0.C. Objective

The aim of this report is to overview the existing image processing algorithms and machine learning-based models, and address whether they could be used for designing the solution the company is looking forward to design. This report will also cover the importance of auxiliary processing and present some widely-used quality metrics that could be computed to measure the performance of the solution to be conceived.

1. Image processing approach

1.A. Introduction to image processing

1.A.1. History of image processing

Digital image processing was made possible thanks to last centuries' breakthroughs in physics, informatics and imaging techniques, and aims at using computers to perform modifications in images (especially enhance, isolate or extract some structure of interest). Image processing's application field is wide, going from defense to surgery. Image processing is the generic term for any method or algorithm aiming at using or modifying images (therefore it theoretically includes deep learning) but for the sake of simplicity, we consider that in the scope of these studies, image processing refers to any tool that is not based on artificial intelligence.

1.A.2. Concept of image segmentation

Segmentation is a subset of image processing and aims at extracting features of interest in images. Several types of segmentations exist, which serve different purposes:

- Semantic segmentation looks for consistency in adjacent objects. On Figure 1 below, the algorithm recognizes the continuation of the chairs even if the table cuts the image in two parts. However, since the chairs are close enough to each other, the algorithm does not recognize the objects' uniqueness and consider them as a whole.
- In boundary segmentation, the algorithm looks for **uniform areas** in the image and is not able to **grasp the continuation across zones of discontinuity** (for instance, in Figure 1, the chairs in the left of images are segmented as two different parts around the table).
- In semantic instance segmentation, principles of semantic segmentation are kept, and the idea of **uniqueness** is added. The algorithm is able to detect the very little distance between the objects and segment them individually (see on the right image in Figure 1).

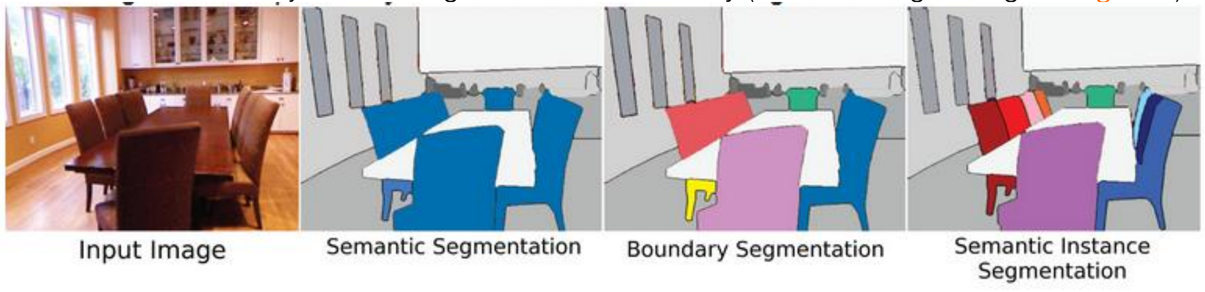


Figure 1 - Different types of segmentation.

1.A.3. Interest of image segmentation

Segmentation is something that can easily be implemented based on the images features (such as light, contrast, histogram, and so on), plus it generally depends on mathematical concepts that are fast and easy to implement. This means that having a good *a priori* knowledge of the set of images is essential, but once it is guaranteed, image processing-based segmentation is a very efficient way to extract features quickly. On the other hand, the elementary blocks of segmentation algorithms are based on mathematical concepts and therefore are implemented in almost all image processing libraries.

1.B. Elementary techniques

There are few pipelines in the literature that are dedicated to detecting several objects in images. As said before, these pipelines are strongly based on the images features and therefore depend on the use case. For example, it is difficult to develop a solution uniquely based on image processing methods that would work for both daily-light and night light-exposed pictures, since the elementary bricks often rely on the frequencies variation (see Figure 2 below).

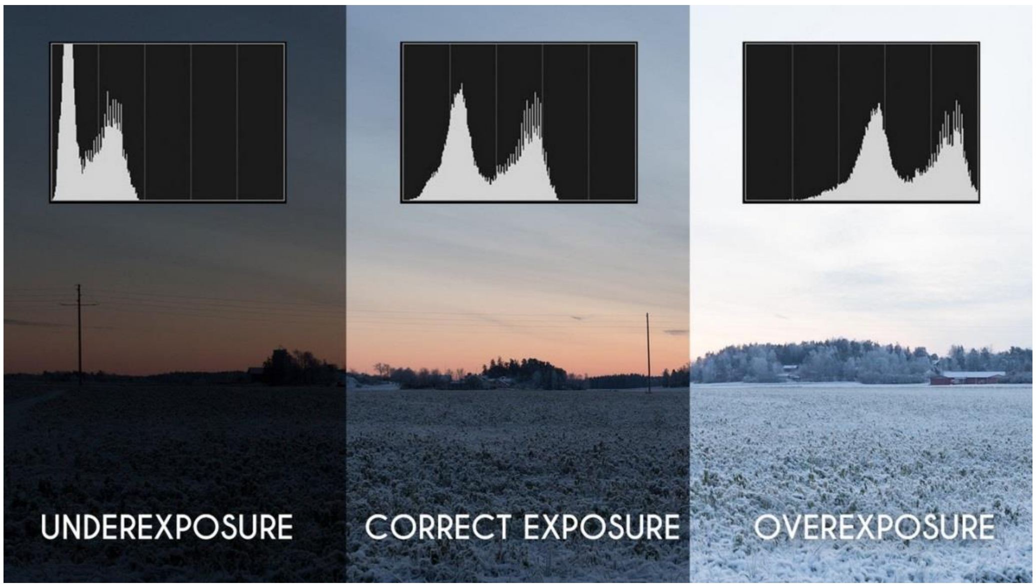


Figure 2 - When the image is underexposed (like during the night), the associated histogram is mainly composed of low frequencies and the contrast is low, when an overexposed image gives an histogram filled with high frequencies.

In 1.C., the few existing methods for multi-object detection will be described. Before this, an overview of the elementary bricks that constitute any segmentation pipeline exposes their strengths and specific usages:

- Contour extraction,
- Noise reduction,
- Morphological mathematics,
- Template matching,
- Hough transformation,
- Histogram equalization.

Image processing is a vast subject, therefore establishing a complete survey of all existing techniques is not possible. On the other hand, most pipelines are mainly based on common grounds, only their parameters are set differently.

1.B.1. Contour extraction

Contours (or edges) can be simply explained as a curve joining all continuous points (along the boundary), having the same color or intensity. Contour extraction is the result of a specific filter applied to the image of interest. Considering the image I and the filter h, applying h on the image I gives the image F described as below:

$$F(x,y) = \sum_{a,b} h(a,b) \cdot I(x+a, y+b) \qquad \textbf{(1)}$$

Generally, the filter kernel (i.e. the size of the filter) is chosen odd so as to define the center of the filter: for instance, if the filter is of size (2n+1)x(2n+1), then indexes *a* and *b* go from *–n* to *+n*.

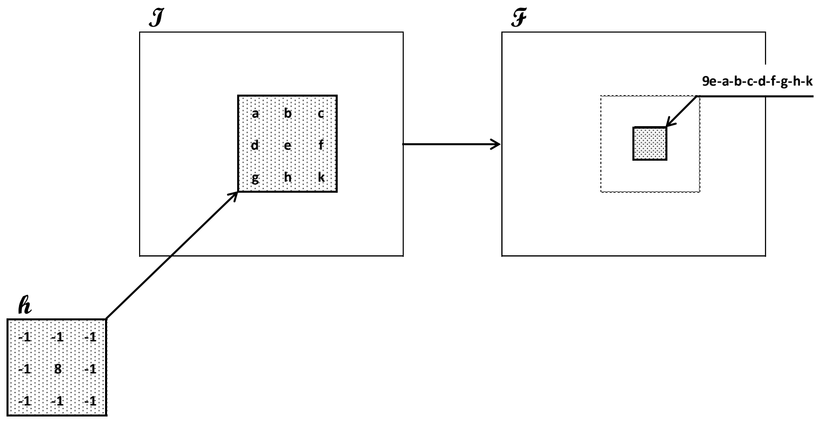


Figure 3 - Linear filtering by a contour extraction pattern.

In an image, low frequencies are associated with slow intensity variations (uniform areas), and high frequencies with fast variations (edges, noise). Generally speaking, contour extraction patterns are high-pass filters which enhance edges and noise in the image. Usually, contour extraction filters are used once the image has been smoothed with a noise-reduction filter.

Canny edge detector is an operator developed in 1986 that uses a multi-stage algorithm to detect a wide range of edges in images [1]. Gaussian smoothing is used to remove noise, then the gradient magnitude is computed to localize edge features. Non-maximum suppression is involved so as to remove spurious features and thresholding yields a binary image [2] where the maximum number of edges is found while trying to reduce the number of “false positive edges” (i.e. the edges created by noise in the image). Canny detector is one of the most popular algorithms for edge detection and is quite accurate [3].

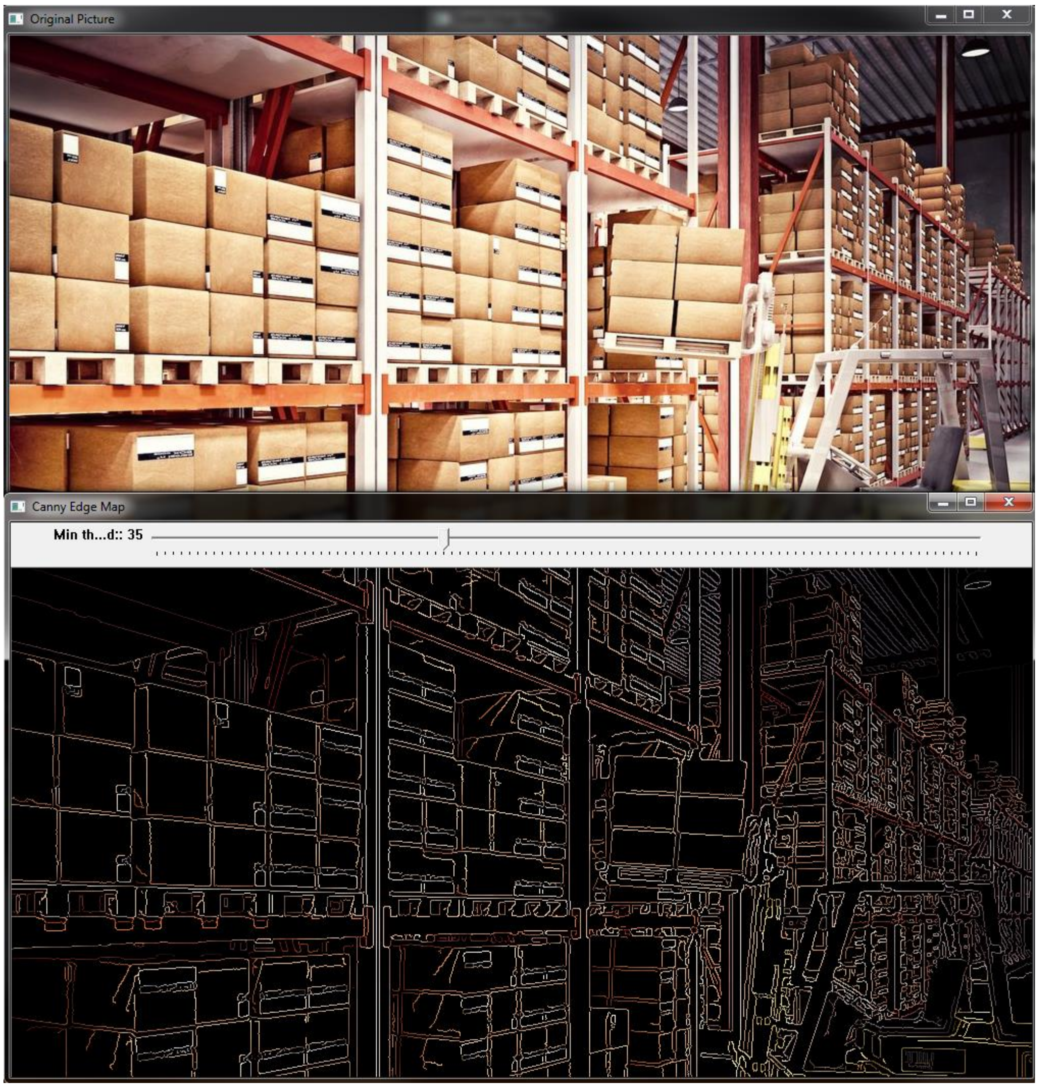


Figure 4 - Result of Canny edge detection with a threshold of 35 in the use-case of a warehouse full of packets.

The Sobel operator creates an image emphasizing edges and therefore is also an important feature for edge detection [4]. This operator created in 1968 computes an approximation of the gradient of the image intensity function. Convolution is used to calculate approximations of the derivatives, which give indications about horizontal and vertical changes in the image [5].

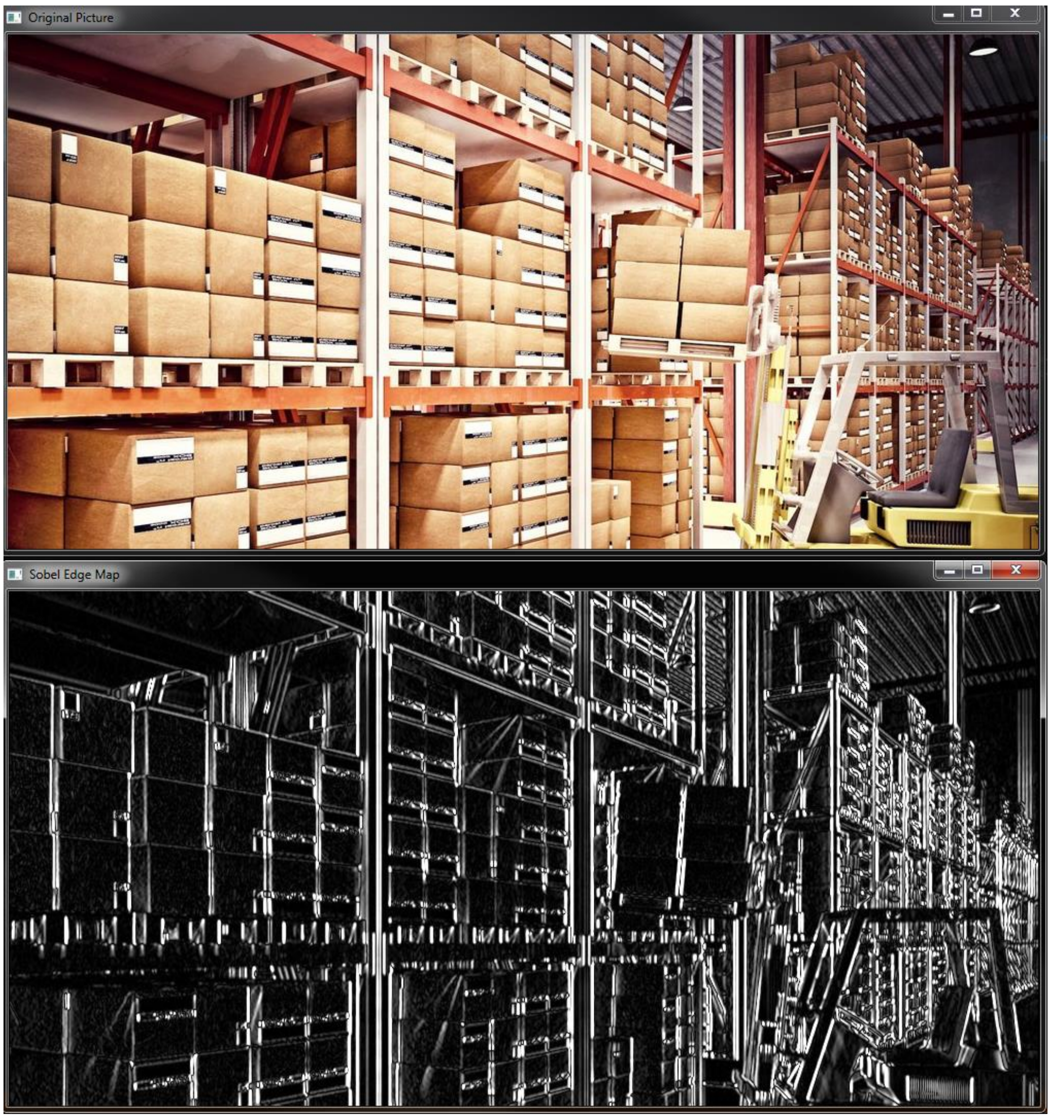
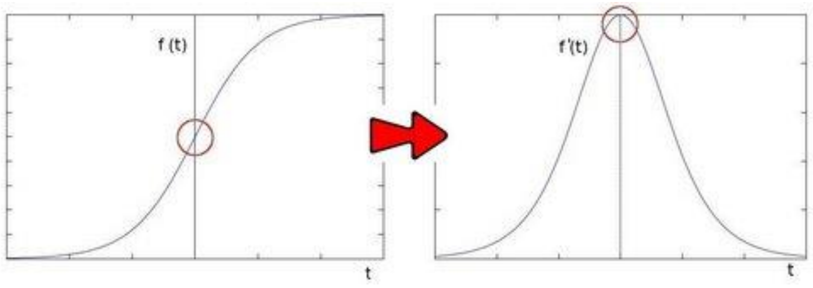
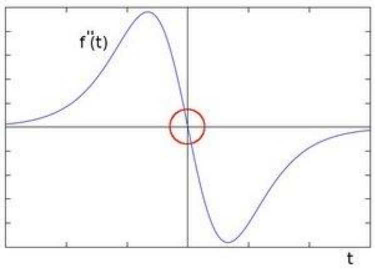


Figure 5 - Result of Sobel edge detection. Contrarily to the Canny operator, horizontal lines are barely considered. It could be worth weighing more horizontal lines in the final image.

It is worth mentioning the other edge detection techniques, that are unfortunately not implemented in the widely used open-source library OpenCV: Deriche [6], Prewitt [7], Robert [8], and so on. However, another operator that can be used in combination of the ones previously presented is the Laplacian [9]. Theoretically, in the edge area, the pixel intensity shows a “jump” or high variation of intensity. An edge is therefore characterized by a maximum in the first derivative [10].



Which means it is also characterized by a zero in the second derivative.



Therefore, identifying the zeros in the second derivative images on both axes can give edges in the original image (zeros can also appear in meaningless locations, which is why it is mandatory to apply filtering where needed).

By computing the Laplacian operator of the original image, which is defined as following [11],

$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \mathbf{(2)}$$

it is possible to find edges in the image.

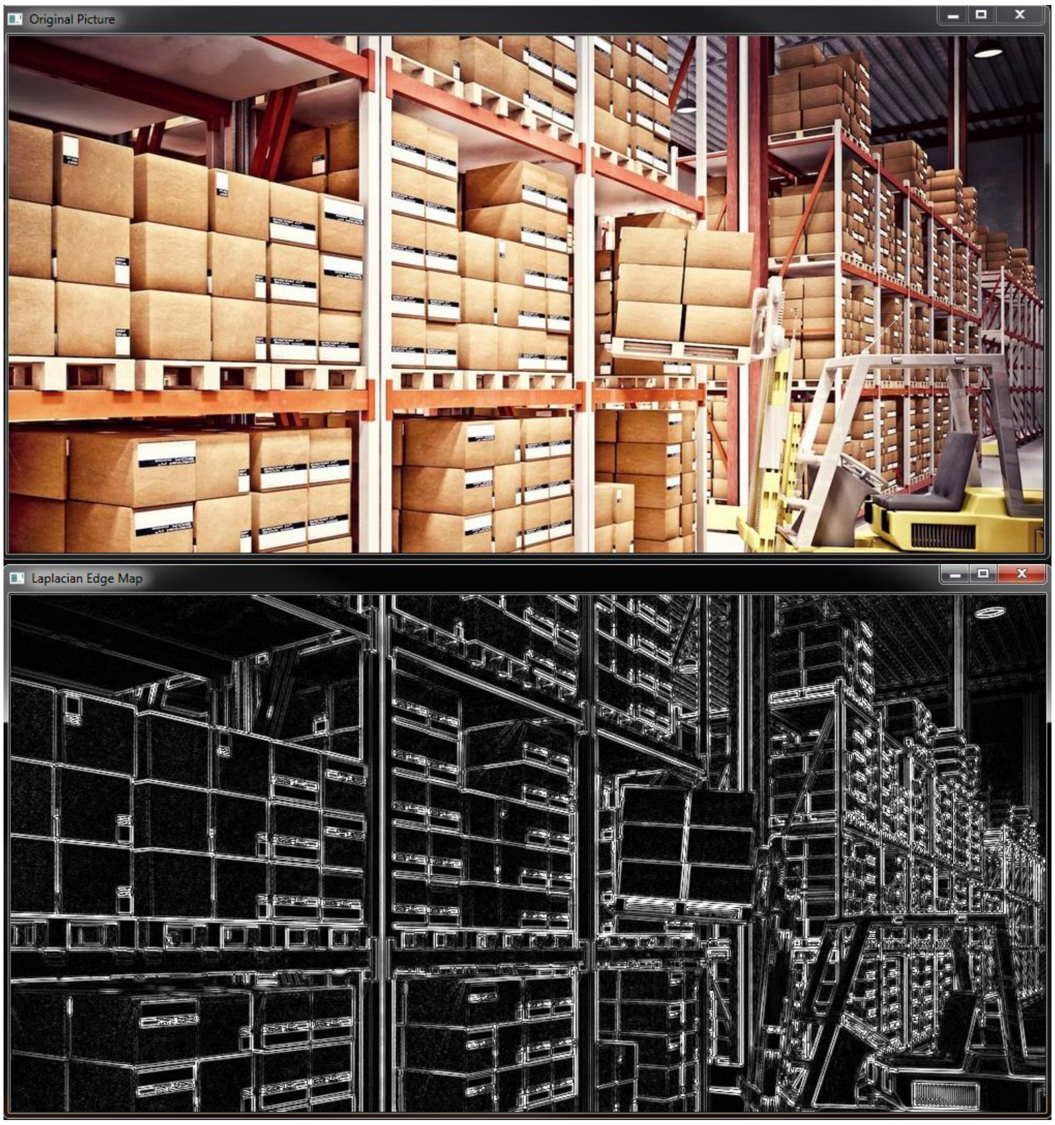


Figure 6 - Result of Laplacian edge detection. Compared to the Sobel operator, edges along axis x are more distinguishable in the image. Theoretically, this is explained by the fact that in edges the color changes only a little (so according to the first derivative, there should be no edge in the region), but it changes fast (because of the proximity between the packets, so there is an important acceleration in this image area: the second derivative along x is consequent).

1.B.2. Noise reduction

Noise reduction is an important step in every segmentation pipeline. As seen before, noise has undesired effects on subsequent steps, e.g. false positive edges appearing in the image because of the area non-uniformity. Many filters were designed to address the noise issue and their use-case depends on the image characteristics and the target output.

Smoothing algorithms are usually used to perform noise reduction, although there are many reasons for smoothing. Smoothing is the result of image filtering with such function:

$$g(i,j) = \sum_{k,l} f(i+k, j+l) \cdot h(k,l) \qquad \textbf{(3)}$$

where $h(k,l)$ is the kernel (i.e. the filter coefficients).

Blurring is the simplest smoothing algorithm and can be seen as a “normalized box”. Each output pixel is the mean of its kernel neighbors where all of them contribute with equal weights [12]. With a kernel of 3, this gives the following:

$$K = \frac{1}{3 * 3} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \textbf{(4)}$$

Although this is powerful to remove noise in images, it also smooths edges, which is not ideal in our case.

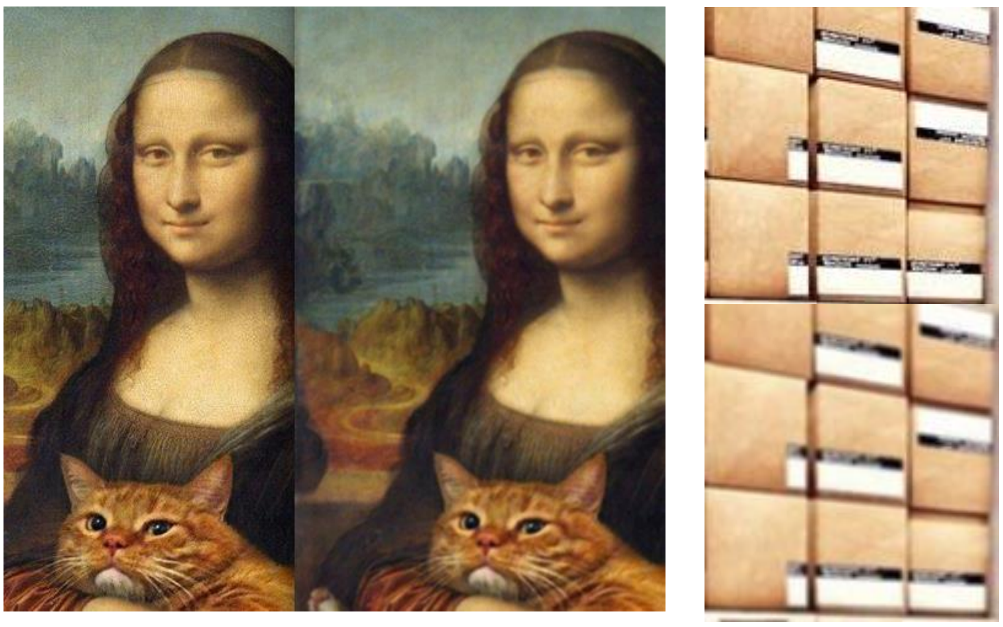


Figure 7 - Effects of the blurring filter on a random image and on a detail of the warehouse picture. Noise is reduced but edges are also less sharp.

Gaussian filtering is probably the most useful algorithm (although not the fastest) [12]. The weighed mean is computed from the neighboring pixels, the farthest influencing the least in the calculus [13].

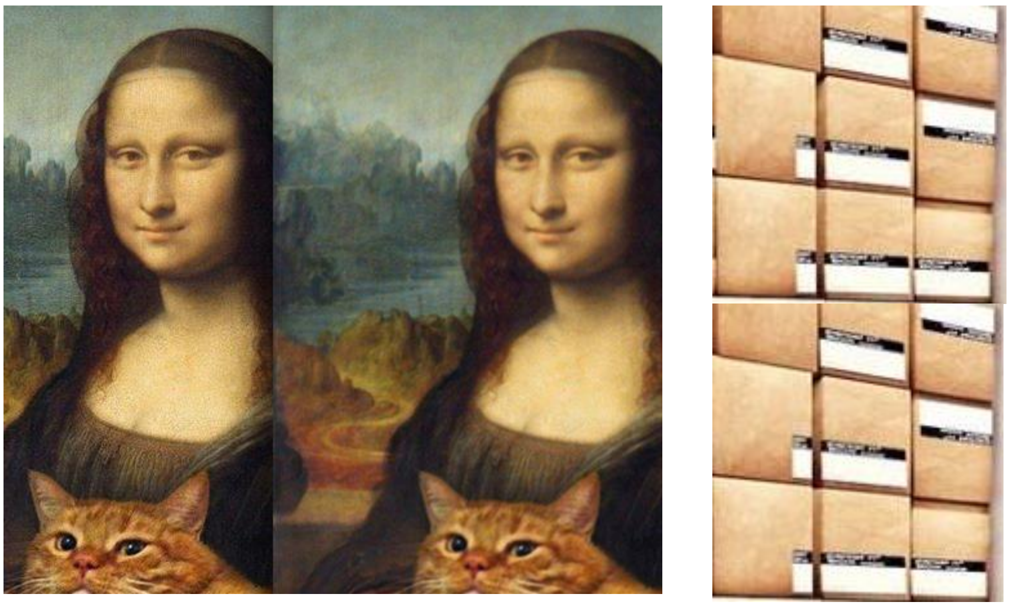


Figure 8 - Effects of the Gaussian filter on a random image and on a detail of the warehouse picture. Edges are way more precise than those affected by the blurring filter.

The median filter [14] runs through each element of the image and replace each pixel value with the median of its neighboring pixels (located in a square neighborhood around the evaluated pixel) [12].

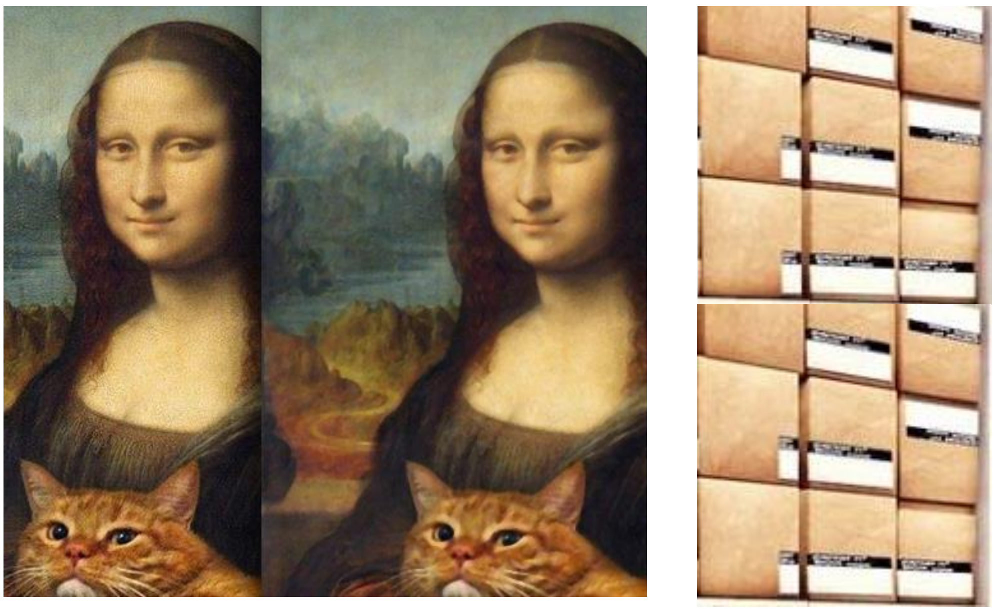


Figure 9 - Effects of the median filter on a random image and on a detail of the warehouse picture. Noise is reduced and contours are quite well preserved.

Finally, the bilateral filter is an attempt of preventing edges smoothing while reducing noise [12]. As for the Gaussian filter, bilateral filtering consider the weighed neighbors, this weight being composed of two features: the first one is the same weighting used by the Gaussian filter, the second taking into account the difference in intensity between the neighboring pixels and the evaluated one. This design relies on the observation that noise artefacts are usually within the same intensity range as the neighboring pixels, while edges pixels are surrounded with areas of very different intensities (that is how edges are defined).

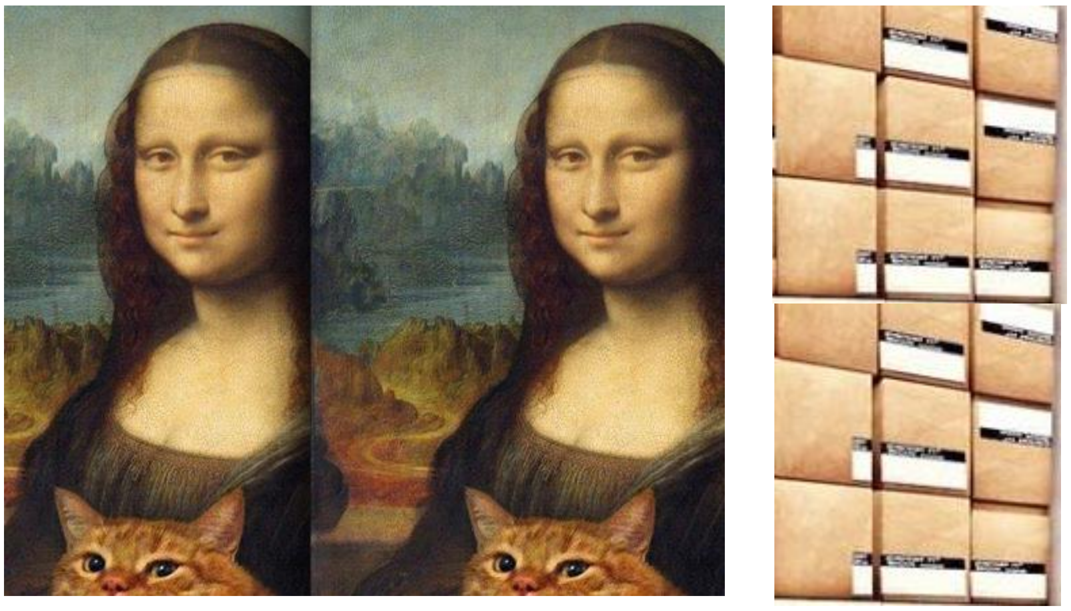


Figure 10 - Effects of the bilateral filter on a random image and on a detail of the warehouse picture. It is almost difficult to see a difference with the chosen kernel, but noise artefacts are slightly smoothed while edges are perfectly preserved.

All results above have been computed with the same kernel size. Below is their respective computational time. It is worth noticing that bilateral filtering is very heavy and should only be used when a very important quality is needed. Depending on the use case, accuracy might need to be sacrificed to computational speed.

Smoothing filter	Execution time (µs)
Blur	1873
Gaussian	1508
Median	1118
Bilateral	251032

Figure 11 - Execution time of smoothing filters available in OpenCV.

1.B.3. Morphological mathematics

Morphological mathematics are basically a set of operations that process images based on shapes. Morphological operations apply a *structuring element* to an input image and generate an output image. They are usually used to **remove noise and isolate elements** and therefore could be used for detecting objects like packets or baggage. The most common operations are performed on binary images (although some image processing libraries offer grayscale operators) and are called *dilation* and *erosion*. Dilation is used to grow image regions based on structuring elements and can be used to suppress undesired holes and gaps in images or to thicken edges.

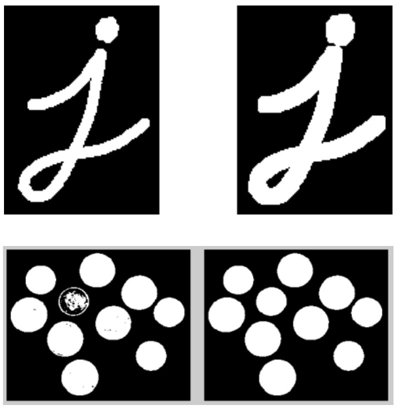


Figure 12 - Examples of dilation with a pixel-shaped structuring element. Dilation can be used to connect disjointed regions or fill gaps.

Similarly, erosion shrinks regions in the images. It computes a local minimum over the area of given structuring element. Erosion can be used to thin elements or remove small components.

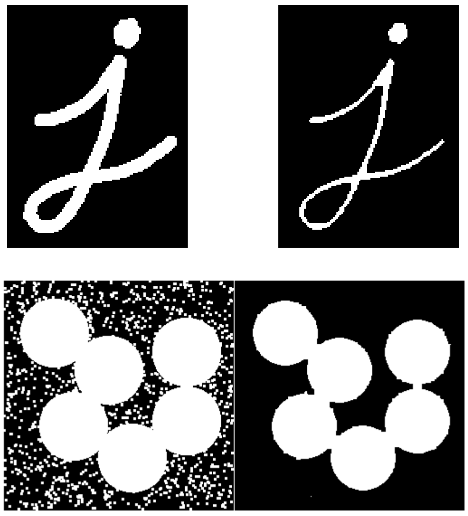


Figure 13 - Examples of erosion with a pixel-shaped structuring element. Erosion can also be used to reduce noise.

These binary operators can be combined in what is called opening (an erosion followed by a dilation) and closing (the opposite). These are useful to get rid of small artefacts while preserving contours on a binary image or either open/close holes in shapes.

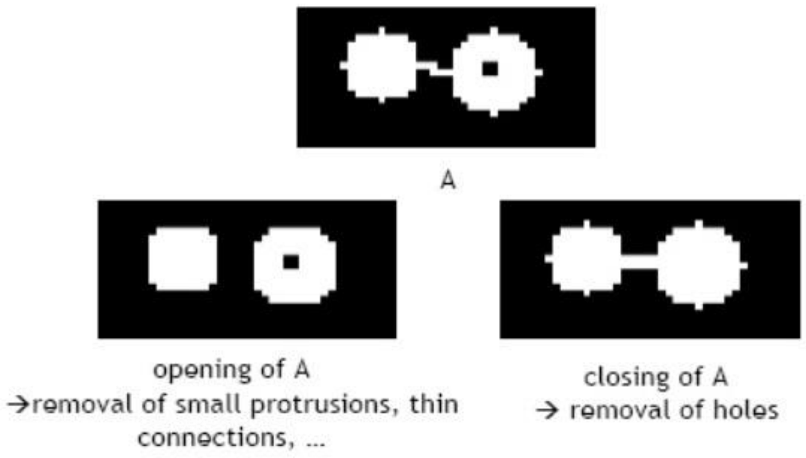


Figure 14 - Comparison of opening and closing results.

By subtracting the erosion from the dilation of an image, we obtain the morphological gradient of this latter, which can be seen as the encirclement of shapes in the image.

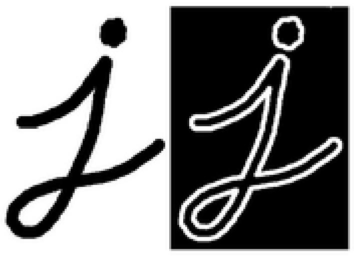


Figure 15 - Morphological gradient of a handwritten letter. This operator is useful for finding the outline of an object.

What is also interesting with these elementary operators is that they can be used to extract lines when combined together. In order to do so, one needs to define a structuring element shaped with linear components (since the structuring element is chosen in the shape and size of features that are wanted to be processed/extracted).

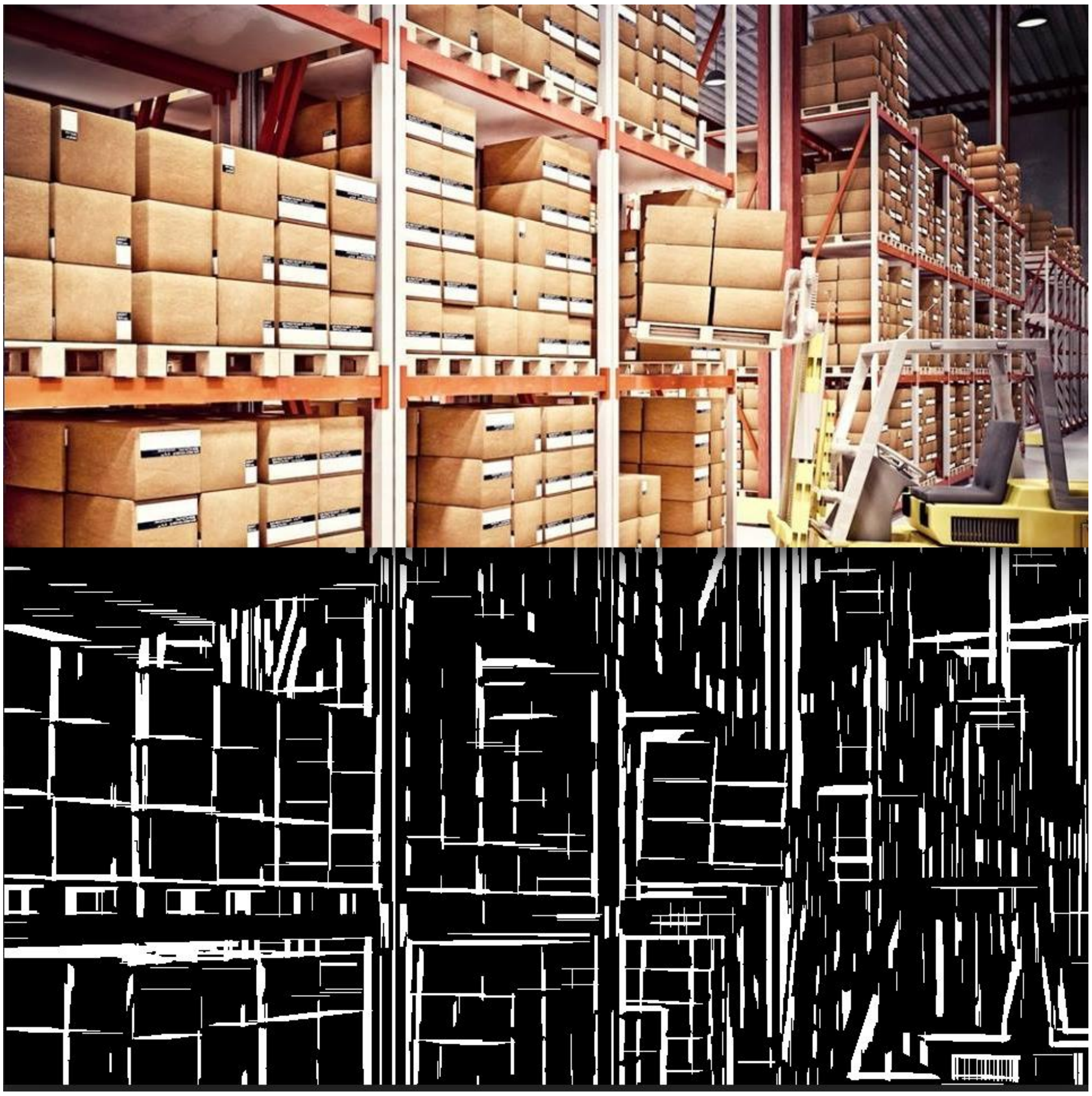


Figure 16 - Edges in the warehouse image extracted with morphological operators tuned with linear structuring elements. This could be a start to detect packets location.

1.B.4. Template matching

Template matching is a technique for finding areas of an image that match a template image. In order to do so, two features are required: an image to search for matchings within, and a patch image which will be compared to the source image. The patch is moved one pixel at a time, from left to right and up to down (this is called *sliding*). At each location, metrics are calculated to determine how good the patch is similar to the area of the image considered. The result is the location with the highest probability of matching.

Template matching could be a way to detect packets if combined with other methods: patch could be a set of pixels whose color would be similar to the packet's ones, and whenever there's a match, it means that: A - there is a slot there and B - it is occupied.

It is worth noticing that in all examples studied, the patch is very complex in shape and contents, therefore it is not very generic. Chances are that the object will not be recognized in many cases if the dataset is composed of varying images (resulting in lots of false negatives).

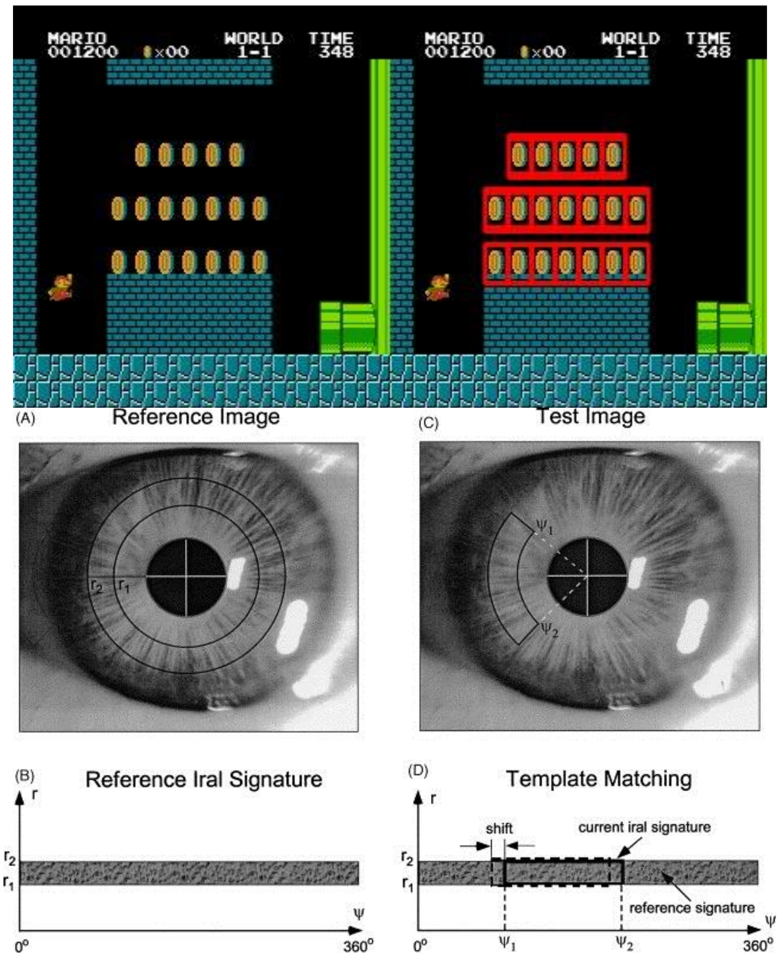


Figure 17 - Example of template matching in a famous video game frame and for uniquely identifying people with their iral print. In the first example the target object is a coin, and sliding across the whole picture leads to the detection of all occurrences of the coin. In the second example we see that it is possible to slide the patch but also to bend it over the image dimensions.

1.B.5. Hough transformation

Hough transformation is briefly introduced because it is a classical and widely-used method for detecting features in images, mostly geometrical patterns. Hough transformation could be used to extract lines in images and therefore detect parking slots or packets shapes.

To explain how this algorithm works it is important to recall that coordinates can be expressed in different systems (e.g. in the Cartesian with the abscissa and ordinate coordinates, or in the Polar with the distance from the frame origin and the angle from the abscissae axis and the point). For computing Hough transform, coordinates are expressed in the Polar coordinate system. Each point can therefore be expressed as (r, θ) where $r = xcos\theta + ysin\theta$. For each point (x_0, y_0), a family of lines going through that point is defined as

$$r_\theta = x_0 cos\theta + y_0 sin\theta \qquad \textbf{(5)}$$

When plotting the family of lines that goes through a point (x_0, y_0), the result is a sinusoid. If the curves of two different points intersect in the plan $\theta - r$, it means they belong to the same line in the input image. This means lines in images can be detected by finding the number of intersections between curves.

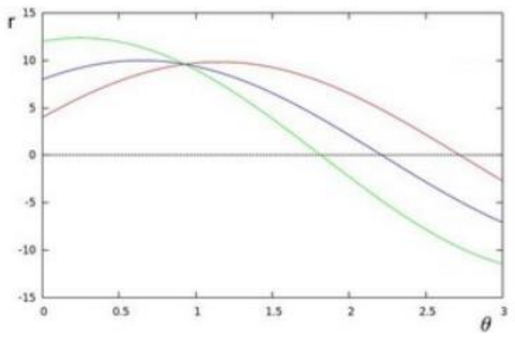


Figure 18 – Families of lines from different points. The intersection indicates that the points corresponding to these three curves all belong to the same line in the original image.

The more curves intersecting means that the line represented by that intersection have more points. In general, we can define a threshold of the minimum number of intersections needed to detect a line. If the number of intersections is above some threshold, then the Hough transform declares it as line with the parameters (θ, r_θ) of the intersection point.

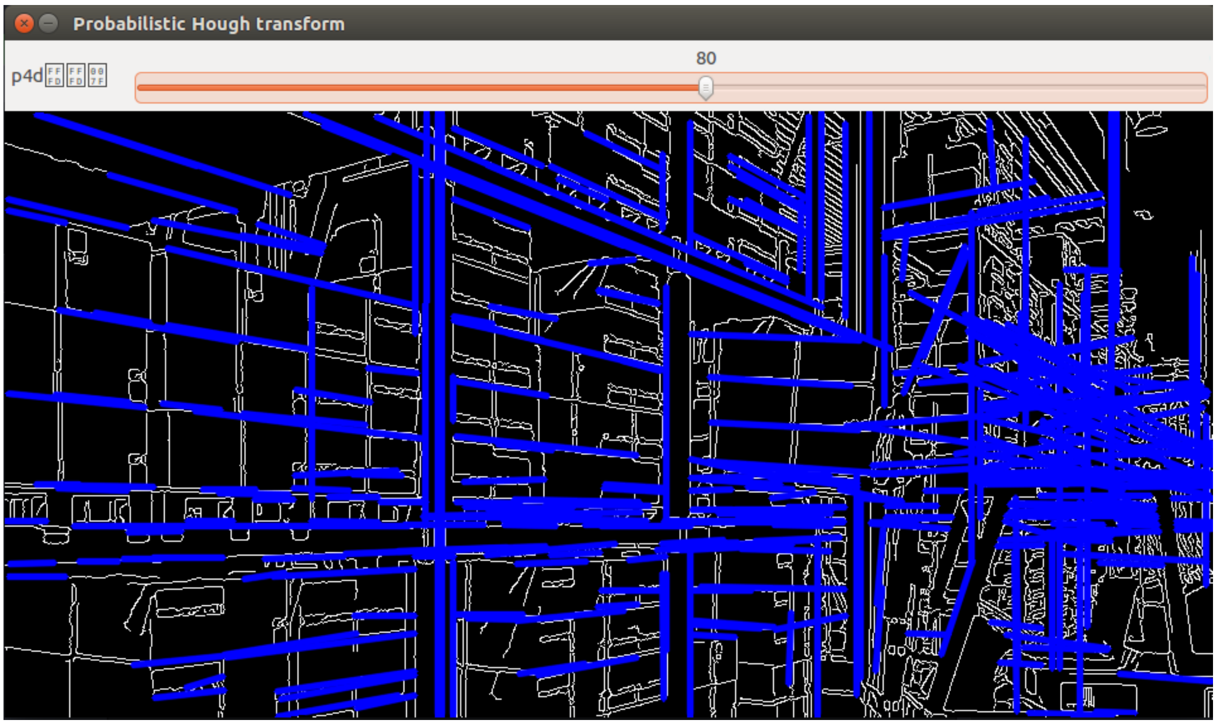


Figure 19 - Detection of lines with Hough transform. There are a lot of undesired detected lines at the right of the image and misses along the vertical axis, both caused by a non-optimal management of noise in the image.

1.B.6. Histogram Equalization

The last operation covered in this part is histogram equalization. Histograms are graphical representations of the intensity distribution of an image. They quantify the number of pixels for each intensity value considered. By equalizing an histogram we ensure frequencies are uniformly present in it, which result in a better contrast.

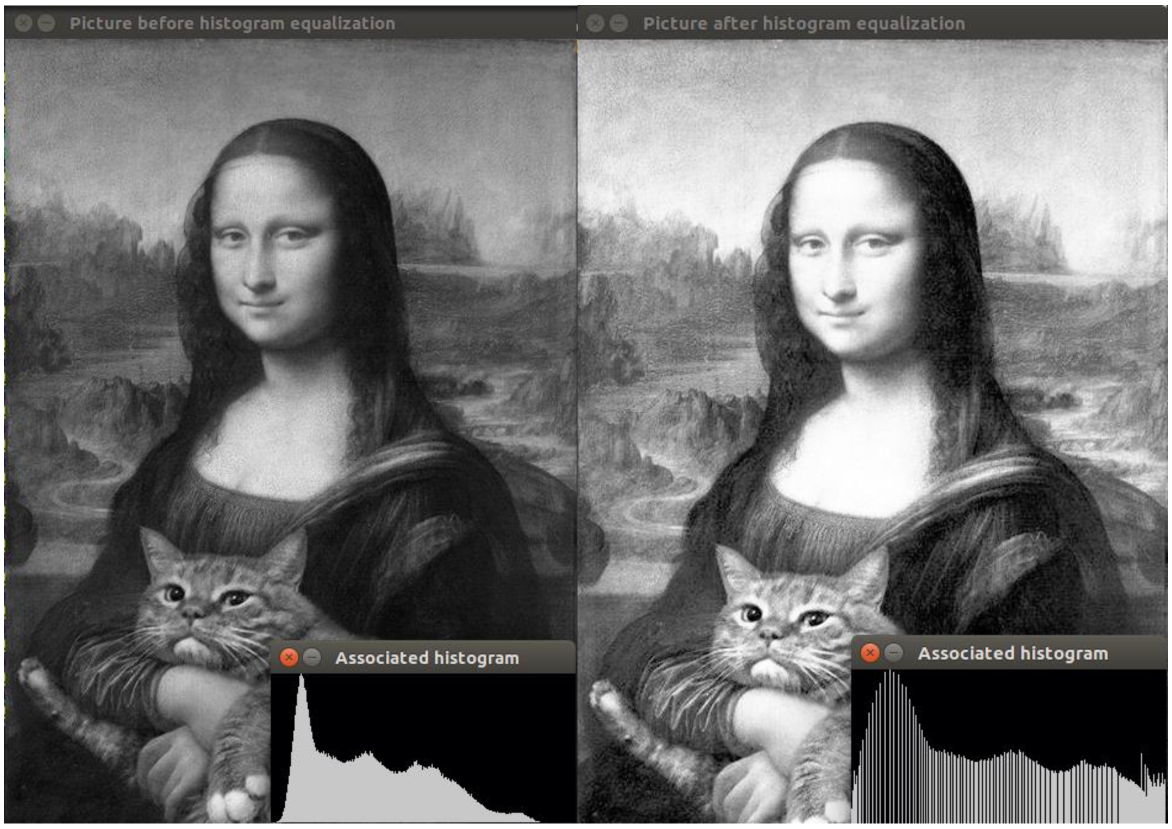

Picture before histogram equalization
Picture after histogram equalization
Associated histogram
Associated histogram

Figure 20 - Histogram equalization of a random image. Histogram of the original image lacks of very low and very high frequencies. Equalizing the histogram widens the range of frequencies, which now extends from the lowest to the highest possible frequency. The modified image seems lighter because the original one had few high frequencies components. Fabric folds also seem sharper.

Histogram equalization is a powerful tool that can be used to ensure that all images in a dataset have similar contrasts (which is interesting in some use-cases where exposure light changes within the day).

1.C. Multi-object detection algorithms

There are very few papers evoking algorithms of multi-object detection, for now, researchers and industrials focus on machine learning-based solutions. Before the emergence of deep learning, limitations and constraints raised by the hardware would not enable the conception of powerful algorithms for detecting multiple objects or people in a frame. Two famous techniques are yet worth mentioning: the Viola-Jones algorithm and the saliency-based detection technique.

1.C.1. Viola-Jones

Viola-Jones detection technique emerged in 2001 and became reference afterwards. It is the first object-detection framework that was developed to provide real-time detection of objects in images and was especially designed for detecting faces.

Viola-Jones detection is based on four main stages, namely:

- Haar features selection
- Creation of an integral image
- Adaboost training
- Cascading classifiers

Haar features selection

The Viola-Jones method is based on the assumption that human faces share common properties. The regularities may be matched by Haar features.

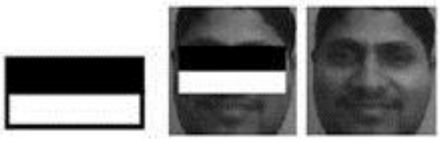
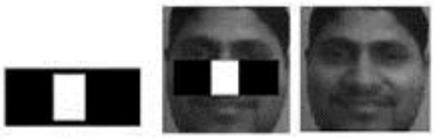
The eye region is darker than the upper-cheeks	
The nose bridge region is brighter than the eyes	

Figure 21 – Examples of common properties assumed on human faces and associated Haar templates.

Organs can also be found based on their assumed size and location, their pixel intensities and oriented gradients, for instance.

Creation of integral images

Integral images are of same size of their originals and each of their pixels is the sum of the neighboring pixels above and on the left. In other words, each pixel (x, y) of the integral image ii is defined as:

$$ii(x,y) = \sum_{x' \leq x,\ y' \leq y} i(x', y') \qquad \textbf{(6)}$$

This representation is interesting because it guarantees constant computation time regardless of the characteristic size (*i.e.* the Haar templates).

Adaboost training

Viola-Jones framework uses boosting to both select the best characteristics and to train classifiers that use them. The boosting algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers,

$$h(x) = sgn\left(\sum_{j=1}^{M} \alpha_j h_j(x)\right) \qquad \textbf{(7)}$$

where each weak classifier is a threshold function based on the associated characteristic c_j.

$$h_j(x) = \begin{cases} -s_j & if\ c_j < \theta_j \\ s_j & otherwise \end{cases} \quad \textbf{(8)}$$

The threshold value θ_j and the polarity $s_j \in \pm 1$ are determined in the training, as well as the coefficients α_j.

Cascading classifiers

Viola-Jones algorithm is based on an exhaustive search in the image, which means it is needed to look for each characteristic at all positions, in all possible scales. This is extremely expensive, which is why cascade classifiers are used to determine whether a sub-window is *definitely not* or *may be* a human face. If the test fails in any of the stages, the sub-window is immediately discarded.

Viola-Jones algorithm is an example of supervised learning and needs thousands of samples to train the classifiers, which may restrict its usage. However, it is implemented in OpenCV under a BSD license, which means there is no commercial restriction on its usage.

1.C.2. Salient-based detection

Another object-detection technique that is not deep learning-based relies on the saliency of objects. By computing contrast and histograms of gradients, it is possible to detect objects in the foreground of images. However, this method has strong drawbacks, for it cannot detect more than two objects in a single image [15] and that it often results in numerous false positives or massive misses [15].

1.C.3. Open source libraries

There are plenty of open source libraries available for image processing for which C++ programming is possible. Below are some of them:

- OpenCV (this one has been used to compute all examples in this benchmark),
- Magick++,
- VIPS,
- SImd,
- BoostGIL,
- CImg …

2. Deep learning approach

2.A. Introduction to deep learning

2.A.1. History of artificial intelligence

Research on artificial intelligence has been ongoing for more than four decades and continues to obtain promising results. Although there is nothing new in the will of creating machines fitted with autonomous learning abilities, application became possible only recently with the continuous increasing of computational speed and improvement of hardware devices.

Artificial intelligence is already part of our daily lives, for it is the reason why a single picture and an application are enough to translate from and into any language. It enables autonomous cars to detect and interpret road signs in real time. There are countless stories of world champions beaten by bots at both board and video games.

Machine learning falls within artificial intelligence, and deep learning is a subfield of machine learning. Figure 22 below presents the fields organization.

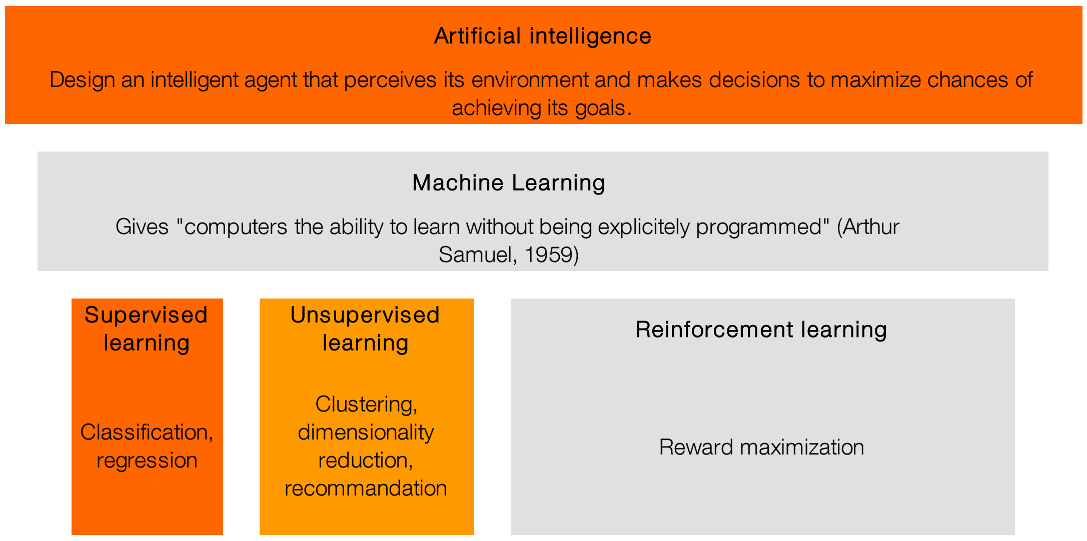
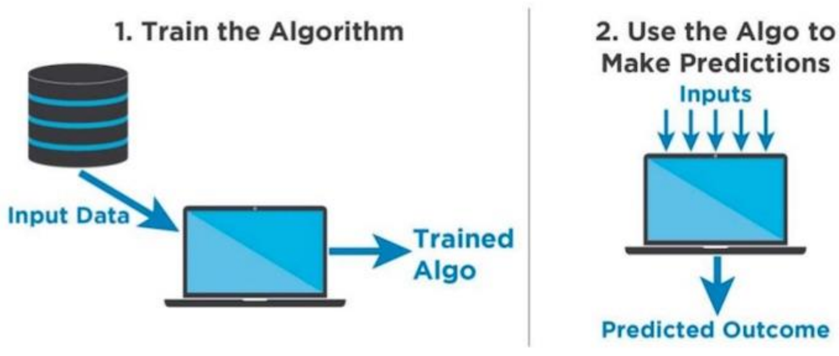

Artificial intelligence
Design an intelligent agent that perceives its environment and makes decisions to maximize chances of achieving its goals.
Machine Learning
Gives "computers the ability to learn without being explicitely programmed" (Arthur Samuel, 1959)
Supervised learning
Classification, regression
Unsupervised learning
Clustering, dimensionality reduction, recommandation
Reinforcement learning
Reward maximization

Figure 22 - Machine learning is one of many subfields of artificial intelligence, concerning the ways that computers learn from experience to improve their ability to think, plan, decide, and act.

2.A.2. Principle of machine learning

Machine learning aims at enabling machines to (1) identify features in data, (2) build models that (3) represent the environment and make predictions without relying on existing rules or models. There are two phases in developing an object detection solution with deep learning. Firstly the solution is fed with labeled data, so that it determines the causal relationship between the input and its result (Figure 23 below describes this process).



$$Y = f(X) + \epsilon$$

```
X (input) = years of higher education
Y (output) = annual income
f = function describing the relationship between X and Y
ε (epsilon) = random error term (positive or negative) with mean zero
```

Figure 23 – Schematization of machine learning basic process and an example of obtained model for guessing the annual income when given the number of years of studies.

Once the relationship is found, the solution is given new, unlabeled data, and computes the result of the causal relationship function.

It is worth mentioning that the training data set shall be wide and extensive enough to prevent the causal relationship to be too specific and therefore not representing what generally happens in the environment. A model that is too specific is called *overfitted* because its variance is high, while a model with high bias is considered *underfitted* (see Figure 24 below).

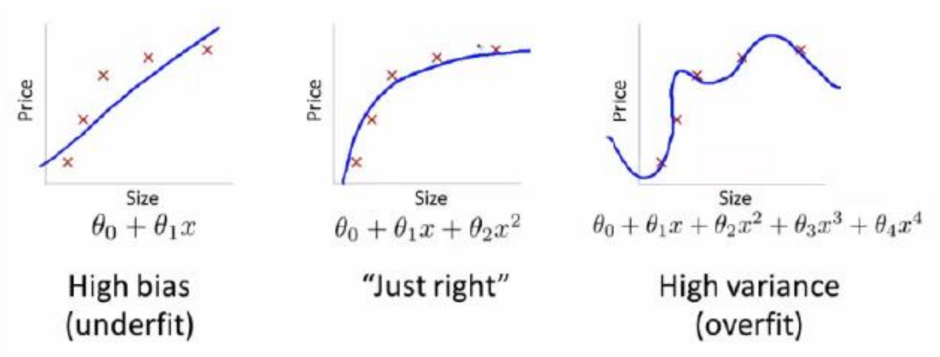


Figure 24 – Example of possible models. It is wanted to be the closest to those in the middle, that is a good compromise between bias and variance.

2.A.3. Why deep learning?

Despite its promising results with simple data, **machine learning cannot be used as it for identifying objects in images** : pixels that compose the images are represented as heavy vectors carrying tremendous amount of data. Since machine learning is based on very simple concepts like linear regression, there is no way to tell whether it could correctly detect and recognize those objects.

In animal vision, object recognition is a very complex process based on **neural networks** with neurons tailored for recognizing different types and sizes of features in images. Instead of a process where all neurons would treat pixels data the same way, it exists several types of neurons and several **layers of abstraction**. Some neurons aim at interpreting small features in the object while others are tailored for recognizing bigger shapes, and the whole lot of data is used to identify the object.

On a biologic aspect, the basic process looks like the following:

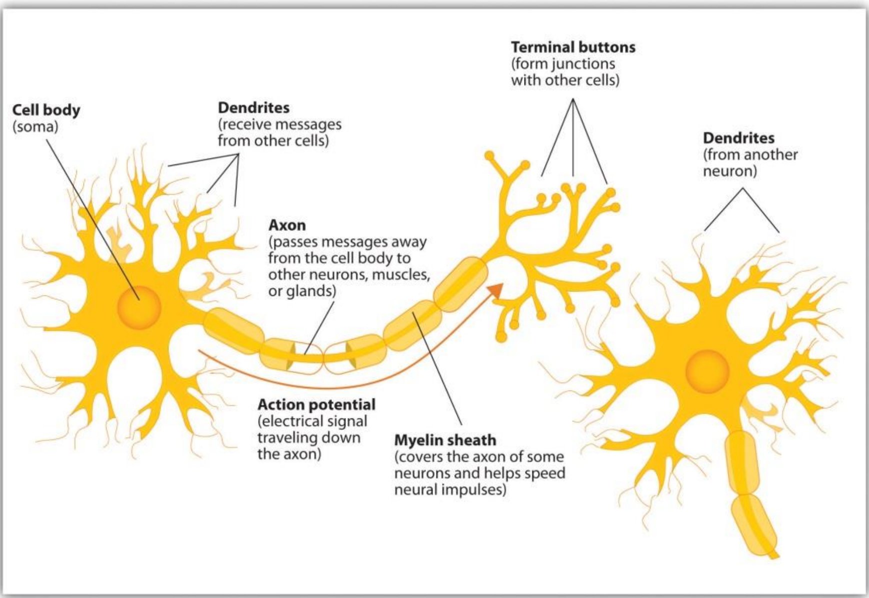

Terminal buttons
(form junctions with other cells)
Cell body
(soma)
Dendrites
(receive messages from other cells)
Dendrites
(from another neuron)
Axon
(passes messages away from the cell body to other neurons, muscles, or glands)
Action potential
(electrical signal traveling down the axon)
Myelin sheath
(covers the axon of some neurons and helps speed neural impulses)

Figure 25 – Description of a biological neuron and its behavior towards other body cells.

Neurons get information (in the form of electrical signals) from other neurons or body cells through their *dendrites*. The data transmitted through the *axon* by each cell influences differently the receiving neuron regarding its *action potential*. Once the action potential of all cells has been considered, the data is transmitted to the next cell through the synapses (or *terminal buttons*).

Similarly, deep learning is based on networks of artificial neurons (see Figure 26 below).

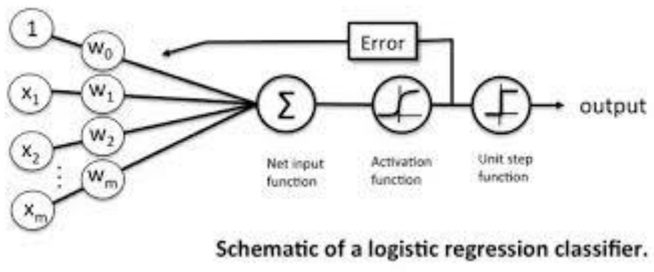


Figure 26 – Artificial transposition of neurons as in deep learning neural networks.

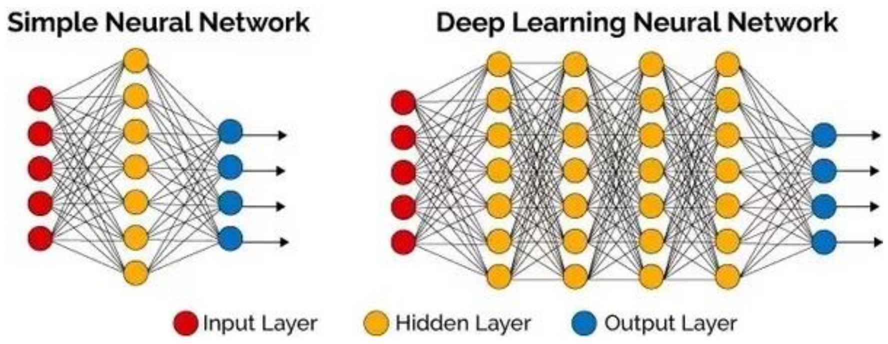


Figure 27 – Comparison of architectures of classic machine learning ("shallow" architecture) and deep learning networks. In this latter the several layers are dedicated to specific types and sizes of features – One layer focuses on small patterns, another looks for global shapes, aso.

In order to do so, networks with numerous hidden layers are constructed (see Figure 27 above). The data carried by all the input cells is considered, then weighed and summed. This result goes into an activation function which is then thresholded and gives the output to feed other neurons. This is the well-organized connection of neurons into networks that allow animal brains (and computers) to recognize objects. At the end of the network (the sink), the result is given in the form of several classes of objects and the associated probabilities. The recognized object belongs to the class with the highest probability: on Figure 28 on next page, the networks inputs are *wolf* with probability 0.1 and *dog* with probability 0.9, so the network recognized a dog.

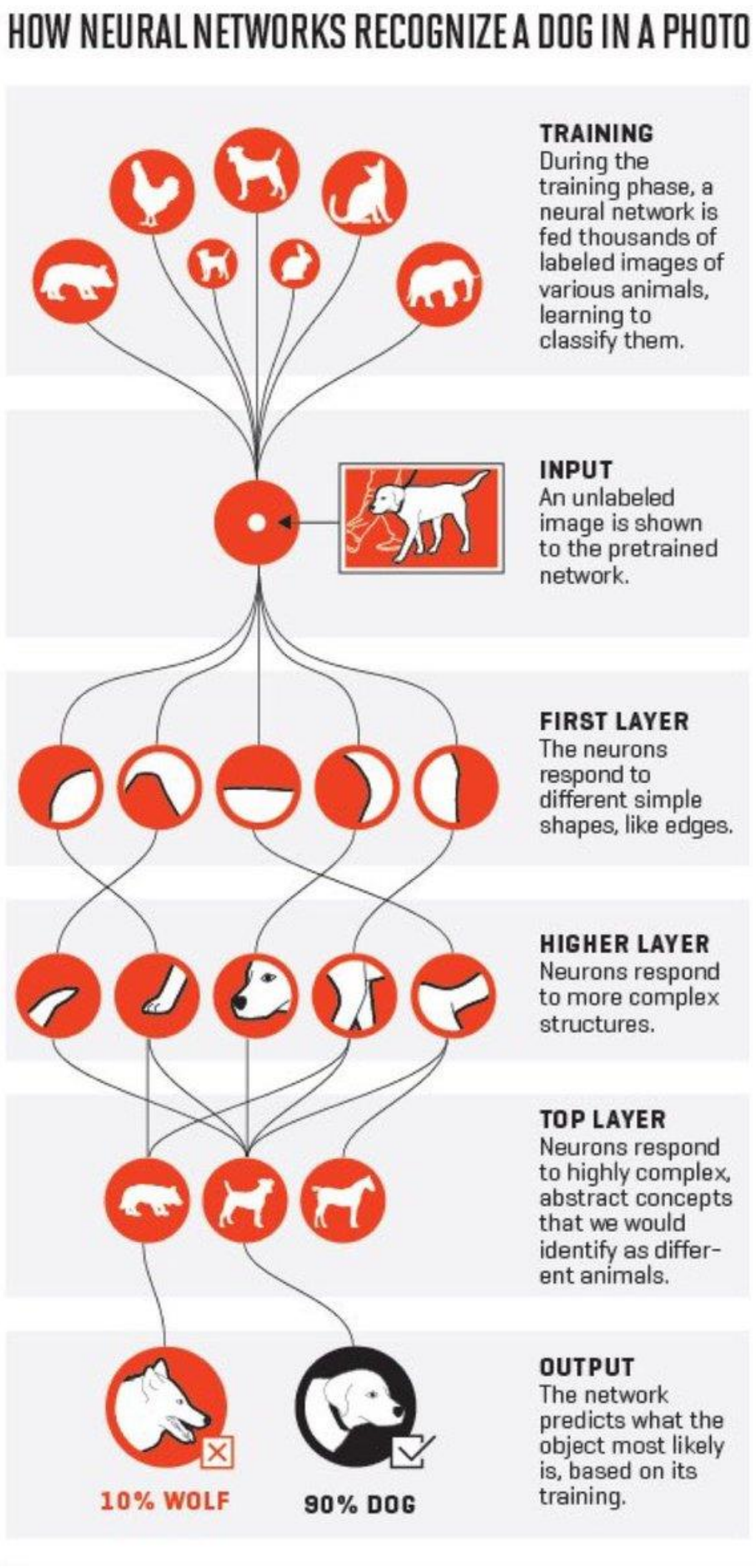

HOW NEURAL NETWORKS RECOGNIZE A DOG IN A PHOTO
TRAINING
During the training phase, a neural network is fed thousands of labeled images of various animals, learning to classify them.
INPUT
An unlabeled image is shown to the pretrained network.
FIRST LAYER
The neurons respond to different simple shapes, like edges.
HIGHER LAYER
Neurons respond to more complex structures.
TOP LAYER
Neurons respond to highly complex, abstract concepts that we would identify as different animals.
OUTPUT
The network predicts what the object most likely is, based on its training.
10% WOLF
90% DOG

Figure 28 – General description of neural networks behavior.

To sum up, machine learning is a great tool for building a model explaining the surrounding environment, but is too simple to accurately recognize objects in images. On the other hand, deep learning brings additional complexity to the neural networks used in machine learning but also drastically increases the chances of correctly identify features and their location in the images. Once the network has been trained to identify features, it uses way less resources for identifying unlabeled data.

Next section presents the different famous deep learning algorithms used in the literature, their strengths and breaches.

2.B. Overview of studied models

The number of architectures and algorithms that are used in deep learning is wide and varied. However, this range shrinks when image recognition issues are addressed. Among them, two architectures are particularly famous for this purpose, namely the Convolutional Neural Networks (CNNs) and Deep Belief Networks (DBNs). Additionally, recent papers present hybrid methods that could improve computational time and output accuracy, but their principle remains the same.

2.B.1. Convolutional Neural Networks

A CNN is a multilayer neural network that was biologically inspired by the animal visual cortex. As a deep network, early layers recognize features (such as edges), and later layers recombine these features into higher-level attributes of the input.

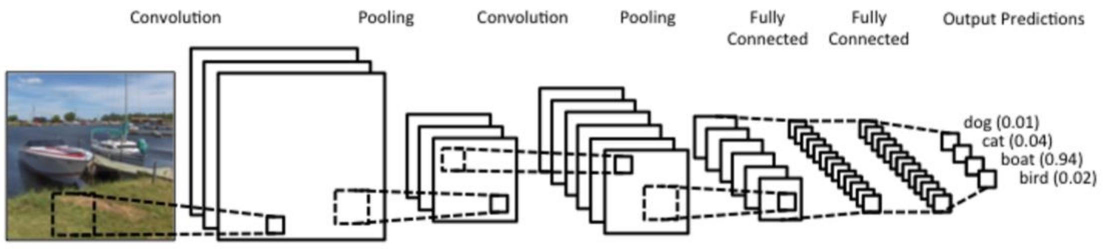


Figure 29 – Principle scheme of convolutional neural networks.

The CNN architecture is made up of several layers that implement feature extraction, and then classification. The input image is divided into receptive fields that feed into a convolutional layer, which then extract features from the input image.

The next step is pooling, which reduces the dimensionality of the extracted features (through down-sampling) while retaining the most important information (typically through max pooling). Another convolution and pooling step is then performed that feeds into a fully connected multilayer perceptron. The final output layer of this network is a set of nodes that identify features of the images (in Figure 29 above, these neurons displayed the possible outcomes along with their probability. Since the "boat" category has highest probability, the conclusion is that the picture contains a boat).

2.B.2. Deep Belief Networks

DBN is a typical network architecture but includes a novel training algorithm. The DBN is a multilayer network (typically deep, including many hidden layers) in which each pair of connected layers is a Restricted Boltzmann Machine (RBM). In this way, a DBN is represented as a stack of RBMs.

In the DBN, the input layer represents the raw sensory inputs, and each hidden layer learns abstract representations of this input. The output layer, which is treated somewhat differently than the other layers, implements the network classification. Training occurs in two steps: unsupervised pretraining and supervised fine-tuning.

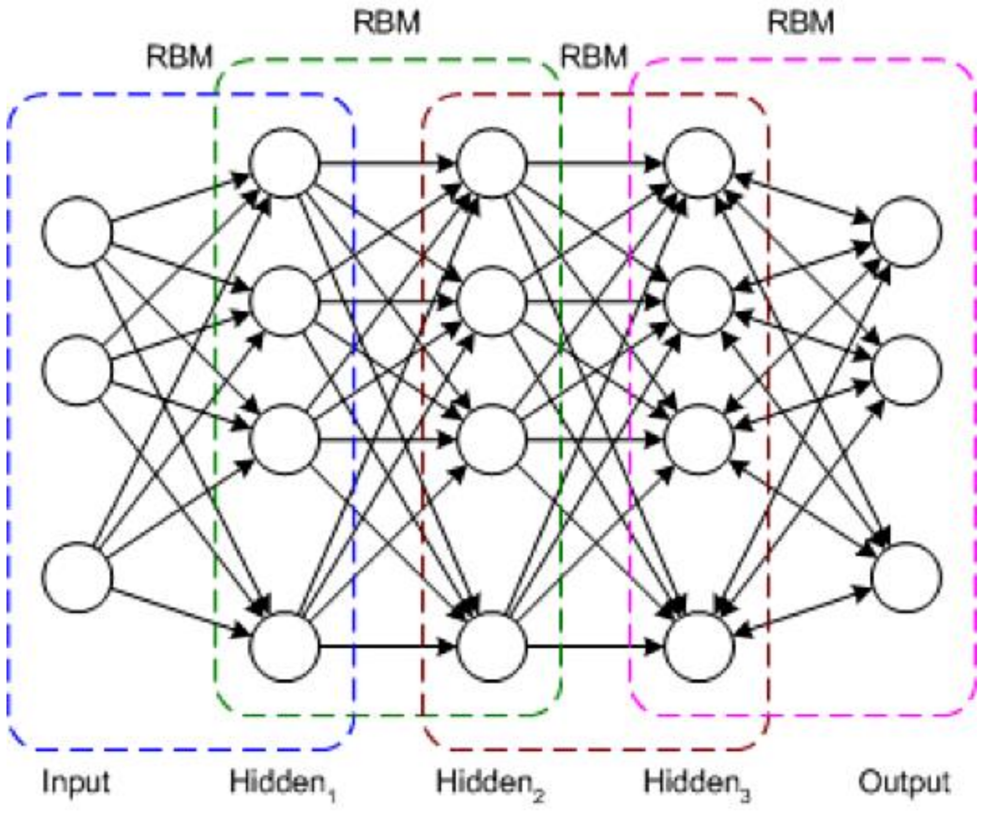


Figure 30 – Principle scheme of deep belief networks.

In unsupervised pretraining, each RBM is trained to reconstruct its input (for example, the first RBM reconstructs the input layer to the first hidden layer). The next RBM is trained similarly, but the first hidden layer is treated as the input (or visible) layer, and the RBM is trained by using the outputs of the first hidden layer as the inputs. This process continues until each layer is pretrained. When the pretraining is complete, fine-tuning begins. In this phase, the output nodes are applied labels to give them meaning (what they represent in the context of the network). Full network training is then applied by using either gradient descent learning or back-propagation to complete the training process.

2.B.3. Open source frameworks

Implementing these deep learning architectures is certainly possible, but starting from scratch can be time-consuming, and they also need time to optimize and mature. Luckily, there are several open source frameworks available to more easily implement and deploy deep learning algorithms. Among all them, below are the most famous ones with which it is possible to deploy in C++, Python or Java.

- Caffe (as well as Caffe2) is released under the BSD license. Caffe has been written in C++ and has been used for image classification. It supports GPU-based acceleration and uses Open Multi-Processing (OpenMP) for parallelizing deep learning algorithms over a cluster of systems.
- Deeplearning4j is released under the Apache license and has been written in Java, even though it has a Python interface. It includes support for RBMs, DBMs and CNNs and benefits of GPU optimization as well.
- TensorFlow, developed by Google and supporting all neural networks interesting in image processing. It is available under the Apache 2.0 license and allows applications deployment in C++, Java and Python. TensorFlow has specialized hardware interfaces.

It is worth noticing that there are a lot more of them [16] but these are the most famous, so they will be the first with which implementation will be tested.

2.C. Inherent constraints

To perform deep learning operations – whether it is training or output determination, powerful hardware is required. Depending on the number of input devices to take into account, the framerate to observe, the image resolution and other constraints, the hardware requirements quickly become important. Solution design will become with processing on a classic GPU but may require usage of cloud resources in the future – it is difficult to be sure without focusing on a use-case.

3. Processing pipeline

Whether it is chosen to design a solution based on pure image processing or relying on deep learning algorithms, some auxiliary processing is necessary. To obtain cohesive and accurate results, it is needed to feed the solution with images of similar characteristics, explaining the fact that some images need prior (and sometimes posterior) processing.

3.A. Pre-processing

Whatever the base of the solution is, it is best that input images have similar dimensions and features. Pre-processing is used for equalizing those features. Below are some of the characteristics that need to be equalized:

- Resolution. When high-resolution is not necessary to identify objects, it is better to downsample the images for a better computation speed.
- Light exposure and contrast. It is difficult to identify features in objects of low-contrast. If we refer back to Figure 2, we notice it is difficult to identify the dissociation between the trees and the ground, which makes segmentation of either of these objects difficult. Contrast can be modified by equalizing the image histogram (as explained in Part 1.B.).
- Noise. High frequencies or motion in images imply undesired noise that may create false edges or lead to false positives. Most of the time, smoothing is enough to remove most of the noise.
- And other features …

To create the pre-processing pipeline that would best equalize the input images, it is necessary to reason on the dataset of images on which the algorithm is trained/designed, otherwise the parameters set for the pre-processing steps will not be optimal. Fine tuning of the parameters is one of the main reasons that directed researchers toward deep learning [17].

3.B. Post-processing

Post-processing is consistently used for enhancing the output segmentation of detected objects. Some of the defaults triggered by common segmentation pipelines are the following:

- Holes in the segmentation shape. It happens when contrast is low or there is remaining noise in the image. To fill them, morphological mathematics can be applied.

- Appearance of tiny spikes along the contour of the segmentation shape. This could be due to a non-optimal tuning of parameters for segmentations techniques. To even this contour we could either use a slight smoothing or try to perform an opening.

- And others …

And of course, reverse operations shall be applied if downsampling, changing of color sets or some others have been performed in the pre-processing pipeline.

3.C. Metrics computation

In order to evaluate the performance of the solution, standard metrics can be computed. There are several ways to measure the agreement between model predictions and human annotations [18]. It might be interesting to use the same evaluation measures as commonly used in research enterprises, thence these standard metrics are presented in the following.

3.C.1. Precision-Recall (PR)

For an output image O, we can convert it to a binary mask M and compute *Precision* and *Recall* by comparing M with ground-truth G[1]:

$$Precision = \frac{|M \cap G|}{|M|} \quad , \qquad Recall = \frac{|M \cap G|}{|G|} \qquad \textbf{(9)}$$

Vulgarly speaking, *Precision* is the fraction of relevant instances among the retrieved instances, while *Recall* is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

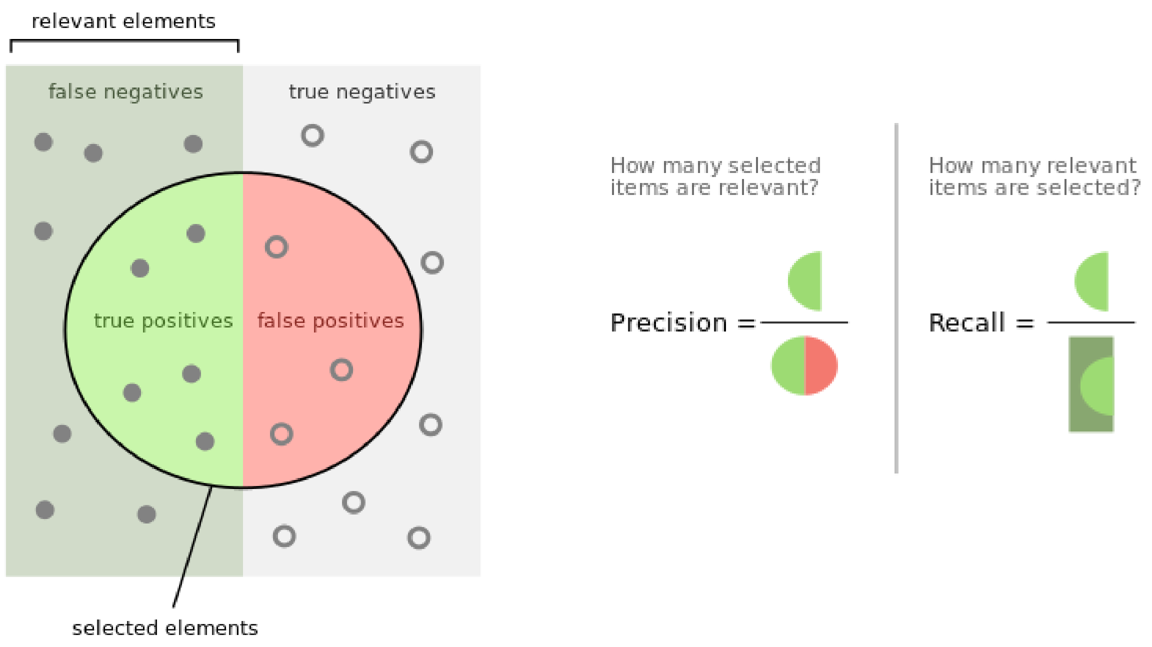


Figure 31 – Graphic explanation of Precision and Recall operators.

3.C.2. F-Measure

Usually, neither Precision nor Recall can comprehensively evaluate the quality of an output image. To this end, the F-measure is proposed as a weighted harmonic mean of them with a non-negative weight β:

$$F_\beta = \frac{(1+\beta^2) \cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} \qquad \textbf{(10)}$$

F_β reaches its best value at 1 and its poorest at 0. When β equals 1, this score is called Sørensen–Dice coefficient or Dice similarity coefficient (DSC).

[1] Ground-truth is how we call the manual segmentation processed by a human operator.

3.C.3. Receiver Operating Characteristics (ROC)

In addition to the PR and F_β operators, we can also support True and False Positives Rates (respectively TPR and FPR):

$$TPR = \frac{|M \cap G|}{|G|}, \qquad FPR = \frac{|M \cap \bar{G}|}{|\bar{G}|} \qquad \textbf{(11)}$$

where $\overline{M}$ and $\bar{G}$ denote the complement of the binary mask M and ground-truth G, respectively. The ROC curve is the plot of TPR versus FPR by varying the binarization threshold.

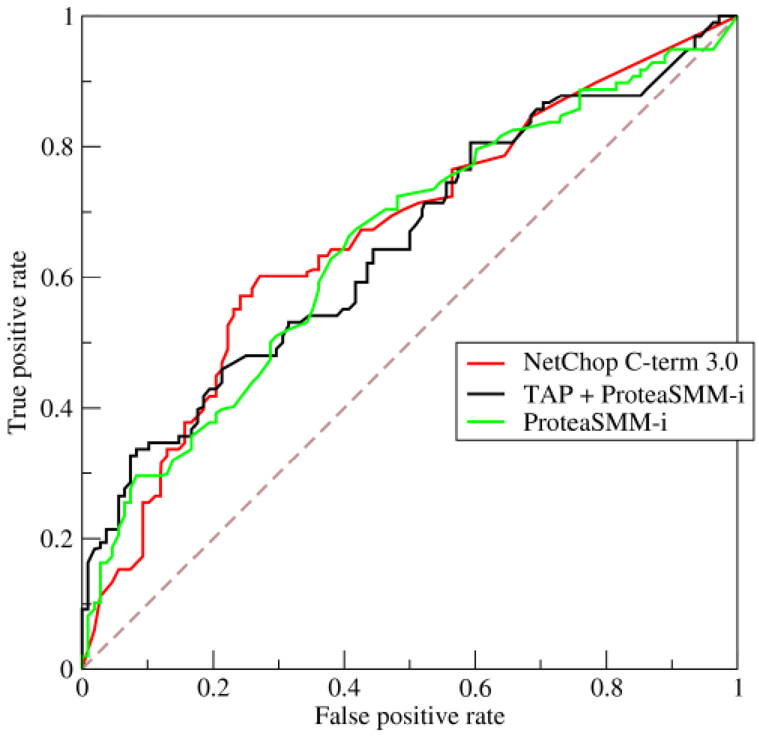


Figure 32 – Example of what do ROC curves look like.

3.C.4. Area Under ROC Curve (AUC)

While ROC is a two-dimensional representation of a model's performance, the AUC distills this information into a single scalar. As the name implies, it is calculated as the area under the ROC curve.

A perfect solution will score an AUC of 1, while poor results will lead to an AUC around 0,5.

3.C.5. Mean Absolute Error (MAE)

The overlap-based evaluation measures introduced above do not consider the true negative assignments, i.e. the areas correctly marked as not belonging to an object. This favors methods that successfully assign object categories but fail to detect regions where there is nothing to identify over methods that successfully "non-detect" regions of very little interest but make mistakes in correctly identify objects. MAE measures the average magnitude of the errors in a set of predictions, without considering their direction.

$$MAE = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j| \qquad (12)$$

It is worth noticing that oftentimes, these metrics do not agree with each other. The number of negative examples (pixels that aren't belonging to an object) is typically much bigger than the number of positives examples (pixels belonging to an object) in evaluating such models. Therefore, PR curves are more informative than ROC curves and can present an over optimistic view of an algorithm's performance [15].

Discussion

It has been decided to design such solution upon the use case of people and luggage detection in buses and trains. In this particular case, deep learning and purely image processing-based techniques are worth considering. However, associated constraints will therefore be different. It is uncertain how much hardware resources are needed to launch deep learning-based methods, while it is easier to estimate the needs for image processing. On the other hand, fine tuning of parameters of the elementary bricks of image processing methods is tricky and prevents easy generalization upon various images.

Image processing	Deep learning
Complex parameters tuning	Few variables to adjust
Few resources needed	Uncertain hardware requirements
Varying computational time	Ultrafast[2]
Uncertain segmentation accuracy	Reliable object recognition

Figure 33 – Comparison of main requirements for the solution to be designed. Advantages of one method counteracts drawbacks of the other one.

Next step is to write the functional requirements referential (design brief) and the functional specifications document. Both deep learning and image processing-based algorithms will be used to build prototypes, their performance will be compared afterwards.

References

[1] J. Canny, « A Computational Approach to Edge Detection », in *Readings in Computer Vision*, San Francisco (CA): Morgan Kaufmann, 1987, p. 184‑203.

[2] H. J. Johnson, M. M. McCormick, et L. Iba, « The ITK Software Guide Book 1: Introduction and Development Guidelines Fourth Edition », p. 888.

[3] « Canny edge detector », *Wikipedia*. 19-janv-2018.

[2] Ultrafast, when given hardware that is powerful enough, like Titan X GPU.

[4] W. Gao, X. Zhang, L. Yang, et H. Liu, « An improved Sobel edge detection », in *2010 3rd International Conference on Computer Science and Information Technology*, 2010, vol. 5, p. 67‑71.

[5] « Sobel operator », *Wikipedia*. 22-févr-2018.

[6] « Deriche Edge Detection ». [En ligne]. Disponible sur: http://www.cs.columbia.edu/~jebara/htmlpapers/UTHESIS/node16.html. [Consulté le: 30-mars-2018].

[7] « Prewitt Operator ». [En ligne]. Disponible sur: https://www.tutorialspoint.com/dip/prewitt_operator.htm. [Consulté le: 30-mars-2018].

[8] « Roberts Edge ». [En ligne]. Disponible sur: http://www.roborealm.com/help/Roberts_Edge.php. [Consulté le: 30-mars-2018].

[9] V. Torre et T. A. Poggio, « On Edge Detection », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, nº 2, p. 147‑163, mars 1986.

[10] « Laplace Operator — OpenCV 2.4.13.6 documentation ». [En ligne]. Disponible sur: https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/laplace_operator/laplace_operator.html. [Consulté le: 30-mars-2018].

[11] R. Horan et M. Lavelle, « Laplacian operator - Basic Mathematics », p. 33.

[12] « OpenCV: Smoothing Images ». [En ligne]. Disponible sur: https://docs.opencv.org/3.3.1/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html. [Consulté le: 30-mars-2018].

[13] G. Deng et L. W. Cahill, « An adaptive Gaussian filter for noise reduction and edge detection », in *1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference*, 1993, p. 1615‑1619 vol.3.

[14] Y. Zhu et C. Huang, « An Improved Median Filtering Algorithm for Image Noise Reduction », *Physics Procedia*, vol. 25, p. 609‑616, 2012.

[15] A. Borji, M. M. Cheng, H. Jiang, et J. Li, « Salient Object Detection: A Benchmark », *IEEE Transactions on Image Processing*, vol. 24, nº 12, p. 5706‑5722, déc. 2015.

[16] I. den Bakker, « Battle of the Deep Learning frameworks — Part I: 2017, even more frameworks and interfaces », *Towards Data Science*, 19-déc-2017. [En ligne]. Disponible sur: https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750. [Consulté le: 12-mars-2018].

[17] L. Deng, « A tutorial survey of architectures, algorithms, and applications for deep learning », *APSIPA Transactions on Signal and Information Processing*, vol. 3, ed 2014.

[18] A. Borji, D. N. Sihite, et L. Itti, « What stands out in a scene? A study of human explicit saliency judgment », *Vision Research*, vol. 91, p. 62-77, oct. 2013.