Machine Vision

A starters guide on machine vision for quality control.

KU Leuven Brugge

Contents

1 Introduction **9**
- 1.1 How a computer sees an image . . . 9
- 1.2 Machine vision pipeline . . . 11
- 1.3 Machine vision applications . . . 11
- 1.4 Machine vision specification . . . 11
  - 1.4.1 Specifications of the test object . . . 12
  - 1.4.2 Specifications of the process environment . . . 12

2 Illumination **14**
- 2.1 Humans vs sensors . . . 14
- 2.2 Illumination in machine vision . . . 15
  - 2.2.1 Light source . . . 16
  - 2.2.2 Light filters . . . 16
  - 2.2.3 Light interactions . . . 17
  - 2.2.4 Light focus . . . 18
  - 2.2.5 Light detection . . . 18
- 2.3 Illumination sources . . . 19
  - 2.3.1 LED . . . 19
  - 2.3.2 OLED . . . 19
  - 2.3.3 Halogen . . . 20
  - 2.3.4 Fluorescent . . . 20
  - 2.3.5 Laser diodes . . . 21
- 2.4 Wavelength of light . . . 21
  - 2.4.1 Visible light . . . 21
  - 2.4.2 UV light . . . 21
  - 2.4.3 IR light . . . 22
- 2.5 Light propagation . . . 22
  - 2.5.1 Direct illumination . . . 22
  - 2.5.2 Diffuse illumination . . . 22
  - 2.5.3 Focused illumination . . . 22
  - 2.5.4 Collimated illumination . . . 23
  - 2.5.5 Structured illumination . . . 23
- 2.6 Angle of illumination . . . 24
  - 2.6.1 Front lighting . . . 24
  - 2.6.2 Back lighting . . . 27
- 2.7 Light manipulation . . . 29
  - 2.7.1 Light control filters . . . 29
  - 2.7.2 Diffusing filters . . . 29
  - 2.7.3 Shortpass filters . . . 29
  - 2.7.4 Longpass filters . . . 29
  - 2.7.5 Neutral density filters . . . 30
  - 2.7.6 Daylight cut filters . . . 30

2.7.7 UV and IR cut filters . . . 31
2.7.8 Coloured filters . . . 31
2.7.9 Polarisation filters . . . 32
2.8 Illumination control . . . 32

3 Lenses **34**
3.1 Pin-hole camera . . . 35
3.2 Lens properties . . . 35
3.2.1 Focal length . . . 35
3.2.2 Magnification . . . 36
3.2.3 Field of View (FoV) . . . 36
3.2.4 F-number . . . 36
3.2.5 Depth of Field (DoF) . . . 37
3.2.6 Optical resolution . . . 37
3.3 Lens types . . . 40
3.3.1 Classification based on magnification . . . 41
3.3.2 Classification based on resolution . . . 41
3.3.3 Lenses for multi-chip cameras . . . 41
3.3.4 Liquid lenses . . . 42
3.3.5 Lens groups . . . 43
3.4 Lens mounts . . . 43
3.5 Lens filters . . . 44
3.6 Lens selection criteria . . . 44

4 Sensors **46**
4.1 Sensor properties . . . 47
4.1.1 Pixels . . . 47
4.1.2 Dimensions . . . 48
4.1.3 Bit depth . . . 49
4.1.4 Spatial resolution . . . 49
4.1.5 Spectral response . . . 50
4.2 Sensor features . . . 51
4.2.1 Frame rate . . . 51
4.2.2 Exposure time . . . 51
4.2.3 Partial scan and ROI . . . 51
4.3 Colour cameras . . . 51
4.3.1 Single-chip colour cameras . . . 52
4.3.2 Multi-chip colour cameras . . . 52
4.4 Specialist cameras . . . 53
4.4.1 Dual sensor camera . . . 53
4.4.2 3D camera . . . 53
4.4.3 High-speed camera . . . 54
4.4.4 X-ray camera . . . 55
4.4.5 Line scan camera . . . 55
4.4.6 Multi-spectral cameras . . . 56
4.4.7 Extended wavelength response . . . 57
4.5 Camera interfaces . . . 58

5 Image processing **60**
5.1 Colour spaces . . . 62
5.2 Monadic operations . . . 64
5.3 Diadic operations . . . 65
5.4 Spatial operations . . . 67
5.4.1 Linear spatial filtering . . . 68

5.4.2 Nonlinear spatial filtering . . . 70
5.5 Shape changing . . . 71
5.5.1 Cropping . . . 71
5.5.2 Image resizing . . . 72
5.5.3 Image pyramids . . . 73
5.5.4 Image warping . . . 74
5.6 Histogram equalisation . . . 76
5.7 Thresholding . . . 77
5.7.1 Global thresholding . . . 78
5.7.2 Automatic thresholding . . . 78
5.7.3 Adaptive thresholding . . . 79
5.8 Morphological transformations . . . 80
5.8.1 Erosion . . . 81
5.8.2 Dilatation . . . 82
5.8.3 Opening . . . 83
5.8.4 Closing . . . 83
5.8.5 Skeletonization . . . 83

6 Feature extraction **85**
6.1 What is a feature? . . . 85
6.2 Feature detection . . . 87
6.2.1 Blob detection . . . 87
6.2.2 Corner detection . . . 89
6.2.3 Edge detection . . . 92
6.3 Feature description . . . 96
6.3.1 HAAR . . . 97
6.3.2 HOG . . . 98
6.3.3 SIFT . . . 99
6.4 Feature matching . . . 99
6.4.1 Applications . . . 100
6.4.2 Feature matching algorithms . . . 101
6.5 Features in object recognition . . . 103

7 Object recognition **105**
7.1 Non data-driven techniques . . . 105
7.1.1 Contour detection . . . 106
7.1.2 Hough transformations . . . 106
7.1.3 Template matching . . . 108
7.2 Data-driven techniques . . . 110
7.2.1 Challenges of object recognition . . . 110
7.2.2 Machine Learning-based techniques . . . 111
7.2.3 Deep Learning-based techniques . . . 115

List of Figures

1.1.1 A picture of an elephant [1] . . . 10
1.1.2 A picture like the computer sees it [1] . . . 10
1.4.1 Selection criteria [2] . . . 12

2.1.1 Visible light spectrum [3] . . . 14
2.1.2 Response of the retina [4] . . . 15
2.1.3 Camera sensor with RGB colour filters . . . 15
2.2.1 Illumination ambiguity . . . 16
2.2.2 Light path . . . 16
2.2.3 Spectral Power Distribution (SPD) [5] . . . 17
2.2.4 Magenta filter [5] . . . 17
2.2.5 Interactions [2] . . . 17
2.2.6 Light interaction at different wavelengths [2] . . . 18
2.2.7 Light interaction with an anti-reflection coated lens [2] . . . 18
2.2.8 Quantum Efficiency (QE) [2] . . . 19
2.3.1 Spectra of different LED's [2] . . . 20
2.3.2 Spectra of a halogen lamp [2] . . . 20
2.3.3 Spectra of a fluorescent lamp [2] . . . 21
2.4.1 Detecting colours using a monochrome camera [3] . . . 22
2.5.1 Direct and diffuse illumination . . . 23
2.5.2 Focused and collimated illumination . . . 23
2.5.3 Structured illumination [3] . . . 24
2.6.1 Angle of illumination [3] . . . 24
2.6.2 Bright field illumination [3] . . . 25
2.6.3 Dome and flat dome illumination . . . 26
2.6.4 Coaxial and advanced coaxial illumination . . . 27
2.6.5 Collimated coaxial and dark field illumination . . . 28
2.6.6 Diffuse and dark field backlight . . . 28
2.6.7 Transmissive illumination and collimated backlight . . . 28
2.7.1 Light control film [3] . . . 29
2.7.2 Shortpass filters [6] . . . 29
2.7.3 Longpass filters [6] . . . 30
2.7.4 Neutral density filters [6] . . . 30
2.7.5 Daylight cuts filters [6] . . . 30
2.7.6 UV cut filters [6] . . . 31
2.7.7 Coloured filters [6] . . . 32
2.7.8 Polarisation filters [6] . . . 32

3.0.1 Camera setup [6] . . . 34
3.1.1 Pin-hole camera . . . 35
3.2.1 Focal length [6] . . . 35
3.2.2 Field of View (FoV) [6] . . . 36

3.2.3 Different lens apertures denoted with their F-number [6] . . . 37
3.2.4 Depth of field (DoF) [6] . . . 37
3.2.5 Circle of confusion [7] . . . 38
3.2.6 Modulation transfer function (MTF) [6] . . . 38
3.2.7 Defect aberration [6] . . . 39
3.2.8 Non-corrected and corrected lens for chromatic aberration . . . 39
3.2.9 Spherical aberration . . . 40
3.2.10 Spatial distortion [6] . . . 40
3.3.1 Lenses for multi-chip cameras [6] . . . 42
3.3.2 Liquid lens [6] . . . 42
3.4.1 Lens mounts [8] . . . 43

4.0.1 Object - lens - sensor setup . . . 46
4.1.1 Pixel sizes [9] . . . 47
4.1.2 Charge-couple-device (CCD) [10] . . . 48
4.1.3 Complementary metal oxide semiconductor (CMOS) [10] . . . 48
4.1.4 Sensor dimensions [10] . . . 49
4.1.5 Bit depth [10] . . . 49
4.1.6 Spatial resolution [10] . . . 50
4.1.7 Spectral response [10] . . . 50
4.2.1 Multiple region of interest [10] . . . 51
4.3.1 Single-chip colour camera with Bayer filter [9] . . . 52
4.3.2 Three-chip colour camera [10] . . . 52
4.4.1 Dual sensor and 3D laser camera . . . 53
4.4.2 Stereo camera [11] . . . 54
4.4.3 Time-of-Flight camera [10] . . . 54
4.4.4 X-ray image [10] . . . 55
4.4.5 Line scan and TDI line scan camera . . . 55
4.4.6 Colour line scan camera [10] . . . 56
4.4.7 Multi-spectral camera with five bands [12] . . . 56
4.4.8 Hyper-spectral camera [12] . . . 57
4.4.9 Extended wavelengths and used sensor materials [10] . . . 57
4.5.1 Data transfer hardware interfaces [10] . . . 58
4.5.2 Data transfer hardware interfaces graph [10] . . . 59

5.0.1 Pixel coordinates . . . 60
5.0.2 A grey scale image consisting of one channel . . . 61
5.0.3 A colour image consisting of three channels [1] . . . 61
5.0.4 Image histograms . . . 62
5.0.5 A 3D plot of an image [1] . . . 62
5.1.1 Colour spaces . . . 63
5.1.2 RGB-channels of a picture [1] . . . 63
5.1.3 HSV versus RGB in changing lighting conditions [1] . . . 64
5.2.1 Monadic image processing algorithm [1] . . . 64
5.2.2 Adjusting contrast and brightness on an image [13] . . . 65
5.3.1 Diadic image processing algorithm [1] . . . 65
5.3.2 Chroma-keying: chromacity histogram of the object image [1] . . . 66
5.3.3 Chroma-keying: overview of the diadic operations [14] . . . 66
5.3.4 Motion detection: a) scene image, b) background estimate, c) result of the subtraction (here the value of zero corresponds to the colour white) [1] . . . 67
5.4.1 Spatial image processing algorithm [1] . . . 68
5.4.2 Convolution example [15] . . . 68
5.4.3 Image smoothing: a) original image, image smoothed with b) a 21×21 mean filter and c) a 31×31 Gaussian filter with $\sigma = 5$ [1] . . . 69

5.4.4 A 7×7 mean filter [16] . . . 69
5.4.5 A 3D Gaussian [1] . . . 70
5.4.6 A 7×7 Gaussian filter [16] . . . 70
5.4.7 Image smoothing: a) original image with salt and pepper noise, b) image smoothed with a median filter [1] . . . 71
5.5.1 Image cropping: a) original image, b) ROI due to cropping [1] . . . 72
5.5.2 Image resizing: a) original image, b) subsampled with $m = 7$, c) restored to the original size by pixel replication [1] . . . 73
5.5.3 Image pyramid where each subsequent image is the half of the previous image [1] . . . 73
5.5.4 Image warping of the original image in the middle: a) Scaling, b) Translation, c) Rotation, d) Shearing 74
5.5.5 Image warping: combination of basic techniques into more complex image warping techniques [17] . . 76
5.6.1 Image histograms: a) original grey scale image, b) histogram of a), c) original colour image, d) histograms of c) [18] . . . 76
5.6.2 Image histogram equalization [18] . . . 77
5.7.1 Histogram of a greyscale image of a cherry [18] . . . 78
5.7.2 Image histogram modelled by 2 overlapping normal distributions [18] . . . 78
5.7.3 Otsu thresholding example [18] . . . 79
5.7.4 Adaptive thresholding: a) original image, b) adaptive thresholding with sub-images of size 101×101, c) adaptive thresholding with sub-images of size 21×21 [18] . . . 79
5.7.5 Adaptive thresholding: a) original image, b) global thresholding, c) adaptive mean thresholding, d) adaptive Gaussian thresholding [18] . . . 80
5.8.1 Morphological image processing algorithm [1] . . . 81
5.8.2 Erosion method: a) original image I and structuring kernel $\mathcal{S}$, b) erosion operation, c) erosion result [18] . . . 81
5.8.3 Erosion example [18] . . . 82
5.8.4 Dilation method: a) original image I and structuring kernel $\mathcal{S}$, b) dilation operation, c) dilation result [18] . . . 82
5.8.5 Dilation example [18] . . . 82
5.8.6 Opening example [19] . . . 83
5.8.7 Closing example [19] . . . 83
5.8.8 Skeletonization example [19] . . . 84

6.1.1 Features examples . . . 86
6.1.2 Flat surfaces, edges and corners as features [19] . . . 86
6.1.3 Blob detection with one object . . . 87
6.1.4 Blob detection with multiple objects . . . 87
6.2.1 K-means clustering example [20] . . . 88
6.2.2 Kernel density estimation . . . 89
6.2.3 Mean shift algorithm: Two attraction basins with their respective modes . . . 89
6.2.4 Features and the intensity variation in x and y direction for: a) flat region, b) edge, c) corner . . . 90
6.2.5 Harris corner detection: a) original image, b) corner response [21] . . . 90
6.2.6 SIFT interest point candidates detection: smoothing an image to obtain a scale space, subtract them from each other to obtain the DoG and find the local extrema of the DoG [22] . . . 91
6.2.7 SIFT features: a) interest point candidates, b) discarding low contrast interest points, c) discarding interest points along an edge [22] . . . 92
6.2.8 Edge intensity profile: a) original image, b) grey level profile along horizontal line $v = 360$, c) close-up view of spike at $u \approx 580$, d) derivative of c [1] . . . 93
6.2.9 Continuous and discrete curves: gradient calculation . . . 93
6.2.10 The effect of deriving a noisy signal [23] . . . 94
6.2.11 Convolution with the derivative kernels: a) horizontal image gradient, b) vertical image gradient [1] . 94
6.2.12 Edge detection: a) edge magnitude, b) edge direction [1] . . . 95
6.2.13 Non-local maximum suppression [19] . . . 95
6.2.14 Hysteresis thresholding [19] . . . 96
6.2.15 Canny edge detection [19] . . . 96

6.3.1 Feature vector descriptions: a) Image patch, b) Image gradients, c) Colour histogram, d) Orientation normalisation . . . 97
6.3.2 HAAR-wavelets [24] . . . 97
6.3.3 HOG: the gradients of a patch represented as arrows and as magnitudes and orientations [25] . . . 98
6.3.4 HOG voting principle [25] . . . 99
6.3.5 SIFT features description [19] . . . 99
6.4.1 Feature matching [19] . . . 100
6.4.2 Object localisation [19] . . . 100
6.4.3 Panorama stitching [19] . . . 101
6.4.4 Stereo vision [19] . . . 101
6.4.5 Brute-force matcher: 178 matches [26] . . . 102
6.4.6 Brute-force matcher: top 15 matches [26] . . . 102
6.4.7 FLANN based matcher [27] . . . 103
6.5.1 Features for object recognition . . . 104

7.1.1 Contour detection [19] . . . 106
7.1.2 Hough transform: image space to mc parameter space [28] . . . 106
7.1.3 Hough transform: image space to $\rho\theta$ parameter space [29] . . . 107
7.1.4 Hough transform: accumulator . . . 107
7.1.5 Hough transform example [30] . . . 108
7.1.6 Template matching algorithm [1] . . . 108
7.1.7 Template matching example: a) template, b) source image, c) correlation map with peak at (42, 54), d) bounding box on object with the highest correlation [22] . . . 109
7.2.1 Occlusion . . . 110
7.2.2 Intra-class variation within the "chair" class [31] . . . 111
7.2.3 The classification process in Machine Learning-based methods [32] . . . 111
7.2.4 Plot of two features extracted from a 'cats' and 'dogs' dataset . . . 112
7.2.5 More complex datasets . . . 112
7.2.6 Bag of words [33] . . . 113
7.2.7 Bag of visual words [33] . . . 113
7.2.8 Generating more complex feature vectors [33] . . . 114
7.2.9 HAAR cascades [34] . . . 115
7.2.10 Paradigm shift [32] . . . 115
7.2.11 A neuron of the human brain vs a perceptron of a neural network [35, 36] . . . 116
7.2.12 Different types of activation functions: Sigmoid, Tanh and ReLu functions [37] . . . 117
7.2.13 Single-layer perceptron [38] . . . 117
7.2.14 Multi-layer perceptron [38] . . . 118
7.2.15 Neural network architectures: a) Feedforward neural network, b) Recurrent neural network and c) Convolutional neural network . . . 119
7.2.16 A 6×6 binary image presented as an input for a feedforward neural network . . . 119
7.2.17 Convolution operation on a 6×6 binary image resulting in a feature map . . . 120
7.2.18 Pooling operation on a feature map: max and mean pooling . . . 120
7.2.19 Convolutional neural network architecture [39] . . . 121
7.2.20 Object detection [40] . . . 121
7.2.21 R-CNN architecture [41] . . . 122
7.2.22 Fast R-CNN architecture [41] . . . 123
7.2.23 Faster R-CNN architecture [41] . . . 123
7.2.24 YOLO object detection [41] . . . 124
7.2.25 Object detection algorithms comparison: accuracy and frames per second [42] . . . 124
7.2.26 Open Images [43] . . . 125
7.2.27 ImageNet [44] . . . 125
7.2.28 DOTA [45] . . . 125
7.2.29 ExDark [46] . . . 126
7.2.30 Image segmentation: a) semantic segmentation, b) instance segmentation [47] . . . 126

7.2.31 U-Net architecture [48] . . . 127
7.2.32 Mask R-CNN architecture [48] . . . 127
7.2.33 COCO data set [49] . . . 128
7.2.34 PASCAL VOC [50] . . . 128
7.2.35 Cityscapes data set [51] . . . 128
7.2.36 CamVid [52] . . . 129
7.2.37 Transfer learning applied to CNN's [53] . . . 129

Chapter 1

Introduction

The manufacturing industry is faced with increasingly stringent quality requirements caused by the trend towards stricter safety standards and higher customer requirements. At the same time, guaranteeing quality is made more difficult by the increasing complexity of the products produced and the limits of manual visual checks. These visual checks can be highly subjective and biased due to the changing moods of the operator. This is also caused by the fact that making profound decisions that determine if a product is 'good enough' is rather vague and mostly objective criteria of making a decision is not always available. To this end, many companies are looking for ways to automate quality checks in order to increase accuracy and speed, and to eliminate the subjective decision making in the process.

Machine vision provides solutions to this and can guarantee more stable and continuous quality checks that are based on objective criteria. However, it is far from straightforward to make a machine vision system detect what you, as a human, want it to detect in a robust way. This is due to the vast diversity of applications, the numerous aspects machine vision consist of, and the difficulty of translating human language like 'good enough', into machine vision software. Machine vision can generally be defined as: *the usage of vision algorithms and techniques in order to extract information/knowledge from vision data that can be used to solve real-world problems.* It distinguishes itself from computer vision which focuses on the development of new/better vision algorithms and techniques in order to extract information/knowledge from vision data, rather than using it in an application.

To fully understand a machine vision system, it is first important to understand how we as humans understand how we see the world and how we make decisions based on what we see. Second, it is important to understand how the combination of light, lenses, and camera sensors mimic the human vision system in order to perceive the world and result in an image of this world. Lastly, it is important to understand how a vision algorithm can extract information and knowledge from the enormous data flows that are generated by a vision system and how they can make decisions based on this information and knowledge. This book covers the core machine vision aspects and reviews the existing hard- and software techniques to obtain a deeper insight in how machine vision applications can be developed.

1.1 How a computer sees an image

If one looks at the imaged scene in Figure 1.1.1, one can clearly describe the content of the image as being a fairly old elephant walking in long grass at a sunny day. One could even make some estimations on the size of the elephant or its trunk. As a human, describing what we see in our daily lives is very straightforward. But this is not the case for a computer. If we want a computer to describe a scene and if we want to understand how computers extract useful information from an image, we also need to reflect on why we see what we see. We know that this is an image of an elephant because we probably once saw an elephant, or a picture of it, in our live while someone told us that this is an elephant. So there is definitely some important aspect of learning and remembering things in comparing new images to earlier experience. But still, we need something to compare to this experience. So what 'features' do we extract from this image in order to be able to compare it to what we saw earlier? This is not a straightforward thing to describe anymore as a human. This is the same as describing the difference between a cat and a dog in an image.

Figure 1.1.1: A picture of an elephant [1]

So although it is very straightforward to describe what's inside an image, it is very difficult to describe why we see something in an image. Yet, this is what we should implement in a computer program in order to analyse images and extract information/knowledge from it. Beside the aspect that it is difficult for humans to describe how we see the world and thus also how a computer should see this, there is also the fact that what we see is not the same as what a computer sees. Firstly, a computer only sees a matrix of numbers, instead of a picture, as visible in Figure 1.1.2. These numbers are typically ranging from 0 (black) to 255 (white). When seeing this matrix of numbers, one can imagine that describing the same scene is not so straightforward anymore. So solely based on a matrix of numbers, machine vision algorithms need to extract useful information.

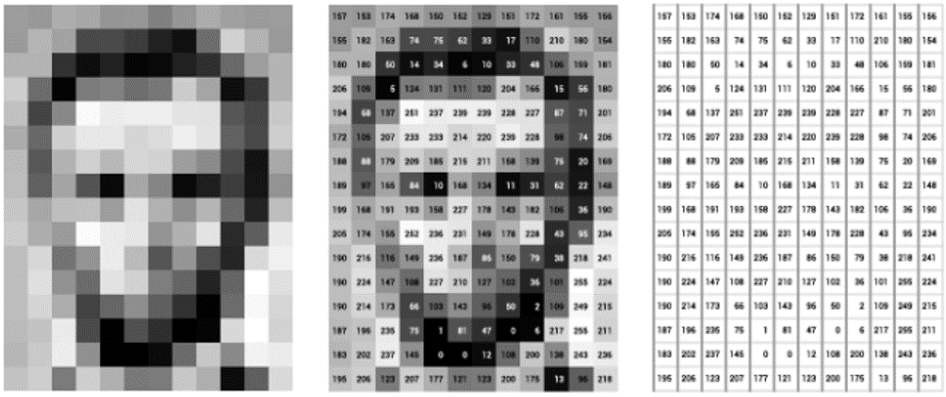


Figure 1.1.2: A picture like the computer sees it [1]

Secondly, there is the problem of variation. One can see an elephant in direct sunlight, under LED light, in the dark, etc. As humans, we have no difficulties in describing a scene in different lighting conditions and we will be able to describe the scene in any lighting condition except in full darkness. But for a computer, a scene under a slightly varied illumination can appear totally different.

So for a computer to extract useful information from an image, plenty of aspects play an important role. A combination of illumination, object properties, lenses, sensors, and filters defines how a scene will appear in an image, which is the starting point for a computer to extract information from. Based on this image, a computer can modify the image in order to highlight and extract the features that it needs to extract from the enormous amount of data that an image consists of. The next section will cover the Machine Vision pipeline that covers all these aspects.

1.2 Machine vision pipeline

A machine vision system consist of numerous aspects that are all of equal high importance. About every aspect, one could give a full graduate course. The full process from light and object to a decision or action can roughly be divided in three main parts: image acquisition, information extraction, and action.

- Image acquisition covers the whole process from light and object to a digital image. It cares about how light propagates from the source, via the object, through the lens, on the camera sensor. Several aspects such as illumination properties, material properties, lens properties, and camera sensor properties are primordial here.
- Information extraction covers the complex aspect of extracting information from a vast amount of data. It covers the image processing steps that attempt to clean and structure the data, as well as the information extraction techniques to obtain insights in the structured data.
- When information is obtained from an image, then it is necessary to do something with this information. The extracted information can be the basis for a decision making process which can subsequently result in an action.

When applying machine vision systems in industry, a division can be made between upstream and downstream systems. In upstream systems, the machine vision system provides feedback to the process which can subsequently use this feedback to modify certain process parameters. Quality check systems are typical examples of upstream systems. In downstream systems, the machine vision system results in commands for a system lower in the hierarchy to for example guide a robot.

A machine vision system can also generally be divided in offline and online systems. In an offline system, there is no need for real-time decision making or data transfer. The data can be gathered at a certain time, but the processing of this data and the decision making can be done later on. In an online system, the data is processed in real-time and also decisions are made in real-time. This requires faster hard-and software.

1.3 Machine vision applications

It goes without saying that there is an enormous variety on possible applications for machine vision. This not only in production or process industry but also in logistics, retail, entertainment, healthcare, etc. Most of the possible applications can be categorised under the following categories:

- Detection of defects
- Detection of presence
- Detection of failures
- Detection of items
- Determining dimensions
- Determining location
- Determining colour
- Determining composition

1.4 Machine vision specification

When developing a new machine vision system for a certain application, it is important to understand all of your requirements and therefore, ask a lot of questions: What object do I want to image? What knowledge do I want to obtain from this image? In what environment will my machine vision setup be situated? Answering these questions starts with understanding the object being inspected and the process environment in which it will be used. Having detailed knowledge about this information helps ensure stable plant operations and ensures that an economically viable application can be implemented. A variety of pertinent factors need to be taken into account when making a selection for an image processing system:

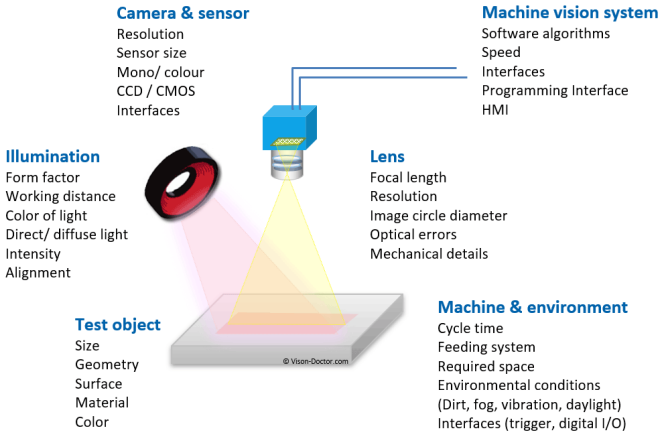


Figure 1.4.1: Selection criteria [2]

1.4.1 Specifications of the test object

- The selection of the perfect camera (resolution, sensor type, etc.) depends on: How big is the object? How big are the recognizable features? Is a colour camera necessary for detection? How precise must the measurement be?
- How far must the image processing system be arranged from the object? Can I inspect from up close or is a handling system (robot, rotary table, linear axis) in the way?
- Which surface and shape does the inspected part have? What material is it made of? Which colour does the object have? By selecting the perfect lighting, defects and objects can be made visible by creating a contrast to the environment.
- How are the characteristics made visible in the camera image? How good is their contrast to the image background? Depending on the task (measuring, counting, presence checks, identification, print inspection, object search, optical character recognition, code reading), different software tools are required.

1.4.2 Specifications of the process environment

- How many parts per second must be checked? Possibly also simpler systems can fulfil these tasks, however, they may not be fast enough.
- How does the feeding work? How fast? Which tolerances in X / Y and Z-axis appear? The software tools must be able to compensate them.
- How much space in the system is available for the inspection cell? Weight and size of the camera (and the lighting) can be important in many systems.
- The process environment (dirt, dust, debris, electromagnetic interference, vibrations) can be strongly disturbing. The image processing system is supposed to be supplemented by appropriate accessories, like protection enclosures, to avoid environmental interference
- How should communication be effected? Which interfaces are required? Must the data be recorded? Is there a need to store images in a database? Which user interface for the staff (HMI) is required? Not every system provides the right possibilities. Also consider the required cable lengths. Not every interface or protocol is capable to communicate over longer distances.

Once all those questions are answered, one can start selecting the right machine vision components and start developing the machine vision setup. The next chapters will review all of the core components machine vision systems

consist of starting with illumination, which is the most important part as illumination provides you with the start of all your data and knowledge that you can possibly get from the system. Further chapters cover the hardware components: lenses and sensors, and the software components: image processing and object detection.

Chapter 2

Illumination

Everything you or a computer can possibly see in an image originates from the different light sources that reach the camera sensor via different media such as the air, filters, object, lenses, etc. Therefore, it is highly important to understand the complete path that light travels starting from the sources and ending in the camera sensor. On this path, there are several aspects that influence the properties of the light that initially originated from the sources. From all the photons that leave the light source, only a small part of them really reaches the camera sensor and contribute to the captured image. The following sections will cover some core aspects that will provide you with insights in the importance and usage of illumination in a machine vision setup.

2.1 Humans vs sensors

As known from physics, light is an electromagnetic wave that consists of two waves that are orthogonal to each other: an electrical wave and a magnetic wave. This electromagnetic wave travels through time and space at the speed of light and at a certain frequency. The speed of light and the frequency of the wave both determine the wavelength of the wave. Depending of the wavelength, light has different properties and can be divided in certain bands. All the possible wavelengths light can have is called the full light spectrum, and within this full spectrum, different spectra can be defined. The most known spectrum is the visible light spectrum, visible in Figure 2.1.1, which reaches from a wavelength of 400 (violet) to 700 (red) nanometers and represents all wavelengths that we as a human can see. But the spectrum of light is definitely not limited to this interval and goes way beyond it both in lower and in higher wavelengths. Light with wavelengths lower than 400 nm is called Infrared light (IR) and light with wavelengths higher than 700 nm is called Ultraviolet light (UV).

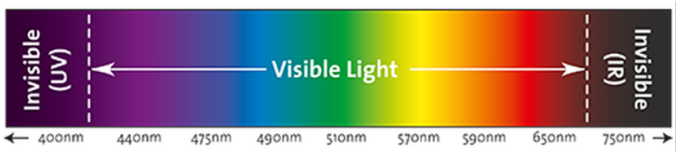


Figure 2.1.1: Visible light spectrum [3]

Humans are trichromats, this means that we observe colours as a sum of three main colours: red, green, and blue. Our eyes have an area, called the retina, that is sensitive to light falling onto it via our lens. This retina consist of cone cells and rod cells. The rod cells are more active at night and only respond to intensity, regardless of the wavelength. Cone cells are more sensitive during the day and respond to particular colours. We have three types of cone cells in our retina: one of them is sensitive to light with large wavelengths (Red-cone), one is sensitive to light with medium wavelengths (Green-cone), and one is sensitive to light with short wavelengths (Blue-cone). The response of the three cells is visualised in Figure 2.1.2. The total response of the cones to incoming light is a 3-vector [R,G,B] that represents the outputs of the three different type of cones which is called the tristimulus.

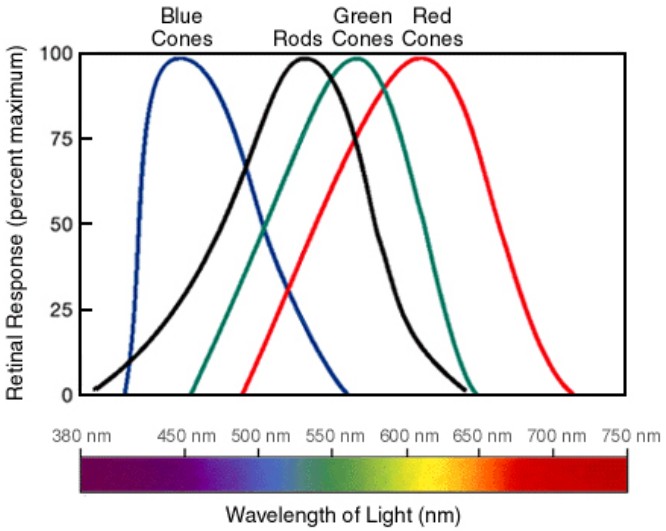


Figure 2.1.2: Response of the retina [4]

The retina of the eye has a central region, called the foveal, which contains most of the cone cells: 60% sense red, 33% sense green, and only 2% sense blue. A sensor in a camera has a regular array of light sensitive photosites on a silicon chip instead of cones and rods cells. Each photosite outputs a signal proportional to the intensity of the light falling on it. In a colour camera, the photosites are covered by colour filters which pass either red, green, or blue light to mimic the response of the three human cones. A sensor with colour filters can be seen in Figure 2.1.3. The spectral response of the filters is equivalent to the cones' response

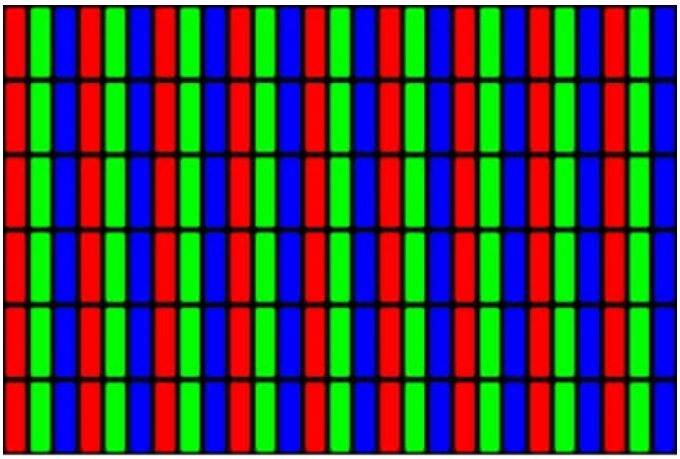


Figure 2.1.3: Camera sensor with RGB colour filters

2.2 Illumination in machine vision

As humans, we can easily describe an object irregardless of the illumination that falls onto the object. If we look at a white page at daylight, or at the same white page at night under a street light, we will define it as a white page in each situation. However, when imaging a white page in those two situations, the image will appear totally different and thus also a machine vision algorithm will interpret the image differently. This is something critical to consider when developing a machine vision system. A good example is shown in Figure 2.2.1. The figure shows the same shoe in two different lighting conditions. In the real scene, the shoe just looks the same in both illuminations. But when imaged, the shoe looks totally different in the two cases. This illustrates the importance of being cautious when working with colours in a machine vision setup. What you see as a human is not what the computer sees. This is the

reason why it is primordial to choose the illumination wisely. A core rule in machine vision is to better choose good illumination and a bad camera than a an expensive camera and bad illumination. Good illumination can eliminate the requirement for costly and time-consuming image processing and can eliminate any variation of environmental light.

Figure 2.2.1: Illumination ambiguity

Considering the importance of illumination when designing a machine vision setup, it is important to consider the full path the light travels through all machine vision components, as visualised in Figure 2.2.2. Light emits from a source, passes through a medium like air or filters, interacts with the object, gets (partially) reflected by this object, passes through air and optional filters again, passes through a lens, and finally gets collected by the camera sensor. During this path, lots of photons get lost, and all these components have an effect of the final image that is taken. The next sections cover each of these aspects.

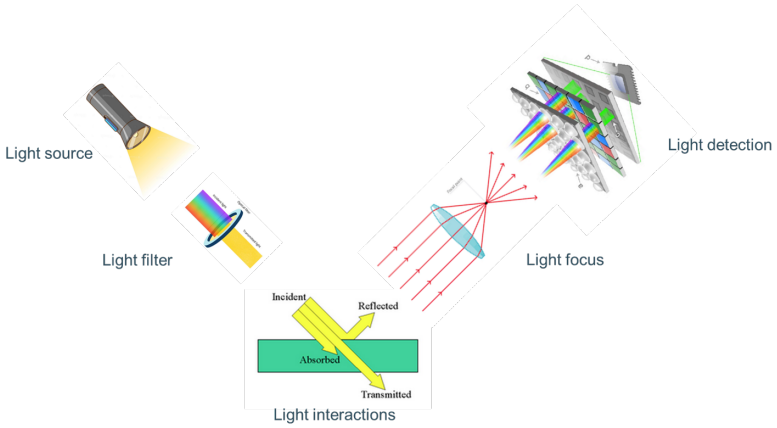


Figure 2.2.2: Light path

2.2.1 Light source

The light source is the origin of all possible information that can reach the camera and that can be visible in the image. Any light source can be characterised by its Spectral Power Distribution (SPD) which indicates the spectral irradiance (power per unit area per unit wavelength) in function of the wavelength. Figure 2.2.3 shows the Spectral Power Distribution of daylight and of a white LED.

2.2.2 Light filters

Light filters filter out a specific range of wavelengths. Absorption filters absorb a specific range of wavelengths and transmit the remaining. Interference filters reflect a specific range of wavelengths and transmit the remaining. Figure 2.2.4 shows the effect of an absorption magenta filter.

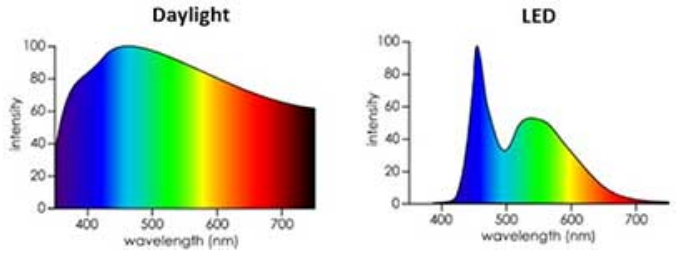


Figure 2.2.3: Spectral Power Distribution (SPD) [5]

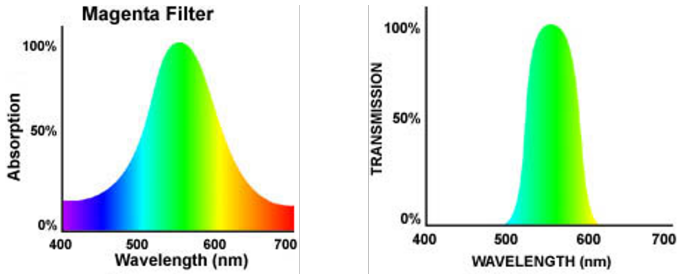


Figure 2.2.4: Magenta filter [5]

2.2.3 Light interactions

When light hits an object, some interaction effects that are dependent on the properties of the object occur. The effects can be absorption, reflection, remission (when it gets reflected after absorption), transmission when the light passes through the object, fluorescence when UV-light hits a fluorescent object that emits visible light, diffusion when light rays get scattered at the surface, diffraction, and polarisation. Usually one of these effects never appears alone, but it is always a combination of several effects. In effect, only a percentage of the light travels to the camera sensor. Figure 2.2.5 shows the different types of interactions.

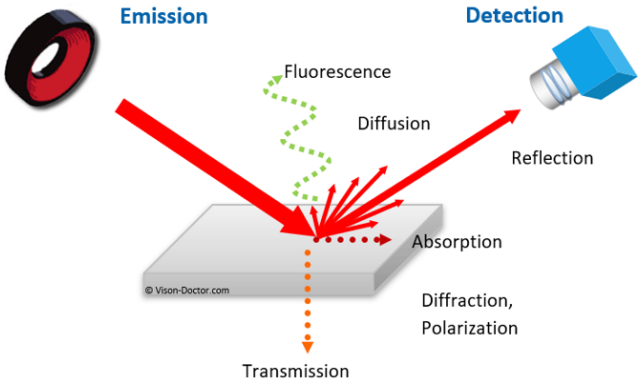


Figure 2.2.5: Interactions [2]

The light interactions are also dependent of the wavelength of the light. The shorter the wavelength of the light used (blue or UV light), the higher is the portion of the stray light at the surface; absorption, remission and transmission are reduced. Surface defects are clearly visible. The longer the wavelength of the light used (red or IR radiation), the more the material is penetrated and surface phenomena visible in the camera image diminish. Viewing below the surface or looking through a body is possible. This dependence is visible in Figure 2.2.6. Also the surface texture, geometry, and colour play a role in the interaction effects. Surface processing may cause more strayed light like for example matte and porous surfaces. Sloped or curved surfaces could also direct the light away from the

camera. And finally different colours will reflect different wavelengths. Red objects will reflects red wavelengths and green objects will reflect green objects.

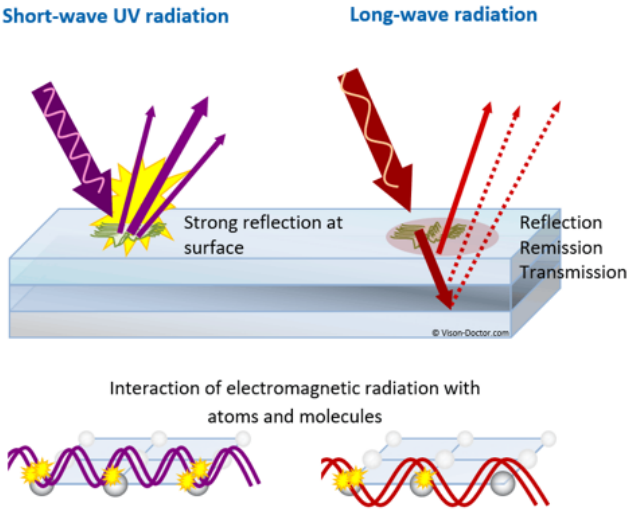


Figure 2.2.6: Light interaction at different wavelengths [2]

2.2.4 Light focus

After the interaction of light with an object, the light gets directed towards the camera. However, before reaching the camera sensor, the light gets focused by a lens and also gets partially absorbed. The spectral transmission of a lens is dependent on the wavelength. The transmission rate of normal glass is only 95%. However, the loss of 5% adds up when different lenses are used in combination, which is often the case. In high-quality lenses, anti-reflection coatings can be used. This is visualised in Figure 2.2.7.

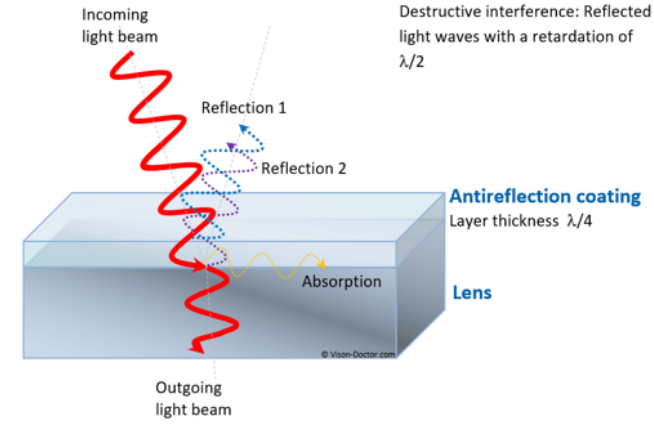


Figure 2.2.7: Light interaction with an anti-reflection coated lens [2]

2.2.5 Light detection

After the lens, the light can finally reach the camera sensor. Attention should be payed that single-chip colour cameras also use filters on the sensor to be able to define a coloured image. The filters again have a certain transmission rate in function of the wavelength. The remaining light is collected onto the camera sensor. The sensor converts light

photons to an electrical signal. This is done by a certain efficiency called the Quantum Efficiency (QE) that denotes how effective the camera converts photons into electrons. The Quantum Efficiency of some CMOS camera sensors is depicted in Figure 2.2.8. The sensitivity of the camera is also dependent on the pixel size, the larger the pixels, the more light can be converted, causing a higher sensitivity but a lower resolution.

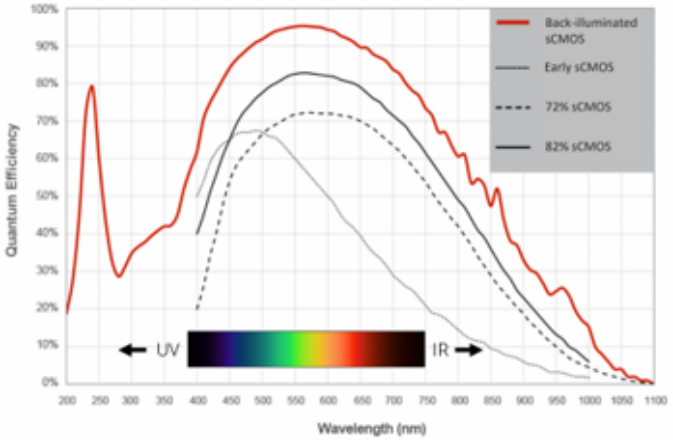


Figure 2.2.8: Quantum Efficiency (QE) [2]

2.3 Illumination sources

Different types of light sources are available for use in machine vision applications. The spectrum of the illumination source depends on the mechanical composition. Today, LED lights cover a major part of industrial applications. Halogen and fluorescent lamps are only frequently used in applications which require large-area illumination, like robot applications or line scan camera inspections where much light and/or the illumination of large areas is required as well.

2.3.1 LED

In an Light Emitting Diode (LED), a wafer of semi-conductor material is doped to create a PN junction and when current is passed through this junction, light is created by the annihilation of electron/hole pairs. The material used in the construction of an LED's PN junction determines the energy of the emitted photons and hence the LED's colour. LED's produce narrow range of wavelengths in the visible spectrum that we perceive as single pure colour. White LED light is created in a similar way to fluorescent lamps, using phosphorescent and fluorescent materials to re-emit part of the light in a wider spectrum, creating white light. This light has a high colour temperature as a consequence of this method and is very suitable for vision applications. The advantages of LED's are a low power consumption, low heat generation, extremely long service life of 30,000 to 100,000 hours, small in size, and applicable in various colours. The disadvantages are that they must be cooled sufficiently and change their colour temperature over their service life. Figure 2.3.1 shows the spectrum that is emitted by several coloured LED's.

2.3.2 OLED

Organic LED or OLED is a more ecological form of LEDs where the light emitting material is organic. The biggest advantage of this technology is the possibility to produce a large light emitting surface in a very thin housing of only a few mm high. As with LEDs, OLEDs offer the possibility of controlling the intensity and the exposure time. OLED technology illuminates intrinsically and as such heat generation is a far lesser problem. This technology is still very young, however the technology is progressing rapidly with new products being announced delivering a longer lifetime (up to 50,000 operating hours), higher efficiency up to 100 Lm/W in a range of colours and illuminator sizes. Figure 2.3.2 shows the spectrum that is emitted by a halogen light.

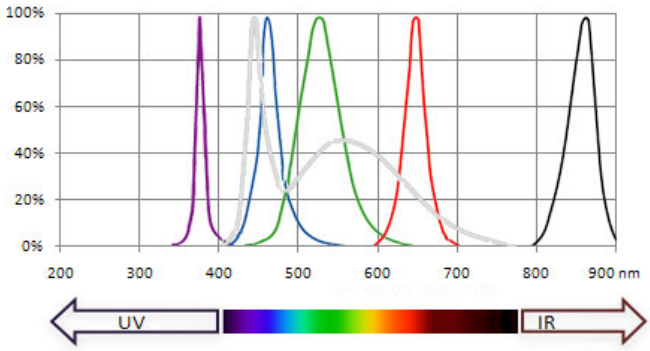


Figure 2.3.1: Spectra of different LED's [2]

2.3.3 Halogen

Halogen light is useful for applications where bright light is required. In halogen lamps, a live conductor like tungsten is heated in a protective gas atmosphere to produce electromagnetic light. Halogen lamps can be used where large areas or lots of light is required for industrial image processing. A halogen light can be combined with a reflector to form a point source. When used in a machine vision application, as halogen lights tend to be hot, illumination can be delivered using optical fibers. The lamp can be mounted remotely to provide bright and uniform light required for line scan camera applications. Some disadvantages are the strong heat generation, high power consumption, large design, and single emission spectrum.

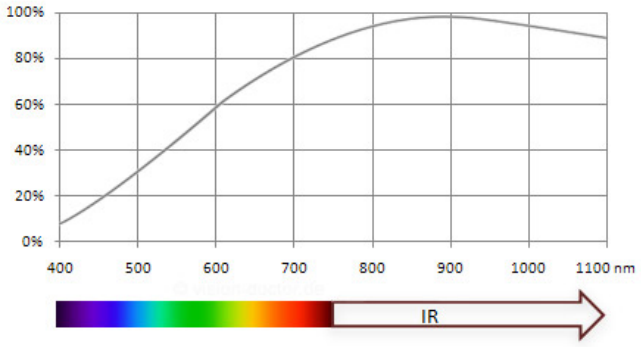


Figure 2.3.2: Spectra of a halogen lamp [2]

2.3.4 Fluorescent

Fluorescent lamps produce light by a chemical reaction. Electric current is passed through the mercury gas filled lamps/tubes to produce ultraviolet light that reacts with the phosphor coating inside the glass to emit fluorescent or glowing white light. A ballast is used to provide a high initial voltage to start the process and then limit the lamp current to sustain the light. For machine vision applications, a high-speed ballast is used to drive the lamps to reduce flicker and make the light intensity consistent. These lamps can illuminate large area's and are relatively inexpensive, are available in many shapes and sizes, and have a reasonable life span. This light is best suited for diffuse lighting, that is, covering a larger area around the object to be imaged. Unlike LED arrays, they are not suitable for directional lighting. Figure 2.3.3 shows the spectrum that is emitted by a fluorescent light.

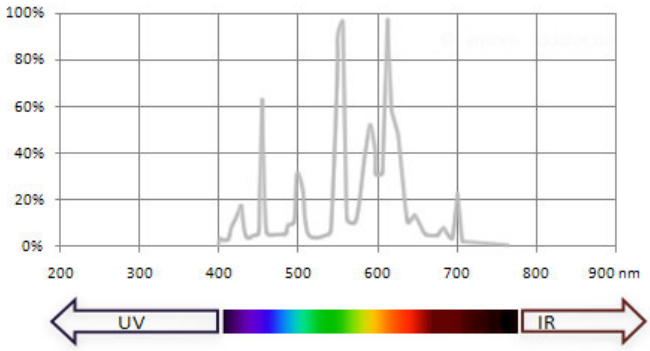


Figure 2.3.3: Spectra of a fluorescent lamp [2]

2.3.5 Laser diodes

Lasers (Light Amplification by Stimulated Emission of Radiation) provide high-intensity light sources. A laser is created when electrons in atoms are excited using electric current. The electrons absorb the energy and when returning to their original state give out photons of lights through a process known as spontaneous emission. Lasers are very powerful but can be easily focused and controlled. They are widely used to provide structured light, such as a dot, circle, line, or any other geometric pattern. Laser lights are widely used for precision measurements. They are expensive and require safety precautions as they are hazardous to health.

2.4 Wavelength of light

When designing a machine vision application, it is important to consider the emitted wavelengths of the light source. This in function of the information that one wants to subtract from the image. Sometimes one wants to obtain information that is not visible to the human eye. In this case, light sources that only emit in the visible spectrum will not provide enough useful information. Instead light sources that emit IR, NIR, SWIR, or UV light are necessary. In addition, also the camera sensor should be selected appropriately in order to be able to image the emitted wavelengths.

2.4.1 Visible light

In most of the machine vision applications, one wants to detect something that is visible by the human eye. In this case, visible light can be used together with a camera that is sensitive to visible light. But still, a choice could be made between white, red, blue, etc. light depending on the application. Sometimes it can be useful to use red or blue light instead of white light. This in order to avoid chromatic aberration which results from the fact that different wavelengths reflect at different angles when passing through a lens. Using coloured light can also increase the intensity of objects with the same colour in the image, improving the contrast with the background. A red object will be more visible in an image when illuminated with red light as this object reflects the red light that falls onto it. A red object will turn less visible when illuminated with, for example, blue light as most of this light will be absorbed by the object. This principle also makes it possible to detect colours using a monochrome camera that only detects intensities. This is visible in Figure 2.4.1.

2.4.2 UV light

Standard techniques using visible light can resolve features to approx 0.5 µm. To visualise accurately smaller features, the visible light spectrum is no longer sufficient. The reduction of the wavelength is the most practical. Using UV light + UV-sensitive camera sensor allows ultra fine resolution images. UV can also be used in fluorescence applications. Some chemical substances absorb UV and re-emit it as visible (mostly green) light.

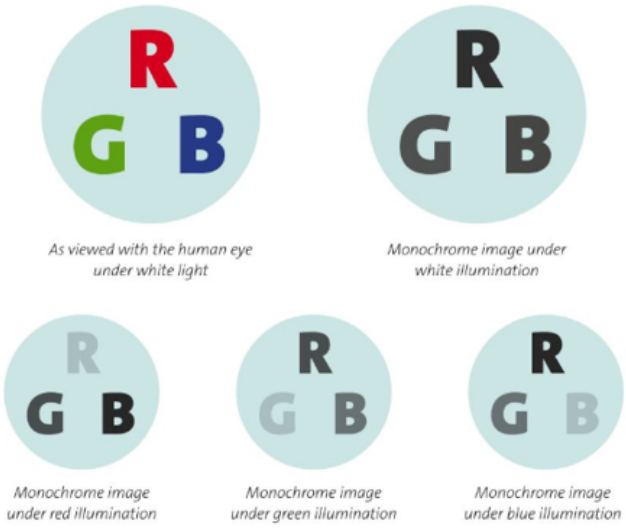


Figure 2.4.1: Detecting colours using a monochrome camera [3]

2.4.3 IR light

IR light can be used to diminish colour effects in monochrome images as IR light reduces colour information. Using IR light it is possible to identify black prints with much better contrast. IR light, however, does not work in sunlight as this has higher levels of IR in its spectrum.

2.5 Light propagation

In addition to the illumination source, it is also important to consider how the light rays propagate through space to the object. There are several types of light propagation that can be manipulated using the correct filters, lenses, diffusers, or mirrors. The most common types are described below.

2.5.1 Direct illumination

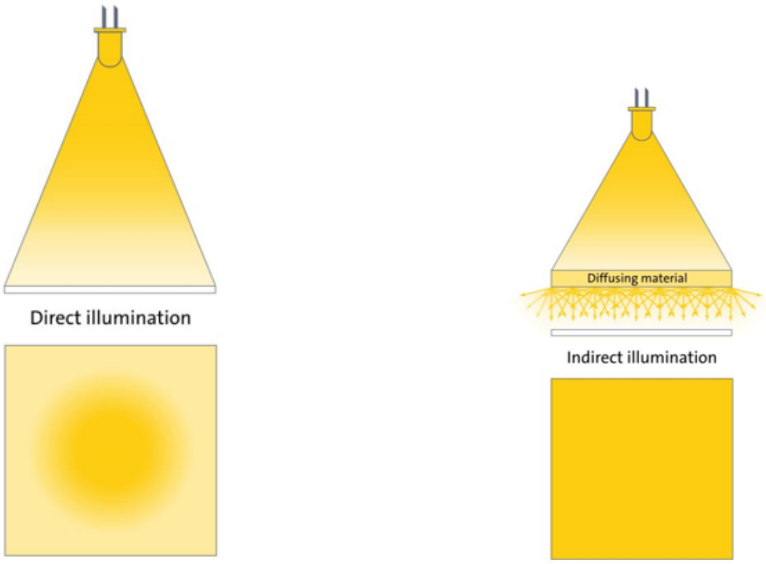
Using direct light, the light rays have an uninterrupted path between the source and the object. Camera and lighting are arranged in such a way that the camera is at the reflection angle of the lighting (angle of incidence = angle of reflection). In this way as much incident light is detected as possible. This works especially well in case of reflecting materials. Bright areas appear bright, dark areas dark in this way. This type of light propagation is visible in Figure 2.5.1a.

2.5.2 Diffuse illumination

In diffused light, the light rays pass through a diffuser between the source and the object. This type of light eliminates reflections and shadows, as light is directed onto the object from multiple angles. Diffusers provide a large illumination area that avoids glares. This type of light propagation is visible in Figure 2.5.1b.

2.5.3 Focused illumination

In focused illumination, light rays are focused on a specific target to increase light intensity at this target. This can be used, for example, together with a line scan camera that only requires a thin line of light as illumination as only a 1-pixel wide line is detected by the sensor at once. However, it is important for all line scan camera applications that the light is as bright as possible and has an extremely high frequency in order to avoid flickering. This type of light propagation is visible in Figure 2.5.2a.

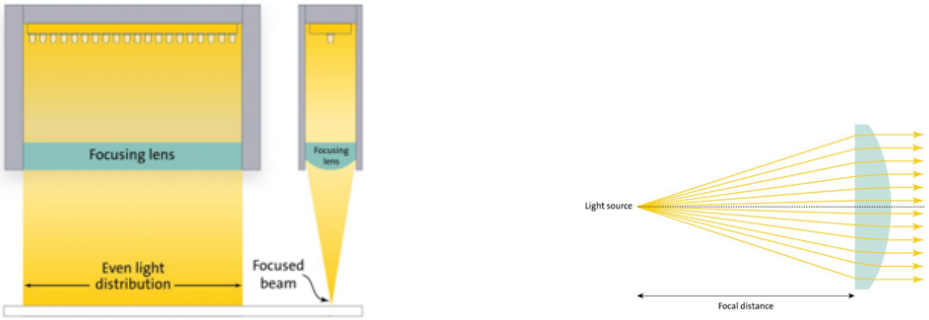


(a) Direct illumination [3] (b) Diffuse illumination [3]

Figure 2.5.1: Direct and diffuse illumination

2.5.4 Collimated illumination

When rays of light in a beam are parallel, the light is collimated. Using light from a collimated light source is especially useful for detecting shallow flaws and dents on flat, reflective objects. It is also used to enabling the reprojection of objects without blurred edges and with clear contrast A collimated beam of light is sometimes referred to as telecentric illumination. This type of light propagation is visible in Figure 2.5.2b.



(a) Focused illumination [3] (b) Collimated illumination [3]

Figure 2.5.2: Focused and collimated illumination

2.5.5 Structured illumination

Structured light goes one step further than focused light. Structured light can output very narrow and sharp illuminated lines produces by lasers or structured LED's. It can be used to project accurate patterns onto a target. Distortions in the line can be translated to height variations (laser triangulation) to obtain height information. This can also be used in alignment applications. This type of light propagation is visible in Figure 2.5.3.

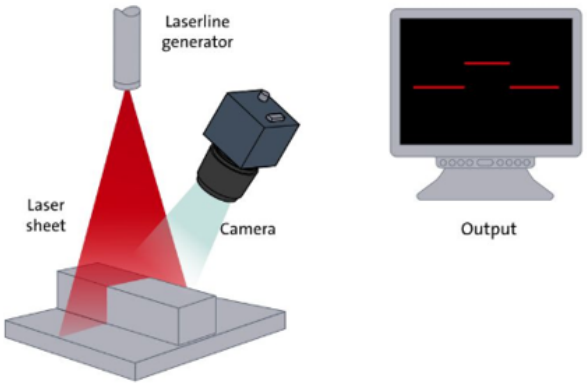


Figure 2.5.3: Structured illumination [3]

2.6 Angle of illumination

Dependent on the type of features that need to be accentuated, the angle at which light rays strike the object need to be adapted. Several angles of illumination exist to this purpose and are described below. A visualisation of all techniques is depicted in Figure 2.6.1.

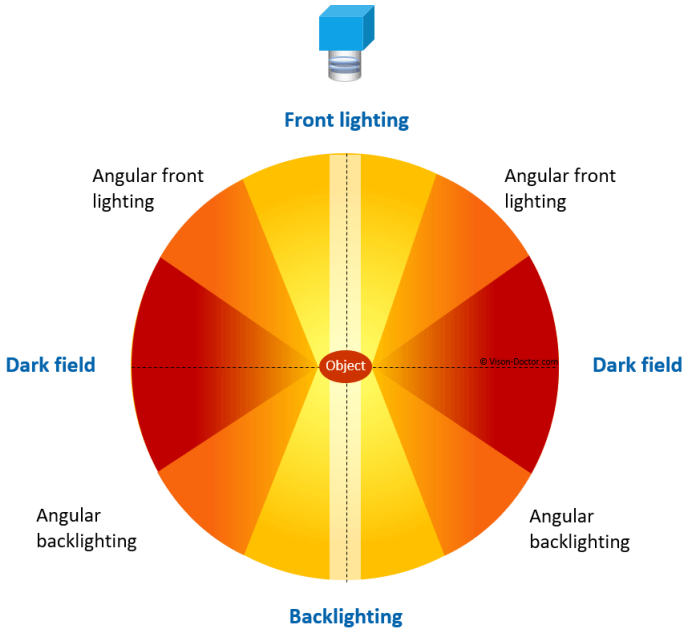


Figure 2.6.1: Angle of illumination [3]

2.6.1 Front lighting

In most of the applications an object needs to be illuminated to be visible at high contrast in the camera. To this purpose, the light should ideally be placed at the same side as the camera, pointing towards the object. This type of illumination is called front lighting. Within front lighting, there are several distinctions that can be made.

2.6.1.1 Bright field illumination

Bright field illumination refers to where light rays from the source are reflected directly into the lens as visible in Figure 2.6.2. This is the most common technique for illuminating diffuse, non-reflective objects. The term bright field refers to the mounting position of the illuminator. If a camera is positioned pointing towards a plain mirror, the bright field is the area in which any reflected light is within the FoV (field of view) of the camera. Light from the side can be radiated at a relatively wide or narrow angle. The influence on the camera image can be significant, in an extreme case the image information can almost be inverted.

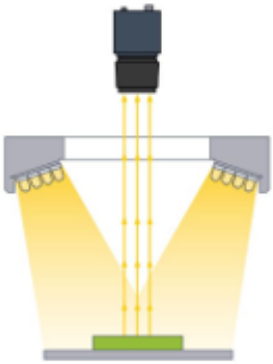


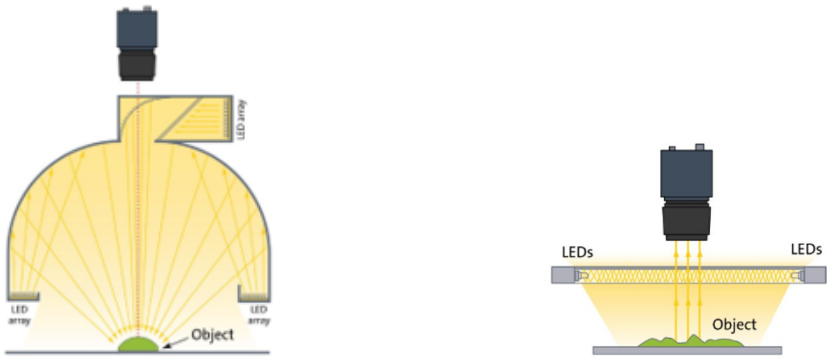
Figure 2.6.2: Bright field illumination [3]

Direct incident illumination This simple lighting technique can be used for non-reflective materials which strongly scatter the light due to their matte, porous, fibrous, non-glossy surface. Direct incident light can be any type of extensively beaming lights. Ideally, a ring light is chosen for smaller illuminated fields, that can be arranged around the lens. Shadows are avoided to the greatest extent due to the absolutely vertical illumination. Halogen lamps and large fluorescence illumination can be used, too. Direct incident light illuminations are inexpensive and do not require any complicated design. All types of lighting such as halogen, fluorescent or LED lighting can be used. Ideal to illuminate large areas. Several lights can be combined, too. Direct light is usually bright because of a directly radiating surface with low beam angle and strong reflection.

Diffuse incident illumination Diffuse light is necessary for many applications, as soon as reflective, polished, glossy or metallic objects must be tested. It is particularly difficult if these surfaces are no longer perfectly flat, but individually shaped, wrinkled, curved or cylindrical. In the easiest case, a simple incident light illumination can be equipped with a diffuser, e.g. an LED ring light with diffusing panel. In this way, the directed light can be homogenised slightly better. Individual LED's, for example, are no longer discernible on a reflecting material. The light, however, is still quite strongly directed and mainly comes from the direction of the illumination.

Dome light A dome light is able to produce truly diffuse light, avoiding reflections on the component to the greatest extent. Due to the homogeneous incidence of light from all spatial directions, irregularities and scratches are also illuminated very softly, uniform material appears homogeneous. These images typically appear rather "featureless". Reflections, shadows, surface irregularities, scratches, etc. are suppressed. Up to a surface angle of almost + / - 45 degrees, structures are illuminated uniformly. Other materials and components with different colours are clearly visible only now. A dome illumination is perfect to inspect metallic and shiny materials and to suppress reflections. Variations of this illumination are only slightly arched lights that can illuminate a much smaller solid angle. Tunnel-like dome illuminations have an elongated dome shape and are ideal for illuminating cylindrical objects. Dome illumination enables an even and homogeneous illumination of the object and can be used to illuminate complex reflective shapes such as ball bearings, printed foil packaging or CDs. A dome light is visible in Figure 2.6.3a.

Flat Dome A different method of producing diffuse, uniform illumination is to use a flat dome. This special form of front light illumination combines the advantages of coaxial front lights with those of common dome illuminations. A flat dome enables a shadow-less illumination of most objects. Using a special hole template on the diffusor, perfect light scattering is secured and homogeneous, diffuse light is spread over the object. As this illumination occupies less room than a dome light, it can also be used for applications with limited space offering close to dome performance. Application examples range from inspection of blister packages to control of metal stamping machines monitoring the cuts on reflective surfaces. In general this technique is suitable for any application requiring an even, shadow-less, diffuse illumination of curved uneven surfaces. A flat dome light is visible in Figure 2.6.3b.



(a) Dome illumination [3] (b) Flat dome illumination [3]

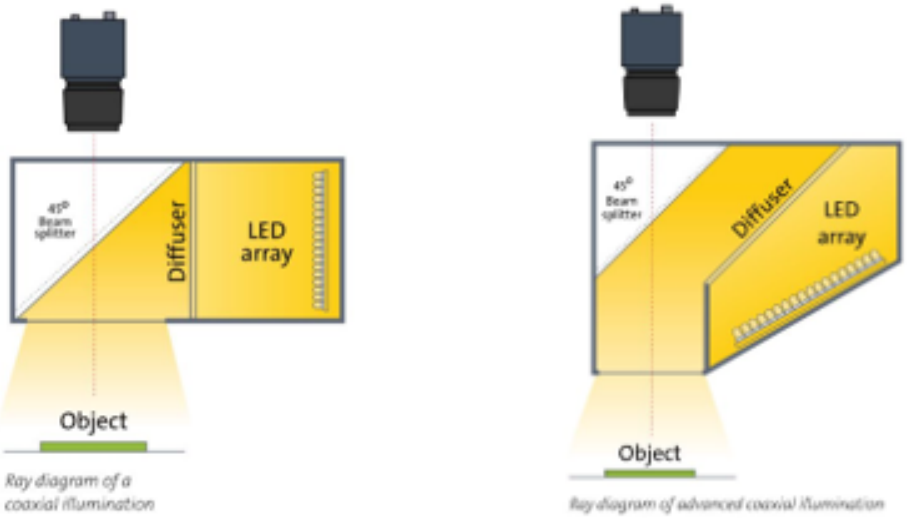
Figure 2.6.3: Dome and flat dome illumination

Coaxial illumination This technique is used for illuminating reflective surfaces. A beam splitter or a semi-transparent mirror is used to divert the light from a light source that is fixed on the side (coaxial), so that it is projected almost parallel to the optical axes of the camera onto the object. This is visible in Figure 2.6.4a. The object reflects the light which reaches the camera through the semi-transparent mirror. The technique of coaxial illumination is ideally suited to plain reflective, 'mirror-like' objects with no (or very little) profile or any surfaces which have diffuse backgrounds. Examples include PCB inspection, reflective labels, polished silicon wafers or print inspection. The image as a whole looks very homogeneous and has only a general linear loss of light, the further the mirror moves away from the lamp. A more advanced form of coaxial illumination exists that uses an additional reflection chamber. This causes that light is not emitted directly from the source onto the object via the semi-transparent mirror and results in a much more homogeneous illumination. This is useful for highly reflective and uneven surfaces. This technique is visible in Figure 2.6.4b.

Collimated coaxial illumination In coaxial illumination, the light rays are somehow parallel to the optical axis which results in a homogeneous illumination of the object. But this is not always sufficient for high-accuracy applications as this type of illumination can cause blurred edges of an object. A more suitable techniques makes sure that the light rays are really parallel to the optical axis by sending them through a lens. This is called collimated coaxial illumination and is visible in Figure 2.6.5a. This technique is useful for detecting shallow flaws and dents on flat, reflective objects. Wherever the surface is perfectly flat, the light is reflected back through the lens, and makes these areas appear bright in the image. Any other surface features appear dark in the image.

2.6.1.2 Dark field illumination

Dark field illumination refers to any illumination where the light rays from the source are not reflected into the lens, but only a proportion of light that is scattered by an uneven surface as visible in Figure 2.6.5b. This technique is mainly used to highlight surface defects, scratches or engraving. Dark field illumination usually uses a low angle ring light that is mounted very close to the object. Dark field is the opposite of bright field because most of the light reflected from the surface of the target will fall outside the FoV of the camera with the camera only "seeing" scattered light that is reflected by a defect on the surface.



(a) Coaxial illumination [3] (b) Advanced coaxial illumination [3]

Figure 2.6.4: Coaxial and advanced coaxial illumination

2.6.2 Back lighting

In some applications, it can be interesting to illuminate an object from the opposite side of the camera. This for example to create a binary image where the object appears black and the surroundings appear white. This type of illumination is called back lighting. Within back lighting, there are several distinctions that can be made.

2.6.2.1 Diffuse backlight illumination

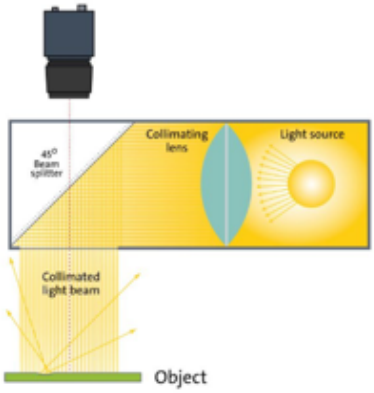
A technique positioning a diffuse illuminator behind the object to give a silhouette. This technique is normally used if 'through holes' or shapes are of interest, or the object is generally opaque with darker/brighter variations in the areas of interest. This principle is visible in Figure 2.6.6a.

2.6.2.2 Dark field illumination

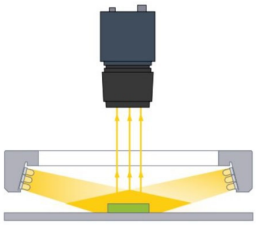
This technique, visible in Figure 2.6.6b, uses a dark field light behind a transparent object to reveal surface features. It can be used on transparent targets such as glass or plastic that need to be checked for defects such as scratches, or to check the quality of embossed or raised lettering. By using this technique, any surface details are revealed as bright artefacts against a dark background. This type of backlighting also makes it possible to selectively examine either the upper or lower surface of a transparent object. The images produced with dark field backlighting benefit from a very high contrast which simplifies any subsequent image processing. The example shows a transparent bowl with heavy scratching on the inside surface. Another form of dark field backlighting uses transmissive illumination, where light is 'injected' into a transparent medium and is only reflected when the surface of the material is scratched, cracked or in some other way deformed.

2.6.2.3 Transmissive illumination

In some specific cases such as with transparent objects, light can be coupled to a transparent medium and is only reflected when the surface of the material is scratched, cracked, etc. due to the transmissive properties. An illustration is visible in Figure 2.6.7a.

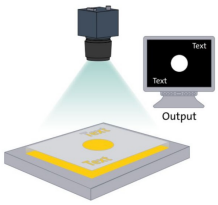


(a) Collimated coaxial illumination [3]

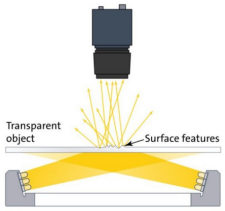


(b) Dark field illumination [3]

Figure 2.6.5: Collimated coaxial and dark field illumination



(a) Diffuse backlight illumination [3]

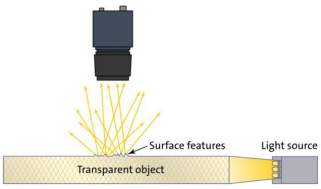


(b) Dark field backlight illumination [3]

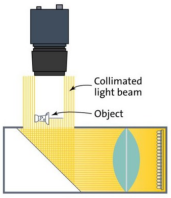
Figure 2.6.6: Diffuse and dark field backlight

2.6.2.4 Collimated backlight illumination

This form of backlighting is useful for measurement tasks that require a homogeneous illumination with no scattered light. It produces a distinct silhouette even when imaging transparent items. When using a standard backlight, stray light can sometimes contaminate the edges of an object which produces an indistinct outline that is difficult to measure accurately. This principle is visible in Figure 2.6.7b.



(a) Transmissive illumination [3]



(b) Collimated backlight illumination [3]

Figure 2.6.7: Transmissive illumination and collimated backlight

2.7 Light manipulation

Properties of the illumination can be constrained by the addition of simple and effective filters both on the light source and on the lens. This is called light manipulation. The following sections mention the most common filters.

2.7.1 Light control filters

Light control filters pseudo collimates light rays with low-cost material instead of using costly lenses. This is visible in Figure 2.7.1.

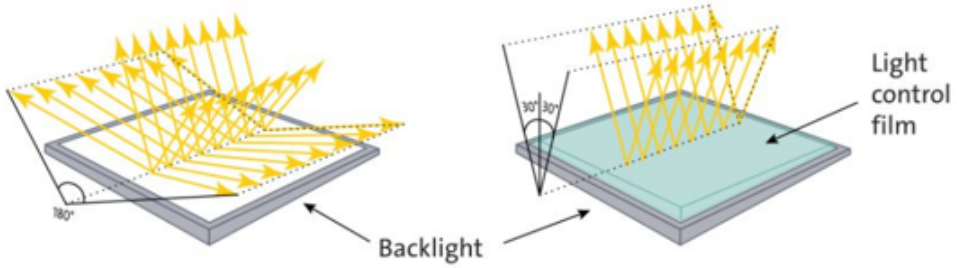


Figure 2.7.1: Light control film [3]

2.7.2 Diffusing filters

Diffusing filters: Allow direct LED illuminations to be turned into diffuse, indirect units.

2.7.3 Shortpass filters

Shortpass filters, visible in Figure 2.7.2, are designed with sharp cut-off wavelength from 615 nm to 850 nm. A broadband anti-reflection coating avoids troublesome reflections. Shortpass filters are ideal for fluorescence and wavelength sorting applications.

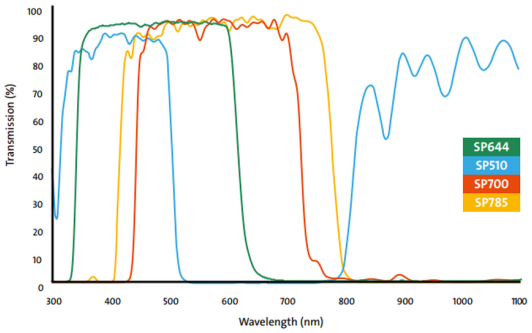


Figure 2.7.2: Shortpass filters [6]

2.7.4 Longpass filters

Longpass filters, visible in Figure 2.7.3, are designed with sharp cut-off wavelength from 400 nm to 820 nm. A broadband anti-reflection coating avoids troublesome reflected light. Longpass filters are ideal for fluorescence and wavelength sorting applications.

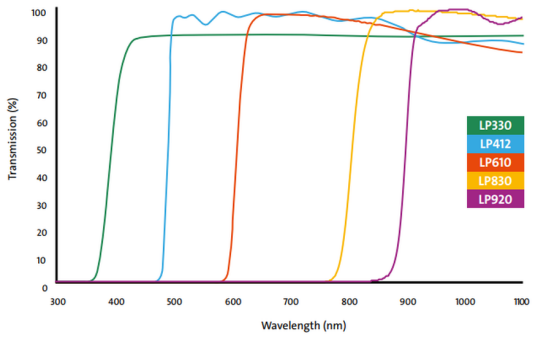


Figure 2.7.3: Longpass filters [6]

2.7.5 Neutral density filters

Neutral density or grey filters, visible in Figure 2.7.4, are used to reduce the transmission of light equally over the entire visible spectrum. Thus the light intensity is reduced without affecting colours and contrast. Typically used in front illuminated applications where light bleed can not be avoided simply by closing the aperture of the lens or by adjusting the camera settings.

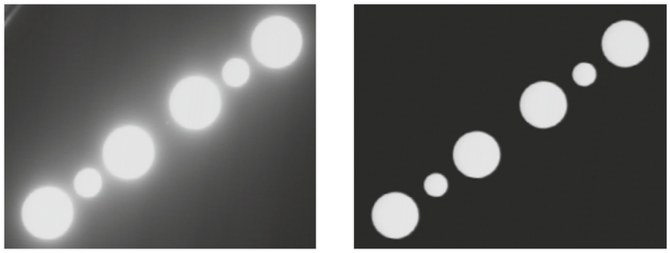


Figure 2.7.4: Neutral density filters [6]

2.7.6 Daylight cut filters

Daylight cut filter, visible in Figure 2.7.5,s block visible light from contaminating the image information. The resultant image is only generated by the IR ranges of the spectrum to which the camera sensor is sensitive. The main use of these lenses is to make the application independent of ambient light such as the changing intensity of sunlight. By combining daylight cut filters with IR illumination, the resultant image is very consistent while visible light levels are changing.

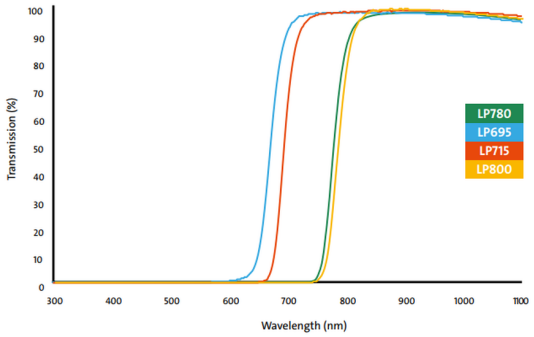


Figure 2.7.5: Daylight cuts filters [6]

2.7.7 UV and IR cut filters

UV cut filters, visible in Figure 2.7.6, are longpass filters that block the UV wavelength range and transit the visible light. They are used in front of a lens on a camera to protect the sensor from UV light, in the case when the camera offers no internal protection. In addition they are often used to protect the front lens. IR cut filters are shortpass filters that block infrared light and transmit the visible light and are typically used with or built-in to colour cameras to mimic the photo responsivity of the human eye which is not sensitive to IR hence making colours appear similar to human perception.

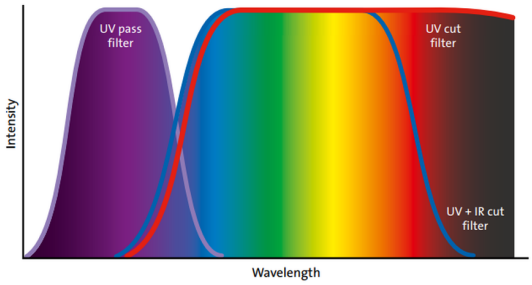


Figure 2.7.6: UV cut filters [6]

2.7.8 Coloured filters

Coloured filters, visible in Figure 2.7.7, in a wide range of tones are used to block certain wavelength ranges of the visible light. In machine vision applications they are used in front of a lens on a camera to improve image contrast. The appropriate wavelength band (colour) has to be selected with respect to the colour of the object and the type or colour of the illumination. Since they affect the composition of the transmitted light, they are usually not suitable for colour cameras. In the case where a colour filter is used on a monochrome camera, objects in the filter colour appear bright, whereas the complementary colour appears dark.

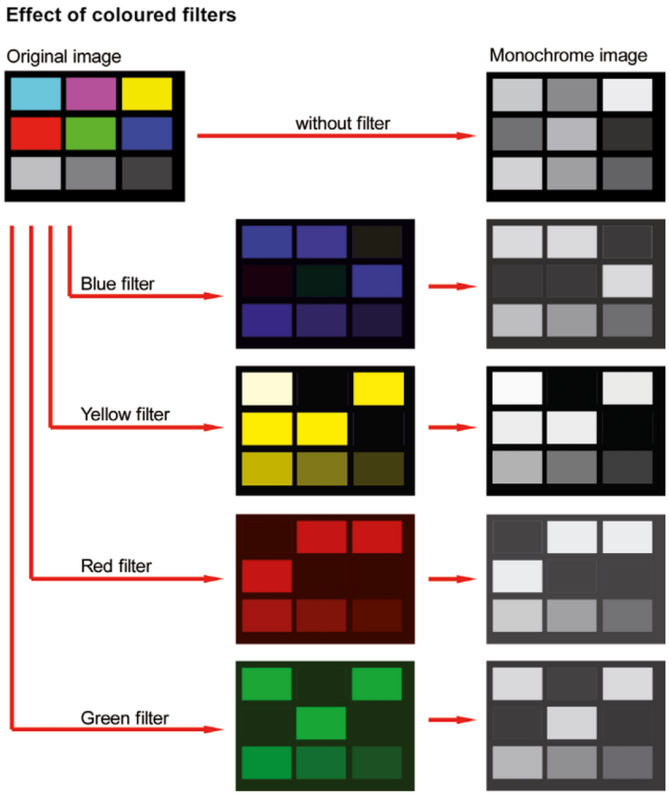


Figure 2.7.7: Coloured filters [6]

2.7.9 Polarisation filters

Polarisation filters, visible in Figure 2.7.8, are used to reduce reflections and highlight materials that inherently polarise light. Light in one polarisation direction is removed, whereas light in the respective perpendicular direction is transmitted. In practical applications one polarisation filter is mounted in front of the illumination source and another one in front of the lens. Polarisation filters on a lens in a rotating threaded mount allow the direction of polarisation to be adjusted. In machine vision applications mainly linear polarisation filters are used.

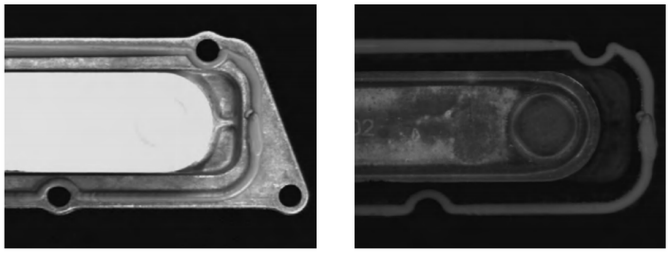


Figure 2.7.8: Polarisation filters [6]

2.8 Illumination control

In several machine vision setups it may not always be required to have the illumination activated permanently. Some techniques exist for illumination control that may be more appropriate such as constant illumination where the light is activated permanently during the whole machine vision process. This is highly energy-consuming as it may not be necessary to activate the light when no image is taken. Pulsating illumination where the light is activated only

when an image is taken. The light pauses between object arrivals. This increases the lifespan of the light. This can be particularly interesting when using LED's as they have the ability to deliver an intensity much higher than their nominal intensity when activated for a short time. Between the pulses they must be allowed to cool down to prevent overheating. Strobe illumination where the light is activated at a certain frequency to prevent motion blurring. Strobe light at a frequency of a fast moving object makes the object to appear static in the image.

Chapter 3

Lenses

When light is emitted by the light source and reflected by the object towards the camera, it reaches the lens. The lens makes sure that the wide bundle of light gets concentrated on the tiny camera sensor. This optical process causes several effects, aberrations, and deformations on the imaged object with respect to the real object. Several lens properties need to be taken into account to make a decision on the suitable lens type for the vision application. Also the selection of the lens should be done simultaneously with selecting the camera sensor. Between the lens and the camera sensor there are two barriers: the aperture and the shutter. Both work in concert to control how much light makes its way to the sensor of a digital camera. The actual design and arrangement of the aperture, shutter, and sensor vary depending on the camera. Figure 3.0.1 illustrates a basic camera setup. In the this chapter, the core components of the lens are discussed and some selection criteria and calculations are provided.

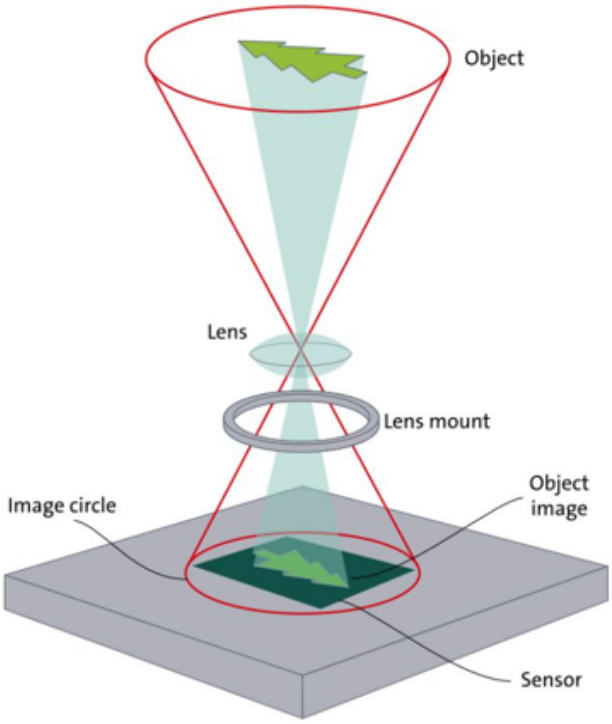


Figure 3.0.1: Camera setup [6]

3.1 Pin-hole camera

The concept of a lens is an extension to the so called 'Camera obscura' phenomenon which uses a pin-hole camera which is able to create an inverted image of a scene on the wall of a darkened room as visible in Figure 3.1.1. Light from the light source reflect onto the object and passes through the pin-hole (aperture) on the wall. The smaller the pin-hole the sharper the image. However, the smaller the pin-hole also the less light that can be projected onto the wall causing the image to appear dark. This darkeness-sharpeness trade-off is mitigated using a lens. Because by using a lens, light can be bundled and converted and thus a larger aperture can still provide a sharp image while letting more light through. This is visible on Figure 3.1.1.

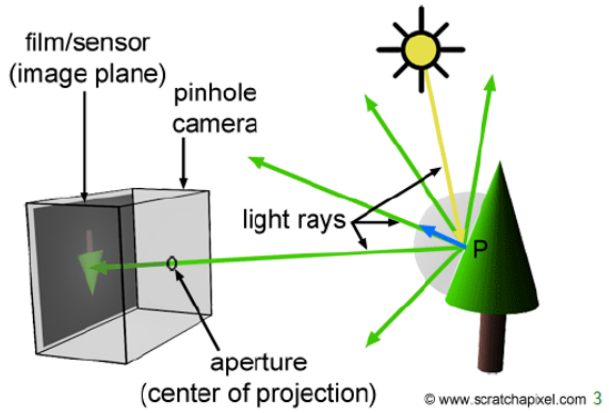


Figure 3.1.1: Pin-hole camera

3.2 Lens properties

3.2.1 Focal length

The focal length of a lens is the distance between its effective centre and the point at which light from infinity will be focused to a single point. The sensor of the camera is positioned behind this point on what is called the image plane. Figure 3.2.1 depicts the focal length of a lens. The focal length of a lens defines its magnification, the smaller the focal length, the larger the magnification. Lenses with fixed elements are called fixed focal lenses. Lenses with a movable focusing unit allow the focal length to be changed. Zoom lenses also have a variable focal length, which implies a variable magnification. Most machine vision lenses have a fixed focal length. The use of zoom lenses is much less prevalent in machine vision applications. These lenses change their focal length by moving lens elements which can make them mechanically unstable and less suitable for making accurate, repeatable measurements.

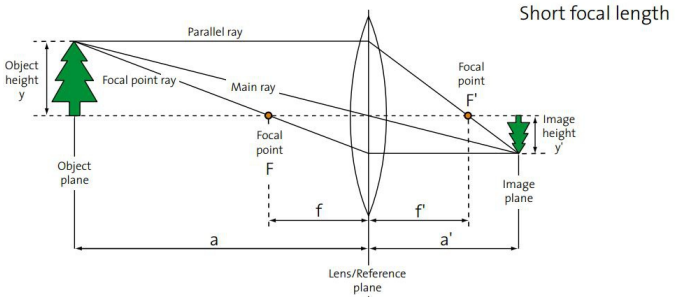


Figure 3.2.1: Focal length [6]

3.2.2 Magnification

The magnification $\beta^{'}$ of a lens describes the ratio between image size and object size and has a direct connection to the focal length $f^{'}$ of the lens and the working distance a. The following diagram shows how to calculate the ratio between object size, working distance and focal length of the lens. The ratio between working distance a, image distance a' and focal length f is calculated as follows:

$$Magnification\,(\beta^{'}) = \frac{a^{'}}{a} = \frac{y^{'}}{y}$$

$$Thin\,lens\,formula: \frac{1}{f}^{'} = \frac{1}{a} = \frac{1}{a'}$$

3.2.3 Field of View (FoV)

The Field of View (FoV) is the maximum area or maximum size of an object that the camera can image. This is direclty related to the magnification and is highly influenced by the focal length and the sensor size. The larger the focal length, the larger the magnification, and the larger an object can be to be focused on the same camera sensor. And the larger this camera sensor, also the larger an object can be to be focused on this sensor. This Field of View is denoted in square meters (m^2) but can also be denoted by an angle. Then there is referred to as the Angular Field of View (AFoV). Figure 3.2.2 shows the FoV for a short and a long focal length.

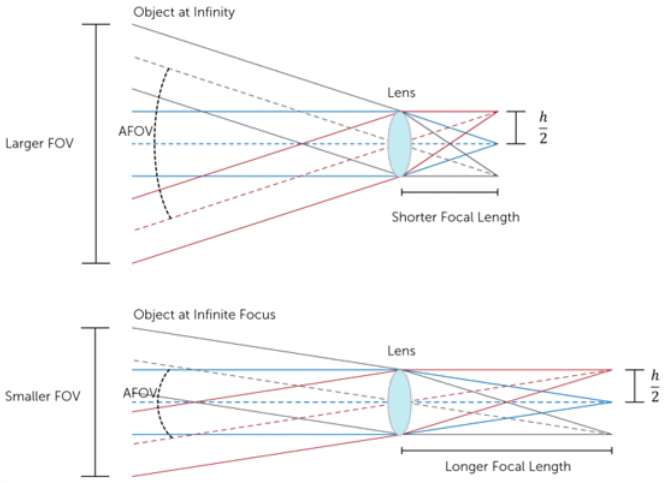


Figure 3.2.2: Field of View (FoV) [6]

3.2.4 F-number

Before the light reaches the lens it passes through the aperture. The aperture is an opening that light travels through. The aperture can be made larger or smaller in order to adjust the amount of light that can pass through it. Therefore it is directly responsible for the image brightness. The amount of light that passes through the lens is also dependent on the exposure time which is the time that the aperture allows light to pass through. The amount of light that can enter the camera at a given exposure time is, beside the aperture, also dependent on the focal length and is denoted by the F-number or F-stop. The F-number is the focal length of the lens divided by the effective aperture diameter and is denoted by f/N (this is a notation, not a formula) with N the F-number that defines the reduction of the maximum opening diameter of the lens. Lenses are always quoted with their maximum aperture and thus by their lowest F-number. An F-number of $f/1.4$ means a reduction of the maximum opening diameter by 1.4 (square root of 2) and corresponds to halving the opening surface and thus also the amount of light that can pass through the lens. With an F-number of 1.4, the effective diameter of the aperture can be calculated by $D = f/1.4$ with f being the focal length. Several F-number increments are depicted in Figure 3.2.3.

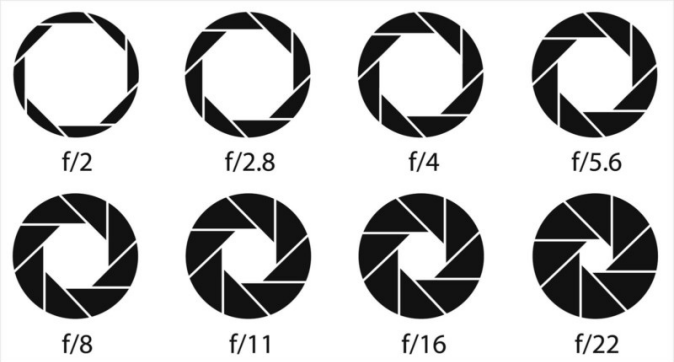


Figure 3.2.3: Different lens apertures denoted with their F-number [6]

3.2.5 Depth of Field (DoF)

Besides the Field of View (FoV) denoting the size of the area that a camera can image, there is also the Depth of Field (DoF) which denotes the range of object distances in the direction of the optical axis in which the lens can produce a sharp image. This Depth of Field is highly influenced by the aperture opening, pixel size, and magnification. The larger the aperture, the more light that can pass, but the lower the DoF. This results from the fact that light rays deform more at the edges of the lens and deform less at the centre of the lens. With a large aperture, more area of the lens further from the centre is used to let light pass through. This causes higher deformation of the image and thus a lower DoF. With a small aperture, less area of the lens further from the centre is used to let the light pass through. This causes a lower deformation of the image and thus a larger DoF. This is visible in Figure 3.2.4. Below a certain aperture opening (¿F8.0), the sharpness of the image starts to be limited due to diffraction.

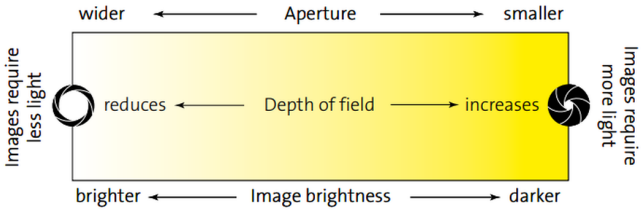


Figure 3.2.4: Depth of field (DoF) [6]

3.2.6 Optical resolution

The optical resolution of a lens refers to the ability of the lens to resolve details in the object that is being imaged. The optical resolution of a lens is defined by the MTF (Modulation transfer function) relating to resolution along with the optical distortion.

3.2.6.1 Circle of Confusion (CoC)

An ideal lens would image a point light source as a point in the image. However, in practice, a lens always produces a circle instead of a point. This optical spot caused by a cone of light rays not coming to a perfect focus is called the Circle of Confusion (CoC) and is e measure of the quality of the lens. The smallest such spot that a lens can produce is called the circle of least confusion. This principle is shown in Figure 3.2.5.

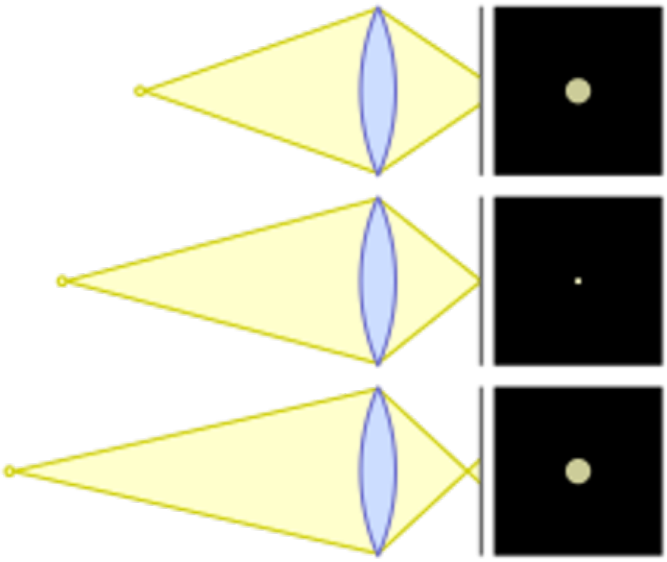


Figure 3.2.5: Circle of confusion [7]

3.2.6.2 Modulation transfer function (MTF)

The modulation transfer function is the quantitative description of the image quality of a lens, considering all aberration. To define the MTF, the lens reproduces lines (grids) with different distances (spatial frequency in line pairs/mm). An example of an MTF is visiblz in Figure 3.2.6. The loss of contrast due to the optical reproduction is shown in the MTF-graph for each spatial frequency. The more line pairs/mm that can be distinguished, the better the resolution of the lens. The ideal lens would produce an image which perfectly matches the object, including all details and brightness variations. In practice this is never completely possible as lenses act as low pass filters. For any lens there is a point at which the modulation is zero. This limit is often called the resolution limit and is usually quoted in line pairs per millimetre (lp/mm), or with some macro lenses in terms of the minimum line size in µm. The MTF deteriorates as you move away from the centre axis of the lens towards the edges.

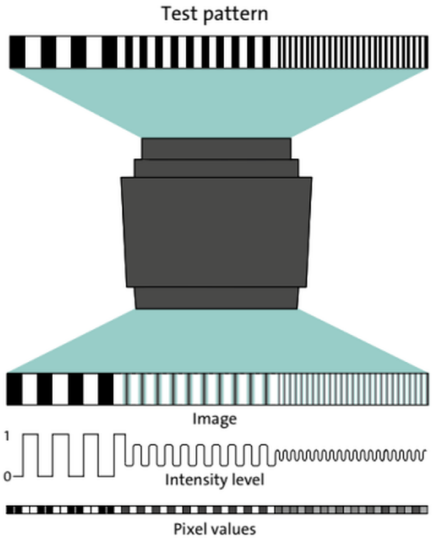


Figure 3.2.6: Modulation transfer function (MTF) [6]

3.2.6.3 Aberration

Using a perfect lens, all rays of light from a single point on an object will be focused to a single point on the image plane. Any lens behaviour that stops this happening is classified as aberration. Aberration can be avoided by decreasing the aperture. There is a rule of thumb that says that most of the lenses have an optimal sharpness at an aperture 2 stops less than the maximal aperture. There are several types of aberrations of which the most common are described below.

Defect aberration Any physical defect or contamination on the surface of a lens will introduce image degradation as the light reaching these defects will not focus correctly. This is shown in Figure 3.2.7.

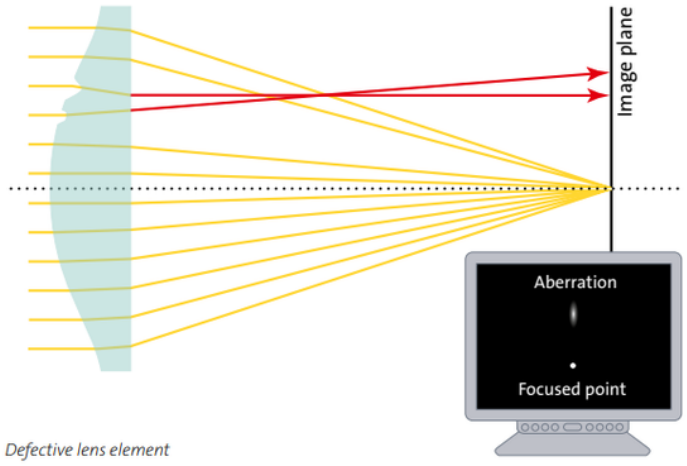
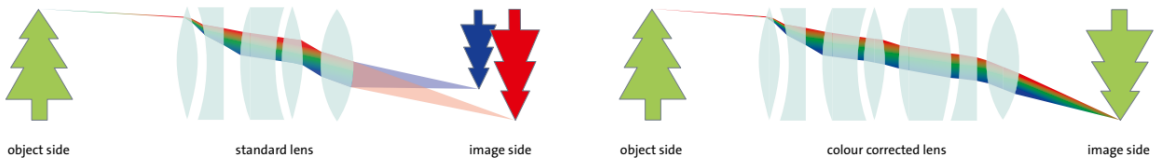


Figure 3.2.7: Defect aberration [6]

Chromatic aberration Chromatic aberration is one of the most common faults observed in spherical lenses which is caused due to light of different wavelengths which breaks at different angles at the lens surface (dispersion). This can be seen in Figure 3.2.8a. The result is a coloured fringe or halo surrounding the image, with the halo colour changing as the focal point of the objective is varied. The effect will increase towards the edges of the image. Chromatic aberration can be avoided by using monochrome light or colour corrected lenses to counteract the effect, this can be seen in Figure 3.2.8b.

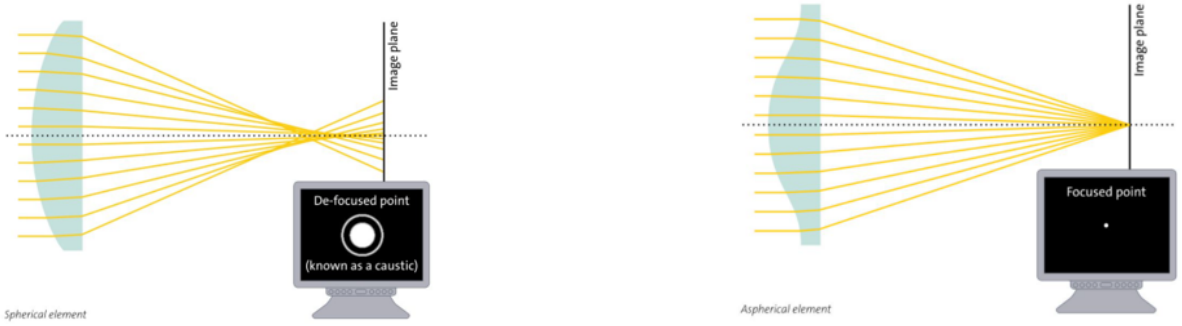


(a) Chromatic aberration on a non-corrected lens [6]

(b) Chromatic aberration corrected lens [6]

Figure 3.2.8: Non-corrected and corrected lens for chromatic aberration

Spherical aberration Spherical aberration is caused by spherical lenses which let parallel beams strike at different positions from the optical axis. Each parallel ray has a different focal point dependent on its distance from the optical axis and result in light from infinity that gets projected onto a circle instead of on a point. A solution is to use parabolic lenses which are much more expensive than spherical lenses. This is shown in Figure 3.2.9.



(a) Lens with spherical aberration [6]

(b) Lens without spherical aberration [6]

Figure 3.2.9: Spherical aberration

3.2.6.4 Spatial distortion

Beside the errors that can be introduced due to incorrect lenses, there are also effects that results from the converging or diverging properties of a lens. These properties cause that straight lines are not projected as straight lines. This causes a deformation of the image with respect to the real object. Such spatial distortions such as 'Pin cushion' and 'Barrel' distortion exist showed in Fiure 3.2.10. A higher focal length will cause more such distortions as it has a stronger converging effect and the light hits the sensor from a greater angle. These distortions can be resolved in software using or by using complex lens designs.

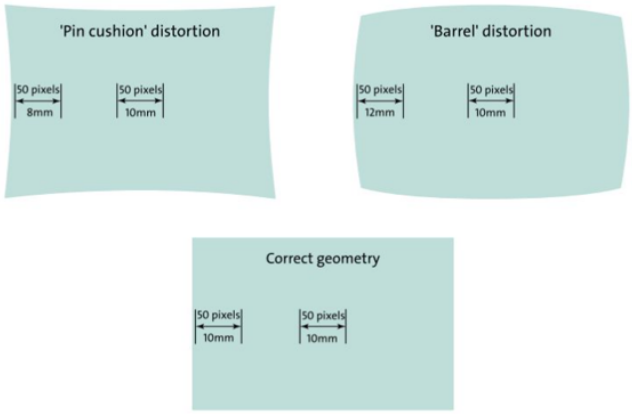


Figure 3.2.10: Spatial distortion [6]

3.3 Lens types

Different lens types exist for the large variety of machine vision applications. Depending on the application, size of the object, required optical resolution, required accuracy, etc. a selection of the lens type should be made. This section reviews the most common lens types that exist. All lens types can be generally divided into fixed focus, and zoom lenses. Fixed focus lenses only have one fixed focal distance that cannot be changed, and thus a fixed working distance and magnification. This causes that the mechanical lens structure is rather simple and is less

sensitive to mechanical defects. Zoom lenses have the ability to change the focal length. This may require a complex mechanical mechanism that can be highly prone to failures or misalignment. Lens types can be classified based on the magnification or the resolution.

3.3.1 Classification based on magnification

Lenses can be developed for different magnifications. Macroscopic lenses produce a magnification smaller than one which causes that the image produced is smaller than the object. These can be categorised in:

- Extreme wide angle lens (Fisheye) - (10mm-24mm)
- Wide angle lens (24mm-35mm)
- Standard lens (35mm-70mm)
- Telephoto lens (70mm - ...)

Microscopic lenses produce a magnification larger than one which causes that the image produced is larger than the object. Telecentric lenses produce a magnification equal to one which causes that the image is not magnified. This has some interesting properties as these lenses do not suffer from spatial distortion as they have an infinite focal length. These lenses are used for specialist measurement applications where perspective projections and incorrect image scaling is not allowed. An important consideration is that the front aperture of the lens needs to be as a minimum the size of the required FoV or size of the object. This makes telecentric lenses for imaging large objects unpractical and very expensive.

3.3.2 Classification based on resolution

Lenses can be developed for different resolutions. There are standard resolution lenses that are most widely used for resolutions less than about a megapixel. But also high resolution lenses exist for improved imaging performance that are especially suited for cameras with a small pixel size or for precise measurement applications.

3.3.3 Lenses for multi-chip cameras

Most colour cameras are monochrome cameras augmented with special colour filters. These have the disadvantage that the resolution of the image decreases due to the need of interpolation among the colour pixels. This can be resolved by a colour camera that splits the incoming light into several spectra using a lens that consists of a set of prisms. These lenses eliminate chromatic aberration. This is shown in Figure 3.3.1.

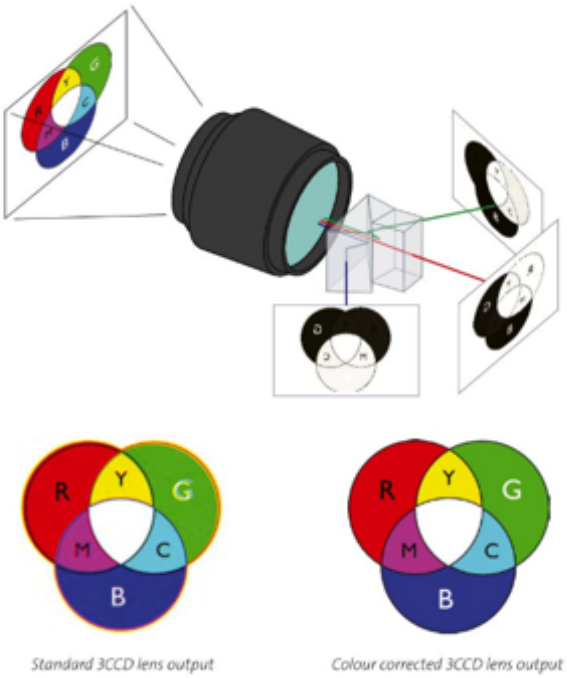


Figure 3.3.1: Lenses for multi-chip cameras [6]

3.3.4 Liquid lenses

When there is a need for rapid change of the focal length of the lens, liquid lenses can be used. These lenses are made of elastic polymer-based materials that can change focus within milliseconds by applying a current to a diaphragm that changes the shape of the lens. The principle of a liquid lens is shown in Figure 3.3.2.

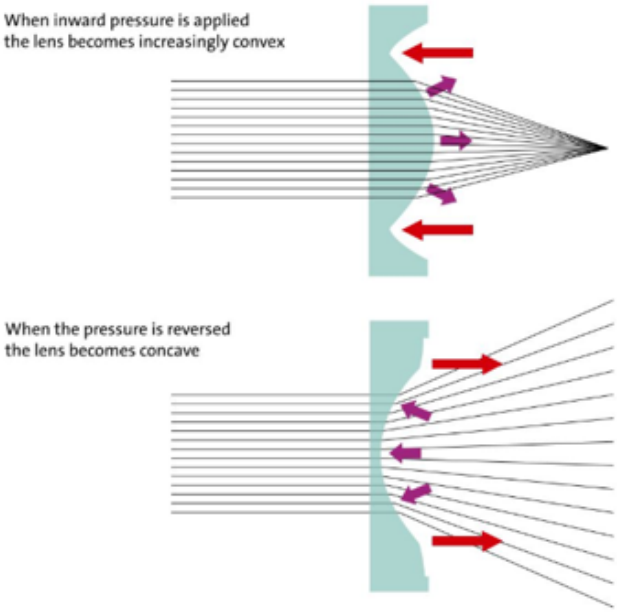


Figure 3.3.2: Liquid lens [6]

3.3.5 Lens groups

A camera lens exists of several lenses both before and after the aperture. These lenses are grouped by functional purpose of the lenses. These functions include light gathering, colour filtering (fluorecence) correcting for several image aberrations,and focusing the image (focus lens). Often lenses are annotated by the dispersion rate: Extra-low dispersion (ED), Low dispersion (LD), Special low dispersion (SLD), extraordinary low dispersion (ELD), and ultra-low dispersion (ULD). The key reason LD, ED, ELD, ULD and SLD glass are important in lens design is because of their ability to reduce levels of chromatic aberration.

3.4 Lens mounts

The function of the lens mount is to attach the lens to the camera. It has to be mechanically stable and defines the distance between the lens and the sensor. Some lens mounts are visible in Figure 3.4.1.

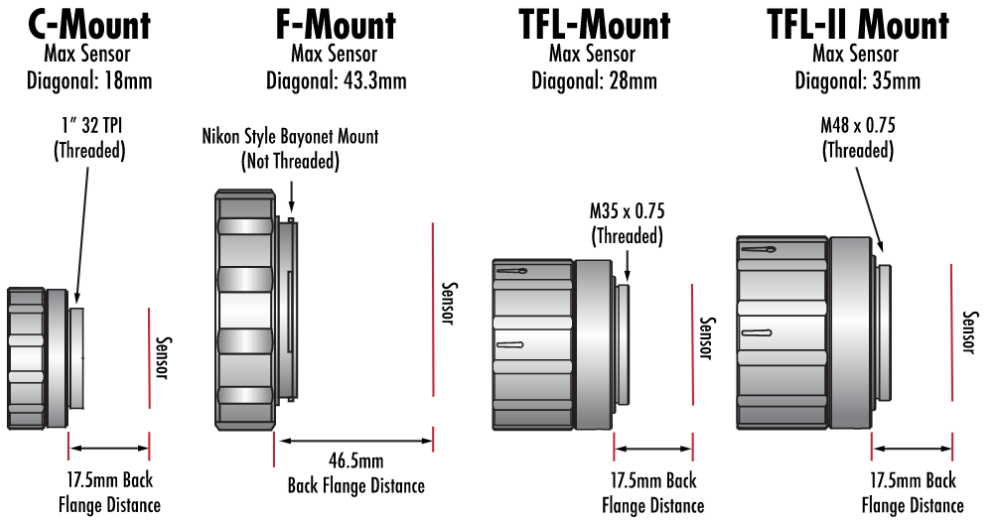


Figure 3.4.1: Lens mounts [8]

S-Mount The S-Mount is a common mount in surveillance CCTV cameras and board-level cameras and is used with M12 lenses. This smaller mount type has a metric thread and pitch of M12x0.5 and no standardised flange distance.

C-Mount The C-Mount standard is one of the most common lens mount types in machine vision. The C-Mount standard features a thread 1" in diameter with 32 threads per inch (TPI). The C-Mount flange distance is 17.526mm. It is ideal for many industrial applications, as the threaded mount provides a robust, controlled interface between the camera and lens.

CS-Mount The CS-Mount standard is the same as the C-Mount standard, but with a reduced flange distance of 12.526mm. A CS-Mount camera can be modified to accommodate C-Mount lenses with the use of a 5mm extension tube between the lens and camera. A CS-Mount lens may appear to be compatible with a C-mount camera because it will thread onto it. However, a CS-Mount lens cannot be used with a C-Mount camera because the lens will focus the image at a location inside the flange and in front of the sensor.

F-Mount The F-Mount standard is a bayonet-style mount common with line scan cameras and large format cameras. It was developed by Nikon and is used on 35mm (43.3mm) SLR photography cameras. The F-Mount standard features a diameter of 44mm and a flange distance of 46.5mm. The spring-loaded bayonet connection provides ease of use for photographers but can contribute to camera-lens misalignment in industrial applications.

TFL & TFL-II Mount The TFL-Mount was designed for use with APS-C (27.9mm) sensors that are much too large for use with a C-Mount, but are still too small for an F-Mount. This mount can be thought of as a larger C-Mount; it has thread dimensions of M35x0.75mm, and the same flange distance as the C-Mount: 17.526mm. The TFL-Mount provides the same robustness as the C-Mount but for larger sized sensors, without the drawbacks of the F-Mount. Similar to the TFL-Mount is the TFL-II Mount which was designed for APS-H (35mm) sensors and consists of the same 17.526mm flange distance but with M48x0.75 thread dimensions.

3.5 Lens filters

Several filters can be applied to a lens in order to change the behaviour of the light striking the lens. This can be done to protect the lens, to restrict certain wavelengths of light, or to polarise the light. It is always good practice to select the lens filter complementary to a light filter which will deliver the best results and is sometimes the only way to image a particular defect. Filters are defined by their spectral transmission, and most of the types used in the machine vision industry will have associated transmission graphs. The most common types were listed in Section 2.7.

3.6 Lens selection criteria

When selecting a lens, it is firstly important to consider the size of the object and thus the Field of View (FOV) that you want to image. Based on this, and together with the size of the sensor, the focal length or magnification of the lens can be determined. Once the required Field of View is known, one can look for the required resolution of the lens. The lens has to fit the resolution of the sensor and the requirements of the application.

- Sensor size - image circle diameter: The lens should illuminate the complete sensor in order to avoid shading.
- Pixel size - optical resolution: The resolution of the sensor and the resolution of the lens need to match. If the resolution of a lens is high, this means that small details of an object can be imaged properly on the sensor. But if the resolution of the sensor is worse than the resolution of the lens, then this correctly projected detail by the lens will not be captured by the sensor. In other words, the pixels on which the details are projected are larger than the detail itself.
- Object resolution - magnification: It is important that details of an object span enough pixels to be observed in the image correctly. A detail that is projected on more pixels allows for better analysis than a detail that is projected onto just one pixel. Thus the magnification of the lens should be selected that projects a detail at the desired number of pixels.

The most important parameters for the lens selection are the Field of View, sensor size, working distance and focal length of the optics. The first two values, however, are mostly defined by the application so that only the working distance for a particular focal length, or vice versa, must be calculated.

Calculate image size To calculate the required image size y' and thus the size of the sensor, we need the focal length of the lens f, the required Field of View y, and the working distance a. We can then calculate the required image size as:

$$y' = \frac{yf}{a - f}$$

Calculate Field of View To calculate the required Field of View y and thus the maximum size of the object, we need the focal length of the lens f, the working distance a, and the required image size y'. We can then calculate the required Field of View as:

$$y = y'(\frac{a}{f} - 1)$$

Calculate working distance To calculate the required working distance a and thus the distance of the object to the lens, we need the focal length of the lens f, the object size y, and the required image size y'. We can then calculate the required working distance as:

$$a = f(\frac{y}{y'} + 1)$$

Chapter 4

Sensors

When light passed through the lens and gets focused, it falls onto the camera sensor. This sensor converts the light particles, called photons, into electrical charge and finally to a digital signal. The sensor is a discretized receiver that consists of an array of photon capturing bins called pixels. In these pixels, the amount of captured light is converted to electrical charge. The type of sensor and the properties of the pixels will determine the properties and quality of the final image. It is therefore highly important to select the right sensor for every specific application. Figure 4.0.1 illustrates a basic concept. In the this chapter, the core components of the camera sensor are discussed.

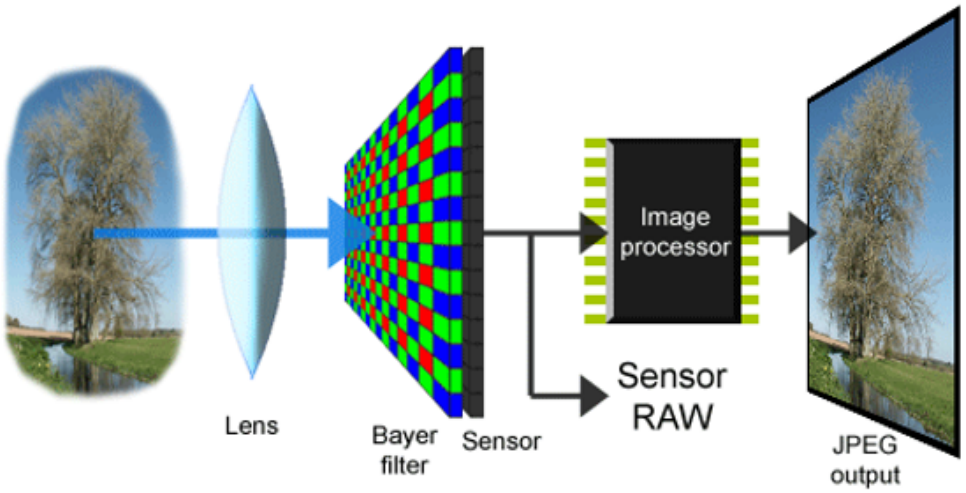


Figure 4.0.1: Object - lens - sensor setup

4.1 Sensor properties

4.1.1 Pixels

A pixel is a small light-sensitive receptor that covers the surface of the sensor. It catches photons and transforms their energy into voltage. A pixel acts as a kind of bin that catches 'balls' while opened. The size of a pixel can very dependent on the sensor used. Figure 4.1.1 shows some common pixel sizes.

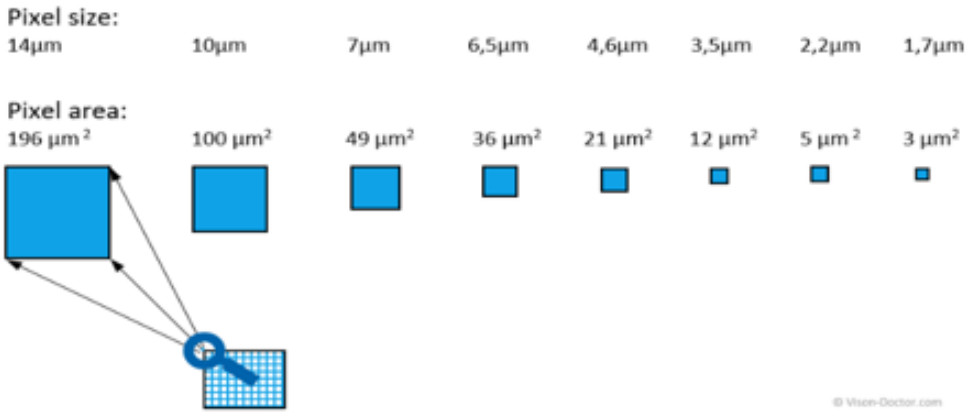


Figure 4.1.1: Pixel sizes [9]

The larger the bin or the pixel, the more light it can receive. The smaller the bin or the pixel, the more light is required to capture an image. Pixels are represented by their Quantum Efficiency (QE) which is the percentage of photons which are converted to electrons. There are two main types of sensor technology used in machine vision cameras, CCD and CMOS, which are described in the next sections.

4.1.1.1 CCD

Charge-couple-device (CCD) pixels convert photons to electrical charge inside the pixel and convert the charge to voltage outside the pixel by means of a shift register. The light rays coming from the lens are projected onto a capacitor array which accumulates electrical charge proportional to the light intensity. Outside the array there is one location where these charges can be converted to voltage as visible in Figure 4.1.2. First, the electrical charge of every capacitor is transferred via a vertical shift register to the neighbouring capacitors until it reaches the last pixel row of the sensor. Second, a transfer register transfers all these charges of the last pixel row to the location where charge can be converted to voltage using a charge amplifier. The speed by which a sensor can transport the charge by one pixel is called the 'pixel lock' and is denoted in Herz. The advantages of this sensor type is that there is a high usage of the light-receiving surface, also called the 'Fill factor' and that there are few defective pixels due to the simple structure. The pixels also have a high uniformity due to the charge-voltage conversion outside the pixel. The disadvantages are that these sensors require a special and expensive production process and that it is a rather slow sensor due to all the shifts.

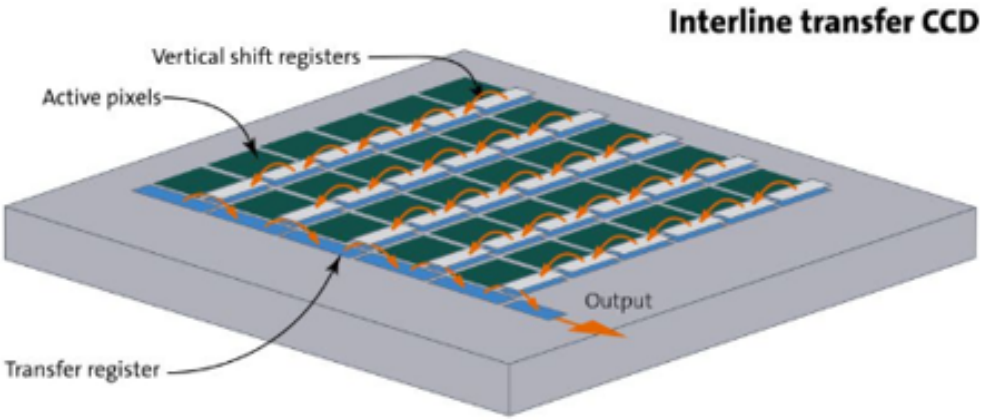


Figure 4.1.2: Charge-couple-device (CCD) [10]

4.1.1.2 CMOS

Complementary metal oxide semiconductor (CMOS) pixels convert photons to electrical charge inside the pixel and also convert the charge to voltage inside the pixel by means of a classical transistor (MOSFET). The light rays coming from the lens are projected onto a capacitor array which accumulates electrical charge proportional to the light intensity. But with CMOS sensors, the charge-to-voltage conversion is done inside the pixels as visible in Figure 4.1.3. Beside that, each pixel also has its own amplifier and noise-correction. The advantage of this sensor type is that the sensor requires a classic production process and is therefore rather cheap. The parallel conversion instead of the sequential conversion also allows for high frame rates. The disadvantages are that these pixels have a lower light-receiving surface (Fill factor) and that there is a lower uniformity due to the charge-to-voltage conversion inside the pixel.

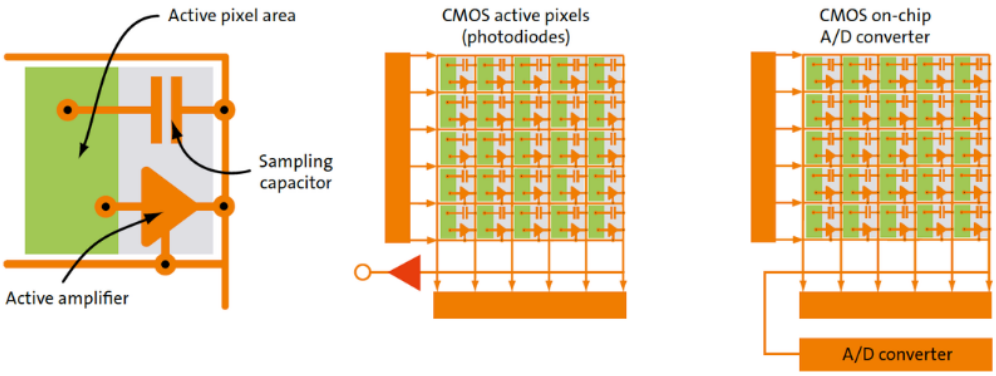


Figure 4.1.3: Complementary metal oxide semiconductor (CMOS) [10]

4.1.2 Dimensions

The dimension of an area scan camera sensor is typically denoted in inches, historically with an aspect ratio 4:3. Common sensor dimensions are denoted in Figure 4.1.4. The sizing convention is a legacy from the days of the early Vidicon tubes and these sizes do not correspond to an exact physical dimension of the sensor. The size of the sensor together with the grade of light concentration of the lens determine the size of the object that can be imaged as well as the resolution that could be obtained. There is always a trade-off between required Field of View and required resolution as a large FoV requires a large sensor and high focus lens which have as a result that tiny details on the object can be focused on an area that is the size of a pixel or smaller. When this is the case, it is very difficult to do any image analysis on the detail. In practice, it is required for a detail to be spanned over multiple pixels. This requires a lower focus of the lens but then also a smaller FoV that can be reached.

Type	Diagonal	W x H in mm
SFF*	43.3 mm	24 x 36
APS C	26.7 mm	22.2 x 14.8
4/3"	21.6 mm	17.3 x 13.0
µFT	21.6 mm	17.3 x 13.0
1.2"	21.4 mm	15.15 x 15.15
1.1"	17.6 mm	14.2 x 10.4
1"	15.8 mm	12.7 x 9.5
1/1.2"	13.3 mm	11.3 x 7.1
2/3"	11.0 mm	8.8 x 6.6
1/1.8"	8.9 mm	7.18 x 5.32
1/2"	8.0 mm	6.4 x 4.8
1/3"	6.0 mm	4.8 x 3.6
1/4"	4.6 mm	3.65 x 2.74

*small field format

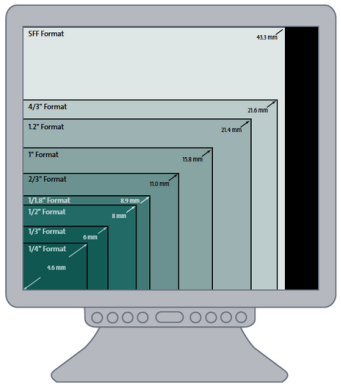


Figure 4.1.4: Sensor dimensions [10]

4.1.3 Bit depth

In a pixel, the charge captured by the pixel gets converted to voltage. In a next step, this analogue voltage gets digitised. This can be done using different discrete steps between no signal (black) and the maximum signal (white) as visible in Figure 4.1.5. Typically, an 8 bit conversion is used where 256 steps exist. A signal is then represented by values ranging from 0 (black) to 255 (white). Using the bit depth it is possible to compute the required transmission of the camera setup.

$$bandwith/s = bitdepth * imagesize * framerate$$

1 Bit (2 Tones)	2 Bit (4 Tones)	3 Bit (8 Tones)	4 Bit (16 Tones)
0	0	0	0
1	1	1	1
	2	2	2
	3	3	3
		4	4
		5	5
		6	6
		7	7
			8
			9
			10
			11
			12
			13
			14
			15

Figure 4.1.5: Bit depth [10]

4.1.4 Spatial resolution

The spatial resolution simply refers to the number of active pixels on the sensor. This can be calculated from the sensor size and the pixel size. The smallest feature in combination with the required Field of View defines the smallest possible spatial resolution. Several common spatial resolutions are visible in Figure 4.1.6.

Name	Resolution	Number of pixels
VGA	640 x 480	0,3 megapixels
SVGA	800 x600	0,48 megapixels
XGA	1024 x 768	0,78 megapixels
SXGA	1280 x 1024	1,3 megapixels
UXGA	1600 x1200	1,9 (2) megapixels
SUXGA	2048 x 1536	3,1 megapixels
-	2048 x 2048	4,0 megapixels
-	2452 x 2054	5,0 megapixels
QUXGA	3200 x 2400	7,7 megapixels
HD	1280 x 720	0,92 megapixels
Full HD	1920 x 1080	2,1 megapixels

Figure 4.1.6: Spatial resolution [10]

For example, consider a 1 x 1 mm feature that needs to be inspected on an object. The size of the object is 100 x 100 mm and the feature can occur anywhere on the object. For proper analysis of the feature, the software requires that the feature occupies 3 x 3 pixels. It will therefore be necessary to have 3 pixels per mm, multiplied by the object size, equals a spatial resolution of 300 x 300 pixels. This will hold true for monochrome cameras, however, special consideration needs to be made when using colour cameras that use a Bayer pattern. Here the colour of each pixel is interpolated using colour information from adjacent pixels. As a guide the minimum resolution should be doubled when using Bayer sensors.

4.1.5 Spectral response

When selecting a camera sensor, it is important to know in what spectral ranges this sensor is sensitive. The human eye is sensitive to wavelengths between 400nm and 700nm. But many camera sensors are able to detect light up to 1000nm. Some spectral responses of camera sensors are shown in Figure 4.1.7.

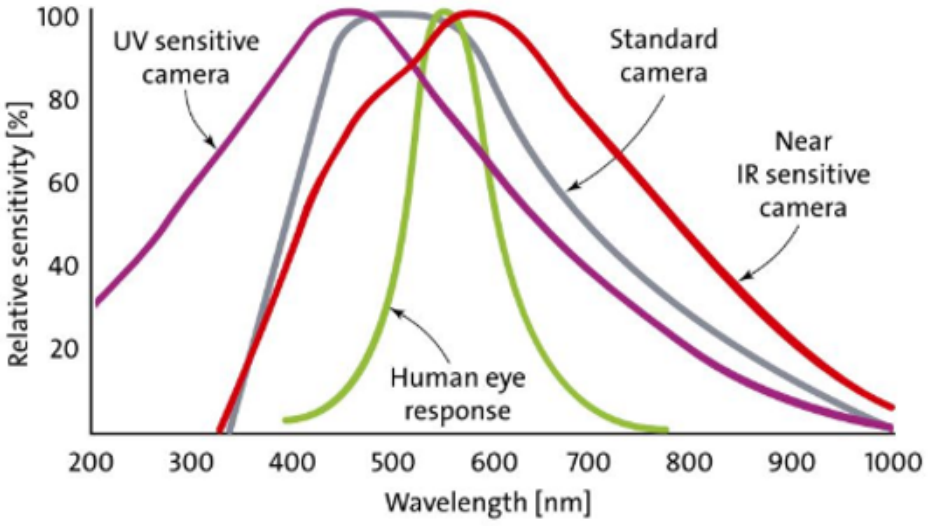


Figure 4.1.7: Spectral response [10]

4.2 Sensor features

4.2.1 Frame rate

The frame rate (fps = frames per second) denotes the number of frames that can be outputted by the camera in a particular time. A complete image from an area scan camera, or one that is built-up from the output of a line scan camera, is called a frame. CCD sensors have lower frame rates than CMOS sensors. To calculate the required frame rate, the frequency with which the object to be viewed is changing has to be considered

4.2.2 Exposure time

The exposure time or shutter time is the time that the sensor array is exposed by light before it converts the light to a digital image. This determines the clarity of the image. The bigger the exposure time, the longer the pixels are exposed to light but the more noise is being gathered on the sensor. When the pixels are illuminated too much, they can saturate, which results in an image where large areas or the whole area appears white (max signal value). Small pixels are saturated quicker than large pixels. Also when the exposure time is too large when capturing moving objects, more pixels can be exposed to light from the same point. This phenomenon is called motion blur as the object in the image seems to be blurred. Short exposure times are thus needed when imaging a fast moving scene.

4.2.3 Partial scan and ROI

It is possible to select a particular region of the sensor to be read out. A distinction can be made between partial scan and Region of Interest (ROI). In a partial scan, only a partial number of horizontal lines is read out. This can be done to increase the frame rate. The charges from all the pixels are moved into the vertical shift registers. The charge from the pixels that are not needed is dumped and excluded from the horizontal (readout) shift registers. Defining a specific Region of Interest (ROI) is the ability of the sensor to only concentrate on a specific area in the image (can be as small as one pixel). This reduces the image size and thus the bandwidth. Also multiple regions of interest are possible, this is called multiple region of interest (MROI). This is visible in Figure 4.2.1.

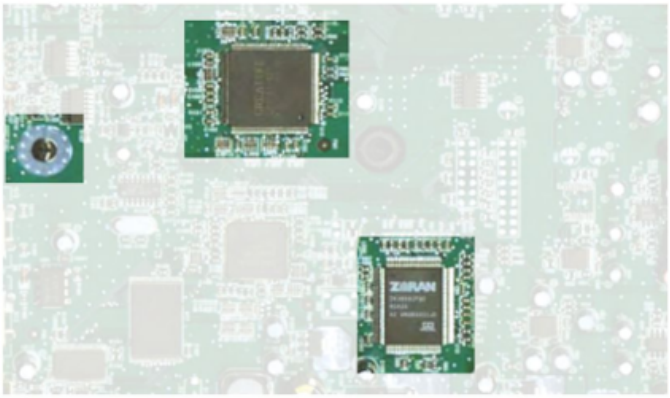


Figure 4.2.1: Multiple region of interest [10]

4.3 Colour cameras

In machine vision it is often required to have some colour information about the object. However, this is not necessary in most of the cases as information extraction algorithms are mostly focused on contrast. Including colour can even be a disadvantage as the spatial resolution can decrease while transmission bandwidth, CPU overhead, and cost can increase. Therefore, one should always be thoughtful when considering the use of colour cameras. Anyway, when there are applications that it is necessary to have colour information, there are two main camera types that exist to this purpose which are described next.

4.3.1 Single-chip colour cameras

In single-chip colour cameras, there is one CCD or CMOS sensor that is overlaid with coloured filters that cover each of the pixels. Each pixels has information of one colour. The most common filters for this are called Bayer filters that contain red, green, end blue filters arranged in a certain pattern as shown in Figure 4.3.1. The Bayer filter contains twice as many green pixels compared to red or blue which mimics the higher sensitivity of green to the human eye. Software is used to interpolate colours for every pixel which causes a loss of resolution. There are also other types of colour filter arrays such as the CYGM filters (Cyan, Yellow, Green, Magenta), also called complementary colour filters.

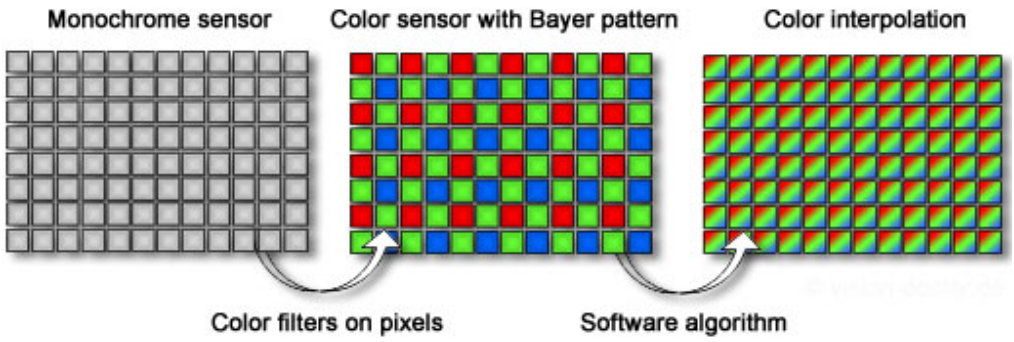


Figure 4.3.1: Single-chip colour camera with Bayer filter [9]

4.3.2 Multi-chip colour cameras

In multi-chip colour cameras, the incoming light is split into bands using prisms. A three-chip colour camera splits white light into red, blue, and green light as seen in Figure 4.3.2. For each of the bands there is a separate sensor. This type of camera is used for accurate colour measurements. The advantages are that the spatial resolution for each band is retained and that the colour fidelity and colour accuracy are improved. The disadvantage is that it is a rather large camera and that is is very expensive due to the special lenses needed.

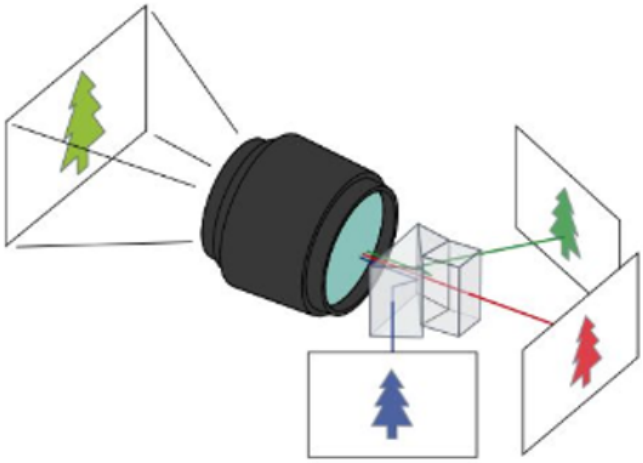


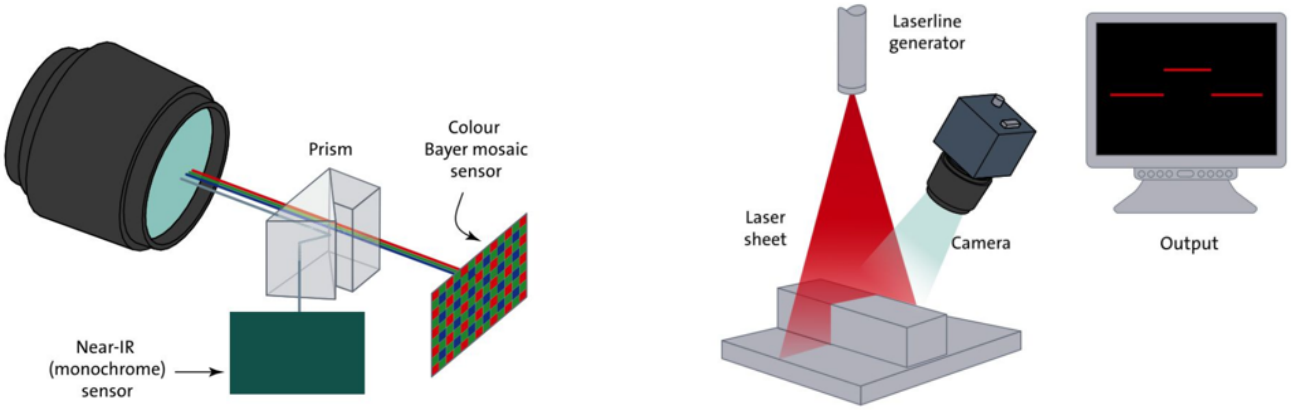
Figure 4.3.2: Three-chip colour camera [10]

4.4 Specialist cameras

Beside the monochrome and colour cameras there exists a large variety on cameras to suit the large variety on machine vision applications. Some of these specialist cameras are described in the next sections.

4.4.1 Dual sensor camera

Similar to the multi-chip colour camera, the dual sensor camera uses a prism to split the light according to the wavelength in two or more light beams to be captured on a different sensor. This technique enables simultaneous capturing and subsequent transmission over two channels. This is visible on Figure 4.4.1a. Not only light can be split but, for example, also the visible light spectrum can be split from the NIR-spectrum. The NIR-channel enables detection of features that are only visible in the long wave range, the near infrared, not visible to the human eye.



(a) Dual sensor camera [10]

(b) 3D laser camera [10]

Figure 4.4.1: Dual sensor and 3D laser camera

4.4.2 3D camera

Beside the standard cameras for capturing a 2D image, also 3D cameras exist that provide a 3D image of the world. Several techniques exist in order to obtain a 3D image of an object. Two of them are described below.

4.4.2.1 3D laser camera

This camera used structured laser light that is projected onto the object. A camera is then positioned towards this object at a certain angle with respect to the laser. Using triangulation, height information can be obtained from deformation of the line profile. This is visible in Figure 4.4.1b.

4.4.2.2 3D stereo camera

Another way of obtaining 3D information is using two cameras observing the same scene. Features in this scene are detected in both cameras and are matched. Using these matched features, the displacement between the two cameras can be calculated. When this displacement is known, depth information can be obtained using triangulation. This process is visible in Figure 4.4.2.

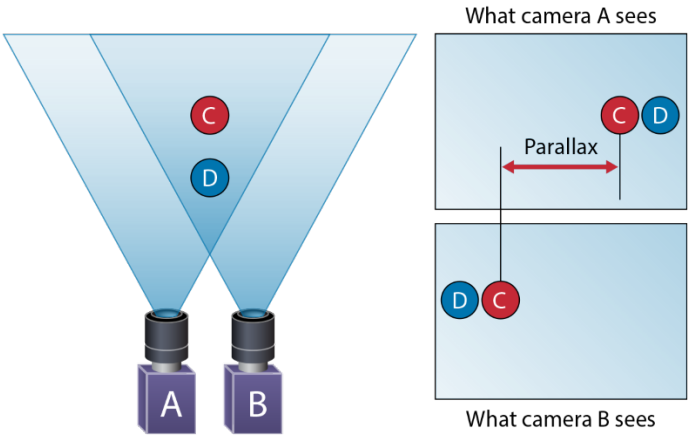


Figure 4.4.2: Stereo camera [11]

4.4.2.3 Time-of-flight camera

A Time-of-Flight (ToF) camera uses time-of-flight techniques to resolve the distance between the camera and the object for each pixel in the image. A laser LED at every pixel emits an artificial light signal. This signal reflects on an object and is captured by the corresponding pixel. By measuring the total time between sending and receiving, the distance is calculated. This process is visible in Figure 4.4.3.

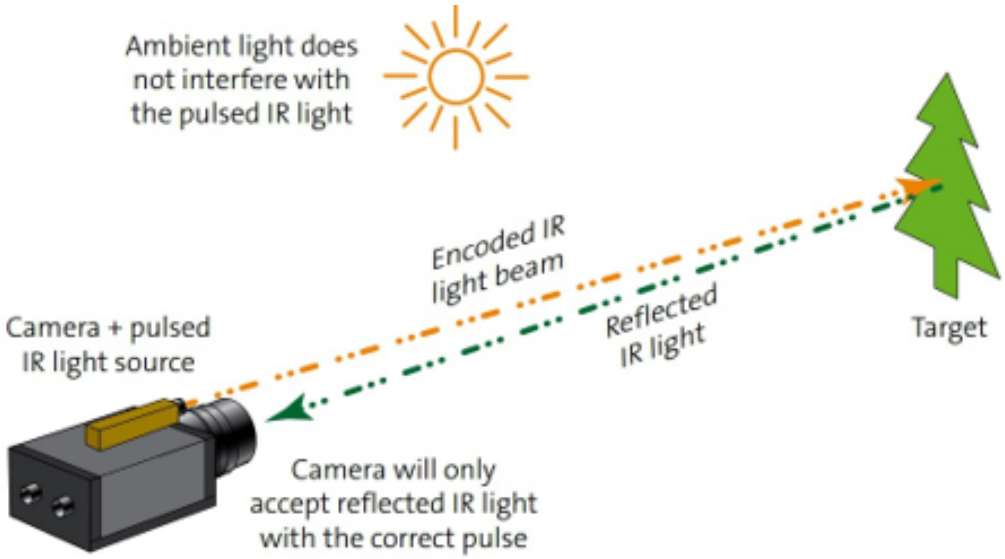


Figure 4.4.3: Time-of-Flight camera [10]

4.4.3 High-speed camera

High-speed cameras use high-end CMOS sensors and can capture more than 1000 frames per second at megapixel resolutions. Due to the large amount of data recorded and the limited bandwidth common to most digital interfaces, data is often recorded onto local memory while the images are transferred later at a lower rate (off-line). The fast capturing of the camera leaves little time for light collection, so high-powered lighting is necessary. High-speed cameras can be very expensive.

4.4.4 X-ray camera

X-ray cameras use and X-ray source and an X-ray detector with the object in between. X-rays move through the object and are attenuated. Higher density materials appear to be darker as they block more X-rays. A phosphor screen converts the X-rays to visible light. An example of an X-ray image is visible in Figure 4.4.4.

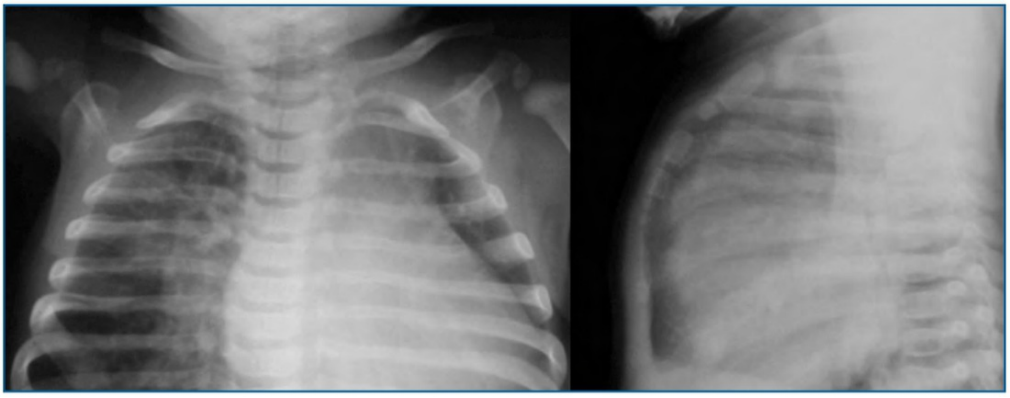
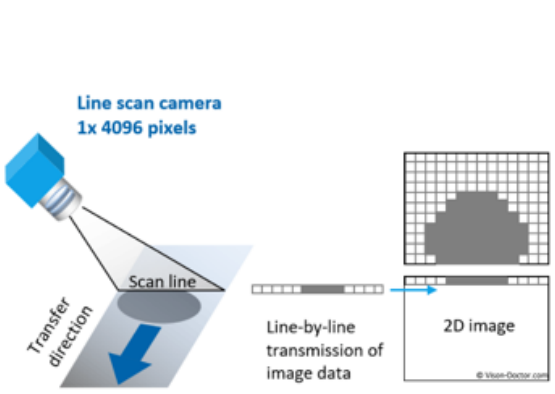


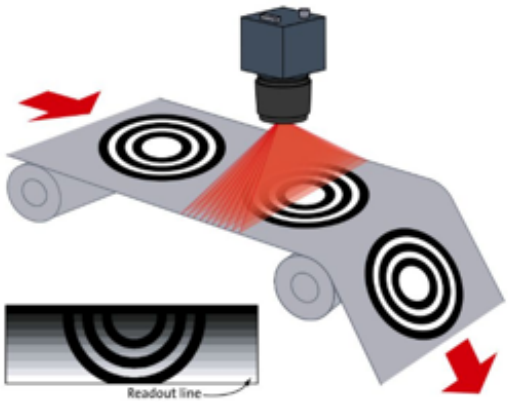
Figure 4.4.4: X-ray image [10]

4.4.5 Line scan camera

The most common cameras are area scan cameras that image a rectangular area using a rectangular sensor. However, for some applications it is more suitable to have a camera that only images a single line of pixels instead of an area. This is especially interesting for continuous applications. Line scan cameras build up an image line by line using a line sensor that passes in a linear motion over an object, or the object passes over the sensor. This makes it possible to capture images of wide objects at a single pass but requires a very high degree of precision. The alignment and timing of the systems are critical to the resulting image fidelity and geometry using incremental encoders. Line scan systems are therefore more complex to implement than area scan systems. Line scan cameras also require a greater level of illumination since a quick camera or object movement require shorter exposure times. Some applications of line scan cameras are inspection of a continuous product, imaging objects of different sizes at a conveyor belt, imaging cylindrical components. A line scan system is visible in Figure 4.4.5a.



(a) Line scan camera [10]



(b) TDI line scan camera [10]

Figure 4.4.5: Line scan and TDI line scan camera

4.4.5.1 TDI line scan cameras

Line scan cameras have the problem that they require a high level of illumination as the speed of the object passing the scanner is rather high and the pixels are rather small. Therefore, a technique exist to this purpose where multiple imaged lines of the object are delayed and integrated. cameras that do this are called Time Delay and Integration (TDI) line scan cameras. Line information is copied line by line synchronously with the object movement and exposed with the same image information. All lines are accumulated afterwards to sum up the relative low light intensities in every pixel. The advantages is that high speeds can be reached with low illumination. Also reflections get averaged as each line looks at a different angle. The disadvantages are sensitivity to misalignment and sensitivity to speed mismatching. A visualisation of a TDI line scan is shown in Figure 4.4.5b.

4.4.5.2 Colour line scan cameras

Also colour line scan cameras exist where the pixels are also overlaid with colour filters. Some variants exist such as single line with RGB triples, dual line, trilinear line, and 3CCD lines with prisms. Each of the types have an analogy with the colour cameras described earlier. All the types are shown in Figure 4.4.6.

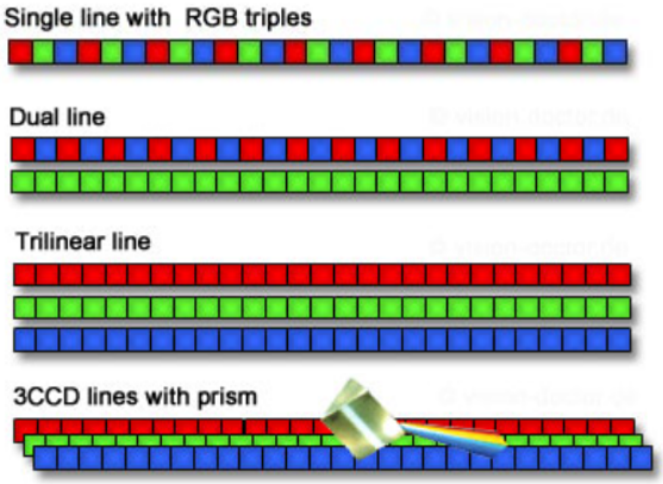


Figure 4.4.6: Colour line scan camera [10]

4.4.6 Multi-spectral cameras

In the standard colour cameras, the incoming light is separated (physically or via interpolation) in three spectral bands: red, green, and blue. This means that data coming from a light source and being distributed over a wide spectrum of wavelengths is compressed in only three spectral channels. This principle highly decreases the amount of information that can be obtained from an image. To mitigate, multi-spectral cameras record the light in more than three bands so something more could be said about the spectral information that is reflected by the object. Multi-spectral cameras generally use three to ten bands with a certain width in which light is recorded separately. A multi-spectral camera with five bands could for example record the amount of red, green, blue, near-infrared, and shortwave infrared that falls onto the sensor as visible in Figure 4.4.7.

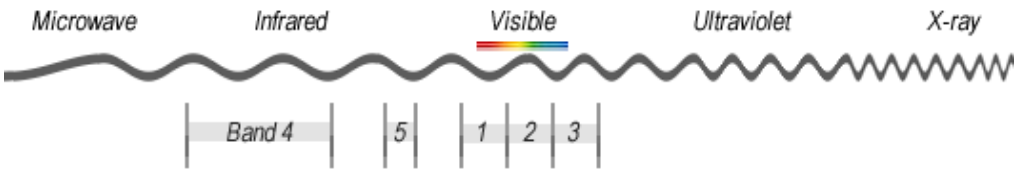


Figure 4.4.7: Multi-spectral camera with five bands [12]

An extension of multi-spectral imaging is hyper-spectral imaging which consists of hundreds much narrower bands from 10 to 20 nm. A hyper-spectral image could have hundreds or thousands of bands as visible in Figure 4.4.8.

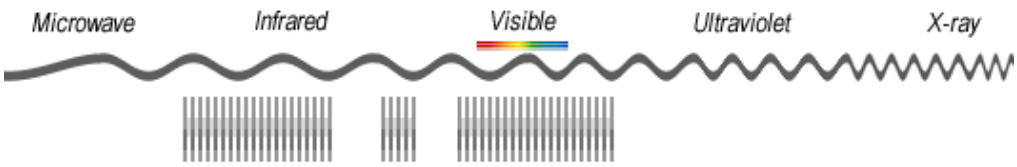


Figure 4.4.8: Hyper-spectral camera [12]

4.4.7 Extended wavelength response

Standard machine vision cameras are limited to recording visible light or near infrared light. This limitation is caused by the silicon which is used in most of the CCD and CMOS sensors. However, light outside this range can also be interesting to record as it can lead to some interesting applications. Figure 4.4.9 shows an overview of the other spectra and their corresponding required sensor material.

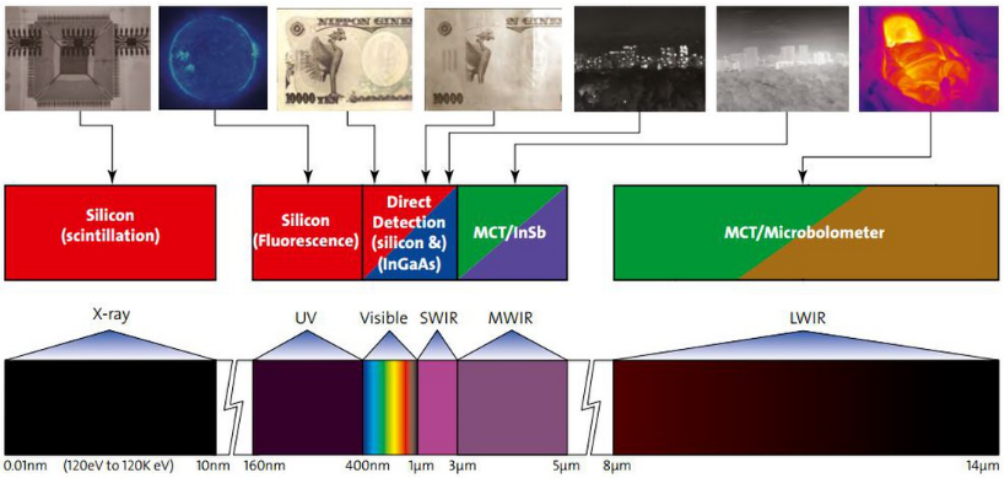


Figure 4.4.9: Extended wavelengths and used sensor materials [10]

4.4.7.1 Imaging UV light

UV light has a shorter wavelength than visible light and has a spectral range of 160 nm to 400 nm. A method to extend a sensor's response to these shorter wavelengths is to use a fluorescent coating on the lens which fluoresces when struck by a UV photon. This fluorescence is then detected by the standard sensor as it produces wavelengths in the visible light.

4.4.7.2 Imaging IR light

Infrared light has a longer wavelength than visible light and has a spectral range of 780 nm to 1 mm. This wide spectral range is further divided into:

NIR (750 nm - 1400 nm) For near infrared applications, the standard silicon based CCD and CMOS sensors are used where possible due to their lower cost compared with other sensor materials. As silicon based sensors only show a limited efficiency in converting photons into electrons in the wavelength greater than 1000 nm. However, manufacturers try to use special circuits or sensor coatings to get the remaining sensitivity beyond 1000 nm.

SWIR (1.4 µm - 3 µm) Short wavelength infrared is a very interesting spectral range. Images created by SWIR cameras are almost similar to those taken by a monochrome CCD camera. Infrared imaging can be used to detect features that are not apparent in visible wavelengths for applications such as inspection of silicon wafers or solar cells, sorting of fruits or vegetables, control of plant growth, recognition of safety features, etc. The highly sensitive NIR-SWIR cameras mainly use InGaAs (Indium Gallium Arsenide) detectors, offering a high quantum efficiency at 900 - 1700 nm and are normally combined with CMOS readout electronics.

MWIR (3 µm - 8 µm) Cameras for midwave infrared are often called thermal cameras and are most commonly based on Mercury Cadmium Telluride (MCT) or Indium Antimonide (InSb) sensors. Again using the photoelectric effect, MCT based sensors offer sensitivity at around 3 - 8 µm, where InGaAs is insensitive. As with shortwave infrared, midwave infrared imaging can be used for detecting reflected or emitted infrared. This sensor type requires cooling to ensure that the detected signal is not saturated by inherent dark current in the camera.

LWIR (8 µm - 15 µm) Longwave infrared detectors (or micro bolometers) detect heat radiation by measuring changes in capacitance or resistance within the structure of the pixels. Commonly based on Amorphous Silicon (ASi) or Vanadium Oxide (VO), these sensors can detect wavelengths of 8 to 14 µm where the measured values relate to the temperature of an object. This enables LWIR cameras to work where there is no infrared source and with objects at much lower temperatures compared to short and midwave infrared. As this technology is not dependent on an external IR source for imaging, it can be used for security and surveillance applications in all lighting conditions. For LWIR, an alternative lens type is required with sapphire, crystalline silicon and germanium being the most common materials used.

4.5 Camera interfaces

Beside the choice of the camera sensor, the choice of the camera data interface is important when designing a machine vision setup. Higher resolutions of CMOS sensors and increasing frame rates require large amounts of data to be transferred between the camera and processor. In this selection process, the needed bandwidth and the image data transmission distance are critical requirements. To mitigate the high requirements, some data transfer hardware interfaces exist including CameraLink, Cameralink HS, GigE Vision, USB3 Vision, CoaXPress. All of them are summarised together with their core properties in the table on Figure 4.5.1. Figure 4.5.2 shows a visualisation on a graph.

Interface	Image Data Throughput	Cable length (copper)	Frame Grabber
GigE Vision	115 MB/s	100 m	No
CameraLink	Up to 850 MB/s	4m-10m*	Yes
USB3 Vision	3-400 MB/s	3-5 m	No
CameraLink HS	2100/3300 MB/s	15 m	Yes
CXP-6	625 MB/s	40 m	Yes
CXP-6 x4	2500 MB/s	40 m	Yes
CXP-12 x4	7200 MB/s	25 m	Yes

Figure 4.5.1: Data transfer hardware interfaces [10]

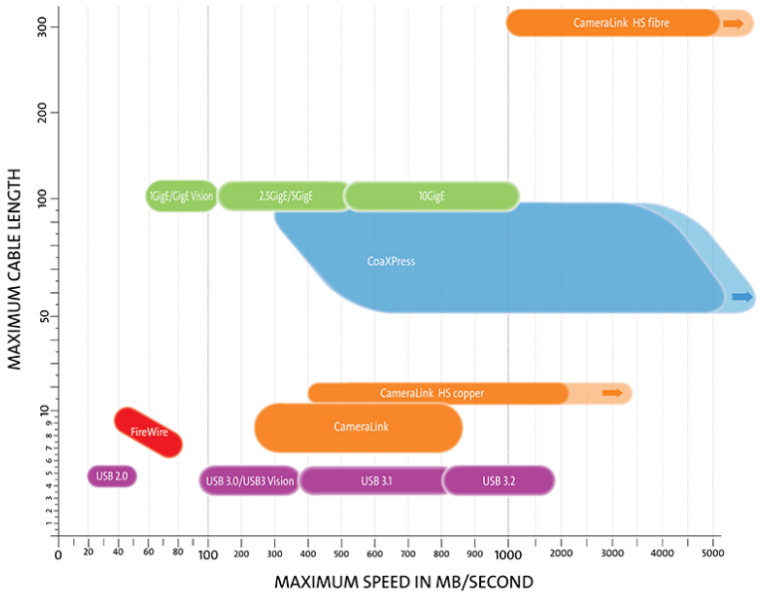


Figure 4.5.2: Data transfer hardware interfaces graph [10]

For relative high bandwidths and no need for long cable lengths, the USB3 standard is the most common and most simple solution for data transfer. When higher cable lengths are required, GigE Vision is more appropriate. However, while GigE Vision offers undoubted flexibility, the maximum bandwidth of 115 MB/s has been a limiting factor for some applications. For those that need higher bandwidths and also need long data transmission distances, the CoaXPress standard offers considerably greater bandwidth, while utilising coaxial cable which is used extensively in industrial, medical and defence applications.

Chapter 5

Image processing

The previous sections elaborated on the core aspects that generate an image and thus define the appearance of an image. This generated raw image is the input to a computer that has the task to extract useful information from it. As mentioned in the introduction, computers see an image as a matrix of numbers. And in fact, every image can be defined as a function I with variables v, which is the horizontal index of the pixel, and u, which is the vertical index of the pixel, both starting from the upper left corner of the image as seen in Figure 5.0.1. The evaluation of the function at a specific coordinate equals the pixel value:

$$pixel\_value = I(u, v)$$

Figure 5.0.2 shows an image that has only one channel which is also called a grey scale image. However, when looking at colour images, there are three channels each representing the blue, green, and red colour intensity present in the image. This is visible in Figure 5.0.3. So for colour images, the evaluation of the function at a specific coordinate equals a list of three values, red , green, and blue:

$$[red, green, blue] = I(u, v)$$

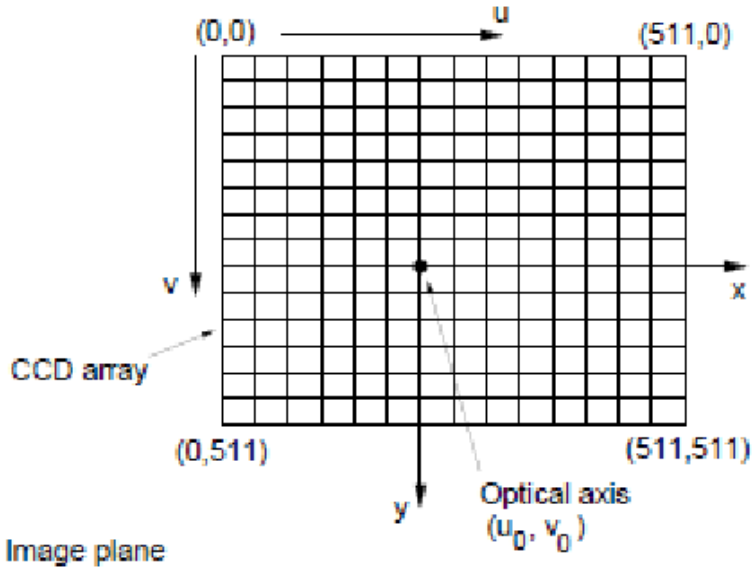


Figure 5.0.1: Pixel coordinates

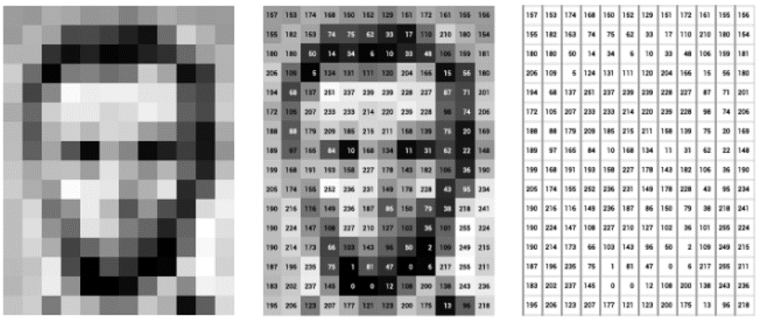


Figure 5.0.2: A grey scale image consisting of one channel

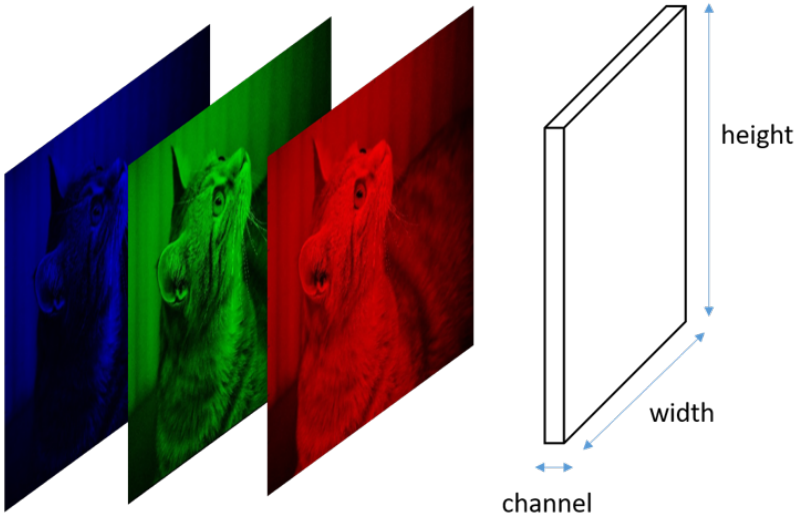
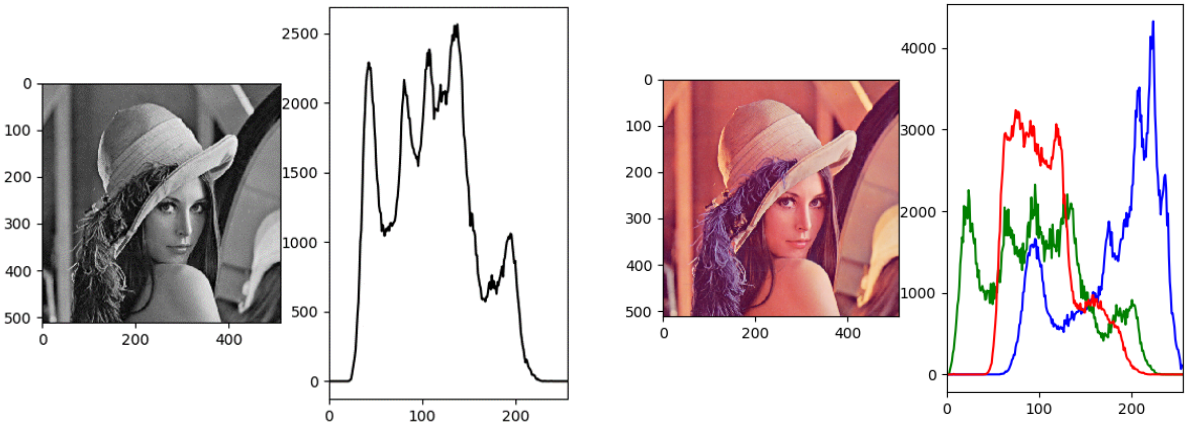


Figure 5.0.3: A colour image consisting of three channels [1]

As mentioned before, as a human, it seems to be impossible to extract information from a raw matrix of numbers. However, as a first image analysis step, some techniques exist to visualise this matrix in a different way which allows us to better understand what's in the image. One interesting way to visualise an image is via its histogram. A histogram of an image keeps track of the number of times a pixel value occurs in the image and displays the number of values of pixels from 0 to 255 on an y-axis of a histogram plot as visible in Figure 5.0.4 for a grey scale and a colour image. Using this histogram some insights could be derived in the distribution of pixel values, saturation, or the contrast in the image.



(a) A histogram of a grey scale image [1]

(b) A histogram of a colour image [1]

Figure 5.0.4: Image histograms

Another way to visualise an image is by plotting it in 3D as visible in Figure 5.0.5. In this way, some insights can be obtained in the derivatives of the image and thus the presence of noise, edges, corners, etc.

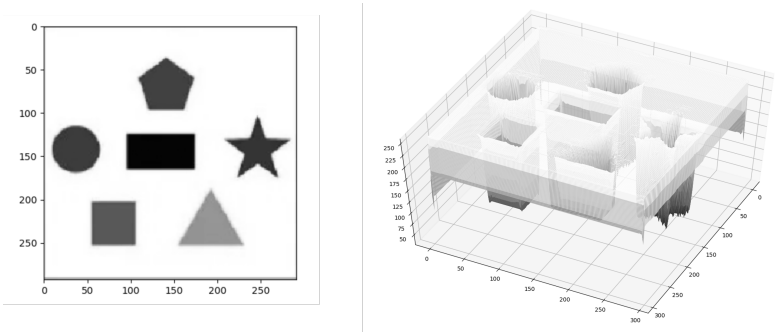
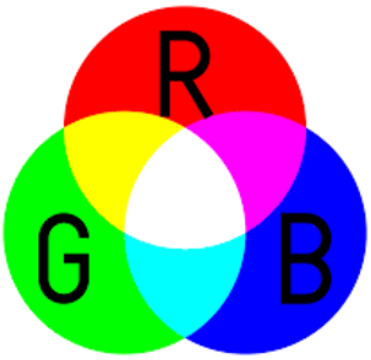


Figure 5.0.5: A 3D plot of an image [1]

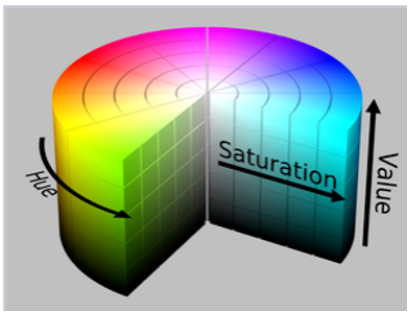
These two basic techniques provide us with some more information of how the matrix looks like. However, more techniques exist to modify an image in order to highlight the features that a computer wants to extract from it. These techniques are called image processing techniques. Image processing is a computational process that transforms an image in order to highlight useful features. This process has the aim to retrieve useful information from images that is otherwise not recognisable. For human purposes, this means enhancing the image or increasing contrast. When looking at the importance of image processing in machine vision, it is usually one of the first steps for feature detection. The following sections will provide an overview of the most commonly used image processing techniques.

5.1 Colour spaces

As seen earlier, a colour image consists of a matrix with three channels R, G, and B. For each pixel, the combination of the corresponding R, G, and B-values results in a specific colour. However, the RGB-representation, shown in Figure 5.1.1a, is not the only colour representation. Other so called 'colour spaces' exist that can represent colours in different ways. A colour space is a 3-dimensional space that contains all possible tristimulus values (all colours and all levels of brightness). The main colour space is RGB, which mimics the tristimulus as humans do using their cone cells as mentioned in Chapter 1. The whole tristimulus space is formed using a linear combination of the three primaries. Figure 5.1.2 shows a picture of a scene and the three R, G, and B-channels it consists of.



(a) RGB colour space



(b) HSV colour space

Figure 5.1.1: Colour spaces

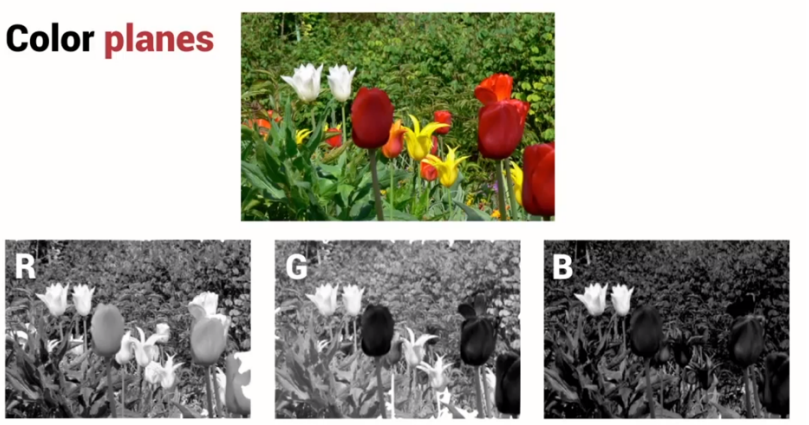


Figure 5.1.2: RGB-channels of a picture [1]

An extension to the RGB representation is the RGBA representation where the A stands for transparency. But other colour spaces exist. Another very common used colour space in machine vision is the HSV-colour space, as seen in Figure 5.1.1b, which is a more intuitive description of a colour in terms of hue, saturation, and value. The hue is the dominant colour or the closest spectral colour. Saturation is the purity or the absence of mixed white. Value is the intensity of the colour or the absence of black. The Hue is represented as an angle in the range of 0 (red) and 360 (violet). Despite the fact that the RGB- colour space mimics the way humans see colours, the HSV space provides us with a more intuitive way of defining colours as this representation only uses one channel to represent colour, which is the Hue channel. If we want to detect a brown object in an image, we need to know what R, G, and B combination represents this colour, which is rather difficult. In HSV context, brown is defined as one particular value in the one dimension Hue value. This makes it much easier for thresholding as we only need to specify a range in one dimension instead of in three dimensions in the RGB-space. Another advantage of the HSV-space is that the Hue channel only contains information about the pure colour, regardless of the lighting conditions which are captured in the other channels, saturation and value. Converting an image to the HSV-space could therefore increase the robustness of machine vision systems in changing light conditions. This is shown in Figure 5.1.3. For a blue object, the pixel values in RGB (left, channels G and B) and HSV-space (right, channels H and S) are plotted for different lighting conditions. In the left image, one can see that the variation of both the G- and the B-channel are high for the RGB-picture. For the HSV-picture, only the Saturation channel shows high variation because this captures the light intensity changes. But the variation in the Hue channel is very low, meaning that this colour space is able to define the object as being blue regardless of the lighting conditions. Beside the RGB and HSV colour spaces, other colour spaces such as HSL and CMYK (used in printing) exist.

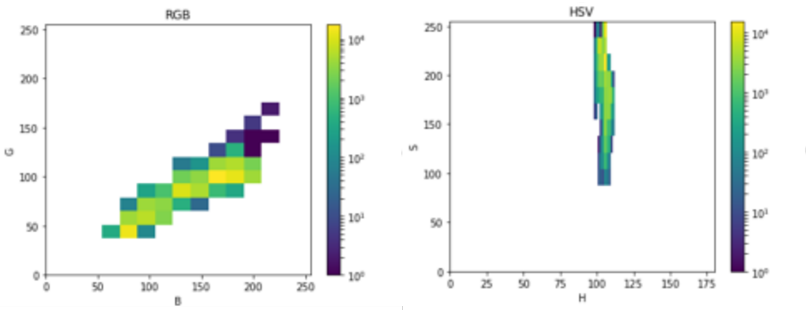


Figure 5.1.3: HSV versus RGB in changing lighting conditions [1]

5.2 Monadic operations

A monadic operation has one image with dimensions W x H as input and outputs an enhanced image with the same dimensions. Every pixel of the input image is transformed by applying the same function.

$$\forall\ u, v \in \mathbf{I} : \mathbf{O}[u, v] = f(\mathbf{I}[u, v])$$

Here, the u value is the column and the v value is the row of the pixel. A schematic representation of a monadic operation is shown in Figure 5.2.1.

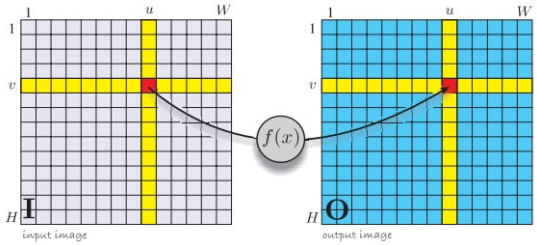


Figure 5.2.1: Monadic image processing algorithm [1]

The function f can be many things, from data type transformation to a simple linear operator, such as summation or multiplication. Some examples are provided:

- **Changing the pixel data type:** For example converting the uint8 format (values ranging from 0 to 255) to double-precision format (values ranging from 0 to 1). Changing the data type is relevant, because some functions only accept double-precision input arguments.
- **Converting a colour image to grey scale:** The following formula shows how the RGB values can be combined to get the corresponding grey scale value.

$$Grey = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B$$

- **Adjusting the contrast:** The contrast can be adjusted by multiplying the pixel value with a factor α. Increasing the contrast means increasing the difference between the darkest and lightest colours of a picture. This effect can be seen on Figure **??**.

$$\mathbf{O}[u, v] = \alpha \times \mathbf{I}[u, v]$$

- **Adjusting the brightness:** The brightness can be adjusted by summing the pixel value and a factor β. Increasing the brightness means increasing the overall lightness of the image, i.e. making dark colours lighter and making light colours whiter. This effect can be seen on figure 5.2.2.

$$\mathbf{O}[u, v] = \mathbf{I}[u, v] + \beta$$

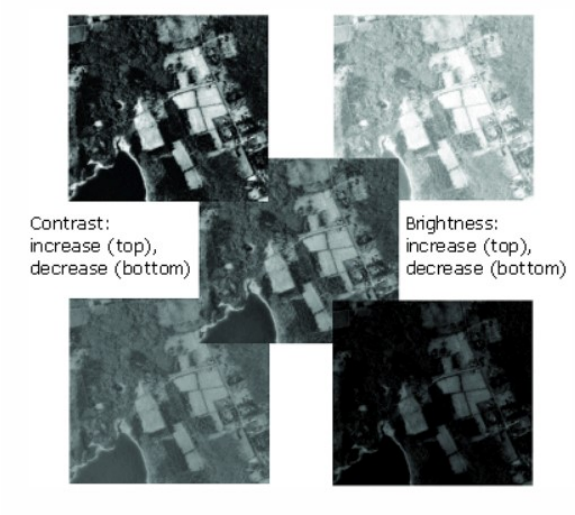


Figure 5.2.2: Adjusting contrast and brightness on an image [13]

5.3 Diadic operations

A diadic operation has two images with dimensions W x H as inputs and outputs an enhanced image with the same dimensions. Each output pixel is equal to a function of the corresponding pixels of the two input images. The same function is applied to obtain all the output pixels.

$$\forall\ u, v \in \mathbf{I_1}, \mathbf{I_2} : \mathbf{O}[u, v] = f(\mathbf{I_1}[u, v], \mathbf{I_2}[u, v])$$

A schematic representation of a diadic operation is shown in Figure 5.3.1.

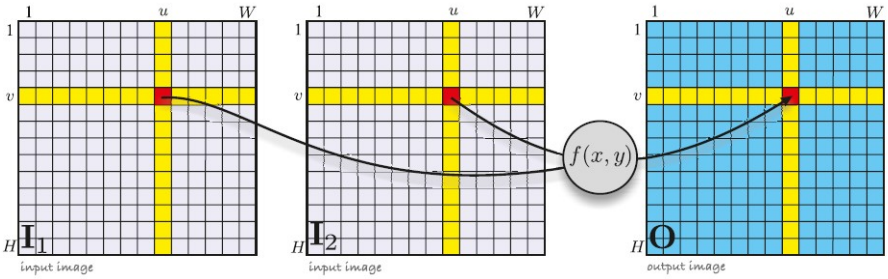


Figure 5.3.1: Diadic image processing algorithm [1]

The function f can be any binary operator, such as addition, subtraction or element-wise multiplication. Two examples, which illustrate applications of diadic operations, are provided:

- **Chroma-keying**: This application is commonly used in movies and is better known as "green screen". The goal is to impose an object on another background. Therefore two images are needed: an image of the object on a green background and an image of the desired background. First, the chromacity is calculated of the object image. The resulting histogram, depicted in Figure 5.3.2, shows two peaks. The higher peak refers to the green background and the lower one refers to the object. Next, a threshold is determined to create a binary

mask. This threshold is situated between the two peaks. Every chromacity value lower than the threshold is true and the pixel is coloured white (value = 1). In the other case, the pixel is coloured black (value = 0).

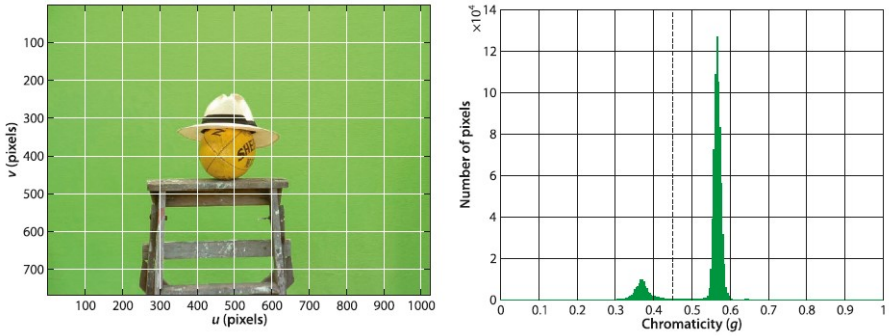


Figure 5.3.2: Chroma-keying: chromacity histogram of the object image [1]

The binary mask is then used in combination with the other two images to perform some diadic operations. These operations are schematically presented in Figure 5.3.3. First, a binary mask to detect the object is element-wise multiplied with the object image. The result is an image of the object with a black background. Then, a binary mask to detect the background is element-wise multiplied with the background image. The result is an image of the background but with the shape of the object in black. Finally, those two results are added. The obtained image presents the object imposed on the desired background.

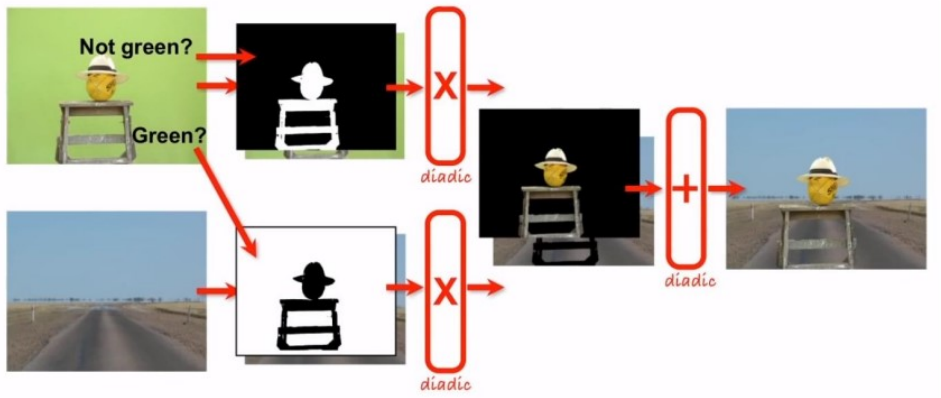


Figure 5.3.3: Chroma-keying: overview of the diadic operations [14]

- **Motion detection:** Sometimes it isn't possible to have a clear background like in the chroma-key example. A method to detect the background from the foreground is by doing motion detection. A sequence of images of a street is used in this example. In each step an image and a background estimate are used. This background estimate shows the static elements of the image because the moving elements are blurred. Subtracting the scene image from the background estimate creates an image where the pixels are bright when it differs from the background. All of the discussed images are shown in Figure 5.3.4.

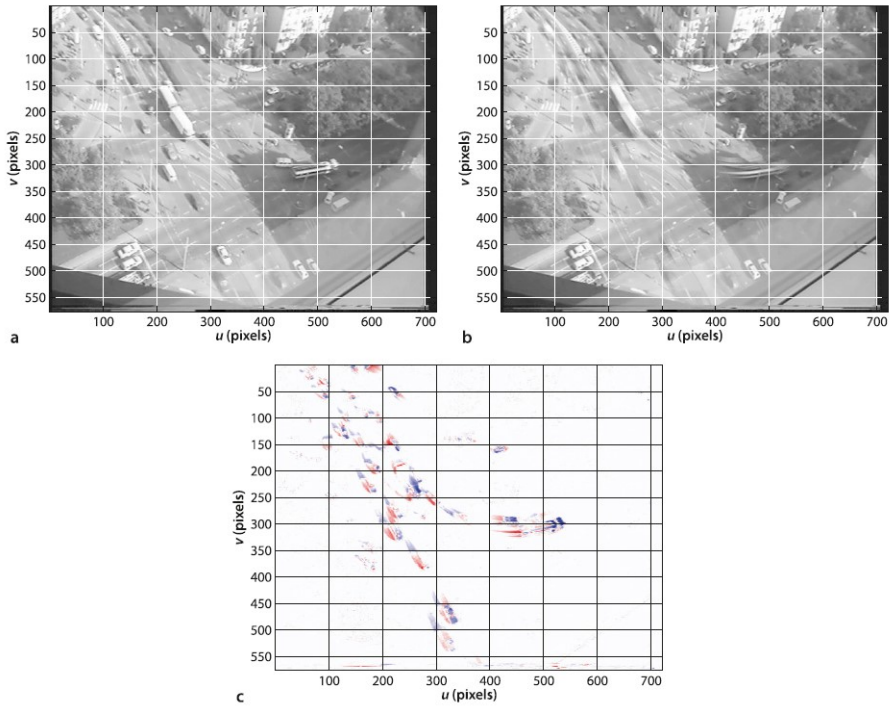


Figure 5.3.4: Motion detection: a) scene image, b) background estimate, c) result of the subtraction (here the value of zero corresponds to the colour white) [1]

5.4 Spatial operations

A spatial operation is similar to a monadic operation in that they both have one W x H image as input and an output image with the same dimensions. The difference is in the function that is applied and the input pixels to that function. In this case, an output pixel is equal to a function of pixels in a region surrounding the corresponding pixel.

$$\forall\ u, v \in \mathbf{I}, \forall\ i, j \in \mathcal{W} : \mathbf{O}[u, v] = f(\mathbf{I}[u + i, v + j])$$

$\mathcal{W}$ is a window, which is usually a $w \times w$ square. Other synonyms for window are region and kernel. A schematic representation of a spatial operation is shown in Figure 5.4.1. When the window is moving over the pixels situated close to the borders, some overlapping can occur with the area outside of the frame, this is called the boundary effect. There are two possible solutions for this. The first is to suppose that every pixel of the window, which is not covered by the image, has a value of 0. The second is to only let the window move over pixels that ensure that window is covered by the frame at all times. This has as a result that the image gets shrinked. Spatial operations are frequently used in image processing because they provide powerful operations. The spatial functions f can be divided into two groups: the linear spatial operators and the nonlinear spatial operators. The remainder of this section will give some different examples within both groups.

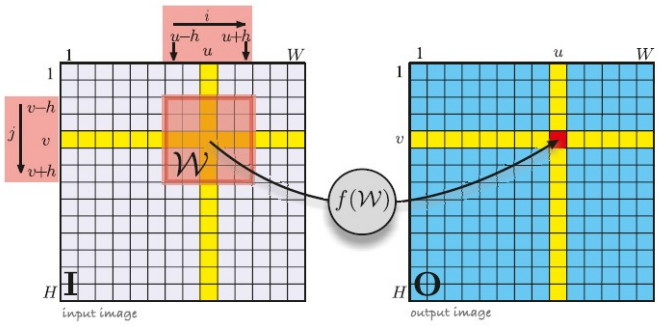


Figure 5.4.1: Spatial image processing algorithm [1]

5.4.1 Linear spatial filtering

One of the most frequently used linear spatial operator in image processing is convolution. Convolution exists of multiplying each filter value of the kernel K with each corresponding pixel value and summing them. This is basically performing a weighted sum [1].

$$\forall\ u, v \in \mathbf{I} : \mathbf{O}[u, v] = \sum_{i,j \in \mathcal{W}} \mathbf{I}[u - i, v - j]\mathbf{K}[i, j]$$

Here is $K \in \mathbb{R}^{w \times w}$ the kernel. An example of how a convolution can be carried out, is provided in Figure 5.4.2. The convolutional filter (or kernel), indicated in red, starts in the upper left corner of the input image and traverses the whole image. The resulting value in the output image is the weighted sum of the input pixels covered by the kernel.

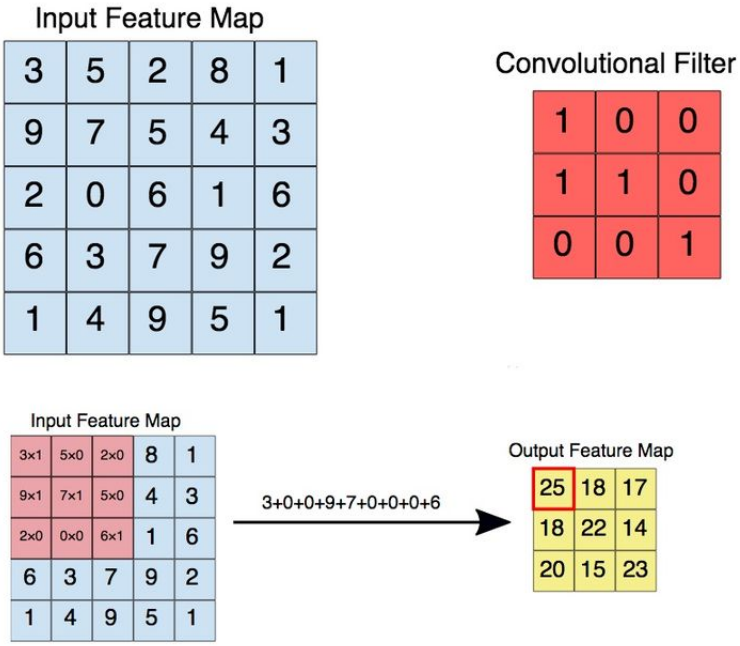


Figure 5.4.2: Convolution example [15]

Executing a convolution is a computationally expensive task. An $N \times N$ input image and a $w \times w$ kernel results in w^2N^2 multiplications and additions. If the image contains multiple channels, then the convolution will be carried out on each image from each channel. The difference between the linear spatial filtering operations lies in the difference in filter values. Two concrete examples are image smoothing and edge detection.

5.4.1.1 Image smoothing

Image smoothing is a process mainly used to remove noise, but also to soften edges and corners in the image. The two kernels, that can be used to perform smoothing, are the mean filter and the Gaussian filter. Both are applied to the Mona Lisa image in Figure 5.4.3. Smoothing with a Gaussian filter is preferred over smoothing with a mean filter because it is more effective for removing Gaussian noise.

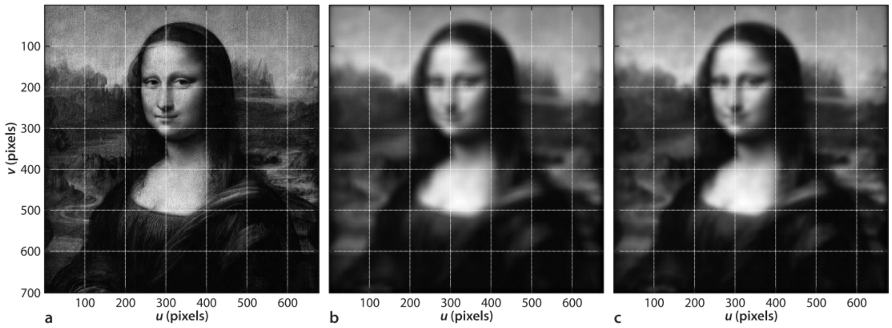


Figure 5.4.3: Image smoothing: a) original image, image smoothed with b) a 21×21 mean filter and c) a 31×31 Gaussian filter with $\sigma = 5$ [1]

- **A mean filter** is a $w \times w$ filter where each filter value is equal to $1/w^2$. Convolution with a mean filter calculates the mean of each region in the image and assign this value to the centred pixel. To give an example of how a mean filter looks like, a 7×7 mean filter is provided below:

1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49

Figure 5.4.4: A 7×7 mean filter [16]

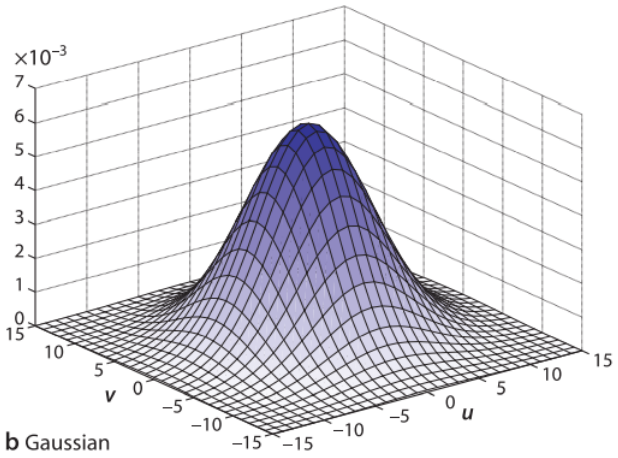


Figure 5.4.5: A 3D Gaussian [1]

- **A Gaussian filter** is a $w \times w$ filter where each filter value is equal the corresponding value of an approximated Gaussian. The formula of a Gaussian is:

$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

The σ value is the standard deviation expressed in pixels. The Gaussian filter can be interpreted as seeing the 3D Gaussian in top view. The centred pixel will have the highest value and all the other values are symmetrical in relation to that centre. As seen in Figure 5.4.6, the sum of the 7×7 kernel values is equal to 1. This is also the volume underneath a Gaussian surface.

0.00	0.00	0.01	0.01	0.01	0.00	0.00
0.00	0.01	0.02	0.03	0.02	0.01	0.00
0.01	0.02	0.05	0.06	0.05	0.02	0.01
0.01	0.03	0.06	0.08	0.06	0.03	0.01
0.01	0.02	0.05	0.06	0.05	0.02	0.01
0.00	0.01	0.02	0.03	0.02	0.01	0.00
0.00	0.00	0.01	0.01	0.01	0.00	0.00

Figure 5.4.6: A 7×7 Gaussian filter [16]

5.4.2 Nonlinear spatial filtering

In addition to performing linear functions on pixels in a moving window, there are also nonlinear functions that can be performed. A concrete application of nonlinear spatial filtering is image smoothing, which has already been covered for linear spatial filtering in section 5.4.1.1. Therefore, only the different nonlinear filters are promptly discussed here [1].

5.4.2.1 Image smoothing

- **A variance filter** is a $w \times w$ filter that computes the variance of the pixels that are covered by the window. This filter acts as an edge detector, because in high heterogeneous areas the value will be rather high and in homogeneous areas rather low. This method is highly computational.
- **A median filter** is a $w \times w$ filter that ranks the covered pixels. The centre pixel of the window will have the median as value. This filter acts very well when noise needs to be removed. This is demonstrated in Figure 5.4.7. The benefit of using this filter over the other described linear filters, is that edges in the images are maintained.

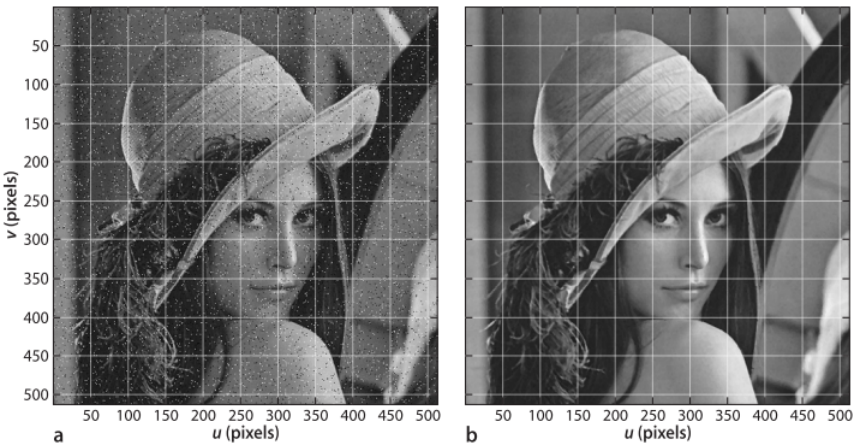


Figure 5.4.7: Image smoothing: a) original image with salt and pepper noise, b) image smoothed with a median filter [1]

5.5 Shape changing

All previous described methods cause a change in the pixel values while the location of the pixels remains. However, also techniques exist that do not change the pixel values, but its location in the image. These are called shape changing techniques and are described in the next sections.

5.5.1 Cropping

Cropping is a simple shape changing method that selects a rectangular box from the source image and presents this box as a region of interest (ROI). An example of cropping is shown in Figure 5.5.1. A 77×22 box is placed around the mouth of the Mona Lisa image. The result of the cropping is the ROI, being Mona Lisa's smile.

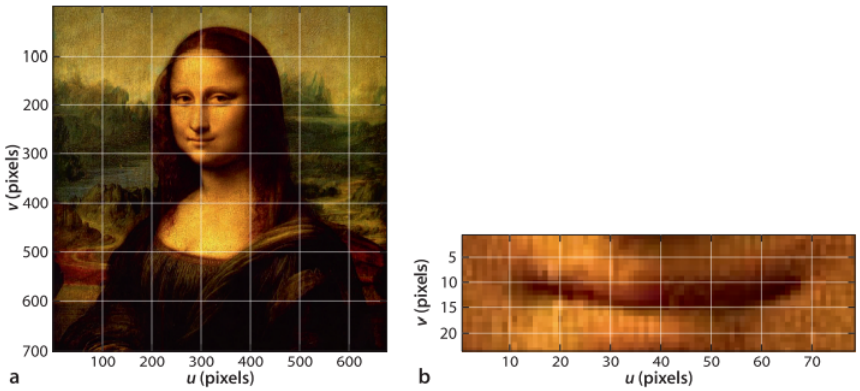


Figure 5.5.1: Image cropping: a) original image, b) ROI due to cropping [1]

5.5.2 Image resizing

The aim of image resizing is to change the dimensions of the original input image while keeping everything in the image (not as in cropping). A possible reason for reducing the dimensions can be to reduce the computational load when processing the image. Subsampling reduces the dimensions by only picking the $n \times m^{th}$ pixels in the required directions. Suppose $m = 7$, then a $N \times N$ image will be reduced to a $N/7 \times N/7$ image. The size of the resulting image is approximately the half of the original image size. As can be seen in Figure 5.5.2, the effect of subsampling is that straight lines appear as curved on the subsampled image. The inverse operation of subsampling is pixel replication. Here, every pixel is replicated into a $m \times m$ block. The effect of replication is that the image appears blurred and also blocked when looking at the straight lines.

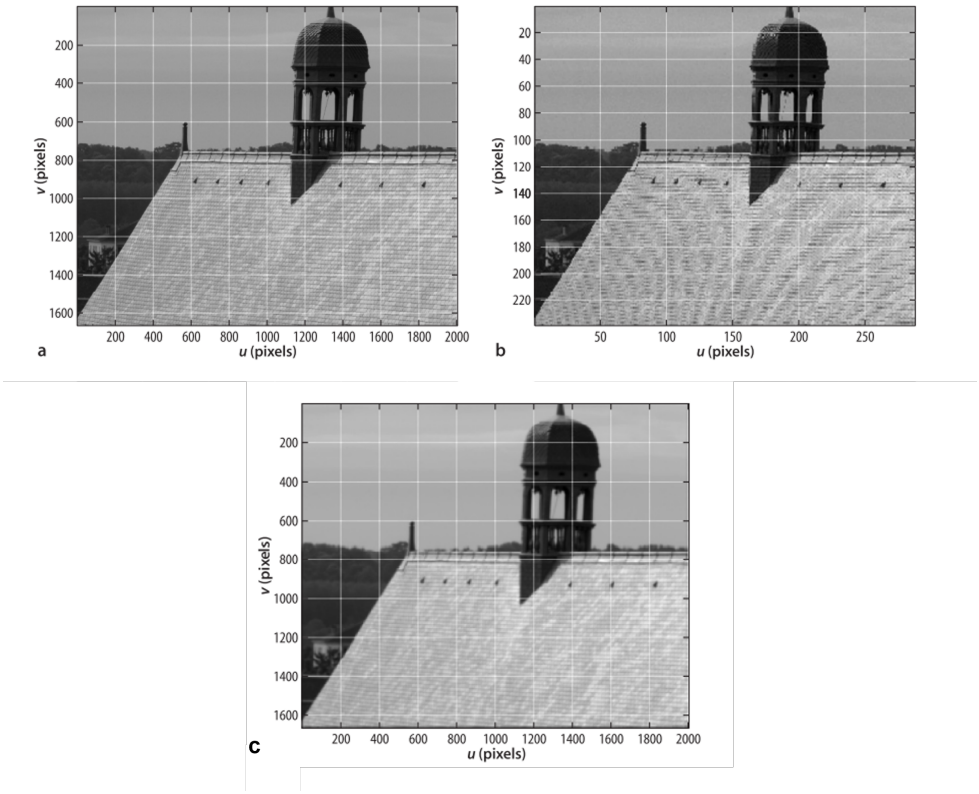


Figure 5.5.2: Image resizing: a) original image, b) subsampled with $m = 7$, c) restored to the original size by pixel replication [1]

5.5.3 Image pyramids

This is a recurring concept in machine vision. An image pyramid exists of a sequence of images which resolution is scaled down in successive steps. An example is shown in Figure 5.5.3 where each subsequent image is the half of the previous image. An application where this concept is used, is object search. The image with the lowest resolution is used to initialise the search, because this requires the lowest amount of pixels to go through. Then the search is refined in the next larger image, until the object is found in the highest resolution image.

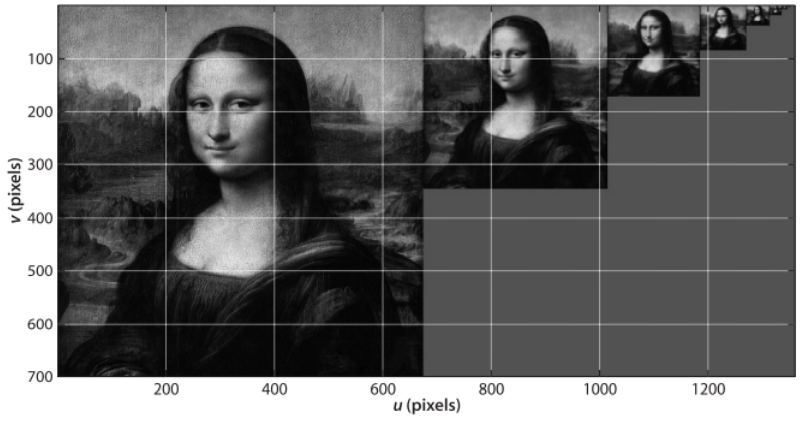


Figure 5.5.3: Image pyramid where each subsequent image is the half of the previous image [1]

5.5.4 Image warping

Image warping is the collection of transformation techniques that are focused on the transformation of pixel coordinates instead of pixel values. The new coordinates (u', v') are calculated by applying functions to the original coordinates (u, v).

$$u' = f_u(u, v) \qquad v' = f_v(u, v)$$

5.5.4.1 Scaling

Scaling will resize the main axis of the image, similar to resizing. If the value S is bigger than 1, the image will be enlarged in size in the selected direction. When the value S is lower than 1, the image will be reduced in size in the selected direction. In matrix notation, this is:

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} S_u & 0 & 0 \\ 0 & S_v & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$u' = S_u.u \;\; \& \;\; v' = S_v.v$$

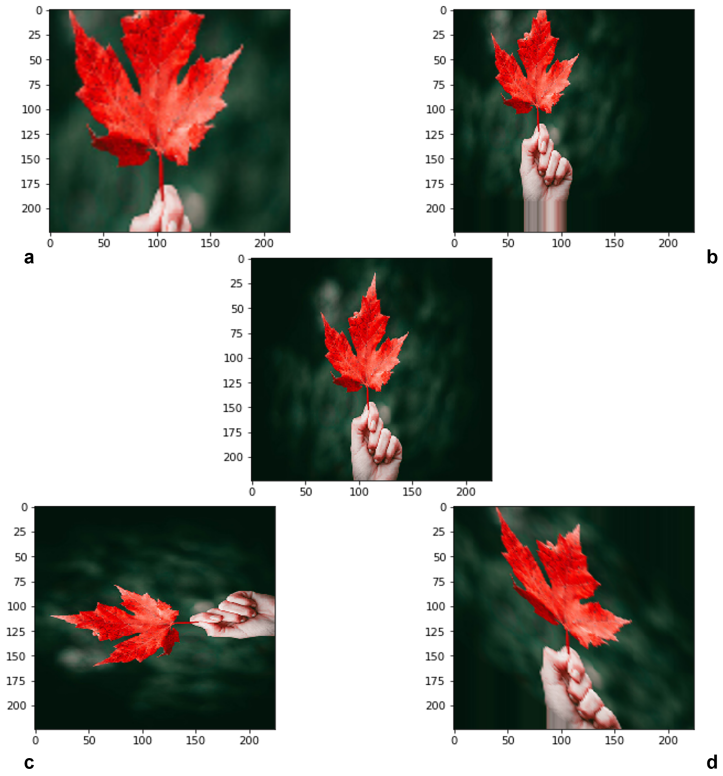


Figure 5.5.4: Image warping of the original image in the middle: a) Scaling, b) Translation, c) Rotation, d) Shearing

5.5.4.2 Translation

Translation will literally translate the image in a linear direction. All the image coordinates are translated by a certain value. If the value t is bigger than 0, the image will shift, dependent of the selected direction, to the right or to the top. When the value t is lower than 0, the image will shift, dependent of the selected direction, to the left or to the bottom. In matrix notation, this is:

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_u \\ 0 & 1 & t_v \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$u' = u + t_u \;\; \& \;\; v' = v + t_v$$

5.5.4.3 Rotation

Rotation will rotate the image around the origin. The point of rotation can be changed from the origin to the centre (u_c, v_c) by subtracting the centre values from the (u, v) values. A positive value for the angle θ will provide a clockwise rotation, while a negative value for the angle θ will provide a counterclockwise rotation. In matrix notation, this is:

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} u - u_c \\ v - v_c \\ 1 \end{bmatrix}$$

$$u' = \cos\theta(u - u_c) - \sin\theta(v - v_c) \;\; \& \;\; v' = \sin\theta(u - u_c) + \cos\theta(v - v_c)$$

5.5.4.4 Shearing

Shearing will stretch the image along its diagonal. It displaces each row or column in a way that the difference between two consecutive rows or columns is equal to the β_u and β_v value respectively. In matrix notation, this is:

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \beta_u & 0 \\ \beta_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$u' = u + \beta_u.v \;\; \& \;\; v' = v + \beta_v.u$$

5.5.4.5 Advanced warping transformations

The above mentioned warping techniques are the basis techniques that can be further combined to obtain more complex transformations such as Euclidean, Similarity, Affine, and Projective transformations which are shown in Figure 5.5.5.

- **Euclidean transformation:** rotation + translation
- **Similarity transformation:** Euclidean transformation + uniform scaling
- **Affine transformation:** Similarity transformation + shearing
- **Projective transformation:** Affine transformation + projective wraps

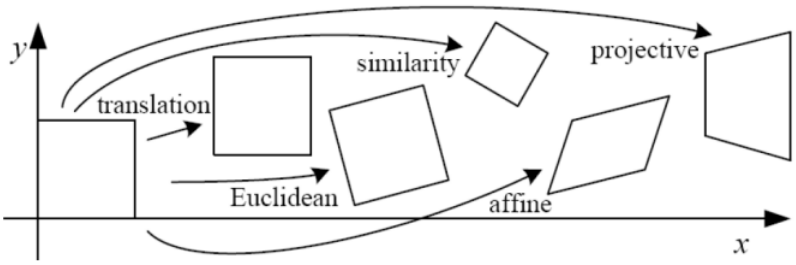


Figure 5.5.5: Image warping: combination of basic techniques into more complex image warping techniques [17]

5.6 Histogram equalisation

A histogram is a graph that shows the distribution of the pixel intensities of an image. This can be obtained for a grey scale image, resulting in one single histogram. But also for a colour image, resulting in one histogram for each channel. Some examples of histograms are provided in Figure 5.6.1. To enhance the image means increasing the global contrast. The effect of increasing that contrast on the histogram is making the distribution flatter. This can be obtained by performing histogram equalisation [16].

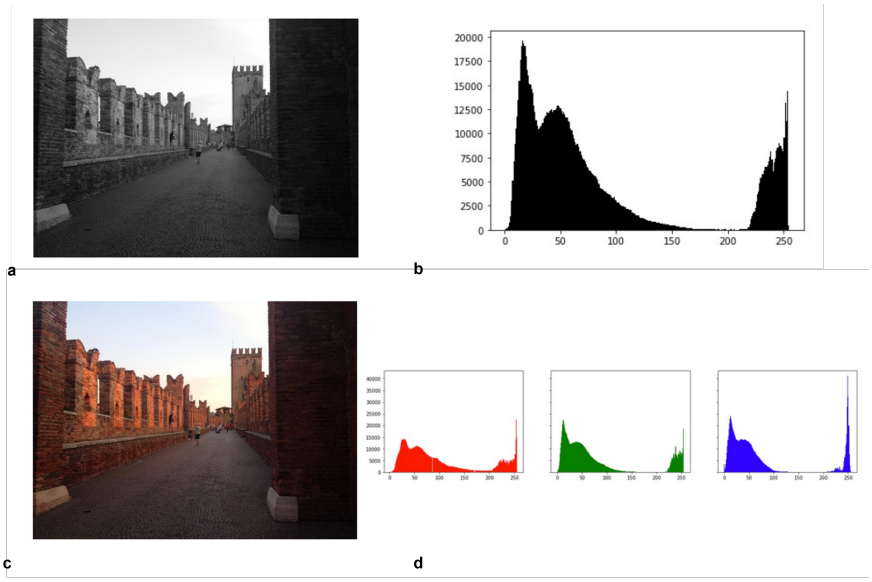


Figure 5.6.1: Image histograms: a) original grey scale image, b) histogram of a), c) original colour image, d) histograms of c) [18]

Histogram equalisation is determining a mapping function from the image histogram. This mapping function is then applied to the pixel intensity to obtain a flatter histogram. The flattening is achieved by moving the histogram bars apart or stack them on top of each other. The bars can not become smaller. For this purpose, inverse transform sampling is used. To illustrate how inverse transform sampling works, a simple example is presented next [54]. Given the histogram, frequencies of each pixel intensity can be determined. Using these frequencies, the pixel intensity probabilities can be calculated (pixel intensity frequency/total pixel frequency). The next step in this example is to calculate the cumulative distributive function (CDF). When multiplying the CDF with the amount of possible grey scale values - 1 and rounding this result off to the nearest lowest natural number, the mapped values are obtained.

Grey level value	CDF	CDF * (levels-1)
0	0.11	0
1	0.22	1
2	0.55	3
3	0.66	4
4	0.77	5
5	0.88	6
6	0.99	6
7	1	7

Table 5.1: Cumulative distributive function and new greyscale values [54]

The result of applying histogram equalization to the original colour image from Figure 5.6.1, is shown in Figure 5.6.2. The vertical scaling on both graphs are different. The result shows us that flattening the image histogram enhances the image significantly. Therefore, the intensity values are more spread out. This improves the contrast in lower local contrast areas.

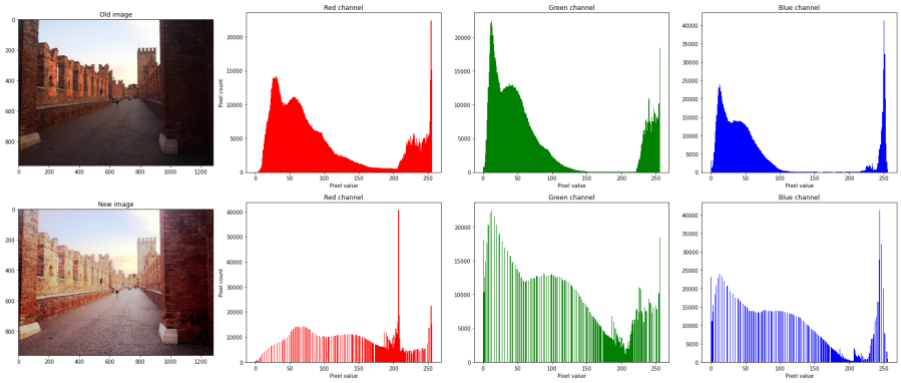


Figure 5.6.2: Image histogram equalization [18]

5.7 Thresholding

Thresholding is a method to make a clear distinction between the foreground object and the background. When looking at the greyscale histogram of Figure 5.7.1, two peaks can be observed. The largest peak corresponds to the background, while the lower corresponds to the object. The goal is to determine a threshold between those two peaks. In general, the threshold is chosen as the local minimum between those peaks. The pixel values are affected by the thresholding as follows:

$$g(u,v) = \begin{bmatrix} 1 \; if \; I(u,i) \geq T \\ 0 \; if \; I(u,i) < T \end{bmatrix}$$

The different threshold determination approaches are discussed next. In all these approaches, the assumption is made that the histogram can be modelled as two normal distributions [18].

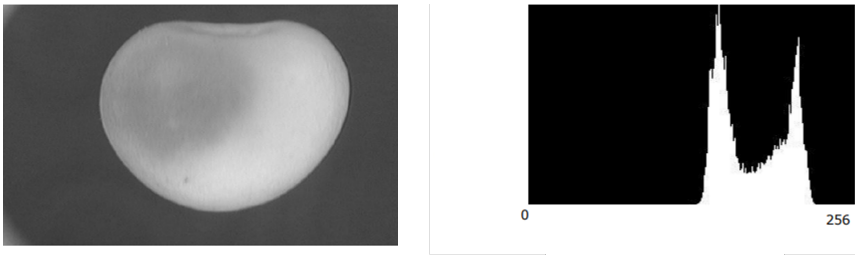


Figure 5.7.1: Histogram of a greyscale image of a cherry [18]

5.7.1 Global thresholding

The threshold T is selected by visual inspection of the image histogram. This method is very fast and intuitive, but doesn't always yield the best possible result. Especially when the two normal distributions overlap. The local minimum of the modelled distribution isn't the same as the local minimum of the real histogram. There is a need for selecting the threshold automatically.

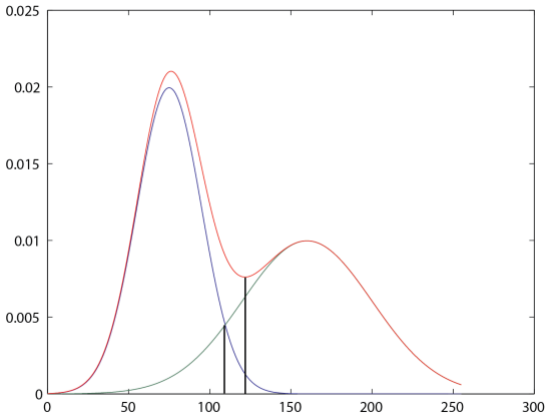


Figure 5.7.2: Image histogram modelled by 2 overlapping normal distributions [18]

5.7.2 Automatic thresholding

The Otsu's method will be explained in this section. This method selects a threshold T so that the weighted sum of the variances within the groups, divided by that threshold, is minimal. The automatic algorithm consists of the following steps:

1. Select an initial threshold estimate T
2. Divide the pixels into 2 groups: G_1, containing all the pixels where the values are $\geq T$ and G_2, containing all the pixels where the values are $< T$
3. Calculate the average intensity values μ_1 and μ_2 for groups G_1 and G_2:

$$w_1(T) = \sum_{g=T}^{255} p(g) \qquad \mu_1(T) = \frac{\sum_{g=T}^{255} p(g).g}{w_1(T)}$$

$$w_2(T) = \sum_{g=0}^{T-1} p(g) \qquad \mu_2(T) = \frac{\sum_{g=0}^{T-1} p(g).g}{w_2(T)}$$

Here g is the pixel value and $p(g)$ is the pixel probability.

4. Calculate the new threshold:

$$T_{new} = \frac{\mu_1 + \mu_2}{2}$$

5. Repeat steps 2-4 until a stopping criterion is reached

An example of Otsu thresholding is provided in Figure 5.7.3. Both object and background are well distinguished from each other.

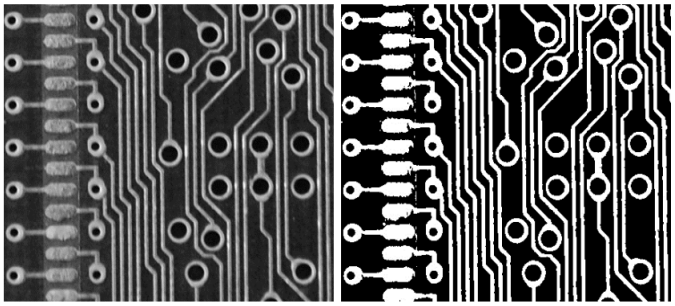


Figure 5.7.3: Otsu thresholding example [18]

5.7.3 Adaptive thresholding

Global thresholding does not perform well when the lighting conditions within the image vary. In these conditions, the use of adaptive thresholding is more suited. Adaptive thresholding divides the original image in sub-images and calculates the threshold for each sub-image. The thresholds for every pixel are interpolated by using bilinear interpolation. The size of the sub-images has an effect on the thresholding. When the size is too small, the resulting image is not segmented properly. This can be seen in Figure 5.7.4. The threshold of each sub-image can be determined in different ways. The two most common approaches in literature are mentioned below:

- **Adaptive mean thresholding**: the threshold value is the mean of the pixel values in the sub-image.
- **Adaptive Gaussian thresholding**: the threshold value is the weighted sum of the pixel values in the sub-image. The weights are specified by a Gaussian window. As shown in Figure 5.7.5, adaptive Gaussian thresholding has a better result than adaptive mean thresholding.

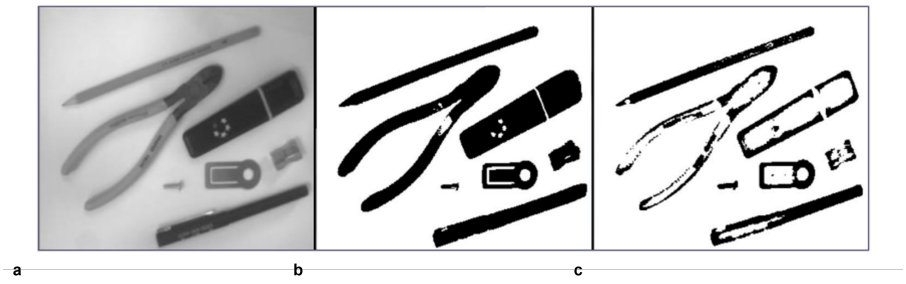


Figure 5.7.4: Adaptive thresholding: a) original image, b) adaptive thresholding with sub-images of size 101×101, c) adaptive thresholding with sub-images of size 21×21 [18]

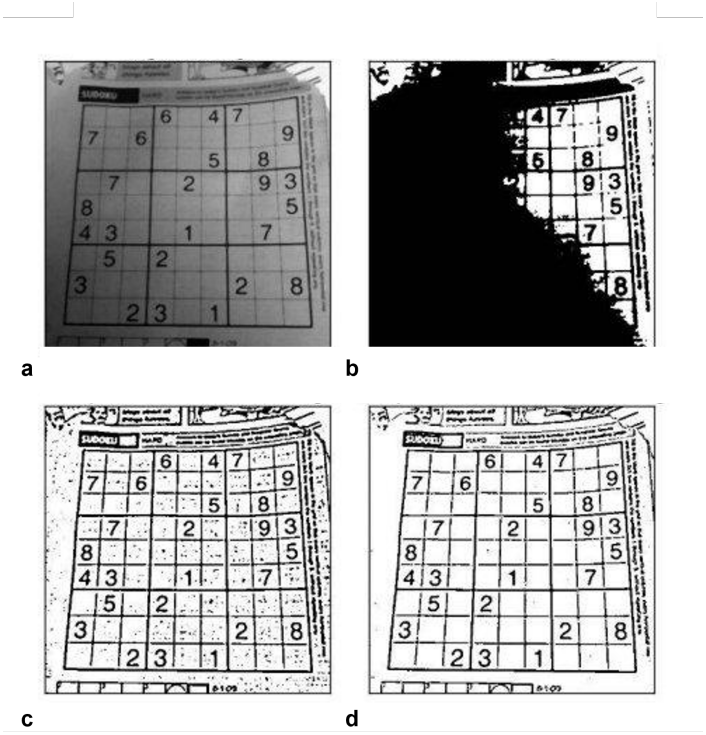


Figure 5.7.5: Adaptive thresholding: a) original image, b) global thresholding, c) adaptive mean thresholding, d) adaptive Gaussian thresholding [18]

5.8 Morphological transformations

Morphological transformations are nonlinear spatial operations, where the output pixel is a function of a selected group of pixels from the region surrounding the corresponding input pixel. This pixel selection is based on the structuring element $\mathcal{S}$. A structuring element $\mathcal{S}$ is a small binary image where a pixel in the input image is selected when the overlapping value of $\mathcal{S}$ for that pixel is 1. Technically, the functioning of the structuring element resembles that of the convolutional kernel. The difference being that a convolution is applied to all the pixels of the overlapping window, while a morphological transformation is only applied to a selection of the pixels of the overlapping window [1].

$$\forall\ u, v \in \mathbf{I}, \forall\ i, j \in \mathcal{S} : \mathbf{O}[u, v] = f(\mathbf{I}[u + i, v + j])$$

A schematic representation of a morphological transformation is shown in Figure 5.8.1. Based on the format of $\mathcal{S}$, a distinction between the different morphological transformations can be made. The different types and their applications will be discussed further in this section.

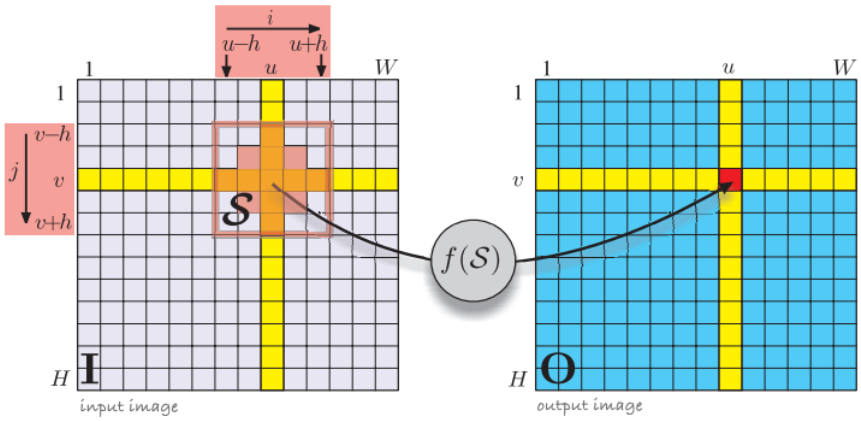


Figure 5.8.1: Morphological image processing algorithm [1]

5.8.1 Erosion

Erosion of I by $\mathcal{S}$ is retaining the group of pixels that when translated by $\mathcal{S}$ is equal to I. This is used in settings where there is noise that has to be removed. So erosion thins objects in a binary image. To illustrate how erosion works, an example is provided in Figure 5.8.2. The central pixel of the structuring element $\mathcal{S}$ overlaps all the pixels belonging to objects, these are the 1 values. When the structuring element only overlaps 1 values, then the output pixel corresponding to the central pixel of $\mathcal{S}$ will have a 1 value ass well. From the moment 1 of the overlapping values by $\mathcal{S}$ is 0, the output pixel of the corresponding central pixel of $\mathcal{S}$ is 0 [18].

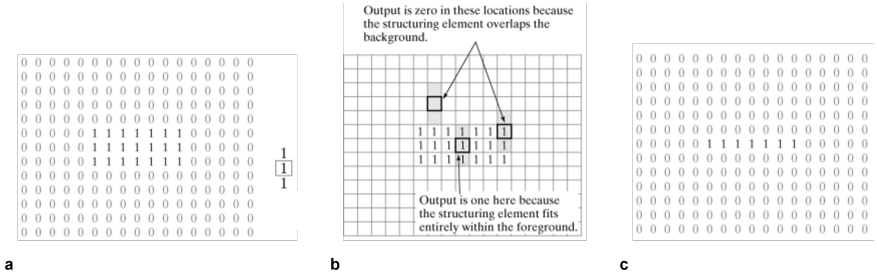


Figure 5.8.2: Erosion method: a) original image I and structuring kernel $\mathcal{S}$, b) erosion operation, c) erosion result [18]

The effect of applying erosion on a real binary image of lines is shown in Figure 5.8.3. Here, it is clearly visible that the lines are thinned in the resulting image. The symbolic representation of an erosion is $\ominus$. Thus, the formula for doing an erosion is:

$$\mathbf{O} = \mathbf{I} \ominus \mathcal{S}$$

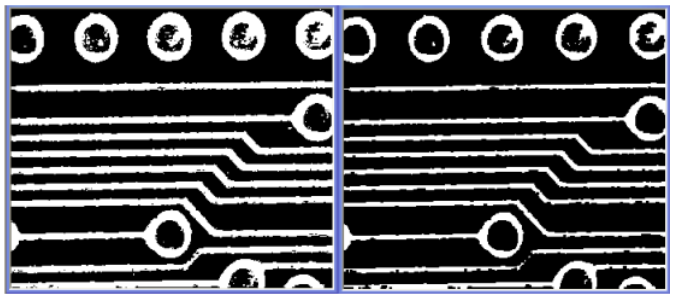


Figure 5.8.3: Erosion example [18]

5.8.2 Dilatation

Dilation of I by $\mathcal{S}$ is retaining the group of pixels that contains the translation of I by $\mathcal{S}$. This is used in settings where small holes needs to be filled or broken segments need to be joined. So dilation thickens objects in a binary image. To illustrate how dilation works, an example is provided in Figure 5.8.4. The central pixel of the structuring element $\mathcal{S}$ overlaps all the pixels belonging to objects, these are the 1 values. When the structuring element overlaps 1 values in the input image, then all the pixels that are covered by $\mathcal{S}$ will all have 1 values ass well [18].

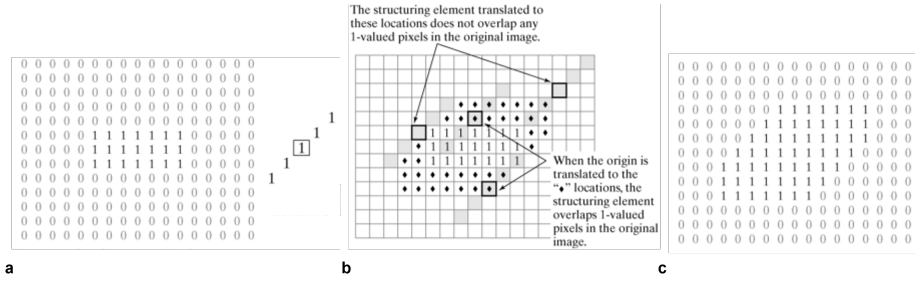


Figure 5.8.4: Dilation method: a) original image I and structuring kernel $\mathcal{S}$, b) dilation operation, c) dilation result [18]

The effect of applying dilation on a real binary image of lines is shown in Figure 5.8.5. Here, it is clearly visible that the lines are thickened in the resulting image. The symbolic representation of a dilation is $\oplus$. Thus, the formula for doing a dilation is:

$$\mathbf{O} = \mathbf{I} \oplus \mathcal{S}$$

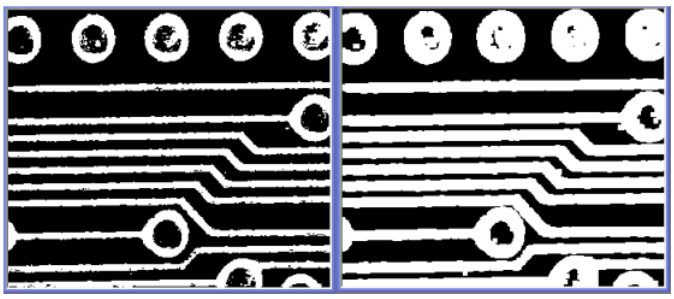


Figure 5.8.5: Dilation example [18]

5.8.3 Opening

Opening of I by $\mathcal{S}$ is performing subsequently an erosion and a dilation. This is used to mainly remove noise from the original image. Erosion is applied first to remove the noise, but this also thins the objects in the image. Dilation restores the original dimensions of the objects. The effect of opening is shown in Figure 5.8.6. The symbolic representation of an opening is $\circ$. Thus, the formula for doing an opening is:

$$\mathbf{O} = \mathbf{I} \circ \mathcal{S} = (\mathbf{I} \ominus \mathcal{S}) \oplus \mathcal{S}$$

Figure 5.8.6: Opening example [19]

5.8.4 Closing

Closing of I by $\mathcal{S}$ is performing subsequently a dilation and an erosion. This is used to mainly close small gaps within objects from the original image. Dilation is applied first to remove the gaps, but this also thickens the objects in the image. Erosion restores the original dimensions of the objects. The effect of closing is shown in Figure 5.8.7. The symbolic representation of a closing is $\bullet$. Thus, the formula for doing a closing is:

$$\mathbf{O} = \mathbf{I} \bullet \mathcal{S} = (\mathbf{I} \oplus \mathcal{S}) \ominus \mathcal{S}$$

Figure 5.8.7: Closing example [19]

5.8.5 Skeletonization

Skeletonization of I is performing subsequent erosions until the area of interest is 1 pixel wide. The result is a skeleton of the original image. The benefit of having such a skeleton is that the unnecessary pixels are removed and only the pixels, that contribute to the shape of that image, remain. This results in a significant data reduction. The obtaining of a skeleton is shown in Figure 5.8.8.

Figure 5.8.8: Skeletonization example [19]

Chapter 6

Feature extraction

As seen in the previous chapters, images contain a lot of data in the form of matrices of pixel values. The chapter on image processing described several techniques to transform these matrices in order to highlight some features and improve contrast. However, these techniques still output a matrix of pixel values and thus a lot of data. But this data does not contain a lot of information compared to the dimensions of this data. Therefore, feature extraction is used to extract information from images and to reduce the data significantly. This data reduction is crucial for a machine vision setup to make a decision or description based on an image as input. The next sections will elaborate on what exactly features are and how they can be detected, described, and used.

6.1 What is a feature?

Features are basically regions/artefacts/pixels of an object in an image that contain lots of information and make it possible to compare and distinguish the object with another object. Given this explanation, the real definition of a feature can best be explained with some examples shown in Figure 6.1.1.

- In the first example, we see a red and a green circle. For a human, both are easily differentiable, and most people can directly point out the red and the green circle. But why? What exact property do we look at when comparing both? In this case it is the colour or hue of the circles, which is just one property or feature. In this case, the useful information that we look at to compare and distinguish two objects is the colour.
- When we look at the lower example it becomes more complex. Imagine that we have a pad where we draw the letter M and the letter J. We can ask ourselves the same question: What properties or features do we look at to compare and distinguish both? If we capture the x, and y-data of the pen in time, we can define the aspect ratio and the time needed to write the letter as possible features that can distinguish both as an J has a higher aspect ratio and can be written much faster.
- Finally, when we look at the middle example of the cat and the dog. What features enable us to compare and distinguish both? This is definitely a much harder question to answer. Because both have two eyes, two ears, paws, a nose, fur of a certain colour, etc. At this point one can reflect on the fact that despite it is very straightforward for humans to describe a scene and compare objects, we do not always know how we do this. Still, in machine vision, we need to be able to describe the proper features to compare and distinguish numerous objects.

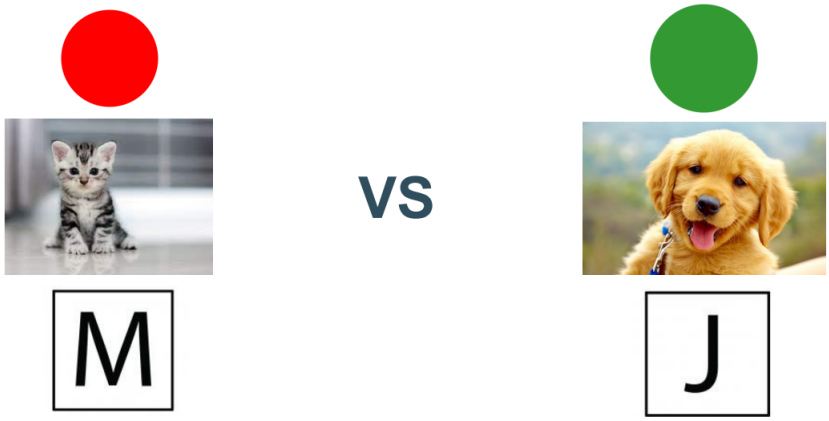


Figure 6.1.1: Features examples

Another approach on how to define features is using the analogy of making a puzzle. Imagine Figure 6.1.2 to be a puzzle and one needs to find the exact location of piece A or B in the image. This is relative hard as these pieces do not contain a lot of information. When looking at piece C or D, the problem becomes more easy as the pieces contain more information, being a line. When looking at piece E or F, the problem is very easy and one can directly find the correct location of the piece in the image. This because these pieces contain a much more informative feature than a line, being a corner. Corners are therefore good features as they make it easy to compare and distinguish them from other pieces in the image.

Figure 6.1.2: Flat surfaces, edges and corners as features [19]

Beside corners and also edges, other useful features are called blobs. Blobs are regions of similar pixels that, for example, have the same colour. Let's take a look at Figure 6.1.3. If we want to obtain the bounding box of the shark, one can look for all the white pixels and obtain the minimum and maximum u and v coordinate to obtain the box. One can compute the centroid of the box knowing these coordinates. But what when more sharks are in the image as in Figure 6.1.4?

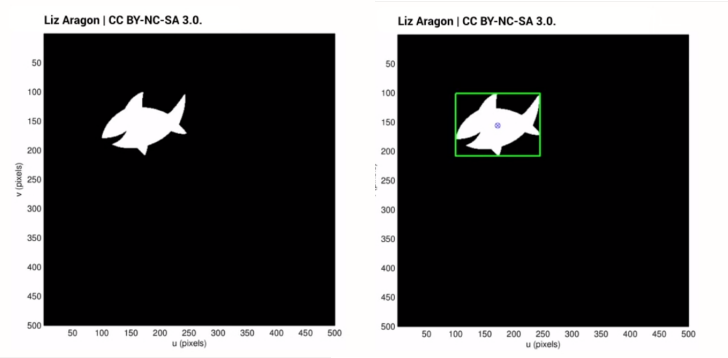


Figure 6.1.3: Blob detection with one object

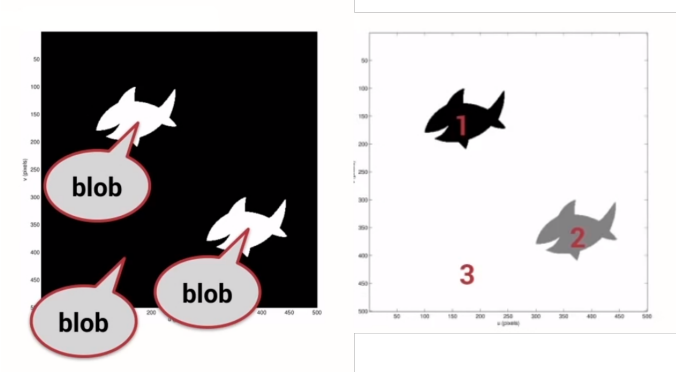


Figure 6.1.4: Blob detection with multiple objects

Here the usefulness of blobs comes into play. A blob is a certain continuous region in which all the pixels have the same value. Each pixel is labelled indicating to which blob it belongs. Every pixel has the same label as its N, S, E, or W neighbour of the same value. The process of obtaining the blobs is called connectivity/blob analysis which can be done using clustering algorithms.

6.2 Feature detection

Now that we know what features are, we want to find them in an image to be able to compare objects in the image and distinguish them with other objects for, for example, classification. Several feature description algorithms exist but we will only cover the ones for blob detection, corner detection, and edge detection.

6.2.1 Blob detection

Blobs are generally detected using clustering algorithms. These are also called unsupervised learning algorithms as they model the data without the usage of labels given by a human to this data.

6.2.1.1 K-means

The K-means algorithm is one of the more well-known clustering algorithms. An assumption that has to be made, is that the number of clusters are known. Clusters are non-overlapping subsets of similar data points. The following steps are performed within this algorithm:

1. Specify the number of clusters K.
2. Initialise the centroids by randomly choosing K data points.

3. Calculate Euclidean distance, used here as a similarity metric, between a data point and all the centroids. Assign the data point to the cluster of the centroid where the Euclidean distance is minimal. Do this for all data points.

4. Update the cluster centroids by computing the average of all the data points of that cluster.

5. Perform steps 3-4 until the centroids stay the same.

The sum of the squared distance between all the data points and centroids is computed to show the variance within the clusters. K-means tries to minimise that sum of squared distance, so that the variation within the clusters is as low as possible. This means that the clusters consist of similar data points. Different initialisation of the centroids will lead to different clusters. Therefore, it is advised to perform K-means multiple times. The best assignment of clusters is then chosen based on the minimal sum of squared distance. In many real-world applications, there is no know-how on how many clusters there are in the data set. In these settings, the mean shift algorithm is a good solution [55].

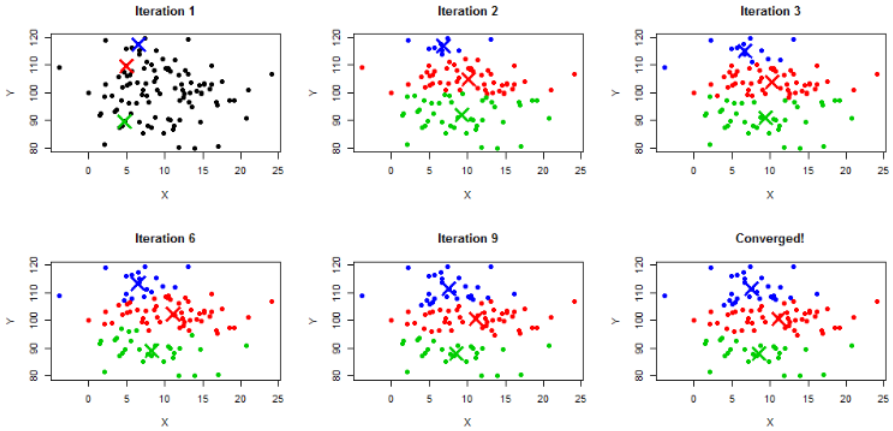


Figure 6.2.1: K-means clustering example [20]

6.2.1.2 Mean shift algorithm

The mean shift algorithm has the advantage that the amount of clusters do not need to be specified. Therefore, this is called a non-parametric algorithm. The main idea here is the same as for K-means, namely to divide the data points in clusters. However, the method of dividing these data points differs in both algorithms. In the mean shift algorithm, a cluster consists of all data points that are in the attraction basin of a mode. The attraction basin is the region for which all trajectories lead to the same mode. Here, a mode is similar to a cluster centroid. The following steps are performed in this algorithm:

1. Model the data point density by kernels with specific bandwidths. An example is shown in Figure 6.2.2.

2. Centre a predefined window on a data point.

3. Compute the mean of this window. The mean is located in regions with a high point density.

4. Centre that window on the calculated mean.

5. Perform steps 3-4 until the mean stays the same.

6. Select a new data point and perform steps 2-5. Do this until all data points are handled.

7. Assign points that lead to nearby modes to the same attraction basin.

An example of two attraction basins with their respective modes are presented in Figure 6.2.3. The lines indicate the way the means change until they converge to the nearby modes [18].

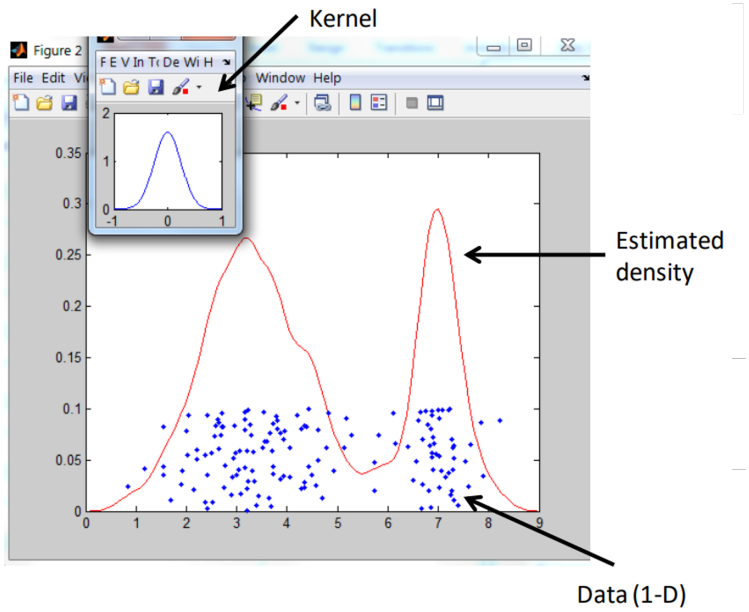


Figure 6.2.2: Kernel density estimation

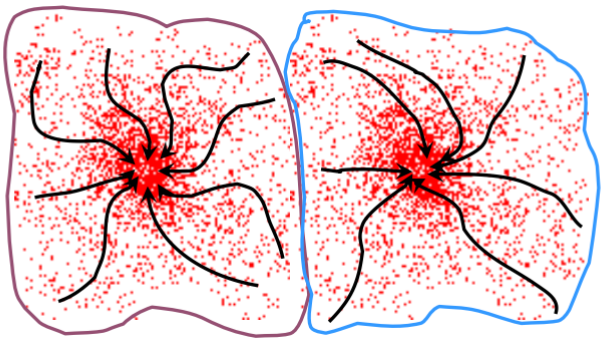


Figure 6.2.3: Mean shift algorithm: Two attraction basins with their respective modes

6.2.2 Corner detection

Two popular algorithms to detect corners are the Harris Corner detection algorithm and the SIFT algorithm. This algorithm searches for regions with maximum variation in all directions.

6.2.2.1 Harris corner detection

The Harris corner detection algorithm checks the variation of pixel intensity in both x and y directions for different regions. These regions are defined by putting a window over the image. This algorithm detects corners when a maximum variation of pixel intensity occurs. The formula for the difference in pixel intensity is:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{window function} [\underbrace{I(x+u,y+v)}_{shifted intensity} - \underbrace{I(x,y)}_{intensity}]^2$$

Here, (u, v) is a set of displacements in the x and y directions. An example of the change in pixel intensity within a region is illustrated in Figure 6.2.4. The flat region consists of pixels of the same value, thus no significant difference in pixel intensity occurs. This can be seen in the intensity variation points, which are centred around the origin. The edge causes a significant difference in pixel intensity in the window and primarily in the x direction. A simple explanation for the intensity variation points being centred around the x axis is because when moving in the y direction, the pixel intensity does not change while moving in the x direction does because the edge is traversed. The corner causes also a significant difference in pixel intensity in the window, but now in both x and y direction. Therefore, the intensity variation points are clustered around the x axis, this corresponds to the vertical edge, and in the 3rd quadrant, this corresponds to the angled edge.

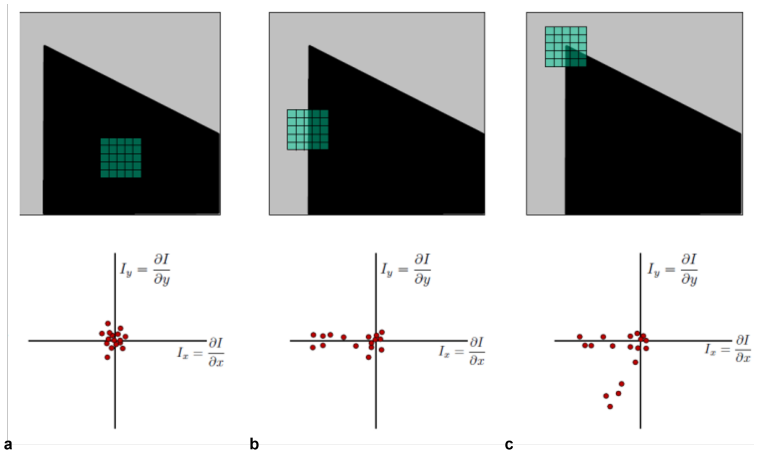


Figure 6.2.4: Features and the intensity variation in x and y direction for: a) flat region, b) edge, c) corner

The Harris corner detection algorithm outputs R-scores, which indicate when a large pixel intensity variation occurs when moving in all directions. When thresholding these R-scores, the corners are made visible. An example of this algorithm applied to a simple image is shown in Figure 6.2.5.

- High R-scores: corners
- Low R-scores: flat areas
- Negative R-scores: edges

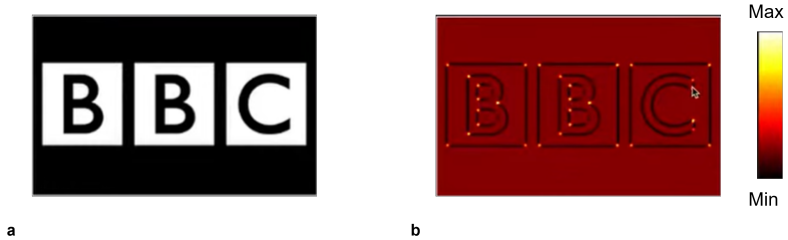


Figure 6.2.5: Harris corner detection: a) original image, b) corner response [21]

6.2.2.2 SIFT

SIFT is the abbreviation for scale invariant feature transform. These SIFT features are invariant to rotation and scaling. When complex objects need to be detected, SIFT features are mainly used over Harris corner detection

features. The improvement over the Harris corner detection features, which are only rotation-invariant, is the scale-invariance. To illustrate this improvement, suppose a corner is detected. When a corner is rotated, the corner still stays a corner and can therefore be detected by the Harris corner detection. But when a corner is scaled, the output may not always again be a corner and can therefore not be detected by the Harris corner detection. This shows the added value of using SIFT features. SIFT features are detected by performing following steps [56]:

1. Obtain an image where the features need to be detected.
2. Create a stack of images is by smoothing the image with a factor $k^n\sigma$. This stack of images is called the Gaussian scale space.
3. Subtract consecutive images of the scale space from each other to obtain Difference of Gaussians (DoG).
4. Find the local extrema for the DoG, for example by placing a 3x3 moving window over the DoG. These are interest point candidates.
5. Discard the weak candidates: low contrast points or points along an edge. The remaining points are the SIFT features.
6. Do steps 2-5 again for the same image but at another scale (space).

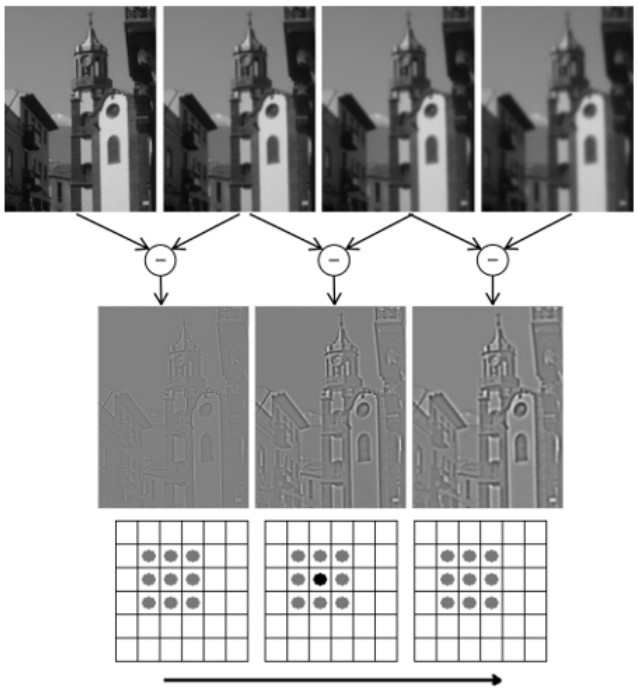


Figure 6.2.6: SIFT interest point candidates detection: smoothing an image to obtain a scale space, subtract them from each other to obtain the DoG and find the local extrema of the DoG [22]

In Figure 6.2.7, interest point candidates are illustrated. Some of them are discarded to acquire the definitive SIFT features.

Figure 6.2.7: SIFT features: a) interest point candidates, b) discarding low contrast interest points, c) discarding interest points along an edge [22]

6.2.3 Edge detection

We saw that beside corners, also edges are fairly good and important features that can describe objects in an image. Detection of object edges in images is therefore key in the detection process. This section first elaborates on the principles of image gradients as these are profound in the detection of edges. Further, this section will describe the Canny Edge Detector algorithm.

6.2.3.1 Image gradients

The concept of an image gradient is key in edge detection. An image gradient is a change in pixel values, which usually occurs when traversing edges. To understand this intuitively, an edge intensity profile is shown in Figure 6.2.8. A horizontal green line is drawn on the original image (upper left) at $v = 360$. The corresponding grey scale values for each u value on that line is represented on the intensity profile (upper right). The first derivative is then taken (bottom right) of the closed up region of the curve (bottom left). This derivative indicates when a change, positive or negative, in the values occur. The peaks of the derivative indicate where an edge in the image is situated.

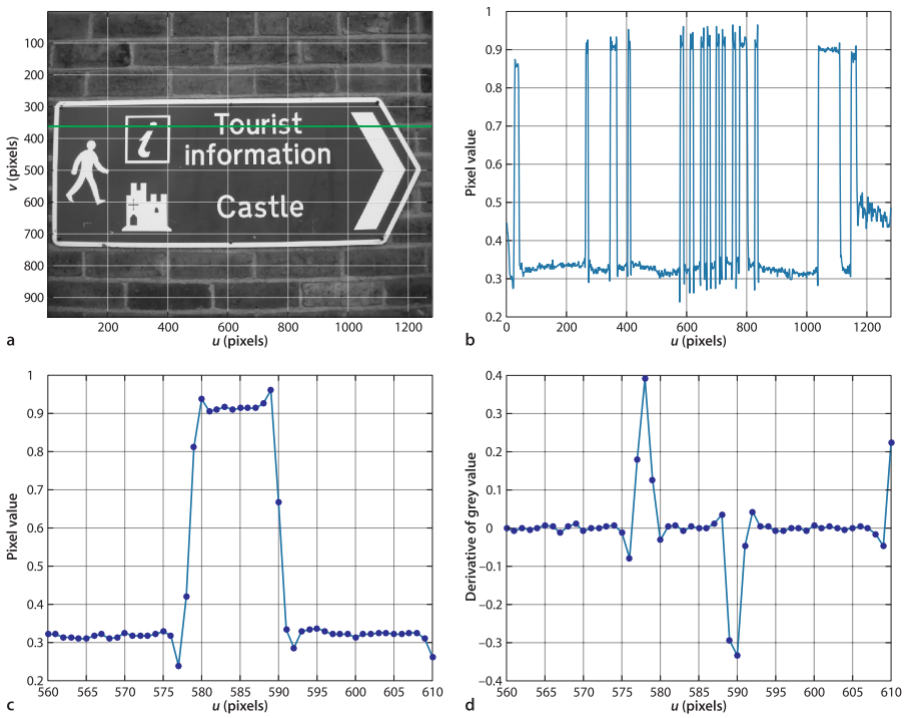


Figure 6.2.8: Edge intensity profile: a) original image, b) grey level profile along horizontal line $v = 360$, c) close-up view of spike at $u \approx 580$, d) derivative of c [1]

The gradient in x-direction of a continuous curve can be calculated by subtracting two successive values of two neighbouring pixels and dividing by the step taken, which is one pixel ($h = 1$) in this case:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The gradient can also be calculated by using the central difference formula where we subtract the values of the pixel before and the pixel after a central pixel x. As we have taken a step of two pixels, h becomes 2:

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

This is shown in Figure 6.2.9.

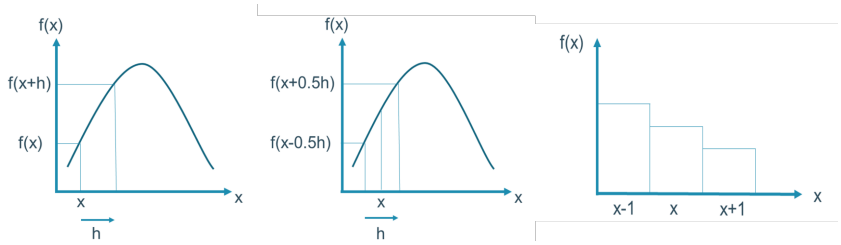


Figure 6.2.9: Continuous and discrete curves: gradient calculation

Taking the gradient in the x-direction of a discrete signal as described above is equal to performing a convolution with the following kernel:

$$K = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

If we slide this kernel on a pixel row from left to right and making the convolution for every pixel, we get for every pixel a value of: minus one times the pixel value before the reference pixel plus one times the pixel value after the central pixel. The resulting row of pixel values denote the first derivative of the original image row.
There is one disadvantage in using the aforementioned kernel. From calculus we know that if we take the derivative of a very noisy function, that we get a very noise derivative function. This can be seen in Figure 6.2.10. Despite seeing a clear edge in function $f(x)$, this is not visible in its derivative. And typically, images contain some form of noise and will thus be confronted with this problem when being derived. A solution for this phenomenon is to smooth the image with, for example, a Gaussian filter, before taking the derivative of it.

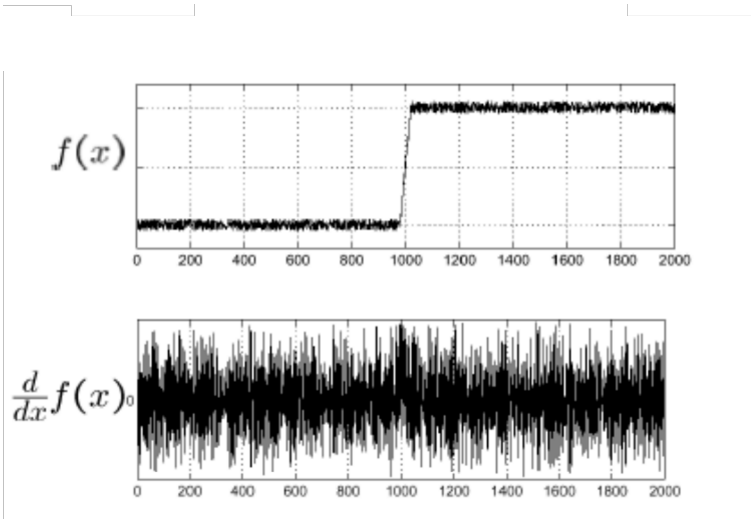


Figure 6.2.10: The effect of deriving a noisy signal [23]

Combining the previous mentioned kernel and a Gaussian smoothing filter in a three by three matrix results in a very common gradient filter called the Sobel filter. The convolution with the S_u kernel calculates the horizontal gradient of the image $\frac{\partial f}{\partial x}$, while the convolution with the S_v kernel calculates the vertical gradient of the image $\frac{\partial f}{\partial x}$.

$$S_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_v = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The results of these operations are shown in Figure 6.2.11. The horizontal gradient of the image highlights the vertical lines in particular. The vertical gradient of the image mainly highlights the horizontal lines.

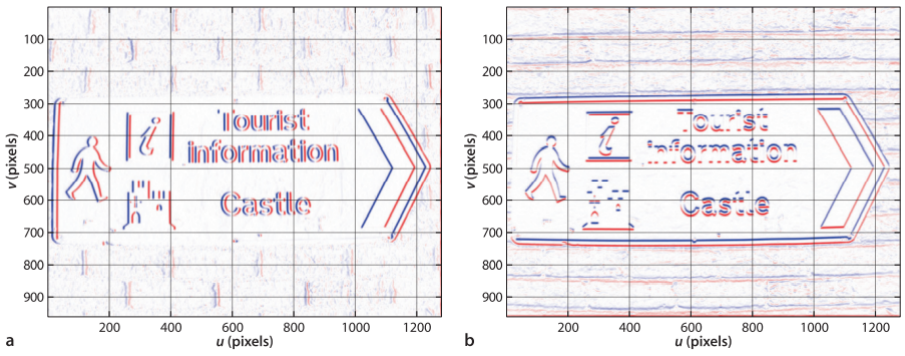


Figure 6.2.11: Convolution with the derivative kernels: a) horizontal image gradient, b) vertical image gradient [1]

The horizontal and vertical gradient can be combined to calculate the total gradient direction and magnitude. These

2 properties are shown in Figure 6.2.12 and can be calculated using following formulas:

$$\theta = \arctan(\frac{\partial f}{\partial x} / \frac{\partial f}{\partial y})$$

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

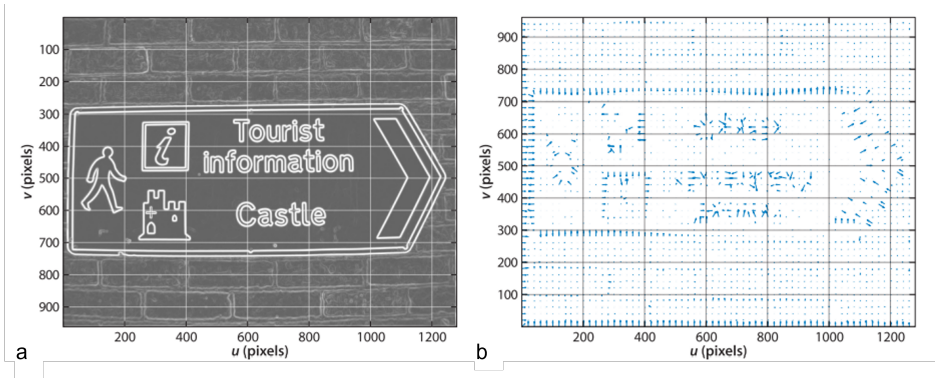


Figure 6.2.12: Edge detection: a) edge magnitude, b) edge direction [1]

6.2.3.2 Canny Edge Detection

A very popular edge detection algorithm that is based on the Sobel filters is called the Canny Edge Detection algorithm. The first step of the Canny Edge Detection algorithm is to apply both the horizontal and vertical Sobel filters to the image and calculating the total edge magnitude and direction. Additionally, the algorithm performs two more steps in order to obtain the edges in an image. The additional two steps that are performed are the non-local maxima suppression and hysteresis thresholding. Non-local maxima suppression tries to remove all the pixels that aren't part of an edge. It does this by scanning all the pixels of the images and determining if a pixel is a local maximum in its neighbourhood in the direction of the gradient. To illustrate this, an example is shown in Figure 6.2.13. Pixel A is situated on the edge, thus the edge direction of that pixel is perpendicular to its gradient direction. The pixels in a small neighbourhood that also have the same gradient direction are pixels B and C. Together with those pixels, it is checked if pixel A is the local maximum. If pixel A isn't the local maximum, then the it's value is put to zero (suppressed). Otherwise the pixel is considered for the next step.

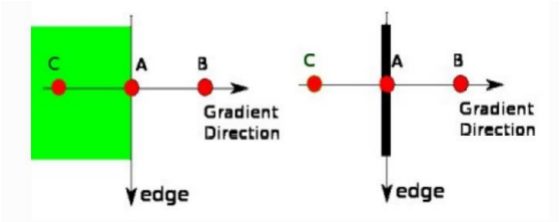


Figure 6.2.13: Non-local maximum suppression [19]

That next step is the hysteresis thresholding. This step creates a binary image by putting the pixels with an intensity bigger than a maximum threshold to one and the pixels with an intensity lower than a minimum threshold to zero. All the pixels which have a intensity that lies between those 2 thresholds are classified edge pixels. It has to be checked if a classified edge pixel really is an edge pixel based on their connectivity. For example, pixel B and C in Figure 6.2.14 are classified edge pixels. Based on the connectivity of pixel C to pixel A, pixel C is an edge pixel and put to one while pixel B is put to zero.

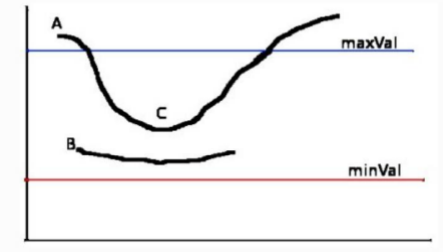


Figure 6.2.14: Hysteresis thresholding [19]

The result of the Canny Edge Detection algorithm is a binary image in which the pixels that belong to an edge are white, and all the other pixels are black. This output can be used for further image feature extraction or for simple object detection.

Figure 6.2.15: Canny edge detection [19]

6.3 Feature description

Detecting features is one thing, but we also need to be able to compare features with each other and distinguish them from each other. Therefore, we need to describe the features, and thus define how they look like. As a human, we describe features of images by explaining the region around the feature in our own words. For example in Figure 6.1.2, we could say that the upper part is the blue sky and the lower part is the grey building. In a similar way, a computer also describes the region around a feature. This is done by describing the feature in the form of a feature vector. A feature vector can be seen as an unique fingerprint, used to differentiate features from each other. This vector contains only the most useful information of the image patch surrounding the feature. Redundant information is dropped. A feature vector can be created in many ways. Some possible basic feature vector descriptors are listed below and shown in Figure 6.3.1. In the subsequent sections, some common feature descriptors are described.

- **Image patch:** An image patch of a region around a feature is taken and all the pixel values of that patch are stored in a vector. For big image patches, this results in long feature vectors. This way of describing a feature is fine when the geometry and appearance of the batch is unchanged. For example, when comparing 2 identical images with only a different illumination, these feature vectors will be different due to the dependency on the absolute pixel values.
- **Image gradients:** Image gradients, as in the difference between pixels, can also be used as feature vectors. This way of describing a feature, has the advantage that these are invariant to absolute pixel value changes.

Unfortunately, it is not invariant to deformations.

- **Colour histogram:** Computing a colour histogram of an image is also a possible way to describe a feature. The advantage of using colour histograms is that it is invariant to deformations, such as scaling and rotations.
- **Orientation normalisation:** The orientation of an image patch is normalised by computing the image gradients in all possible directions and using the dominant gradient direction as direction of the image patch. Here, the patch as well as the orientation angle are stored as a the feature description.

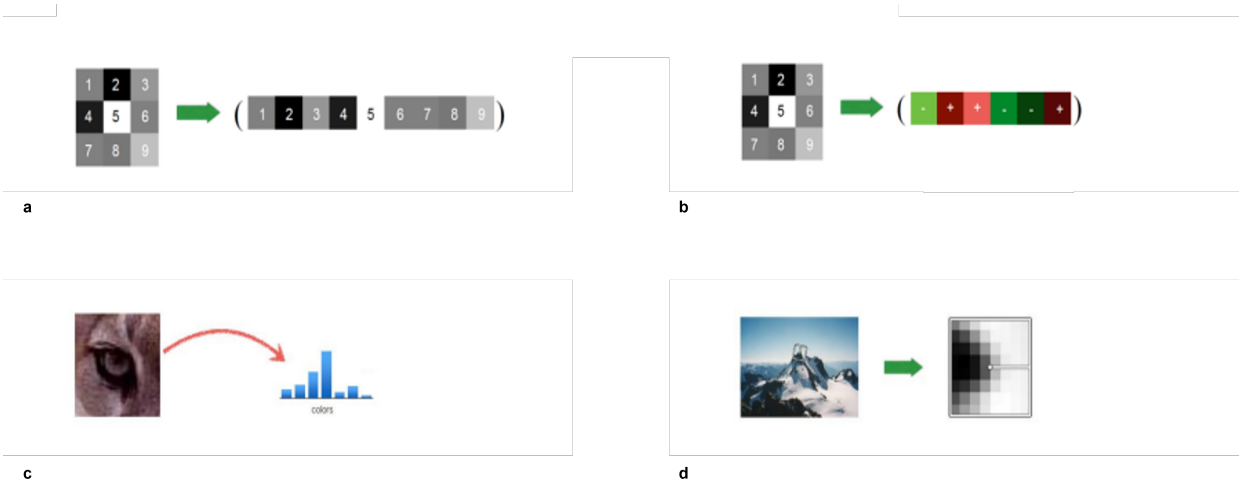


Figure 6.3.1: Feature vector descriptions: a) Image patch, b) Image gradients, c) Colour histogram, d) Orientation normalisation

6.3.1 HAAR

HAAR features are somehow atypical features that are used to detect object in an image and that are represented by HAAR-wavelets. A HAAR-wavelet is a filter containing white and black pixels, positioned at a fixed point in a sliding window, that can be convolved over an image. One HAAR-wavelet actually acts as a basic classifier that outputs a positive result if the feature represented by the HAAR-wavelet is present and a negative result otherwise. Some of these HAAR-wavelets are visible in Figure 6.3.2. Every HAAR-wavelet represent a certain feature which can be a vertical or horizontal edge, a vertical or horizontal line, an corner, etc.

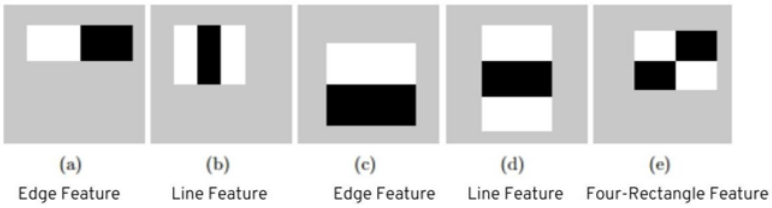


Figure 6.3.2: HAAR-wavelets [24]

The HAAR-wavelet generates a result by subtracting the sum of all the pixel values under the black area of the wavelet from the sum of all the pixel values in the white area. If the output equals 1, then the HAAR-feature represented by the wavelet is fully present in the image patch. To detect if a feature is present, a threshold is set. If the output is larger than the threshold, the feature is present, otherwise it is not present.

6.3.2 HOG

Histogram of Oriented Gradients (HOG) is another method of presenting features in the form of a feature vector. HOG does this by computing the distribution of gradient directions and stores this as an 9-dimensional vector. The use of gradients is convenient because these have large values in the proximity of edges and corners. Therefore, a distinction can be made between non-essential information, such as the monotone background, and essential information. More information about calculating the gradient magnitude and direction can be found in section 6.2.3.1. HOG uses the following steps to obtain a feature vector [25]:

1. Resize the image to 32×64 or 64×128.
2. Divide the resized image in HOG cells. For a 32×64 image, the preferred HOG cell size is 4×4. For a 64×128 image, the preferred HOG cell sizes are 8×8 and 16×16.
3. Calculate the gradient magnitude and orientation per pixel of the HOG cell. In Figure 6.3.3, the gradient magnitude and orientation per pixel of a 8×8 HOG cell are computed. This results in 128 numbers.

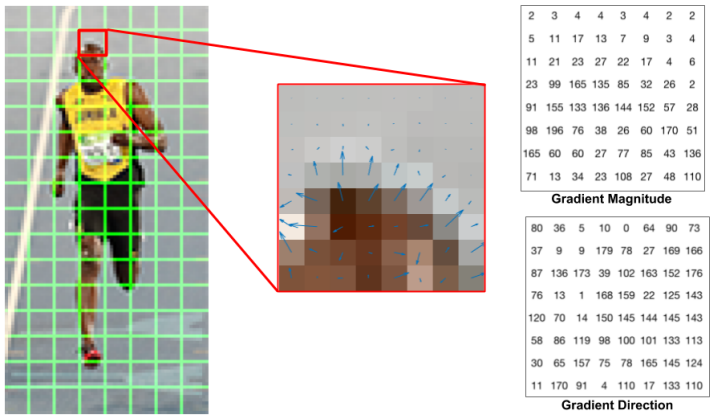


Figure 6.3.3: HOG: the gradients of a patch represented as arrows and as magnitudes and orientations [25]

4. Assign each gradient to a histogram bin. The histogram is divided in bins of 20 degrees, resulting in 9 bins. The assignment is done by a voting principle. An example is provided in Figure 6.3.4. The gradient encircled with the blue circle has a gradient orientation of 80 degrees. Therefore, the gradient magnitude of 2 is easily assigned to the 80 degrees bin. When a gradient orientation is situated between 2 bins, then the magnitude is divided as follows:

$$gradient\ magnitude \times (\frac{|bin\ degrees - gradient\ degrees|}{bin\ degrees})$$

The gradient encircled with the red circle has a gradient orientation of 10 degrees. The gradient magnitude is divided in 2 and 2 and assigned to the bin of 0 degrees and 10 degrees. If the magnitude orientation is greater then 160 degrees, then the magnitude is divided between the 160 degrees and 0 degrees bin [57].

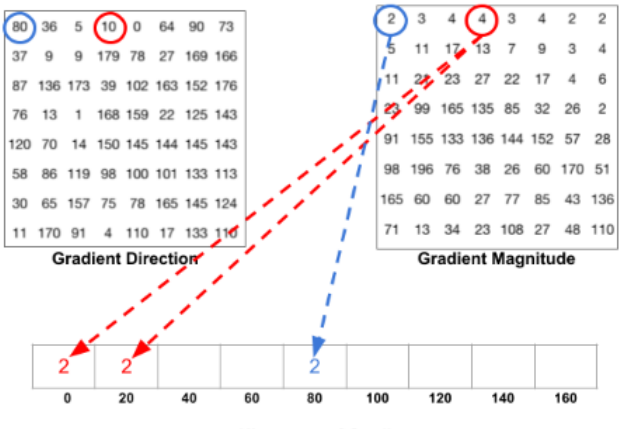


Figure 6.3.4: HOG voting principle [25]

5. The histogram is converted to a 9 dimensional feature vector.

6.3.3 SIFT

After detecting the SIFT features, discussed in section 6.2.2.2, comes describing these features. A 16×16 grid is placed around the keypoint. This grid is further divided in 16 4×4 blocks. Similar to HOG, an 8 bin orientation histogram is made for each 4×4 block. Each of those 16 histograms is concatenated and converted to a 128-dimensional feature vector [58].

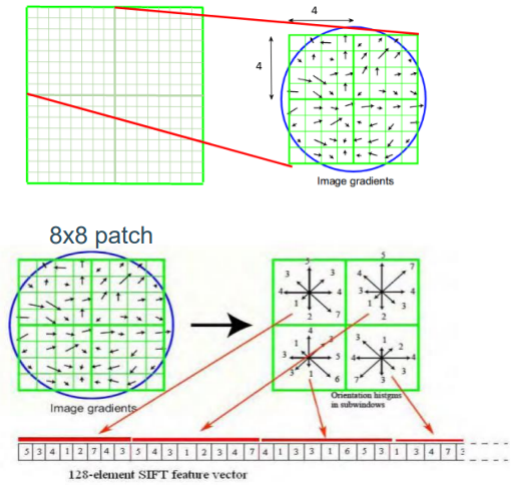


Figure 6.3.5: SIFT features description [19]

6.4 Feature matching

For several reasons, it might be interesting to find similar features in different images. This for example to find the location of an object into a scene, to find the spatial transformation between two images, or in stereo vision where two cameras are used to obtain 3D information from the scene. Feature matching is the process of matching the same features in two different images of the same scene. In both images, features are detected and feature descriptors for each feature are obtained. For every feature in image one, the corresponding feature with the most equal feature descriptors in image two is searched. Figure 6.4.1 shows some feature matches between two images.

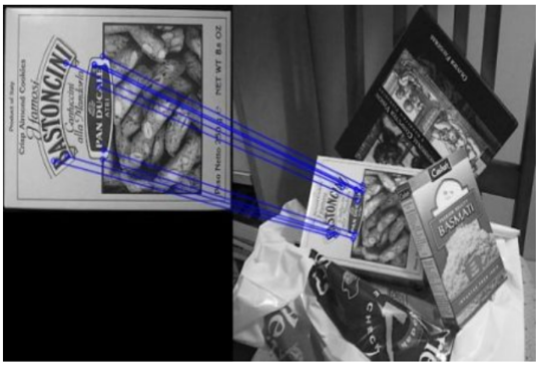


Figure 6.4.1: Feature matching [19]

6.4.1 Applications

Feature matching can be useful in several applications:

- Object localisation: One can have an image of an object and an image of the object located in a scene. Using feature matching of detected and described features in both images, the object in image one can be located in image two. Figure 6.4.2 shows this.

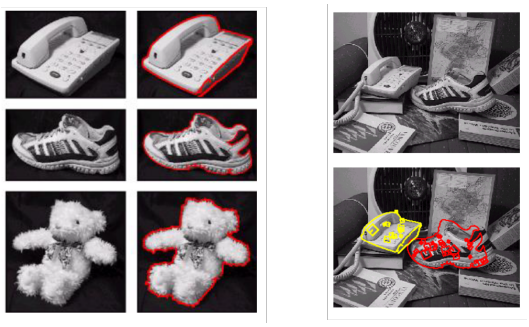


Figure 6.4.2: Object localisation [19]

- Panorama stitching: In panorama stitching, multiple images of the same scene but from different viewpoints are stitched together. To have a smooth stitching the second image should be aligned with the first image, and the third one with the aligned second one, etc. The alignment can be done by finding the matrix that defines the displacement the camera made between two images and using this matrix to align the second image. Transforming an image in such a way is called warping as seen in Section 5.5.4 and the used transformation is perspective transformation. To find the matrix, an algorithm can be used that finds the matrix based on the corresponding features in the two images. If an object is visible in two successive images, the features of the object in both images can be matched and the displacement of the features is an indication of the displacement matrix of the camera. An example of camera stitching is shown in Figure 6.4.3

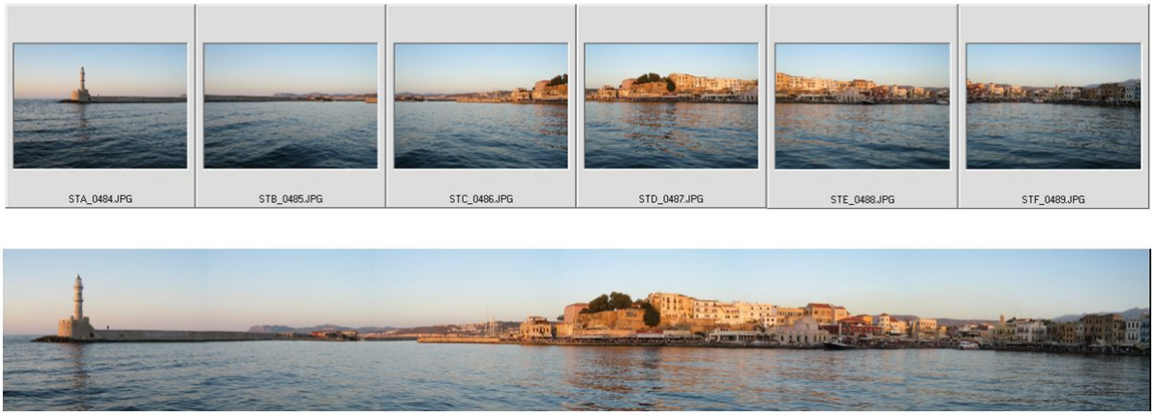


Figure 6.4.3: Panorama stitching [19]

- Stereo vision: In stereo vision, two cameras are used to obtain 3D information of the imaged scene. If we have two images of the same scene taken from a slightly different position, points of the scene will be displaced a tiny bit in both images. The displacement, also called disparity, is an indication of the depth of the point. The larger the displacement, the closer the point to the camera. To find the disparity between two points, corresponding points need to be found in both images. This is done using feature matching. An example is shown in Figure 6.4.4.

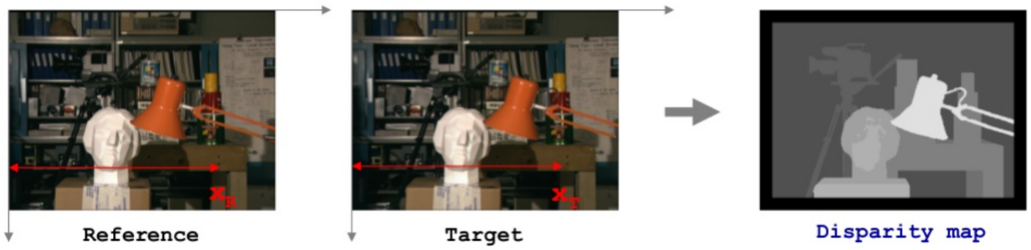


Figure 6.4.4: Stereo vision [19]

6.4.2 Feature matching algorithms

For feature matching, several algorithms exist. We will cover Brute-force matching and FLANN-based matching.

6.4.2.1 Brute-force matching

The most simple feature matcher is Brute-force matching. This method simply calculates the Euclidean distance between a feature vector in the first image and all feature vectors in the second image. From the second image it selects the feature vector that is the closest to the feature vector of the first image based on this Euclidean distance. This is done for all the descriptors of the first image. Brute-force matching is applied to the book cover example shown in Figure 6.4.5.

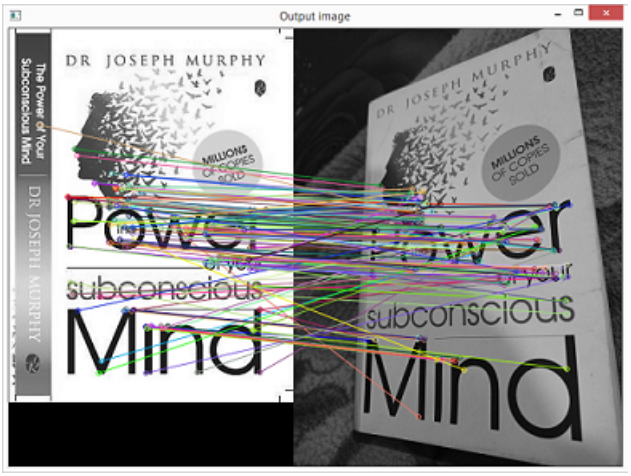


Figure 6.4.5: Brute-force matcher: 178 matches [26]

The brute-force matcher has 178 matches as a result. These are ordered according to the quality of the match. Due to the insufficient visibility in Figure 6.4.5, the top 15 matches are chosen and visualised in Figure 6.4.6.

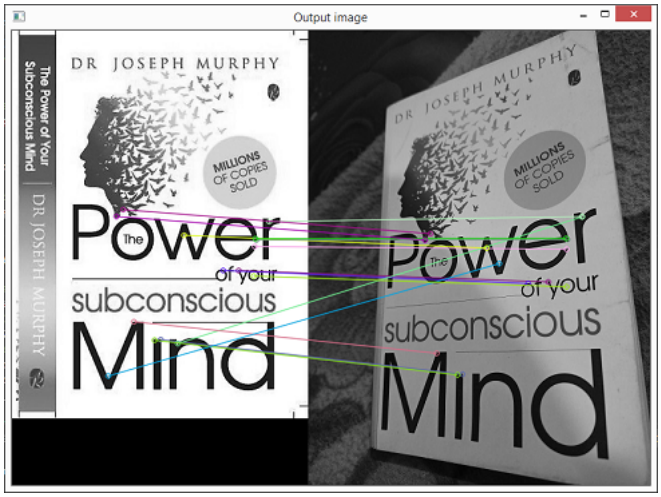


Figure 6.4.6: Brute-force matcher: top 15 matches [26]

6.4.2.2 FLANN based matching

FLANN is the abbreviation of Fast Library for Approximate Nearest Neighbours. This library contains multiple algorithms optimised for nearest neighbour search in large data sets and for high dimensional features. This method of feature matching is much faster than brute-force matching. This is not the only difference between the two. While a brute-force matcher tries to find the best possible matches, the FLANN based matcher searches for good matching candidates and therefore not directly the best possible matches. The quality of matching is optmised by choosing the best FLANN parameters. FLANN based matching applied to a book cover is illustrated in Figure 6.4.7.

Figure 6.4.7: FLANN based matcher [27]

6.5 Features in object recognition

The previous section described feature matching as a possible usage of features. Another very important usage of features arises in object recognition. In object recognition it is all about finding objects in an image and describing these objects. To do so, it is crucial to be able to compare and distinguish objects from each other. Therefore, features play an important role. Let's look at the simple example of detecting cats and dogs in an image. This is a classification problem in which we have two classes 'cat' and 'dog'. So if we get an image of a cat or a dog, we need to be able to define the image as either 'cat' or 'dog'. As mentioned in the first chapter, we as humans can easily do this classification but we have difficulties in defining how we exactly distinguish cats and dogs in order to perform the classification. And thus we have difficulties in telling a computer how to classify the images. But now with the knowledge of features, we have a notion of how this can be done, disregarded if we as humans understand what these detected object features exactly mean. Because for every image, we could extract the features and see that the features of 'cat' images are different from features of 'dog' images. A simple classification algorithm could then be developed that classifies images, based on the features extracted from it. This process is visualised in Figure 6.5.1. Extracting features and using them to classify objects is the basis of all simple and highly complex object recognition techniques. These techniques are described in the next chapter.

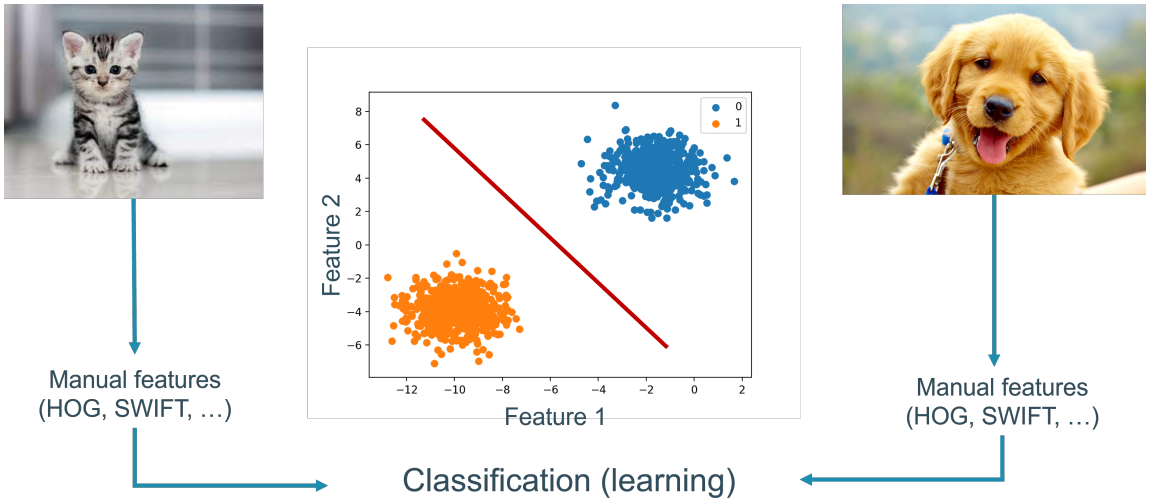


Figure 6.5.1: Features for object recognition

Chapter 7

Object recognition

In many applications, recognition of one or multiple objects in an image or video is required. This may involve recognising people, animals, mechanical defects, presence of a component, etc. The goal is to teach a machine to understand the content of images just like humans do. Object recognition can be divided into two parts: (i) object detection: this is the task of finding an object (person, cat, dog, etc.) in an image and locating it with, for example, a bounding box. (ii) object classification: this is the task of defining what the object is by assigning it to a certain class (person, cat, dog, etc.). As stated before, all object recognition algorithms are based on two main parts: (i) feature extraction algorithms that extract useful features from an image, and (ii) detection or classification algorithms that classify or detect objects based on these features. And based on the nature of the feature extraction algorithm, object recognition algorithms can be divided into object recognition using non data-driven techniques and data-driven techniques. And the object recognition algorithms that use data-driven techniques can be further divided into Machine Learning-based techniques and Deep Learning-based techniques. The type of technique is highly dependent on the type of object that needs to be recognised. Simple objects such as lines or geometric shapes can be detected using non data-driven techniques. More complex objects that show low variance or are imaged in a controlled environment can be detected using Machine Learning techniques. And very complex objects with high variation and imaged in different environments can be detected using Deep Learning techniques. All three techniques will be described in the next sections.

7.1 Non data-driven techniques

When talking about non data-driven and data-driven techniques, we refer to whether or not an algorithm learns features and/or learns a classifier or detector from data. In machine vision applications this is of course image or video data. Non data-driven techniques do not learn from vast amounts of data but rely on hard-coded methods that are programmed by humans. These hard-coded methods rely on so called 'magic numbers' which are algorithm parameters that need to be set to obtain a certain result. Let say that we want to classify squares and triangles. One approach to do this is to threshold the square or triangle from the background, applying the Canny Edge Detection algorithm, and counting the number of edges. Based on the counted number, a classification between 'square' or 'triangle' can be made. So first, features (number of lines) are extracted from the image data, and a classification is made based on these features. These methods all use parameters to be set such as the threshold value to threshold the image. If we would have illuminated the objects with LED light and tuned the thresholding parameter to obtain good results, we would probably have a failing algorithm when the LED lights are turned off because the colours in the image change and the threshold parameter does not hold anymore. We could adapt the parameter to provide good results in several situations, but if we are dealing with lots of changes (high variation) in illumination, positioning, object sizes, this becomes a tedious process. Therefore, non data-driven methods are mainly used to detect simple objects such as lines, circles, or other mathematical shapes, in controlled environments with low variation. Two of these techniques, Hough Transforms and Template Matching, are described in the next sections.

7.1.1 Contour detection

Contour detection is a very simple form of object detection where an algorithm searches for blob edges that form a closed curve in a binary image. The shapes does not need to be mathematical shapes. Typically, the algorithm also returns some properties of the contours such as centre coordinate, surface area, aspect ratio, etc. Contours can be used to count objects, measure objects, categorise objects based on their shapes, etc. The results of contour detection are visible in Figure 7.1.1.

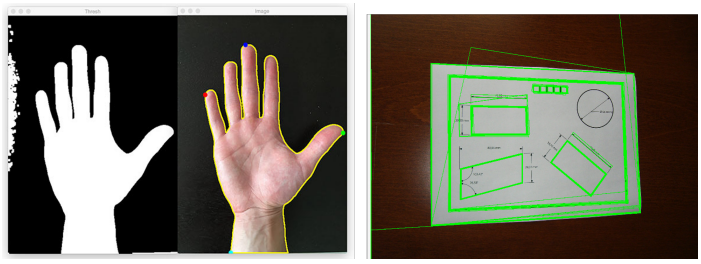


Figure 7.1.1: Contour detection [19]

7.1.2 Hough transformations

The Hough transformation is a method that can easily detect mathematical shapes such as lines and circles. This method needs a binary image, which can be the result of a thresholding or Canny edge operation, as input. To better understand the principle of a Hough transformation, a simple example is provided in the form of line detection. As shown in Figure 7.1.2, some data points are aligned in the image space. A line can be presented by the following formula:

$$v = mu + c$$

When looking at data point 1, the u and v values are known. For certain m and c parameter values, the line formula will be satisfied. These parameter values can be represented as a line in the parameter space. When transforming the remaining data points to the parameter space, this will result in multiple lines with different angles which all intersect in one point. That point in the parameter space is represented by the line, where all the data points are a part of, in the image space. So, a point in the image space will be transformed to a line in the parameter space, while a line in the image space will be transformed to a point in the parameter space [59].

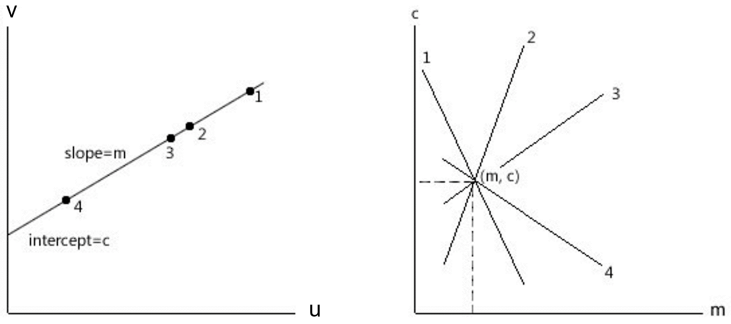


Figure 7.1.2: Hough transform: image space to mc parameter space [28]

A problem with choosing m and c as parameters is that m can have infinite values. This will lead to a lot of available memory and computational power requirements. Therefore, a better parameterisation can be found in choosing the ρ and θ parameters. The benefits of using these parameters are the finite values of ρ and θ. The formula of a line using these new parameters is:

$$u\ sin\theta + v\ cos\theta + \rho = 0$$

A point in the image space will now be presented as a sinusoidal wave in the new parameter space. As shown in Figure 7.1.3, the intersection between 2 sinusoidal waves can now be presented as a line in the image space.

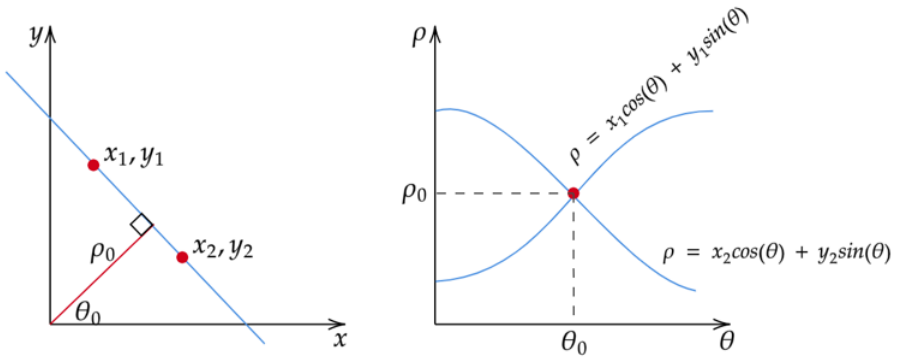


Figure 7.1.3: Hough transform: image space to $\rho\theta$ parameter space [29]

The detection of lines in images is based on the values of an accumulator. An accumulator is a discretized 2D grid where the row values represent ρ values and the columns represent θ values. An example of an accumulator is shown in Figure 7.1.4. The process of detecting lines starts with initialising the values of the accumulator to zero. Then, for each data point (u, v) in the binary image all the different angles $\theta = 0, 1, ..., 180$ are put into the line formula to obtain the corresponding ρ values. The corresponding values of the (ρ, θ) pairs are incremented by one. When this is done for all the data points, the detected lines are obtained by searching for the maxima of the accumulator. Note that these maxima can also be defined by a specific threshold value.

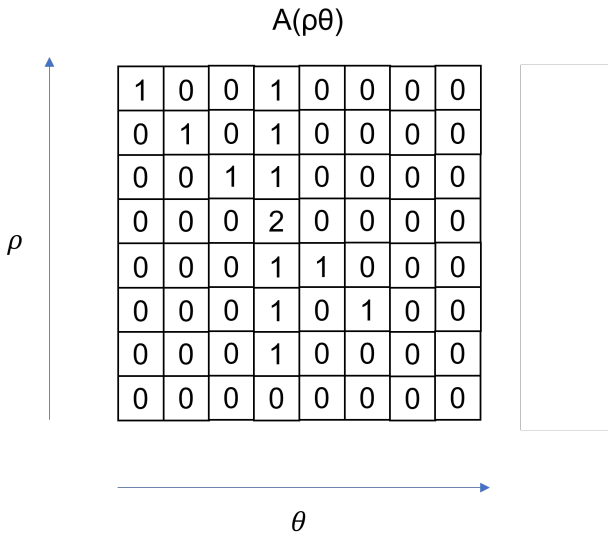


Figure 7.1.4: Hough transform: accumulator

An example of performing a Hough transformation for line detection is shown in Figure 7.1.5. The Hough transform of the binary image consists of multiple sinusoidal waves. Two distinct intersection points can be observed. These points correspond to the detected lines in the output image. Hough transforms can also be used to detect circles.

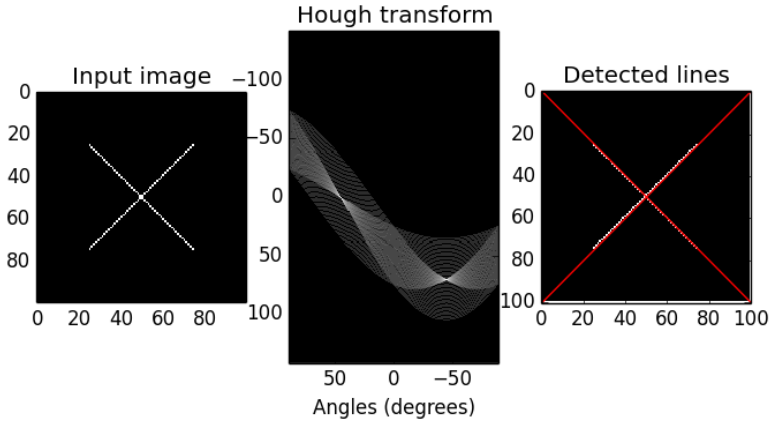


Figure 7.1.5: Hough transform example [30]

7.1.3 Template matching

Template matching uses (a part of) an image as kernel, called a template. It tries to find the most similar regions of the image in relation to the template [1].

$$\forall\ u, v \in \mathbf{I} : \mathbf{O}[u, v] = s(T, \mathcal{W})$$

T is a $w \times w$ window representing the template, while $\mathcal{W}$ is the $w \times w$ window centred around the pixel with coordinates (u, v) of the image. A schematic representation of template matching is shown in Figure 7.1.6.

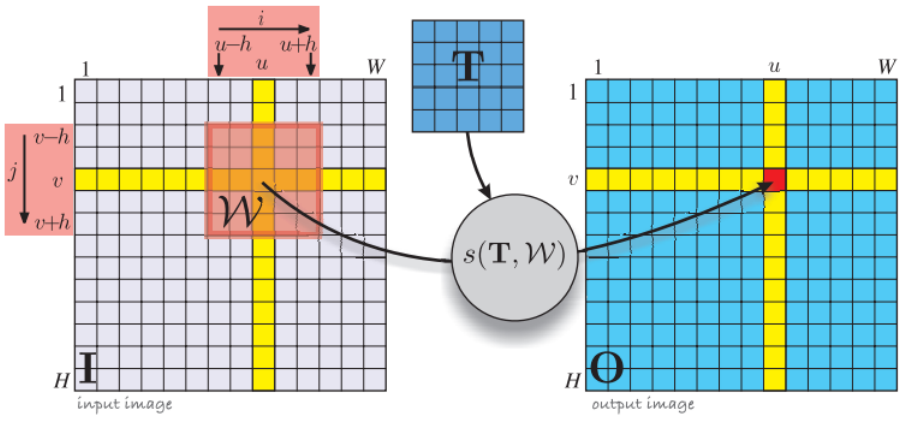


Figure 7.1.6: Template matching algorithm [1]

The similarity between the images is determined by the similarity function s. The most common similarity functions are discussed next.

- **Sum of absolute differences (SAD)**: SAD calculates the sum of the absolute difference value between the corresponding pixels of image I_1 and I_2. When the two images are equal, the SAD will be zero. So the lower

the SAD value, the more similar the compared images.

$$s(I_1, I_2) = \sum_{(u,v)\in I_1} |I_1[u,v] - I_2[u,v]|$$

- **Sum of squared differences (SSD)**: The idea of SSD closely resembles that of SAD. The more the SSD value increases, the more dissimilar the images are.

$$s(I_1, I_2) = \sum_{(u,v)\in I_1} (I_1[u,v] - I_2[u,v])^2$$

- **Normalized cross correlation (NCC)**: NCC gives a value in the -1 to 1 range. Similar images will return a 1 value. In practice, images with a NCC score bigger than 0.8 will be considered as similar. The benefit of using NCC over SAD and SSD is that it is invariant to intensity changes.

$$s(I_1, I_2) = \frac{\sum_{(u,v)\in I_1} I_1[u,v] I_2[u,v]}{\sqrt{\sum_{(u,v)\in I_1} I_1^2[u,v] \sum_{(u,v)\in I_1} I_2^2[u,v]}}$$

The problem with template matching is that it is not invariant to scaling. The template has to have the same dimensions as the parts in the way they appear in the image. The orientation of this template is also important. In Figure 7.1.7 a template of a round object and the correlation map is shown. There is a peak present on pixel (42, 54). This is the upper left corner of the box bounding the round object.

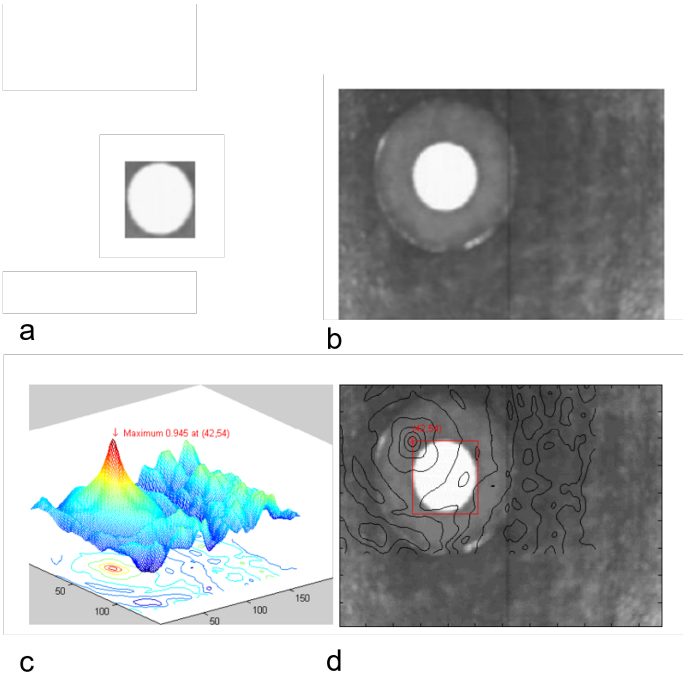


Figure 7.1.7: Template matching example: a) template, b) source image, c) correlation map with peak at (42, 54), d) bounding box on object with the highest correlation [22]

7.2 Data-driven techniques

We saw that simple non data-driven object recognition techniques fall short in situations where there is a lot of variation in the images. When more complex situations get imaged, and when the objects to detect are more complex and show high variation, data-driven techniques are recommended. Two main divisions can be mentioned: Machine Learning-based methods and Deep Learning-based methods. Both will be described in the next sections after having a section on the real challenges in object recognition.

7.2.1 Challenges of object recognition

There are a lot of challenges, which complicate the recognition of objects, and which cause the need for data-driven methods. The most important ones will be discussed below.

1. Variable circumstances:
   - Camera position: the position of the camera has an influence of the orientation of the objects on the image.
   - Illumination: different illumination in an image can make it difficult to detect the desired object.
   - Shape: for some non data-driven detection methods, such as template matching, is the changing of the object shape detrimental to the detection of it.
2. Occlusion: occlusion within an image occurs when 2 objects come too close and merge or combine with each other. This can make the detection of these objects hard [60].

Figure 7.2.1: Occlusion

3. Infinite object categories
4. Intra-class variation: within a class, there are a lot of variations of the objects. For example the class "chair" consists of many types: armchair, folding chair, ladderback chair, etc.

Figure 7.2.2: Intra-class variation within the "chair" class [31]

7.2.2 Machine Learning-based techniques

We saw that the non data-driven methods use hard-coded classical features (number of lines) and hard-coded classification algorithms (4 lines = square, 3 lines = triangle). Instead of being hard coded, in Machine Learning-based methods, the classification algorithm will be learned from data. What this exactly means will be described in the next section. This section is followed by two popular Machine Learning-based object detectors.

7.2.2.1 Image classification

An introduction to this section is already made in Section 6.5 where the usage of features in object detection was described. We stated that in order to classify cats from dogs, we could have a vast number of images of cats and dogs, extract features (SIFT, SURF, HOG, etc.) from all images, and classify the cats and dogs based on these features. A classifier therefore needs to be 'trained' in order to learn from the seen data and to be able to classify the classes. Here we assume that in the feature vector space, the features of both classes can be easily separated. As the features we extract are representations that are defined by humans, we refer to them as being manual features. The classification process is shown in Figure 7.2.3. To be noted is that classification is referred to as supervised learning. This means that the ground truth of the images is known and labelled by humans. The database thus contains of images from cats and dogs where for each image it is kwown if it is a cat or a dog.

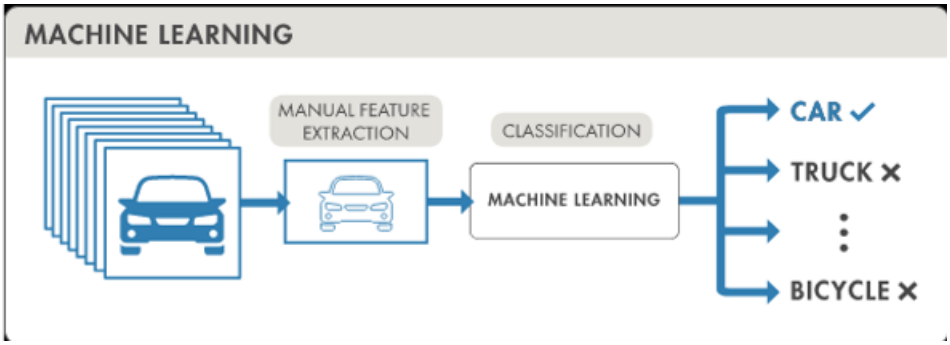


Figure 7.2.3: The classification process in Machine Learning-based methods [32]

.

To go a little bit deeper in the word 'training' we will built further on the binary classification problem of cats and dogs. Assume that we extract feature vectors from the database and that we only show two of the feature values and plot these for every image in the dataset. The result is visible in Figure 7.2.4. Here we can clearly see two clusters each representing either cats or dogs. However, it needs to be mentioned that in reality, the data will not be that clean and a possible separation line (the classifier) will not be identified so easily as a straight line. But for this easy example the data is thus clearly separable using a line model classifier which we can model as:

$$y = ax_1 + bx_2 + c$$

For every data point (and thus image), we can obtain the features x_1 and x_2 and can calculate the model output y. If $y > 0.5$ the image gets classified as 'cat', and if $y < 0.5$ the image gets classified as 'dog'. A learning or training procedure then boils down to finding the model parameters a, b, c in order for the classifier to separate the two classes

as good as possible. The learning process starts with an arbitrarily set of parameters and calculates how good the separation is. If it is good, the parameters are kept, otherwise the parameters are modified in order to improve the goodness of separation. This is basically how every classification algorithm works. Some simple Machine Learning classifiers can be used such as Linear Regression or Nonlinear regression. But for more complex datasets such as visible in Figure 7.2.5, more complex algorithms are needed such as Support Vector Machines (SVM) and Decision Trees.

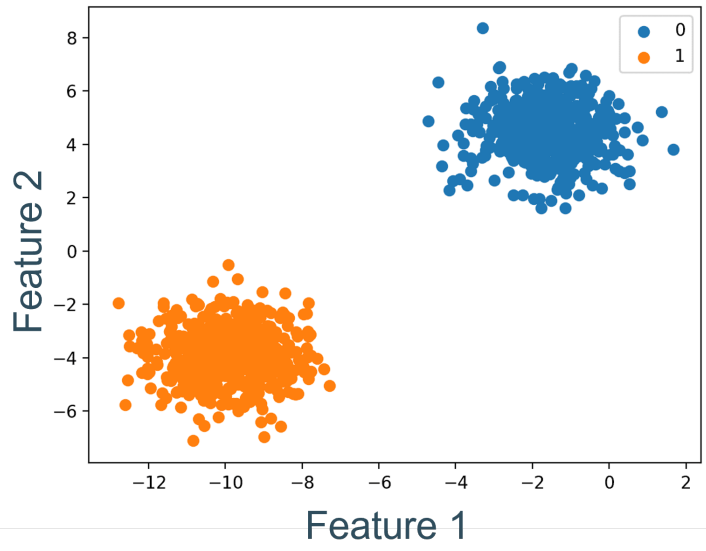


Figure 7.2.4: Plot of two features extracted from a 'cats' and 'dogs' dataset

.

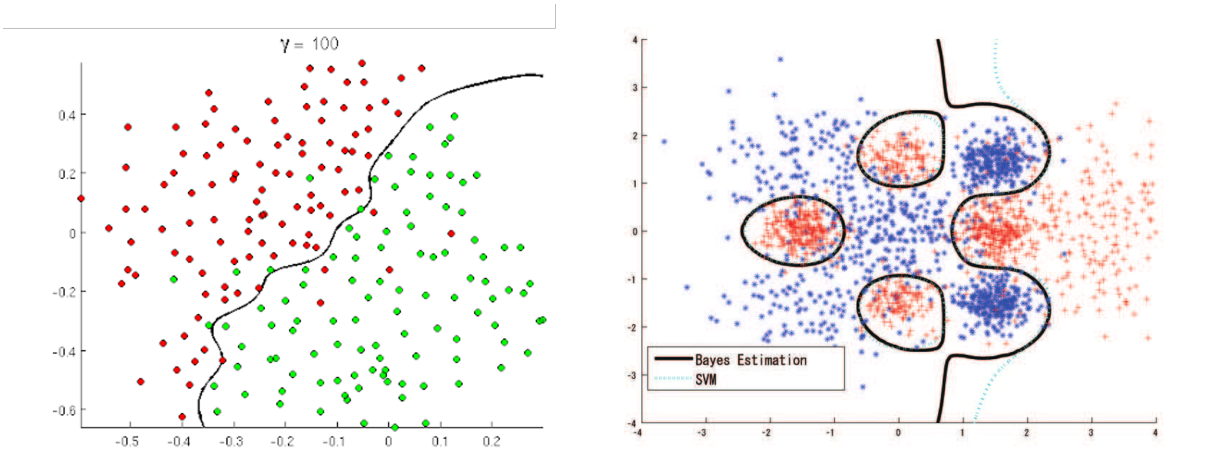


Figure 7.2.5: More complex datasets

.

7.2.2.2 Bag of (Visual) Words

In the previous section we talked about classification using manual features like SIFT, SURF, ORB, etc. However, we see that classification using these features is able to classify fairly simple objects but is not successful in classifying more complex objects in changing circumstances. Another attempt to provide more successful classification

algorithms is made using the Bag of (Visual) Words algorithm that makes use of even more complex features in order to model the complex world. It uses a combination of SIFT, SURF, ORB, etc. features in order to obtain a more complex feature representation. Bag of Words is a common concept in the field of information retrieval and text analysis to compare documents. In written documents, the context, meaning, and connotation can easily be described by only a set of keywords from this document. The Bag of Words concept summarises a document using a set of keywords and uses this set to compare the document to other documents and thus to classify. A document with keywords 'brain, perception, cell' is fundamentally different from a document with keywords 'China, trade, imports'. This disregarding the order the keywords appear in the document. A visualisation is made in Figure 7.2.6. In vision, these 'words' are complex features that are derived from more simpler features as seen in Figure 7.2.7.

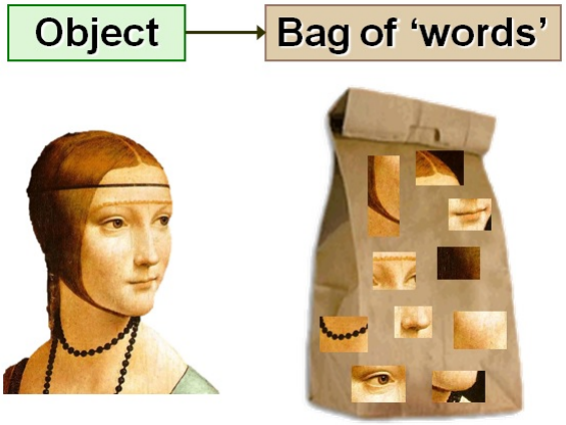


Figure 7.2.6: Bag of words [33]

.

Figure 7.2.7: Bag of visual words [33]

.

When we want to perform image classification with a number of classes, we need a dataset with images of each class. From each image, simple features are extracted. Using a clustering algorithm, similar features are combined in clusters to represent more complex features that represent visual words. Finally, every image in the dataset is encoded in a histogram of visual words, which is now the more complex feature vector representing the image. This process is visible in Figure 7.2.8.

Figure 7.2.8: Generating more complex feature vectors [33]

.

Now that each image in the dataset is represented by a new and more complex feature vector, the classical classification algorithms can be used to classify based on these new feature vectors. For any new image, the simple features can be extracted, combined into a histogram, and being compared to the labelled images in the dataset to be classified.

7.2.2.3 HAAR-Cascades

In Section 6.3.1, we saw the HAAR-wavelet as a possible feature to describe object in an image. If a particular HAAR-wavelet is convoluted over an image patch and returns a result that is higher than the threshold belonging to that HAAR-wavelet, than the feature that is represented by the HAAR-wavelet is present in that image patch. These HAAR-wavelts are used in an image classification algorithm called the Viola-Jones HAAR cascades. In this approach, a set of HAAR-wavelets that is able to perform a good classification on a dataset is learned from the data. The algorithm is most common for the detection of faces. For the detection of faces, the dataset contains a large number of images with a face (positive images), and images without a face (negative images). This is also a supervised learning algorithm as the dataset is labelled (images containing a face are labelled as positive, and images that do not contain a face are labelled as negative). The training procedure involves convolving all the images in the dataset with thousands of different HAAR-wavelets and corresponding thresholds. One HAAR-wavelet being convolved over an image is seen in Figure 7.2.9. The red window contains a HAAR-wavelet at a specific position within that red window and slides over the whole image. It performs the typical HAAR computation as seen in Section 6.3.1. If a HAAR-wavelet outputs a result higher than the threshold (and thus classifies the image patch bounded by the red window as positive), and the image patch also contains a face (is a positive image), then this wavelet is considered to be a good classifier for detecting images and is added to the set of good wavelets. If a HAAR-wavelet outputs a result lower than the threshold (and thus classifies the image patch bounded by the red window as negative), and the image does contain a face, then this wavelet is considered to be a bad classifier for detecting images and is discarded. In this way the learning algorithm generates a set of HAAR-wavelets that manage to properly classify an image patch as including a face of including no face. If the algorithm is shown a new image it has never seen before, it convolves the set of good HAAR-wavelets over the image and classifies it as positive if the majority of wavelets outputs a result higher than the corresponding thresholds.

Figure 7.2.9: HAAR cascades [34]

.

7.2.3 Deep Learning-based techniques

Different from the non data-driven methods, the Machine Learning methods use a classifier that learns from a large dataset and used manual features to learn from. However, these methods still face strong limitations when being applied to images with very large variations and very complex objects. This is because there is a limitation in how good the human designed features can describe an object. And this can directly be linked to the first chapter of this book where we stated that it is very straightforward for a human to describe what's inside an image, yet it is very difficult to describe why we see something in an image, or said differently, what features extract to describe what's inside an image. This is basically a limitation in the human abilities, we just cannot design the proper features that allow us to detect anything in any image. And this is where Deep Learning comes into play. Deep Learning uses neural networks to automate both the classification and the feature extraction process. Based on data, these methods do not only learn a classifier based on data, but also learn what features are needed to perform a good classification. We can speak from a paradigm shift in the computer vision domain in which human defined features get replaced by features that are automatically learned by a computer program. This paradigm shift is denoted in Figure 7.2.10. The next sections will cover the basics of neural networks and will go deeper in some Deep Learning based classification, detection, and segmentation models.

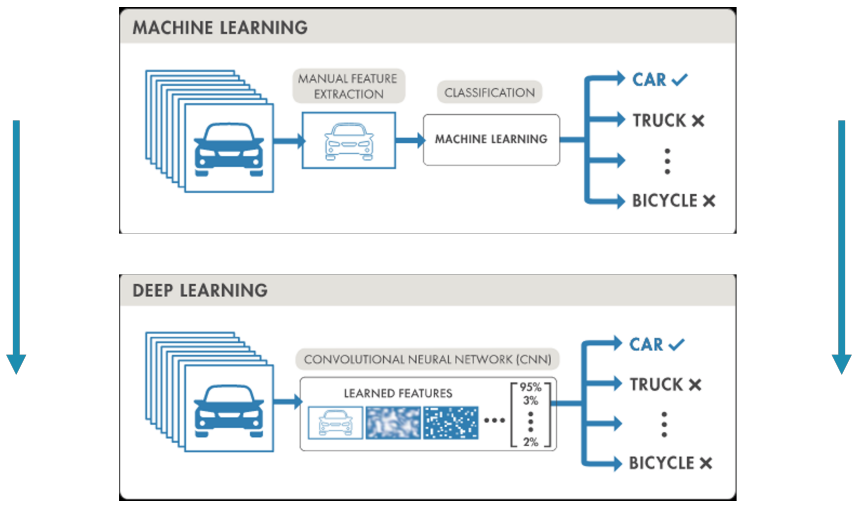


Figure 7.2.10: Paradigm shift [32]

.

7.2.3.1 Neural networks

Neural networks are powerful computing systems, whose architecture and functioning is based on the human brain. These can be used for many purposes in many domains. In this section, a progressive buildup of a neural network will be provided. This will help in intuitively understanding the functioning of such a network.

Perceptron A perceptron, also called a neuron, is the building block of a neural network and is modelled after the neuron of a human brain. This model is also called the McCulloch-Pitts neuron. As shown in Figure 7.2.11, the neuron structure consists of 3 important elements: the dendrites, the axon and the axon terminals, also called the synapses. The dendrites of the neuron collect input signals coming from other neurons or sense organs. This neural signal is transmitted through the axon and send to other receivers through the synapses [61].

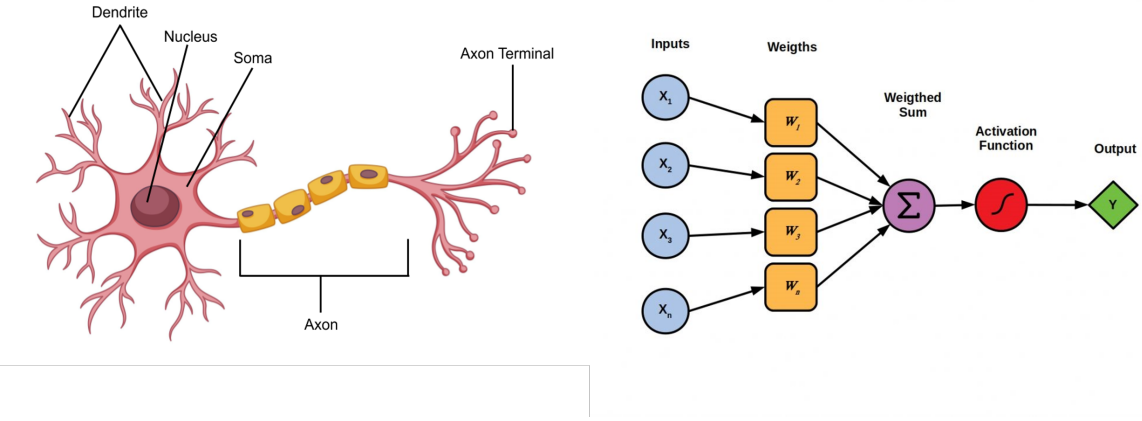


Figure 7.2.11: A neuron of the human brain vs a perceptron of a neural network [35, 36]

The perceptron shows a similar structure. The perceptron collects input signals, which could be pixel values. These input values are processed by doing 2 operations:

- **Linear operation**: a weighted sum of the input signals is calculated. The results is called the activation a. Here, each connection between neurons is annotated by a weight.

$$a = \sum_{i}^{n} w_i \times x_i + b$$

  Here, w_i is the weight corresponding to the input x_i and b is the bias.

- **Nonlinear operation**: an activation function f is performed on the activation a. This function introduces a non-linearity in the neural network. The most common activation functions are the Sigmoid, Tanh and ReLu functions.

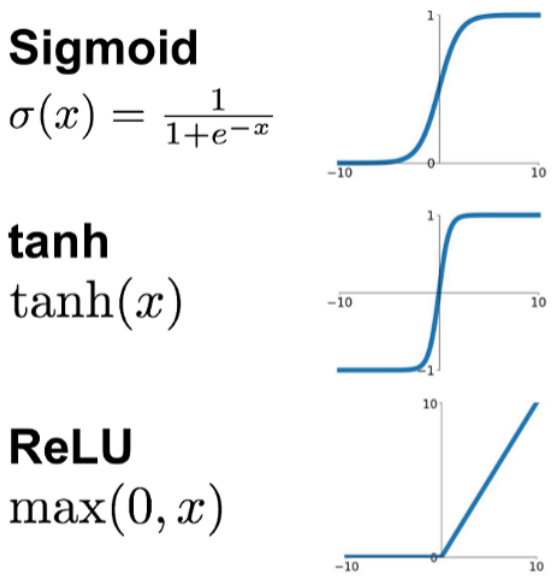


Figure 7.2.12: Different types of activation functions: Sigmoid, Tanh and ReLu functions [37]

Single-layer perceptron When connecting all inputs to multiple neurons, a single-layer perceptron is obtained. In Figure 7.2.13, such a single-layer perceptron is shown. Here, it is observable that only one layer of neurons, namely the output neurons, are present. When counting the layers in neural networks, the input layer is never taken into account. A special type of layer is a fully connected layer. In this layer, every input is connected to the neuron and this for every neuron of that layer. Simple classification can be completed with such single-layer perceptrons.

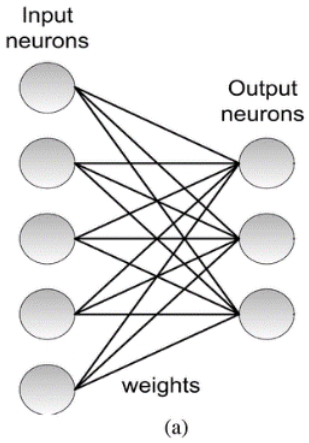


Figure 7.2.13: Single-layer perceptron [38]

Multi-layer perceptron When multiple layers of neurons are present, a multi-layer perceptron is obtained. In Figure 7.2.14, such a multi-layer perceptron is shown. The layers between the input and output layers are called hidden layers. Here, the multi-layer perceptron only has 1 hidden layer. Expanding a neural network by adding more layers results in a more complex neural networks. Training such a network is harder and more time-consuming. For example, the multi-layer perceptron in Figure 7.2.14 has 26 weights (21 interconnection weights and 5 bias weights) to train:

- Weights for the first layer: $5 \times 3 + 3 = 18$

- Weights for the second layer: $3 \times 2 + 2 = 8$
- Total weights: $18 + 8 = 26$

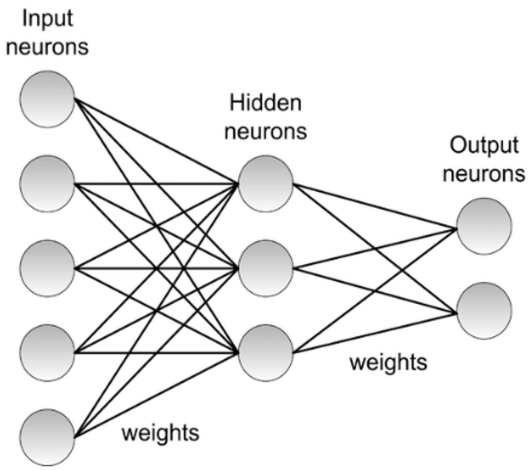


Figure 7.2.14: Multi-layer perceptron [38]

Deep neural networks Deep neural networks are complex neural networks which contain 2 or more hidden layers. The addition of more hidden layers has a beneficial effect on the accuracy, but decreases the explainability of the neural network. Deep neural networks can also perform more complex tasks. As already shown in the previous section, adding sequential layers to the network increases the amount of weights tremendously [62]. Training a neural network means learning the weights of that neural network. Here, the obtained data set is divided in 3 sets: the training set, the validation set and the test set. The training set is used to train the neural network on. The validation set, in turn, is used to decide when overfitting is present. Overfitting is the event that occurs when the neural network learns the details of the training set too well. This results in a too complex networks, which is not very well generalised. This is consistent with Occam's razor: when multiple models are suited to solve a problem, the simpler model is preferred. The validation set is used to measure the performance of the trained neural network. There are many types of neural network architectures. Feedforward neural networks are used in a wide variety of applications, where the data is non-sequential nor time-dependent [63]. On the other hand, recurrent neural networks are used in applications, where the data is sequential and time dependent. More in particular in natural language processing. But the focus here will be on convolutional neural networks. This architecture is specialised in vision applications.

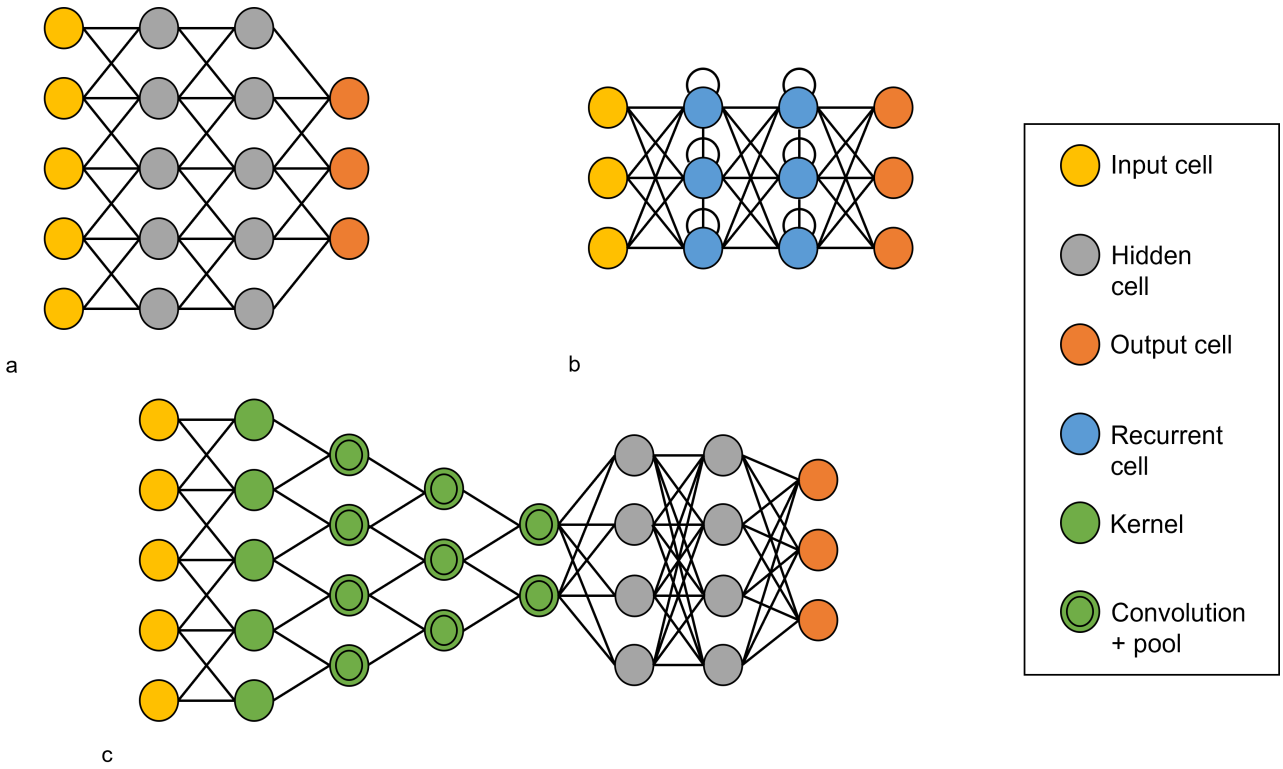


Figure 7.2.15: Neural network architectures: a) Feedforward neural network, b) Recurrent neural network and c) Convolutional neural network

Convolutional neural networks Convolutional neural networks are specialised in handling data with a grid like structure, such as images. When using images in feedforward neural networks, these data can be presented by traversing the image row per row and assigning each pixel to an input cell. An example of a 6×6 binary image is shown in Figure 7.2.16.

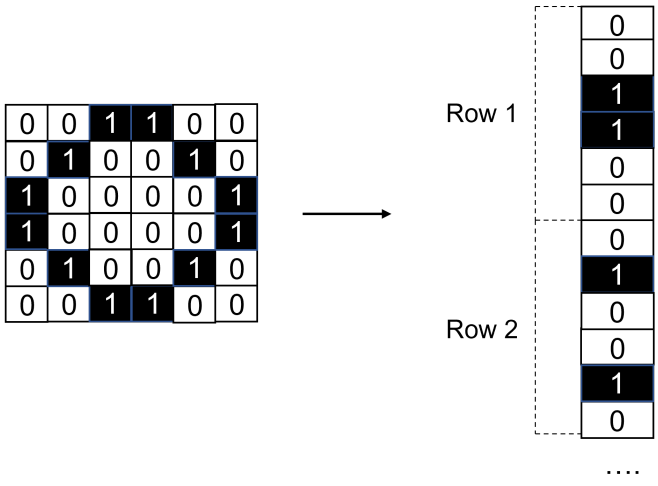


Figure 7.2.16: A 6×6 binary image presented as an input for a feedforward neural network

This means that per extra node in the first hidden layer, the amount of added weights is 36. But in practice using such small images does not occur. Take for example a 100×100 image. This would result in 10.000 extra weights per

first hidden layer node. It becomes clear that using a feedforward neural network for images is not scalable. For this reason, convolutional neural networks are more suited for working with images. Aside this benefit, these networks also take into consideration the correlation between pixels. This implies that a pixel of a specific colour is highly likely surrounded by other pixels of that colour. Each convolutional neural network is mainly constructed of 3 types of layers: the convolutional, the pooling and the fully connected layer [64] [65].

- **The convolutional layer** performs a dot product between 2 matrices. The first matrix being a filter or kernel and the second matrix being a part of the image. The filter consists of trainable parameters. Thus, training a convolutional neural network implies that these filter parameters are learned. The filter slides over the whole image's width and height resulting in a feature map. The sliding size of the kernel is called the stride. In Figure 7.2.17, an example of a convolution is shown. Convolution is a linear operation, but images are not linear at all. Therefore, after performing a convolution a non-linearity layer is placed. Some examples of non-linearity functions which are performed on the feature map are a Sigmoid, Tanh or ReLu function.

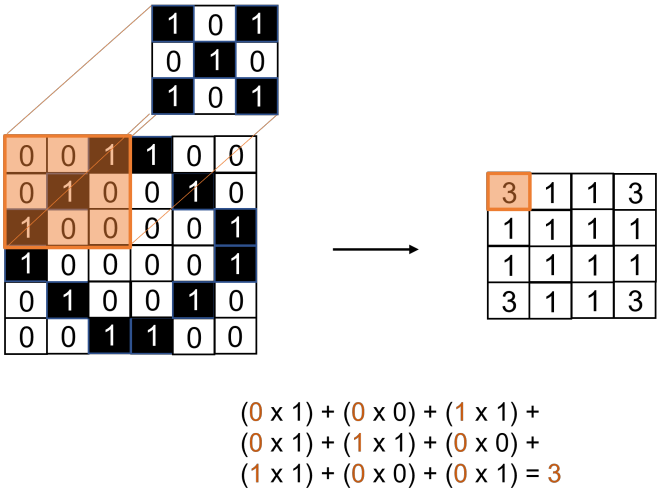


Figure 7.2.17: Convolution operation on a 6×6 binary image resulting in a feature map

- **The pooling layer** is conventionally placed behind the convolutional layer. The objective of the pooling operation is to reduce the amount of data that needs to be processed, while preserving the most important information from these features. Pooling does this by summarising the data from the feature map. There are different pooling functions, but the most important ones are max and mean pooling. Both are visualised in Figure 7.2.18. A 2×2 filter with a stride of 2 is applied in both cases.

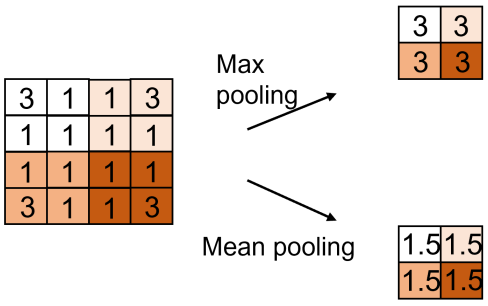


Figure 7.2.18: Pooling operation on a feature map: max and mean pooling

- **The fully connected layer** is always positioned before the output layer of the convolutional neural network. Full connectivity is present between all neurons of the previous and the current layer. This is done to carry out

the mathematical operations regarding the classification process. Therefore, using 2 fully connected layers has a better effect than using only one. At the end of the convolutional neural network, a softmax layer is placed. This softmax layer computes the probabilities that the input image belongs to a certain class. For binary image classification, the softmax layer only contains 2 neurons. While for multi-class classification with n classes, the layer consists of n neurons [39].

A general convolutional neural network is presented in Figure 7.2.19. All the different types of layers are present. This network also shows that each layer represents different features. The deeper the network is, the more detailed the features are.

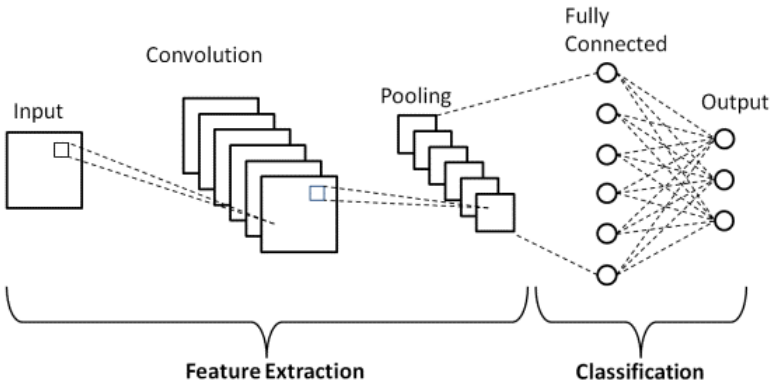


Figure 7.2.19: Convolutional neural network architecture [39]

Convolutional neural networks can be used for different purposes. In the interest of quality control, object detection and image segmentation are very important. These 2 subrelated object recognition tasks will be further discussed next.

7.2.3.2 Object detection

The logic behind object detection is two-fold. First, the algorithm localises every object within the image. The localisation is performed by placing a bounding box around every object. The second step is the classification of each object. Here, a label is assigned to each bounding box. In Figure 7.2.20, labelled bounding boxes are placed around both dogs and the cat. The bounding boxes are always rectangular. So, object detection does not give any information about the shape of the objects, nor about the area of the object in the image [40].

Figure 7.2.20: Object detection [40]

In the following paragraphs, some state-of-the-art object detection architectures are looked at in more detail. In addition, also some important open-source data sets for object detection will be discussed.

R-CNN , or region-based convolutional neural network, is a deep learning algorithm used to detect objects within an image. The architecture is shown in Figure 7.2.21. The algorithm starts with the selective search of 2000 regions of interest (ROI), called region proposals. These region proposals are, in turn, warped into squares and used as inputs for the CNN. The role of the CNN is to detect the features within the region proposal. The softmax layer is cut off, thus the CNN outputs a feature vector of 4096 elements. This feature vector is subsequently fed to a support vector machine (SVM), which classifies if the object is present within the region proposal, and a bounding box regressor, which is used to improve the bounding box precision [41].

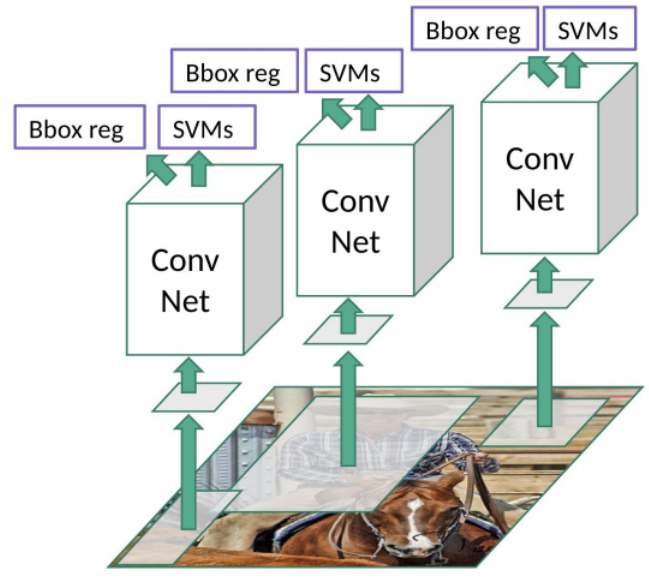


Figure 7.2.21: R-CNN architecture [41]

The most prominent problems with the R-CNN algorithm are:

- The high computation time ($\approx 1'/image$) due to the classification of 2000 region proposals. This problem makes it not possible to implement R-CNN in real-time applications.
- The performance is highly dependent of the fixed selective search algorithm. No learning occurs at this stage.

Fast R-CNN is an improved version of R-CNN, which aims to reduce the high computational time drastically. The Fast R-CNN architecture, which is presented in Figure 7.2.22, is similar to the R-CNN architecture but differs drastically at the input of the CNN. Instead of presenting all region proposals to the CNN, only the input image is fed to the CNN in Fast R-CNN. This results in a convolutional feature map from where the region proposals are identified and warped to a fixed square size. These are, in turn, offered to fully connected layers. A softmax layer is used instead of a SVM on the obtained ROI feature vector. This is done to predict the class of the region proposal, whether than to predict if the object is found within the region. The bbox regressor has the same functionality as in R-CNN [41].

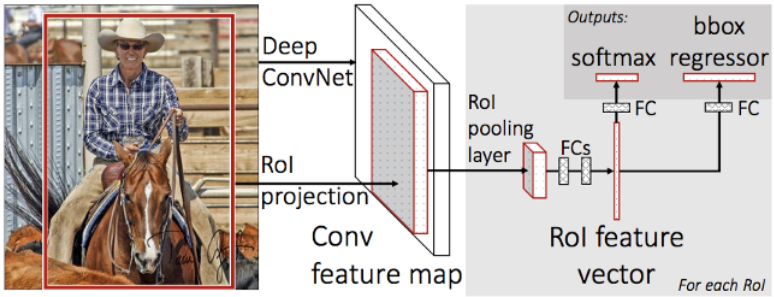


Figure 7.2.22: Fast R-CNN architecture [41]

The reason why Fast R-CNN lowers the computational time drastically is because all the 2000 region proposals do not have to pass the CNN. The CNN is only passed once by the whole image. Further tests signified that including region proposals is a huge bottleneck in the Fast R-CNN algorithm.

Faster R-CNN is an improvement over Fast R-CNN by removing the selective search algorithm which selects the region proposals. Instead, the network tries to learn these proposals. Similarly as in Fast R-CNN, the whole image is fed to the CNN. This results in a convolutional feature map, which is then used to learn the region proposals through the region proposal network. The remaining part of the algorithm is identical to that of Fast R-CNN. The architecture of this algorithm is displayed in Figure 7.2.23 [41].

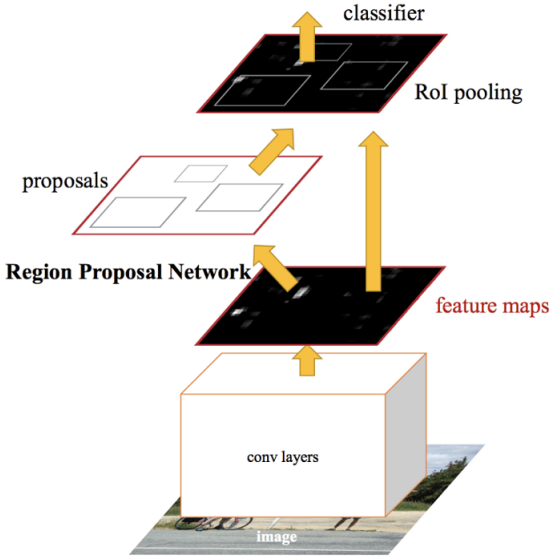


Figure 7.2.23: Faster R-CNN architecture [41]

The improvement in computational time is such that this algorithm can be used in real-time applications.

YOLO , the abbreviation of You Only Look Once, differs from the above discussed region based algorithms. This network does not use regions to localise objects. Instead, it looks at parts of the image which contain a high probability of that object. As shown in Figure 7.2.24, a single network determines the bounding boxes and the class probabilities. These results are combined to obtain the final object detections [41].

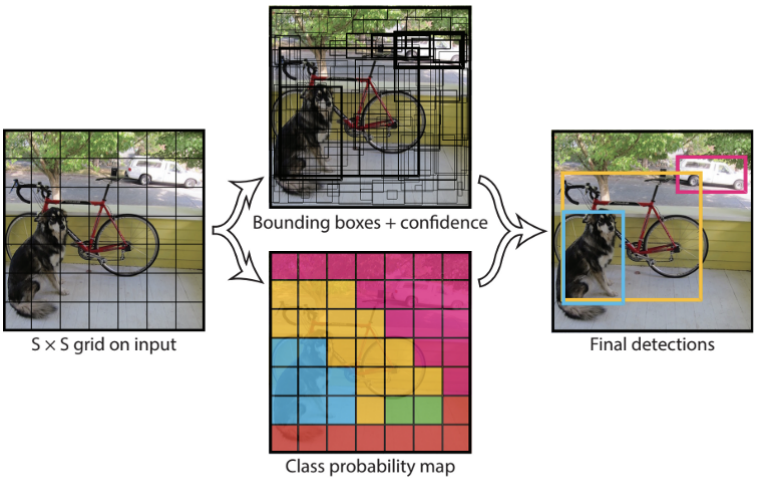


Figure 7.2.24: YOLO object detection [41]

YOLO is orders of magnitude faster than the region based algorithms. The only downside of using YOLO is that it has difficulties with detecting small objects. To end this list of object detection algorithms, a performance comparison of R-CNN, Fast R-CNN, Faster R-CNN and YOLO is provided in Figure 7.2.25. The performance is based on 2 metrics, namely the accuracy and the frames per second. The region based algorithms have a decent accuracy, but the amount of frames that can be processed per second is small. While, YOLO, by contrast, has an accuracy in the same range but the frames per second are definitely higher. A clear trade-off between accuracy and frames per second is observable in the successive versions of the YOLOv2 algorithm. This is obvious because achieving a higher accuracy is accompanied with more or longer computations, which is detrimental for the amount of frames per second that can be processed.

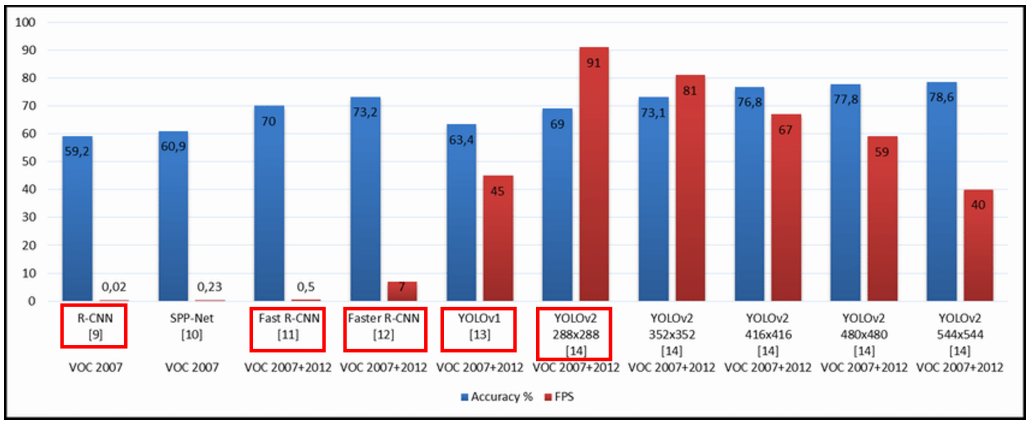


Figure 7.2.25: Object detection algorithms comparison: accuracy and frames per second [42]

Datasets

- **Open Images**: this data set from Google contains approximately 9 million images annotated with image-level labels and object bounding boxes [43].

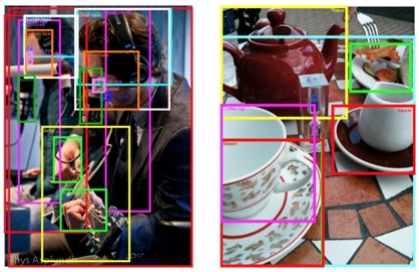


Figure 7.2.26: Open Images [43]

- **ImageNet**: this data set is organised according to the WordNet hierarchy. Each node of the hierarchy contains hundreds and thousands of images [44].

Figure 7.2.27: ImageNet [44]

- **Data set for Object deTection in Aerial images - DOTA**: this data set contains aerial images to do object detection [45].

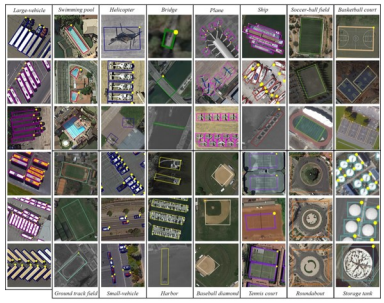


Figure 7.2.28: DOTA [45]

- **Exclusively Dark (ExDark) image data set**: this data set contains unique low-light images with 12 object categories [46].

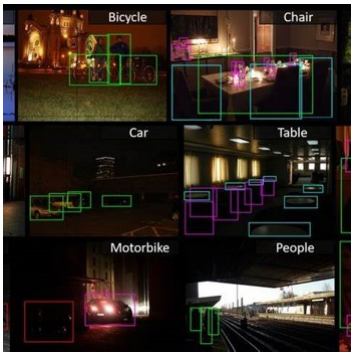


Figure 7.2.29: ExDark [46]

7.2.3.3 Image segmentation

Image segmentation subdivides the image in segments belonging to specific objects. This method is more fine-grained than object detection because it provides a class label for each pixel of the image. Therefore, information about the specific shape of the objects can be obtained. The result of image segmentation is a collection of pixel-wise masks for each object within the image. Two different types of image segmentation exist, namely semantic segmentation and instance segmentation. Semantic segmentation provides an identical colour representation for objects of the same class. Instance segmentation differs from semantic segmentation in that a different colour representation is provided for each object of the same class. The difference between the 2 types is shown in Figure 7.2.30 [40].

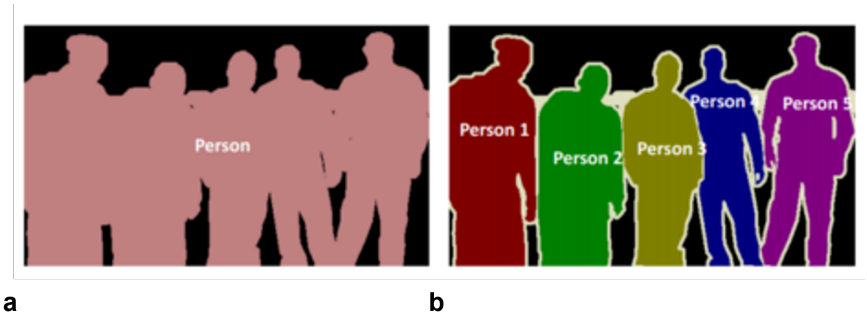


Figure 7.2.30: Image segmentation: a) semantic segmentation, b) instance segmentation [47]

An encoder-decoder architecture is the basic architecture which is used in image segmentation. The encoder part is responsible for feature extraction, while the decoder part is responsible for the mask generation. Next, the most commonly used architectures for both semantic as instance segmentation will be discussed. In addition, also some segmentation data sets, which can be used for training, will be presented.

U-Net is a fully convolutional neural network that is used for semantic segmentation.. Fully convolutional means that each fully connected layer of the standard CNN architecture is replaced by a convolutional layer. The left part of the U-Net, shown in Figure 7.2.31, takes care of the feature extraction of the image. The right part of the U-net uses transposed convolutions to obtain the segmentation map of the image. This architecture also uses skip connections. These type of connections skip multiple layers in the architecture to combine features from the encoder part with features from the decoder part. This results in a better localisation [48].

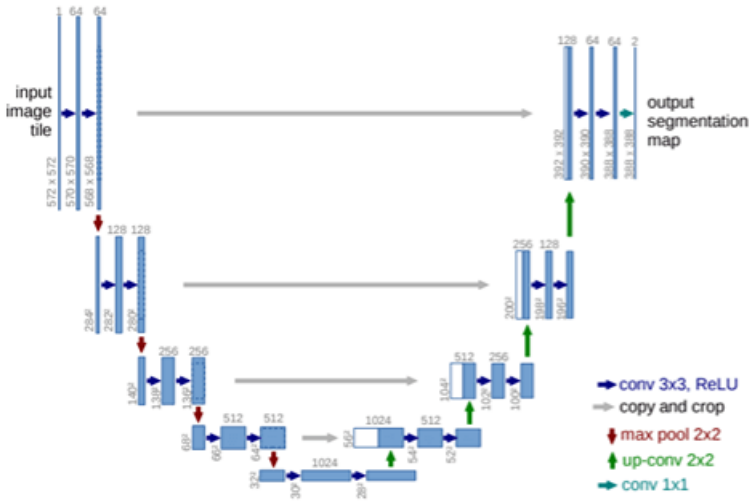


Figure 7.2.31: U-Net architecture [48]

Mask R-CNN is an architecture based on faster R-CNN that is used for instance segmentation. The faster R-CNN architecture is combined with the architecture of a fully convolutional network as can be seen in Figure 7.2.32. Mask R-CNN outputs a binary mask for each RoI. Another change that is present in the Mask R-CNN architecture is the use of RoIAlign instead of using RoIPool. RoIAlign is used because this reduces the loss of data due to misalignments introduced by RoIPool [18].

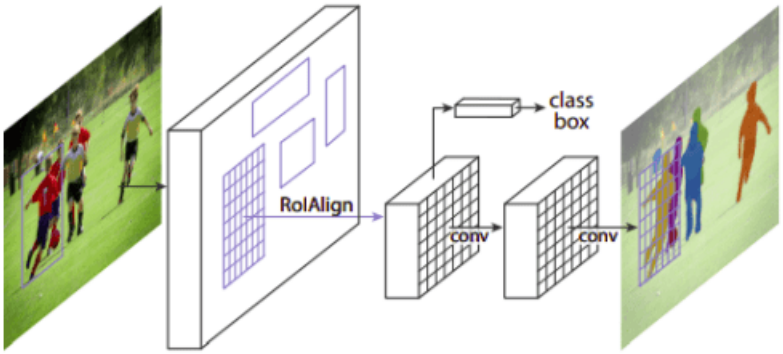


Figure 7.2.32: Mask R-CNN architecture [48]

Datasets

- **Common Objects in COntext - COCO data set**: this data set from Microsoft contains 328k images with 91 labels for instance segmentation [49].

Figure 7.2.33: COCO data set [49]

- **PASCAL Visual Object Classes - PASCAL VOC**: this data set contains 9963 images out of 20 classes [50].

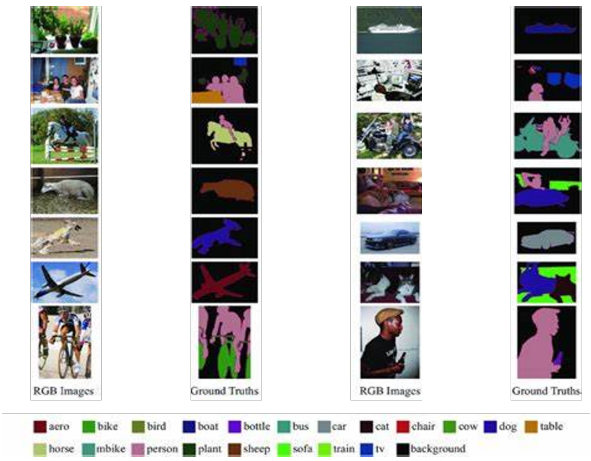


Figure 7.2.34: PASCAL VOC [50]

- **The Cityscapes data set**: this data set contains images of city scenes [51].

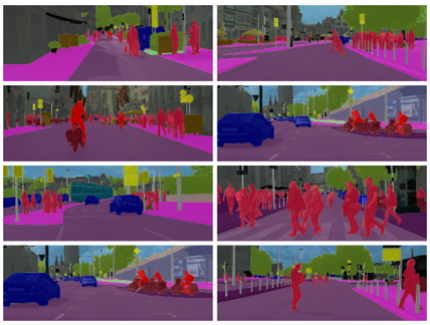


Figure 7.2.35: Cityscapes data set [51]

- **The Cambridge-driving Labeled Video database - CamVid**: this data set contains images from 32 semantic classes for motion-based segmentation [52].

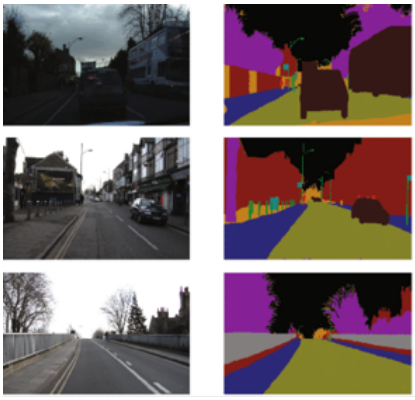


Figure 7.2.36: CamVid [52]

7.2.3.4 Transfer learning

In reality, it is very hard to train a neural network from scratch. This is due to several considerations: 1) insufficient computing power at its disposal, 2) a shortage of knowledge regarding building a neural network and 3) not having enough labeled training data. A solution to this problem comes in the form of transfer learning. Transfer learning retains the bulk of a network that is used for a different but related problem and is trained on a lot of labeled data. The classification layers of that neural network are removed and replaced with new layers appropriate for the new specific problem. This can be seen as a transfer of knowledge from one neural network to another because the weights are transferred. In vision applications, transfer learning consists of transferring the trained weights of the convolution layers and adding new fully connected layers. The small labeled data set is then used to train the weights of these new layers. This method is illustrated in Figure 7.2.37 [66].

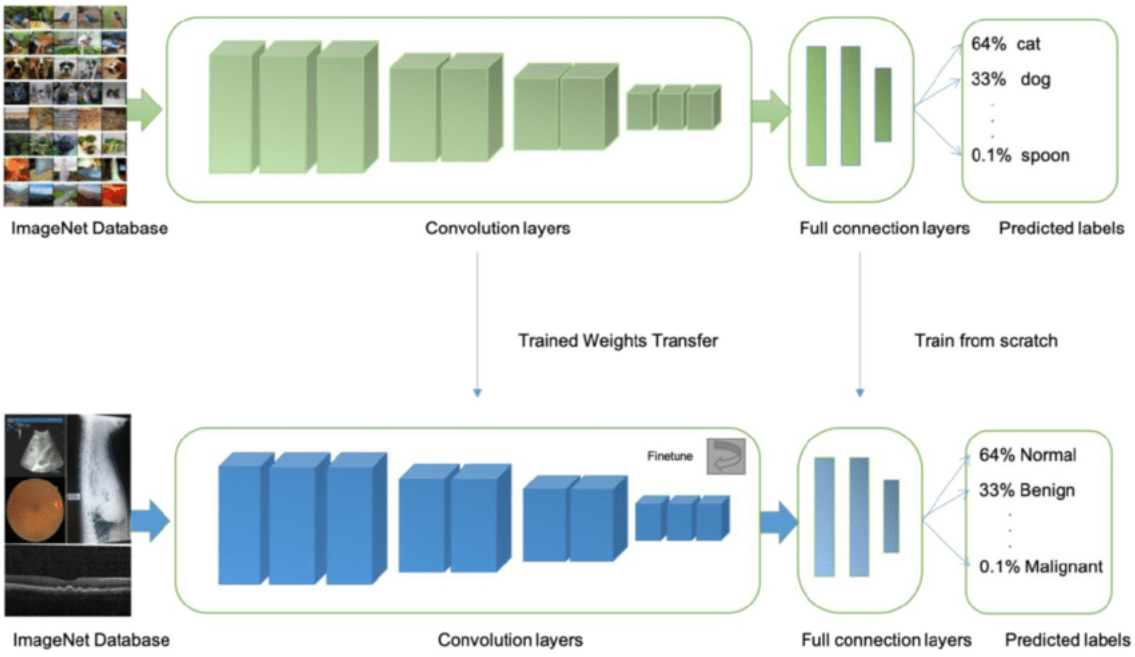


Figure 7.2.37: Transfer learning applied to CNN's [53]

The main advantages of using transfer learning are not needing a lot of labeled data, saving training time and a better performance of the neural network. A lot of pre-trained models are made available online for free and can be implemented to solve a specific vision problem.

Bibliography

[1] Peter Corke. *Robotics, Vision and Control*, volume 118. Springer International Publishing, 2017.

[2] Vision Doctor. Industrial machine vision. https://www.vision-doctor.com/en. Accessed 27-Jan-2023.

[3] Stemmer Imaging. Illumination. https://www.stemmer-imaging.com/en/knowledge-base/illumination/. Accessed 27-Jan-2023.

[4] Gigahertz-Optik. The perception of color. https://www.gigahertz-optik.com/en-us/service-and-support/knowledge-base/basics-light-measurement/light-color/color-perception/. Accessed 01-Feb-2023.

[5] Leddynamics. Finding the connection between tunable white and circadian lighting. https://leddynamics.com/finding-the-connection-between-tunable-white-and-circadian-lighting. Accessed 10-aug-2023.

[6] Stemmer Imaging. Optics. https://www.stemmer-imaging.com/en/knowledge-base/optics/. Accessed 27-Jan-2023.

[7] Wikipedia. Circle of confusion. https://en.wikipedia.org/wiki/circle_of_confusion/. Accessed 27-Jan-2023.

[8] Kyle Firestone. Lens mounts. https://www.edmundoptics.eu/knowledge-center/application-notes/imaging/lens-mounts/. Accessed 01-Feb-2023.

[9] Vision Doctor. Camera. https://www.vision-doctor.com/en/camera.html. Accessed 01-Feb-2023.

[10] Stemmer Imaging. Cameras. https://www.stemmer-imaging.com/en/knowledge-base/cameras/. Accessed 27-Jan-2023.

[11] ams OSRAM AG. How stereo vision works. https://ams.com/stereovision. Accessed 01-Feb-2023.

[12] GISGeography. Multispectral vs hyperspectral imagery explained. https://gisgeography.com/multispectral-vs-hyperspectral-imagery-explained/. Accessed 01-Feb-2023.

[13] ArcGIS. Contrast and brightness function. https://desktop.arcgis.com/en/arcmap/latest/manage-data/raster-and-images/contrast-and-brightness-function.htm. [Accessed 24-Nov-2022].

[14] Robot Academy. Diadic image processing. https://robotacademy.net.au/lesson/dyadic-image-processing/. Accessed 27-Jan-2023.

[15] Baptiste Pesquet. Convolutional neural networks. https://www.bpesquet.fr/mlhandbook/algorithms/convolutional_neural_ 2021. [Accessed 24-Nov-2022].

[16] Floris De Feyter, Toon Goedeme, and Patrick Vandewalle. Image enhancement computer vision-lecture 1, 2022.

[17] Richard Szeliski. Image alignment and stitching: A tutorial, 2006.

[18] Floris De Feyter and Toon Goedeme. Image segmentation computer vision-lecture 4, 2022.

[19] OpenCV. Opencv. https://opencv.org/. Accessed 27-Jan-2023.

[20] Datanerd. K means clustering. https://datanerdblog.wordpress.com/2016/06/13/k-means-clustering/comment-page-1/, 06 2016. Accessed 14-Dec-2022.

[21] Shree K. Nayar. Corner detection. https://www.youtube.com/watch?v=ZHwkG90Yvw, 03 2021. Accessed 21-Dec-2022.

[22] Floris De Feyter and Toon Goedeme. Object detection computer vision-lecture 3, 2022.

[23] Medium. Gradient and laplacian filter, difference of gaussians (dog). https://medium.com/jun94-devpblog/cv-3-gradient-and-laplacian-filter-difference-of-gaussians-dog-7c22e4a9d6cc. Accessed 27-Jan-2023.

[24] OpenCV. Cascade classifier. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. Accessed 22-Dec-2022.

[25] Satya Mallick. Histogram of oriented gradients explained using opencv. https://learnopencv.com/histogram-of-oriented-gradients/, 12 2016. Accessed 22-Dec-2022.

[26] srishivansh5404. Feature matching using brute force in opencv. https://www.geeksforgeeks.org/feature-matching-using-brute-force-in-opencv/, 01 2023. Accessed 03-Jan-2023.

[27] Strahinja Zivkovic. Feature matching methods comparison in opencv. https://datahacker.rs/feature-matching-methods-comparison-in-opencv/, 01 2021. Accessed 03-Jan-2023.

[28] The hough transform: The basics. https://aishack.in/tutorials/hough-transform-basics/. Accessed 01-Feb-2023.

[29] Socret Lee. Lines detection with hough transform. https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549, 05 2020. Accessed 01-Feb-2023.

[30] Straight line hough transform. https://sharky93.github.io/docs/gallery/auto$_e$*xamples/plot_line_hough_transform.html.A*
$Feb - 2023$.

[31] The English Teacher. Types of chairs: 25 different chair styles with esl pictures. https://eslforums.com/types-of-chairs/, 06 2019. Accessed 05-Jan-2023.

[32] Matlab. Object recognition 3 things you need to know. https://nl.mathworks.com/solutions/image-video-processing/object-recognition.html. Accessed 27-Jan-2023.

[33] PyImageSearch. The bag of (visual) words model. https://customers.pyimagesearch.com/the-bag-of-visual-words-model/. Accessed 27-Jan-2023.

[34] Towards Machine Learning. Face detection using opencv. https://towardsmachinelearning.org/face-detection-using-opencv/. Accessed 27-Jan-2023.

[35] Suman Sapkota. Artificial neural network back then. https://tsumansapkota.github.io/algorithm/2020/05/24/Neural-Network-Then/, 05 2020. Accessed 05-Jan-2023.

[36] Rainergewalt. Perceptrons - these artificial neurons are the fundamentals of neural networks. https://starship-knowledge.com/neural-networks-perceptrons, 10 2020. Accessed 05-Jan-2023.

[37] Jadon Shruti. Introduction to different activation functions for deep learning. https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092, 03 2018. Accessed 05-Jan-2023.

[38] Luis Camuñas-Mesa, Bernabé Linares-Barranco, and Teresa Serrano-Gotarredona. Neuromorphic spiking neural networks and their memristor-cmos hardware implementations. 08 2019. Accessed 05-Jan-2023.

[39] M.K. Gurucharan. Basic cnn architecture: Explaining 5 layers of convolutional neural network. https://www.upgrad.com/blog/basic-cnn-architecture/, 07 2022. Accessed 06-Jan-2023.

[40] pawangfg. Object detection vs object recognition vs image segmentation. https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/, 06 2022. Accessed 12-Dec-2022.

[41] Rohith Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e, 07 2018. Accessed 10-Jan-2023.

[42] Ceren Melek, Elena Sőnmez, and Songul Varlı Albayrak. Object detection in shelf images with yolo. 07 2019. Accessed 10-Jan-2023.

[43] Freesion. Open images dataset v5 (bounding boxes) - download. https://www.freesion.com/article/1482223540/. Accessed 31-Jan-2023.

[44] Recklexx. Imagenet. https://zhuanlan.zhihu.com/p/46438189, 10 2018. Accessed 31-Jan-2023.

[45] Gui-Song et al. Xia. Dota: a large-scale dataset for object detection in aerial images. 11 2017.

[46] Yuen Peng Loh and Chee Seng Chan. Getting to know low-light images with the exclusively dark dataset. 05 2018.

[47] Patrick Langechuan Liu. Single stage instance segmentation - a review. https://towardsdatascience.com/single-stage-instance-segmentation-a-review-1eeb66e0cc49, 04 2020. Accessed 30-Jan-2023.

[48] Derrick Mwiti and Katherine Li. Image segmentation: Architectures, losses, datasets, and frameworks. https://neptune.ai/blog/image-segmentation, 31 2023. Accessed 30-Jan-2023.

[49] Deval Shah. Coco dataset: All you need to know to get started. https://www.v7labs.com/blog/coco-dataset-guide:̃text=The%20COCO%20%28Common%20Objects%20in%20Context%29%20dataset%20is,object%20categories%20and%
31 2023. Accessed 31-Jan-2023.

[50] ZeYu Wang, YanXia Wu, Shuhui Bu, Pengcheng Han, and Guoyin Zhang. Structural inference embedded adversarial networks for scene parsing. *PLOS ONE*, 13:e0195114, 04 2018.

[51] Cityscapes Dataset. The cityscapes dataset. https://www.cityscapes-dataset.com/. Accessed 31-Jan-2023.

[52] Yan Wu, Tao Yang, Junqiao Zhao, Linting Guan, and Jiqian Li. Fully combined convolutional network with soft cost function for traffic scene parsing. pages 725–731, 07 2017.

[53] Jie Xu, Kanmin Xue, and Kang Zhang. Current status and future trends of clinical diagnoses via image-based deep learning. 10 2019.

[54] tutorialspoint. Histogram equalization. https://www.tutorialspoint.com/dip/histogram_equalization.htm. Accessed 24-Nov-2022.

[55] Imad Dabbura. K-means clustering: Algorithm, applications, evaluation methods, and drawbacks. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a, 09 2018. Accessed 14-Dec-2022.

[56] Shree K. Nayar. Sift detector. https://www.youtube.com/watch?v=ram-jbLJjFg, 03 2021. Accessed 21-Dec-2022.

[57] Skillcate. Hog intuition. https://www.youtube.com/watch?v=5nZGnYPyKLU, 06 2021. Accessed 22-Dec-2022.

[58] Shree K. Nayar. Sift descriptor. https://www.youtube.com/watch?v=IBcsS8gPzE, 03 2021. Accessed 23-Dec-2022.

[59] Shree K. Nayar. Hough transform. https://www.youtube.com/watch?v=XRBcxkZREg, 03 2021. Accessed 01-Feb-2023.

[60] Yun Cheong and Wei Chew. The application of image processing to solve occlusion issue in object tracking. *MATEC Web of Conferences*, 152:03001, 02 2018.

[61] Ashutosh Sharma. Structure of neurons: What is a neuron?, definition, types, structure, parts and function. https://testbook.com/learn/biology-neurons-structure-types-diagram/:̃text=Structure%20of%20Neuron%201%20Each%20neuron%20has%20a,layer%20called%20the%20myelin%20sheath.
06 2021. Accessed 05-Jan-2023.

[62] Jonathan Johnson. What's a deep neural network? deep nets explained. https://www.bmc.com/blogs/deep-neural-network/, 07 2020. Accessed 05-Jan-2023.

[63] John et al. McGonagle. Feedforward neural networks. brilliant.org. Accessed 05-Jan-2023.

[64] Mayank Mishra. Convolutional neural networks, explained. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939, 08 2020. Accessed 06-Jan-2023.

[65] Josh Starmer. Neural networks part 8: Image classification with convolutional neural networks (cnns). https://www.youtube.com/watch?v=HGwBXDKFk9I, 03 2021. Accessed 06-Jan-2023.

[66] Niklas Donges. What is transfer learning? exploring the popular deep learning approach. https://builtin.com/data-science/transfer-learning, 08 2022. Accessed 31-Jan-2023.