Working with Image Data

Image Data

Image data is photographic or trace object representing the underlying pixel data of an area of an image element, which is created, collected, and stored using image constructor devices[1].

Image data is typically stored in a variety of de facto industry standard proprietary formats. These often reflect the most popular image processing systems. Other graphic image formats, such as TIFF, GIF, PCX, etc., are used to store ancillary image data. Most software read such formats and allow you to display this data[2].

Image vision is very easy for a human. 80% of our perception can be attained from the visual image. We can easily distinguish different objects. What about machines? A robot may think of this task differently. It might think about the color, shape, sizes, position, brightness, and so on then, it can tell that this is a queen, for example.

Figure 1. Image recognition for humans is easy but difficult for a machine

Visual information is the most important type of information perceived, processed, and interpreted by the human brain. One-third of the cortical area of the human brain is dedicated to visual information processing.

[1] https://www.igi-global.com/dictionary/image-data-mining-based-on-wavelet-transform-for-visualization-of-the-unique-characteristics-of-image-data/51770
[2] http://planet.botany.uwc.ac.za/nisl/GIS/GIS_primer/page_18.htm

Remote Sensing

Remote sensing can be done from space (using satellite platforms), from the air (using aircraft), and from the ground (using static and vehicle-based systems). The same type of sensor, such as a multispectral digital frame camera, may be deployed on all three types of platforms for different applications. Each platform has unique advantages and disadvantages in terms of spatial coverage, access, and flexibility.

There are most common types of sensors used for mapping and image analysis. These include aerial cameras, film, digital, and sensors found on commercial satellites. New cameras and sensors are being introduced every year as the remote sensing industry grows and technology advances. The principles of sensor design introduced in this lesson will apply to new as well as older instruments used for image data capture. This content will focus on optical sensors, those which passively record reflected and radiant energy in the visible and near-visible wavelength bands of the electromagnetic spectrum. Other parts in this curriculum delve into both active sensors (such as lidar and radar) and passive sensors that operate outside the optical portion of the spectrum (thermal and passive microwave).

Image Data Formats

Image data are stored in a rectangular matrix of rows and columns. Radiometric resolution determines how many gradations of brightness can be stored for each cell (pixel) in the matrix; 8-bit resolution, where each pixel contains an integer value from 0 to 255, is most common. Modern sensors often collect data at higher resolution, and advanced image processing software can make use of these values for analysis. However, the human eye cannot detect very small differences in brightness.

In a grayscale image, 0 = black and 255 = white, and there is just one 8-bit value for each pixel. However, in a natural color image, there is an 8-bit value for red, an 8-bit brightness value for green, and an 8-bit value for blue. Therefore, each pixel in a color image requires 3 distinct values to be stored in the file. There are three possible ways to organize these values in a raster file.

- BIP - Band Interleaved by Pixel: The red value for the first pixel is written to the file, followed by the green value for that pixel, followed by the blue value for that pixel, and so on for all the pixels in the image.
- BIL - Band Interleaved by Line: All of the red values for the first row of pixels are written to the file, followed by all of the green values for that row followed by all the blue values for that row, and so on for every row of pixels in the image.

- BSQ - Band Sequential: All of the red values for the entire image are written to the file, followed by all of the green values for the entire image, followed by all the blue values for the entire image.

Orthoimages are delivered in a variety of image formats, compressed and uncompressed. The most common are TIF and JPG.

Other popular image formats you may encounter are[3]:

- ECW (link is external): developed by ERMapper, now owned by ERDAS. Uses wavelet compression to reduce file size.
- GRID (link is external): developed by Esri; supported by some remote sensing software packages, but not as common as other formats.
- IMG (link is external): developed by ERDAS for Imagine; supported by many GIS and remote sensing software packages.
- JP2 (link is external): JPEG 2000, developed by the JPEG group and widely supported by most GIS and remote sensing software packages.
- SID (link is external): MrSid, developed by Lizard Tech. Uses wavelet compression to reduce file size. Read by many software packages but requires a proprietary software license to create.

Component of image processing includes a scene, illumination, camera and a computer.

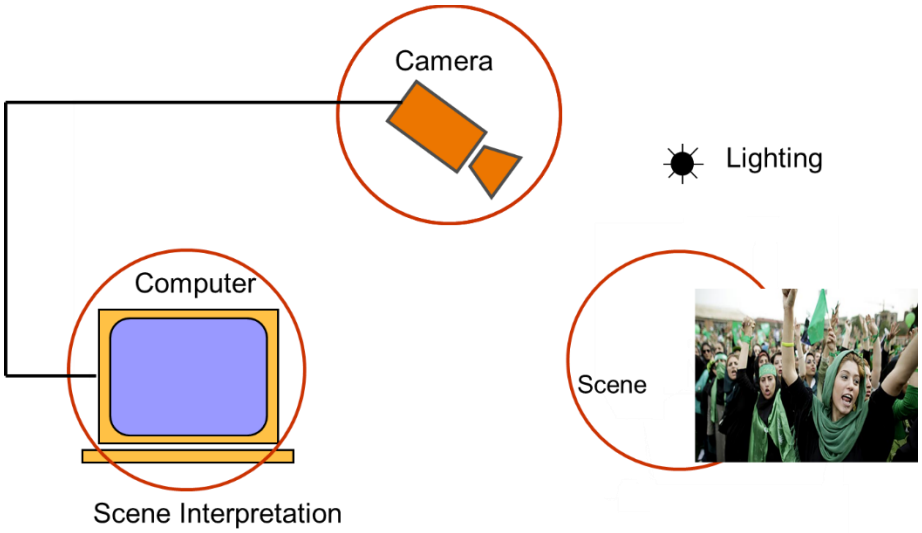


Figure 2. Component of basic image processing

Image Processing

Image processing is a method to perform some operations on an image to get an enhanced image or extract some useful information from it. It is a type of signal processing in which input is an image, and output may be an image or characteristics/features associated with that image.

[3] https://www.e-education.psu.edu/geog480/node/458

Nowadays, image processing is among rapidly growing technologies. It forms a core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools.
- Analysing and manipulating the image.
- Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing, namely, analogue and digital image processing. Analogue image processing can be used for hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in the manipulation of digital images by using computers. The three general phases that all types of data have to undergo while using digital techniques are pre-processing, enhancement, and display, information extraction[4].

Key Stages in Digital Image Processing

Usually, Image Processing system includes treating images as two-dimensional signals while applying already set signal processing methods to them. Figure 3 shows the key stages of digital image processing.

Image Acquisition: This is the first step or process of the fundamental steps of digital image processing. Image acquisition could be as simple as being given an image that is already in digital form. Generally, the image acquisition stage involves preprocessing, such as scaling, etc.

Image Enhancement: Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured or simply to highlight certain features of interest in an image. Such as changing brightness & contrast, etc.

Image Restoration: Image restoration is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.

[4] https://sisu.ut.ee/imageprocessing/book/1

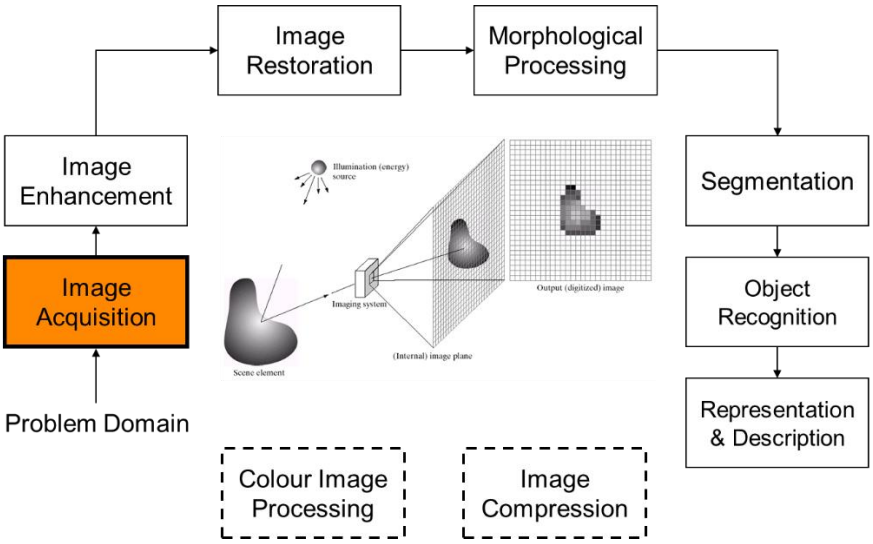


Figure 3. Component of basic image processing

Morphological Processing: Morphological processing deals with tools for extracting image components that are useful in the representation and description of shape.

Segmentation: Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward a successful solution of imaging problems that require objects to be identified individually.

Representation and Description: Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region or all the points in the region itself. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. Description deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

Object recognition: Recognition is the process that assigns a label, such as, “vehicle” to an object based on its descriptors.

Color Image Processing: Color image processing is an area that has been gaining its importance because of the significant increase in the use of digital images over the Internet. This may include color modeling and processing in a digital domain etc.

Compression: Compression deals with techniques for reducing the storage required to save an image or the bandwidth to transmit it. Particularly in the uses of internet it is very much necessary to compress data.

Image processing techniques

Sampling and quantization

In order to become suitable for digital processing, an image function f(x,y) must be digitized both spatially and in amplitude. Typically, a frame grabber or digitizer is used to sample and quantize the analogue video signal. Hence in order to create an image which is digital, we need to covert continuous data into digital form. There are two steps in which it is done:

- Sampling
- Quantization

The sampling rate determines the spatial resolution of the digitized image, while the quantization level determines the number of grey levels in the digitized image. The magnitude of the sampled image is expressed as a digital value in image processing. The transition between continuous values of the image function and its digital equivalent is called quantization.

The number of quantization levels should be high enough for human perception of fine shading details in the image. The occurrence of false contours is the main problem in image which has been quantized with insufficient brightness levels.

In this part, we will focus on two key stages in digital image processing. Sampling and quantization will be defined properly. Spatial and grey-level resolutions will be introduced, and examples will be provided. An introduction to implementing the shown examples in MATLAB will also be given here. (https://sisu.ut.ee/imageprocessing/book/2)

Resizing image

Image interpolation occurs when you resize or distort your image from the one-pixel grid to another. Image resizing is necessary when you need to increase or decrease the total number of pixels, whereas remapping can occur when you are correcting for lens distortion or rotating an image. Zooming refers to increasing the quantity of pixels so that when you zoom an image, you will see more detail.

Interpolation works by using known data to estimate values at unknown points. Image interpolation works in two directions and tries to achieve a best approximation of a pixel's intensity based on the values at surrounding pixels. Common interpolation algorithms can be grouped into two categories: adaptive and non-adaptive. Adaptive methods change depending on what they are interpolating, whereas non-adaptive methods treat all pixels equally. Non-adaptive algorithms include: nearest neighbor, bilinear, bicubic, spline, sinc, lanczos and others. Adaptive algorithms include many proprietary algorithms in licensed software such as: Qimage, PhotoZoom Pro and Genuine Fractals.

Many compact digital cameras can perform both optical and digital zoom. A camera performs an optical zoom by moving the zoom lens so that it increases the magnification of light. However, a digital zoom degrades quality by simply interpolating the image. Even though the photo with digital zoom contains the same number of pixels, the detail is clearly far less than with an optical zoom.

Aliasing and image enhancement

Digital sampling of any signal, whether sound, digital photographs, or other, can result in apparent signals at frequencies well below anything present in the original. Aliasing occurs

when a signal is sampled at less than twice the highest frequency present in the signal. Signals at frequencies above half the sampling rate must be filtered out to avoid the creation of signals at frequencies not present in the original sound. Thus digital sound recording equipment contains low-pass filters that remove any signals above half the sampling frequency.

Since a sampler is a linear system, then if an input is a sum of sinusoids, the output will be a sum of sampled sinusoids. This suggests that if the input contains no frequencies above the Nyquist frequency, then it will be possible to reconstruct each of the sinusoidal components from the samples. This is an intuitive statement of the Nyquist-Shannon sampling theorem.

Anti-aliasing is a process that attempts to minimize the appearance of aliased diagonal edges. Anti-aliasing gives the appearance of smoother edges and higher resolution. It works by taking into account how much an ideal edge overlaps adjacent pixels.

Image enhancement: contrast enhancement

Image enhancement techniques have been widely used in many applications of image processing where the subjective quality of images is important for human interpretation. Contrast is an important factor in any subjective evaluation of image quality. Contrast is created by the difference in luminance reflected from two adjacent surfaces. In other words, contrast is the difference in visual properties that makes an object distinguishable from other objects and the background. In visual perception, contrast is determined by the difference in the colour and brightness of the object with other objects. Our visual system is more sensitive to contrast than absolute luminance; therefore, we can perceive the world similarly regardless of the considerable changes in illumination conditions. Many algorithms for accomplishing contrast enhancement have been developed and applied to problems in image processing.

In this part, we will focus on contrast enhancement. Linear and non-linear transformation functions such as image negatives, logarithmic transformations, power-law transformations, and piecewise linear transformations will be discussed. Histogram process and histogram of four basic grey-level characteristics will be introduced.

If the contrast of an image is highly concentrated on a specific range, e.g. an image is very dark, the information may be lost in those areas which are excessively and uniformly concentrated. The problem is to optimize the contrast of an image in order to represent all the information in the input image.

Spatial filtering is a form of finite impulse response (FIR) filtering. The filter is actually a mask of weights arranged in a rectangular pattern. The process is one of sliding the mask along the image and performing a multiply and accumulate operation on the pixels covered by the mask

Arithmetic and logic operations

Image arithmetic applies one of the standard arithmetic operations or a logical operator to two or more images. The operators are applied in a pixel-by-pixel way, i.e. the value of a pixel in the output image depends only on the values of the corresponding pixels in the input images. Hence, the images must be of the same size. Although image arithmetic is the most simple form of image processing, there is a wide range of applications. A main advantage of arithmetic operators is that the process is very simple and therefore fast.

Logical operators are often used to combine two (mostly binary) images. In the case of integer images, the logical operator is normally applied in a bitwise way.

Spatial domain filtering

Filtering is a technique for modifying or enhancing an image. Spatial domain operation or filtering (the processed value for the current pixel processed value for the current pixel depends on both itself and surrounding pixels). Hence Filtering is a neighborhood operation in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel.

Spatial filtering is a form of finite impulse response (FIR) filtering. The filter is actually a mask of weights arranged in a rectangular pattern. The process is one of sliding the mask along with the image and performing a multiply and accumulate operation on the pixels covered by the mask.

Feature extraction from images

In real life, all the data we collect are in large amounts. To understand this data, we need a process. Manually, it is not possible to process them. Here's when the concept of feature extraction comes in. Suppose you want to work with some of the big machine learning projects or the coolest and popular domains such as deep learning, where you can use images to make a project on object detection. Making projects on computer vision where you can work with thousands of interesting projects in the image data set. To work with them, you have to go for feature extraction.

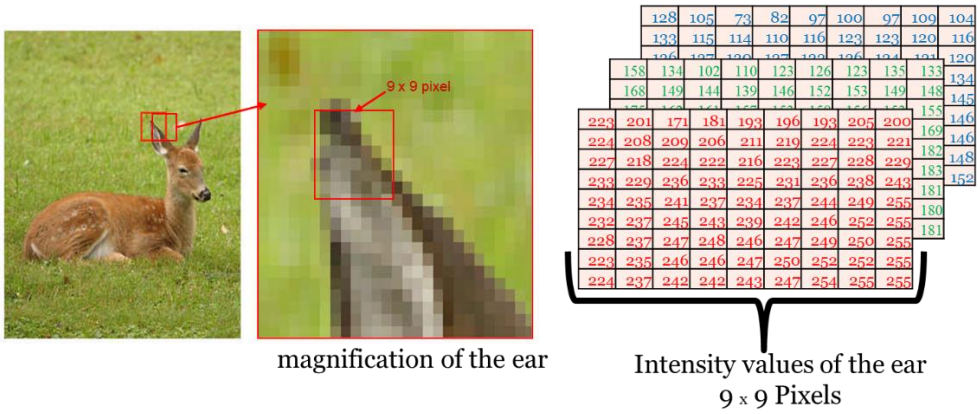
First, we need to understand how a machine can read and store images. Loading the image, reading them and then processing them through the machine is difficult because the machine does not have eyes like us. Let's have a look at how a machine understands an image. Machines see any images in the form of a matrix of numbers. The size of this matrix actually depends on the number of pixels of the input image. The Pixel Values for each of the pixels stands for or describe how bright that pixel is and what color it should be. So In the simplest case of the binary images, the pixel value is a 1-bit number indicating either foreground or background.

So pixels are the numbers or pixel values that denote the pixel's intensity or brightness.

Smaller numbers closer to zero help to represent black, and the larger numbers closer to 255 denote white. So this is the concept of pixels and how the machine sees the images without eyes through the numbers. The dimensions of the image 28 x 28. And if you want to check, then by counting the number of pixels, you can verify. But, for the case of a coloured image, we have three Matrices or channels: Red, Green and Blue. So in these three matrices, each of the matrix has values between 0-255 which represents the intensity of the colour of that pixel.

If we consider a coloured image, for example, the image below, we can easily understand the image. So being a human, you have eyes to see and say it is a deer coloured image. But how a computer can understand it is a coloured or black and white image? Computer just

look different numbers and using advanced object recognition algorithms computer can identify what type of image it is.



So you can see, we also have three matrices which represents the channel of RGB – (for the three color channels – Red, Green, and Blue). On the right, we have three matrices. These three channels are superimposed and used to form a coloured image. So this is how a computer can differentiate between the images.

Additional Sources:

https://www.mygreatlearning.com/blog/feature-extraction-in-image-processing/

https://www.analyticsvidhya.com/blog/2019/08/3-techniques-extract-features-from-image-data-machine-learning-python/

https://link.springer.com/chapter/10.1007/978-1-4302-5930-5_2

Different types of Feature

Features are the basic attributes or aspects which clearly help us identify the particular object, image, or anything. Features are the marked properties which are unique. While working on an image dataset we need to extract the features of different images which will help us segregate the images based on certain features or aspects. There is no exact definition of the features of an image but things like the shape, size, orientation, etc. constitute the feature of the image. Extracting these features can be done using different techniques using python.

Features as Pixel Value

The number of pixels in an image is the same as the size of the image for grayscale images, we can find the pixel features by reshaping the shape of the image and returning the array form of the image[5]. Let's see step-by-step process:

a. Importing the required libraries

[5] https://analyticsindiamag.com/image-feature-extraction-using-scikit-image-a-hands-on-guide/

We will look at all the aspects of the image so we need to import different libraries including NumPy, pandas, etc.

```
#step a
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#matplotlib inline
from skimage.io import imread, imshow
```

b. Loading the image

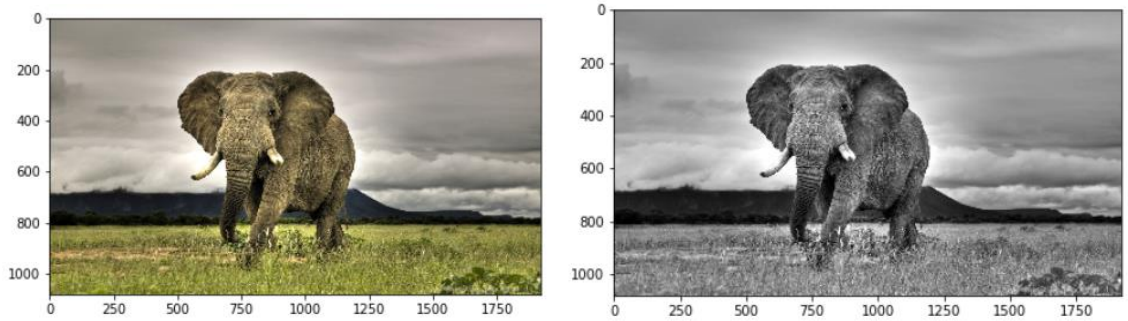
You can use any image from your system. I have an image named'elephant.jpg' for which I will be performing feature extraction. You can just save this image into your local directory and just follow the steps.

```
#step b

image1 = imread('elephant.jpg')
imshow(image1);

image2 = imread('elephant.jpg', as_gray=True)
imshow(image2);
```

The figure below will be display if you run the code above.



c. Analyzing both the images

Let's see the shape of the image

```
#step c

print(image1.shape)
print(image2.shape)
```

you will see the shape of the image as below:

```
(1080, 1920, 3)
(1080, 1920)
```

Here we can see that the colored image contains rows, columns, and channels as it is a colored image there are three channels RGB while grayscale pictures have only one channel. So we can clearly identify the colored and grayscale images by their shapes.

Let's look at the size of the images.

```
print(image1.size)
print(image2.size)
```

```
6220800
2073600
```

Image size is the product of the rows, columns, and channels. Despite being the same images grayscale is smaller in size because it has only 1 channel.

d. Extract pixel features

The number of pixels in an image is the same as the size of the image for grayscale images we can find the pixel features by reshaping the shape of the image and returning the array form of the image. Similarly, we can find the pixel feature for the colored image.

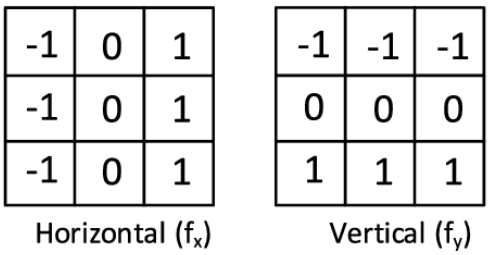
```
#step d

#for gray scale image
pixel_feat1 = np.reshape(image2, (1080 * 1920)

#for color image
pixel_feat2 = np.reshape(image1, (1080 * 1920 * 3)
```

Extracting Edge Features

Edges in an image are the corners where the pixel change drastically, as the images are stored in array form, we can visualize different values and see where the change in pixel value is higher but doing it manually takes time, Scikit-Image provides functions for image edge features extraction namely:
Prewitt Kernel: It is an edge detection kernel that works separately for both horizontal and vertical axis.



Sobel Kernel: It works on creating images with emphasis on edges
Canny Algorithm: Canny Algorithm is an edge detection technique that uses a multi-stage algorithm to detect a wide range of edges in images.

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

x filter

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

y filter

An example for edge detection using sobel algorithm is shown below:

```
from skimage import filters

# prewitt kernel
pre_hor = prewitt_h(image2)
pre_ver = prewitt_v(image2)

# Sobel Kernel
ed_sobel = filters.sobel(image2)

#canny algorithm
can = feature.canny(image2)

imshow(pre_ver, cmap='gray');
imshow(pre_hor, cmap='gray');
imshow(ed_sobel, cmap='gray');
imshow(can, cmap=’gray);
```

The utput will be like these:

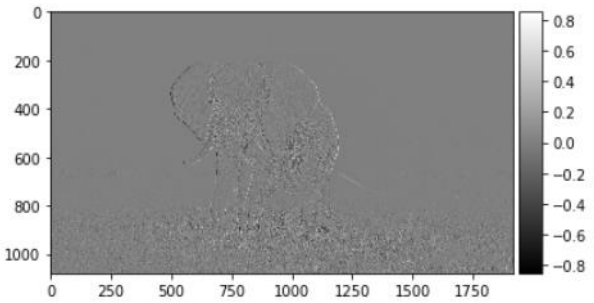
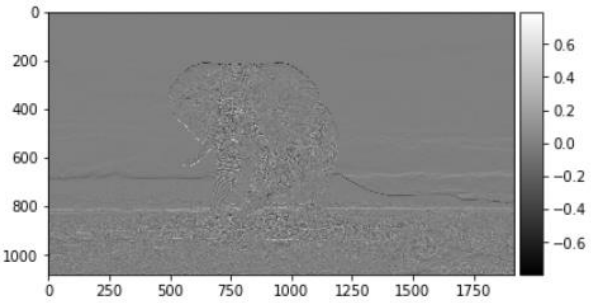


Figure 4(a). edge detection using prewitt vertical kernel



3(b) edge detection using prewitt horizontal kernel

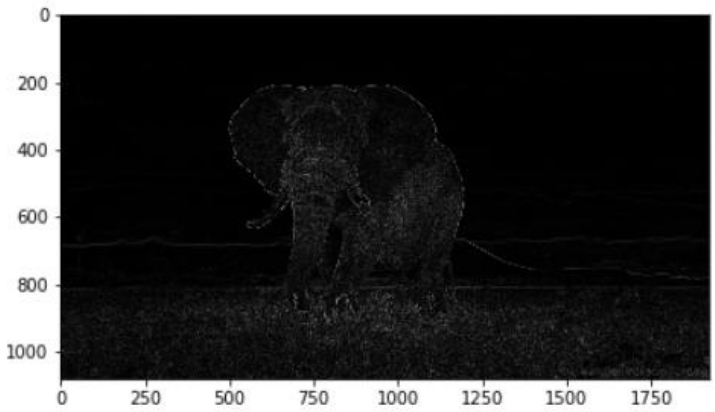


Figure 5(c). edge detection using sobel kernel

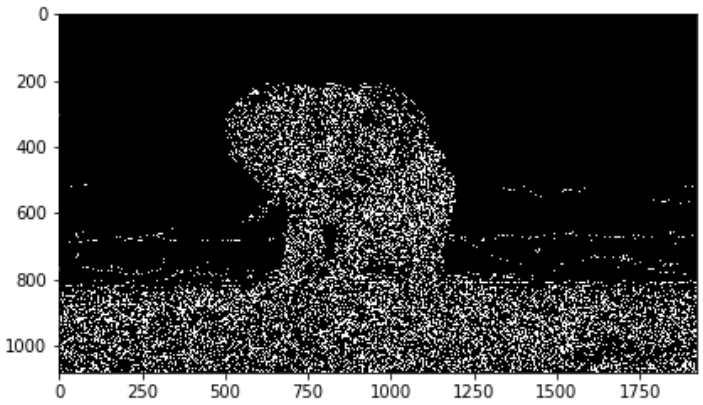


Figure 6(d). edge detection using Canny algorithms

Extracting Corner features:

A corner is an area of an image that has a large variation in pixel color intensity values in all directions. One popular algorithm for detecting corners in an image is called the Harris Corner Detector. Here is some basic code for the **Harris Corner Detector**[6].

Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation, and illumination.

Now run the code below and try to understand:

[6] https://medium.com/data-breach/introduction-to-harris-corner-detector-32a88850b3f6#:~:text=Compared%20to%20the%20previous%20one,distinguishing%20between%20edges%20and%20corners.

```
from matplotlib import pyplot as plt

from skimage import data
from skimage.feature import corner_harris, corner_subpix, corner_peaks
from skimage.transform import warp, AffineTransform
from skimage.draw import ellipse

# Sheared checkerboard
tform = AffineTransform(scale=(1.3, 1.1), rotation=1, shear=0.7,
                        translation=(110, 30))
image = warp(data.checkerboard()[:90, :90], tform.inverse,
             output_shape=(200, 310))
# Ellipse
rr, cc = ellipse(160, 175, 10, 100)
image[rr, cc] = 1
# Two squares
image[30:80, 200:250] = 1
image[80:130, 250:300] = 1

coords = corner_peaks(corner_harris(image), min_distance=5, threshold_rel=0.02)
coords_subpix = corner_subpix(image, coords, window_size=13)

fig, ax = plt.subplots()
ax.imshow(image, cmap=plt.cm.gray)
ax.plot(coords[:, 1], coords[:, 0], color='cyan', marker='o',
        linestyle='None', markersize=6)
ax.plot(coords_subpix[:, 1], coords_subpix[:, 0], '+r', markersize=15)
ax.axis((0, 310, 200, 0))
plt.show()
```

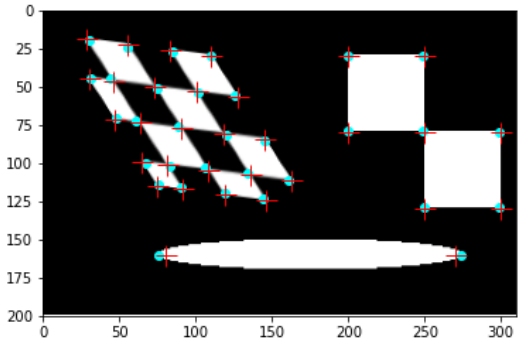


Figure 7. Corner detection using Harris corner detector algorithm

Scale Invariant Feature Transform (SIFT)

When we rotate an image or change its size, how can we make sure the features don't change? The methods I've used above aren't good at handling this scenario. OpenCV has an algorithm called SIFT that is able to detect features in an image regardless of changes to its size or orientation. This property of SIFT gives it an advantage over other feature detection algorithms, which fail when you make transformations to an image.

Here is an example of code that uses SIFT:

```
# Python Code for SIFT feature
import numpy as np
import cv2 as cv

# Read the image
img = cv.imread('chessboard.jpg')

# Convert to grayscale
gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)

# Find the features (i.e. keypoints) and feature descriptors in the image
sift = cv.SIFT_create()
kp, des = sift.detectAndCompute(gray,None)

# Draw circles to indicate the location of features and the feature's orientation
img=cv.drawKeypoints(gray,kp,img,flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

# Save the image
cv.imwrite('sift_with_features_chessboard.jpg',img)
```

Figure 8. Example of SIFT feature detection

Each of those circles indicates the size of that feature. The line inside the circle indicates the orientation of the feature:

Speeded-Up Robust Features (SURF)

SURF is a faster version of SIFT. It is another way to find features in an image. The python code is given below:

```
# Python Code for SURF features

import numpy as np
import cv2 as cv

# Read the image
img = cv.imread('chessboard.jpg')

# Find the features (i.e. keypoints) and feature descriptors in the image
surf = cv.xfeatures2d.SURF_create(400)
kp, des = sift.detectAndCompute(img,None)

# Draw circles to indicate the location of features and the feature's orientation
img=cv.drawKeypoints(gray,kp,img,flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYP
OINTS)

# Save the image
cv.imwrite('surf_with_features_chessboard.jpg',img)
```

Binary Robust Independent Elementary Features (BRIEF)

The BRIEF algorithm uses a smoothed image, and it selects a set of pixel pairs in a specific way n_d nd(x, y), and then compare the gray values between these pixel pairs. BRIEF does not calculate the descriptor but directly finds a binary string. For example, the gray values of the first point pair are p and q respectively. If p is less than q, the result is 1, otherwise it is 0. In this way, compare n d point pairs to get one n_d ndDimensional binary string.

n d can be 128, 256, 512. OpenCV provides support for these, but by default it is 256 (OpenC uses bytes to represent them, so these values correspond to 16, 32, 64, respectively). When we get these binary strings, we can use Hamming distance to match them. A very important point is: BRIEF is a feature descriptor, it does not provide a method to find features. So we have to use other feature detectors, such as SIFT and SURF. The original literature recommends the use of the CenSurE feature detector, this algorithm is very fast. Moreover, the description of the key points of CenSurE by the BRIEF algorithm is better than the description of the key points of SURF. Briefly, BRIEF is a fast method for calculating and matching feature point descriptors. This algorithm can achieve a high recognition rate unless there is a large rotation in the plane. Python code with example is included below:

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('simple.jpg',0)

# Initiate FAST detector
star = cv.xfeatures2d.StarDetector_create()
```

```
# Initiate BRIEF extractor
brief = cv.xfeatures2d.BriefDescriptorExtractor_create()

# find the keypoints with STAR
kp = star.detect(img,None)

# compute the descriptors with BRIEF
kp, des = brief.compute(img, kp)

print( brief.descriptorSize() )
print( des.shape )
```

Figure 9. Example for BRIEF features

Deep Networks as Feature Extractors: Feature extraction using manual process is time consuming and tedious task. For automatic feature extraction is also possible using deep learning methods. For this automatic feature extraction you need to have advanced machine learning and deep learning knowledge. This blog is good read for the interested readers.