VIDEO PROCESSING REPORT

OPENCV Library

Prepared by:

Jessé Ferreira Filho

Academic Year : 2018-2019

UFR Sciences et Techniques de Tours - Site de Blois

Contents

List of Figures

1 Introduction

In order to put in practice the content learned in class, this report code project has as goal explore and explain algorithms from OpenCV library.

2 What is OpenCV?

OpenCV (Open Source Computer Vision Library [9]) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms.

3 Filters

Filters are used to perform various linear or non-linear filtering operations on 2D images. It means that for each pixel location (x,y) in the source image (normally, rectangular), its neighborhood is considered and used to compute the response. In case of a linear filter, it is a weighted sum of pixel values. In case of morphological operations, it is the minimum or maximum values, and so on.

Before take off the explanation and algorithms, it important highlght some terms such as blur and kernels. **Blur** can combine the value of each pixel with its neighbors. This is really useful for eliminating small variations or noise. **Kernel** is a small matrix within representation of pixels. It is used in several algorithm of image processing. This is accomplished by doing a convolution between a kernel and an image.

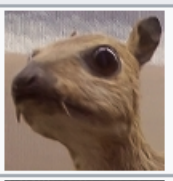
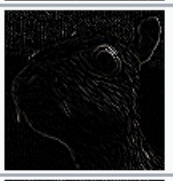
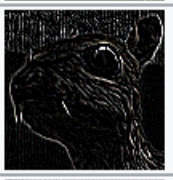
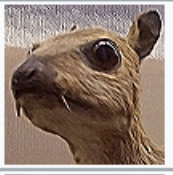
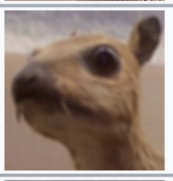
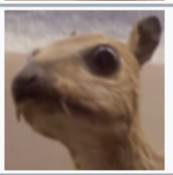
Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5 × 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Figure 1: Kernels examples to compare results and usage. [7]

3.1 2D Convolution

In this filter we start with a kernel with a pair of values unpaired e.g. (3,3) (5,5), which is simply a small matrix of weights. Then this mask/kernel will running way over the 2D input data (image processed), performing an element wise multiplication or/and may combining **averaging**, **mean** or **gaussian** smoothing methods, then it will return a single output pixel of each slice. The result of this process will be an another 2D matrix with less pixels.

Consequently the size of Kernel directly determines the quantity of input data that will be processed within this process. In some case where we have not input data it will normalized. [3]

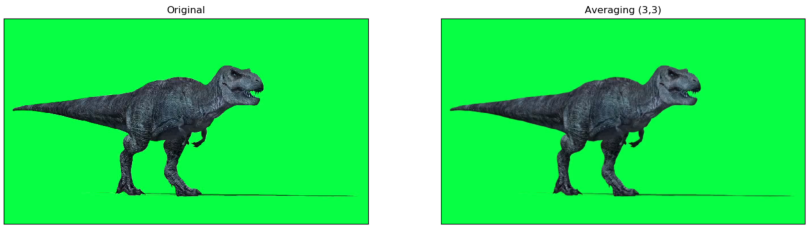


Figure 2: 2D Convolution Kernel 3x3.

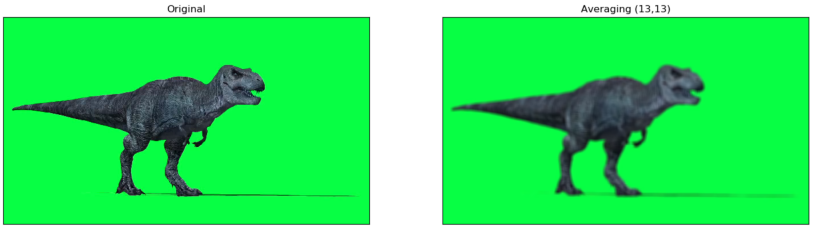


Figure 3: 2D Convolution Kernel 13x13.

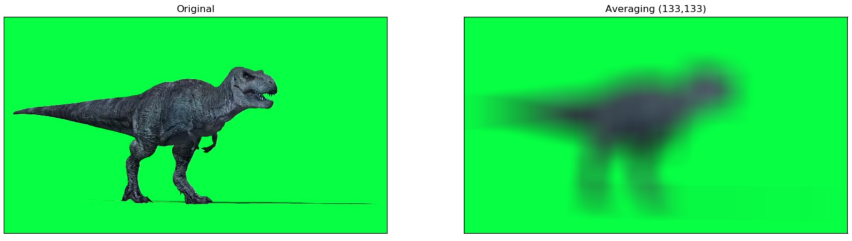


Figure 4: 2D Convolution Kernel 113x113.

This filter keep this kernel above a pixel in a matrix(5x5), add all the 25 pixels below this kernel, take its average and replace the central pixel with the new average value. It continues this operation for all the pixels in the image.

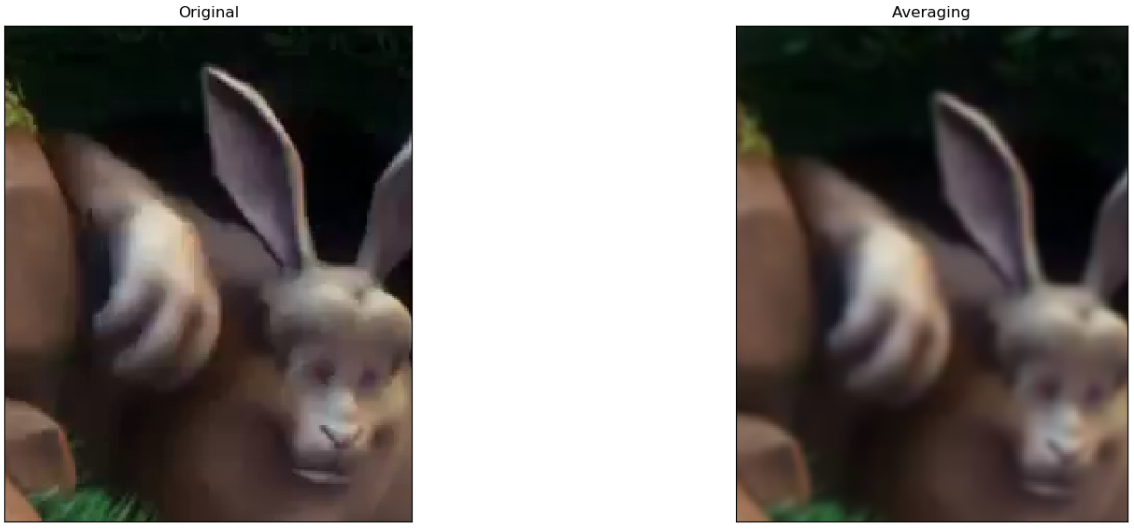


Figure 5: Blurring images can reduce noise with a cost of loss information.

3.1.1 Image Smoothing by Averaging

A little bit similar to Image 2D Convolution process, but in a small scale. It simply takes the average of all the pixels under kernel area and replace the central element.

The idea of this filtering is simply to replace each pixel value in an image by the mean value of its neighbors, but from a small kernell. The main goal is reduce a huge area and quantity of noise.

In this time the kernel is automatically configured by two functions *blur()* and *boxFilter()*. It can calculate by integral characteristics such as over each pixel neighborhood considering the size of windows.

3.1.2 Image Smoothing by Gaussian

This filter calculate the ideal frequency domain to each pixel based in a different kernel that represents the shape of a Gaussian ('bell-shaped'), it means that on this function is focused on contour of forms noise reduction.

Figure 6: Using Gaussian smoothing methods

3.1.3 Image Smoothing by Median

Like the others Median filtering is a simple function that reducing the amount of intensity variation between one pixel and the next. It is often used to reduce small grain of noise in images and very effective at removing noise while preserving edges.

The mechanism it is similar to the others, but in this time the kernel, replacing each value with the median value of neighbouring pixels. All number in the kernel is sorted and then the central pixel is the chosen to be replaced.

4 Contours

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition [2]. Three important tips should be take in consideration when we will work tracking contours:

- For better accuracy, use binary images;
- Before finding contours, apply threshold or canny edge detection;
- To improve the process, we need to have a best contrast possible between the object and the background such as black and white;

A binary image or "bi-tonal", it is also called bi-level or two-level is a image that has only possible values for each pixel. The names black-and-white, B&W, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images. A binary image can be stored in memory as a bitmap, a packed array of bits [1].

4.1 Thresholding

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images (Shapiro, et al. 2002). An image segmentation has two objectives:

- To decompose the image into parts;
- To perform a change of representation;

4.1.1 Simple Thresholding

Here, the matter is straight forward. If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black).

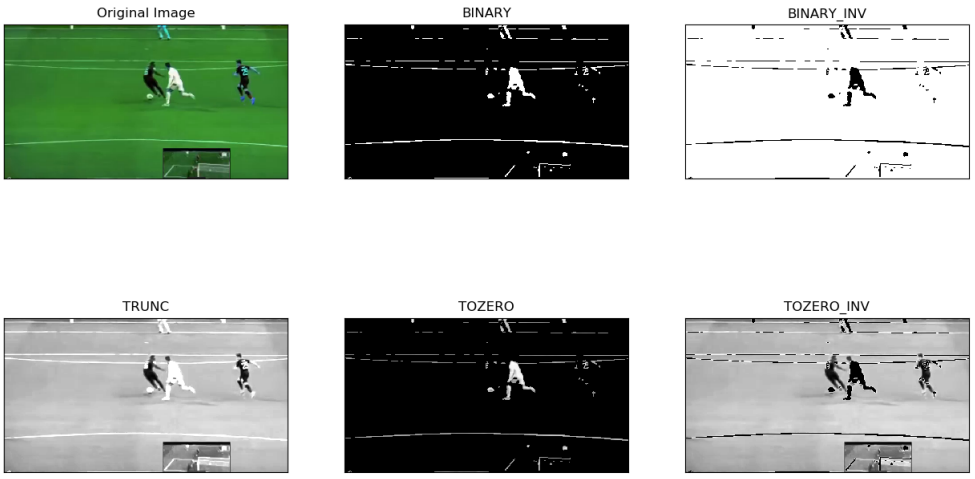


Figure 7: Simple Thresholding image processing

- THRESH_BINARY: This function transforms a image to a binary image (thresh = 0 and maxValue = 255)
- THRESH_BINARY_INV: just the opposite of binary thresholding.
- THRESH_TRUNC: This function get the threshold value of a source pixel and then if it is great than to a global thresh passing it to grayscale. maxValue is ignored.
- THRESH_TOZERO: If the source pixel value is greater than the threshold passing to grayscale. Otherwise it is set to zero. maxValue is ignored.
- THRESH_TOZERO_INV:just the opposite of to zero thresholding.

4.1.2 Adaptive Thresholding

This algorithms is called like that because it more useful for different lighting conditions in different areas. In this, the algorithm calculate the threshold for a small regions of the image. So we get different thresholds for different regions of the same im-

age and it gives us better results for images with varying illumination. This process it maded by three input params; Adaptive Method (ADAPTIVE_THRESH_MEAN_C and ADAPTIVE_THRESH_GAUSSIAN_C), Block Size (size of neighbourhood area) and C (constant).

- ADAPTIVE_THRESH_MEAN_C : threshold value is the mean of neighbourhood area.
- ADAPTIVE_THRESH_GAUSSIAN_C : threshold value is the weighted sum of neighbourhood values where weights are a gaussian window.

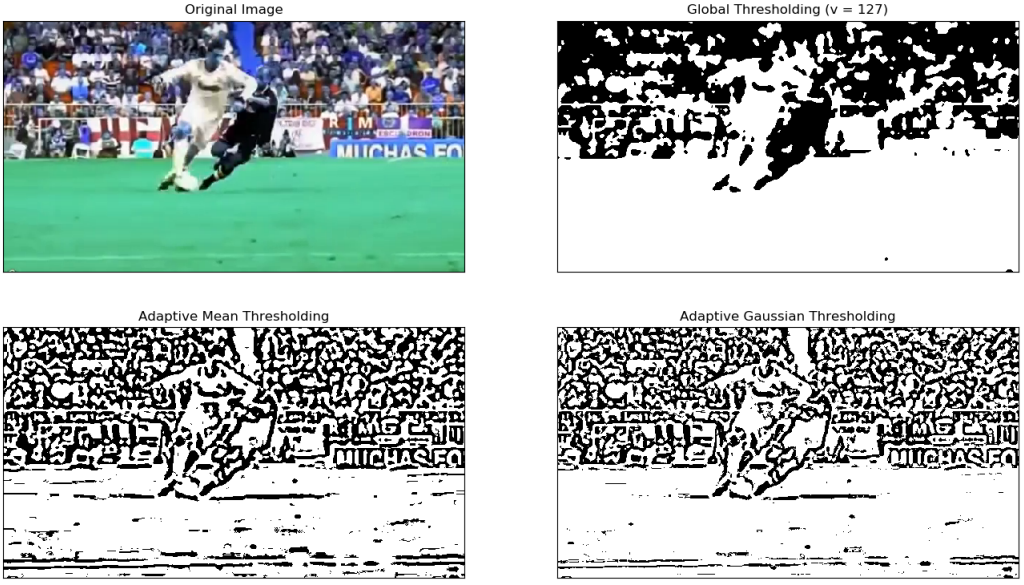


Figure 8: Adaptive Thresholding image processing

4.2 Canny Edge Detection

Developed by John F. Canny this algorithm multi-stage has Noise Reduction, Finding Intensity Gradient of the Image, Non-maximum Suppression and Hysteresis Thresholding. Together this algorithm can do detection of edges with low error rate following the process:

1. Apply Gaussian filter to smooth and reduce image noise.
2. Find the intensity gradients
3. Find the most evident edge.
4. Search by potential edges
5. Finalize the detection of edges by suppressing irrelevant edges.

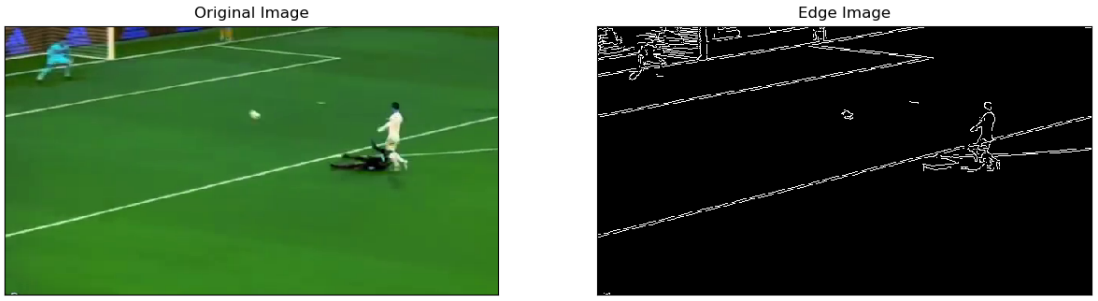


Figure 9: Using Canny Edge Detection

4.3 Image Gradients

In order to compare the three type of gradients we may chose a video with a uniform background. Fill that be came more easy to identify which gradient filters proprieties.

Sobel operators is a joint Gausssian smoothing plus differentiation operation, so it is more resistant to noise, because we can test and specify the direction of derivatives. In another hand Laplacian is the divergence of the gradient in a higher dimensions (More than 2 directions), where it represents the flux density of the gradient flow of a function.

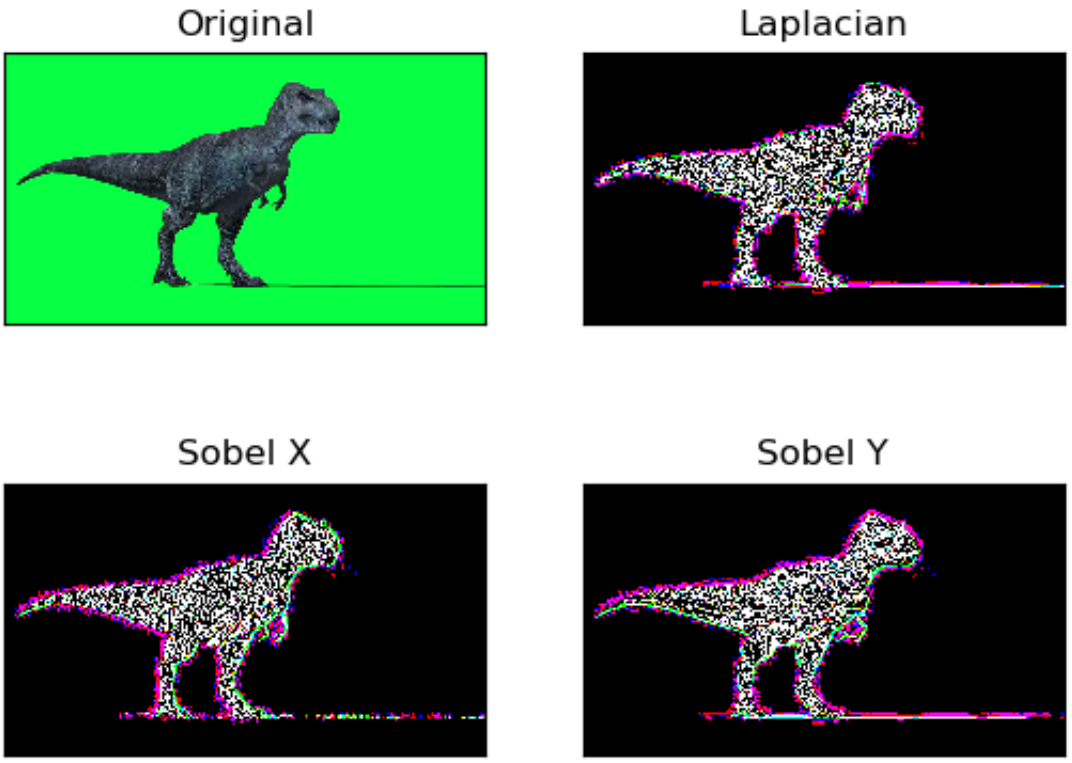


Figure 10: Using Gradients functions

5 Morphological Transformations

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation.

The basic idea of erosion is like as in 2D convolution, the kernell scan the each image pixel considering it as only 1 if all pixel within the kernel are 1, otherwise it is eroded (made to zero). This approach is useful for removing small white noises, detach two connected objects such as a chroma key background and an object. Dilation is just opposite of erosion. It increases the white region in the image or size of foreground object increases.

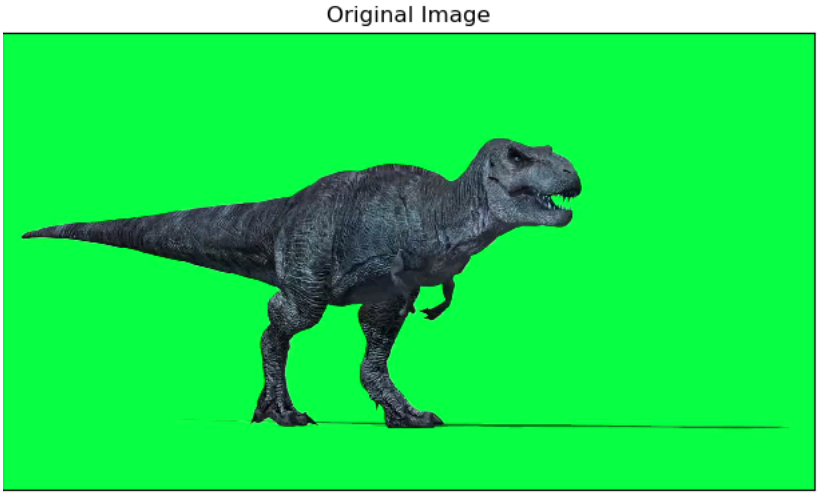


Figure 11: Before Morphological Transformations Process

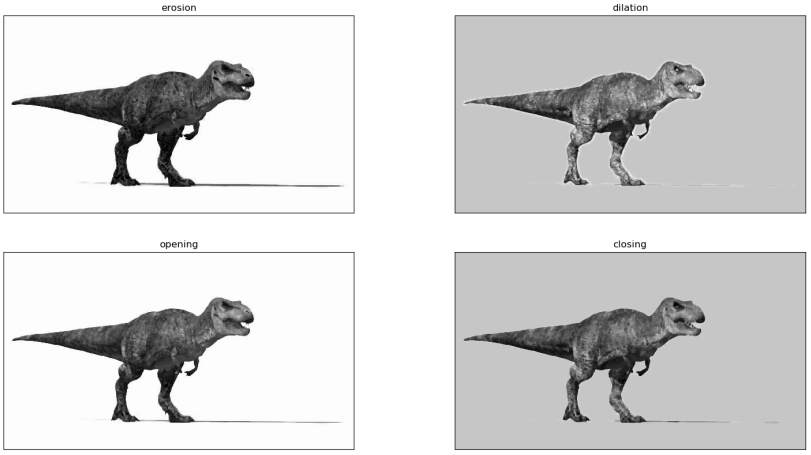


Figure 12: After Morphological Transformations Process

Normally, in cases like noise removal, erosion is followed by dilation (Opening). Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object[8]. The opposite operation is called Closing, when dilation followed by Erosion.

6 Arithmetic Operations on Images

It is the process of usage of arithmetic operations on images like addition, subtraction, bitwise operations, to create double-exposures, composites or finding changes between two images.

6.1 Subtraction

This operation work with the subtraction of pixel from to different frames. We can work with sequential frames what means that we can detected changes and distances between two frames. The output of this operation is a Third images that contains the pixel values that are simply those of the first image minus the corresponding pixel values from the second image.

If the pixel values in the input images are actually vectors rather than scalar values (e.g. for color images) then the individual components (e.g. red, blue and green components) are simply subtracted separately to produce the output value, as we can see at Figure 13.

Figure 13: Color images subtraction.

7 Changes Detection of a Plan

There is several methods that allow us to perform detection of plans Changes, one more performed than others, but the right pick always depends of the problematic, in this report we had the opportunity discuss about some of the most common between them:

- Image Subtraction;
- Image Segmentation;
- Median Filtering;

Using `Image Subtraction` we can easily detect changes from two different overlapped frames that have the same dimension/channels. The expected output of this method is a third images that contains the pixel values from the first image minus the corresponding pixel values from the second image, or from a constant.

Indeed we can performer detection of plans changes by combining Image Subtraction and Image Segmentation. That allow a great evidence of change by parser the vector of pixel to gray scale and combing it with THRESHBINARY or THRESH_TOZERO, depending of the case. It is this combination, that I judge, in this case, to be the most performed, because it is particular simple and the result is easily observable. The follow Figure 14 show it.

Figure 14: Tracking train movement with THRESH_TOZERO and Subtraction.

The function used was THRESH_TOZERO, because in the original video there are most white presence in train. So it was greater than the thresh value found. We got a clear distinction between the source and output pixels, even in a lightly environment.

Another approach is to use Median filtering for background formation, by appropriation of the middle element given a spatial location, in order to be able to subtracting changes from each set of values Pixel based[10]. That can be used to detection of a very small pixel changes e.g. measure movement in space.

8 Creating Images Resume of a Plan

Creation images resume of a plan is the first step to an analyses for objects detection. Called `Histogram of Oriented Gradients` this approach is based on several

techniques, in resume, this method is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information [5].

Figure 15: Original Frame before preprocessing step.

OpenCV has a function `HOGDescriptor` that converts an image of size width x height x 3 (channels) to a feature vector, however the main input to this function is a resume of image or simply a sample. In some cases it is useful apply robust algorithms to detect which part of this frame is more useful extract a sample.

Figure 16: Sample frame before processing (Left). Sample after Sobel gradient processing (Right).

In general this sample/resume frame is extracted from the original frame, and then segmented in patch respecting the aspect ratio of 1:2. It is important, because will

eliminate the overlapping of block that can cause redundancies into data vector affecting the final result.

9 Descriptor and list of samples a same plan

Here the frames is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells.

Figure 17: 8x8 Blocks to processing.

After this process of calculating the horizontal and vertical gradients that contains 2 values (magnitude and direction), each block of 8x8 will result a histogram that can be a parser to a vector of magnitude and direction.

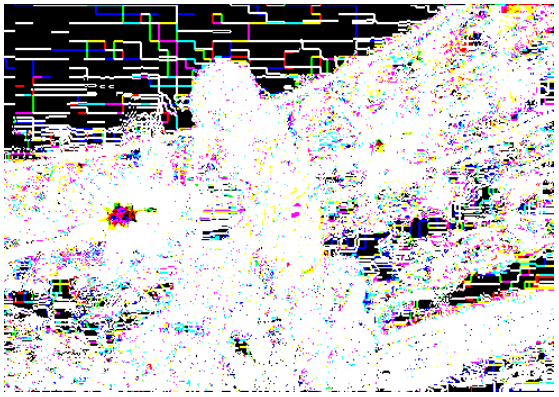


Figure 18: Horizontal and vertical gradients distribution in a frame.

The mechanism used on this report was HOG. It is simple and perform in a fast and easily way results from pieces of a frame. This pieces is called of patch and it provides a compact representation of the features of this frame[5].

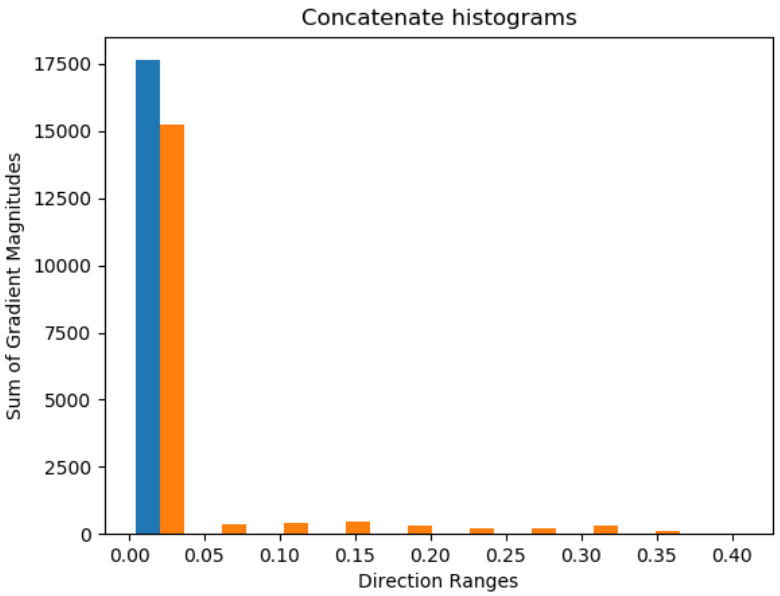


Figure 19: Concatenated and normalized histogram

From this operation several operation can be used such as plot histograms of magnitude, deep learning based SVM for detection of particular objects or person. This approach is particular useful because it maintain only important information, angles and contours, rising the possibility to identify form and deepness.

10 Calculation of SIFT descriptors and similarities

In order to calculate the similarities and also differences between two images using some distance calculation, SIFT (Scale-Invariant Feature Transform) is an algorithm that extract keypoints and compute its descriptors. This approach have 4 relevant points:

- Scale-space extrema detection, a difference-of-Gaussian function to identify potential interest points.
- Keypoint localization, model to determine location and scale.

- Orientation assignment, based on local image gradient directions, it get orientation, scale, and location for each feature.
- Keypoint descriptor, is a image representation measured.

On this report I tested two methods, Brute-Force and FLANN based Matcher, both are SIFT Descriptors based on ratio test.

Brute-Force Matcher consist into apply several best match for find any similarities with **knnMatch** techniques. It get k best matches for each descriptor from a query set[4].

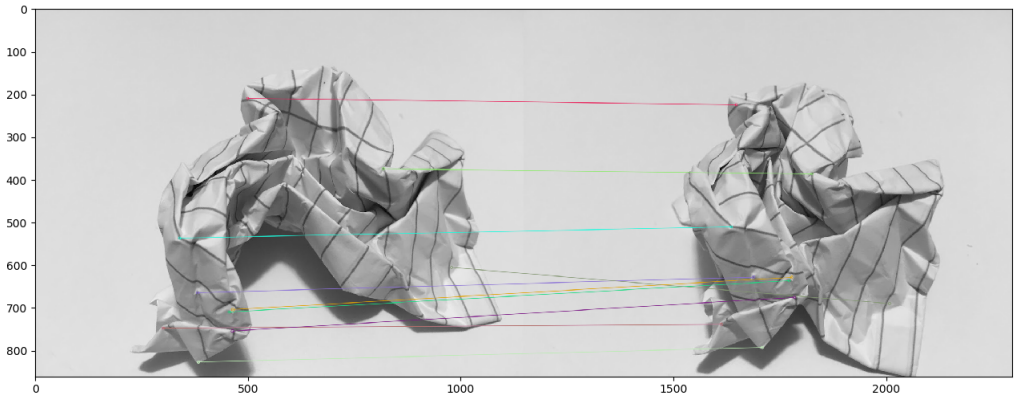


Figure 20: Brute-Force Matcher with a threshold of 0.5.

Built over Approximate Nearest Neighbors this approach contains a collection of algorithms optimized. It is more precisely than BF and in large dataset may be more fast, because this methods create a mask of identifies and only make matches if it is really good.

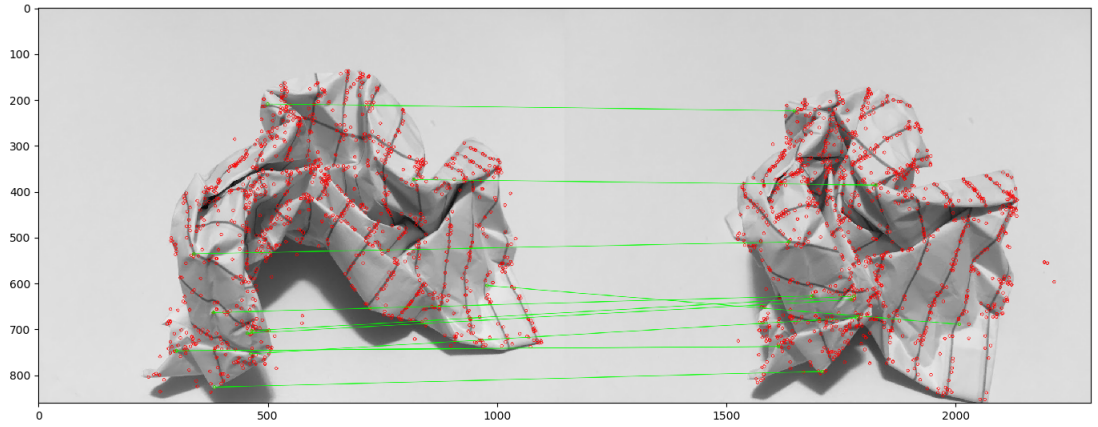


Figure 21: FLANN Matcher with a threshold of 0.5.

Thus, differences images should not show the matches.

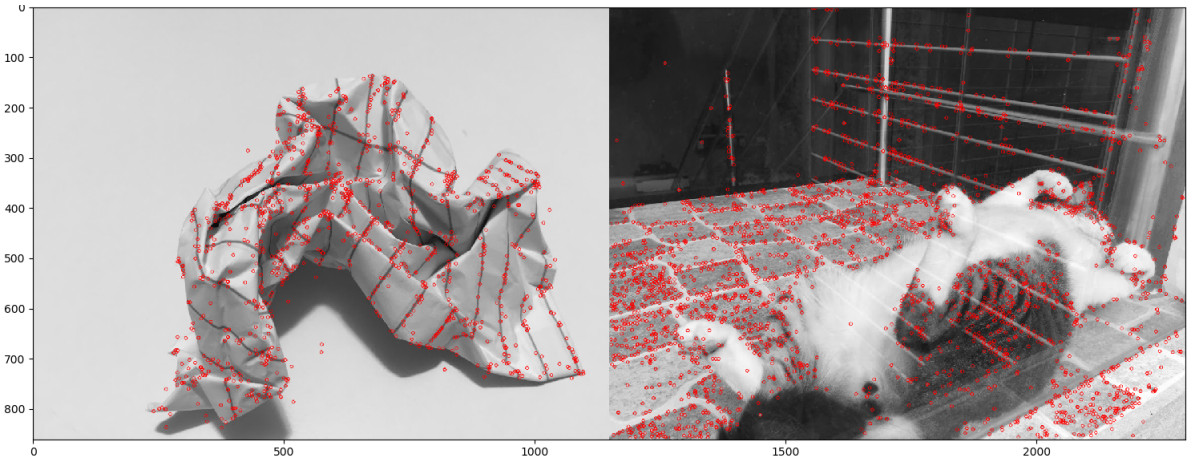


Figure 22: FLANN Matcher in a bad match.

Consequently, in order to get the best similarity possible, I chose to use FLANN Matcher. One time kPoint was determined it can be use to provide, based in a given threshold, not only matches, but also distinguish between sets or sequence of images.

11 Find the best similarity.

Based on the grubbed information from the last section, for this problem we need to give a dataset of images, and find the best similarity, eliminating, at least, images

that do not are none matches, that are called outliers. Thus, getting only good matches providing correct estimation, that are called inliers. In order to perform a calibration test, the image query is also into the sequence of 6 images.

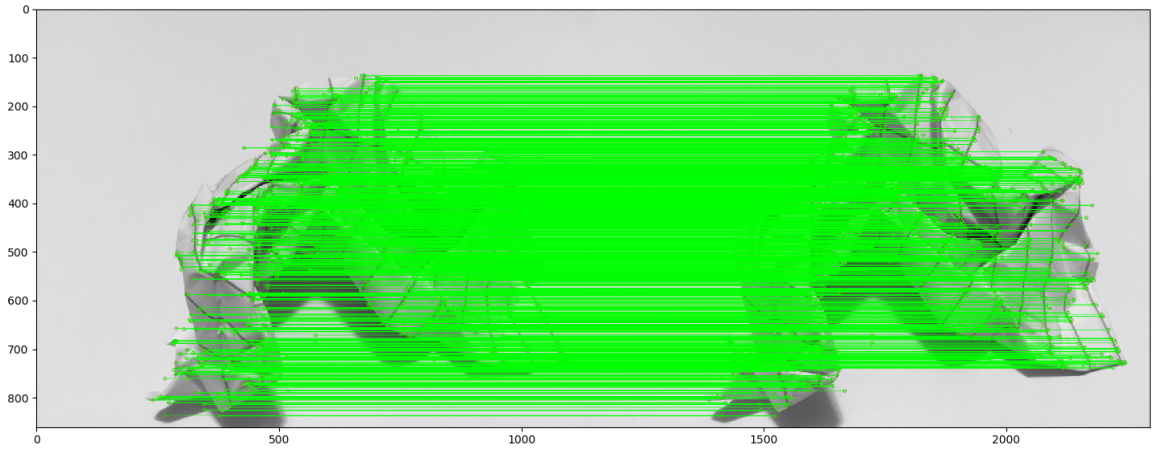


Figure 23: FLANN Matcher only inliers with flag RANSAC

After calibration of the test, using reshape and perform N count of the good matches. The query was refined, then the subset with a near similarity can be also founded without no problem.

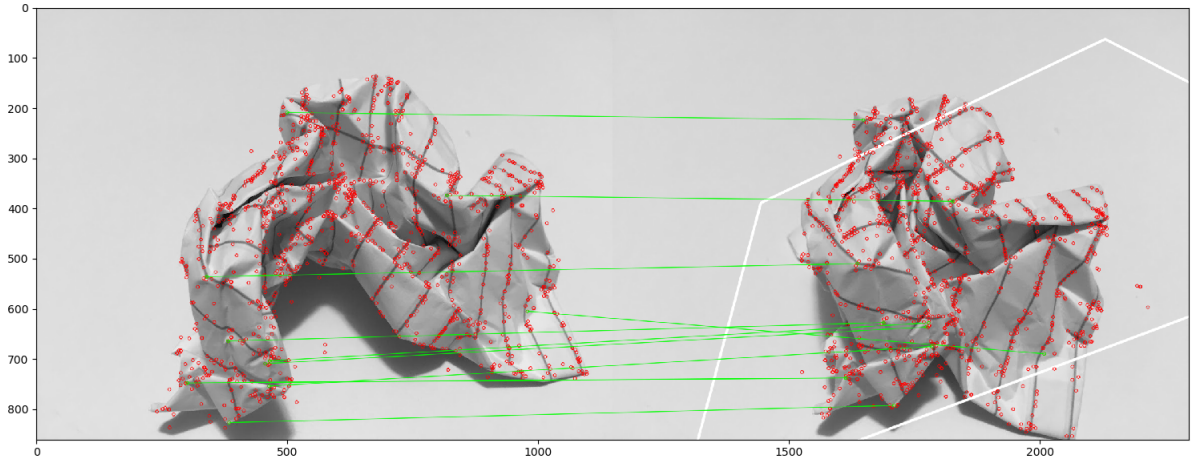


Figure 24: FLANN Matcher the subset with near similarity found.

Once with a 3x3 transformation matrix, the corners of query Image is transformed to corresponding points in train Image.

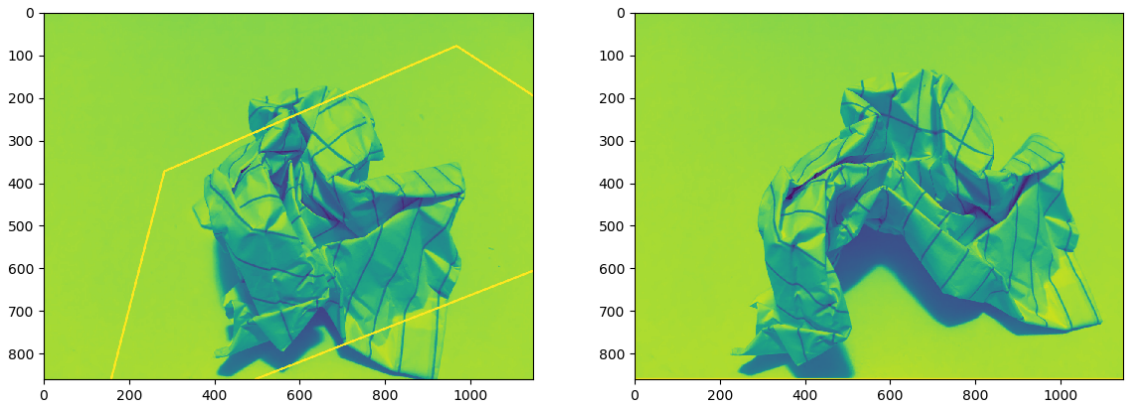


Figure 25: Query image (Right). Train image (Left)

12 Creation of samples and Calculation of SIFT descriptors and similarities of Plan

On this moment all things considered was take a resume of a frame and compare it with stucked images. It does not work perfectly, because of changes of plan, hardware limitation and luminosity. But it is possible to create and to use an image as a set of query, and then when the right frame get the best match correspondence with the rights points the plot of FLANN appears.

Figure 26: Origin video frame before the best match.

Although the size of this sample is too small than the frame of video, we can see that the 10 best matches appears clearly. It occurs because into resume picture we have

only the most important and also SIFT algorithms has a high tolerance to aspect ratio and others features, mainly if it normalized.

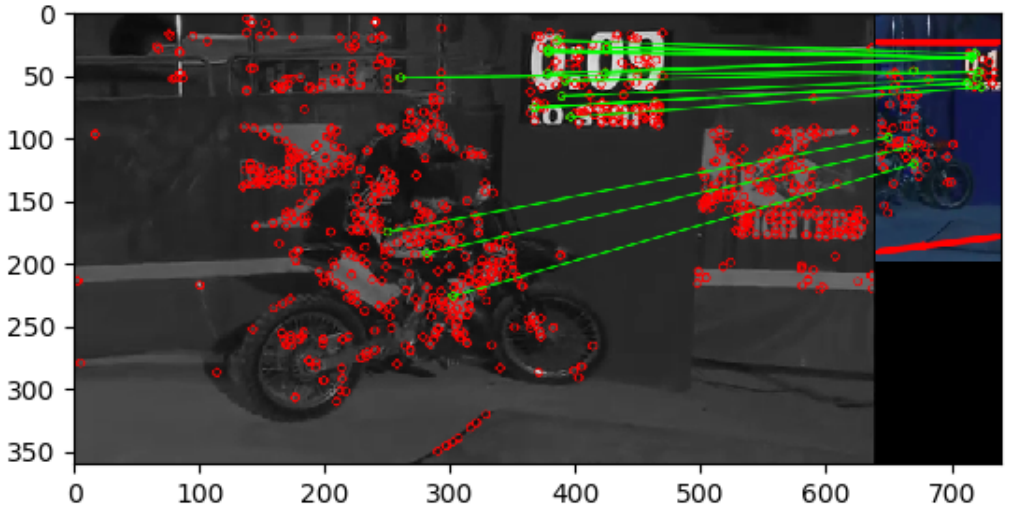


Figure 27: After match of the best 10 kpoint.

13 Conclusion

In conclusion OpenCV is a robust and power tool for processing video and images. Despite of all modules are not easily available in order to combine all function here implemented, in general is pretty simple to use opencv filters, image segmentation and operations.

However is not easy to understand and learn in a short time, although there are a lot of function, approach and techniques very fun and useful to machine learn field such as SVM, KNN, so on. In deed, I liked to work with magnitude and Direction in resume of plan was incredible awesome, it very easy to pass all night doing it.

The most difficult exercise was the last one because of the method protected by patent, that is not available on common packages, then when I do the downgrade to work with SIFT other method stops to work. The **error: This algorithm is patented and is excluded in this configuration;**, what is very disappointed. But the implementation was done, after a uninstallation of Opencv from conda, and then O did the installation from pip repository. All implementation and pictures and videos used within code and report are on my github [6].

References

[1] Binary image. https://en.wikipedia.org/wiki/Binary_image. accessed: 10.02.2019.

[2] Contours : Getting started. https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html. accessed: 10.02.2019.

[3] convolutions for deep learning. https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1. accessed: 03.02.2019.

[4] descriptor matchers. https://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_descriptor_matchers.html. accessed: 10.02.2019.

[5] histogram of oriented gradients. https://www.learnopencv.com/histogram-of-oriented-gradients/. accessed: 10.02.2019.

[6] Jesse ferreira filho github. https://github.com/jessefilho/opencv_td. accessed: 10.02.2019.

[7] Kernel image processing. https://en.wikipedia.org/wiki/Kernel_(image_processing). accessed: 03.02.2019.

[8] morphological ops. https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html. accessed: 10.02.2019.

[9] opencv. http://opencv.org. accessed: 03.02.2019.

[10] Pixel-based change detection methods. https://tel.archives-ouvertes.fr/tel-01510938/document. accessed: 10.02.2019.