# RECONOCIMIENTO DE FORMAS EN BASE A DETECCIÓN DE BORDES, BINARIZACIÓN POR UMBRAL, APLICACIÓN DE FILTROS Y OPERACIONES MORFOLÓGICAS

Christian Hernández,
Enero 2021.

Universidad Politécnica Salesiana.
Ingeniería de Sistemas.
Intelligence Artificial.

## Abstract

In recent years digital processing of images has gained an important place in the field of information technology and computing. Presently, it is the base of a growing variety of applications which includes: medical diagnosis, remote perception, space exploration, computer vision, etc. As a result of the reduction of computers prices, presently, the digital processing of images can be accomplished with a personal computer. The present paper offers a brief introduction to this area of computing. It refers to the main theories and methods, and shows the results of these theories and methods when used with certain images

# Capítulo 1

## Definición

El procesamiento digital de imágenes ha adquirido, en años recientes, un papel importante en las tecnologías de la información y el cómputo. Actualmente, es la base de una creciente variedad de aplicaciones que incluyen diagnosis médica, percepción remota, exploración espacial, visión por computadora, etc. Como resultado directo de la reducción en el precio de las computadoras, el procesamiento digital de imágenes actualmente se puede efectuar en una computadora personal. El presente trabajo proporciona una breve introducción a esta área de la informática y de la computación haciendo referencia a las principales teorías y métodos; asimismo, se muestran los resultados de estas teorías y métodos cuando se aplican a imágenes dadas

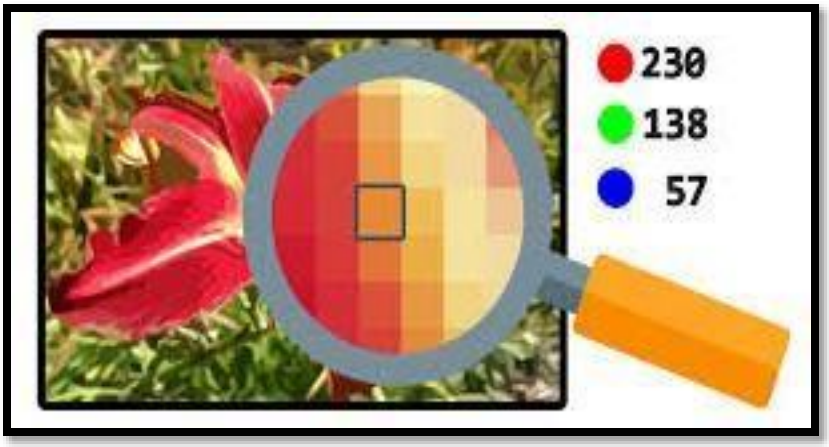



Figura1: Procesamiento de Imágenes

**Procesamiento Digital de Imágenes**

Es todo un conjunto de técnicas que tiene como objetivos, obtener una representación más adecuada de una imagen, o bien, es también utilizado en la extracción de características relevantes de esta, las cuales muchas veces no son perceptibles a primera vista

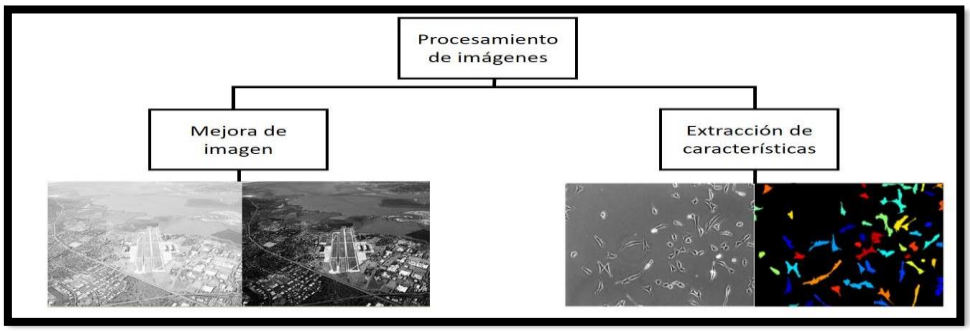



Figura2: Imágenes

**Ventajas**

- Método simple y sencillo de implementar.
- Fácil asociación del concepto de frecuencia con ciertas características de la imagen; cambios de tonalidad suaves implican frecuencias bajas y cambios bruscos frecuencias altas.
- Proporciona flexibilidad en el diseño de soluciones de filtrado.
- Rapidez en el filtrado al utilizar el Teorema de la Convolución.

**Desventajas**

- Se necesitan conocimientos en varios campos para desarrollar una aplicación para el procesamiento de imágenes.
- El ruido no puede ser eliminado completamente.

**Umbralización:** Un método básico para diferenciar un objeto del fondo de la imagen es mediante binarización

**Operaciones Morfológicas en imágenes binarias**

La morfología matemática es una herramienta muy utilizada en el procesamiento de imágenes. Las operaciones morfológicas pueden simplificar los datos de una imagen, preservar las características esenciales y eliminar aspectos irrelevantes. Teniendo en cuenta que la identificación y descomposición de objetos, la extracción de rasgos, la localización de defectos e incluso los

defectos en líneas de ensamblaje están suma-mente relacionados con las formas, es obvio el papel de la morfología matemática.

- La morfología matemática se puede usar, entre otros, con los siguientes objetivos: Preprocesamiento de imágenes (supresión de ruido, simplificación de for-mas).
- •Destacar la estructura de objetos (extraer el esqueleto, marcado de obje-tos, envolvente convexa, ampliación, reducción).
- •Descripción cualitativa de objetos (área, perímetro, diámetro, etc)

**ACTIVIDADES POR DESARROLLAR**

**Parte 1.** Desarrollar un programa que permita generar ruido de sal y pimienta y aplicar filtros para reducir dicho ruido. Para ello, deberá llevar a cabo:

1. Programar un *método* que genere un porcentaje de ruido de sal o pimienta en un video, considerando las dimensiones del mismo. Se deberá poder ingresar un porcentaje de ruido a través de dos *trackbars*
    a. (uno para sal y otro para pimienta).

**Video Original**

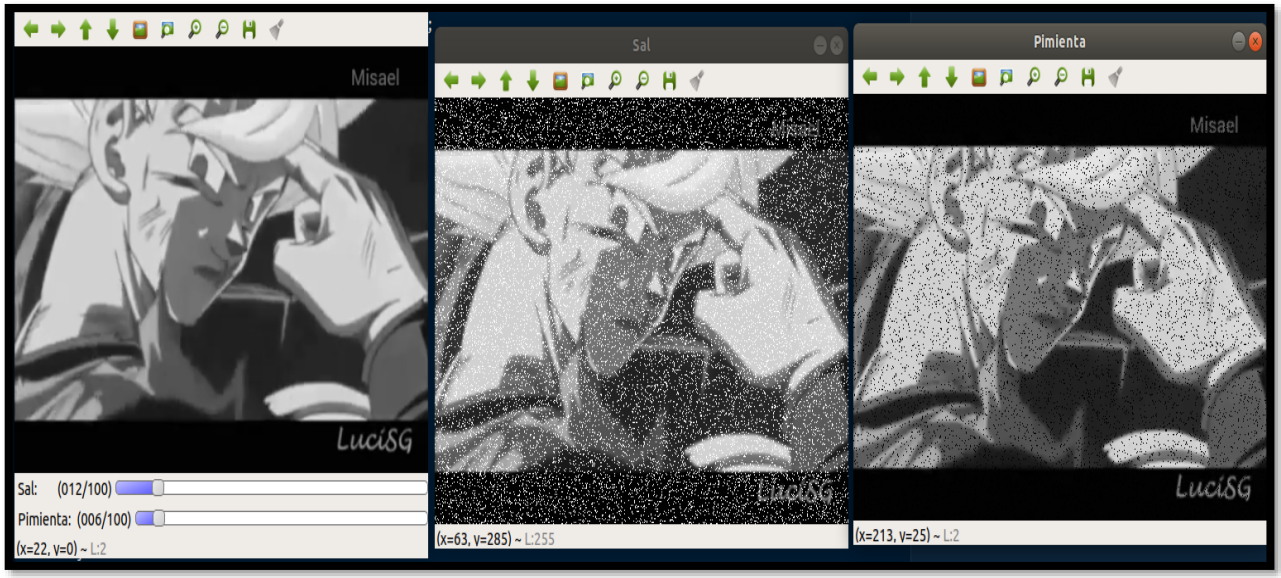
- Sal=012 Pimienta=006
- X=22 ,Y=0 L=2

**Sal**

- Sal=012 Pimienta=006
- X=63 ,Y=235 L=255

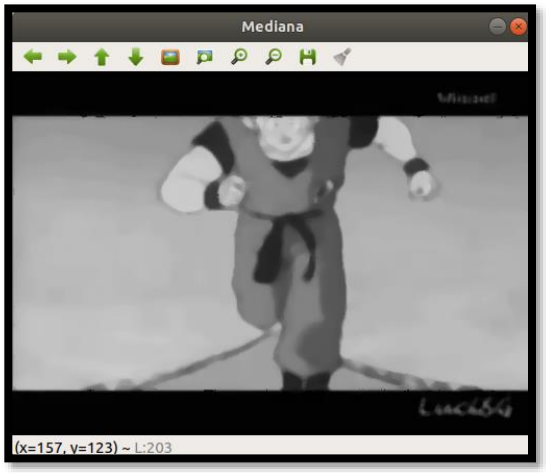
**Pimienta**

- Sal=012 Pimienta=006
- X=213 ,Y=25 L=2

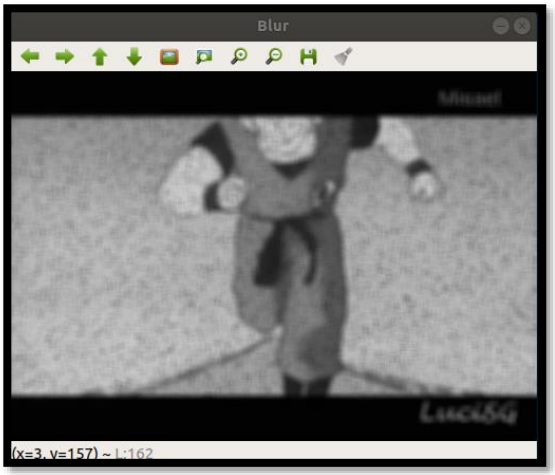


2. Programar una función para aplicar los siguientes filtros: mediana, *blur*, Gaussiano, probando con diferente tamaño de máscara.

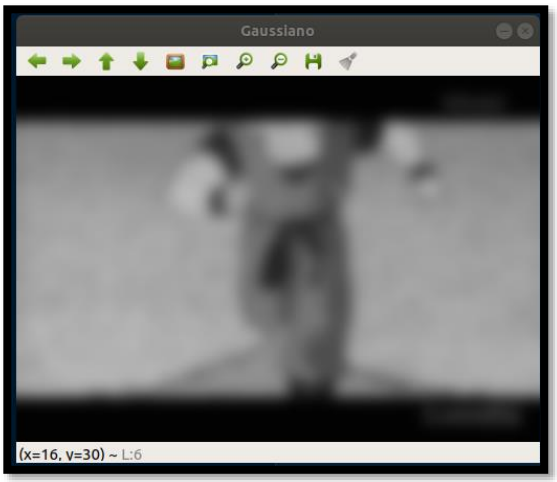
- **Mediana**
- X=157 ,Y=123 L=203




- ***Blur***
- X=3 ,Y=157 L=162



- **Gaussiano**
- X=16 ,Y=30 L=6



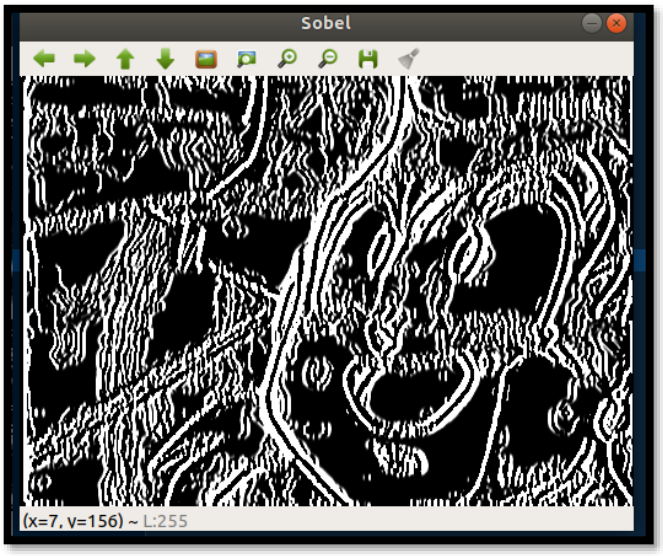

3. Compare los resultados obtenidos por cada filtro, y reflexione cuál ha obtenido mejores resultados

- **Mediana**
- X=157 ,Y=123 L=203

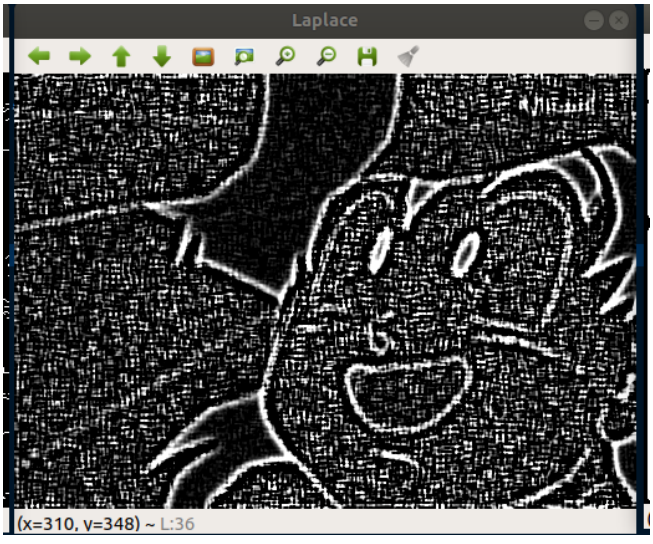

- El filtro de la mediana podemos ver que se observa con más claridad el video que se está proyectando teniendo resultados en x =157 ,y= 123.

4. Aplicar al menos 2 algoritmos de detección de bordes y comparar los resultados de usar o no filtros de suavizado

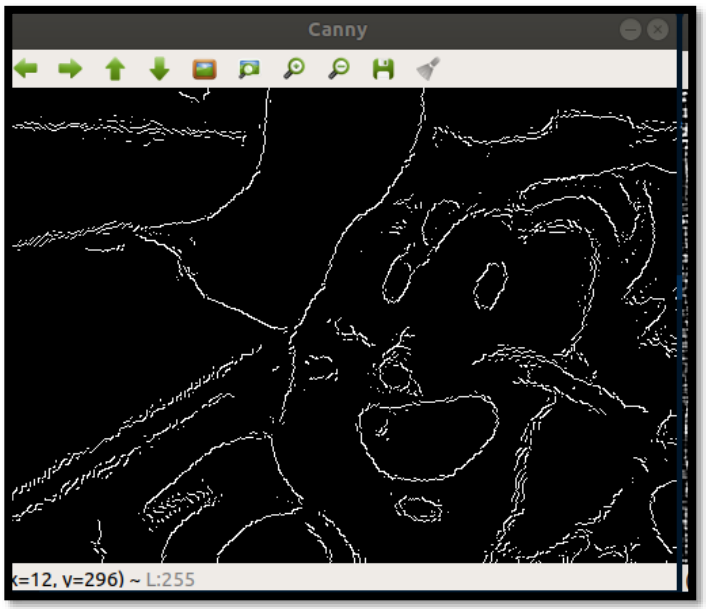
- **Sobel**
- X=7 ,Y=156 L=255




- **Laplace**
- X=310 ,Y=348 L=36

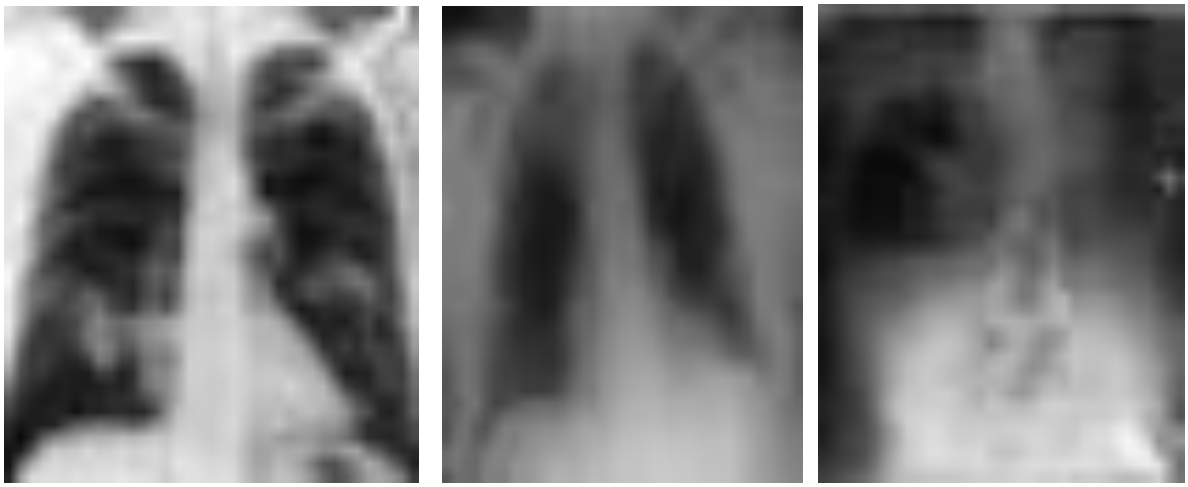


- **Canny**
- X=12 ,Y=296 L=255




**Parte 2**. Desarrollar un programa que permita aplicar operaciones morfológicas para mejorar la calidad de imágenes médicas, para ello deberá realizar las siguientes tareas:

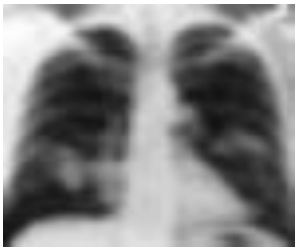
1. Seleccionar 3 imágenes médicas a las que se les aplicarán las operaciones morfológicas. Las imágenes deben estar en escala de grises y deben corresponder a radiografías, angiografías, TACs, etc.



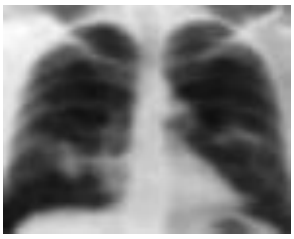
2. Aplicar las siguientes operaciones sobre las imágenes, probando al menos 3 tamaños de máscaras (de tamaño aproximado de 37x37, como se sugiere en el artículo “Using morphological transforms to enhance the contrast of medical images” ):

**Primera Imagen**

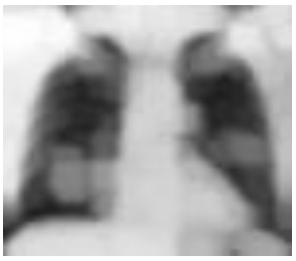
- Original



- Erosión



- Dilatación

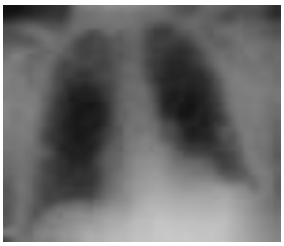


- Top Hat

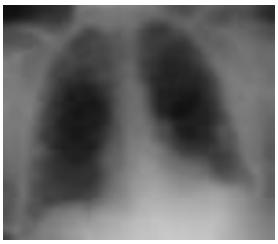
- Black Hat

**Segunda Imagen**

- Original

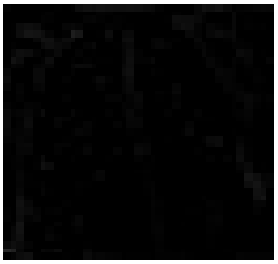


- Erosión



- Dilatación

- Top Hat



- Black Hat

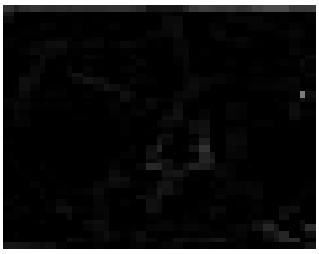
**Tercera Imagen**

- Original

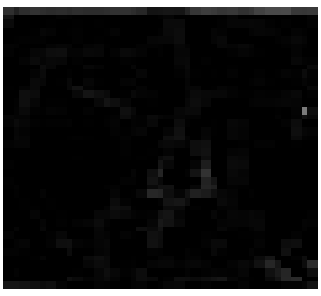
- Erosión

- Dilatación

- Top Hat



- Black Hat



## Código Parte 1

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <opencv2/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/imgcodecs/imgcodecs.hpp>
#include <opencv2/video/video.hpp>
#include <opencv2/videoio/videoio.hpp>

using namespace cv;
using namespace std;
Mat frame;
Mat sal;
Mat pimienta;
Mat mediana;
Mat blurv;
Mat gaussiano;
Mat laplace;
Mat canny;
Mat sobel;

int salV = 0;
int pimientaV = 0;


void functionTrackS(int v, void *p){
    cout << "Sal: " << v << endl;
}
void functionTrackP(int v, void *p){
    cout << "Pimienta: " << v << endl;
}

int main(int argc, char *argv[]){

    VideoCapture video("./goku.mp4");
    if(video.isOpened()){
        namedWindow("Video",WINDOW_AUTOSIZE);
        createTrackbar("Sal: ", "Video", &salV, 100, functionTrackS, NULL);
        createTrackbar("Pimienta: ", "Video", &pimientaV, 100, functionTrackP, NULL);
```

```cpp
  while(3==3){
      video >> frame;

      if(!frame.empty()){
          cvtColor(frame, frame, COLOR_BGR2GRAY);

          resize(frame, frame, Size(500,350));
          sal=frame.clone();
          pimienta=frame.clone();

          //Aplicando filtos de Sal
          int sx=0, sy=0;
          double porcentajeS = (double)salV/100;
          int totalS = ((int)(porcentajeS * ((double) frame.rows*frame.cols)));

          while(totalS>0){
              sx = rand() % frame.cols;
              sy = rand() % frame.rows;
              sal.at<uchar>(sy,sx) = 255;
              totalS--;
          }
          //Apliacando filtro de Pimienta
         int px=0, py=0;
          double porcentajeP = (double)pimientaV/100;
          int totalP = ((int)(porcentajeP * ((double) frame.rows*frame.cols)));

          while(totalP>0){
              px = rand() % frame.cols;
              py = rand() % frame.rows;
              pimienta.at<uchar>(py,px) = 0;
              totalP--;
          }
```

```
                //Aplicacion de Filtros
                //Mediana
                medianBlur(pimienta,mediana,7);

                //Blur
                blur(pimienta,blurv, Size(7,7));

                //Aplicaccion de filtro gaussiano
                GaussianBlur(pimienta,gaussiano,Size(),7);

               //Apliacando deteccion de Bordes
               int tresh =30;

               Canny(gaussiano, canny, tresh,tresh*5,5);
               Laplacian(blurv,laplace,CV_8UC1,5);
               Sobel(mediana,sobel,CV_8UC1,3,0,9);

               imshow("Video", frame);
               imshow("Sal", sal);
               imshow("Pimienta", pimienta);
               imshow("Mediana", mediana);
               imshow("Blur", blurv);
               imshow("Gaussiano", gaussiano);
               imshow("Canny",canny);
               imshow("Laplace",laplace);
               imshow("Sobel",sobel);

                if(waitKey(100)==27)
                    break;
            }else
                break;
        }
    }
    destroyAllWindows();
    return 0;
}
```

Código Parte 2

```
#include <iostream>
#include <fstream>
#include <opencv2/opencv.hpp>

using namespace cv;

int main(int argc, char** argv )
{
    Mat src, dst,dst1,dst2 ,dst3, kernel,kernel1;

    src = cv::imread("imagen3.jpg");
     resize(src, src, Size(),1,1);


    //Dilatacion
    kernel = cv::getStructuringElement(MORPH_RECT, Size(5, 5));
    cv::dilate(src, dst, kernel);

    //Erosion
    kernel = cv::getStructuringElement(MORPH_ELLIPSE, Size(3, 3));
   cv::erode(src, dst1, kernel);
    //TOP HAT
   cv::morphologyEx(src, dst2, MORPH_TOPHAT, kernel1);

    //TOP BLACK
   cv::morphologyEx(src, dst3, MORPH_BLACKHAT, kernel1);

    cv::imshow("Original", src);
    cv::imshow("Dilatacion", dst);
    cv::imshow("Erosion", dst1);
    cv::imshow("Top Hat", dst2);
    cv::imshow("Black Hat", dst2);
    cv::waitKey(0);
  destroyAllWindows();
    return 0;
}
```

**Conclusiones**

Aplicar los aspectos relacionados con las operaciones morfológicas apertura, cierre, erosión, dilatación, Top-Hat, Black-Hat, la ecualización de histograma y los elementos matemáticos usados en los filtros/kernels tanto para suavizado como para detección de bordes.

**Recomendaciones**

- Haber asistido a las sesiones de clase.
- Consultar con el docente las dudas que puedan surgir al momento de realizar la práctica
- Tener opencv para poder realiza la actividad que son envidas
-

**Referencia**


- Cuesta, U., Niño, J. y Rodríguez, J. (2017). The Cognitive Processing of an Educational App with Electroencephalogram and "Eye Tracking". Comunicar, 25(52), 41-50. DOI: 10.3916/ C52-2017-04
- Dalal, N., y Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. En 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 1, 886-893). IEEE. DOI: 10.1109/CVPR.2005.177
- Euromonitor International. (Agosto del 2017). Digital Consumer Profiles: How Latin Americans will shop and spend digitally. Análisis del consumidor. Base de datos Euromonitor International.
- Hidalgo, I., y Sánchez, R. (2015). Reconocimiento de caracteres mediante imágenes en contadores de gas en entornos reales (Trabajo de fin de grado). Universidad Complutense de Madrid, España.
- Kaur, S. (2016). An automatic number plate recognition system under image processing. International Journal of Intelligent Systems and Applications, 8(3), 14-25. DOI: 10.5815/ijisa.2016.03.02