|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | **PRÁCTICA DE LABORATORIO** | |
|  | | | | |
| **CARRERA**:Computacion | | | | **ASIGNATURA**:VISIÓN POR COMPUTADOR |
| **NRO. PRÁCTICA**: | 3.1 | **TÍTULO PRÁCTICA**:Ejercicio Participación Clase: Operaciones con imágenes | | |
| **OBJETIVO ALCANZADO: Pixelear un triangulo en negativo su interior** | | | | |
| **ACTIVIDADES DESARROLLADAS** | | | | |
| **1.Codigo Fuente:**  **#include <iostream>**  #include <cstdlib>  // Librerías de OpenCV  #include <opencv2/core/core.hpp> // Contiene las definiciones base de matrices y estructuras  #include <opencv2/highgui/highgui.hpp> // Interfaz gráfica de usuario  #include <opencv2/imgproc/imgproc.hpp> // Procesamiento de imágenes  #include <opencv2/imgcodecs/imgcodecs.hpp> // Códecs para leer los distintos formatos de imágenes (JPG, PNG, etc.)  #include <opencv2/video/video.hpp> // Lectura de vídeos  #include <opencv2/videoio/videoio.hpp> // Lectura y escritura de videos  #include <opencv2/objdetect/objdetect.hpp> // Para realizar detección de objetos  #include <random>  #include <cmath>  //#include <opencv2/opencv.hpp> // Permite cargar todas las librerías de OpenCV  using namespace std;  using namespace cv; // Espacio de nombres de OpenCV para llamar a los métodos de forma más directa  int main(int argc, char \*argv[]){  int puntos[2]={-1,-1};  namedWindow("Original", WINDOW\_AUTOSIZE);  namedWindow("Muestra", WINDOW\_AUTOSIZE);  namedWindow("Resultado", WINDOW\_AUTOSIZE);  Mat imagenOriginalTriangulo = imread("SIMON\_BOLIVAR.png");  Mat imagenTrianguloResul = imread("SIMON\_BOLIVAR\_ORIGINAL.png");  Mat imagenOriginal = imread("SIMON\_BOLIVAR\_ORIGINAL.png");  Vec3b pixel;  Vec3b pixel1;  bool bandera=false;  for(int i=0;i<imagenOriginalTriangulo.cols;i++){  bandera=false;  puntos[0]=-1;  puntos[1]=-1;  for(int j=0;j<imagenOriginalTriangulo.rows;j++){  pixel = imagenOriginalTriangulo.at<Vec3b>(j,i);  pixel1 = imagenTrianguloResul.at<Vec3b>(j,i);  //Comparacion en cada rango de color  if(pixel1[0]!=pixel[0]||pixel1[1]!=pixel[1]||pixel1[2]!=pixel[2]){  //si no coincide la imagen con triangulo con la imagen original guarda el punto  if(bandera==false){  puntos[0]=j;  bandera=true;  } else {  puntos[1]=j;  }  }  }  if(puntos[1]==-1){  puntos[1]=puntos[0];  }  //Valida si los vectores de los puntos estan vacios  if(puntos[0]!=-1&&puntos[1]!=-1){  //pinda o cambia el pixel a negativo recorriendo los puntos obtenidos de la comparacionde imagenes  for(int b=puntos[0];b<=puntos[1];b++){  Vec3b pixel3 = imagenTrianguloResul.at<Vec3b>(b,i);  Vec3b pixelNegativo = Vec3b(255-pixel3[0],255-pixel3[1],255-pixel3[2]);  //Coloca el pixel en negativo en la imagen resultado  imagenTrianguloResul.at<Vec3b>(b,i) = Vec3b(pixelNegativo);  }  }  }  //Muestra las imagenes  imshow("Muestra",imagenOriginalTriangulo);  imshow("Resultado",imagenTrianguloResul);  imshow("Original",imagenOriginal);  waitKey(0);  } | | | | |
| **2.Imagenes Original y ejercicio:**    **3. Imagen resultant:** | | | | |
| **RESULTADO(S) OBTENIDO(S)**:  Se realizo la practica correctamente con la edicion de pixeles. | | | | |
| **RECOMENDACIONES**:  **Se recomienda la comparacion de imagenes, ya que permite hacer el recorrido de forma mas limpia.** | | | | |
|  | | | | |

***Nombre de estudiante*: Willan Mendieta**