

Programación Orientada a Objetos

Dr. Marco Aurelio Nuño Maganda

Universidad Politecnica de Victoria
Ingeniería en Tecnologías de la Información
Cuatrimestre Mayo - Agosto 2024

mnunom@upv.edu.mx

July 1, 2024

Breve CV del Facilitador

- Doctor en Ciencias Computacionales por parte del INAOE (2009).
- Profesor de Tiempo Completo de la UPV desde 2009.
- Miembro del Sistema Nacional de Investigadores - Nivel Candidado (2014-2016), Nivel I (2020-2022), Nivel I (2023-2027)
- 17 tesis dirigidas a nivel maestría.
- Asignaturas impartidas en el pasado
 - Licenciatura: Cómputo en Dispositivos Móviles, Graficación por Computadora Avanzada, Lenguajes y Automatas, Programación Orientada a Objetos
 - Maestría: Visión por computadora, Tópicos Selectos de Imagenología, Fundamentos de Sistemas de Información
- Miembro del Núcleo Académico Básico (NAB) de la maestría en Ingeniería de la UPV.

Horario de la Clase

■ Días y horas de clase

	Lunes	Martes	Miércoles	Jueves	Viernes
iti-271221		13:00 - 14:55	13:00 - 14:55	13:00-13:55	13:00-13:55

■ Fechas Importantes:

- Inicio de Cursos: 29/Abril
- Fin de Cursos: 23/Agosto (16/Agosto)
- Dias no hábiles oficiales: 1 de mayo (miercoles), 22 al 26 de julio, 29 de julio al 2 de agosto (Vacaciones).

Plataforma Virtual para el Curso

- Nombre de la clase: **Programación Orientada a Objetos- Mayo - Agosto 2024**
- Código de clase en Classroom: **k5zirbz**
- Enlace Meet para sesiones no presenciales:
<https://meet.google.com/akj-srks-egv>

Reglas básicas

- Se recomienda puntualidad y asistencia a las sesiones.
- Respecto hacia el profesor y hacia sus compañeros y compañeras.
- No se permite el ingreso y/o ingestión de **Alimentos** ni **Bebidas** de ningún tipo a la clase.
- No se permite usar **AUDIFONOS O DISPOSITIVOS MANO-LIBRES EN CLASE. De detectar esta situación, se amonestará al estudiante y de reiterar, dicho estudiante será expulsado de CURSO por el resto del CUATRIMESTRE, sin derecho a réplica.**

Uso del Teléfono Inteligente

- Se recomienda no utilizarlo durante el transcurso de la clase. Depende del comportamiento del grupo que esto no sea aplicado...

Resguardo del teléfono inteligente

De ser necesario, se solicitará al INICIO de la CLASE a todos los asistentes a la clase (incluyendo al profesor) guardar su telefono en una caja, la cual será cerrada, regresando su telefono al finalizar la SESION.



Pase de Lista

- Se pasa lista al inicio de la clase. En caso de reincorporación tardía, se pone un retardo.
- DOS RETARDOS equivalen a una INASISTENCIA, que no es JUSTIFICABLE.
- Para justificar una inasistencia, es necesario cumplir con los siguientes pasos:
 - Agendar una asesoría de la clase mediante el SIITA. Una vez hecho esto, solicitar al profesor confirmación para actualizar su registro de inasistencia de tal día en la lista de asistencia de la clase.
 - En el TEMA de la ASESORIA debe poner **“JUSTIFICACION DE INASISTENCIA DEL DIA X/YY/ZZZZ”**. De no hacer lo anterior, no será considerada dicha justificación.
 - **DEBEN** agendar una asesoria por cada fecha de **INASISTENCIA (5 faltas, 5 asesorias)**.
 - **NO ES NECESARIO ENVIAR** correo electrónico al profesor –

Alumnos con Empleo (1)

- Al NO alcanzar un 80% de asistencia, el estudiante pierde su derecho de ser EVALUADO

Alumnos VIPs

En caso de tener un empleo formal dentro o fuera de la ciudad, es necesario entregar una **constancia laboral** que acredite el horario que se esta cubriendo (en el caso de locales, este horario se debe empalmar con el de la materia). En esa constancia debe acreditar que se esta haciendo labores de manera presencial en tal ubicacion. Esto lo dispensa solo del requisito de las asistencias, mas no de los proyectos que deban entregarse. Incluso pudiera solicitarle presentar avance de manera “remota” durante alguna de las clases. Enviar esa constancia con copia para el director de carrera.

Alumnos con Empleo (2)

- La justificación de inasistencias por *actividad laboral* se considerará a partir del momento de la recepción de dicha constancia en el correo del instructor (y no a partir de la fecha indicada en la constancia), por lo que si se recibe de manera tardía (con mas de una semana de retardo), dichas inasistencias NO SERAN justificadas.
- La justificación será válida si el estudiante programa **POR LO MENOS** dos asesorías por semana. De no hacerlo, pierde el beneficio de la justificación y se aplican las reglas anteriormente establecidas.

Unidades

- 1 Manejo de Errores y Excepciones
 - 1 Errores y Excepciones
 - 2 Manejo de Errores y Excepciones
- 2 Manejo de Objetos Gráficos
 - 1 Componentes Gráficos
 - 2 Librerías
 - 3 Manejo de Eventos
- 3 Concurrencia
 - 1 Hilos
 - 2 Concurrencia y Sincronización
- 4 Programación para Red
 - 1 Sockets
 - 2 Conexión a Base de Datos

Evaluación (1)

- Para cada unidad del curso, se consideran 3 aspectos:
 - Ejercicios o investigaciones especiales (1)- 25%
 - Proyecto Individual - 35%
 - Proyecto en Equipo - 40%
- Para aprobar el curso, es obligatorio:
 - Tener calificación aprobatoria en todas las unidades (100-100-40 no da calificación aprobatoria).
 - Tener por lo menos dos asesorías por semana (Registrarlas por semana, no 30 asesorías al final del cuatrimestre)
 - Cumplir con el 80% de asistencia mínimo, incluyendo aquellas inasistencias justificadas debidamente mediante el SIITA

Evaluación (2)

Para cada unidad, habra sesiones de “teoria”, sesiones de seguimiento de proyectos y sesiones de esparcimiento

- En las sesiones de teoria, el profesor presentara uno o varios temas
- En las sesiones de seguimiento de proyectos, de manera aleatoria se nombrara al integrante de equipo individual o en equipo. En el caso de que un integrante individual no responda, se le bajarán 5 puntos a su calificación del proyecto
- En las sesiones de esparcimiento, se permitirá a los estudiantes trabajar en proyectos pendientes, pero se contabilizará la asistencia.

Evaluación (3)

Sesiones de Seguimiento de proyectos

- En el caso de que el integrante del equipo seleccionado aleatoriamente no responda satisfactoriamente lo cuestionado, se le bajaran 5 puntos a su calificación del proyecto a todos los integrantes del equipo
- En el caso de los proyectos en equipo, el integrante seleccionado es aleatorio. Si en una primera ronda le toco al integrante A, en una segunda ronda posiblemente le toque al integrante B

Evaluación (4)

Lo que se debe presentar en una sesion de seguimiento de proyectos

- En un trabajo individual
 - Compartir pantalla de la ejecucion del avance del proyecto
 - Explicar con recursos multimedia los pasos para la resolucion del proyecto
 - Establecer el avance desde la ultima entrega
- En un trabajo grupal
 - Compartir pantalla de la ejecucion del avance del proyecto
 - Explicar con recursos multimedia los pasos para la resolucion del proyecto
 - Desglosar como se repartio el trabajo entre los integrantes del equipo
 - Establecer el avance desde la ultima entrega

Evaluación (5)

Acerca de los proyectos

- Aleatorios y DIFERENTES para la mayoría (preferentemente para cada integrante)
- Equipos: Proyectos diferentes para cada equipo, e Integrantes de los mismos formados de manera ALEATORIA!!

Fragmentación de equipos

- Si llegar a ocurrir que en un proyecto en equipo no hay un acuerdo para trabajar en equipo (Hay dos o mas entregas del proyecto asignado por partes diferentes dentro del mismo equipo)

Penalización

Cada “fragmento” de equipo recibe una penalizacion de 25 puntos mas las penalizaciones acumuladas por otros rubros.

- esta regla **NO APLICA** cuando hay uno o varios “desertores” del equipo (y hay una sola entrega del proyectos en equipo)

Acerca de Exención

- Cuando el profesor realizar alguna mecánica para excentar un proyecto (Individual/Equipo/Asignación especial) y uno o varios estudiantes completan lo solicitado, existen dos posibilidades:
 - El estudiante acepta excentar la elaboración de dicho proyecto o actividad, pero al hacer esto asume que la calificación asignada es 70.
 - El estudiante decide hacer el proyecto a pesar de haber excentado. En este caso el estudiante se hace acreedor a 20 puntos que puede aplicar sobre la calificación de dicho proyecto.

Cartucho de Recuperación (REC)

- Estudiante tiene derecho a solicitar un ÚNICO proyecto de recuperación aplicable a un solo proyecto o actividad.
- Esta solicitud debe HACERLA es estudiante - El profesor NO ES RESPONSABLE de informar al estudiante cuando tiene un ADEUDO.
- Si el proyecto no entregado es individual, se asigna otro proyecto diferente.
- Si el proyecto es en equipo, de común acuerdo con los integrantes pueden trabajar en otro proyecto diferente en equipo, o recibir una asignación individual de un proyecto diferente.
- La calificación recuperada será asignada siempre y cuando cumpla con el porcentaje de falta mínimo necesario para aprobar. Además, debe haber agendado el % de asesorías proporcional al tiempo de cuatrimestre transcurrido.
- El nuevo proyecto asignado esta diseñado para que el estudiante invierta en él por lo menos 1 SEMANA. Si lo solicita un día antes de terminar el cuatrimestre, posiblemente no tendrá tiempo de llevarlo a cabo.

Reporte Técnico de Desarrollo de Práctica

- Para cada práctica realizada, entregar un documento (**únicamente en formato PDF***) con las siguientes secciones:
 - Introducción
 - Desarrollo Experimental
 - Resultados
 - Conclusiones
 - **Referencias**
- Para GENERAR este reporte es necesario utilizar la plantilla en LATEX (**únicamente usando LATEX***) localizada en el siguiente enlace:
<https://www.overleaf.com/read/dgkhvfwnygvc>

Reporte Técnico de Desarrollo de Práctica

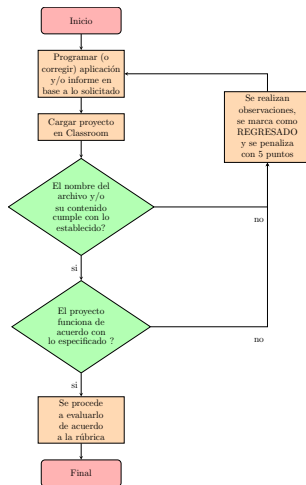
- Bajo ninguna circunstancia deben incluir **CÓDIGO FUENTE**. Si pueden incluir diagrama de flujo, Pseudocódigo, Diagrama E-R, Diagrama de Clases, de Casos de USO, etc. De incluir código fuente, solo tendrá un 50% del valor en la calificación.
- En caso de trabajos individuales o en EQUIPO, deben emplear la plantilla LaTeX que se provee. En caso de utilizar algo diferente a LaTeX u otra plantilla de LaTeX, la calificación proporcional del informe será **DESESTIMADA**.
- En caso de trabajos en equipo, se debe agregar los integrantes al inicio del INFORME. **El trabajo solo cuenta para aquellos integrantes mencionados en el informe (y que dicho nombre se encuentre registrado tal cual en la lista). Una vez ENTREGADO, si hay OMISIONES de los integrantes, no se realizará CORRECCION alguna, se debe asumir la consecuencias que esto conlleva.**

Ponderación del Informe en la Calificación del Proyecto

- Proyecto: 66 Puntos
 - Ejecución y Funcionalidad: 45 Puntos
 - Modularidad: 13 Puntos
 - Documentación: 8 Puntos
- Informe: 34 Puntos
 - Uso adecuado de Latex: 5 Puntos
 - Organización y Redacción: 6 Puntos
 - Referencias en formato adecuado: 8 Puntos
 - Evidencia del trabajo realizado: 8 Puntos
 - Sin faltas de ortografía ni errores de dedo: 7 Puntos

Flujo de Evaluación

- Proyecto que no este entregado de acuerdo con las especificaciones, será regresado (y penalizado)
- Puede dar tantas vueltas como el estudiante desee
- Se recomienda LEER con cuidado la sección de entregables en esta diapositiva



Entregables de proyecto individual (1)

- Crear un archivo ZIP con el siguiente formato de nombre:
 - **iti-271221_uX_nuno_maganda_marco_aurelio**
- Dentro, debe contener lo siguiente:
 - **iti-271221_uX_nuno_maganda_marco_aurelio_source** (Carpeta con código fuente de la aplicación)
 - **iti-271221_uX_nuno_maganda_marco_aurelio_latex** (Carpeta con código fuente del informe)
 - **iti-271221_uX_nuno_maganda_marco_aurelio.apk** (Instalable)
 - **iti-271221_uX_nuno_maganda_marco_aurelio.pdf** (Informe)
- Donde:
 - **X** es el número de unidad a un dígito (1, 2, etc)
 - **Sustituir con sus apellidos y nombres de manera apropiada**

Entregables de proyecto individual (2)

- En el caso que un proyecto individual sea asignado en equipo a varios estudiantes, el archivo entregable DEBE MANEJARSE como la de un proyecto individual
 - Solo un integrante del equipo carga en la plataforma el entregable individual.
 - El informe debe llevar los nombres de los integrantes del equipo que trabajaron (Si se omite a alguien, se asume que no trabajo en el proyecto).
 - NO ES NECESARIO que los otros integrantes marquen en el sistema la tarea como entregada, ya que se conoce su situación desde que se asigna el proyecto. El profesor ya sabe que ustedes van en equipo con el estudiante que hizo la entrega, y por eso deben asegurarse que en el informe entregado, vayan anotados sus nombres.

Entregables de proyectos en equipo

- Crear un archivo ZIP con el siguiente formato de nombre:
 - **iti-271221_eq_NN_uX**
- Dentro, debe contener lo siguiente:
 - **iti-271221_eq_NN_uX_source** (Carpeta con código fuente de la aplicación)
 - **iti-271221_eq_NN_uX_latex** (Carpeta con código fuente del informe)
 - **iti-271221_eq_NN_uX.pdf** (Informe)

Donde:

- **NN** es el número de equipo a dos dígitos (01, 02, etc)
 - **X** es el número de unidad a un dígito (1, 2, etc)
- En cada entrega, **UN SOLO INTEGRANTE DEL EQUIPO** deberá cargar los siguientes archivos en la carpeta asignada por Github para trabajar

Entregables de asignaciones especiales

- Crear un archivo ZIP con el siguiente formato de nombre:
 - **iti-271221_aeX_uY_nuno_maganda_marco_aurelio**
- Dentro, debe contener lo siguiente:
 - **iti-271221_aeX_uY_nuno_maganda_marco_aurelio_source** (Carpeta con código fuente de la aplicación - Cuando aplique)
 - **iti-271221_aeX_uY_nuno_maganda_marco_aurelio_latex** (Carpeta con código fuente del informe o diapositivas)
 - **iti-271221_aeX_uY_nuno_maganda_marco_aurelio.apk** (Instalable - Solo aplicaciones móviles)
 - **iti-271221_aeX_uY_nuno_maganda_marco_aurelio.pdf** (Informe)
- Donde:
 - **X** es el número de asignación dentro de la unidad a un dígito (1, 2, etc)
 - **Y** es el número de unidad a un dígito (1, 2, etc)
 - **Sustituir con sus apellidos y nombres de manera apropiada**

Nombres de Archivos Entregables

En el caso de nombres y apellidos acentuados, con diéresis o con virgulilla (~), sustituir de acuerdo con las siguientes reglas:

- Sustituir N/n por Ñ/ñ
- Sustituir A/a por Á/á
- Sustituir E/e por É/é
- Sustituir I/i por Í/í
- Sustituir O/o por Ó/ó
- Sustituir U/u por Ú/ú
- Sustituir U/u por Ü/ü

Camino a la Morici3n...

Asignacion Especial U1 - Thunar

Archivo Editar Ver Ir Marcadores Ayuda

< > ^ Home marco Escritorio 2024_LA Asignacion Especial U1 16_Asignacion Es... UEZ ALEJANDRA

Nombre	Tamaño	Tipo	Fecha de modificaci3n
15_Iti_271154_u1_olivares_rodriguez_brayan	4.0 KiB	carpeta	hoy
16_Asignacion Especial U1_RODRIGUEZ ALEJANDRA	4.0 KiB	carpeta	hoy
Asignaci3n Especial 7-1.zip	722.3 KiB	archivador Zip	19/02/24
Asignaci3n Especial Adri3n.zip	791.1 KiB	archivador Zip	26/02/24
Asignacion Especial U1_RODRIGUEZ ALEJANDRA.zip	3.7 MiB	archivador Zip	26/02/24
Asignaci3n Especial U1 _ Diapositivas C++ (134 - 183) _ Trevi3oGandarilla.zip	146.0 KiB	archivador Zip	12/03/24
Asignaci3n Especial U1 DL.zip	3.5 MiB	archivador Zip	27/02/24
Asignaci3n Especial U1 DL(1).zip	255.9 KiB	archivador Zip	hoy
Asignaci3n Especial U1 Stacks DL.zip	3.5 MiB	archivador Zip	27/02/24
Asignacion_Especial_Melchor.zip	3.9 MiB	archivador Zip	24/02/24
C++ Palmero (86 - 133).zip	478.7 KiB	archivador Zip	27/02/24
C++ Palmero (86 - 133)(1).zip	478.7 KiB	archivador Zip	27/02/24
Diapositivas C++ (1 a 41) - P3rez Omar.zip	330.0 KiB	archivador Zip	27/02/24
Iti-271154 u1 MARTINEZ HERRERA JOSE GUADALUPE latex.zip	5.1 MiB	archivador Zip	26/02/24
Iti-271154_u1_Cantu_Sanchez_Abraham_Isai.zip	3.7 MiB	archivador Zip	29/02/24
Iti-271154_u1_Cantu_Sanchez_Abraham_Isai(1).zip	3.7 MiB	archivador Zip	29/02/24
Iti-271154_u1_coyoy_lopez_mario.zip	3.4 MiB	archivador Zip	27/02/24
Iti-271154_u1_moreno_ledesma_ximena_abigail_Queues.zip	278.0 KiB	archivador Zip	22/02/24
Iti-271154_u1_raga_reyes_luis_antonio.rar	692.4 KiB	archivador RAR	25/02/24
Iti-2130072_u1_torres_colorado_juan_daniel.zip	3.3 MiB	archivador Zip	26/02/24
Iti-2130328_u1_pena_cuellar_aldo_de_jesus.zip	3.7 MiB	archivador Zip	26/02/24
Iti-2130328_uX1_pena_cuellar_aldo_de_jesus.zip	3.7 MiB	archivador Zip	26/02/24
Iti_271154_u1_olivares_rodriguez_brayan.zip	3.3 MiB	archivador Zip	26/02/24
SearchTrees-3-RBtree MORENO FLORES DAFNE PATRICIA Asignacion Especial U1.zip	3.4 MiB	archivador Zip	27/02/24
SearchTrees-3-RBtree MORENO FLORES DAFNE PATRICIA Asignacion Especial U1(1).zip	3.4 MiB	archivador Zip	27/02/24

Selecci3n: 2 archivos: 4.0 MiB (4 231 606 bytes)

Asignacion Especial U2 Y... Terminal - marco@marco... Terminal - marco@marco... Asignacion Especial U1 - T...

15:36 2024-04-01

Fechas importantes de entrega de proyectos (1)

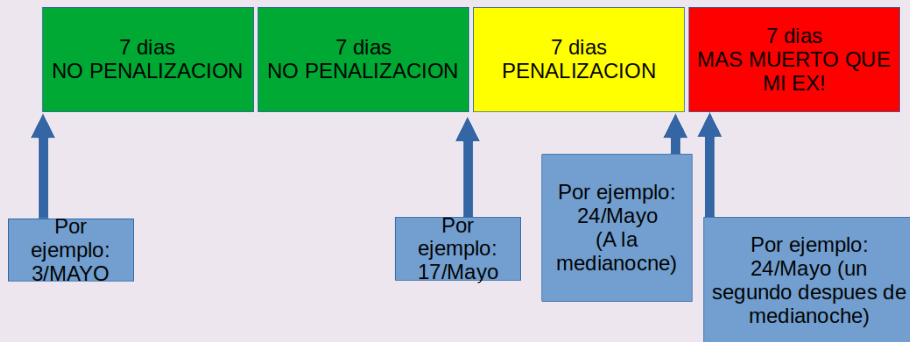
- Fecha de asignación: fecha en que se da a conocer al grupo el trabajo a elaborar
- Fecha de entrega sin penalización: 14 días naturales después de la fecha de asignación
- Proyecto entregado después de la fecha de penalización se le aplica una penalización de 20 PUNTOS
- Fecha de cierre: 21 días naturales después de la fecha de asignación.

Regla “CANTU”

- Ningún proyecto será revisado después de la fecha de cierre. Se programarán las entregas para cerrar y no permitir entregas tardías.

Fechas importantes de entrega de proyectos (2)

Grafo "DAFNE"



- Ningún proyecto será revisado despues de la fecha de cierre. Se programarán las entregas para cerrar y no permitir entregas tardías.

LINUX

En orden de dificultad

- Linux instalado de manera emulada usando VirtualBox o VMWare.
- Crear una USB o HD booteable (con persistencia) y bootear desde su laptop solo para las clases y los proyectos.
- Linux instalado de manera nativa. Distribuciones recomendadas: **Mint, Ubuntu, Lubuntu, Xubuntu, Debian**

**** Tienen la opción de no INSTALAR LINUX, pero la evaluación será realiza en una PC con Linux instalado**

Software Utilizado

Sobre una instalación de Linux, se debe instalar lo siguiente:

- Navegador Chrome/Firefox actualizado
- LaTeX para edición de reportes
- Python3
- Otras librerías (se especificarán conforme se vayan utilizando)

Se buscan integrantes para ingresar al
Salon de la fama del PLAGIO

Plagio

- Reprobación automática a quien reproduzca códigos de otros compañeros y los reporte como suyos, además de una nota en su expediente con copia para el consejo de calidad



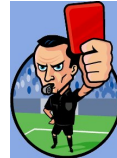
- Reprobación automática a quien copie códigos de Internet y los reporte como suyos, además de una nota en su expediente con copia para el consejo de calidad



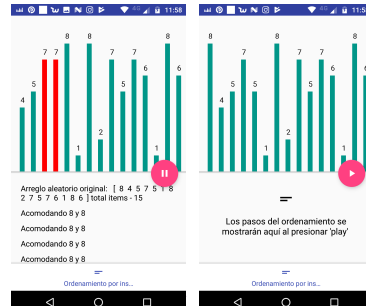
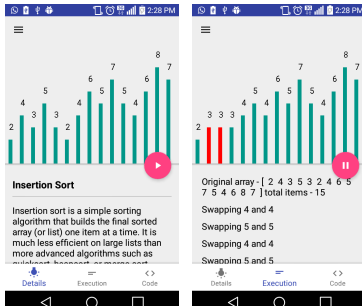
```

1 //Histogram equalization using C++ Image Processing | Programming Techniques | Mustafa Firdous
2
3 #include <iostream>
4 #include <opencv2/opencv.hpp>
5 #include <opencv2/imgproc/imgproc.hpp>
6
7 using std::cout;
8 using std::endl;
9 using namespace cv;
10 using namespace std;
11
12 //Funcion que inicializa todos los valores a 0
13 void initializeImage (cv::Mat& image)
14 {
15     for(int i = 0; i < image.rows; i++)
16     {
17         histogram[i] = 0;
18     }
19 }
20
21 //Calculando el numero de pixeles de cada valor de intensidad
22 for(int i = 0; i < image.rows; i++)
23 {
24     for(int j = 0; j < image.cols; j++)
25     {
26         histogram[image.at<uchar>(i,j)]++;
27     }
28 }
29
30 //Funcion para calcular el histograma
31 void calculateHistogram (cv::Mat& image, cv::Mat& histogram)
32 {
33     for(int i = 0; i < image.rows; i++)
34     {
35         histogram[i] = calculateHistogram (image.at<uchar>(i,0));
36     }
37 }
38
39 //Funcion que muestra los histogramas
40 void showHistograms (cv::Mat& histogram, const char* name)
41 {
42     cv::imshow(name, histogram);
43     cv::waitKey(0);
44 }
45
46 //Calculando los histogramas
47 int main()
48 {
49     cv::Mat image = cv::imread("image.jpg");
50     cv::Mat histogram(256, 1);
51     cv::Mat imageCopy = image.clone();
52     cv::Mat imageCopy2 = imageCopy.clone();
53
54     //Calculando la maxima intensidad del histograma
55     int max = 0;
56     for(int i = 0; i < histogram.rows; i++)
57     {
58         if(histogram[i] > max)
59             max = histogram[i];
60     }
61
62     //Normalizando el histograma
63     for(int i = 0; i < histogram.rows; i++)
64     {
65         histogram[i] = (float) histogram[i] / max;
66     }
67
68     //Calculando el histograma normalizado
69     cv::Mat histogram2(256, 1);
70     for(int i = 0; i < histogram.rows; i++)
71     {
72         histogram2[i] = histogram[i] * 255;
73     }
74
75     //Calculando el histograma normalizado
76     cv::Mat histogram3(256, 1);
77     for(int i = 0; i < histogram.rows; i++)
78     {
79         histogram3[i] = histogram2[i] / 255;
80     }
81
82     //Calculando el histograma normalizado
83     cv::Mat histogram4(256, 1);
84     for(int i = 0; i < histogram.rows; i++)
85     {
86         histogram4[i] = histogram3[i] * 255;
87     }
88
89     //Calculando el histograma normalizado
90     cv::Mat histogram5(256, 1);
91     for(int i = 0; i < histogram.rows; i++)
92     {
93         histogram5[i] = histogram4[i] / 255;
94     }
95
96     //Calculando el histograma normalizado
97     cv::Mat histogram6(256, 1);
98     for(int i = 0; i < histogram.rows; i++)
99     {
100         histogram6[i] = histogram5[i] * 255;
101     }
102
103     //Calculando el histograma normalizado
104     cv::Mat histogram7(256, 1);
105     for(int i = 0; i < histogram.rows; i++)
106     {
107         histogram7[i] = histogram6[i] / 255;
108     }
109
110     //Calculando el histograma normalizado
111     cv::Mat histogram8(256, 1);
112     for(int i = 0; i < histogram.rows; i++)
113     {
114         histogram8[i] = histogram7[i] * 255;
115     }
116
117     //Calculando el histograma normalizado
118     cv::Mat histogram9(256, 1);
119     for(int i = 0; i < histogram.rows; i++)
120     {
121         histogram9[i] = histogram8[i] / 255;
122     }
123
124     //Calculando el histograma normalizado
125     cv::Mat histogram10(256, 1);
126     for(int i = 0; i < histogram.rows; i++)
127     {
128         histogram10[i] = histogram9[i] * 255;
129     }
130
131     //Calculando el histograma normalizado
132     cv::Mat histogram11(256, 1);
133     for(int i = 0; i < histogram.rows; i++)
134     {
135         histogram11[i] = histogram10[i] / 255;
136     }
137
138     //Calculando el histograma normalizado
139     cv::Mat histogram12(256, 1);
140     for(int i = 0; i < histogram.rows; i++)
141     {
142         histogram12[i] = histogram11[i] * 255;
143     }
144
145     //Calculando el histograma normalizado
146     cv::Mat histogram13(256, 1);
147     for(int i = 0; i < histogram.rows; i++)
148     {
149         histogram13[i] = histogram12[i] / 255;
150     }
151
152     //Calculando el histograma normalizado
153     cv::Mat histogram14(256, 1);
154     for(int i = 0; i < histogram.rows; i++)
155     {
156         histogram14[i] = histogram13[i] * 255;
157     }
158
159     //Calculando el histograma normalizado
160     cv::Mat histogram15(256, 1);
161     for(int i = 0; i < histogram.rows; i++)
162     {
163         histogram15[i] = histogram14[i] / 255;
164     }
165
166     //Calculando el histograma normalizado
167     cv::Mat histogram16(256, 1);
168     for(int i = 0; i < histogram.rows; i++)
169     {
170         histogram16[i] = histogram15[i] * 255;
171     }
172
173     //Calculando el histograma normalizado
174     cv::Mat histogram17(256, 1);
175     for(int i = 0; i < histogram.rows; i++)
176     {
177         histogram17[i] = histogram16[i] / 255;
178     }
179
180     //Calculando el histograma normalizado
181     cv::Mat histogram18(256, 1);
182     for(int i = 0; i < histogram.rows; i++)
183     {
184         histogram18[i] = histogram17[i] * 255;
185     }
186
187     //Calculando el histograma normalizado
188     cv::Mat histogram19(256, 1);
189     for(int i = 0; i < histogram.rows; i++)
190     {
191         histogram19[i] = histogram18[i] / 255;
192     }
193
194     //Calculando el histograma normalizado
195     cv::Mat histogram20(256, 1);
196     for(int i = 0; i < histogram.rows; i++)
197     {
198         histogram20[i] = histogram19[i] * 255;
199     }
200
201     //Calculando el histograma normalizado
202     cv::Mat histogram21(256, 1);
203     for(int i = 0; i < histogram.rows; i++)
204     {
205         histogram21[i] = histogram20[i] / 255;
206     }
207
208     //Calculando el histograma normalizado
209     cv::Mat histogram22(256, 1);
210     for(int i = 0; i < histogram.rows; i++)
211     {
212         histogram22[i] = histogram21[i] * 255;
213     }
214
215     //Calculando el histograma normalizado
216     cv::Mat histogram23(256, 1);
217     for(int i = 0; i < histogram.rows; i++)
218     {
219         histogram23[i] = histogram22[i] / 255;
220     }
221
222     //Calculando el histograma normalizado
223     cv::Mat histogram24(256, 1);
224     for(int i = 0; i < histogram.rows; i++)
225     {
226         histogram24[i] = histogram23[i] * 255;
227     }
228
229     //Calculando el histograma normalizado
230     cv::Mat histogram25(256, 1);
231     for(int i = 0; i < histogram.rows; i++)
232     {
233         histogram25[i] = histogram24[i] / 255;
234     }
235
236     //Calculando el histograma normalizado
237     cv::Mat histogram26(256, 1);
238     for(int i = 0; i < histogram.rows; i++)
239     {
240         histogram26[i] = histogram25[i] * 255;
241     }
242
243     //Calculando el histograma normalizado
244     cv::Mat histogram27(256, 1);
245     for(int i = 0; i < histogram.rows; i++)
246     {
247         histogram27[i] = histogram26[i] / 255;
248     }
249
250     //Calculando el histograma normalizado
251     cv::Mat histogram28(256, 1);
252     for(int i = 0; i < histogram.rows; i++)
253     {
254         histogram28[i] = histogram27[i] * 255;
255     }
256
257     //Calculando el histograma normalizado
258     cv::Mat histogram29(256, 1);
259     for(int i = 0; i < histogram.rows; i++)
260     {
261         histogram29[i] = histogram28[i] / 255;
262     }
263
264     //Calculando el histograma normalizado
265     cv::Mat histogram30(256, 1);
266     for(int i = 0; i < histogram.rows; i++)
267     {
268         histogram30[i] = histogram29[i] * 255;
269     }
270
271     //Calculando el histograma normalizado
272     cv::Mat histogram31(256, 1);
273     for(int i = 0; i < histogram.rows; i++)
274     {
275         histogram31[i] = histogram30[i] / 255;
276     }
277
278     //Calculando el histograma normalizado
279     cv::Mat histogram32(256, 1);
280     for(int i = 0; i < histogram.rows; i++)
281     {
282         histogram32[i] = histogram31[i] * 255;
283     }
284
285     //Calculando el histograma normalizado
286     cv::Mat histogram33(256, 1);
287     for(int i = 0; i < histogram.rows; i++)
288     {
289         histogram33[i] = histogram32[i] / 255;
290     }
291
292     //Calculando el histograma normalizado
293     cv::Mat histogram34(256, 1);
294     for(int i = 0; i < histogram.rows; i++)
295     {
296         histogram34[i] = histogram33[i] * 255;
297     }
298
299     //Calculando el histograma normalizado
300     cv::Mat histogram35(256, 1);
301     for(int i = 0; i < histogram.rows; i++)
302     {
303         histogram35[i] = histogram34[i] / 255;
304     }
305
306     //Calculando el histograma normalizado
307     cv::Mat histogram36(256, 1);
308     for(int i = 0; i < histogram.rows; i++)
309     {
310         histogram36[i] = histogram35[i] * 255;
311     }
312
313     //Calculando el histograma normalizado
314     cv::Mat histogram37(256, 1);
315     for(int i = 0; i < histogram.rows; i++)
316     {
317         histogram37[i] = histogram36[i] / 255;
318     }
319
320     //Calculando el histograma normalizado
321     cv::Mat histogram38(256, 1);
322     for(int i = 0; i < histogram.rows; i++)
323     {
324         histogram38[i] = histogram37[i] * 255;
325     }
326
327     //Calculando el histograma normalizado
328     cv::Mat histogram39(256, 1);
329     for(int i = 0; i < histogram.rows; i++)
330     {
331         histogram39[i] = histogram38[i] / 255;
332     }
333
334     //Calculando el histograma normalizado
335     cv::Mat histogram40(256, 1);
336     for(int i = 0; i < histogram.rows; i++)
337     {
338         histogram40[i] = histogram39[i] * 255;
339     }
340
341     //Calculando el histograma normalizado
342     cv::Mat histogram41(256, 1);
343     for(int i = 0; i < histogram.rows; i++)
344     {
345         histogram41[i] = histogram40[i] / 255;
346     }
347
348     //Calculando el histograma normalizado
349     cv::Mat histogram42(256, 1);
350     for(int i = 0; i < histogram.rows; i++)
351     {
352         histogram42[i] = histogram41[i] * 255;
353     }
354
355     //Calculando el histograma normalizado
356     cv::Mat histogram43(256, 1);
357     for(int i = 0; i < histogram.rows; i++)
358     {
359         histogram43[i] = histogram42[i] / 255;
360     }
361
362     //Calculando el histograma normalizado
363     cv::Mat histogram44(256, 1);
364     for(int i = 0; i < histogram.rows; i++)
365     {
366         histogram44[i] = histogram43[i] * 255;
367     }
368
369     //Calculando el histograma normalizado
370     cv::Mat histogram45(256, 1);
371     for(int i = 0; i < histogram.rows; i++)
372     {
373         histogram45[i] = histogram44[i] / 255;
374     }
375
376     //Calculando el histograma normalizado
377     cv::Mat histogram46(256, 1);
378     for(int i = 0; i < histogram.rows; i++)
379     {
380         histogram46[i] = histogram45[i] * 255;
381     }
382
383     //Calculando el histograma normalizado
384     cv::Mat histogram47(256, 1);
385     for(int i = 0; i < histogram.rows; i++)
386     {
387         histogram47[i] = histogram46[i] / 255;
388     }
389
390     //Calculando el histograma normalizado
391     cv::Mat histogram48(256, 1);
392     for(int i = 0; i < histogram.rows; i++)
393     {
394         histogram48[i] = histogram47[i] * 255;
395     }
396
397     //Calculando el histograma normalizado
398     cv::Mat histogram49(256, 1);
399     for(int i = 0; i < histogram.rows; i++)
400     {
401         histogram49[i] = histogram48[i] / 255;
402     }
403
404     //Calculando el histograma normalizado
405     cv::Mat histogram50(256, 1);
406     for(int i = 0; i < histogram.rows; i++)
407     {
408         histogram50[i] = histogram49[i] * 255;
409     }
410
411     //Calculando el histograma normalizado
412     cv::Mat histogram51(256, 1);
413     for(int i = 0; i < histogram.rows; i++)
414     {
415         histogram51[i] = histogram50[i] / 255;
416     }
417
418     //Calculando el histograma normalizado
419     cv::Mat histogram52(256, 1);
420     for(int i = 0; i < histogram.rows; i++)
421     {
422         histogram52[i] = histogram51[i] * 255;
423     }
424
425     //Calculando el histograma normalizado
426     cv::Mat histogram53(256, 1);
427     for(int i = 0; i < histogram.rows; i++)
428     {
429         histogram53[i] = histogram52[i] / 255;
430     }
431
432     //Calculando el histograma normalizado
433     cv::Mat histogram54(256, 1);
434     for(int i = 0; i < histogram.rows; i++)
435     {
436         histogram54[i] = histogram53[i] * 255;
437     }
438
439     //Calculando el histograma normalizado
440     cv::Mat histogram55(256, 1);
441     for(int i = 0; i < histogram.rows; i++)
442     {
443         histogram55[i] = histogram54[i] / 255;
444     }
445
446     //Calculando el histograma normalizado
447     cv::Mat histogram56(256, 1);
448     for(int i = 0; i < histogram.rows; i++)
449     {
450         histogram56[i] = histogram55[i] * 255;
451     }
452
453     //Calculando el histograma normalizado
454     cv::Mat histogram57(256, 1);
455     for(int i = 0; i < histogram.rows; i++)
456     {
457         histogram57[i] = histogram56[i] / 255;
458     }
459
460     //Calculando el histograma normalizado
461     cv::Mat histogram58(256, 1);
462     for(int i = 0; i < histogram.rows; i++)
463     {
464         histogram58[i] = histogram57[i] * 255;
465     }
466
467     //Calculando el histograma normalizado
468     cv::Mat histogram59(256, 1);
469     for(int i = 0; i < histogram.rows; i++)
470     {
471         histogram59[i] = histogram58[i] / 255;
472     }
473
474     //Calculando el histograma normalizado
475     cv::Mat histogram60(256, 1);
476     for(int i = 0; i < histogram.rows; i++)
477     {
478         histogram60[i] = histogram59[i] * 255;
479     }
480
481     //Calculando el histograma normalizado
482     cv::Mat histogram61(256, 1);
483     for(int i = 0; i < histogram.rows; i++)
484     {
485         histogram61[i] = histogram60[i] / 255;
486     }
487
488     //Calculando el histograma normalizado
489     cv::Mat histogram62(256, 1);
490     for(int i = 0; i < histogram.rows; i++)
491     {
492         histogram62[i] = histogram61[i] * 255;
493     }
494
495     //Calculando el histograma normalizado
496     cv::Mat histogram63(256, 1);
497     for(int i = 0; i < histogram.rows; i++)
498     {
499         histogram63[i] = histogram62[i] / 255;
500     }
501
502     //Calculando el histograma normalizado
503     cv::Mat histogram64(256, 1);
504     for(int i = 0; i < histogram.rows; i++)
505     {
506         histogram64[i] = histogram63[i] * 255;
507     }
508
509     //Calculando el histograma normalizado
510     cv::Mat histogram65(256, 1);
511     for(int i = 0; i < histogram.rows; i++)
512     {
513         histogram65[i] = histogram64[i] / 255;
514     }
515
516     //Calculando el histograma normalizado
517     cv::Mat histogram66(256, 1);
518     for(int i = 0; i < histogram.rows; i++)
519     {
520         histogram66[i] = histogram65[i] * 255;
521     }
522
523     //Calculando el histograma normalizado
524     cv::Mat histogram67(256, 1);
525     for(int i = 0; i < histogram.rows; i++)
526     {
527         histogram67[i] = histogram66[i] / 255;
528     }
529
530     //Calculando el histograma normalizado
531     cv::Mat histogram68(256, 1);
532     for(int i = 0; i < histogram.rows; i++)
533     {
534         histogram68[i] = histogram67[i] * 255;
535     }
536
537     //Calculando el histograma normalizado
538     cv::Mat histogram69(256, 1);
539     for(int i = 0; i < histogram.rows; i++)
540     {
541         histogram69[i] = histogram68[i] / 255;
542     }
543
544     //Calculando el histograma normalizado
545     cv::Mat histogram70(256, 1);
546     for(int i = 0; i < histogram.rows; i++)
547     {
548         histogram70[i] = histogram69[i] * 255;
549     }
550
551     //Calculando el histograma normalizado
552     cv::Mat histogram71(256, 1);
553     for(int i = 0; i < histogram.rows; i++)
554     {
555         histogram71[i] = histogram70[i] / 255;
556     }
557
558     //Calculando el histograma normalizado
559     cv::Mat histogram72(256, 1);
560     for(int i = 0; i < histogram.rows; i++)
561     {
562         histogram72[i] = histogram71[i] * 255;
563     }
564
565     //Calculando el histograma normalizado
566     cv::Mat histogram73(256, 1);
567     for(int i = 0; i < histogram.rows; i++)
568     {
569         histogram73[i] = histogram72[i] / 255;
570     }
571
572     //Calculando el histograma normalizado
573     cv::Mat histogram74(256, 1);
574     for(int i = 0; i < histogram.rows; i++)
575     {
576         histogram74[i] = histogram73[i] * 255;
577     }
578
579     //Calculando el histograma normalizado
580     cv::Mat histogram75(256, 1);
581     for(int i = 0; i < histogram.rows; i++)
582     {
583         histogram75[i] = histogram74[i] / 255;
584     }
585
586     //Calculando el histograma normalizado
587     cv::Mat histogram76(256, 1);
588     for(int i = 0; i < histogram.rows; i++)
589     {
590         histogram76[i] = histogram75[i] * 255;
591     }
592
593     //Calculando el histograma normalizado
594     cv::Mat histogram77(256, 1);
595     for(int i = 0; i < histogram.rows; i++)
596     {
597         histogram77[i] = histogram76[i] / 255;
598     }
599
600     //Calculando el histograma normalizado
601     cv::Mat histogram78(256, 1);
602     for(int i = 0; i < histogram.rows; i++)
603     {
604         histogram78[i] = histogram77[i] * 255;
605     }
606
607     //Calculando el histograma normalizado
608     cv::Mat histogram79(256, 1);
609     for(int i = 0; i < histogram.rows; i++)
610     {
611         histogram79[i] = histogram78[i] / 255;
612     }
613
614     //Calculando el histograma normalizado
615     cv::Mat histogram80(256, 1);
616     for(int i = 0; i < histogram.rows; i++)
617     {
618         histogram80[i] = histogram79[i] * 255;
619     }
620
621     //Calculando el histograma normalizado
622     cv::Mat histogram81(256, 1);
623     for(int i = 0; i < histogram.rows; i++)
624     {
625         histogram81[i] = histogram80[i] / 255;
626     }
627
628     //Calculando el histograma normalizado
629     cv::Mat histogram82(256, 1);
630     for(int i = 0; i < histogram.rows; i++)
631     {
632         histogram82[i] = histogram81[i] * 255;
633     }
634
635     //Calculando el histograma normalizado
636     cv::Mat histogram83(256, 1);
637     for(int i = 0; i < histogram.rows; i++)
638     {
639         histogram83[i] = histogram82[i] / 255;
640     }
641
642     //Calculando el histograma normalizado
643     cv::Mat histogram84(256, 1);
644     for(int i = 0; i < histogram.rows; i++)
645     {
646         histogram84[i] = histogram83[i] * 255;
647     }
648
649     //Calculando el histograma normalizado
650     cv::Mat histogram85(256, 1);
651     for(int i = 0; i < histogram.rows; i++)
652     {
653         histogram85[i] = histogram84[i] / 255;
654     }
655
656     //Calculando el histograma normalizado
657     cv::Mat histogram86(256, 1);
658     for(int i = 0; i < histogram.rows; i++)
659     {
660         histogram86[i] = histogram85[i] * 255;
661     }
662
663     //Calculando el histograma normalizado
664     cv::Mat histogram87(256, 1);
665     for(int i = 0; i < histogram.rows; i++)
666     {
667         histogram87[i] = histogram86[i] / 255;
668     }
669
670     //Calculando el histograma normalizado
671     cv::Mat histogram88(256, 1);
672     for(int i = 0; i < histogram.rows; i++)
673     {
674         histogram88[i] = histogram87[i] * 255;
675     }
676
677     //Calculando el histograma normalizado
678     cv::Mat histogram89(256, 1);
679     for(int i = 0; i < histogram.rows; i++)
680     {
681         histogram89[i] = histogram88[i] / 255;
682     }
683
684     //Calculando el histograma normalizado
685     cv::Mat histogram90(256, 1);
686     for(int i = 0; i < histogram.rows; i++)
687     {
688         histogram90[i] = histogram89[i] * 255;
689     }
690
691     //Calculando el histograma normalizado
692     cv::Mat histogram91(256, 1);
693     for(int i = 0; i < histogram.rows; i++)
694     {
695         histogram91[i] = histogram90[i] / 255;
696     }
697
698     //Calculando el histograma normalizado
699     cv::Mat histogram92(256, 1);
700     for(int i = 0; i < histogram.rows; i++)
701     {
702         histogram92[i] = histogram91[i] * 255;
703     }
704
705     //Calculando el histograma normalizado
706     cv::Mat histogram93(256, 1);
707     for(int i = 0; i < histogram.rows; i++)
708     {
709         histogram93[i] = histogram92[i] / 255;
710     }
711
712     //Calculando el histograma normalizado
713     cv::Mat histogram94(256, 1);
714     for(int i = 0; i < histogram.rows; i++)
715     {
716         histogram94[i] = histogram93[i] * 255;
717     }
718
719     //Calculando el histograma normalizado
720     cv::Mat histogram95(256, 1);
721     for(int i = 0; i < histogram.rows; i++)
722     {
723         histogram95[i] = histogram94[i] / 255;
724     }
725
726     //Calculando el histograma normalizado
727     cv::Mat histogram96(256, 1);
728     for(int i = 0; i < histogram.rows; i++)
729     {
730         histogram96[i] = histogram95[i] * 255;
731     }
732
733     //Calculando el histograma normalizado
734     cv::Mat histogram97(256, 1);
735     for(int i = 0; i < histogram.rows; i++)
736     {
737         histogram97[i] = histogram96[i] / 255;
738     }
739
740     //Calculando el histograma normalizado
741     cv::Mat histogram98(256, 1);
742     for(int i = 0; i < histogram.rows; i++)
743     {
744         histogram98[i] = histogram97[i] * 255;
745     }
746
747     //Calculando el histograma normalizado
748     cv::Mat histogram99(256, 1);
749     for(int i = 0; i < histogram.rows; i++)
750     {
751         histogram99[i] = histogram98[i] / 255;
752     }
753
754     //Calculando el histograma normalizado
755     cv::Mat histogram100(256, 1);
756     for(int i = 0; i < histogram.rows; i++)
757     {
758         histogram100[i] = histogram99[i] * 255;
759     }
760
761     //Calculando el histograma normalizado
762     cv::Mat histogram101(256, 1);
763     for(int i = 0; i < histogram.rows; i++)
764     {
765         histogram101[i] = histogram100[i] / 255;
766     }
767
768     //Calculando el histograma normalizado
769     cv::Mat histogram102(256, 1);
770     for(int i = 0; i < histogram.rows; i++)
771     {
772         histogram102[i] = histogram101[i] * 255;
773     }
774
775     //Calculando el histograma normalizado
776     cv::Mat histogram103(256, 1);
777     for(int i = 0; i < histogram.rows; i++)
778     {
779         histogram103[i] = histogram102[i] / 255;
780     }
781
782     //Calculando el histograma normalizado
783     cv::Mat histogram104(256, 1);
784     for(int i = 0; i < histogram.rows; i++)
785     {
786         histogram104[i] = histogram103[i] * 255;
787     }
788
789     //Calculando el histograma normalizado
790     cv::Mat histogram105(256, 1);
791     for(int i = 0; i < histogram.rows; i++)
792     {
793         histogram105[i] = histogram104[i] / 255;
794     }
795
796     //Calculando el histograma normalizado
797     cv::Mat histogram106(256, 1);
798     for(int i = 0; i < histogram.rows; i++)
799     {
800         histogram106[i] = histogram105[i] * 255;
801     }
802
803     //Calculando el histograma normalizado
804     cv::Mat histogram107(256, 1);
805     for(int i = 0; i < histogram.rows; i++)
806     {
807         histogram107[i] = histogram106[i] / 255;
808     }
809
810     //Calculando el histograma normalizado
811     cv::Mat histogram108(256, 1);
812     for(int i = 0; i < histogram.rows; i++)
813     {
814         histogram108[i] = histogram107[i] * 255;
815     }
816
817     //Calculando el histograma normalizado
818     cv::Mat histogram109(256, 1);
819     for(int i = 0; i < histogram.rows; i++)
820     {
821         histogram109[i] = histogram108[i] / 255;
822     }
823
824     //Calculando el histograma normalizado
825     cv::Mat histogram110(256, 1);
826     for(int i = 0; i < histogram.rows; i++)
827     {
828         histogram110[i] = histogram109[i] * 255;
829     }
830
831     //Calculando el histograma normalizado
832     cv::Mat histogram111(256, 1);
833     for(int i = 0; i < histogram.rows; i++)
834     {
835         histogram111[i] = histogram110[i] / 255;
836     }
837
838     //Calculando el histograma normalizado
839     cv::Mat histogram112(256, 1);
840     for(int i = 0; i < histogram.rows; i++)
841     {
842         histogram112[i] = histogram111[i] * 255;
843     }
844
845     //Calculando el histograma normalizado
846     cv::Mat histogram113(256, 1);
847     for(int i = 0; i < histogram.rows; i++)
848     {
849         histogram113[i] = histogram112[i] / 255;
850     }
851
852     //Calculando el histograma normalizado
853     cv::Mat histogram114(256, 1);
854     for(int i = 0; i < histogram.rows; i++)
855     {
856         histogram114[i] = histogram113[i] * 255;
857     }
858
859     //Calculando el histograma normalizado
860     cv::Mat histogram115(256, 1);
861     for(int i = 0; i < histogram.rows; i++)
862     {
863         histogram115[i] = histogram114[i] / 255;
864     }
865
866     //Calculando el histograma normalizado
867     cv::Mat histogram116(256, 1);
868     for(int i = 0; i < histogram.rows; i++)
869     {
870         histogram116[i] = histogram115[i] * 255;
871     }
872
873     //Calculando el histograma normalizado
874     cv::Mat histogram117(256, 1);
875     for(int i = 0; i < histogram.rows; i++)
876     {
877         histogram117[i] = histogram116[i] / 255;
878     }
879
880     //Calculando el histograma normalizado
881     cv::Mat histogram118(256, 1);
882     for(int i = 0; i < histogram.rows; i++)
883     {
884         histogram118[i] = histogram117[i] * 255;
885     }
886
887     //Calculando el histograma normalizado
888     cv::Mat histogram119(256, 1);
889     for(int i = 0; i < histogram.rows; i++)
890     {
891         histogram119[i] = histogram118[i] / 255;
892     }
893
894     //Calculando el histograma normalizado
895     cv::Mat histogram120(256, 1);
896     for(int i = 0; i < histogram.rows; i++)
897     {
898         histogram120[i] = histogram119[i] * 255;
899     }
900
901     //Calculando el histograma normalizado
902     cv::Mat histogram121(256, 1);
903     for(int i = 0; i < histogram.rows; i++)
904     {
905         histogram121[i] = histogram120[i] / 255;
906     }
907
908     //Calculando el histograma normalizado
909     cv::Mat histogram122(256, 1);
910     for(int i = 0; i < histogram.rows; i++)
911     {
912         histogram122[i] = histogram121[i] * 255;
913     }
914
915     //Calculando el histograma normalizado
916     cv::Mat histogram123(256, 1);
917     for(int i = 0; i < histogram.rows; i++)
918     {
919         histogram123[i] = histogram122[i] / 255;
920     }
921
922     //Calculando el histograma normalizado
923     cv::Mat histogram124(256, 1);
924     for(int i = 0; i < histogram.rows; i++)
925     {
926         histogram124[i] = histogram123[i] * 255;
927     }
928
929     //Calculando el histograma normalizado
930     cv::Mat histogram125(256, 1);
931     for(int i = 0; i < histogram.rows; i++)
932     {
933         histogram125[i] = histogram124[i] / 255;
934     }
935
936     //Calculando el histograma normalizado
937     cv::Mat histogram126(256, 1);
938     for(int i = 0; i < histogram.rows; i++)
939     {
940         histogram126[i] = histogram125[i] * 255;
941     }
942
943     //Calculando el histograma normalizado
944     cv::Mat histogram127(256, 1);
945     for(int i = 0; i < histogram.rows
```

- Reprobación automática a quien copie códigos de Internet y los reporte como suyos, además de una nota en su expediente con copia para el consejo de calidad



<https://github.com/naman14/AlgorithmVisualizer-Android>



“Finalmente son jóvenes que están en la preparatoria y que deben de leer su convocatoria con toda claridad, si no cumplen con los requisitos, si no pueden leer una convocatoria que dice tienes que traer número uno esto, número dos esto, número tres esto, no están listos para ser **estudiantes de educación superior**, así lo digo con toda claridad”.

Sara Ladrón de Guevara.

Rectora de la Universidad Veracruzana (2013-2017 y 2017-2021).

CONCLUSIÓN

