

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 2, 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 Mviles, Graficación por Computadora Avanzada, Lenguajes y Automátas, 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 maestria 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.
- Respeto 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.

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/dgkhvfwynygvc>

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

- 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
- Proyecto: 66 Puntos
 - Ejecución y Funcionalidad: 45 Puntos
 - Modularidad: 13 Puntos
 - Documentación: 8 Puntos

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 (solo si se trata de una aplicación móvil))
 - **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.apk** (Instalable - Solo aplicaciones móviles)
 - **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 archivos en el classroom.

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, Cuando aplique)
 - **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

Penalizaciones por Entregas Incompletas

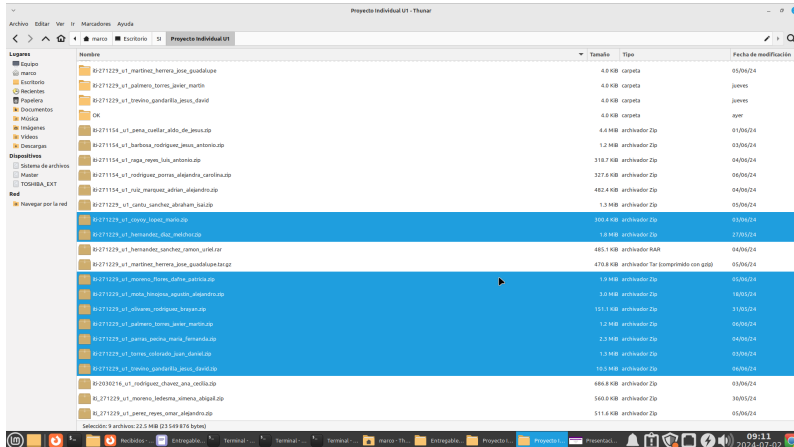
- Proyecto que no este entregado de acuerdo con las especificaciones, será penalizado. Dos escenarios posibles:
 - El proyecto puede revisarse (completo o con faltas al formato).
 - El proyecto NO puede revisarse (falta codigo fuente, informe, APK, no se compila por alguna falla, etc). En automático el proyecto queda descartado.

Se recomienda LEER con cuidado la sección de entregables de esta presentación. Las penalizaciones son acumulables.

Falta	Penalizacion
Nombre Archivo	8
Tipo de Archivo	7
Estructura de Directorios	6
Falta o Error en Script	6
Poner ZIPs dentro del ZIP	8
Incluir ejecutables (aplica solo cuando el lenguaje es C++)	8

Premio a la Compresión Lectora 2024

A pesar de las instrucciones en esta presentación, alguien va a hacer las cosas mal



Salon de la Fama de Entregas Completas e Incompletas

Nombre
iti-271229_u1_trevino_gandarilla_jesus_david_source
iti-271229_u1_trevino_gandarilla_jesus_david_latex
iti-271229_u1_trevino_gandarilla_jesus_david.pdf

Nombre
iti-271229_u1_coyoy_lopez_mario_source
iti-271229_u1_coyoy_lopez_mario_latex
iti-271229_u1_coyoy_lopez_mario.pdf

Nombre
iti-271229_u1_olivares_rodriguez_brayan_source
iti-271229_u1_olivares_rodriguez_brayan_latex
iti-271229_u1_olivares_rodriguez_brayan.pdf

Nombre
iti-271229_u1_martinez_herrera_jose_guadalupe_source
REPORTE INDIVIDUAL U2.zip
REPORTE_INDIVIDUAL_U2.pdf

Nombre
iti-271154_u1_rodriguez_porras_alejandra_carolina_latex.zip
iti-271154_u1_rodriguez_porras_alejandra_carolina_source.zip
iti_271154_u1_rodriguez_porras_alejandra_carolina_latex .pdf

Nombre
iti-271229_u1_parras_pecina_maria_fernanda_latex
iti-271229_u1_parras_pecina_maria_fernanda_source

Fechas importantes de entrega de proyectos (1)

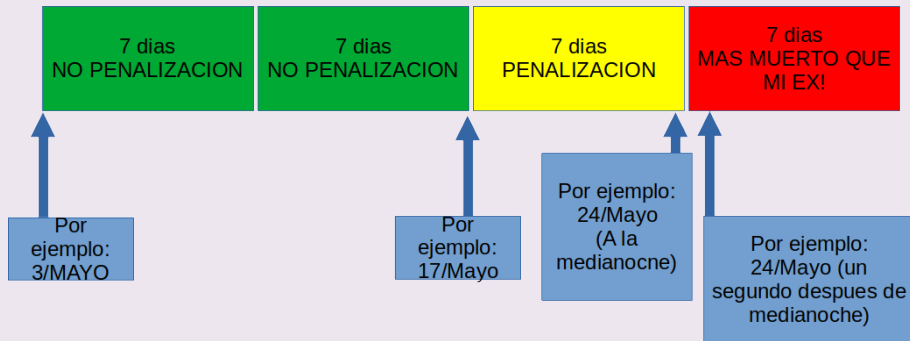
- Fecha de asignación: fecha en que se da a conocer al grupo el trabajo a elaborar
- Fecha de entrega sin penalización: 14 días naturales después de la fecha de asignación
- Proyecto entregado después de la fecha de penalización se le aplica una penalización de 20 PUNTOS
- Fecha de cierre: 21 días naturales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



- En el momento de publicar la tarea, se incluirá la fecha para no penalización y fecha de cierre.

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

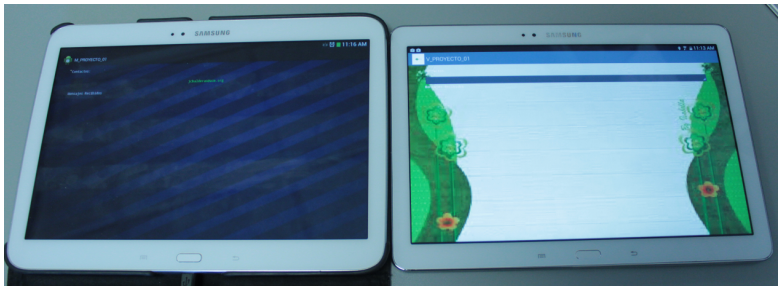
Sobre una instalación de Linux, se debe instalar lo siguiente:

- 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



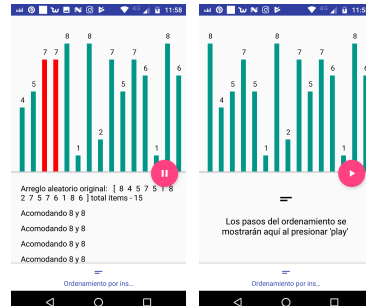
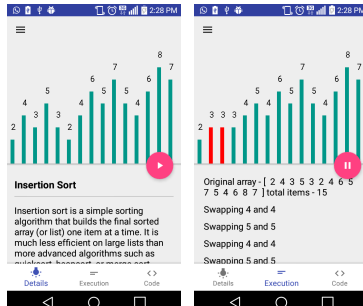
```

1 // Histogram equalization using C++ Image Processing | Programming Techniques | Mustafa Piril
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 initHistImage (int histImage[])
14 {
15     // initialization all intensity values to 0
16     for(int i = 0; i < 256; i++)
17     {
18         histImage[i] = 0;
19     }
20 }
21
22 //Calculando el numero de pixeles de cada valor de intensidad
23 for(int i = 0; i < image.cols; i++)
24 {
25     for(int j = 0; j < image.rows; j++)
26     {
27         histImage[histImage.at<uchar>(i,j)]++;
28     }
29 }
30
31 //Funcion acumulativa del histograma
32 void calcHistImage (int histImage[])
33 {
34     calcHistImage[0] = histImage[0];
35     for(int i = 1; i < 256; i++)
36     {
37         calcHistImage[i] = calcHistImage[i-1] + histImage[i];
38     }
39 }
40
41 //Funcion que devuelve los histogramas
42 void histImageCalc (int histImage[], const char* name)
43 {
44     int hist[256];
45     for(int i = 0; i < 256; i++)
46     {
47         hist[i] = histImage[i];
48     }
49 }
50
51 //Calculando los histogramas
52 int hist_w = 128; int hist_h = 128;
53 int bin_w = cvRound( (double) hist_w/256);
54 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
55
56 // Find the maxium intensity element from Histogram
57 int bin = hist[0];
58 for(int i = 1; i < 256; i++)
59 {
60     if(hist[i] > bin)
61     {
62         bin = hist[i];
63     }
64 }
65
66 // normalize the histogram between 0 and histImage.rows
67 for(int i = 0; i < 256; i++)
68 {
69     hist[i] = (float) hist[i] / bin;
70 }
71
72 // Find the maxium intensity element from Histogram
73 int bin = hist[0];
74 for(int i = 1; i < 256; i++)
75 {
76     if(hist[i] > bin)
77     {
78         bin = hist[i];
79     }
80 }
81
82 // normalize the histogram between 0 and histImage.rows
83 for(int i = 0; i < 256; i++)
84 {
85     hist[i] = (float) hist[i] / bin;
86 }
87
88 // Find the maxium intensity element from Histogram
89 int bin = hist[0];
90 for(int i = 1; i < 256; i++)
91 {
92     if(hist[i] > bin)
93     {
94         bin = hist[i];
95     }
96 }
97
98 // normalize the histogram between 0 and histImage.rows
99 for(int i = 0; i < 256; i++)
100 {
101     hist[i] = (float) hist[i] / bin;
102 }
103
104 // Find the maxium intensity element from Histogram
105 int bin = hist[0];
106 for(int i = 1; i < 256; i++)
107 {
108     if(hist[i] > bin)
109     {
110         bin = hist[i];
111     }
112 }
113
114 // normalize the histogram between 0 and histImage.rows
115 for(int i = 0; i < 256; i++)
116 {
117     hist[i] = (float) hist[i] / bin;
118 }
119
120 // Find the maxium intensity element from Histogram
121 int bin = hist[0];
122 for(int i = 1; i < 256; i++)
123 {
124     if(hist[i] > bin)
125     {
126         bin = hist[i];
127     }
128 }
129
130 // normalize the histogram between 0 and histImage.rows
131 for(int i = 0; i < 256; i++)
132 {
133     hist[i] = (float) hist[i] / bin;
134 }
135
136 // Find the maxium intensity element from Histogram
137 int bin = hist[0];
138 for(int i = 1; i < 256; i++)
139 {
140     if(hist[i] > bin)
141     {
142         bin = hist[i];
143     }
144 }
145
146 // normalize the histogram between 0 and histImage.rows
147 for(int i = 0; i < 256; i++)
148 {
149     hist[i] = (float) hist[i] / bin;
150 }
151
152 // Find the maxium intensity element from Histogram
153 int bin = hist[0];
154 for(int i = 1; i < 256; i++)
155 {
156     if(hist[i] > bin)
157     {
158         bin = hist[i];
159     }
160 }
161
162 // normalize the histogram between 0 and histImage.rows
163 for(int i = 0; i < 256; i++)
164 {
165     hist[i] = (float) hist[i] / bin;
166 }
167
168 // Find the maxium intensity element from Histogram
169 int bin = hist[0];
170 for(int i = 1; i < 256; i++)
171 {
172     if(hist[i] > bin)
173     {
174         bin = hist[i];
175     }
176 }
177
178 // normalize the histogram between 0 and histImage.rows
179 for(int i = 0; i < 256; i++)
180 {
181     hist[i] = (float) hist[i] / bin;
182 }
183
184 // Find the maxium intensity element from Histogram
185 int bin = hist[0];
186 for(int i = 1; i < 256; i++)
187 {
188     if(hist[i] > bin)
189     {
190         bin = hist[i];
191     }
192 }
193
194 // normalize the histogram between 0 and histImage.rows
195 for(int i = 0; i < 256; i++)
196 {
197     hist[i] = (float) hist[i] / bin;
198 }
199
200 // Find the maxium intensity element from Histogram
201 int bin = hist[0];
202 for(int i = 1; i < 256; i++)
203 {
204     if(hist[i] > bin)
205     {
206         bin = hist[i];
207     }
208 }
209
210 // normalize the histogram between 0 and histImage.rows
211 for(int i = 0; i < 256; i++)
212 {
213     hist[i] = (float) hist[i] / bin;
214 }
215
216 // Find the maxium intensity element from Histogram
217 int bin = hist[0];
218 for(int i = 1; i < 256; i++)
219 {
220     if(hist[i] > bin)
221     {
222         bin = hist[i];
223     }
224 }
225
226 // normalize the histogram between 0 and histImage.rows
227 for(int i = 0; i < 256; i++)
228 {
229     hist[i] = (float) hist[i] / bin;
230 }
231
232 // Find the maxium intensity element from Histogram
233 int bin = hist[0];
234 for(int i = 1; i < 256; i++)
235 {
236     if(hist[i] > bin)
237     {
238         bin = hist[i];
239     }
240 }
241
242 // normalize the histogram between 0 and histImage.rows
243 for(int i = 0; i < 256; i++)
244 {
245     hist[i] = (float) hist[i] / bin;
246 }
247
248 // Find the maxium intensity element from Histogram
249 int bin = hist[0];
250 for(int i = 1; i < 256; i++)
251 {
252     if(hist[i] > bin)
253     {
254         bin = hist[i];
255     }
256 }
257
258 // normalize the histogram between 0 and histImage.rows
259 for(int i = 0; i < 256; i++)
260 {
261     hist[i] = (float) hist[i] / bin;
262 }
263
264 // Find the maxium intensity element from Histogram
265 int bin = hist[0];
266 for(int i = 1; i < 256; i++)
267 {
268     if(hist[i] > bin)
269     {
270         bin = hist[i];
271     }
272 }
273
274 // normalize the histogram between 0 and histImage.rows
275 for(int i = 0; i < 256; i++)
276 {
277     hist[i] = (float) hist[i] / bin;
278 }
279
280 // Find the maxium intensity element from Histogram
281 int bin = hist[0];
282 for(int i = 1; i < 256; i++)
283 {
284     if(hist[i] > bin)
285     {
286         bin = hist[i];
287     }
288 }
289
290 // normalize the histogram between 0 and histImage.rows
291 for(int i = 0; i < 256; i++)
292 {
293     hist[i] = (float) hist[i] / bin;
294 }
295
296 // Find the maxium intensity element from Histogram
297 int bin = hist[0];
298 for(int i = 1; i < 256; i++)
299 {
300     if(hist[i] > bin)
301     {
302         bin = hist[i];
303     }
304 }
305
306 // normalize the histogram between 0 and histImage.rows
307 for(int i = 0; i < 256; i++)
308 {
309     hist[i] = (float) hist[i] / bin;
310 }
311
312 // Find the maxium intensity element from Histogram
313 int bin = hist[0];
314 for(int i = 1; i < 256; i++)
315 {
316     if(hist[i] > bin)
317     {
318         bin = hist[i];
319     }
320 }
321
322 // normalize the histogram between 0 and histImage.rows
323 for(int i = 0; i < 256; i++)
324 {
325     hist[i] = (float) hist[i] / bin;
326 }
327
328 // Find the maxium intensity element from Histogram
329 int bin = hist[0];
330 for(int i = 1; i < 256; i++)
331 {
332     if(hist[i] > bin)
333     {
334         bin = hist[i];
335     }
336 }
337
338 // normalize the histogram between 0 and histImage.rows
339 for(int i = 0; i < 256; i++)
340 {
341     hist[i] = (float) hist[i] / bin;
342 }
343
344 // Find the maxium intensity element from Histogram
345 int bin = hist[0];
346 for(int i = 1; i < 256; i++)
347 {
348     if(hist[i] > bin)
349     {
350         bin = hist[i];
351     }
352 }
353
354 // normalize the histogram between 0 and histImage.rows
355 for(int i = 0; i < 256; i++)
356 {
357     hist[i] = (float) hist[i] / bin;
358 }
359
360 // Find the maxium intensity element from Histogram
361 int bin = hist[0];
362 for(int i = 1; i < 256; i++)
363 {
364     if(hist[i] > bin)
365     {
366         bin = hist[i];
367     }
368 }
369
370 // normalize the histogram between 0 and histImage.rows
371 for(int i = 0; i < 256; i++)
372 {
373     hist[i] = (float) hist[i] / bin;
374 }
375
376 // Find the maxium intensity element from Histogram
377 int bin = hist[0];
378 for(int i = 1; i < 256; i++)
379 {
380     if(hist[i] > bin)
381     {
382         bin = hist[i];
383     }
384 }
385
386 // normalize the histogram between 0 and histImage.rows
387 for(int i = 0; i < 256; i++)
388 {
389     hist[i] = (float) hist[i] / bin;
390 }
391
392 // Find the maxium intensity element from Histogram
393 int bin = hist[0];
394 for(int i = 1; i < 256; i++)
395 {
396     if(hist[i] > bin)
397     {
398         bin = hist[i];
399     }
400 }
401
402 // normalize the histogram between 0 and histImage.rows
403 for(int i = 0; i < 256; i++)
404 {
405     hist[i] = (float) hist[i] / bin;
406 }
407
408 // Find the maxium intensity element from Histogram
409 int bin = hist[0];
410 for(int i = 1; i < 256; i++)
411 {
412     if(hist[i] > bin)
413     {
414         bin = hist[i];
415     }
416 }
417
418 // normalize the histogram between 0 and histImage.rows
419 for(int i = 0; i < 256; i++)
420 {
421     hist[i] = (float) hist[i] / bin;
422 }
423
424 // Find the maxium intensity element from Histogram
425 int bin = hist[0];
426 for(int i = 1; i < 256; i++)
427 {
428     if(hist[i] > bin)
429     {
430         bin = hist[i];
431     }
432 }
433
434 // normalize the histogram between 0 and histImage.rows
435 for(int i = 0; i < 256; i++)
436 {
437     hist[i] = (float) hist[i] / bin;
438 }
439
440 // Find the maxium intensity element from Histogram
441 int bin = hist[0];
442 for(int i = 1; i < 256; i++)
443 {
444     if(hist[i] > bin)
445     {
446         bin = hist[i];
447     }
448 }
449
450 // normalize the histogram between 0 and histImage.rows
451 for(int i = 0; i < 256; i++)
452 {
453     hist[i] = (float) hist[i] / bin;
454 }
455
456 // Find the maxium intensity element from Histogram
457 int bin = hist[0];
458 for(int i = 1; i < 256; i++)
459 {
460     if(hist[i] > bin)
461     {
462         bin = hist[i];
463     }
464 }
465
466 // normalize the histogram between 0 and histImage.rows
467 for(int i = 0; i < 256; i++)
468 {
469     hist[i] = (float) hist[i] / bin;
470 }
471
472 // Find the maxium intensity element from Histogram
473 int bin = hist[0];
474 for(int i = 1; i < 256; i++)
475 {
476     if(hist[i] > bin)
477     {
478         bin = hist[i];
479     }
480 }
481
482 // normalize the histogram between 0 and histImage.rows
483 for(int i = 0; i < 256; i++)
484 {
485     hist[i] = (float) hist[i] / bin;
486 }
487
488 // Find the maxium intensity element from Histogram
489 int bin = hist[0];
490 for(int i = 1; i < 256; i++)
491 {
492     if(hist[i] > bin)
493     {
494         bin = hist[i];
495     }
496 }
497
498 // normalize the histogram between 0 and histImage.rows
499 for(int i = 0; i < 256; i++)
500 {
501     hist[i] = (float) hist[i] / bin;
502 }
503
504 // Find the maxium intensity element from Histogram
505 int bin = hist[0];
506 for(int i = 1; i < 256; i++)
507 {
508     if(hist[i] > bin)
509     {
510         bin = hist[i];
511     }
512 }
513
514 // normalize the histogram between 0 and histImage.rows
515 for(int i = 0; i < 256; i++)
516 {
517     hist[i] = (float) hist[i] / bin;
518 }
519
520 // Find the maxium intensity element from Histogram
521 int bin = hist[0];
522 for(int i = 1; i < 256; i++)
523 {
524     if(hist[i] > bin)
525     {
526         bin = hist[i];
527     }
528 }
529
530 // normalize the histogram between 0 and histImage.rows
531 for(int i = 0; i < 256; i++)
532 {
533     hist[i] = (float) hist[i] / bin;
534 }
535
536 // Find the maxium intensity element from Histogram
537 int bin = hist[0];
538 for(int i = 1; i < 256; i++)
539 {
540     if(hist[i] > bin)
541     {
542         bin = hist[i];
543     }
544 }
545
546 // normalize the histogram between 0 and histImage.rows
547 for(int i = 0; i < 256; i++)
548 {
549     hist[i] = (float) hist[i] / bin;
550 }
551
552 // Find the maxium intensity element from Histogram
553 int bin = hist[0];
554 for(int i = 1; i < 256; i++)
555 {
556     if(hist[i] > bin)
557     {
558         bin = hist[i];
559     }
560 }
561
562 // normalize the histogram between 0 and histImage.rows
563 for(int i = 0; i < 256; i++)
564 {
565     hist[i] = (float) hist[i] / bin;
566 }
567
568 // Find the maxium intensity element from Histogram
569 int bin = hist[0];
570 for(int i = 1; i < 256; i++)
571 {
572     if(hist[i] > bin)
573     {
574         bin = hist[i];
575     }
576 }
577
578 // normalize the histogram between 0 and histImage.rows
579 for(int i = 0; i < 256; i++)
580 {
581     hist[i] = (float) hist[i] / bin;
582 }
583
584 // Find the maxium intensity element from Histogram
585 int bin = hist[0];
586 for(int i = 1; i < 256; i++)
587 {
588     if(hist[i] > bin)
589     {
590         bin = hist[i];
591     }
592 }
593
594 // normalize the histogram between 0 and histImage.rows
595 for(int i = 0; i < 256; i++)
596 {
597     hist[i] = (float) hist[i] / bin;
598 }
599
600 // Find the maxium intensity element from Histogram
601 int bin = hist[0];
602 for(int i = 1; i < 256; i++)
603 {
604     if(hist[i] > bin)
605     {
606         bin = hist[i];
607     }
608 }
609
610 // normalize the histogram between 0 and histImage.rows
611 for(int i = 0; i < 256; i++)
612 {
613     hist[i] = (float) hist[i] / bin;
614 }
615
616 // Find the maxium intensity element from Histogram
617 int bin = hist[0];
618 for(int i = 1; i < 256; i++)
619 {
620     if(hist[i] > bin)
621     {
622         bin = hist[i];
623     }
624 }
625
626 // normalize the histogram between 0 and histImage.rows
627 for(int i = 0; i < 256; i++)
628 {
629     hist[i] = (float) hist[i] / bin;
630 }
631
632 // Find the maxium intensity element from Histogram
633 int bin = hist[0];
634 for(int i = 1; i < 256; i++)
635 {
636     if(hist[i] > bin)
637     {
638         bin = hist[i];
639     }
640 }
641
642 // normalize the histogram between 0 and histImage.rows
643 for(int i = 0; i < 256; i++)
644 {
645     hist[i] = (float) hist[i] / bin;
646 }
647
648 // Find the maxium intensity element from Histogram
649 int bin = hist[0];
650 for(int i = 1; i < 256; i++)
651 {
652     if(hist[i] > bin)
653     {
654         bin = hist[i];
655     }
656 }
657
658 // normalize the histogram between 0 and histImage.rows
659 for(int i = 0; i < 256; i++)
660 {
661     hist[i] = (float) hist[i] / bin;
662 }
663
664 // Find the maxium intensity element from Histogram
665 int bin = hist[0];
666 for(int i = 1; i < 256; i++)
667 {
668     if(hist[i] > bin)
669     {
670         bin = hist[i];
671     }
672 }
673
674 // normalize the histogram between 0 and histImage.rows
675 for(int i = 0; i < 256; i++)
676 {
677     hist[i] = (float) hist[i] / bin;
678 }
679
680 // Find the maxium intensity element from Histogram
681 int bin = hist[0];
682 for(int i = 1; i < 256; i++)
683 {
684     if(hist[i] > bin)
685     {
686         bin = hist[i];
687     }
688 }
689
690 // normalize the histogram between 0 and histImage.rows
691 for(int i = 0; i < 256; i++)
692 {
693     hist[i] = (float) hist[i] / bin;
694 }
695
696 // Find the maxium intensity element from Histogram
697 int bin = hist[0];
698 for(int i = 1; i < 256; i++)
699 {
700     if(hist[i] > bin)
701     {
702         bin = hist[i];
703     }
704 }
705
706 // normalize the histogram between 0 and histImage.rows
707 for(int i = 0; i < 256; i++)
708 {
709     hist[i] = (float) hist[i] / bin;
710 }
711
712 // Find the maxium intensity element from Histogram
713 int bin = hist[0];
714 for(int i = 1; i < 256; i++)
715 {
716     if(hist[i] > bin)
717     {
718         bin = hist[i];
719     }
720 }
721
722 // normalize the histogram between 0 and histImage.rows
723 for(int i = 0; i < 256; i++)
724 {
725     hist[i] = (float) hist[i] / bin;
726 }
727
728 // Find the maxium intensity element from Histogram
729 int bin = hist[0];
730 for(int i = 1; i < 256; i++)
731 {
732     if(hist[i] > bin)
733     {
734         bin = hist[i];
735     }
736 }
737
738 // normalize the histogram between 0 and histImage.rows
739 for(int i = 0; i < 256; i++)
740 {
741     hist[i] = (float) hist[i] / bin;
742 }
743
744 // Find the maxium intensity element from Histogram
745 int bin = hist[0];
746 for(int i = 1; i < 256; i++)
747 {
748     if(hist[i] > bin)
749     {
750         bin = hist[i];
751     }
752 }
753
754 // normalize the histogram between 0 and histImage.rows
755 for(int i = 0; i < 256; i++)
756 {
757     hist[i] = (float) hist[i] / bin;
758 }
759
760 // Find the maxium intensity element from Histogram
761 int bin = hist[0];
762 for(int i = 1; i < 256; i++)
763 {
764     if(hist[i] > bin)
765     {
766         bin = hist[i];
767     }
768 }
769
770 // normalize the histogram between 0 and histImage.rows
771 for(int i = 0; i < 256; i++)
772 {
773     hist[i] = (float) hist[i] / bin;
774 }
775
776 // Find the maxium intensity element from Histogram
777 int bin = hist[0];
778 for(int i = 1; i < 256; i++)
779 {
780     if(hist[i] > bin)
781     {
782         bin = hist[i];
783     }
784 }
785
786 // normalize the histogram between 0 and histImage.rows
787 for(int i = 0; i < 256; i++)
788 {
789     hist[i] = (float) hist[i] / bin;
790 }
791
792 // Find the maxium intensity element from Histogram
793 int bin = hist[0];
794 for(int i = 1; i < 256; i++)
795 {
796     if(hist[i] > bin)
797     {
798         bin = hist[i];
799     }
800 }
801
802 // normalize the histogram between 0 and histImage.rows
803 for(int i = 0; i < 256; i++)
804 {
805     hist[i] = (float) hist[i] / bin;
806 }
807
808 // Find the maxium intensity element from Histogram
809 int bin = hist[0];
810 for(int i = 1; i < 256; i++)
811 {
812     if(hist[i] > bin)
813     {
814         bin = hist[i];
815     }
816 }
817
818 // normalize the histogram between 0 and histImage.rows
819 for(int i = 0; i < 256; i++)
820 {
821     hist[i] = (float) hist[i] / bin;
822 }
823
824 // Find the maxium intensity element from Histogram
825 int bin = hist[0];
826 for(int i = 1; i < 256; i++)
827 {
828     if(hist[i] > bin)
829     {
830         bin = hist[i];
831     }
832 }
833
834 // normalize the histogram between 0 and histImage.rows
835 for(int i = 0; i < 256; i++)
836 {
837     hist[i] = (float) hist[i] / bin;
838 }
839
840 // Find the maxium intensity element from Histogram
841 int bin = hist[0];
842 for(int i = 1; i < 256; i++)
843 {
844     if(hist[i] > bin)
845     {
846         bin = hist[i];
847     }
848 }
849
850 // normalize the histogram between 0 and histImage.rows
851 for(int i = 0; i < 256; i++)
852 {
853     hist[i] = (float) hist[i] / bin;
854 }
855
856 // Find the maxium intensity element from Histogram
857 int bin = hist[0];
858 for(int i = 1; i < 256; i++)
859 {
860     if(hist[i] > bin)
861     {
862         bin = hist[i];
863     }
864 }
865
866 // normalize the histogram between 0 and histImage.rows
867 for(int i = 0; i < 256; i++)
868 {
869     hist[i] = (float) hist[i] / bin;
870 }
871
872 // Find the maxium intensity element from Histogram
873 int bin = hist[0];
874 for(int i = 1; i < 256; i++)
875 {
876     if(hist[i] > bin)
877     {
878         bin = hist[i];
879     }
880 }
881
882 // normalize the histogram between 0 and histImage.rows
883 for(int i = 0; i < 256; i++)
884 {
885     hist[i] = (float) hist[i] / bin;
886 }
887
888 // Find the maxium intensity element from Histogram
889 int bin = hist[0];
890 for(int i = 1; i < 256; i++)
891 {
892     if(hist[i] > bin)
893     {
894         bin = hist[i];
895     }
896 }
897
898 // normalize the histogram between 0 and histImage.rows
899 for(int i = 0; i < 256; i++)
900 {
901     hist[i] = (float) hist[i] / bin;
902 }
903
904 // Find the maxium intensity element from Histogram
905 int bin = hist[0];
906 for(int i = 1; i < 256; i++)
907 {
908     if(hist[i] > bin)
909     {
910         bin = hist[i];
911     }
912 }
913
914 // normalize the histogram between 0 and histImage.rows
915 for(int i = 0; i < 256; i++)
916 {
917     hist[i] = (float) hist[i] / bin;
918 }
919
920 // Find the maxium intensity element from Histogram
921 int bin = hist[0];
922 for(int i = 1; i < 256; i++)
923 {
924     if(hist[i] > bin)
925     {
926         bin = hist[i];
927     }
928 }
929
930 // normalize the histogram between 0 and histImage.rows
931 for(int i = 0; i < 256; i++)
932 {
933     hist[i] = (float) hist[i] / bin;
934 }
935
936 // Find the maxium intensity element from Histogram
937 int bin = hist[0];
938 for(int i = 1; i < 256; i++)
939 {
940     if(hist[i] > bin)
941     {
942         bin = hist[i];
943     }
944 }
945
946 // normalize the histogram between 0 and histImage.rows
947 for(int i = 0; i < 256; i++)
948 {
949     hist[i] = (float) hist[i] / bin;
950 }
951
952 // Find the maxium intensity element from Histogram
953 int bin = hist[0];
954 for(int i = 1; i < 256; i++)
955 {
956     if(hist[i] > bin)
957     {
958         bin = hist[i];
959     }
960 }
961
962 // normalize the histogram between 0 and histImage.rows
963 for(int i = 0; i < 256; i++)
964 {
965     hist[i] = (float) hist[i] / bin;
966 }
967
968 // Find the maxium intensity element from Histogram
969 int bin = hist[0];
970 for(int i = 1; i < 256; i++)
971 {
972     if(hist[i] > bin)
973     {
974         bin = hist[i];
975     }
976 }
977
978 // normalize the histogram between 0 and histImage.rows
979 for(int i = 0; i < 256; i++)
980 {
981     hist[i] = (float) hist[i] / bin;
982 }
983
984 // Find the maxium intensity element from Histogram
985 int bin = hist[0];
986 for(int i = 1; i < 256; i++)
987 {
988     if(hist[i] > bin)
989     {
990         bin = hist[i];
991     }
992 }
993
994 // normalize the histogram between 0 and histImage.rows
995 for(int i = 0; i < 256; i++)
996 {
997     hist[i] = (float) hist[i] / bin;
998 }
999
1000 // Find the maxium intensity element from Histogram
1001 int bin = hist[0];
1002 for(int i = 1; i < 256; i++)
1003 {
1004     if(hist[i] > bin)
1005     {
1006         bin = hist[i];
1007     }
1008 }
1009
1010 // normalize the histogram between 0 and histImage.rows
1011 for(int i = 0; i < 256; i++)
1012 {
1013     hist[i] = (float) hist[i] / bin;
1014 }
1015
1016 // Find the maxium intensity element from Histogram
1017 int bin = hist[0];
1018 for(int i = 1; i < 256; i++)
1019 {
1020     if(hist[i] > bin)
1021     {
1022         bin = hist[i];
1023     }
1024 }
1025
1026 // normalize the histogram between 0 and histImage.rows
1027 for(int i = 0; i < 256; i++)
1028 {
1029     hist[i] = (float) hist[i] / bin;
1030 }
1031
1032 // Find the maxium intensity element from Histogram
1033 int bin = hist[0];
1034 for(int i = 1; i < 256; i++)
1035 {
1036     if(hist[i] > bin)
1037     {
1038         bin = hist[i];
1039     }
1040 }
1041
1042 // normalize the histogram between 0 and histImage.rows
1043 for(int i = 0; i < 256; i++)
1044 {
1045     hist[i] = (float) hist[i] / bin;
1046 }
1047
1048 // Find the maxium intensity element from Histogram
1049 int bin = hist[0];
1050 for(int i = 1; i < 256; i++)
1051 {
1052     if(hist[i] > bin)
1053     {
1054         bin = hist[i];
1055     }
1056 }
1057
1058 // normalize the histogram between 0 and histImage.rows
1059 for(int i = 0; i < 256; i++)
1060 {
1061     hist[i] = (float) hist[i] / bin;
1062 }
1063
1064 // Find the maxium intensity element from Histogram
1065 int bin = hist[0];
1066 for(int i = 1; i < 256; i++)
1067 {
1068     if(hist[i] > bin)
1069     {
1070         bin = hist[i];
1071     }
1072 }
1073
1074 // normalize the histogram between 0 and histImage.rows
1075 for(int i = 0; i < 256; i++)
1076 {
1077     hist[i] = (float) hist[i] / bin;
1078 }
1079
1080 // Find the maxium intensity element from Histogram
1081 int bin = hist[0];
1082 for(int i = 1; i < 256; i++)
1083 {
1084     if(hist[i] > bin)
1085     {
1086         bin = hist[i];
1087     }
1088 }
1089
1090 // normalize the histogram between 0 and histImage.rows
1091 for(int i = 0; i < 256; i++)
1092 {
1093     hist[i] = (float) hist[i] / bin;
1094 }
1095
1096 // Find the maxium intensity element from Histogram
1097 int bin = hist[0];
1098 for(int i = 1; i < 256; i++)
1099 {
1100     if(hist[i] > bin)
1101     {
1102         bin = hist[i];
1103     }
1104 }
1105
1106 // normalize the histogram between 0 and histImage.rows
1107 for(int i = 0; i < 256; i++)
1108 {
1109     hist[i] = (float) hist[i] / bin;
1110 }
1111
1112 // Find the maxium intensity element from Histogram
1113 int bin = hist[0];
1114 for(int i = 1; i < 256; i++)
1115 {
1116     if(hist[i] > bin)
1117     {
1118         bin = hist[i];
1119     }
1120 }
1121
1122 // normalize the histogram between 0 and histImage.rows
1123 for(int i = 0; i < 256; i++)
1124 {
1125     hist[i] = (float) hist[i] / bin;
1126 }
1127
1128 // Find the maxium intensity element from Histogram
1129 int bin = hist[0];
1130 for(int i = 1; i < 256; i++)
1131 {
1132     if(hist[i] > bin)
1133     {
1134         bin = hist[i];
1135     }
1136 }
1137
1138 // normalize the histogram between 0 and histImage.rows
1139 for(int i = 0; i < 256; i++)
1140 {
1141     hist[i] = (float) hist[i] / bin;
1142 }
1143
1144 // Find the maxium intensity element from Histogram
1145 int bin = hist[0];
1146 for(int i = 1; i < 256; i++)
1147 {
1148     if(hist[i] > bin)
1149     {
1150         bin = hist[i];
1151     }
1152 }
1153
1154 // normalize the histogram between 0 and histImage.rows
1155 for(int i = 0; i < 256; i++)
1156 {
1157     hist[i] = (float) hist[i] / bin;
1158 }
1159
1160 // Find the maxium intensity element from Histogram
1161 int bin = hist[0];
1162 for(int i
```

- 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

ODIAME AHORA
POR EXIGIRTE
COMO ESTUDIANTE
PARA QUE NO ME TENGAS QUE
ODIAR DESPUÉS
POR HACERTE
UN PROFESIONAL MEDIOCRE