

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.

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

- 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.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




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




Salon de la Fama de Entregas Completas e Incompletas



Nombre
 iti-271229_u1_palmero_torres_javier_martin_latex
 iti-271229_u1_palmero_torres_javier_martin_source
 iti-271229_u2_palmero_torres_javier_martin.pdf

Nombre
 iti_271229_u1_perez_reyes_omar_alejandro_latex
 iti_271229_u1_perez_reyes_omar_alejandro_source
 iti_271229_u1_perez_reyes_omar_alejandro_latex.pdf

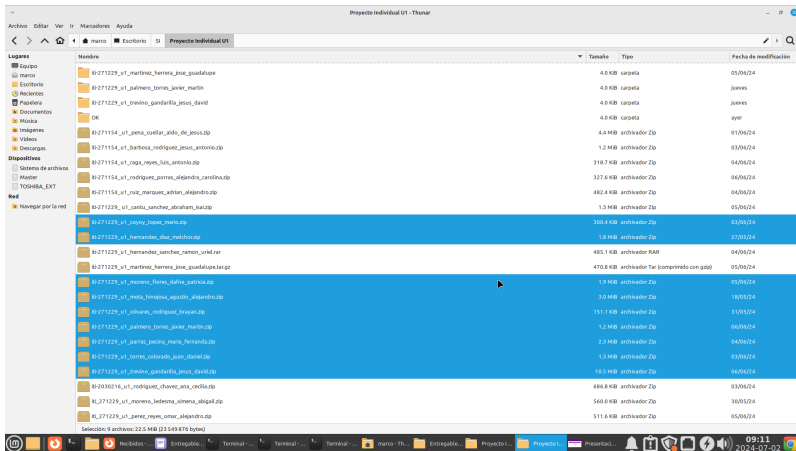
Nombre
 iti-271229_u1_olivares_rodriguez_brayan_source
 iti-271229_u1_olivares_rodriguez_brayan_latex
 iti-271229_u1_olivares_rodriguez_brayan.pdf

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

Premio a la Compresión Lectora 2024



Fechas importantes de entrega de proyectos (1)

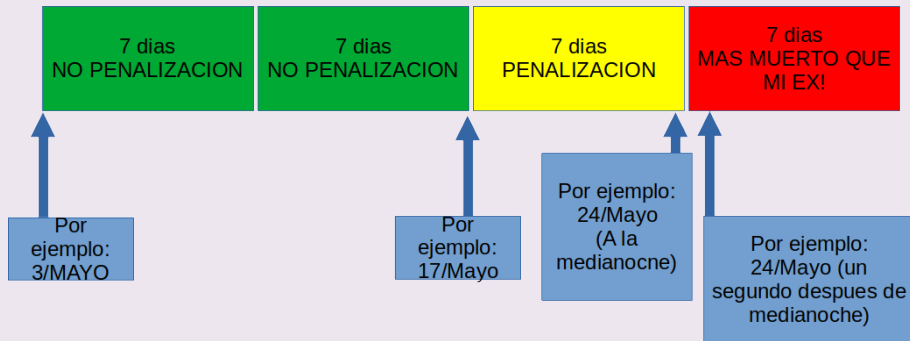
- Fecha de asignación: fecha en que se da a conocer al grupo el trabajo a elaborar
- Fecha de entrega sin penalización: 14 días naturales después de la fecha de asignación
- Proyecto entregado después de la fecha de penalización se le aplica una penalización de 20 PUNTOS
- Fecha de cierre: 21 días naturales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

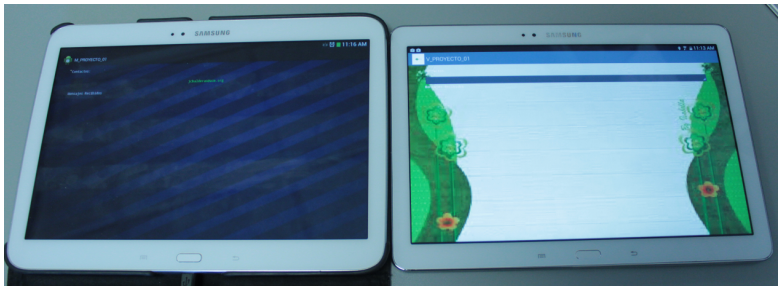
Sobre una instalación de Linux, se debe instalar lo siguiente:

- 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



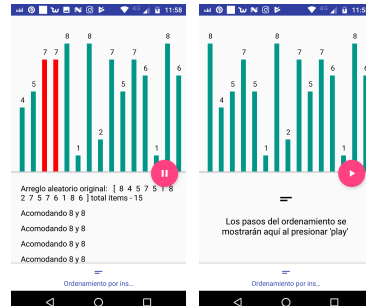
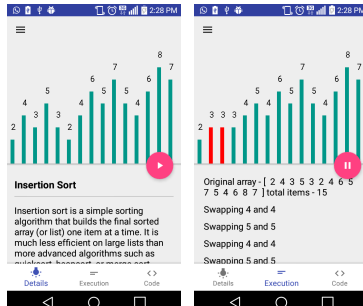
```

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 // Calculando los histogramas
73 int hist_w = 128; int hist_h = 128;
74 int bin_w = cvRound( (double) hist_w/256);
75 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
76
77 // Find the maxium intensity element from Histogram
78 int bin = hist[0];
79 for(int i = 1; i < 256; i++)
80 {
81     if(hist[i] > bin)
82     {
83         bin = hist[i];
84     }
85 }
86
87 // normalize the histogram between 0 and histImage.rows
88 for(int i = 0; i < 256; i++)
89 {
90     hist[i] = (float) hist[i] / bin;
91 }
92
93 // Calculando los histogramas
94 int hist_w = 128; int hist_h = 128;
95 int bin_w = cvRound( (double) hist_w/256);
96 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
97
98 // Find the maxium intensity element from Histogram
99 int bin = hist[0];
100 for(int i = 1; i < 256; i++)
101 {
102     if(hist[i] > bin)
103     {
104         bin = hist[i];
105     }
106 }
107
108 // normalize the histogram between 0 and histImage.rows
109 for(int i = 0; i < 256; i++)
110 {
111     hist[i] = (float) hist[i] / bin;
112 }
113
114 // Calculando los histogramas
115 int hist_w = 128; int hist_h = 128;
116 int bin_w = cvRound( (double) hist_w/256);
117 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
118
119 // Find the maxium intensity element from Histogram
120 int bin = hist[0];
121 for(int i = 1; i < 256; i++)
122 {
123     if(hist[i] > bin)
124     {
125         bin = hist[i];
126     }
127 }
128
129 // normalize the histogram between 0 and histImage.rows
130 for(int i = 0; i < 256; i++)
131 {
132     hist[i] = (float) hist[i] / bin;
133 }
134
135 // Calculando los histogramas
136 int hist_w = 128; int hist_h = 128;
137 int bin_w = cvRound( (double) hist_w/256);
138 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
139
140 // Find the maxium intensity element from Histogram
141 int bin = hist[0];
142 for(int i = 1; i < 256; i++)
143 {
144     if(hist[i] > bin)
145     {
146         bin = hist[i];
147     }
148 }
149
150 // normalize the histogram between 0 and histImage.rows
151 for(int i = 0; i < 256; i++)
152 {
153     hist[i] = (float) hist[i] / bin;
154 }
155
156 // Calculando los histogramas
157 int hist_w = 128; int hist_h = 128;
158 int bin_w = cvRound( (double) hist_w/256);
159 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
160
161 // Find the maxium intensity element from Histogram
162 int bin = hist[0];
163 for(int i = 1; i < 256; i++)
164 {
165     if(hist[i] > bin)
166     {
167         bin = hist[i];
168     }
169 }
170
171 // normalize the histogram between 0 and histImage.rows
172 for(int i = 0; i < 256; i++)
173 {
174     hist[i] = (float) hist[i] / bin;
175 }
176
177 // Calculando los histogramas
178 int hist_w = 128; int hist_h = 128;
179 int bin_w = cvRound( (double) hist_w/256);
180 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
181
182 // Find the maxium intensity element from Histogram
183 int bin = hist[0];
184 for(int i = 1; i < 256; i++)
185 {
186     if(hist[i] > bin)
187     {
188         bin = hist[i];
189     }
190 }
191
192 // normalize the histogram between 0 and histImage.rows
193 for(int i = 0; i < 256; i++)
194 {
195     hist[i] = (float) hist[i] / bin;
196 }
197
198 // Calculando los histogramas
199 int hist_w = 128; int hist_h = 128;
200 int bin_w = cvRound( (double) hist_w/256);
201 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
202
203 // Find the maxium intensity element from Histogram
204 int bin = hist[0];
205 for(int i = 1; i < 256; i++)
206 {
207     if(hist[i] > bin)
208     {
209         bin = hist[i];
210     }
211 }
212
213 // normalize the histogram between 0 and histImage.rows
214 for(int i = 0; i < 256; i++)
215 {
216     hist[i] = (float) hist[i] / bin;
217 }
218
219 // Calculando los histogramas
220 int hist_w = 128; int hist_h = 128;
221 int bin_w = cvRound( (double) hist_w/256);
222 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
223
224 // Find the maxium intensity element from Histogram
225 int bin = hist[0];
226 for(int i = 1; i < 256; i++)
227 {
228     if(hist[i] > bin)
229     {
230         bin = hist[i];
231     }
232 }
233
234 // normalize the histogram between 0 and histImage.rows
235 for(int i = 0; i < 256; i++)
236 {
237     hist[i] = (float) hist[i] / bin;
238 }
239
240 // Calculando los histogramas
241 int hist_w = 128; int hist_h = 128;
242 int bin_w = cvRound( (double) hist_w/256);
243 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
244
245 // Find the maxium intensity element from Histogram
246 int bin = hist[0];
247 for(int i = 1; i < 256; i++)
248 {
249     if(hist[i] > bin)
250     {
251         bin = hist[i];
252     }
253 }
254
255 // normalize the histogram between 0 and histImage.rows
256 for(int i = 0; i < 256; i++)
257 {
258     hist[i] = (float) hist[i] / bin;
259 }
260
261 // Calculando los histogramas
262 int hist_w = 128; int hist_h = 128;
263 int bin_w = cvRound( (double) hist_w/256);
264 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
265
266 // Find the maxium intensity element from Histogram
267 int bin = hist[0];
268 for(int i = 1; i < 256; i++)
269 {
270     if(hist[i] > bin)
271     {
272         bin = hist[i];
273     }
274 }
275
276 // normalize the histogram between 0 and histImage.rows
277 for(int i = 0; i < 256; i++)
278 {
279     hist[i] = (float) hist[i] / bin;
280 }
281
282 // Calculando los histogramas
283 int hist_w = 128; int hist_h = 128;
284 int bin_w = cvRound( (double) hist_w/256);
285 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
286
287 // Find the maxium intensity element from Histogram
288 int bin = hist[0];
289 for(int i = 1; i < 256; i++)
290 {
291     if(hist[i] > bin)
292     {
293         bin = hist[i];
294     }
295 }
296
297 // normalize the histogram between 0 and histImage.rows
298 for(int i = 0; i < 256; i++)
299 {
300     hist[i] = (float) hist[i] / bin;
301 }
302
303 // Calculando los histogramas
304 int hist_w = 128; int hist_h = 128;
305 int bin_w = cvRound( (double) hist_w/256);
306 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
307
308 // Find the maxium intensity element from Histogram
309 int bin = hist[0];
310 for(int i = 1; i < 256; i++)
311 {
312     if(hist[i] > bin)
313     {
314         bin = hist[i];
315     }
316 }
317
318 // normalize the histogram between 0 and histImage.rows
319 for(int i = 0; i < 256; i++)
320 {
321     hist[i] = (float) hist[i] / bin;
322 }
323
324 // Calculando los histogramas
325 int hist_w = 128; int hist_h = 128;
326 int bin_w = cvRound( (double) hist_w/256);
327 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
328
329 // Find the maxium intensity element from Histogram
330 int bin = hist[0];
331 for(int i = 1; i < 256; i++)
332 {
333     if(hist[i] > bin)
334     {
335         bin = hist[i];
336     }
337 }
338
339 // normalize the histogram between 0 and histImage.rows
340 for(int i = 0; i < 256; i++)
341 {
342     hist[i] = (float) hist[i] / bin;
343 }
344
345 // Calculando los histogramas
346 int hist_w = 128; int hist_h = 128;
347 int bin_w = cvRound( (double) hist_w/256);
348 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
349
350 // Find the maxium intensity element from Histogram
351 int bin = hist[0];
352 for(int i = 1; i < 256; i++)
353 {
354     if(hist[i] > bin)
355     {
356         bin = hist[i];
357     }
358 }
359
360 // normalize the histogram between 0 and histImage.rows
361 for(int i = 0; i < 256; i++)
362 {
363     hist[i] = (float) hist[i] / bin;
364 }
365
366 // Calculando los histogramas
367 int hist_w = 128; int hist_h = 128;
368 int bin_w = cvRound( (double) hist_w/256);
369 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
370
371 // Find the maxium intensity element from Histogram
372 int bin = hist[0];
373 for(int i = 1; i < 256; i++)
374 {
375     if(hist[i] > bin)
376     {
377         bin = hist[i];
378     }
379 }
380
381 // normalize the histogram between 0 and histImage.rows
382 for(int i = 0; i < 256; i++)
383 {
384     hist[i] = (float) hist[i] / bin;
385 }
386
387 // Calculando los histogramas
388 int hist_w = 128; int hist_h = 128;
389 int bin_w = cvRound( (double) hist_w/256);
390 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
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 // Calculando los histogramas
409 int hist_w = 128; int hist_h = 128;
410 int bin_w = cvRound( (double) hist_w/256);
411 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
412
413 // Find the maxium intensity element from Histogram
414 int bin = hist[0];
415 for(int i = 1; i < 256; i++)
416 {
417     if(hist[i] > bin)
418     {
419         bin = hist[i];
420     }
421 }
422
423 // normalize the histogram between 0 and histImage.rows
424 for(int i = 0; i < 256; i++)
425 {
426     hist[i] = (float) hist[i] / bin;
427 }
428
429 // Calculando los histogramas
430 int hist_w = 128; int hist_h = 128;
431 int bin_w = cvRound( (double) hist_w/256);
432 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
433
434 // Find the maxium intensity element from Histogram
435 int bin = hist[0];
436 for(int i = 1; i < 256; i++)
437 {
438     if(hist[i] > bin)
439     {
440         bin = hist[i];
441     }
442 }
443
444 // normalize the histogram between 0 and histImage.rows
445 for(int i = 0; i < 256; i++)
446 {
447     hist[i] = (float) hist[i] / bin;
448 }
449
450 // Calculando los histogramas
451 int hist_w = 128; int hist_h = 128;
452 int bin_w = cvRound( (double) hist_w/256);
453 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
454
455 // Find the maxium intensity element from Histogram
456 int bin = hist[0];
457 for(int i = 1; i < 256; i++)
458 {
459     if(hist[i] > bin)
460     {
461         bin = hist[i];
462     }
463 }
464
465 // normalize the histogram between 0 and histImage.rows
466 for(int i = 0; i < 256; i++)
467 {
468     hist[i] = (float) hist[i] / bin;
469 }
470
471 // Calculando los histogramas
472 int hist_w = 128; int hist_h = 128;
473 int bin_w = cvRound( (double) hist_w/256);
474 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
475
476 // Find the maxium intensity element from Histogram
477 int bin = hist[0];
478 for(int i = 1; i < 256; i++)
479 {
480     if(hist[i] > bin)
481     {
482         bin = hist[i];
483     }
484 }
485
486 // normalize the histogram between 0 and histImage.rows
487 for(int i = 0; i < 256; i++)
488 {
489     hist[i] = (float) hist[i] / bin;
490 }
491
492 // Calculando los histogramas
493 int hist_w = 128; int hist_h = 128;
494 int bin_w = cvRound( (double) hist_w/256);
495 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
496
497 // Find the maxium intensity element from Histogram
498 int bin = hist[0];
499 for(int i = 1; i < 256; i++)
500 {
501     if(hist[i] > bin)
502     {
503         bin = hist[i];
504     }
505 }
506
507 // normalize the histogram between 0 and histImage.rows
508 for(int i = 0; i < 256; i++)
509 {
510     hist[i] = (float) hist[i] / bin;
511 }
512
513 // Calculando los histogramas
514 int hist_w = 128; int hist_h = 128;
515 int bin_w = cvRound( (double) hist_w/256);
516 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
517
518 // Find the maxium intensity element from Histogram
519 int bin = hist[0];
520 for(int i = 1; i < 256; i++)
521 {
522     if(hist[i] > bin)
523     {
524         bin = hist[i];
525     }
526 }
527
528 // normalize the histogram between 0 and histImage.rows
529 for(int i = 0; i < 256; i++)
530 {
531     hist[i] = (float) hist[i] / bin;
532 }
533
534 // Calculando los histogramas
535 int hist_w = 128; int hist_h = 128;
536 int bin_w = cvRound( (double) hist_w/256);
537 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
538
539 // Find the maxium intensity element from Histogram
540 int bin = hist[0];
541 for(int i = 1; i < 256; i++)
542 {
543     if(hist[i] > bin)
544     {
545         bin = hist[i];
546     }
547 }
548
549 // normalize the histogram between 0 and histImage.rows
550 for(int i = 0; i < 256; i++)
551 {
552     hist[i] = (float) hist[i] / bin;
553 }
554
555 // Calculando los histogramas
556 int hist_w = 128; int hist_h = 128;
557 int bin_w = cvRound( (double) hist_w/256);
558 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
559
560 // Find the maxium intensity element from Histogram
561 int bin = hist[0];
562 for(int i = 1; i < 256; i++)
563 {
564     if(hist[i] > bin)
565     {
566         bin = hist[i];
567     }
568 }
569
570 // normalize the histogram between 0 and histImage.rows
571 for(int i = 0; i < 256; i++)
572 {
573     hist[i] = (float) hist[i] / bin;
574 }
575
576 // Calculando los histogramas
577 int hist_w = 128; int hist_h = 128;
578 int bin_w = cvRound( (double) hist_w/256);
579 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
580
581 // Find the maxium intensity element from Histogram
582 int bin = hist[0];
583 for(int i = 1; i < 256; i++)
584 {
585     if(hist[i] > bin)
586     {
587         bin = hist[i];
588     }
589 }
590
591 // normalize the histogram between 0 and histImage.rows
592 for(int i = 0; i < 256; i++)
593 {
594     hist[i] = (float) hist[i] / bin;
595 }
596
597 // Calculando los histogramas
598 int hist_w = 128; int hist_h = 128;
599 int bin_w = cvRound( (double) hist_w/256);
600 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
601
602 // Find the maxium intensity element from Histogram
603 int bin = hist[0];
604 for(int i = 1; i < 256; i++)
605 {
606     if(hist[i] > bin)
607     {
608         bin = hist[i];
609     }
610 }
611
612 // normalize the histogram between 0 and histImage.rows
613 for(int i = 0; i < 256; i++)
614 {
615     hist[i] = (float) hist[i] / bin;
616 }
617
618 // Calculando los histogramas
619 int hist_w = 128; int hist_h = 128;
620 int bin_w = cvRound( (double) hist_w/256);
621 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
622
623 // Find the maxium intensity element from Histogram
624 int bin = hist[0];
625 for(int i = 1; i < 256; i++)
626 {
627     if(hist[i] > bin)
628     {
629         bin = hist[i];
630     }
631 }
632
633 // normalize the histogram between 0 and histImage.rows
634 for(int i = 0; i < 256; i++)
635 {
636     hist[i] = (float) hist[i] / bin;
637 }
638
639 // Calculando los histogramas
640 int hist_w = 128; int hist_h = 128;
641 int bin_w = cvRound( (double) hist_w/256);
642 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
643
644 // Find the maxium intensity element from Histogram
645 int bin = hist[0];
646 for(int i = 1; i < 256; i++)
647 {
648     if(hist[i] > bin)
649     {
650         bin = hist[i];
651     }
652 }
653
654 // normalize the histogram between 0 and histImage.rows
655 for(int i = 0; i < 256; i++)
656 {
657     hist[i] = (float) hist[i] / bin;
658 }
659
660 // Calculando los histogramas
661 int hist_w = 128; int hist_h = 128;
662 int bin_w = cvRound( (double) hist_w/256);
663 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
664
665 // Find the maxium intensity element from Histogram
666 int bin = hist[0];
667 for(int i = 1; i < 256; i++)
668 {
669     if(hist[i] > bin)
670     {
671         bin = hist[i];
672     }
673 }
674
675 // normalize the histogram between 0 and histImage.rows
676 for(int i = 0; i < 256; i++)
677 {
678     hist[i] = (float) hist[i] / bin;
679 }
680
681 // Calculando los histogramas
682 int hist_w = 128; int hist_h = 128;
683 int bin_w = cvRound( (double) hist_w/256);
684 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
685
686 // Find the maxium intensity element from Histogram
687 int bin = hist[0];
688 for(int i = 1; i < 256; i++)
689 {
690     if(hist[i] > bin)
691     {
692         bin = hist[i];
693     }
694 }
695
696 // normalize the histogram between 0 and histImage.rows
697 for(int i = 0; i < 256; i++)
698 {
699     hist[i] = (float) hist[i] / bin;
700 }
701
702 // Calculando los histogramas
703 int hist_w = 128; int hist_h = 128;
704 int bin_w = cvRound( (double) hist_w/256);
705 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
706
707 // Find the maxium intensity element from Histogram
708 int bin = hist[0];
709 for(int i = 1; i < 256; i++)
710 {
711     if(hist[i] > bin)
712     {
713         bin = hist[i];
714     }
715 }
716
717 // normalize the histogram between 0 and histImage.rows
718 for(int i = 0; i < 256; i++)
719 {
720     hist[i] = (float) hist[i] / bin;
721 }
722
723 // Calculando los histogramas
724 int hist_w = 128; int hist_h = 128;
725 int bin_w = cvRound( (double) hist_w/256);
726 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
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 // Calculando los histogramas
745 int hist_w = 128; int hist_h = 128;
746 int bin_w = cvRound( (double) hist_w/256);
747 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
748
749 // Find the maxium intensity element from Histogram
750 int bin = hist[0];
751 for(int i = 1; i < 256; i++)
752 {
753     if(hist[i] > bin)
754     {
755         bin = hist[i];
756     }
757 }
758
759 // normalize the histogram between 0 and histImage.rows
760 for(int i = 0; i < 256; i++)
761 {
762     hist[i] = (float) hist[i] / bin;
763 }
764
765 // Calculando los histogramas
766 int hist_w = 128; int hist_h = 128;
767 int bin_w = cvRound( (double) hist_w/256);
768 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
769
770 // Find the maxium intensity element from Histogram
771 int bin = hist[0];
772 for(int i = 1; i < 256; i++)
773 {
774     if(hist[i] > bin)
775     {
776         bin = hist[i];
777     }
778 }
779
780 // normalize the histogram between 0 and histImage.rows
781 for(int i = 0; i < 256; i++)
782 {
783     hist[i] = (float) hist[i] / bin;
784 }
785
786 // Calculando los histogramas
787 int hist_w = 128; int hist_h = 128;
788 int bin_w = cvRound( (double) hist_w/256);
789 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
790
791 // Find the maxium intensity element from Histogram
792 int bin = hist[0];
793 for(int i = 1; i < 256; i++)
794 {
795     if(hist[i] > bin)
796     {
797         bin = hist[i];
798     }
799 }
800
801 // normalize the histogram between 0 and histImage.rows
802 for(int i = 0; i < 256; i++)
803 {
804     hist[i] = (float) hist[i] / bin;
805 }
806
807 // Calculando los histogramas
808 int hist_w = 128; int hist_h = 128;
809 int bin_w = cvRound( (double) hist_w/256);
810 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
811
812 // Find the maxium intensity element from Histogram
813 int bin = hist[0];
814 for(int i = 1; i < 256; i++)
815 {
816     if(hist[i] > bin)
817     {
818         bin = hist[i];
819     }
820 }
821
822 // normalize the histogram between 0 and histImage.rows
823 for(int i = 0; i < 256; i++)
824 {
825     hist[i] = (float) hist[i] / bin;
826 }
827
828 // Calculando los histogramas
829 int hist_w = 128; int hist_h = 128;
830 int bin_w = cvRound( (double) hist_w/256);
831 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
832
833 // Find the maxium intensity element from Histogram
834 int bin = hist[0];
835 for(int i = 1; i < 256; i++)
836 {
837     if(hist[i] > bin)
838     {
839         bin = hist[i];
840     }
841 }
842
843 // normalize the histogram between 0 and histImage.rows
844 for(int i = 0; i < 256; i++)
845 {
846     hist[i] = (float) hist[i] / bin;
847 }
848
849 // Calculando los histogramas
850 int hist_w = 128; int hist_h = 128;
851 int bin_w = cvRound( (double) hist_w/256);
852 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
853
854 // Find the maxium intensity element from Histogram
855 int bin = hist[0];
856 for(int i = 1; i < 256; i++)
857 {
858     if(hist[i] > bin)
859     {
860         bin = hist[i];
861     }
862 }
863
864 // normalize the histogram between 0 and histImage.rows
865 for(int i = 0; i < 256; i++)
866 {
867     hist[i] = (float) hist[i] / bin;
868 }
869
870 // Calculando los histogramas
871 int hist_w = 128; int hist_h = 128;
872 int bin_w = cvRound( (double) hist_w/256);
873 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
874
875 // Find the maxium intensity element from Histogram
876 int bin = hist[0];
877 for(int i = 1; i < 256; i++)
878 {
879     if(hist[i] > bin)
880     {
881         bin = hist[i];
882     }
883 }
884
885 // normalize the histogram between 0 and histImage.rows
886 for(int i = 0; i < 256; i++)
887 {
888     hist[i] = (float) hist[i] / bin;
889 }
890
891 // Calculando los histogramas
892 int hist_w = 128; int hist_h = 128;
893 int bin_w = cvRound( (double) hist_w/256);
894 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
895
896 // Find the maxium intensity element from Histogram
897 int bin = hist[0];
898 for(int i = 1; i < 256; i++)
899 {
900     if(hist[i] > bin)
901     {
902         bin = hist[i];
903     }
904 }
905
906 // normalize the histogram between 0 and histImage.rows
907 for(int i = 0; i < 256; i++)
908 {
909     hist[i] = (float) hist[i] / bin;
910 }
911
912 // Calculando los histogramas
913 int hist_w = 128; int hist_h = 128;
914 int bin_w = cvRound( (double) hist_w/256);
915 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
916
917 // Find the maxium intensity element from Histogram
918 int bin = hist[0];
919 for(int i = 1; i < 256; i++)
920 {
921     if(hist[i] > bin)
922     {
923         bin = hist[i];
924     }
925 }
926
927 // normalize the histogram between 0 and histImage.rows
928 for(int i = 0; i < 256; i++)
929 {
930     hist[i] = (float) hist[i] / bin;
931 }
932
933 // Calculando los histogramas
934 int hist_w = 128; int hist_h = 128;
935 int bin_w = cvRound( (double) hist_w/256);
936 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
937
938 // Find the maxium intensity element from Histogram
939 int bin = hist[0];
940 for(int i = 1; i < 256; i++)
941 {
942     if(hist[i] > bin)
943     {
944         bin = hist[i];
945     }
946 }
947
948 // normalize the histogram between 0 and histImage.rows
949 for(int i = 0; i < 256; i++)
950 {
951     hist[i] = (float) hist[i] / bin;
952 }
953
954 // Calculando los histogramas
955 int hist_w = 128; int hist_h = 128;
956 int bin_w = cvRound( (double) hist_w/256);
957 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
958
959 // Find the maxium intensity element from Histogram
960 int bin = hist[0];
961 for(int i = 1; i < 256; i++)
962 {
963     if(hist[i] > bin)
964     {
965         bin = hist[i];
966     }
967 }
968
969 // normalize the histogram between 0 and histImage.rows
970 for(int i = 0; i < 256; i++)
971 {
972     hist[i] = (float) hist[i] / bin;
973 }
974
975 // Calculando los histogramas
976 int hist_w = 128; int hist_h = 128;
977 int bin_w = cvRound( (double) hist_w/256);
978 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
979
980 // Find the maxium intensity element from Histogram
981 int bin = hist[0];
982 for(int i = 1; i < 256; i++)
983 {
984     if(hist[i] > bin)
985     {
986         bin = hist[i];
987     }
988 }
989
990 // normalize the histogram between 0 and histImage.rows
991 for(int i = 0; i < 256; i++)
992 {
993     hist[i] = (float) hist[i] / bin;
994 }
995
996 // Calculando los histogramas
997 int hist_w = 128; int hist_h = 128;
998 int bin_w = cvRound( (double) hist_w/256);
999 int histImageHist_h, hist_w, cv_RNG, Scalar(255, 255, 255);
1000
1001 // Find the maxium intensity element from Histogram
1002 int bin = hist[0];
1003 for(int i = 1; i < 256; i++)
1004 {
1005     if(hist[i] > bin)
1006     {
1007         bin = hist[i];
1008     }
1009 }
1
```

- 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

