

Programación Móvil

Dr. Marco Aurelio Nuño Maganda

Universidad Politecnica de Victoria
Ingeniería en Tecnologías de la Información
Cuatrimestre Septiembre - Diciembre 2024

mnunom@upv.edu.mx

August 29, 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.

Plataforma Virtual para el Curso

- Nombre de la clase: **Programación Móvil-Septiembre - Diciembre 2024**
- Código de clase en Classroom: **xv543pb**
- Enlace Meet para sesiones no presenciales:
<https://meet.google.com/jsb-nmxy-dht>

Horario de la Clase

■ Días y horas de clase

Clave de Grupo	Lunes	Martes	Miércoles	Jueves	Viernes
ITI-271311	14:00- 14:54	8:50- 10:40	14:00-14:54	9:45 - 10:40	12:05 - 13:00
ITI-271304		11:10 -13:00	9:45-12:05	11:10 -13:00	

■ Fechas Importantes:

- Inicio de Cursos: 2/Septiembre
- Fin de Cursos: 13/Diciembre
- Dias no hábiles oficiales: 16 de septiembre (lunes), 1 de octubre (martes) y 18 de noviembre (lunes).

Plataforma Virtual para el Curso

- Nombre de la clase: **Programación Móvil- Septiembre - Diciembre 2024**
- Código de clase en Classroom: **xv543pb**
- Enlace Meet para sesiones no presenciales:
<https://meet.google.com/jsb-nmxy-dht>

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.
- Solo se puede usar **AUDIFONOS O DISPOSITIVOS MANO-LIBRES EN CLASE** previa autorización por parte del profesor. Cualquier uso no AUTORIZADO es motivo de amonestación al estudiante, y expulsión en caso de reincidir 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.
- Al NO alcanzar un 80% de asistencia, el estudiante pierde su derecho de ser EVALUADO.
- Se deja un margen del 20% de inasistencia que el estudiante puede manejar a lo que mas le convenga.
- 15 semanas, 3 sesiones por semana, 45 sesiones (8 inasistencias al cuatrimestre por motivos meramente lúdicos, personales o por tragedias [el vocho te va a dejar tirado por lo menos 4 veces al año.]).

Justificacion de Inasistencia (1)

Para justificar una inasistencia, es necesario cumplir con los siguientes pasos:

- Ingresar al apartado de Google classroom creado para dicho fin y cargar un archivo PDF para cada día (o lapso) de inasistencias a justificar.
- Formato del nombre del PDF (todo en minúsculas):

iti-<clavegrupo>-<apat>-<amat>-<nombre>-<dd1>-<mm1>-<yyyy1>-<dd2>-<mm2>-<yyyy2>.pdf

- Donde (Si pone otros datos, dicha justificación queda anulada):
 - <clavegrupo> es la clave que viene en su horario.
 - <apat>-<amat>-<nombre> son sus nombres. Debe utilizar guiones bajo para sustituir los espacios en su datos (ver ejemplo).
 - <dd1>-<mm1>-<yyyy1> es la fecha de inicio de inasistencia
 - <dd2>-<mm2>-<yyyy2> es la fecha de fin de inasistencia (si es faltó solo día, use la misma 02-09-2024-02-09-2024)

■ Ejemplos

iti-552244-nuño-maganda-marco_aurelio-02-09-2024-02-09-2024.pdf

iti-001233-del_sagrado_corazon-hernandez-michael_jackson-02-09-2024-02-10-2024.pdf

Justificación de Inasistencia (2)

- El interior de ese PDF debe una digitalización del documento que justifique su inasistencia.
- Solo se reciben inasistencias por motivos médicos (receta legible con su Nombre) y de trabajo (Citas al SAT, Pasaporte, VISA, entrevista de trabajo).
- Debe venir resaltado el nombre y el motivo de inasistencia.
- No tendran validez impresiones de pantalla, correos electronicos de sus tutores, cartas manuscritas de algun padre o tutor, etc.
- Los motivos meramente personales quedan cubiertos por el 20% de faltas que se conviene para el estudiante.

Alumnos con Empleo (1)

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.

- 1 Introducción al cómputo móvil
 - 1 Fundamentos de Programación móvil
 - 2 Tipos de datos y expresiones
 - 3 Entornos de desarrollo de aplicaciones móviles
 - 4 Estructura de proyectos móviles
- 2 Diseño de Aplicaciones Móviles
 - 1 Interfaz de usuario
 - 2 Desarrollo de Aplicaciones Móviles
 - 3 Servicios y Notificaciones en Aplicaciones Móviles
- 3 Empleo de sensores en dispositivos móviles
 - 1 Gestión de Sensores
 - 2 Tópicos selectos de programación móvil

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

Evaluación (2)

Para cada unidad, habrá sesiones de “teoría”, sesiones de seguimiento de proyectos y sesiones de esparcimiento

- En las sesiones de teoría, el profesor presentará uno o varios temas
- En las sesiones de seguimiento de proyectos, de manera aleatoria se nombrará 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 realiza alguna mecánica para exentar un proyecto (Individual/Equipo/Asignación especial) y uno o varios estudiantes completan lo solicitado, existen dos posibilidades:
 - El estudiante acepta exentar 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 exentado. 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)

- Sustituir **iti-0000** por la clave de grupo (ver su horario)
- Crear un archivo ZIP con el siguiente formato de nombre:
 - **iti-000000_uX_nuno_maganda_marco_aurelio**
- Dentro, debe contener lo siguiente:
 - **iti-000000_uX_nuno_maganda_marco_aurelio_source** (Carpeta con código fuente de la aplicación)
 - **iti-000000_uX_nuno_maganda_marco_aurelio_latex** (Carpeta con código fuente del informe)
 - **iti-000000_uX_nuno_maganda_marco_aurelio.apk** (Instalable (solo si se trata de una aplicación móvil))
 - **iti-000000_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**
- NO DEBE HABER otros archivos .ZIP dentro del ZIP PRINCIPAL

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

- Sustituir **iti-0000** por la clave de grupo (ver su horario)
- Crear un archivo ZIP con el siguiente formato de nombre:
 - **iti-000000_eq_NN_uX**
- Dentro, debe contener lo siguiente:
 - **iti-000000_eq_NN_uX_source** (Carpeta con código fuente de la aplicación)
 - **iti-000000_eq_NN_uX_latex** (Carpeta con código fuente del informe)
 - **iti-000000_eq_NN_uX.apk** (Instalable - Solo aplicaciones móviles)
 - **iti-000000_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.
- NO DEBE HABER otros archivos .ZIP dentro del ZIP PRINCIPAL

Entregables de asignaciones especiales

- Sustituir **iti-0000** por la clave de grupo (ver su horario)
- Crear un archivo ZIP con el siguiente formato de nombre:
 - **iti-000000_aeX_uY_nuno_maganda_marco_aurelio**
- Dentro, debe contener lo siguiente:
 - **iti-000000_aeX_uY_nuno_maganda_marco_aurelio_source** (Carpeta con código fuente de la aplicación - Cuando aplique)
 - **iti-000000_aeX_uY_nuno_maganda_marco_aurelio_latex** (Carpeta con código fuente del informe o diapositivas)
 - **iti-000000_aeX_uY_nuno_maganda_marco_aurelio.apk** (Instalable - Solo aplicaciones móviles, Cuando aplique)
 - **iti-000000_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**
- NO DEBE HABER otros archivos .ZIP dentro del ZIP PRINCIPAL

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

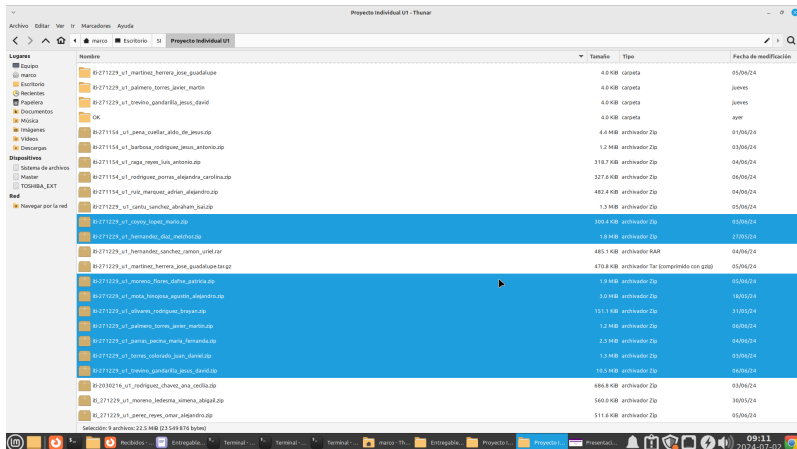
- Proyecto que no este entregado de acuerdo con las especificaciones, será penalizado. Dos escenarios posibles:
 - El proyecto puede revisarse (completo o con faltas al formato).
 - El proyecto NO puede revisarse (falta codigo fuente, informe, APK, no se compila por alguna falla, etc). En automático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

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

Detalle	Puntos de penalización sobre calificación final
Nombre Archivo	8
Tipo de Archivo	8
Estructura de Directorios	8
Falta o Error en Script	8
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

Respecto a las Asignaciones Especiales

Importante

Si en algún momento del curso, la asignación especial consiste en desarrollar una presentación de un tema explicado en un BLOG, capítulo de libro, tutorial, etc., el LENGUAJE en el que deben hacerse las DIAPOSITIVAS es el MISMO que en el que está explicado dicho BLOG (a menos que se establezca un lenguaje diferente de manera explícita en el momento de asignar dicha tarea).

Nombre de Aplicación (Al crear su proyecto en Android Studio)

- Para los proyectos individuales (en MAYUSCULAS):
 - Nombre de la Aplicacion: Z_CLAVEGRUPO_UX_APAT_AMAT_NOMBRE
 - X es Numero de unidad en un digito
 - APAT, AMAT, NOMBRE - Datos del estudiante (Guiones bajo sustituyendo los espacios en blanco)
 - Ejemplos:
 - Z_ITI-271234_U1_NUNO_MAGANDA_MARCO_AURELIO
 - Z_ITI-284433_U3_GUZMAN_LOERA_JOAQUIN
- X = Numero de unidad en un digito
- Z_U es Z_U (literal). La razon es para que al instalar la aplicacion en el dispositivo de prueba, quede al final

Nombre de Aplicación (Al crear su proyecto en Android Studio)

- Para lo proyectos en equipo (en MAYUSCULAS):
 - Nombre de la Aplicacion: Z_CLAVEGRUPO_UX_E<NUMEROEQUIPO<
 - CLAVEGRUPO - ITI-RRRRRRR (COMPLETA Y EN MAYUSCULAS)
 - NUMERO DE EQUIPO a DOS DIGITOS: 01, 02, 03
 - Ejemplos:
 - Z_ITI-271234_U1.E01
 - Z_ITI-284433_U3.E04
- X = Numero de unidad en un digito
- Z_U es Z_U (literal). La razon es para que al instalar la aplicacion en el dispositivo de prueba, quede al final

Nombre de Aplicación (Al crear su proyecto en Android Studio)

- Para las aplicaciones en Asignacion Especial (En caso de que se les solicite entregar el APK y el codigo fuente) (EN MAYUSCULAS):
 - Nombre de la Aplicacion: Z_AEX_APAT_AMAT_NOMBRE
 - X Numero de unidad en un digito (1, 2, 3, etC)
 - APAT, AMAT, NOMBRE = Datos del estudiante
 - Z es Z, es para que al instalar la aplicacion quede al final del listado
 - Si llegara a ser en equipo, utilizar la nomenclatura de un proyecto en equipo dejando el inicio de la APP sin cambios (Z_AEX)

Package de su Aplicación

- Para proyectos individuales:
upvictoria_sep_dic_2024.iti_271086.pi1u1.nuno_maganda
- Sustituir iti_271086 por su GRUPO
- Sustituir nuno_maganda por sus apellidos paterno y materno respectivamente
- Ajustar pi1uX para proyectos proyecto individuales (proyecto individual 1 unidad X=1,2,3, etc)
- Si el proyecto es individual, debe llevar los apellidos paterno y materno
- Para proyectos en equipo:
upvictoria.pm_sep_dic_2023.iti_271086.pg1uX_eqYY
- Sustituir iti_271086 por su GRUPO
- Ajustar pg1uX para proyectos en equipo (proyecto grupal 1 unidad X = 1,2,3, etc)
-
- donde YY es el numero del equipo a dos digitos
- Para asignaciones especiales:
upvictoria.pm_sep_dic_2023.iti_271086.ae1uX.nuno_maganda

Fechas importantes de entrega de proyectos (1)

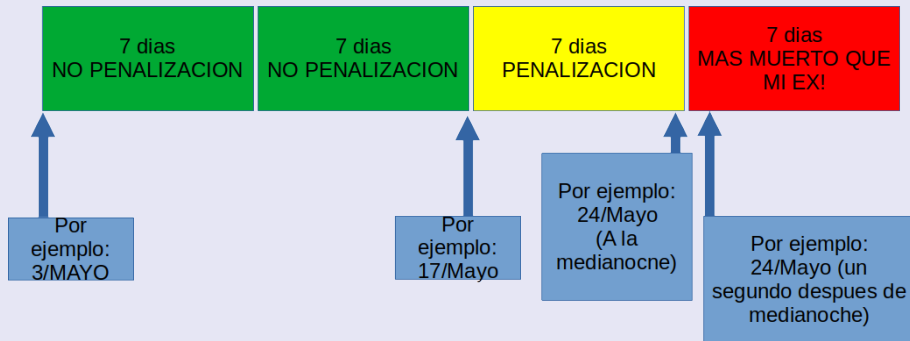
- Fecha de asignación: fecha en que se da a conocer al grupo el trabajo a elaborar
- Fecha de entrega sin penalización: 14 días naturales después de la fecha de asignación
- Proyecto entregado después de la fecha de penalización se le aplica una penalización de 25 PUNTOS
- Fecha de cierre: 21 días naturales 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.

¿Es posible obtener una calificación negativa?

SI

Calificación asignada después de revisión	70
(-) Debería entregarse 17/Mayo, lo entregó 24/Mayo (1 minuto antes de la medianoche)	25
(-) Nombre Archivo MAL	8
(-) Tipo de Archivo MAL	8
(-) Estructura de Directorios INCORRECTA	8
(-) Faltas o Errores en Script	8
(-) ZIPs dentro del ZIP	8
(-) Incluir ejecutables (aplica solo cuando el lenguaje es C++)	8
(-) Penalización por Fragmentación de Equipo	25
Calificación Final	-28

LINUX

Recomendaciones

- No es obligatorio instalarlo, pero es recomendable por cuestión de desempeño.
- Si no quieren formatear computadora, se recomienda utilizar un HD booteable (SSD con persistencia) y bootear desde su laptop o computadora.
- Si lo instalan de manera nativa, puede ser cualquier distribución (**Mint, Ubuntu, Lubuntu, Xubuntu, Debian**).

Dispositivo Físico con Android

- Teléfono Inteligente/Tablet con Android Instalado (No afecta si no es la última versión)

Software Utilizado

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

- Android Studio
- Scrcpy (<https://github.com/Genymobile/scrcpy>)
- Navegador Chrome/Firefox actualizado
- LaTeX para edición de reportes

Lenguajes Utilizados

Los lenguajes soportados por Android Studio son:

- Kotlin
- Java

Se hará énfasis en el lenguaje Kotlin, dado que es el lenguaje recomendado para nuevas aplicaciones, sin embargo, podría trabajarse también en Java dependiendo del proyecto

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

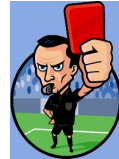


```

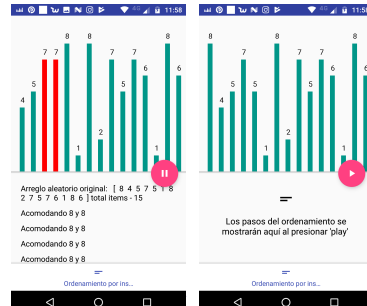
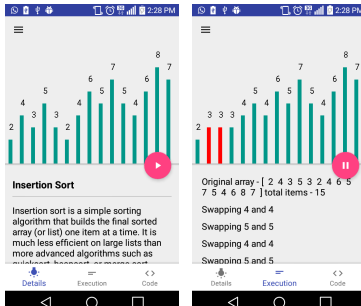
1 //Histogram equalization using C++ Image Processing
2 #include <iostream>
3 #include <opencv2/opencv.hpp>
4 #include <opencv2/imgproc/imgproc.hpp>
5 using namespace cv;
6 using namespace std;
7 using namespace cv;
8 using namespace cv;
9
10 //Funcion que inicializa todos los valores a 0
11 void initializeImage (int histogram[])
12 {
13     for(int i = 0; i < 256; i++)
14     {
15         histogram[i] = 0;
16     }
17 }
18
19 //Calculando el numero de pixeles de cada valor de intensidad
20 for(int i = 0; i < image.rows; i++)
21 {
22     for(int j = 0; j < image.cols; j++)
23     {
24         histogram[histImage.at<uchar>(i,j)]++;
25     }
26 }
27
28 //Funcion acumulativa del histograma
29 void calculateCumulativeHistogram (int histogram[])
30 {
31     int cumulativeHist = 0;
32     for(int i = 0; i < 256; i++)
33     {
34         cumulativeHist = histogram[i] + cumulativeHist;
35     }
36 }
37
38 //Funcion que devuelve los histogramas
39 void normalizeHistogram (int histogram[], const char* name)
40 {
41     int hist[256];
42     for(int i = 0; i < 256; i++)
43     {
44         hist[i] = histogram[i];
45     }
46
47     //Normalizando los histogramas
48     int hist_w = 255; int hist_h = 400;
49     int bin_w = cvRound((double) hist_w/256);
50     Mat histImage(hist_h, hist_w, CV_8UC1, Scalar(0,0,0));
51
52     //Find the maximum intensity element from histogram
53     int max = 0;
54     for(int i = 0; i < 256; i++)
55     {
56         if(hist[i] > max)
57             max = hist[i];
58     }
59
60     //Normalizing the histogram between 0 and histImage.rows
61     for(int i = 0; i < 256; i++)
62     {
63         hist[i] = (int)((double)hist[i]*histImage.rows/max);
64     }
65 }
66
67 //Histogram equalization using C++ Image Processing
68 #include <iostream>
69 #include <opencv2/opencv.hpp>
70 #include <opencv2/imgproc/imgproc.hpp>
71 using namespace cv;
72 using namespace std;
73 using namespace cv;
74 using namespace cv;
75
76 void initializeImage (int histogram[])
77 {
78     // initialization all intensity values to 0
79     for(int i = 0; i < 256; i++)
80     {
81         histogram[i] = 0;
82     }
83 }
84
85 // calculate the use of pixels for each intensity values
86 for(int i = 0; i < image.rows; i++)
87 {
88     for(int j = 0; j < image.cols; j++)
89     {
90         histogram[histImage.at<uchar>(i,j)]++;
91     }
92 }
93
94 void calculateCumulativeHistogram (int histogram[])
95 {
96     cumulativeHist = 0;
97     for(int i = 0; i < 256; i++)
98     {
99         cumulativeHist = histogram[i] + cumulativeHist;
100     }
101 }
102
103 void histogramEqualization (int histogram[], const char* name)
104 {
105     int hist[256];
106     for(int i = 0; i < 256; i++)
107     {
108         hist[i] = histogram[i];
109     }
110
111     // Find the maximum intensity element from histogram
112     int max = 0;
113     for(int i = 0; i < 256; i++)
114     {
115         if(hist[i] > max)
116             max = hist[i];
117     }
118
119     // Normalizing the histogram between 0 and histImage.rows
120     for(int i = 0; i < 256; i++)
121     {
122         hist[i] = (int)((double)hist[i]*histImage.rows/max);
123     }
124 }
125
126 //Histogram equalization using C++ Image Processing
127 #include <iostream>
128 #include <opencv2/opencv.hpp>
129 #include <opencv2/imgproc/imgproc.hpp>
130 using namespace cv;
131 using namespace std;
132 using namespace cv;
133 using namespace cv;
134
135 void initializeImage (int histogram[])
136 {
137     // initialization all intensity values to 0
138     for(int i = 0; i < 256; i++)
139     {
140         histogram[i] = 0;
141     }
142 }
143
144 // calculate the use of pixels for each intensity values
145 for(int i = 0; i < image.rows; i++)
146 {
147     for(int j = 0; j < image.cols; j++)
148     {
149         histogram[histImage.at<uchar>(i,j)]++;
150     }
151 }
152
153 void calculateCumulativeHistogram (int histogram[])
154 {
155     cumulativeHist = 0;
156     for(int i = 0; i < 256; i++)
157     {
158         cumulativeHist = histogram[i] + cumulativeHist;
159     }
160 }
161
162 void histogramEqualization (int histogram[], const char* name)
163 {
164     int hist[256];
165     for(int i = 0; i < 256; i++)
166     {
167         hist[i] = histogram[i];
168     }
169
170     // Find the maximum intensity element from histogram
171     int max = 0;
172     for(int i = 0; i < 256; i++)
173     {
174         if(hist[i] > max)
175             max = hist[i];
176     }
177
178     // Normalizing the histogram between 0 and histImage.rows
179     for(int i = 0; i < 256; i++)
180     {
181         hist[i] = (int)((double)hist[i]*histImage.rows/max);
182     }
183 }
184
185 //Histogram equalization using C++ Image Processing
186 #include <iostream>
187 #include <opencv2/opencv.hpp>
188 #include <opencv2/imgproc/imgproc.hpp>
189 using namespace cv;
190 using namespace std;
191 using namespace cv;
192 using namespace cv;
193
194 void initializeImage (int histogram[])
195 {
196     // initialization all intensity values to 0
197     for(int i = 0; i < 256; i++)
198     {
199         histogram[i] = 0;
200     }
201 }
202
203 // calculate the use of pixels for each intensity values
204 for(int i = 0; i < image.rows; i++)
205 {
206     for(int j = 0; j < image.cols; j++)
207     {
208         histogram[histImage.at<uchar>(i,j)]++;
209     }
210 }
211
212 void calculateCumulativeHistogram (int histogram[])
213 {
214     cumulativeHist = 0;
215     for(int i = 0; i < 256; i++)
216     {
217         cumulativeHist = histogram[i] + cumulativeHist;
218     }
219 }
220
221 void histogramEqualization (int histogram[], const char* name)
222 {
223     int hist[256];
224     for(int i = 0; i < 256; i++)
225     {
226         hist[i] = histogram[i];
227     }
228
229     // Find the maximum intensity element from histogram
230     int max = 0;
231     for(int i = 0; i < 256; i++)
232     {
233         if(hist[i] > max)
234             max = hist[i];
235     }
236
237     // Normalizing the histogram between 0 and histImage.rows
238     for(int i = 0; i < 256; i++)
239     {
240         hist[i] = (int)((double)hist[i]*histImage.rows/max);
241     }
242 }
243
244 //Histogram equalization using C++ Image Processing
245 #include <iostream>
246 #include <opencv2/opencv.hpp>
247 #include <opencv2/imgproc/imgproc.hpp>
248 using namespace cv;
249 using namespace std;
250 using namespace cv;
251 using namespace cv;
252
253 void initializeImage (int histogram[])
254 {
255     // initialization all intensity values to 0
256     for(int i = 0; i < 256; i++)
257     {
258         histogram[i] = 0;
259     }
260 }
261
262 // calculate the use of pixels for each intensity values
263 for(int i = 0; i < image.rows; i++)
264 {
265     for(int j = 0; j < image.cols; j++)
266     {
267         histogram[histImage.at<uchar>(i,j)]++;
268     }
269 }
270
271 void calculateCumulativeHistogram (int histogram[])
272 {
273     cumulativeHist = 0;
274     for(int i = 0; i < 256; i++)
275     {
276         cumulativeHist = histogram[i] + cumulativeHist;
277     }
278 }
279
280 void histogramEqualization (int histogram[], const char* name)
281 {
282     int hist[256];
283     for(int i = 0; i < 256; i++)
284     {
285         hist[i] = histogram[i];
286     }
287
288     // Find the maximum intensity element from histogram
289     int max = 0;
290     for(int i = 0; i < 256; i++)
291     {
292         if(hist[i] > max)
293             max = hist[i];
294     }
295
296     // Normalizing the histogram between 0 and histImage.rows
297     for(int i = 0; i < 256; i++)
298     {
299         hist[i] = (int)((double)hist[i]*histImage.rows/max);
300     }
301 }
302
303 //Histogram equalization using C++ Image Processing
304 #include <iostream>
305 #include <opencv2/opencv.hpp>
306 #include <opencv2/imgproc/imgproc.hpp>
307 using namespace cv;
308 using namespace std;
309 using namespace cv;
310 using namespace cv;
311
312 void initializeImage (int histogram[])
313 {
314     // initialization all intensity values to 0
315     for(int i = 0; i < 256; i++)
316     {
317         histogram[i] = 0;
318     }
319 }
320
321 // calculate the use of pixels for each intensity values
322 for(int i = 0; i < image.rows; i++)
323 {
324     for(int j = 0; j < image.cols; j++)
325     {
326         histogram[histImage.at<uchar>(i,j)]++;
327     }
328 }
329
330 void calculateCumulativeHistogram (int histogram[])
331 {
332     cumulativeHist = 0;
333     for(int i = 0; i < 256; i++)
334     {
335         cumulativeHist = histogram[i] + cumulativeHist;
336     }
337 }
338
339 void histogramEqualization (int histogram[], const char* name)
340 {
341     int hist[256];
342     for(int i = 0; i < 256; i++)
343     {
344         hist[i] = histogram[i];
345     }
346
347     // Find the maximum intensity element from histogram
348     int max = 0;
349     for(int i = 0; i < 256; i++)
350     {
351         if(hist[i] > max)
352             max = hist[i];
353     }
354
355     // Normalizing the histogram between 0 and histImage.rows
356     for(int i = 0; i < 256; i++)
357     {
358         hist[i] = (int)((double)hist[i]*histImage.rows/max);
359     }
360 }
361
362 //Histogram equalization using C++ Image Processing
363 #include <iostream>
364 #include <opencv2/opencv.hpp>
365 #include <opencv2/imgproc/imgproc.hpp>
366 using namespace cv;
367 using namespace std;
368 using namespace cv;
369 using namespace cv;
370
371 void initializeImage (int histogram[])
372 {
373     // initialization all intensity values to 0
374     for(int i = 0; i < 256; i++)
375     {
376         histogram[i] = 0;
377     }
378 }
379
380 // calculate the use of pixels for each intensity values
381 for(int i = 0; i < image.rows; i++)
382 {
383     for(int j = 0; j < image.cols; j++)
384     {
385         histogram[histImage.at<uchar>(i,j)]++;
386     }
387 }
388
389 void calculateCumulativeHistogram (int histogram[])
390 {
391     cumulativeHist = 0;
392     for(int i = 0; i < 256; i++)
393     {
394         cumulativeHist = histogram[i] + cumulativeHist;
395     }
396 }
397
398 void histogramEqualization (int histogram[], const char* name)
399 {
400     int hist[256];
401     for(int i = 0; i < 256; i++)
402     {
403         hist[i] = histogram[i];
404     }
405
406     // Find the maximum intensity element from histogram
407     int max = 0;
408     for(int i = 0; i < 256; i++)
409     {
410         if(hist[i] > max)
411             max = hist[i];
412     }
413
414     // Normalizing the histogram between 0 and histImage.rows
415     for(int i = 0; i < 256; i++)
416     {
417         hist[i] = (int)((double)hist[i]*histImage.rows/max);
418     }
419 }
420
421 //Histogram equalization using C++ Image Processing
422 #include <iostream>
423 #include <opencv2/opencv.hpp>
424 #include <opencv2/imgproc/imgproc.hpp>
425 using namespace cv;
426 using namespace std;
427 using namespace cv;
428 using namespace cv;
429
430 void initializeImage (int histogram[])
431 {
432     // initialization all intensity values to 0
433     for(int i = 0; i < 256; i++)
434     {
435         histogram[i] = 0;
436     }
437 }
438
439 // calculate the use of pixels for each intensity values
440 for(int i = 0; i < image.rows; i++)
441 {
442     for(int j = 0; j < image.cols; j++)
443     {
444         histogram[histImage.at<uchar>(i,j)]++;
445     }
446 }
447
448 void calculateCumulativeHistogram (int histogram[])
449 {
450     cumulativeHist = 0;
451     for(int i = 0; i < 256; i++)
452     {
453         cumulativeHist = histogram[i] + cumulativeHist;
454     }
455 }
456
457 void histogramEqualization (int histogram[], const char* name)
458 {
459     int hist[256];
460     for(int i = 0; i < 256; i++)
461     {
462         hist[i] = histogram[i];
463     }
464
465     // Find the maximum intensity element from histogram
466     int max = 0;
467     for(int i = 0; i < 256; i++)
468     {
469         if(hist[i] > max)
470             max = hist[i];
471     }
472
473     // Normalizing the histogram between 0 and histImage.rows
474     for(int i = 0; i < 256; i++)
475     {
476         hist[i] = (int)((double)hist[i]*histImage.rows/max);
477     }
478 }
479
480 //Histogram equalization using C++ Image Processing
481 #include <iostream>
482 #include <opencv2/opencv.hpp>
483 #include <opencv2/imgproc/imgproc.hpp>
484 using namespace cv;
485 using namespace std;
486 using namespace cv;
487 using namespace cv;
488
489 void initializeImage (int histogram[])
490 {
491     // initialization all intensity values to 0
492     for(int i = 0; i < 256; i++)
493     {
494         histogram[i] = 0;
495     }
496 }
497
498 // calculate the use of pixels for each intensity values
499 for(int i = 0; i < image.rows; i++)
500 {
501     for(int j = 0; j < image.cols; j++)
502     {
503         histogram[histImage.at<uchar>(i,j)]++;
504     }
505 }
506
507 void calculateCumulativeHistogram (int histogram[])
508 {
509     cumulativeHist = 0;
510     for(int i = 0; i < 256; i++)
511     {
512         cumulativeHist = histogram[i] + cumulativeHist;
513     }
514 }
515
516 void histogramEqualization (int histogram[], const char* name)
517 {
518     int hist[256];
519     for(int i = 0; i < 256; i++)
520     {
521         hist[i] = histogram[i];
522     }
523
524     // Find the maximum intensity element from histogram
525     int max = 0;
526     for(int i = 0; i < 256; i++)
527     {
528         if(hist[i] > max)
529             max = hist[i];
530     }
531
532     // Normalizing the histogram between 0 and histImage.rows
533     for(int i = 0; i < 256; i++)
534     {
535         hist[i] = (int)((double)hist[i]*histImage.rows/max);
536     }
537 }
538
539 //Histogram equalization using C++ Image Processing
540 #include <iostream>
541 #include <opencv2/opencv.hpp>
542 #include <opencv2/imgproc/imgproc.hpp>
543 using namespace cv;
544 using namespace std;
545 using namespace cv;
546 using namespace cv;
547
548 void initializeImage (int histogram[])
549 {
550     // initialization all intensity values to 0
551     for(int i = 0; i < 256; i++)
552     {
553         histogram[i] = 0;
554     }
555 }
556
557 // calculate the use of pixels for each intensity values
558 for(int i = 0; i < image.rows; i++)
559 {
560     for(int j = 0; j < image.cols; j++)
561     {
562         histogram[histImage.at<uchar>(i,j)]++;
563     }
564 }
565
566 void calculateCumulativeHistogram (int histogram[])
567 {
568     cumulativeHist = 0;
569     for(int i = 0; i < 256; i++)
570     {
571         cumulativeHist = histogram[i] + cumulativeHist;
572     }
573 }
574
575 void histogramEqualization (int histogram[], const char* name)
576 {
577     int hist[256];
578     for(int i = 0; i < 256; i++)
579     {
580         hist[i] = histogram[i];
581     }
582
583     // Find the maximum intensity element from histogram
584     int max = 0;
585     for(int i = 0; i < 256; i++)
586     {
587         if(hist[i] > max)
588             max = hist[i];
589     }
590
591     // Normalizing the histogram between 0 and histImage.rows
592     for(int i = 0; i < 256; i++)
593     {
594         hist[i] = (int)((double)hist[i]*histImage.rows/max);
595     }
596 }
597
598 //Histogram equalization using C++ Image Processing
599 #include <iostream>
600 #include <opencv2/opencv.hpp>
601 #include <opencv2/imgproc/imgproc.hpp>
602 using namespace cv;
603 using namespace std;
604 using namespace cv;
605 using namespace cv;
606
607 void initializeImage (int histogram[])
608 {
609     // initialization all intensity values to 0
610     for(int i = 0; i < 256; i++)
611     {
612         histogram[i] = 0;
613     }
614 }
615
616 // calculate the use of pixels for each intensity values
617 for(int i = 0; i < image.rows; i++)
618 {
619     for(int j = 0; j < image.cols; j++)
620     {
621         histogram[histImage.at<uchar>(i,j)]++;
622     }
623 }
624
625 void calculateCumulativeHistogram (int histogram[])
626 {
627     cumulativeHist = 0;
628     for(int i = 0; i < 256; i++)
629     {
630         cumulativeHist = histogram[i] + cumulativeHist;
631     }
632 }
633
634 void histogramEqualization (int histogram[], const char* name)
635 {
636     int hist[256];
637     for(int i = 0; i < 256; i++)
638     {
639         hist[i] = histogram[i];
640     }
641
642     // Find the maximum intensity element from histogram
643     int max = 0;
644     for(int i = 0; i < 256; i++)
645     {
646         if(hist[i] > max)
647             max = hist[i];
648     }
649
650     // Normalizing the histogram between 0 and histImage.rows
651     for(int i = 0; i < 256; i++)
652     {
653         hist[i] = (int)((double)hist[i]*histImage.rows/max);
654     }
655 }
656
657 //Histogram equalization using C++ Image Processing
658 #include <iostream>
659 #include <opencv2/opencv.hpp>
660 #include <opencv2/imgproc/imgproc.hpp>
661 using namespace cv;
662 using namespace std;
663 using namespace cv;
664 using namespace cv;
665
666 void initializeImage (int histogram[])
667 {
668     // initialization all intensity values to 0
669     for(int i = 0; i < 256; i++)
670     {
671         histogram[i] = 0;
672     }
673 }
674
675 // calculate the use of pixels for each intensity values
676 for(int i = 0; i < image.rows; i++)
677 {
678     for(int j = 0; j < image.cols; j++)
679     {
680         histogram[histImage.at<uchar>(i,j)]++;
681     }
682 }
683
684 void calculateCumulativeHistogram (int histogram[])
685 {
686     cumulativeHist = 0;
687     for(int i = 0; i < 256; i++)
688     {
689         cumulativeHist = histogram[i] + cumulativeHist;
690     }
691 }
692
693 void histogramEqualization (int histogram[], const char* name)
694 {
695     int hist[256];
696     for(int i = 0; i < 256; i++)
697     {
698         hist[i] = histogram[i];
699     }
700
701     // Find the maximum intensity element from histogram
702     int max = 0;
703     for(int i = 0; i < 256; i++)
704     {
705         if(hist[i] > max)
706             max = hist[i];
707     }
708
709     // Normalizing the histogram between 0 and histImage.rows
710     for(int i = 0; i < 256; i++)
711     {
712         hist[i] = (int)((double)hist[i]*histImage.rows/max);
713     }
714 }
715
716 //Histogram equalization using C++ Image Processing
717 #include <iostream>
718 #include <opencv2/opencv.hpp>
719 #include <opencv2/imgproc/imgproc.hpp>
720 using namespace cv;
721 using namespace std;
722 using namespace cv;
723 using namespace cv;
724
725 void initializeImage (int histogram[])
726 {
727     // initialization all intensity values to 0
728     for(int i = 0; i < 256; i++)
729     {
730         histogram[i] = 0;
731     }
732 }
733
734 // calculate the use of pixels for each intensity values
735 for(int i = 0; i < image.rows; i++)
736 {
737     for(int j = 0; j < image.cols; j++)
738     {
739         histogram[histImage.at<uchar>(i,j)]++;
740     }
741 }
742
743 void calculateCumulativeHistogram (int histogram[])
744 {
745     cumulativeHist = 0;
746     for(int i = 0; i < 256; i++)
747     {
748         cumulativeHist = histogram[i] + cumulativeHist;
749     }
750 }
751
752 void histogramEqualization (int histogram[], const char* name)
753 {
754     int hist[256];
755     for(int i = 0; i < 256; i++)
756     {
757         hist[i] = histogram[i];
758     }
759
760     // Find the maximum intensity element from histogram
761     int max = 0;
762     for(int i = 0; i < 256; i++)
763     {
764         if(hist[i] > max)
765             max = hist[i];
766     }
767
768     // Normalizing the histogram between 0 and histImage.rows
769     for(int i = 0; i < 256; i++)
770     {
771         hist[i] = (int)((double)hist[i]*histImage.rows/max);
772     }
773 }
774
775 //Histogram equalization using C++ Image Processing
776 #include <iostream>
777 #include <opencv2/opencv.hpp>
778 #include <opencv2/imgproc/imgproc.hpp>
779 using namespace cv;
780 using namespace std;
781 using namespace cv;
782 using namespace cv;
783
784 void initializeImage (int histogram[])
785 {
786     // initialization all intensity values to 0
787     for(int i = 0; i < 256; i++)
788     {
789         histogram[i] = 0;
790     }
791 }
792
793 // calculate the use of pixels for each intensity values
794 for(int i = 0; i < image.rows; i++)
795 {
796     for(int j = 0; j < image.cols; j++)
797     {
798         histogram[histImage.at<uchar>(i,j)]++;
799     }
800 }
801
802 void calculateCumulativeHistogram (int histogram[])
803 {
804     cumulativeHist = 0;
805     for(int i = 0; i < 256; i++)
806     {
807         cumulativeHist = histogram[i] + cumulativeHist;
808     }
809 }
810
811 void histogramEqualization (int histogram[], const char* name)
812 {
813     int hist[256];
814     for(int i = 0; i < 256; i++)
815     {
816         hist[i] = histogram[i];
817     }
818
819     // Find the maximum intensity element from histogram
820     int max = 0;
821     for(int i = 0; i < 256; i++)
822     {
823         if(hist[i] > max)
824             max = hist[i];
825     }
826
827     // Normalizing the histogram between 0 and histImage.rows
828     for(int i = 0; i < 256; i++)
829     {
830         hist[i] = (int)((double)hist[i]*histImage.rows/max);
831     }
832 }
833
834 //Histogram equalization using C++ Image Processing
835 #include <iostream>
836 #include <opencv2/opencv.hpp>
837 #include <opencv2/imgproc/imgproc.hpp>
838 using namespace cv;
839 using namespace std;
840 using namespace cv;
841 using namespace cv;
842
843 void initializeImage (int histogram[])
844 {
845     // initialization all intensity values to 0
846     for(int i = 0; i < 256; i++)
847     {
848         histogram[i] = 0;
849     }
850 }
851
852 // calculate the use of pixels for each intensity values
853 for(int i = 0; i < image.rows; i++)
854 {
855     for(int j = 0; j < image.cols; j++)
856     {
857         histogram[histImage.at<uchar>(i,j)]++;
858     }
859 }
860
861 void calculateCumulativeHistogram (int histogram[])
862 {
863     cumulativeHist = 0;
864     for(int i = 0; i < 256; i++)
865     {
866         cumulativeHist = histogram[i] + cumulativeHist;
867     }
868 }
869
870 void histogramEqualization (int histogram[], const char* name)
871 {
872     int hist[256];
873     for(int i = 0; i < 256; i++)
874     {
875         hist[i] = histogram[i];
876     }
877
878     // Find the maximum intensity element from histogram
879     int max = 0;
880     for(int i = 0; i < 256; i++)
881     {
882         if(hist[i] > max)
883             max = hist[i];
884     }
885
886     // Normalizing the histogram between 0 and histImage.rows
887     for(int i = 0; i < 256; i++)
888     {
889         hist[i] = (int)((double)hist[i]*histImage.rows/max);
890     }
891 }
892
893 //Histogram equalization using C++ Image Processing
894 #include <iostream>
895 #include <opencv2/opencv.hpp>
896 #include <opencv2/imgproc/imgproc.hpp>
897 using namespace cv;
898 using namespace std;
899 using namespace cv;
900 using namespace cv;
901
902 void initializeImage (int histogram[])
903 {
904     // initialization all intensity values to 0
905     for(int i = 0; i < 256; i++)
906     {
907         histogram[i] = 0;
908     }
909 }
910
911 // calculate the use of pixels for each intensity values
912 for(int i = 0; i < image.rows; i++)
913 {
914     for(int j = 0; j < image.cols; j++)
915     {
916         histogram[histImage.at<uchar>(i,j)]++;
917     }
918 }
919
920 void calculateCumulativeHistogram (int histogram[])
921 {
922     cumulativeHist = 0;
923     for(int i = 0; i < 256; i++)
924     {
925         cumulativeHist = histogram[i] + cumulativeHist;
926     }
927 }
928
929 void histogramEqualization (int histogram[], const char* name)
930 {
931     int hist[256];
932     for(int i = 0; i < 256; i++)
933     {
934         hist[i] = histogram[i];
935     }
936
937     // Find the maximum intensity element from histogram
938     int max = 0;
939     for(int i = 0; i < 256; i++)
940     {
941         if(hist[i] > max)
942             max = hist[i];
943     }
944
945     // Normalizing the histogram between 0 and histImage.rows
946     for(int i = 0; i < 256; i++)
947     {
948         hist[i] = (int)((double)hist[i]*histImage.rows/max);
949     }
950 }
951
952 //Histogram equalization using C++ Image Processing
953 #include <iostream>
954 #include <opencv2/opencv.hpp>
955 #include <opencv2/imgproc/imgproc.hpp>
956 using namespace cv;
957 using namespace std;
958 using namespace cv;
959 using namespace cv;
960
961 void initializeImage (int histogram[])
962 {
963     // initialization all intensity values to 0
964     for(int i = 0; i < 256; i++)
965     {
966         histogram[i] = 0;
967     }
968 }
969
970 // calculate the use of pixels for each intensity values
971 for(int i = 0; i < image.rows; i++)
972 {
973     for(int j = 0; j < image.cols; j++)
974     {
975         histogram[histImage.at<uchar>(i,j)]++;
976     }
977 }
978
979 void calculateCumulativeHistogram (int histogram[])
980 {
981     cumulativeHist = 0;
982     for(int i = 0; i < 256; i++)
983     {
984         cumulativeHist = histogram[i] + cumulativeHist;
985     }
986 }
987
988 void histogramEqualization (int histogram[], const char* name)
989 {
990     int hist[256];
991     for(int i = 0; i < 256; i++)
992     {
993         hist[i] = histogram[i];
994     }
995
996     // Find the maximum intensity element from histogram
997     int max = 0;
998     for(int i = 0; i < 256; i++)
999     {
1000         if(hist[i] > max)
1001             max = hist[i];
1002     }
1003
1004     // Normalizing the histogram between 0 and histImage.rows
1005     for(int i = 0; i < 256; i++)
1006     {
1007         hist[i] = (int)((double)hist[i]*histImage.rows/max);
1008     }
1009 }
1010
1011 //Histogram equalization using C++ Image Processing
1012 #include <iostream>
1013 #include <opencv2/opencv.hpp>
1014 #include <opencv2/imgproc/imgproc.hpp>
1015 using namespace cv;
1016 using namespace std;
1017 using namespace cv;
1018 using namespace cv;
1019
1020 void initializeImage (int histogram[])
1021 {
1022     // initialization all intensity values to 0
1023     for(int i = 0; i < 256; i++)
1024     {
1025         histogram[i] = 0;
1026     }
1027 }
1028
1029 // calculate the use of pixels for each intensity values
1030 for(int i = 0; i < image.rows; i++)
1031 {
1032     for(int j = 0; j < image.cols; j++)
1033     {
1034         histogram[histImage.at<uchar>(i,j)]++;
1035     }
1036 }
1037
1038 void calculateCumulativeHistogram (int histogram[])
1039 {
1040     cumulativeHist = 0;
1041     for(int i = 0; i < 256; i++)
1042     {
1043         cumulativeHist = histogram[i] + cumulativeHist;
1044     }
1045 }
1046
1047 void histogramEqualization (int histogram[], const char* name)
1048 {
1049     int hist[256];
1050     for(int i = 0; i < 256; i++)
1051     {
1052         hist[i] = histogram[i];
1053     }
1054
1055     // Find the maximum intensity element from histogram
1056     int max = 0;
1057     for(int i = 0; i < 256; i++)
1058     {
1059         if(hist[i] > max)
1060             max = hist[i];
1061     }
1062
1063     // Normalizing the histogram between 0 and histImage.rows
1064     for(int i = 0; i < 256; i++)
1065     {
1066         hist[i] = (int)((double)hist[i]*histImage.rows/max);
1067     }
1068 }
1069
1070 //Histogram equalization using C++ Image Processing
1071 #include <iostream>
1072 #include <opencv2/opencv.hpp>
1073 #include <opencv2/imgproc/imgproc.hpp>
1074 using namespace cv;
1075 using namespace std;
1076 using namespace cv;
1077 using namespace cv;
1078
1079 void initializeImage (int histogram[])
1080 {
1081     // initialization all intensity values to 0
1082     for(int i = 0; i < 256; i++)
1083     {
1084         histogram[i] = 0;
1085     }
1086 }
1087
1088 // calculate the use of pixels for each intensity values
1089 for(int i = 0; i < image.rows; i++)
1090 {
1091     for(int j = 0; j < image.cols; j++)
1092     {
1093         histogram[histImage.at<uchar>(i,j)]++;
1094     }
1095 }
1096
1097 void calculateCumulativeHistogram (int histogram[])
1098 {
1099     cumulativeHist = 0;
1100     for(int i = 0; i < 256; i++)
1101     {
1102         cumulativeHist = histogram[i] + cumulativeHist;
1103     }
1104 }
1105
1106 void histogramEqualization (int histogram[], const char* name)
1107 {
1108     int hist[256];
1109     for(int i = 0; i < 256; i++)
1110     {
1111         hist[i] = histogram[i];
1112     }
1113
1114     // Find the maximum intensity element from histogram
1115     int max = 0;
1116     for(int i = 0; i < 256; i++)
1117     {
1118         if(hist[i] > max)
1119             max = hist[i];
1120     }
1121
1122     // Normalizing the histogram between 0 and histImage.rows
1123     for(int i = 0; i < 256; i++)
1124     {
1125         hist[i] = (int)((double)hist[i]*histImage.rows/max);
1126     }
1127 }
1128
1129 //Histogram equalization using C++ Image Processing
1130 #include <iostream>
1131 #include <opencv2/opencv.hpp>
1132 #include <opencv2/imgproc/imgproc.hpp>
1133 using namespace cv;
1134 using namespace std;
1135 using namespace cv;
1136 using namespace cv;
1137
1138 void initializeImage (int histogram[])
1139 {
1140     // initialization all intensity values to 0
1141     for(int i = 0; i < 256; 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>



Frase célebre

“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

