Laboratorio de Análisis y Procesamiento de Imágenes
Módulo II
Fechas: 19 de febrero - 13 de marzo de 2021

Universidad de los Andes

Proyecto 2:

Herramientas para Clasificación de Imágenes

1. Objetivos

- Utilizar e implementar herramientas teóricas para resolver un problema de clasificación.
- Familiarizarse con las librerías de Python para el procesamiento de imágenes.

2. Parámetros de entrega

Entrega por Brightspace:

- Adjuntar un único archivo con los códigos de la siguiente forma:

  main_CódigoEstudiante1_CódigoEstudiante2.py. **Es obligatoria la entrega de un archivo *.py***. **NO** se recibirán archivos con otra extensión. Su entrega conllevará una penalización sobre la calificación final del proyecto.

  - El código enviado debe tener una longitud máxima de 500 líneas. Se generará una penalización sobre la calificación final de cada entrega por cada 10 líneas adicionales. Esto con el objetivo de promover el uso de estructuras iterativas y operaciones vectoriales, que aumenten la eficiencia de sus códigos (aspecto fundamental en tareas de programación).

- Adjuntar **TODO** el código que se entregó en el ítem anterior, en formato *.txt*. Llámelo de igual manera: *CódigoEstudiante1_CódigoEstudiante2.txt*. Esto con el fin de poder evaluar en Brightspace automáticamente cualquier intento de copia o similitud entre los algoritmos. **Cualquier intento de copia/plagio será tratado de acuerdo al reglamento de la Universidad**. **Aquel grupo que no incluya este ítem en cualquiera de sus entregas tendrá una nota de 0 en la entrega que corresponda**.
- Adjuntar un único archivo con el informe en PDF con el mismo nombre del primer ítem: *CódigoEstudiante1_CódigoEstudiante2.pdf*. El informe debe presentarse con el formato CVPR (para más información del informe, ver sección de **Parámetros de calificación**).

- En el caso de que no tenga compañero, **DEBE** utilizar un cero como el código de su compañero, i.e. *main_CódigoEstudiante_0.py* para el archivo de código, *CódigoEstudiante_0.pdf* para el informe y *CódigoEstudiante_0.txt* para el archivo de texto.
- **NO** se permiten archivos comprimidos tales como *zip, rar, 7z, tar, gz, xz, iso, cbz*, etc. Aquel grupo que envíe su informe como un archivo comprimido no tendrá calificación.
- **No se recibirá ningún archivo por algún medio diferente a Brightspace**.
- Este proyecto tiene una duración de tres semanas y está compuesto por tres entregas (una por semana). Estas se realizarán para las siguientes fechas:

  - Entrega 1: Domingo 28 de febrero de 2021
  - Entrega 2: Domingo 7 de marzo de 2021
  - Entrega 3: Domingo 14 de marzo de 2021

  El plazo de entrega será hasta las 7:59 a.m. de las fechas establecidas. El vínculo para el envío de los documentos dejará de estar disponible luego de esta hora. **No se recibirán documentos o archivos después de la hora máxima de entrega** (tenga en cuenta que la hora queda registrada en el envío de Brightspace). Cada estudiante tiene varios intentos en Brightspace para cada entrega, pueden hacer entregas parciales y se tendrá en cuenta únicamente el último intento para la calificación.
- Cada entrega del presente miniproyecto recibirá una calificación por separado. Estas contarán con los siguientes pesos sobre la nota final del curso:

  - Entrega 1: 3.33 %
  - Entrega 2: 3.33 %
  - Entrega 3: 3.33 %

3. Reglas Generales

- La asistencia a la sección de laboratorio **inscrita** es **obligatoria**. De acuerdo con el Reglamento General de Estudiantes de Pregrado de la Universidad de los Andes, la inasistencia a más del 20 % de las clases de laboratorio resultará en la reprobación de la materia completa (laboratorio y magistral).
- Los informes deben realizarse **únicamente** con la pareja. Esto quiere decir que, aunque es válido discutir los problemas con sus compañeros, la solución y el código deben ser de su completa autoría. Está prohibido copiar literalmente el algoritmo y/o procedimiento desarrollado por otro grupo.

  - Si llega a obtener un código de Internet, asociado al problema a resolver, este debe estar **debidamente referenciado** y usted debe entenderlo por completo. Sin embargo, todos los ejercicios que requieren una implementación propia, deben ser de su completa autoría. En estos casos, no podrán tomar implementaciones (o partes) de un autor externo y referenciarlas.

4. Parámetros de Calificación

Resultados:

1. Todos los códigos deben mantener orden y coherencia en la ejecución de comandos, es decir, cada vez que se muestre una figura, el programa debe esperar para que se presione una tecla, para así continuar con la siguiente y así sucesivamente (para esto utilice en Python `input("Press Enter to continue...")`. Si quieren contrastar dos o más imágenes utilicen `subplot`.
2. Toda figura debe estar numerada y debe llevar su título y descripción en el informe.
3. El código debe estar debidamente comentado.
4. Nunca utilice rutas absolutas para leer o guardar archivos. Este es el error más común en la ejecución de los códigos.

    - Para generar rutas utilice `os.path.join` en Python, ya que puede que corramos los laboratorios usando Linux o Windows y los separadores de archivos cambian dependiendo del sistema operativo.
    - Asuma que dentro de la carpeta de ejecución del código se encuentran los archivos necesarios para el laboratorio.
    - **Ejemplo:** Dentro del código principal, el estudiante quiere leer la imagen `im.png` que está dentro de una carpeta de imágenes en la misma ruta que el *main*.

      Forma incorrecta:
      `skimage.io.imread('C:/Estudiante/docs/ElProyecto/ims/im.png')`
      Forma correcta:
      `skimage.io.imread(os.path.join('ims','im.png'))`

Informe:

Todos los laboratorios deben realizarse en formato LaTeX o Word, pueden obtener una plantilla del formato en el siguiente vínculo. Cabe resaltar que los informes no deben contener ninguna sección de artículo científico, esto significa que no deben incluir ninguna división como resultados, *abstract* o conclusiones. Por consiguiente, **deben responder únicamente a las preguntas del informe, de forma organizada**. También, deben incluir las imágenes requeridas de sus resultados y documentarlas debidamente. Por último, el informe tiene una longitud máxima de 4 páginas, excluyendo referencias y anexos. Se pueden incluir imágenes en la sección de Anexos pero las imágenes principales deben ser parte del informe.

Bonos:

Cada pareja ganará puntos que le suben la nota por cumplir la siguiente característica:

1. Desarrollar el informe en LaTeX. Aquellas personas que lo desarrollen en LaTeX, deben escribir al final del informe **Realizado en LaTeX**. De lo contrario no se contará el bono. Los grupos que intenten reproducir la frase en un informe realizado en Word tendrán 0 en la nota de dicho proyecto. Para poder escribir el logo utilice el comando "$\LaTeX$" en su informe de Latex.

Estos puntos se asignarán de acuerdo al criterio de las profesoras.

5. Procedimiento Entrega 1

5.1. Cross-Correlación y Aplicaciones (50 %)

Como se ha visto en la sección teórica, los filtros que se usan para suavizar o mejorar la imagen, se implementan mediante una cross-correlación de estos con la imagen. La ecuación 1 presenta la fórmula de cross-correlación discreta que será implementada en esta entrega.

$$g(x,y) = f(x,y) * w(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{1}$$

Siendo $f(x,y)$ la imagen de tamaño M, $w(x,y)$ el filtro de tamaño $m \times n$ y $g(x,y)$ la imagen resultante. Tenga en cuenta que $a = \frac{m-1}{2}$ y $b = \frac{n-1}{2}$. Una manera matricial de ver esta operación se puede observar en la Figura 1.

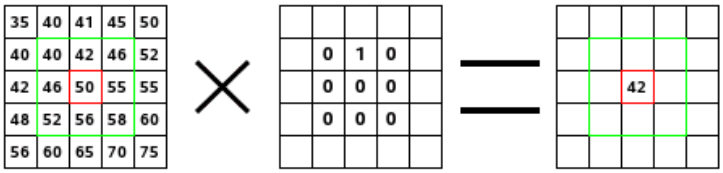


Figura 1: Representación matricial de la cross-correlación.

Para el desarrollo del punto 5.1, deberán utilizar como entrada las imágenes y filtros definidos a continuación. Descargue las imágenes originales, en formato RGB, del siguiente vínculo. Almacene las imágenes como *roses.jpg* y *noisy_roses.jpg*.

(a) (b)

Figura 2: Imágenes a utilizar. Visualización en escala de grises.

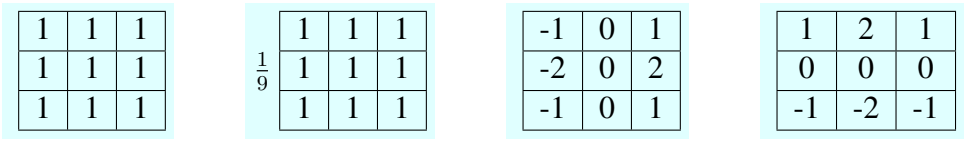


Figura 3: *Kernels* a utilizar.

Además de estos *kernels*, deberán utilizar un filtro Gaussiano cuadrado con parámetros (tamaño y σ) variables. Para generarlo, pueden hacer uso de la siguiente función:

```
import numpy as np

def gaussian_kernel(size, sigma):
    size = int(size)//2
    x, y = np.mgrid[-size:size+1, -size:size+1]
    normal = 1/(2.0 * np.pi * sigma**2)
    g = np.exp(-((x**2 + y**2)/(2.0 * sigma**2))) * normal
    return g
```

Nota: El tamaño del filtro generado por esta función es de $size \times size$. Evite confusiones al utilizar este parámetro.

5.1.1. Función MyCCorrelation

Cree una función que realice la cross-correlación de cualquier imagen en escala de grises, con cualquier kernel cuadrado de dimensiones impares. La función a adjuntar debe llamarse *MyCCorrelation_Código1_Código2*. La función también debe recibir como parámetro la condición de frontera a utilizar en la cross-correlación (*fill* ($valor = 0$)*, symm, valid*), cuyas descripciones pueden encontrar en la documentación de la función `scipy.signal.correlate2d` de Python.

Para este punto no pueden utilizar funciones nativas de Python relacionadas con el método. Únicamente pueden hacer uso de la librería NumPy. Está prohibido utilizar códigos de terceros en este punto (i.e. Internet o autores externos).

La función debe tener la siguiente sintaxis:

```
def MyCCorrelation_Código1_Código2(image, kernel, boundary_condition):
    ...
    return CCorrelation
```

Es muy importante que compare sus resultados con la función `correlate2d` y asegurarse de arrojar el mismo resultado. Pruebe de tantas formas como pueda, para estar seguro de que funcionan igual. Para ello, puede utilizar una imagen de su elección y los *kernels* definidos en la Figura 3. **No debe reportar estas pruebas en el informe**. Estas deben ser utilizadas únicamente para verificar y garantizar el correcto funcionamiento de sus implementaciones.

1. En el informe, responda: ¿En qué consiste cada modo/condición de frontera de su función de cross-correlación? ¿Cuál condición resulta en una imagen de menor tamaño?

2. Utilice la imagen 2a (*roses.jpg*) y un *kernel* de la Figura 3 de su elección. Convierta la imagen a escala de grises y aplique la cross-correlación con su método y con la función `correlate2d` de Python. Elija un modo/condición de frontera y verifique que utiliza estas funciones con la misma opción.

   2.1. En un subplot, visualice la imagen original, el resultado de su cross-correlación y el resultado de la función de Python.

   2.2. Calcule y reporte el error cuadrático medio pixel a pixel entre los resultados de ambos métodos.

3. Responda: ¿Cuál es la relación de la operación de cross-correlación con la convolución? ¿Cómo se define la cross-correlación normalizada? ¿Cómo es su fórmula? ¿Qué ventaja permite esta operación de normalización frente a la cross-correlación convencional?

5.1.2. Aplicaciones de Cross-Correlación

En esta sección, se estudiará la aplicación de filtros para la **remoción de ruido** en imágenes. Deben utilizar la función de cross-correlación implementada en el punto anterior. Por otro lado, deben utilizar la imagen con ruido como entrada. Recuerde convertir esta imagen RGB a escala de grises antes de proceder con los ejercicios.

1. Aplique los *kernels* 3a y 3b a la imagen. En un subplot, visualice la imagen original, junto con el resultado de la cross-correlación con ambos *kernels*.

2. Responda: ¿Qué está haciendo cada filtro? ¿Qué diferencias observa entre ambos resultados? ¿Por qué la suma de los elementos del kernel debe ser igual a 1? ¿Qué sucede en casos donde la suma da enteros mayores a 1? Respalde su análisis con la visualización generada.

3. Utilizando la función correspondiente, genere un *kernel* Gaussiano de tamaño 5×5 y $\sigma = 1$. Aplique este *kernel* a la imagen. En un subplot, visualice este resultado junto con la cross-correlación de la imagen con el *kernel* 3b.

4. Responda: ¿En qué se diferencian los *kernels* implementados en el punto anterior? ¿Qué diferencias observa entre los resultados de ambos procesos de filtrado?

5. Genere tres *kernels* Gaussianos con tamaño fijo y σ variable. En un subplot, visualice el resultado de la cross-correlación con cada uno de estos.

6. Responda: ¿Cuál es el efecto en la imagen de variar el parámetro sigma (σ)? Justifique su respuesta con teoría y las imágenes del punto anterior.

7. Ahora genere tres *kernels* Gaussianos con σ fijo y tamaño variable. En un subplot, visualice el resultado de la cross-correlación con cada uno de estos.

8. Responda: ¿Cuál es el efecto en la imagen si se varía el tamaño del filtro? Justifique su respuesta con teoría y las imágenes del punto anterior.

En la siguiente sección, se estudiará la aplicación de filtros para la **obtención de características** de imágenes. Para ello, se utilizarán los *kernels* 3c y 3d definidos al comienzo de la sección. En un ejercicio previo, estos filtros fueron aplicados sobre la imagen sin ruido y en escala de grises, para obtener los siguientes resultados:

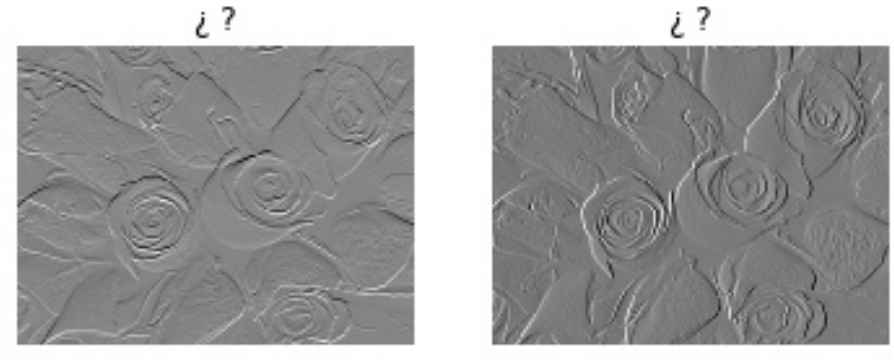
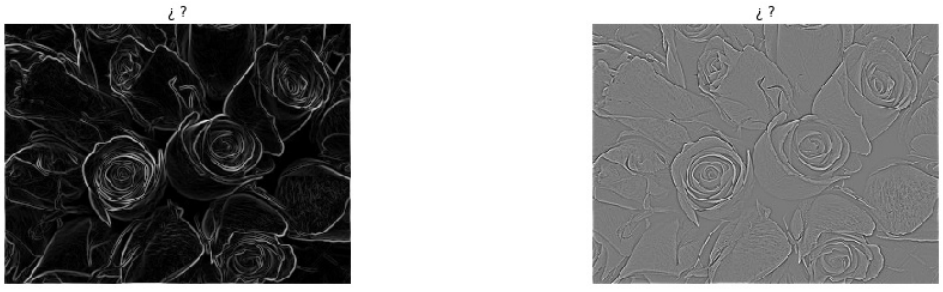


Figura 4: Aplicación de cross-correlación con *kernels* variados.

Su labor es reproducir la Figura 4, aplicando su función de cross-correlación con los *kernels* apropiados. En el informe deben reportar el subplot con la misma configuración y resultados, completando también los títulos de manera apropiada. Además, deben responder: ¿Qué nombre recibe cada uno de los filtros? ¿Qué están haciendo? ¿En qué situación(es) resulta útil su aplicación?

BONO (+0.3)

Se estudiarán **aplicaciones compuestas de procesos de filtrado** sobre imágenes. Para ello, de nuevo, deberán utilizar su implementación de cross-correlación sobre la imagen sin ruido. Su labor (opcional) es reproducir las imágenes de la siguiente figura, aplicando **únicamente** los *kernels* 3c y 3d para la primera imagen y un Gaussiano para la segunda. Puede obtener una otra vista de las imágenes haciendo *click* sobre estas.



(a) (b)

Figura 5: Resultados de operaciones sobre imágenes filtradas con *kernels* variados.

Ayuda: Consideren la imagen 5a como el resultado de una operación sobre procesos individuales de cross-correlación de la imagen con los *kernels* 3c y 3d. Por otro lado, la imagen 5b muestra las frecuencias altas de la imagen original. Utiliza únicamente un *kernel* Gaussiano para su construcción.

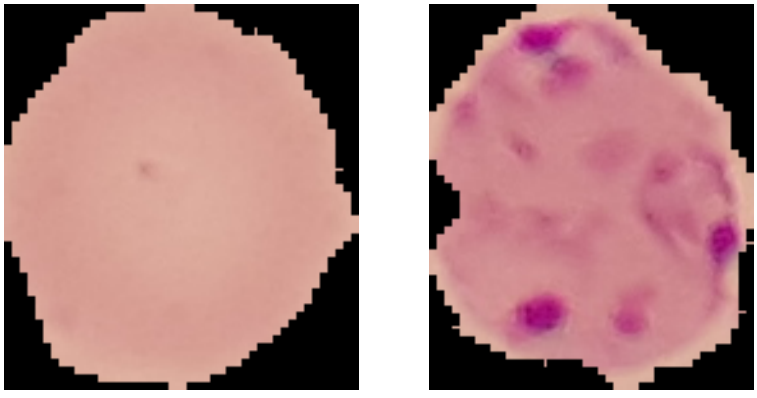
Para recibir el bono completo (0.3 puntos adicionales en la nota de esta entrega), debe incluir en el informe una descripción de los procesos de filtrado y operaciones realizadas sobre la imagen para obtener cada uno de los resultados. Además, debe discutir sobre la utilidad de estas operaciones en el análisis y procesamiento de imágenes. Para la imagen 5a, discuta sobre el efecto de aplicar un filtro de suavizado previo a los otros *kernels* y la operación. Por otro lado, para la imagen 5b, ¿hay alguna manera alternativa de obtener los mismos resultados con otro filtro?

5.2. Problema Biomédico (50 %)

La Inteligencia Artificial y el desarrollo de la Visión por Computador han acelerado el análisis de imágenes biomédicas en diversas tareas de investigación. El diagnóstico de enfermedades mediante imágenes de microscopía está incluido en este avance. En este miniproyecto usted trabajará con imágenes celulares para diagnóstico de malaria.

La malaria es una enfermedad causada por parásitos que se transmiten al ser humano por la picadura de mosquitos hembra infectados del género *Anopheles* [1]. Los parásitos infectan los eritrocitos y se reproducen en su interior por un periodo de 48 a 72 horas. Tras este periodo, rompen la célula para salir e infectar otras. La malaria causa una muerte cada 15 segundos anualmente y es considerada una enfermedad endémica [2].

Una ONG que lucha contra esta enfermedad se puso en contacto con usted, pues desean implementar una metodología de diagnóstico de malaria de bajo costo utilizando imágenes microscópicas de células sanguíneas. El objetivo es determinar si una célula se encuentra sana (Figura 6a) o está infectada con el parásito (Figura 6b). Para evaluar su propuesta algorítmica, la ONG provee una muestra de su base de datos disponible en el siguiente vínculo.



(a) (b)

Figura 6: Muestra de imágenes de la base de datos provista. (a) Célula sanguínea sana, (b) Célula sanguínea con parásito causante de malaria.

En la primera entrega, usted realizará un preprocesamiento sencillo de sus imágenes mediante la especificación de histogramas. El objetivo es resaltar los pixeles que presentan una coloración morada más profunda y que corresponden a los parásitos en una célula infectada. Como sabe de la teoría, uno de los usos más comunes de la especificación de histogramas es resaltar intensidades deseadas en una imagen a partir de otra imagen.
En la base de datos provista, usted encontrará la carpeta 'template'. Esta carpeta contiene tres diferentes imágenes de células, las cuales se utilizarán para realizar la especificación.
Responda en el informe:

1. ¿Cuál es el objetivo de la ecualización y la especificación sobre los histogramas de las imágenes? ¿Qué efecto tienen estos procedimientos sobre las imágenes? ¿Cómo se relaciona la especificación de histogramas con la ecualización?

2. ¿Para que sirve la ecualización de histogramas? ¿Qué esperaría si hace este procedimiento sobre las imágenes en la 6a y 6b? ¿Realizar este proceso resolvería el objetivo principal? ¿Por qué?

3. ¿Para que sirve la especificación de histogramas? Antes de observar las imágenes de la base de datos, ¿cómo espera que sea el histograma objetivo para resaltar las intensidades deseadas en las imágenes de malaria? ¿Cómo debería ser la distribución de intensidades de la imagen para realizar la especificación? ¿Es una imagen donde predominan los tonos claros, oscuros o neutros? Justifique sus respuestas.

4. Observe las imágenes de la carpeta 'template'. ¿Hay alguna imagen que cumpla con las características que describió en el punto anterior? ¿Cuál(es) de las imágenes lo hace?

5. Ya sabiendo cómo son las imágenes de malaria en 'train' y las imágenes de las células en 'template'. Responda qué efecto espera que tenga la imagen *'Parasitized.png'* al especificar su histograma con cada una de las imágenes en 'template'. Justifique.

6. Para probar de manera empírica si acertó en sus anteriores respuestas, cree una función que realice la ecualización de una imagen y la especificación para que esa imagen tenga el histograma de una imagen de referencia. Adicionalmente, esta función debe generar un subplot como se muestra en la Figura 7. Por lo que, su función debe plotear en la primera columna la imagen original, la imagen original ecualizada, la imagen de referencia, la imagen de referencia ecualizada y la imagen especificada y en la segunda columna sus respectivos histogramas. La figura debe permitir correcta visibilidad, incluir ejes donde corresponde y estar debidamente titulada. Esta función debe tener las siguientes características:

   Sintaxis:

   ```
   def myImagePreprocessor(image, target_hist, action):
       ...
       return matched_image
   ```

   Parámetros de entrada:

   - `image`: Imagen en escala de grises.

- `target_hist`: Imagen con el histograma que buscamos imponer en la imagen de entrada.
- `action`: Parámetro de tipo string para guardar ('save') **o** mostrar ('show') el subplot. Para

Respuesta:

- `matched_image`: Imagen de entrada, cuyos niveles de gris han sido modificados mediante la especificación del histograma de entrada.

Para el punto anterior, puede hacer uso de las funciones *match_histograms* y *equalize_hist* de la librería *exposure* de *skimage*.

7. Para la imagen *'Parasitized.png'* de la carpeta 'train', utilice la anterior función con cada una de las imágenes en 'template'. Guarde las figuras generadas y repórtelas en su informe. Analice los resultados obtenidos y compare lo que obtuvo con sus respuestas en los anteriores ítems. ¿Cuál(es) de sus respuestas no coinciden con sus resultados experimentales? ¿Con cuál de las imágenes obtiene un mejor resultado? ¿Esto concuerda con su anterior respuesta? Justifique sus respuestas.

8. Utilice la imagen con la que obtuvo el mejor resultado para especificar su histograma en la imagen de entrenamiento *Uninfected.png*. Reporte estos resultados en su informe y analice si la imagen procesada presenta una mejoría respecto a la original.
   Nota: Mientras realiza sus experimentos, configure el parámetro *action* en 'save'. A la hora de subir el código, configure este parámetro en 'show'. Dadas las dimensiones de la visualizaciones, podrá reportar estas figuras en la sección de Anexos de su informe.

Figura 7: Estructura para subplot para la función myImagePreprocessor.

6. Procedimiento Entrega 2

6.1. Filtro Mediano Adaptativo (40 %)

6.1.1. Función MyAdaptMedian

Implemente una función que devuelva una imagen procesada por el filtro mediano adaptativo. Esta función recibe una imagen en escala de grises con datos de tipo *uint8*, un tamaño de ventana y retorna una imagen procesada del mismo tamaño de la original. Esta función la probaremos solamente con tamaños de ventana impares.

Para este punto no pueden utilizar funciones nativas de Python relacionadas con el método. Únicamente pueden hacer uso de la librería NumPy. Está prohibido utilizar códigos de terceros en este punto (i.e. Internet o autores externos).

La función implementada debe tener la siguiente sintaxis:

```
def MyAdaptMedian_Codigo1_Codigo2(gray_image, window_size, max_window_size):
    ...
    return filtered_image
```

Figura 8: Imagen original sin ruido.

6.1.2. Aplicaciones de filtrado

A la imagen de la figura anterior, se le aplicó dos tipos de ruido para producir las imágenes *noisy1.jpg* y *noisy2.jpg*. El objetivo es que, mediante el filtrado, puedan reducir lo mejor posible este ruido (y retornar a un resultado similar a aquel de la Figura 8).

1. Descargue las imágenes RGB ruidosas del siguiente vínculo y conviértalas a escala de grises.

2. Explique brevemente (máximo un párrafo), en qué consisten los ruidos sal y pimienta, uniforme y gaussiano. A partir de sus explicaciones, analice qué tipo de ruido tienen las imágenes descargadas. ¿Es el mismo tipo de ruido para ambas imágenes?

3. Pruebe con 3 tamaños de ventana diferentes aplicar su función de filtro mediano adaptativo a las imágenes con ruido. Muestre sus experimentos en un subplot con títulos las imágenes con ruido y las respuestas después del filtrado. Responda ¿cómo cambia la imagen resultante al variar el tamaño de la ventana?

4. Pruebe filtrar ambas imágenes con distintos *kernels* Gaussianos, que varíen en tamaño y valor de σ. Utilice la función `gaussian_kernel` definida en la Sección **5.1**. Dado que unos experimentos similares se han realizado en la entrega pasada, no es necesario reportar visualizaciones para sus pruebas. Experimente con diferentes parámetros y halle la combinación que permita reducir de mejor manera el ruido en una o ambas de las imágenes dadas para este punto.

5. Muestre en un subplot el resultado del mejor filtrado para cada imagen y diga con qué filtro y qué parámetros se produce la mejor imagen.

6.2. Continuación Problema Biomédico (60 %)

En esta entrega, se evaluará el efecto de la especificación de las imágenes en su clasificación a través de un método conocido como *Template matching*. Este procedimiento consiste en la selección de un *kernel* que permite identificar patrones específicos dentro de una imagen, característicos de la clase a la que pertenece la imagen y que la diferencian de otras. Este *kernel* es aplicado a la imagen por medio de la cross-correlación normalizada. A partir de sus respuestas (cualitativa y cuantitativa), se puede determinar si en una imagen existe el patrón de interés, lo que permite su clasificación.

1. Modifique la función implementada en la entrega anterior (Sección 5.1.1) para realizar la cross-correlación normalizada. Esta función debe tener las siguiente sintaxis:

   ```
   def MyNormCCorrelation_Código1_Código2(image, kernel, boundary_condition):
       ...
       return normalized_ccorrelation
   ```

   Por ser una nueva versión de su función anterior, debe tener los mismos parámetros de entrada, pero una respuesta normalizada. Para su implementación, revise las preguntas teóricas propuestas en el numeral 3 de la Sección 5.1.1.

2. Analice las imágenes de entrenamiento y defina dos *kernel* que permitan identificar un patrón diferenciador entre ambas clases. Un kernel debe definirlo a partir de las imágenes originales, y otro a partir de las imágenes preprocesadas. Tenga en cuenta que la respuesta a los *kernel* será utilizada para la clasificación de sus imágenes. En este punto, no debe basarse en las imágenes de prueba. En una figura dentro del informe, debidamente titulada, reporte los *kernel* seleccionados y justifique brevemente su elección. Responda: ¿Cómo espera que sean las respuestas cualitativas de cada tipo de imagen al *kernel* seleccionado? ¿Espera que la respuesta cuantitativa de las imágenes con células sanas sea mayor o menor que aquella de las células infectadas? ¿Por qué?

3. Utilizando la función de cross-correlación normalizada recién implementada, filtre sus imágenes utilizando los *kernel* seleccionados. Este procedimiento lo deberá realizar en sus imágenes en escala de gris originales y en sus imágenes en escala de gris ecualizadas.

4. Defina un medio para resumir la magnitud de cada respuesta en un único valor. Esta será la respuesta cuantitativa de la cross-correlación normalizada y servirá próximamente para realizar la clasificación binaria en de sus imágenes.

5. A partir de la respuesta generada por los *kernel*, podrá clasificar automáticamente cada una de sus imágenes. Esto lo realizará a partir de la hipótesis realizada sobre la magnitud de las respuestas en una de las preguntas teóricas del numeral 2. Estas condiciones definirán un criterio para asignar a cada imagen una etiqueta.

6. En el informe, presente sus resultados en dos tablas, una para cada kernel. Esta deberá tener la estructura que muestra la Figura 9.

7. En el informe, analice los resultados obtenidos. ¿Cómo se comparan las respuestas a la cross-correlación normalizada de las imágenes preprocesadas (con el histograma especificado) frente a las originales?

Imagen	**Con histograma especificado**	**Respuesta al *kernel* 1**	**Predicción**
1	Sí	0.9	Parasitized
2	Sí	0.2	Uninfected
…	…	…	…
10	No	0.65	Parasitized

Imagen	**Con histograma especificado**	**Respuesta al *kernel* 2**	**Predicción**
1	Sí	0.6	Uninfected
2	Sí	0.1	Uninfected
…	…	…	…
10	No	0.9	Parasitized

Figura 9: Estructura para tabla de resultados de *template matching*.

8. Responda: ¿Por qué cree que el *template matching* funciona para clasificación? ¿Qué limitaciones tiene este método para otro tipo de bases de datos y tareas?

7. Procedimiento Entrega 3

7.1. Matriz de Confusión, Precisión y Cobertura (30 %)

7.1.1. Función MyConfMatrix

En esta sección, deberán crear una función que calcule la matriz de confusión de un método. Esta debe generalizar a problemas multiclase (3 o más clases). A partir de este cálculo, se deben obtener la precisión y cobertura por clase y promedio alcanzadas por el método. El nombre de la función debe ser *MyConfMatrix_Código1_Código2* y recibe como entrada las anotaciones y las predicciones del método. La función debe retornar la matriz de confusión, precisión y cobertura para cada clase y precisión y cobertura promedio.

La sintaxis de la función a implementar debe ser la siguiente:

```
def MyConfMatrix_Código1_Código2(gt, pred):
    ...
    return conf_matrix, prec_class, rec_class, mean_prec, mean_rec
```

Parámetros de entrada:

- `gt`: Vector con las anotaciones.
- `pred`: Vector con las predicciones.

Respuesta:

- `conf_matrix`: Matriz de tamaño $n \times n$ (donde n es el número de clases) con las respectivas entradas provenientes de la clasificación.
- `prec_class`: Diccionario con la precision calculada por clase. Debe tener la estructura `{'no. clase': precision}`.
- `rec_class`: Diccionario con la cobertura calculada por clase. Debe tener la estructura `{'no. clase': recall}`.
- `precision`: Precisión promedio, reportada con tres decimales.
- `recall`: Cobertura promedio, reportada con tres decimales.

Para este punto no pueden utilizar funciones nativas de Python relacionadas con el método. Únicamente pueden hacer uso de la librería NumPy. Está prohibido utilizar códigos de terceros en este punto (i.e. Internet o autores externos).

7.1.2. Aplicación

Para evaluar el funcionamiento de su implementación, utilice la información provista en el siguiente vínculo. Allí encontrará los datos de la anotación y las predicciones de un método externo de clasificación de especies de flores en cuatro categorías. En esta instancia, pueden utilizar funciones nativas de Python **únicamente** para comprobar el correcto funcionamiento de sus implementaciones. Esta verificación es de alta importancia, pues este método de evaluación se debe utilizar en la siguiente sección.

En el informe:

1. Reporte la matriz de confusión, precisión y cobertura por clase y precisión y cobertura promedio del método de prueba, obtenidos a través de su implementación.

2. Responda: ¿Cuál es la relación entre la precisión y la cobertura?

3. Considere un problema de clasificación en tres clases (A, B y C), con una matriz de confusión descrita por la estructura en la Figura 10. Para una de las clases, complete las celdas en blanco utilizando las siguientes entradas: **TP** (*true positives*), **TN** (*true negatives*), **FP** (*false positives*) y **FN** (*false negatives*).

		Predicciones		
		A	B	C
Anotaciones	A			
	B			
	C			

Figura 10: Estructura a completar para matriz de confusión.

4. Responda: ¿Cómo se define la F-Medida? ¿Cuál es su relación con las métricas de precisión y cobertura?

5. Responda: Además de las predicciones, ¿qué información adicional debe ser provista por un método de clasificación para poder construir una curva de precisión-cobertura?

7.2. Etapa Final Problema Biomédico (70 %)

Ahora que ha logrado una clasificación inicial de sus imágenes, quiere probar si el uso de las imágenes a color en lugar de en escala de gris puede ser una buena aproximación a este problema. Por tal motivo, decide basarse en los distintos espacios de color para encontrar un método que le permita clasificar sus imágenes.

1. En primer lugar, debe implementar una función que genere el histograma concatenado de color para sus imágenes.

   Sintaxis:

```
MyColorHist_Codigo1_Codigo2(color_image, space, plot = True):
    ...
    return hist
```

Parámetros de entrada:

- `color_image`: La imagen a color para la que desea obtener el histograma concatenado de color.
- `space`: Un *string* representando el espacio de color a utilizar. Puede ser 'RGB', 'HSV' o 'Lab'.
- `plot`: Parámetro opcional para mostrar la imagen original, los tres canales de la imagen y el histograma concatenado en el espacio de color designado. La figura debe estar debidamente titulada. Para construir esta figura, puede utilizar como base el código que encuentra al final de esta sección (7.2).

Respuesta:

- `hist`: Histograma concatenado y normalizado de la imagen a color.

Para este punto, puede hacer uso de funciones nativas de Python para las transformaciones entre los espacios de color y para generar histogramas. Sin embargo, no puede utilizar funciones ni implementaciones (nativas o externas) que generen histogramas de color concatenados.

2. Utilice su función para visualizar los histogramas concatenados de sus imágenes de entrenamiento (*train*) en los tres espacios de color designados. En el informe, responda: ¿Hay algún espacio que maximice la diferencia visual entre las dos clases de imágenes? ¿Hay algún canal específico que la maximice? Respalde su respuesta con visualizaciones de aquellos canales.

3. Para realizar una comparación cuantitativa de sus histogramas, utilizará una función de *kernel de intersección* de histogramas.

   3.1. Responda: ¿Cómo se define formalmente un *kernel de intersección*? ¿De qué manera realiza una comparación entre histogramas? ¿Cómo es el valor del *kernel* cuando dos histogramas son iguales, comparado al valor que toma cuando estos son disímiles?

   3.2. Realice una implementación propia del *kernel de intersección* para dos histogramas con igual número de *bins*. La función debe cumplir con las siguientes características:

   Sintaxis:

   ```
   MyIntersectionKernel_Código1_Código2(hist1, hist2):
       ...
       return value
   ```

   Parámetros de entrada:

   - `hist1, hist2`: Histogramas a comparar. Asuma que contienen el mismo número de *bins*.

   Respuesta:

   - `value`: Respuesta numérica representando la intersección entre `hist1` e `hist2`.

3.3. Responda: ¿Qué otras funciones existen para la comparación de histogramas? ¿Por qué se usan estas distancias para histogramas y no distancias como la euclidiana?

4. Utilizando las funciones descritas en los literales 1 y 3*b*, deberá comparar cada una de sus imágenes de prueba (*test*) con sus dos imágenes de entrenamiento (*train*). Para ello, debe obtener el histograma concatenado de color para todas las imágenes (entrenamiento y prueba) con un mismo número de *bins*. Posteriormente, debe comparar cada uno de los histogramas de prueba con cada uno de los histogramas de entrenamiento, utilizando la función *kernel de intersección*. Este proceso debe repetirse para los tres espacios de color considerados (RGB, HSV y Lab).

5. Los valores que obtenga para la función de intersección, le permitirán clasificar sus imágenes de prueba. Esto lo hará determinando cuál imagen de entrenamiento tiene una intersección mayor a su imagen de prueba. En consecuencia, la etiqueta que deberá asignarle a esa imagen de prueba corresponderá a la etiqueta de la imagen de entrenamiento cuyo histograma tiene una intersección mayor con el histograma de la imagen de prueba. Esta clasificación debe realizarse de forma automática, mediante su código. En el informe, debe presentar sus resultados en una tabla con la estructura de la Figura 11.

Imagen	Espacio de color	Intersección con clase 'Uninfected'	Intersección con clase 'Parasitized'	Predicción
1	RGB	0.8	0.2	Uninfected
2	RGB	0.6	0.35	Uninfected
…	…	...	...	...
1	HSV	0.5	0.3	Uninfected
…	…	…	…	…
10	Lab	0.1	0.85	Parasitized

Figura 11: Estructura para tabla de resultados para intersección de histogramas de color concatenados.

Nota: Dadas las dimensiones de esta tabla, podrá reportarla en la sección de Anexos. No es necesario imprimir esta tabla en código.

6. En este punto, cuenta usted con cinco (5) métodos para la clasificación de sus imágenes: *Template matching* con imágenes en escala de gris, *Template matching* con imágenes en escala de gris especificadas y kernel de intersección con imágenes a color para cada espacio. Deberá utilizar la función implementada en la Sección 7.1 para evaluar sus métodos y así podrá comparar su desempeño a través de las métricas de precisión y cobertura promedio y F-Medida. Para su evaluación, utilice las etiquetas que se encuentran en el archivo `csv` dentro de la carpeta de datos.

7. En el informe, muestre los resultados obtenidos en una tabla como se muestra en la Figura 12.

8. Analice los resultados obtenidos. ¿Cuál es el método que da mejores resultados? ¿Por qué cree que este método es mejor? ¿Resulta importante utilizar la información del color para la clasificación de imágenes?

Método	Precisión	Cobertura	F-Medida
Template matching + imágenes originales	0.2	0.8	0.32
Template matching + imágenes especificadas	0.9	0.1	0.18
…	…	…	…
Kernel de intersección + RGB	0.4	0.2	0.27

Figura 12: Estructura para tabla de comparación de métodos de clasificación.

Código de visualización

```
plot_image(im, im1, im2, im3, hist):
    fig = plt.figure(figsize=(8,8))
    ax1 = plt.subplot2grid((3, 4), (0, 0), rowspan=1, colspan=1)
    ax2 = plt.subplot2grid((3, 4), (0, 1), rowspan=1, colspan=1)
    ax3 = plt.subplot2grid((3, 4), (0, 2), rowspan=1, colspan=1)
    ax4 = plt.subplot2grid((3, 4), (0, 3), rowspan=1, colspan=1)
    ax5 = plt.subplot2grid((3, 4), (2, 0), rowspan=1, colspan=4)

    ax1.imshow(image)
    ax1.set_title('Title Image')
    ax1.set_axis_off()

    ax2.imshow(image1)
    ax2.set_title('Title Image 1')
    ax2.set_axis_off()

    ...

    ax5.plot(hist)
    ax5.set_title('Title Histogram')
```

BONO (+0.3)

La distancia chi-cuadrado (χ^2) se utiliza comúnmente para la comparación de histogramas. Para obtener este bono, deberá realizar su propia implementación para calcular esta distancia entre dos histogramas con igual número de *bins*. La función debe cumplir con las siguientes características:

Sintaxis:

```
MyChi2Distance_Código1_Código2(hist1, hist2):
```

```
...
return value
```

Parámetros de entrada:

- `hist1, hist2`: Histogramas a comparar. Asuma que contienen el mismo número de *bins*.

Respuesta:

- `value`: Respuesta numérica representando la distancia χ^2 entre `hist1` e `hist2`.

Posteriormente, deberá reproducir la clasificación realizada con el *kernel de intersección*. Para ello, debe utilizar la implementación bono para realizar la comparación entre las imágenes de entrenamiento y prueba. Obtendrá así, un sexto método de clasificación de imágenes que deberá incluir en las comparaciones de la Figura 12.

El bono será otorgado únicamente si se cumplen todos los pasos en el flujo de clasificación, desde la implementación de la distancia χ^2 hasta la evaluación del método que la utiliza para realizar la clasificación de imágenes.

Referencias

[1] Organización Mundial de la Salud. *Paludismo*, Organización Mundial de la Salud, 30 de noviembre de 2020. [En línea]. Disponible en `https://www.who.int/es/news-room/fact-sheets/detail/malaria`.

[2] Biblioteca Nacional de Medicina de los EE. UU. *Malaria*, MedlinePlus, 25 de agosto 2020. [En línea]. Disponible en `https://medlineplus.gov/spanish/ency/article/000621.htm`.