Laboratorio de Análisis y Procesamiento de Imágenes
Módulo II
Fechas de realización: 11 de septiembre - 3 de octubre de 2020

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 o *.ipynb*.
- 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 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 el proyecto**.
- 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: Sábado 19 de septiembre de 2020
  - Entrega 2: Sábado 26 de septiembre de 2020
  - Entrega 3: Sábado 3 de octubre de 2020

  El plazo de entrega será hasta las 11:59 p.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.

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.

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.
5. 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.

6. 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*.

   6.1. Forma incorrecta:
   `skimage.io.imread('C:/Estudiante/docs/ElProyecto/ims/im.png')`

   6.2. 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.** Recuerden incluir imágenes de sus resultados y documentarlas debidamente. Por último, el informe tiene una longitud máxima de 4 páginas. 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

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 verlo se puede observar en la Figura 1.

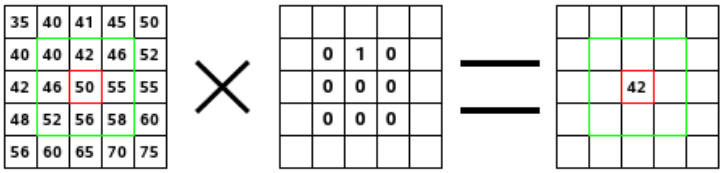


Figura 1: Representación matricial de la cross-correlación.

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 *daisy.jpg* y *noisy_daisy.jpg*.

(a) (b)

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

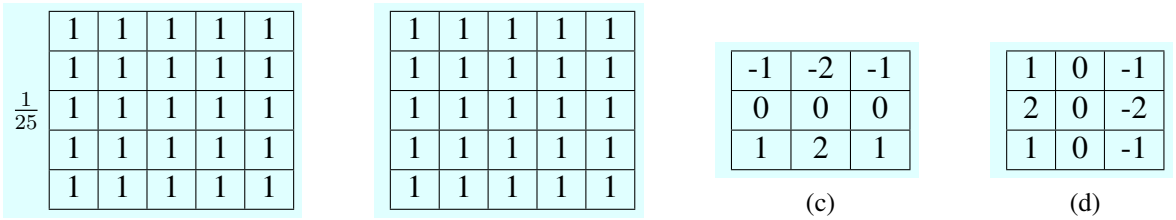


Figura 3: *Kernels* a utilizar.

Además de estos *kernels*, deberán utilizar un filtro Gaussiano 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
```

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, wrap, valid*), cuyas descripciones pueden encontrar en la documentación de la función `scipy.signal.correlate2d` de Python.

Para este punto no pueden utilizar las funciones nativas de Python relacionadas con el método.

La función debe tener la siguiente sintaxis:

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

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 condición de frontera de su función de cross-correlación? ¿Cuál condición resulta en una imagen de menor tamaño?

2. En este vínculo encontrará una matriz pequeña y un *kernel* de prueba. Utilice estas dos entradas para evaluar gráficamente la equivalencia en funcionamiento de su implementación, con respecto a la función `correlate2d` de Python. En un subplot, visualice la imagen original (matriz), el resultado de su cross-correlación y el resultado de la función de Python.

3. Utilice la imagen 2a (*daisy.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.

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

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

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* 3a.

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:

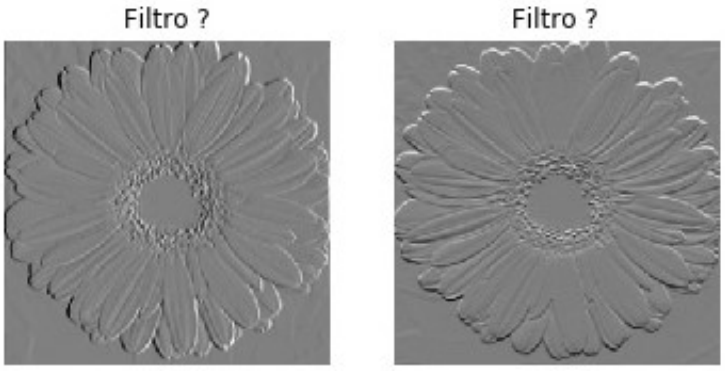
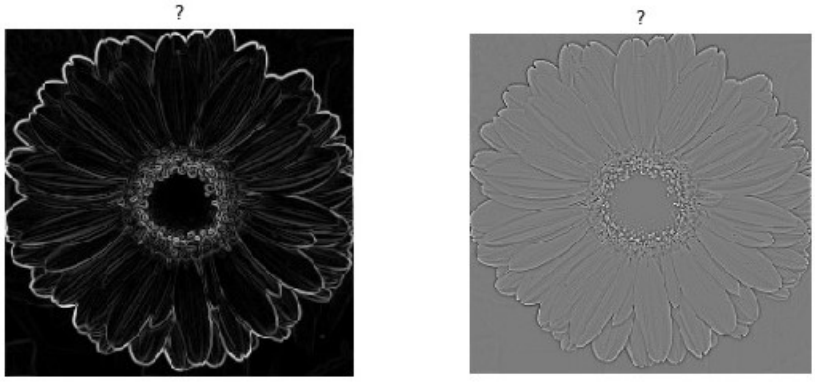


Figura 4: Aplicación de cross-correlación con *kernels* variados.

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 está haciendo cada uno de los filtros? ¿En qué situación(es) resulta útil su aplicación?

Para la última sección, se estudiarán **aplicaciones compuestas de procesos de filtrado** sobre imágenes. De nuevo, deben utilizar su implementación de cross-correlación sobre la imagen sin ruido.

Su labor es reproducir las imágenes de la siguiente figura, aplicando **únicamente** un *kernel* Gaussiano y los *kernels* 3c y 3d.



(a) (b)

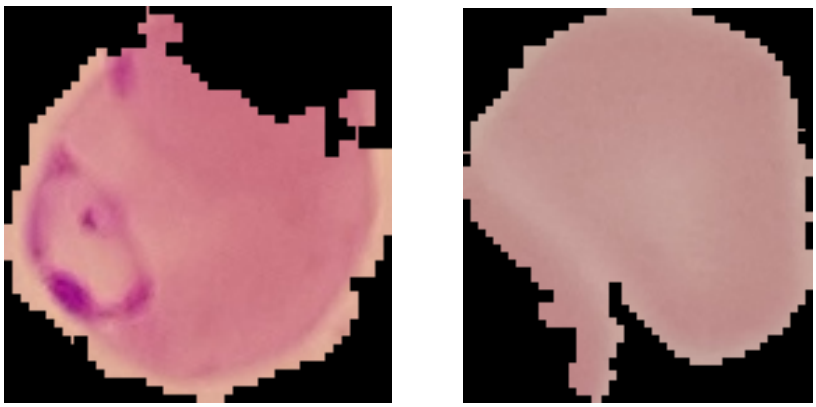
Figura 5: Resultados de operaciones sobre imágenes filtradas con *kernels* variados.

En el informe deben incluir una descripción detallada de los procesos de filtrado y operaciones realizadas sobre la imagen para obtener cada uno de los resultados. Además, deben 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?

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 un *kernel* Gaussiano para su construcción.

5.2. Problema Biomédico

El análisis de imágenes médicas utilizando Inteligencia Artificial ha aumentado y ha sido aplicado en diversas tareas. Una de estas es el diagnóstico de enfermedades por medio de imágenes de microscopía. En este miniproyecto usted trabajará con imágenes celulares para diagnóstico de malaria. La malaria es una enfermedad causada por un parásito y es transmitida por un vector infectado, en este caso por mosquitos *Anopheles* que han sido infectados con alguno de los parásitos que producen la enfermedad. 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 [1].
Una ONG que lucha contra esta enfermedad se puso en contacto con usted, pues desean implementar una metodología de diagnóstico de malaria a bajo costo utilizando imágenes microscópicas de células sanguíneas buscando determinar si una célula se encuentra infectada con el parásito (Figura 6a) o está sana (Figura 6b). Puede descargar la base de datos en este vínculo.



(a) (b)

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

En esta primera entrega, usted realizará el pre-procesamiento de sus imágenes por medio del método de Especificación de histogramas. Su objetivo será resaltar los píxeles que presentan una coloración morada y que corresponden a los parásitos en una célula infectada. Como sabe, el método de especificación de histogramas puede ser utilizado para resaltar intensidades deseadas en una imagen. Para

realizar este procedimiento deberá implementar una función que permita la especificación de histogramas y la aplicará a sus imágenes de forma que se resalten las intensidades que corresponden a los parásitos, tenga en cuenta que para aplicar este procedimiento deberá convertir sus imágenes a escala de gris.
Para este punto **solo puede usar la librería NumPy**. **NO podrá hacer uso** de funciones de ecualización ni especificación desarrolladas por otras personas, sean de Internet o de terceros.

1. Escriba su propia función de especificación de histogramas llamada `MyHistEsp_Codigo1_Codigo2(image, target_hist)` que toma de entrada:

   - `image` Una imagen en escala de grises.
   - `target_hist` El histograma que buscamos especificar (imponer) en la imagen de entrada.

   La función debe retornar:

   - `new_image` La imagen de entrada cuyos niveles de gris han sido modificados por medio de la especificación del histograma de entrada.

2. Para utilizar como paso intermedio en el proceso de especificación de histograma escriba una función de ecualización de histograma llamada `MyHistEq_Codigo1_Codigo2(hist)` que toma de entrada:

   - `hist` El histograma a ecualizar.

   La función debe retornar:

   - `histeq` El histograma de entrada ecualizado.
   - `table` Un diccionario cuyas llaves corresponden a los niveles de gris del histograma original y cuyos valores son los s_k redondeados obtenidos tras realizar la ecualización. Por ejemplo para un histograma con niveles de gris 0, 1, 2 y 3 y s_k redondeados 1, 3, 5 y 6, el diccionario estaría definido como {0:1, 1:3, 2:5, 3:6}.

3. Muestre un subplot en donde se vea el histograma seleccionado como su plantilla para realizar la especificación (`target_hist`), su imagen original (en escala de gris) y su histograma, la imagen ecualizada y su histograma, la imagen especificada y su histograma. La estructura de su subplot deberá ser la mostrada en la Figura 7, recuerde incluir títulos y grafique solo el subplot completo no cada una de las imágenes separadas. Incluya esta gráfica en su informe.

4. Explique cómo seleccionó su histograma objetivo, ¿qué factores tuvo en cuenta?

5. ¿Qué efecto tienen la ecualización y la especificación en las imágenes de células sanas?

6. ¿Qué efecto tienen la ecualización y la especificación en las imágenes de células con el patógeno?

7. ¿De qué otra forma cree que se podría utilizar la ecualización de histogramas para mejorar el desempeño de un algoritmo? ¿Cómo se podría utilizar la especificación de histogramas con este mismo fin?

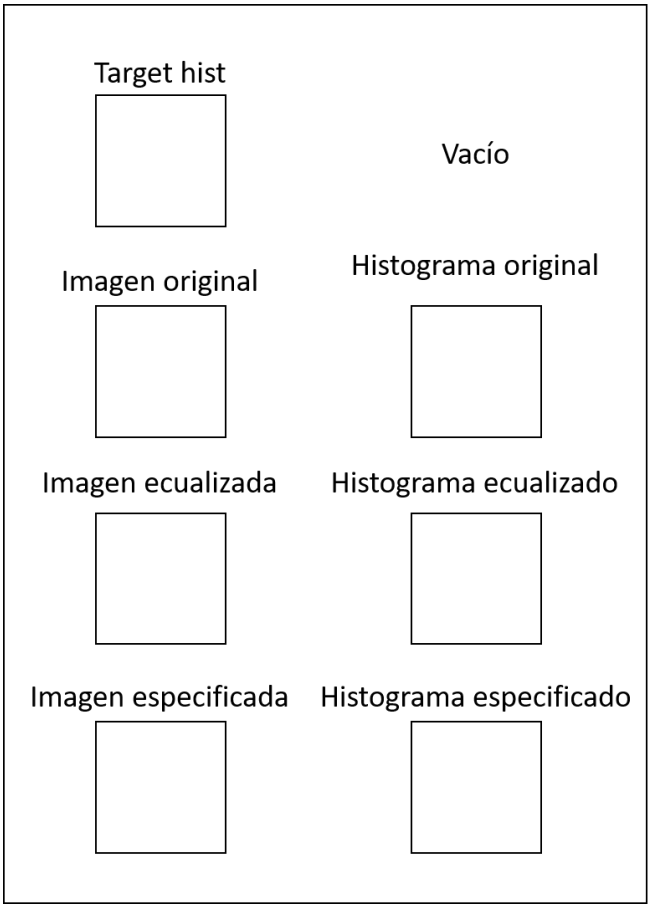


Figura 7: Estructura para subplot de Especificación de Histogramas

6. Procedimiento Entrega 2

6.1. Filtro Mediano Adaptativo

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 su implementación, solo pueden utilizar el paquete NumPy de Python. No está permitida la utilización de códigos externos, sean de Internet o de terceros.

La función implementada debe tener la siguiente sintaxis:

```
def MyAdaptMedian_Codigo1_Codigo2(image, window_size, max_window_size):
    ...
    return
```

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 *noisy_1.jpg* y *noisy_2.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 ruidosas del siguiente vínculo.

2. Explique brevemente (máximo un párrafo), en qué consisten los ruidos sal y pimienta, uniforme y gaussiano. Analice qué tipo de ruido tienen las imágenes. ¿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

Ya teniendo sus imágenes preprocesadas, es momento de continuar con su método de clasificación de células para el diagnóstico de malaria. Como primera etapa, hará uso de un procedimiento conocido como *Template Matching*. Este procedimiento consiste en la selección de un kernel que permita identificar patrones específicos dentro de una imagen, que son características de la clase a la que pertenece dicha imagen y la diferencian de otras clases. Este kernel es aplicado por medio de la *cross-correlation* y a partir de la respuesta obtenida se puede determinar si en la imagen existe el patrón de interés.

1. Defina dos kernels que permitan identificar los patrones en cada clase. Para ello, debe utilizar las imágenes que encuentra en la carpeta de entrenamiento y no debe basarse en las imágenes de prueba. Debe mostrar sus kernels seleccionados en el informe y explicar la razón de su elección. Tenga en cuenta que la respuesta al kernel será utilizada para la clasificación de sus imágenes.

2. Utilizando la función de Cross-Correlación que realizó en la Sección 5.1, filtre sus imágenes utilizando su kernel seleccionado. Este procedimiento lo deberá realizar en sus imágenes en escala de gris sin pre-procesamiento y en sus imágenes en escala de gris especificadas.

3. A partir de la respuesta generada por el kernel, podrá clasificar sus imágenes. Decida qué respuesta es la esperada para cada clase y determinado kernel, por ejemplo, podría decir que para su kernel 1 lo esperado es que la clase de células con malaria tengan una respuesta muy alta y que las células sanas tengan una baja respuesta o ninguna.

4. A partir de las condiciones que decidió en el ítem anterior, asigne a cada imagen la etiqueta correspondiente.

5. Presente en una tabla sus resultados. La tabla deberá tener la estructura que se muestra en la Figura 9.

6. En el informe analice los resultados obtenidos a partir de lo que evidencia, ¿cree que las imágenes especificadas presentan un mejor o peor resultado?¿cuál kernel cree que da un mejor resultado?

7. ¿Por qué cree que el *Template Matching* funciona para clasificación?

Imagen	Kernel	Prepocesamiento	Respuesta al kernel	Predicción
1	1	Si	0.8	Parasited
2	2	Si	0.1	Uninfected
. . .	. . .	. . .	. . .	. . .
10	1	No	0.7	Uninfected

Figura 9: Estructura para tabla de resultados de *Template Matching*

8. ¿Qué limitaciones cree que tiene este método para otro tipo de bases de datos o tareas?

7. Procedimiento Entrega 3

7.1. Matriz de Confusión, Precisión y Cobertura

En esta sección, deberán crear una función que calcule la matriz de confusión de un método. A partir de este cálculo, se deben obtener la precisión y cobertura alcanzadas por el método. El nombre de la función debe ser *myConfusionMatrix_Código1_Código2* y recibe como entrada un vector con las anotaciones (`gt`) el vector con las predicciones (`pred`). La función debe retornar la matriz de confusión, precisión y cobertura correspondientes a las entradas.

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

```
def myConfusionMatrix_Código1_Código2(gt,pred):
    ...
    return conf_matrix, precision, recall
```

Para su implementación, no puede utilizar funciones nativas de Python relacionadas a este método.

Tenga en cuenta que sus funciones deben generalizar a problemas multiclase (3 o más clases). En estos casos, su método debe retornar la precisión y cobertura promedio para todas las clases.

Para evaluar el funcionamiento de su implementación, utilice la información provista en el siguiente vínculo. Allí encontrará los datos de la anotación (*groundtruth*) y las predicciones de un método externo de clasificación de especies de flores en cinco 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 promedio del método de prueba, obtenidos a través de sus implementaciones.
2. Responda: ¿Cuál es la relación entre la precisión y la cobertura?
3. En un problema multiclase: ¿Qué otro nombre recibe la cobertura promedio?
4. 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. Complete las celdas en blanco utilizando las siguientes entradas: **TP** (*true positives*), **TN** (*true negatives*), **FP** (*false positives*) y **FN** (*false negatives*).

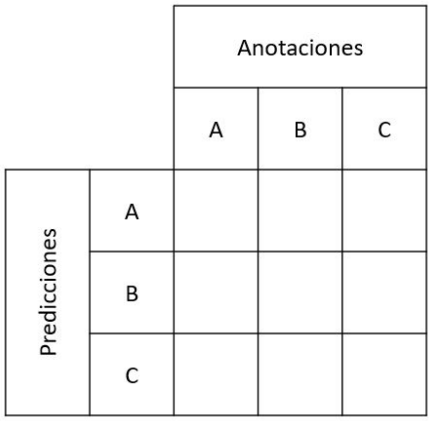


Figura 10: Estructura a completar para matriz de confusión.

7.2. Etapa Final Problema Biomédico

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. Deberá implementar una función que genere el histograma concatenado de color para sus imágenes.

   Su función se debe llamar `MyColorHist_Código1_Código2` y debe recibir los siguientes parámetros:

   - `image`: La imagen a color para la que desea obtener el histograma concatenado de color.
   - `espacio`: El espacio de color a utilizar, puede ser RGB, HSV o Lab.

   Esta función debe tomar la imagen de entrada, transformarla al espacio de color deseado y retornar el histograma concatenado y normalizado de la imagen a color. Puede hacer uso de funciones de Python para las transformaciones entre los espacios de color y para generar histogramas, sin embargo no puede utilizar funciones ni implementaciones que generen histogramas de color concatenados.

2. Deberá hacer uso de la función compareHist de OpenCV. Esta función les permite calcular la distancia entre dos histogramas, pueden encontrar un tutorial para el uso de la misma en el siguiente vínculo.

3. Utilizando las funciones descritas en los literales anteriores, debe comparar cada una de sus imágenes de prueba con sus dos imágenes de entrenamiento. Es decir, deberá obtener el histograma concatenado de color para cada imagen de entrenamiento y de prueba y comparar cada imagen de prueba con cada imagen de entrenamiento utilizando las distancias kernel de intersección y χ^2 para cada espacio de color.

4. Los resultados que obtenga para las distancias le permitirá clasificar sus imágenes de prueba. Esto lo hará determinando cuál imagen de entrenamiento tiene una distancia menor a su imagen de prueba. La etiqueta que deberá asignarle a esa imagen de prueba corresponderá a la etiqueta de la imagen de entrenamiento cuyo histograma tiene una distancia menor al histograma de la imagen de prueba. Sus resultados los deberá mostrar en una tabla con la estructura de la Figura 12.

Imagen	Tipo de distancia	Espacio de color	Distancia a clase Uninfected	Distancia a clase Parasited	Predicción
1	Kernel de intersección	RGB	0.8	0.3	Parasited
2	Kernel de intersección	HSV	0.1	0.9	Uninfected
. . .	. . .	. . .	. . .		. . .
10	χ^2	Lab	0.2	0.5	Uninfected

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

5. Ahora tiene varios métodos para la clasificación de sus imágenes: *Template matching* con imágenes en escala de gris, *Template matching* con imágenes en escala de gris especificadas con sus dos kernels, distancia de kernel de intersección con imágenes a color y distancia χ^2 en imágenes a color para cada espacio de color. Utilizando las métricas de precisión y cobertura que implementó en la Sección 7.1 deberá evaluar sus distintos métodos y así podrá comparar sus resultados. Para esto haga uso de las etiquetas que encuentra en el archivo `csv` que encuentra en la carpeta de datos.

6. Muestre los resultados obtenidos en una tabla como se muestra en la Figura 12.

Método	Precisión	Cobertura
Kernel de intersección + RGB	0.2	0.8
Template matching + Kernel 1	0.9	0.1
. . .	. . .	. . .
χ^2 + HSV	0.4	0.2

Figura 12: Estructura para tabla de resultados de precisión y cobertura para comparación de todos los métodos.

7. Analice los resultados obtenidos. ¿Cuál es el método que da mejores resultados? ¿Por qué cree que este método es mejor?

8. Defina y muestre la ecuación de la distancia de kernel de intersección y de la distancia χ^2.

9. ¿Por qué se usan estas distancias para histogramas y no distancias como la euclidiana?

Referencias

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