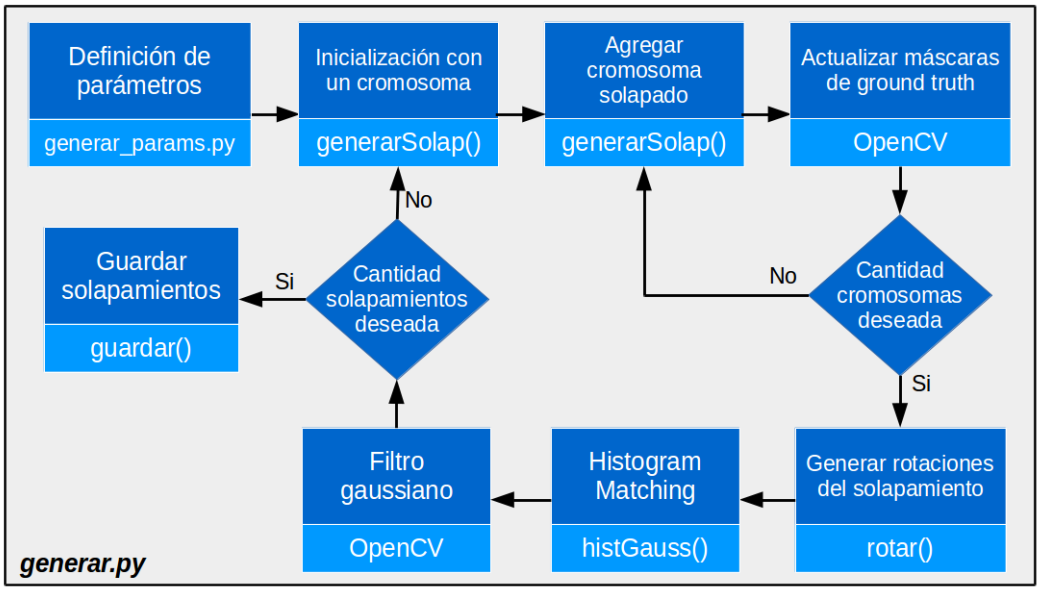
Referencia del módulo generar

Script que aplica las funciones definidas en 'generarAux.py' para la generación de solapamientos sintéticos a partir de los parámetros definidos en '**generar_params.py**'.

Descripción detallada

Script que aplica las funciones definidas en 'generarAux.py' para la generación de solapamientos sintéticos a partir de los parámetros definidos en '**generar_params.py**'.

En la siguiente imagen puede verse cómo se relacionan entre sí.



Los cromosomas individuales en imágenes separadas se obtienen mediante la función ‘procesarCariogramas()’ definida en el archivo ‘separarKaryo.py’.

En el archivo ‘generar_params.py’ se definen los parámetros con los que se crearán los solapamientos sintéticos. Ver la documentación de dicho archivo para ver listados los parámetros.

El proceso comienza con la creación de una imagen en blanco con un cromosoma e inicializando la lista que contendrá el ground truth de la correcta separación del solapamiento (‘masksCanales’). Posteriormente, se entra en un bucle en el que se desplaza un poco al azar el solapamiento que se tiene hasta el momento con ‘desplazarResize()’ antes de agregar un cromosoma al azar al mismo con ‘generarSolap()’. Luego, se actualizan las máscaras de ground truth teniendo en cuenta qué cromosomas son visibles mediante el uso de operaciones lógicas entre ellas de OpenCV. Se sale del mencionado bucle cuando se logra un solapamiento con la cantidad buscada de cromosomas o se recomienza por algún error.

Una vez que se tiene un solapamiento, el mismo rota un ángulo al azar la cantidad de veces indicada en ‘generar_params,py’ mediante la función ‘rotar()’. Luego, a cada uno se le aplica el proceso de histogram matching con un desvío al azar entre los parámetros definidos con el uso de la función ‘histGauss()’. Seguido a esto, cada imagen se filtra con un kernel gaussiano de 3x3 con una media al azar entre otros parámetros definidos. Por último, los solapamientos se centran en una imagen de tamaño definido en ‘generar_params.py’ con ‘desplazarResize()’.

El proceso se repite hasta obtener la cantidad de solapamientos deseada para las cantidades de cromosomas configuradas. Éstos se guardan mediante la función ‘guardar()’.

Referencia del archivo generar_params.py

Script que define los parámetros que se utilizan para la generación de los datos.

Descripción detallada

Script que define los parámetros que se utilizan para la generación de los datos.

En la siguiente tabla se detallan los parámetros, indicando los que trae por defecto que corresponden al dataset Gen5todosKaryoNormVar.

Parámetro	Detalle	Valor
directorio	Carpeta en que están los cromosomas individuales.	“./Separados/raw/”
out_path	Directorio de salida de los datos generados.	“./salida/”
total_cario	Cantidad total de cariogramas.	536
cuantos_raw	Cantidad deseada de archivos de salida.	12
intervalo	Cantidad de cariogramas que se utilizan para generar cada archivo.	40
tamImagen	Tupla que indica el tamaño de las imágenes genradas.	(256,256)
cant_angles	Cantidad de rotaciones que se realizan para cada solapamiento.	4
CantDespl	Cantidad de desplazamientos que se realizan para cada rotación.	1
train_cant	Cantidad deseada de solapamientos generados por cariograma y por cantidad de cromosomas para train.	10
test_cant	Similar al anterior para test.	10
cant_crom_min	Cantidad mínima de cromosomas por solapamiento.	2
cant_crom_max	Cantidad máxima de cromosomas por solapamiento.	5
porcMinSolap	Porcentaje mínimo de solapamiento respecto al cromosoma agregado.	0,01
porcMaxSolap	Porcentaje máximo de solapamiento medido respecto al nuevo cromosoma que se agrega.	0,27
minNewEP	Lista en la que se elige al azar la cantidad mínima de end points nuevos por solapamiento.	[1,2]
maxIntentos	Cantidad máxima de intentos para agregar un nuevo cromosoma antes de cambiarlo por otro.	20
maxIntentosAux	Máximo de intentos para agregar un cromosoma con ‘generarSolap()’.	20
minStd	Desvío mínimo para la aplicación de ‘histGauss()’.	50
maxStd	Desvío máximo para la aplicación de ‘histGauss()’.	70
minBlur	Desvío mínimo para la aplicación del filtro gaussiano de 3x3.	0
maxBlur	Desvío máximo para la aplicación del filtro gaussiano de 3x3.	0,7
cant_clases	Cantidad de clases contando la correspondiente al fondo.	24

Referencia del paquete generarAux

Archivo que incluye funciones auxiliares utilizadas por 'generar.py' tales como rotar, recortar, agregar cromosoma, guardar imágenes, etc.

Funciones

def **rotar** (img, mascara, angle, maxTam=(0, 0))

Dada una imagen y su máscara, los rota una cantidad de ángulos con la función 'rotate()' de scikit-image, rellenando con blanco los píxeles extra de la imagen y con negro los de las máscaras.

def **desplazarResize** (img, mascara, tamImagenSalida, despl=-1, fondo=255)

Función que re-dimensiona una imagen y su máscara, dando la posibilidad de desplazar la original dentro de la nueva más grande.

def **cantEP** (mask)

Función que calcula la cantidad de end points que tiene una imagen binaria.

def **suavizarPegada** (data1, mask1, data2, mask2, sigmaGauss=1.5)

Adiciona dos clusters de cromosomas haciendo que los píxeles de borde queden suaves.

def **generarSolap** (img1, mascara1, img2, porcMinSolap, porcMaxSolap, minNewEP, maxIntentos=30)

Agrega a un solapamiento 'img1' otro cromosoma 'img2', quedando 'img2' por detrás de 'img1'.

def **guardar** (salida, tamImagen, nombre, folder="./")

Función que se encarga de guardar las imágenes generadas como archivo ".npz".

def **recortar** (img, mascaras, margen, fondo=255)

Función que dada una imagen y su máscara o lista de máscaras las recorta al mínimo cuadrado que la contiene dejando un margen dado por 'margen'.

def **histGauss** (source, sigma=60)

Función que aplica histogram matching con una gaussiana de media 128 y desvío indicado.

Documentación de las funciones

def generarAux.cantEP (*mask*)

Función que calcula la cantidad de end points que tiene una imagen binaria.

Se utiliza la función 'thinning()' de OpenCV para el cálculo del esqueleto morfológico, para luego convolucionar con un filtro de 3x3 con todos números 1 y un 10 en el centro. Entonces, se cuentan los píxeles que resulten con un valor de 11: quiere decir que tienen sólo un vecino.

Parámetros:

mask	Imagen binaria de dos dimensiones en formato arreglo de NumPy.

Devuelve:

Entero que indica la cantidad de end points de la máscara.

def generarAux.desplazarResize (*img*, *mascara*, *tamImagenSalida*, *despl*= `-1`, *fondo*= `255`)

Función que re-dimensiona una imagen y su máscara, dando la posibilidad de desplazar la original dentro de la nueva más grande.

Primero se genera una imagen con color de fondo igual a 'fondo' y máscaras de fondo negro. Luego, según 'despl' se genera desplazamientos al azar (-1), se centra (-2) o se aplica (si fuera una tupla que indique el desplazamiento en x y en y). Finalmente se aplica dicho desplazamiento.

Parámetros:

img	Imagen 2D en formato arreglo de NumPy.
mascara	Imagen binaria 2D en formato arreglo de NumPy o lista que contenga imágenes de ese tipo, ya que pueden ser las máscaras de los cromosomas que ya se agregaron.
tamImagenSalida	Tupla que indica el tamaño que debe tener la imagen de salida.

despl	Tupla que indica el desplazamiento en x y en y. Si es un entero igual a -1, se calcula un desplazamiento al azar que no supere el tamaño de la imagen. Si es -2, la imagen original se centra dentro el nuevo lienzo.
fondo	Color de fondo con el que se rellena la imagen redimensionada.

Devuelve:

Tupla con imagen y máscara/lista de máscaras 2D en formato de arreglo de NumPy.

def generarAux.generarSolap (*img1*, *mascara1*, *img2*, *porcMinSolap*, *porcMaxSolap*, *minNewEP*, *maxIntentos* = `30`)

Agrega a un solapamiento 'img1' otro cromosoma 'img2', quedando 'img2' por detrás de 'img1'.

Primero se obtiene la máscara del cromosoma a agregar y se rellenan los agujeros con la función 'rellenoAgujeros()' del módulo de 'preprocesamiento'. Luego, se itera rotando 'img2' un ángulo al azar y agregándolo al solapamiento hasta que cumpla con las condiciones de porcentaje de solapamiento y cantidad de end points nuevos. Si se supera el máximo de intentos, se devuelve una máscara con todos 255 indicando que se falló y que se debe intentar agregar otro cromosoma.

Parámetros:

img1	Imagen 2D en formato arreglo de NumPy correspondiente al solapamiento actual.
mask1	Imagen binaria 2D en formato arreglo de NumPy.
img2	Imagen 2D en formato NumPy correspondiente al cromosoma que se agregará.
porcMinSolap	Porcentaje mínimo de solapamiento que debe tener el nuevo cluster medido respecto al cromosoma que se agrega.
porcMaxSolap	Porcentaje máximo de solapamiento del cromosoma agregado.
minNewEP	Cantidad mínima de end points que debe tener el nuevo cluster.
maxIntentos	Cantidad máxima de intentos que se prueba agregar el nuevo cromosoma.

Devuelve:

Tupla que contiene la imagen generada y una lista con las máscaras de cada cromosoma. Si se supera la cantidad máxima de intentos, la última máscara de la lista es una imagen con todos los píxeles 255.

def generarAux.guardar (*salida*, *tamImagen*, *nombre*, *folder* = `"./"`)

Función que se encarga de guardar las imágenes generadas como archivo ".npz".

Primero se definen las matrices de NumPy en las que se almacenan las imágenes generadas y las máscaras. Éstas tienen 4 dimensiones tal como se utilizan en PyTorch posteriormente: la primera es la cantidad de datos, la segunda la cantidad de canales (siempre 1 en este caso) y las restantes el tamaño de la imagen. Luego se recorren las listas de 'salida' para ir completándolas.

Por último se guarda la inversa de las imágenes mediante la función 'savez_compressed()' de NumPy puesto que la misma es eficiente comprimiendo matrices sparse. Los datos se guardan bajo la clave ‘data’ y los máscaras bajo la clave ‘maskLineal’.

Parámetros:

salida	Tupla que contiene tres listas: una con las imágenes de los solapamientos generados, otra que es a su vez una lista con las máscaras correspondientes a cada cromosoma y una tercera que es una lista que contiene el número de clase del mismo.
tamImagen	Tupla con el tamaño de la imagen que se guarda.
nombre	Nombre del archivo '.npz' que se genera.
folder	Carpeta de salida en que se guarda el archivo generado.

def generarAux.histGauss (*source*, *sigma* = `60`)

Función que aplica histogram matching con una gaussiana de media 128 y desvío indicado.

Parámetros:

source	Imagen o píxeles a los que se aplicará.
sigma	Desvío de la gaussiana que se utilizará.

Devuelve:

Imagen o píxeles corregidos según la gaussiana.

def generarAux.recortar (*img*, *mascaras*, *margen*, *fondo* = `255`)

Función que dada una imagen y su máscara o lista de máscaras las recorta al mínimo cuadrado que la contiene dejando un margen dado por 'margen'.

Se comienza buscando el mínimo rectángulo que contiene a los cromosomas de la imagen, sabiendo el color de fondo de la misma, con la función 'boundingRect' de OpenCV. Se genera la imagen de salida con color 'fondo' y se copia la información relevande de la imagen original. También se generan máscaras de salida con el mismo rectángulo obtenido anteriormente.

Parámetros:

img	Imagen 2D en formato arreglo de NumPy.
mascaras	Imagen binaria 2D en formato arreglo de NumPy o lista que contenga imágenes de ese tipo, ya que pueden ser las máscaras de los cromosomas que se fueron agregando.
margen	Entero que indica el margen que se deja a cada lado del mínimo rectángulo.
fondo	Color de fondo que se utiliza en la búsqueda del mínimo rectángulo.

Devuelve:

Tupla con imagen y máscara/lista de máscaras 2D en formato de arreglo de NumPy.

def generarAux.rotar (*img*, *mascara*, *angle*, *maxTam* = `(0,0)`)

Dada una imagen y su máscara, los rota una cantidad de ángulos con la función 'rotate()' de scikit-image, rellenando con blanco los píxeles extra de la imagen y con negro los de las máscaras.

Luego aplica una apertura morfológica en las máscaras ya que la rotación provoca una interpolación en los bordes de la imagen que produce una difuminación del mismo. Además, estas máscaras se aplican en la imagen para evitar dicha interpolación. Por último se recorta la imagen con '**recortar()**' porque la rotación suele agrandar la imagen. Antes de devolverla, se controla que al rotar la imagen no supere el tamaño máximo especificado (si 'maxTam' no es (0,0)).

Parámetros:

img	Imagen 2D en formato arreglo de NumPy.
mascara	Imagen binaria 2D en formato arreglo de NumPy o lista que contenga imágenes de ese tipo, ya que pueden ser las máscaras de los cromosomas que se fueron agregando.
angle	Ángulo que se giran las imágenes.
maxTam	Tupla que indica el tamaño máximo que puede tener la imagen. Si es (0,0) no se controla.

Devuelve:

Tupla con imagen y máscara/lista de máscaras 2D en formato de arreglo de NumPy. Devuelve -1 si la imagen rotada es mayor a 'maxTam'.

def generarAux.suavizarPegada (*data1*, *mask1*, *data2*, *mask2*, *sigmaGauss* = `1.5`)

Adiciona dos clusters de cromosomas haciendo que los píxeles de borde queden suaves.

Para ello, primero se obtiene el contorno de la zona de solapamiento mediante operaciones morfológicas de OpenCV. Luego se suman las dos imágenes de cromosomas 'data1' y 'data2' normalmente. A continuación, se le aplica un filtro gaussiano con desvío igual a 'sigmaGauss' en el borde obtenido anteriormente.

Parámetros:

data1	Imagen 2D en formato arreglo de NumPy.
mask1	Imagen binaria 2D en formato arreglo de NumPy.
data2	Imagen 2D en formato arreglo de NumPy, con la zona de solapamiento ya quitada previamente de forma de permitir la suma con 'data1' para generar el solapamiento.
mask2	Imagen binaria 2D en formato arreglo de NumPy, con la zona de solapamiento ya quitada previamente.
sigmagauss	Desvío utilizado en el filtro gaussiano de 3x3 para la suavización del borde de la zona de solapamiento.

Devuelve:

Tupla con la imagen y una lista con las máscaras 2D en formato de arreglo de NumPy.

Referencia del paquete separarKaryo

Archivo que define y utiliza las funciones para extraer los cromosomas de los cariogramas para la posterior generación de datos.

Funciones

def **compare** (c1, c2)

Dada dos listas de contornos, compara cuál de las dos está más arriba y a la izquierda tomando como referecia el punto más inferior y a la derecha de cada componente conexa.

def **cmp_to_key** (mycmp)

Función que transforma la función de comparación en una 'key' aceptada por la función 'sorted' que viene por defecto en Python.

def **dividirCariograma** (img, minTam=200)

Función que dada una imagen que es un cariograma con fondo blanco, guarda una subimagen de cada cromosoma en salida.

def **guardarSeparados** (separados, dirSalida)

Función que se encarga de guardar los cromosomas extraídos en la carpeta de salida.

def **procesarCariogramas** (directorio, cuantos, salida)

Función que toma tantos cariogramas como se le indique para extraer los cromosomas individuales.

Descripción detallada

Archivo que define y utiliza las funciones para extraer los cromosomas de los cariogramas para la posterior generación de datos.

En la última línea del archivo contiene el llamado a la función ‘procesarCariogramas()’ utilizado para extraer los cromosomas de todos los cariogramas que poseen 46 cariogramas.

Documentación de las funciones

def separarKaryo.cmp_to_key (*mycmp*)

Función que transforma la función de comparación en una 'key' aceptada por la función 'sorted' que viene por defecto en Python.

Parámetros:

mycmp	Función de comparación.

Devuelve:

Clase compatible con el parámetro 'key' de 'sorted'.

def separarKaryo.compare (*c1*, *c2*)

Dada dos listas de contornos, compara cuál de las dos está más arriba y a la izquierda tomando como referencia el punto más inferior y a la derecha de cada componente conexa.

Se utiliza para ordenar espacialmente los contornos obtenidos del cariograma de arriba hacia abajo y de izquierda a derecha con la función 'sorted' que viene integrada en Python.

Parámetros:

c1	Lista de puntos obtenidos con la función 'findContours()' de OpenCV: es un arreglo de enteros de 3 dimensiones.
c2	Lista de puntos obtenidos con la función 'findContours()' de OpenCV: es un arreglo de enteros de 3 dimensiones.

Devuelve:

Un entero que es 0 si son iguales, mayor a 0 si 'c1' es mayor y menor a 0 si 'c2' es mayor.

def separarKaryo.dividirCariograma (*img*, *minTam* = 200)

Función que dada una imagen que es un cariograma con fondo blanco, guarda una sub-imagen de cada cromosoma en salida.

Les agrega un número incremental para evitar repetidos y la extensión '.tiff'. Además las ordena de acuerdo a su posición espacial, empezando por la esquina superior izquierda y continuando hacia la derecha. Para ello, aplica un umbral de 254 sabiendo que el fondo de los mismos es blanco. A la máscara obtenida, le rellena los agujeros con la función 'rellenoAgujeros()' del módulo de 'preprocesamiento' de la herramienta de segmentación automática de cromosomas y se detectan las componentes conexas con 'findContours()' de OpenCV. Para cada componente conexa, si es mayor al tamaño indicado 'minTam', la imagen se agrega a la lista que se devuelve posteriormente.

Parámetros:

img	Cadena de texto que indica el directorio del cariograma.

Devuelve:

Lista de imágenes que contienen los cromosomas extraídos del cariograma.

def separarKaryo.guardarSeparados (*separados*, *dirSalida*)

Función que se encarga de guardar los cromosomas extraídos en la carpeta de salida.

Se guardan de la forma "cco.png", donde 'cc' es la clase del cromosoma y 'o' es igual a 'a' o a 'b', para guardar las dos ocurrencias de cada clase en el cariograma. Se utiliza la función 'imwrite()' de OpenCV.

Parámetros:

separados	Lista de imágenes de dos dimensiones con formato arreglo de NumPy.
dirSalida	Cadena de texto que indica la carpeta de salida.

def separarKaryo.procesarCariogramas (*directorio*, *cuantos*, *salida*)

Función que toma tantos cariogramas como se le indique para extraer los cromosomas individuales.

Primero se obtienen todos los archivos del directorio pasado como parámetro y luego se van leyendo una a una en un bucle. Para cada una se crea una carpeta dentro del directorio de salida con nombre "rawN", donde N es el número de cariograma a procesar. Luego se extraen los cromosomas con '**dividirCariograma()**' y se guardan con '**guardarSeparados()**' si es que tienen la cantidad de cromosomas indicada en 'cuantos'. Además, en la carpeta de salida se guarda el cariograma original para saber de donde provienen los cromosomas extraídos.

Parámetros:

directorio	Cadena de texto que indica la carpeta en la que se encuentran los cariogramas.
cuantos	Cantidad de cromosomas que debe tener el cariograma para procesarlos. Usado para obtener sólo los que poseían 46 cromosomas.
salida	Cadena de texto que indica la carpeta de salida.