

EL5206-Laboratorio de Inteligencia Computacional y Robótica  
Área Procesamiento de Imágenes

## Informe Proyecto Final

**Alumno:** Tomás Lara A.  
**Profesor:** Claudio Pérez.  
**Profesor Auxiliar:** Juan Pérez.  
**Fecha Entrega:** 29/12/2018

## Índice

<b>Lista de figuras</b>	<b>1</b>
<b>1. Introducción y descripción del problema.</b>	<b>3</b>
<b>2. Desarrollo.</b>	<b>4</b>
2.1. Cálculo de vector de características . . . . .	4
2.1.1. Transformación a formato HSV . . . . .	4
2.1.2. División de imagen . . . . .	4
2.1.3. Generación de histogramas normalizados. . . . .	5
2.2. Medida de similitud . . . . .	6
2.3. Orden de resultados según relevancia. . . . .	7
2.3.1. Generación de la base de datos. . . . .	7
2.3.2. Cálculo de comparaciones . . . . .	8
2.4. Evaluación de los métodos . . . . .	8
2.4.1. Rank . . . . .	8
2.4.2. Rank Normalizado . . . . .	9
2.5. Implementación de metodología alternativa . . . . .	9
2.5.1. Descripción de métodos . . . . .	9
2.5.2. Método escogido e implementación . . . . .	10
2.6. Implementación de integración de sistemas. . . . .	11
<b>3. Resultados.</b>	<b>13</b>
3.1. Métodos originales . . . . .	13
3.2. Método propuesto: HOG . . . . .	16
3.3. Resultados utilizando IRP. . . . .	18
<b>4. Conclusión</b>	<b>21</b>
<b>Bibliografía</b>	<b>22</b>

## Lista de figuras

1.1. Esquema de búsqueda de CBIR(1) . . . . .	3
2.1. División no rectangular de la imagen . . . . .	5
2.2. Metodología de construcción de histogramas paraHOG . . . . .	11
3.1. Imágenes sugeridas, 5 divisiones . . . . .	14
3.2. Imágenes sugeridas, división 3x3 . . . . .	14
3.3. Imágenes sugeridas, división 4x4 . . . . .	14
3.4. Imágenes sugeridas, división 6x6 . . . . .	14
3.5. Imágenes sugeridas, query1 . . . . .	15
3.6. Imágenes sugeridas, query2 . . . . .	15
3.7. Imágenes sugeridas, query3 . . . . .	16
3.8. Imágenes sugeridas, query11 . . . . .	16
3.9. Imágenes sugeridas, query1, método HOG . . . . .	17
3.10. Imágenes sugeridas, query2, método HOG . . . . .	17
3.11. Imágenes sugeridas, query3, método HOG . . . . .	17
3.12. Imágenes sugeridas, query11, método HOG . . . . .	17
3.13. Imágenes sugeridas, query53, método HOG . . . . .	18
3.14. Imágenes sugeridas, query1, método IRP . . . . .	19
3.15. Imágenes sugeridas, query2, método IRP . . . . .	19
3.16. Imágenes sugeridas, query3, método IRP . . . . .	20
3.17. Imágenes sugeridas, query11, método IRP . . . . .	20
3.18. Imágenes sugeridas, query3, método IRP . . . . .	20
3.19. Imágenes sugeridas, query11, método IRP . . . . .	20

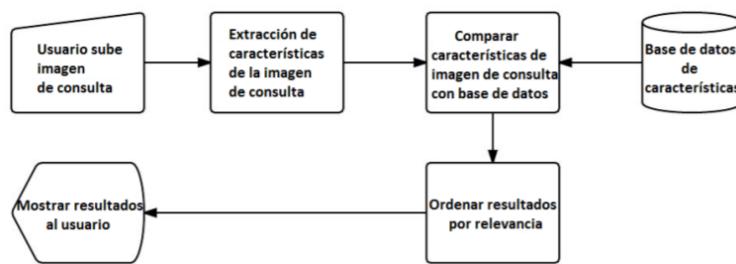
## Listado de Códigos

1.	Transformación de imagen a formato HSV . . . . .	4
2.	Implementación de función que divide imagen en 4x4 celdas . . . . .	4
3.	Función que calcula el vector de características para una imagen. . . . .	6
4.	Función que calcula la distancia euclídea entre dos vectores de características . . . . .	6
5.	Función que genera la base de datos asociada a una metodología . . . . .	7
6.	Función que calcula la distancia de una imagen Query a todas las figuras de la base de datos. . . . .	8
7.	Función que calcula el vector de características para una imagen con el método HOG . . . . .	11
8.	Función valoresIRPIndividual . . . . .	12

## 1. Introducción y descripción del problema.

El presente informe, enmarcado en el contexto del curso EL-5206/Laboratorio de Inteligencia Computacional y Robótica busca dar cuenta del desarrollo y resultados obtenidos a partir del trabajo sobre el proyecto final del curso, donde el área escogida corresponde al área de procesamiento de imágenes, tercera unidad del curso.

El problema a abordar corresponde al desarrollo e implementación de un algoritmos de *Content Based Image Retrieval (CBIR)*, utilizando diferentes metodologías para la extracción de características de la imagen. Esta metodología consiste en realizar una búsqueda de imágenes similares en una base de datos a partir de una consulta realizada mediante una imagen (1). El sistema se basa en el esquema presentado en la figura 1.1. En esta metodología, se realiza en primer lugar la construcción de una base de datos de características, para esto, se realiza la extracción de características de un conjunto de imágenes pertenecientes a una base de datos. Posteriormente, cuando un usuario realiza una consulta al sistema, las características de la imagen que ingresa son extraídas utilizando el mismo método para posteriormente realizar una comparación entre el vector de características de la imagen ingresada y todos los vectores de características de la base de dato, con el fin de obtener valores de comparación entre los elementos. Utilizando los valores de comparación es posible ordenar los resultados por cercanía a la figura original y de esta forma presenta al usuario los mejores resultados en términos de similitud.



**Figura 1.1:** Esquema de búsqueda de CBIR(1)

Utilizando esta metodología se realizarán pruebas utilizando diferentes métodos de selección de características, con el fin de comparar mediante una métrica definida.

El presente informe se organiza de la siguiente manera. En primer lugar, la presente sección plantea el problema a resolver y la forma general de la metodología a utilizar. Posteriormente, en la sección desarrollo se presentará un resumen de las funciones y actividades realizadas en pos de el establecimiento del sistema, donde la sección se organiza en base a actividades solicitadas en el enunciado. A continuación se presentan los principales resultados obtenidos a partir de las metodologías de CBIR, presentando resultados tanto cuantitativos (métricas), como cualitativos (imágenes presentadas). Finalmente se resumen las principales conclusiones obtenidas a partir del desarrollo del proyecto final.

## 2. Desarrollo.

---

## 2. Desarrollo.

En la presente sección se presenta el desarrollo realizado con el fin de implementar las funcionalidades solicitadas en los diferentes puntos del enunciado de la tarea.

### 2.1. Cálculo de vector de características.

#### 2.1.1. Transformación a formato HSV

En primer lugar, se solicita transformar la imagen desde formato RGB (*Red, Green, Blue*, formato normal de importación de imágenes) a HSV (*Hue,Saturation,Value*), con el fin de utilizar este formato en trabajo posterior. Para esta actividad es posible utilizar el método `cvtColor` existente en *openCV*. El proceso se implementa en la función `rgb2HSV` donde se recibe el nombre de la imagen y se retorna la imagen en formato HSV, correspondiente a una matriz de tamaño de la imagen para cada uno de los canales. La implementación se presenta en el código 1.

```

1 #Funcion que transforme del espacio RGB al HSV
2 def rgb2HSV (nombreImagen):
3     #Lectura
4     readImage = cv2.imread(nombreImagen)
5     #Transformacion de espacio. Devuelve una matriz 3d
6     imageHSV = cv2.cvtColor(readImage, cv2.COLOR_BGR2HSV)
7     return imageHSV

```

**Código 1:** Transformación de imagen a formato HSV

#### 2.1.2. División de imagen

Para la división de imagen, en el caso rectangular se desarrollan tres funciones, correspondientes a `div9Celdas`, `div16Celdas` y `div36Celdas`. Estas funciones, realizan la lectura de la imagen y dividen esta en 9, 16 y 36 zonas, respectivamente. Como salida, se entregan 9,16 y 36 matrices para cada uno de los canales (H,S,V), con el tamaño de cada una de las secciones recortadas. En este caso, se presenta la implementación de la función `div16Celdas` (código 2), donde el desarrollo de las restantes dos funciones es análogo a este.

```

1 #Funcion que recorte la imagen en 4x4 celdas. Devuelve 16 submatrices.
2 def div16Celdas(imHSV):
3     H = imHSV[:, :, 0]
4     S = imHSV[:, :, 1]
5     V = imHSV[:, :, 2]
6     #Dimensiones de la imagen
7     alto, ancho, prof = imHSV.shape
8     #Nuevas dimensiones
9     nuevoAlto = int(alto/4)
10    nuevoAncho = int(ancho/4)
11    #Creacion de las tres matrices (inicialmente vacias).
12    Hdiv = np.zeros([nuevoAlto, nuevoAncho, 16])
13    Sdiv = np.zeros([nuevoAlto, nuevoAncho, 16])
14    Vdiv = np.zeros([nuevoAlto, nuevoAncho, 16])
15    #Dividir imagen

```

## 2. Desarrollo.

```

16     vecDivisionesAlto = [0,nuevoAlto,nuevoAlto,nuevoAlto*2,nuevoAlto*2,nuevoAlto*3,nuevoAlto*3,
17         nuevoAlto*4]
18     vecDivisionesAncho = [0,nuevoAncho,nuevoAncho,nuevoAncho*2,nuevoAncho*2,nuevoAncho*3,
19         nuevoAncho*3,nuevoAncho*4]
20     count = 0
21     count1 = 0
22     while count1 <= 7:
23         count2 = 0
24         while count2 <= 7:
25             Hdiv[:, :, count] = H[vecDivisionesAlto[count1]:vecDivisionesAlto[count1+1],
26             vecDivisionesAncho[count2]:vecDivisionesAncho[count2+1]]
27             Sdiv[:, :, count] = S[vecDivisionesAlto[count1]:vecDivisionesAlto[count1+1],
28             vecDivisionesAncho[count2]:vecDivisionesAncho[count2+1]]
29             Vdiv[:, :, count] = V[vecDivisionesAlto[count1]:vecDivisionesAlto[count1+1],
30             vecDivisionesAncho[count2]:vecDivisionesAncho[count2+1]]
31             count += 1
32             count2 += 2
33             count1 +=2
34
35     return np.uint8(Hdiv),np.uint8(Sdiv),np.uint8(Vdiv)

```

**Código 2:** Implementación de función que divide imagen en 4x4 celdas

Para el caso del recorte irregular en secciones no rectangulares, presentado en la figura 2.1, se generan las funciones `cuatroSectores` y `ellipseCentral`, la primera de las cuales genera las divisiones 1,2,3 y 4, mientras que la segunda genera la división de forma tal de generar el elipse central. Cabe notar, que si bien se desarrollan dos funciones para el recorte irregular, estas no son utilizadas en el cálculo de histogramas, donde se prefiere el uso de máscaras, dado que permite un correcto desarrollo del cálculo.



**Figura 2.1:** División no rectangular de la imagen

### 2.1.3. Generación de histogramas normalizados.

A continuación, deben calcularse los histogramas los histogramas normalizados para cada canal y normalizarlos, para posteriormente concatenar estos, dando origen a un vector de dimensión 23 para cada división de la imagen, donde se utilizaron 8 bins para el canal H, 12 bins para el canal S y 3 bins para el canal V, según indicaciones de enunciado. Luego de calculado este vector para cada subdivisión de la imagen, se concatenan los vectores de cada una de estas, originando el vector de características asociado a la imagen.

Las funciones asociadas a esta sección corresponden a las siguientes:

## 2. Desarrollo.

- **normHist:** Realiza la normalización de histogramas, la normalización se realiza dividiendo el histograma en la suma de sus componentes.
- **histRects:** Realiza el cálculo del vector de características para las divisiones rectangulares de la imagen. Esto se realiza de acuerdo al número de divisiones indicado, donde dependiendo de esta se utiliza uno de los métodos de recorte rectangular presentado en la sección anterior.
- **histNoCuadrado:** Realiza el cálculo del vector de características para la división no rectangular de la imagen. Esto lo realiza utilizando máscaras, las cuales son definidas mediante la función `defMask` dependiendo de la sección de imagen cuyo histograma se pretenda obtener, donde las secciones de imagen se identifican mediante la nomenclatura presentada en la figura 2.1. Utilizando esta máscara las funciones `histRedondo` y `histOtrosRedondo` calculan los histogramas asociados a la zona 5 (figura 2.1) y las restantes zonas, respectivamente. Posterior a este cálculo, se concatenan todos los histogramas para generar el vector de características.
- **histSectores:** Realiza el cálculo de histograma para una imagen utilizando la función correspondiente al número de divisiones indicado. Su implementación se presenta en el código 3

```

1 #Funcion que calcule con el nombre de la imagen.
2 def histSectores(nombreImagen,nDivisiones):
3     #Transformacion a HSV
4     imHSV = rgb2HSV(nombreImagen)
5     if(nDivisiones==5):
6         out = histNoCuadrado(imHSV)
7     else:
8         out = histRects(imHSV,nDivisiones)
9     return np.ravel(out)

```

**Código 3:** Función que calcula el vector de características para una imagen.

## 2.2. Medida de similitud

Con el fin de implementar una medida de similitud entre vectores de características, se escoge el uso de la distancia euclíadiana para realizar la comparación, de esta manera mientras más parecidos son los histogramas, menor es la distancia entre los vectores de características, y por tanto, se esperaría que las imágenes fuesen parecidas. La distancia euclíadiana se calcula de la manera presentada en la ecuación 2.1

$$dist(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\|_2 \quad (2.1)$$

La implementación realizada para el cálculo de la distancia euclíadiana entre dos imágenes se presenta en el código 4. Donde se hace uso de la función `euclidean` implementada en `scipy`.

```

1 def compareEuc(hist1,hist2):
2     #Comparar.
3     distancia = dist.euclidean(hist1, hist2,1)
4     return distancia

```

**Código 4:** Función que calcula la distancia euclíadiana entre dos vectores de características

## 2. Desarrollo.

Se realizan pruebas sobre imágenes de la misma clase, imágenes notoriamente diferentes y la misma imagen, con el fin de observar el comportamiento de la función sobre los histogramas, donde fue posible notar que imágenes similares presentan una menor distancia (cero en el caso en que se compara con la misma imagen) que en comparación con imágenes notoriamente diferentes.

### 2.3. Orden de resultados según relevancia.

#### 2.3.1. Generación de la base de datos.

Con el fin de realizar la búsqueda de imágenes similares a una imagen *query* consultada, se debe generar en primer lugar la base de datos, correspondiente a un conjunto de vectores de características asociados a figuras presentes en la carpeta *img\_database*. Con este fin se generan las funciones *incorporarDataBaseHist*, esta función realiza el cálculo de características para cada una de las imágenes de la carpeta mencionada, dado un número de divisiones, parámetro que define la metodología de extracción de características. Como salida, la función entrega los nombres de las imágenes, sus clases correspondientes, extraídas mediante la función *extraerClase* y la matriz de histogramas, donde cada fila corresponde al vector de características asociado a cada imagen. La función *incorporarDataBaseHist* se presenta en el código 5

```

1 def incorporarDataBaseHist(nDivisiones):
2     #Función que calcule los histogramas del Database.
3     # Inicializa el diccionario index, para almacenar el nombre de imagen con su histograma
4     path = "img_database/*.jpg"
5     #Creacion de arreglos
6     nombres = np.array([])
7     clases = np.array([])
8     histos = np.array([])
9     count = 0
10    #Loop.
11    for imPath in glob.glob(path):
12        #Nombre del archivo
13        filename = imPath[imPath.rfind("\\")+1:]
14        #Agregar nombre y clase
15        nombres = np.append(nombres,str(filename))
16        clases = np.append(clases,extraerClase(filename))
17        #Obtener el histograma
18        histImagen = histSectores(imPath,nDivisiones)
19        #Incluir histograma
20        if (count==0):
21            histos=histImagen.T
22            count = 1
23        elif(count==1):
24            histos = np.concatenate(([histos],[histImagen.T]),axis=0)
25            count = 2
26        else:
27            histos = np.concatenate((histos,np.array([histImagen])),axis=0)
28    return nombres,clases,histos

```

**Código 5:** Función que genera la base de datos asociada a una metodología

## 2. Desarrollo.

### 2.3.2. Cálculo de comparaciones

Para el cálculo de las comparaciones, es decir, el cálculo de la distancia entre el vector de características de una imagen de la carpeta *img\_query* con respecto a todos los vectores de características existentes en la base de datos se hace uso de dos funciones. En primer lugar, se utiliza la función *distanciaUnaImagen*, esta función calcula la distancia del vector de características de una imagen, con respecto a la matriz de vectores de características de todas las imágenes de la base de datos, donde esta matriz esta definida en base al número de divisiones en que se desea dividir la imagen. La implementación de esta función se presenta en el código 6

```

1 def distanciaUnaImagen(nombreImagenQuery,divisiones):
2     #Obtener histograma de la imagen query.
3     histQuery = histSectores(nombreImagenQuery,divisiones)
4     #Seleccion de la matriz de histogramas a comparar.
5     if (divisiones==5):
6         histComparar = histos5
7     elif (divisiones==9):
8         histComparar = histos9
9     elif (divisiones==16):
10        histComparar = histos16
11    elif (divisiones==36):
12        histComparar = histos36
13    #Comparaciones:
14    comparaciones = np.array([])
15    for h in histComparar:
16        distancia = compareEuc(h,histQuery)
17        comparaciones = np.append(comparaciones,distancia)
18    return comparaciones

```

**Código 6:** Función que calcula la distancia de una imagen Query a todas las figuras de la base de datos.

Posteriormente, la función *compararQueries*, toma la función presentada en 6 para calcular las distancias asociadas a todas las imágenes de la query, obteniendo de esta forma una matriz de distancias donde cada fila corresponde a los valores de distancia asociados a cada imagen de la query con respecto a la base de datos, por lo que el número de columnas corresponde al número de imágenes en la base de datos. La implementación se adjunta en el código, no se adjunta a este informe dado la similitud de implementación que presenta con la función 5, con la diferencia en el uso de la función *distanciaUnaImagen*.

### 2.4. Evaluación de los métodos

Para la evaluación de los distintos métodos utilizados para la resolución del problema de CBIR, asociados a las diferencias de metodología en la selección de características se utilizan dos métricas las cuales se presentan, con su explicación, en las siguientes subsecciones.

#### 2.4.1. Rank

En primer lugar, se utiliza como métrica del desempeño, el valor de *Rank Promedio*. donde *Rank* se define en la ecuación 2.2. Siendo,  $N_{rel}$  el número de imágenes relevantes para la imagen buscada, definido como el número de imágenes en la base de datos que corresponden a la misma clase que la imagen que se esta buscando. Por otro lado, el término  $R_i$  corresponde a la posición que tienen esta imágenes en un arreglo ordenado de acuerdo al valor de distancia entre la imagen de *query* y las imágenes relevantes en la base de datos. Este

## 2. Desarrollo.

---

valor se calcula para cada imagen, luego, el Rank medio se calcula como el promedio de los valores de Rank obtenidos por las imágenes de *query*

$$Rank = \frac{1}{N_{rel}} \sum_{i=1}^{N_{rel}} R_i \quad (2.2)$$

Dada la definición de la métrica, algoritmos que presenten menores valores de Rank serán mejores, dado que las imágenes de la misma clase que la imagen buscado presentan en promedio menores posiciones, es decir, son consideradas más similares a la imagen buscada.

Para el cálculo de esta métrica se desarrollan las funciones `rankQuery` y `rankIndividual` para el cálculo de *Rank* de todas las imágenes de *query* e imágenes individuales respectivamente.

### 2.4.2. Rank Normalizado

Como métrica definitiva se realiza el cálculo del *Rank Normalizado*, este se define en la ecuación 2.3. en esta ecuación la variable *N* corresponde al tamaño de la base de datos. El Rank normalizado promedio se calcula como el promedio de los obtenidos para todas la imágenes de *query*.

$$\bar{Rank} = \frac{1}{N_{rel}N} \left( \sum_{i=1}^{N_{rel}} R_i - \frac{N_{rel}(N_{rel} + 1)}{2} \right) \quad (2.3)$$

Para el cálculo se utilizan las funciones `rankNormalizadoQuery` y `rankNormalizadoIndividual` para el cálculo de *Rank Normalizado* de todas las imágenes de *query* e imágenes individuales respectivamente.

## 2.5. Implementación de metodología alternativa

Posterior al desarrollo de la metodología utilizando hisogramas basados en la descomposición de colores HSV realizada anteriormente se plantea investigar que otros tipos de características son utilizadas para abordar el problema de CBIR y posteriormente implementar al menos un método alternativo a los ya implementados con el fin de compararlos a través de la métrica *Rank Normalizado*. En las secciones siguientes se explicarán brevemente metodologías alternativas para la resolución del problema, específicamente, para la realización de la extracción de características. Posteriormente, se realizará una explicación más detallada del método escogido, así como la implementación realizada para el funcionamiento de este.

### 2.5.1. Descripción de métodos

La extracción de características corresponde a un punto crucial en el desarrollo de un sistema de CBIR. Se realiza una breve investigación acerca de algunos métodos existentes, los cuales se describen brevemente en la presente sección, con información recopilada utilizando las fuentes (6) y (2). Existen diferentes metodologías para realizar la extracción de características de imágenes que deseen ser utilizadas en un sistema de CBIR, de manera general, estos pueden ser clasificados en características de color (como las indicadas en enunciado), textura y forma. Estos métodos se describen brevemente a continuación:

- **Características de Color:** Corresponden a métodos ampliamente utilizados, dentro de los métodos existentes se encuentra la extracción de histogramas de color, ya sea en formatos RGB o HSV, Momentos geométricos, donde estos se pueden definir como un promedio ponderado (de una manera a elección) de las intensidades de píxel, métodos más cercanos a las probabilidades como “momentos de color”, entre otros.

## 2. Desarrollo.

- **Características de textura:** Los métodos de textura se basan en propiedades de homogeneidad independientes del color a la intensidad de luz. Estos métodos centran su atención en la disposición de superficies en la imagen para generar diferentes vectores de características. Dentro de esta categoría existen diferentes metodologías, tales como el uso de filtro de Gabor, Transformadas Wavelet o de Fourier, entre otros.
- **Características de forma:** Si bien esta metodología no es tan usada en el contexto, es de utilidad cuando la forma del objeto corresponde a una característica determinante. Dentro de los métodos existentes en esta categoría se encuentran los descriptores de Fourier, el método DAISY, entre otros.

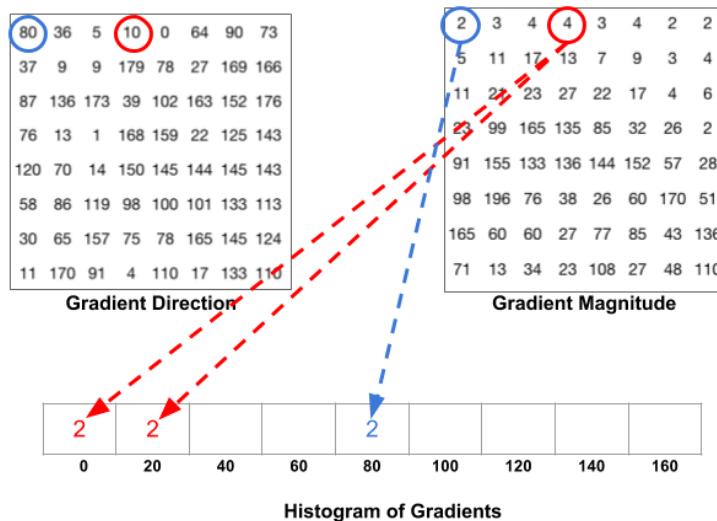
### 2.5.2. Método escogido e implementación

Se escoge como método alternativo a implementar el método de extracción de características a partir de histogramas de orientación de gradientes (*HOG, Histogram of Oriented Gradients*). La idea de utilizar esta metodología se obtiene a partir de la referencia (2), en la que se realiza una comparación de resultados obtenidos por diferentes metodologías de extracción de características. A partir de los resultados obtenidos se observa un buen desempeño de HOG, en algunos de los casos mostrados. Para el entendimiento del método se estudia la referencia (3), mientras que para la implementación final se utiliza la librería *scikit-image*, la que contiene una serie de métodos de extracción de características, dentro de los que se encuentra HOG (4).

La extracción de características mediante el método HOG (*Histogram of Oriented Gradients*) utiliza histogramas de direcciones de gradientes como características. Estos histogramas tienen asociados a cada uno de sus *bins* un número, relacionado con la magnitud de los gradientes que tengan una inclinación similar a la orientación asociada a los *bins*. Esta metodología es útil dado que la magnitud de los gradientes en la imagen es mayo en zonas en que se tienen bordes o esquinas, es decir, cambios abruptos, por lo que permite describir la imagen en relación con sus bordes y obtener una representación de “variaciones” de esta. El procedimiento general realizado para la extracción de características se presenta a continuación(3):

- se utiliza la imagen en su representación BGR, obtenida al importar con openCV.
- Para el caso desarrollado, puesto que al utilizar imágenes de distintos tamaños el número de bins del histograma final varía, se realiza un re escalamiento de la imagen, con el fin de que todas queden con un tamaño definido de 2000 por 2000.
- Se calculan las componentes de gradientes en los ejes x e y. Para esto se utiliza la función Sobel de openCV. A partir de esto se tiene una representación cartesiana de los gradientes existentes en la imagen.
- Se realiza el cambio a una representación polar de gradiente, de modo que este quede representado por una magnitud y un ángulo asociado.
- Se realiza el cálculo por celdas de la imagen de los histogramas asociados. Para la formación de los histogramas se realiza una asociación entre ángulo y magnitud de la forma presentada en la figura 2.2, donde, cada bin del histograma se asocia a un ángulo de gradiente, luego, si dentro de la matriz de ángulos de gradientes, se tiene una correspondencia con uno de los valores del histograma, el valor de magnitud asociado suma directamente a este bin, en cambio, si el valor de ángulo se encuentra entre los dos ángulos correspondientes a dos bin, la magnitud asociada se reparte equitativamente en los casilleros correspondientes.

## 2. Desarrollo.



**Figura 2.2:** Metodología de construcción de histogramas paraHOG

- Posterior a estos es posible realizar una concatenación de los histogramas para obtener el vector de características.

En la implementación en código la implementación de la extracción de características a través de HOG se obtiene a partir de la función `hog` del módulo `scikit-images` (4). La implementación de la función asociada al cálculo del vector de características se presenta en el código 7

```

1 #Funcion que calcule el vector de caracteristicas de una imagen
2 def caracteristicasHOG(imRGB):
3     #Resize de la imagen, para que funciones con todas las imagenes.
4     imagenResize = cv2.resize(imRGB, (2000, 2000))
5     #Calculo del vector de caracteristicas
6     vectorCar = hog(imagenResize, orientations = 9, pixels_per_cell = (100,100),
7                     cells_per_block = (1,1), block_norm = 'L2-Hys', visualize = False, feature_vector = True,
8                     multichannel = True)
9     return vectorCar

```

**Código 7:** Función que calcula el vector de características para una imagen con el método HOG

### 2.6. Implementación de integración de sistemas.

Con el fin de integrar los dos mejores algoritmos encontrados, correspondientes a la división en secciones no rectangulares y mediante el método HOG se utiliza el algoritmo IRP *Inverse Rank Position Algorithm*. Este algoritmo genera una lista única de similitudes a partir de varias (en este caso dos) listas de similitudes. De manera analítica el algoritmo IRP se expresa de la manera presentada en la ecuación 2.4 (5).

$$IRP(fig) = \frac{1}{\sum_{i=1}^{N_{listas}} \frac{1}{P_i(fig)}} \quad (2.4)$$

En la ecuación, el término  $P_i(fig)$  corresponde a la posición en la lista (ordenada) correspondiente a la técnica  $i$ .  $N_{listas}$  corresponde al número de listas a utilizar para IRP, en este caso se utilizarán dos, la

## 2. Desarrollo.

---

correspondiente a la metodología de 5 divisiones (no rectangulares) y la correspondiente al método HOG.

Las funciones desarrolladas para el cálculo del vector de orden de IRP se encuentran en el archivo *ProyectoImagenes(IRP)*. Dentro de las funciones existentes destacan *valoresIRPIndividual*, la que permite el cálculo del vector de valores a partir de los vectores de distancia obtenidos para los dos métodos, HOG y 5 divisiones. Otra función de gran importancia corresponde a *ordenarPorIRP*, esta función, permite realizar el cálculo de vector de IRP sobre todas las imágenes que componen la carpeta *img\_query*. Posteriormente, el mismo archivo contiene función para el cálculo de *Rank Normalizado Promedio* y la visualización de las imágenes, de manera análoga a la desarrollada para las actividades anteriores.

Dada su importancia se presenta el código desarrollado para la función *valoresIRPIndividual*, el cual se presenta en el código 8

```

1 def valoresIRPIndividual(vecComp1,vecComp2):
2     largoComp = len(vecComp1)
3     #Nombres de imagen
4     name1 = np.arange(largoComp) + 1
5     name2 = np.arange(largoComp) + 1
6     #Ordenar y obtener el orden de indices.
7     ind1 = ordenarIRP(name1,vecComp1)
8     ind2 = ordenarIRP(name2,vecComp2)
9     #Obtener los valores de EIRP asociados a cada imagen en orden.
10    valoresIRP = np.zeros_like(vecComp1)
11    count = 0
12    for i in name1:
13        pos1 = np.where(ind1 == i)[0][0] + 1
14        pos2 = np.where(ind2 == i)[0][0] + 1
15        valorIRP = 1/((1/pos1)+(1/pos2))
16        valoresIRP[count] = valorIRP
17        count += 1
18    return valoresIRP

```

**Código 8:** Función *valoresIRPIndividual*

### 3. Resultados.

#### 3.1. Métodos originales

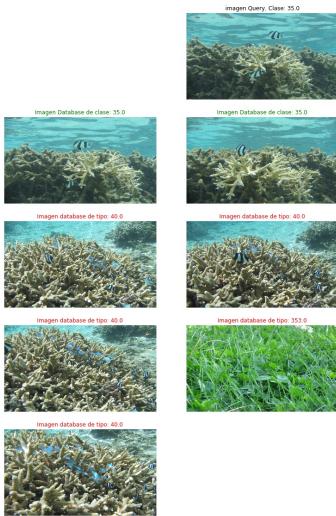
En la presente sección se obtienen los resultados utilizando los métodos de selección de características presentes en el enunciado del proyecto (1), correspondientes a la división en celdas de 3x3, 4x4 y 6x6 además de la división irregular presentada previamente. En primer lugar se presentan los valores de *Rank promedio* y *Rank Normalizado promedio*, con el fin de realizar una evaluación de los distintos métodos, los valores de las métricas se encuentran resumidos en la tabla 3.1. A partir de los resultados presentados en la tabla 3.1 es posible observar que el mecanismo de extracción de características mejor evaluado corresponde al que utiliza la división de la imagen en zonas no rectangulares, presentando un desempeño notoriamente mejor en la métrica *Rank Normalizado.*, por lo que este correspondería a un mejor método de extracción de características.

Método	Rank Promedio	Rank Normalizado Promedio
<b>3x3</b>	56.1812	0.4559
<b>4x4</b>	67.3145	0.4533
<b>6x6</b>	77.1546	0.4476
<b>celdas no rectangulares</b>	34.3224	0.0792

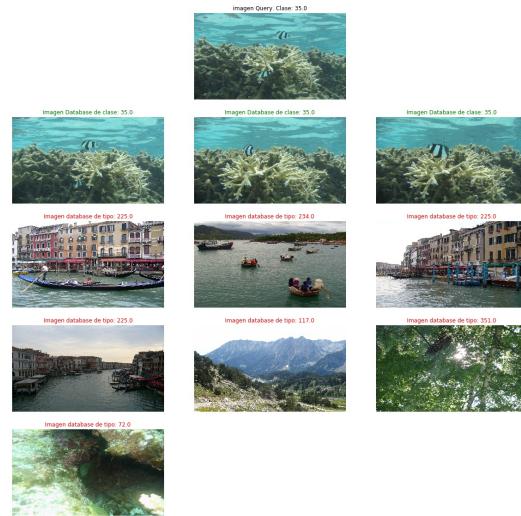
**Cuadro 3.1:** Métricas métodos de división indicados.

A continuación se presentan resultados cualitativos, en relación a la cercanía existente entre la imagen de query y las imágenes mostradas en pantalla como las más cercanas. En primer lugar, se presenta una comparación de métodos, es decir, se presentan las imágenes sugeridas utilizando las diferentes metodologías de extracción de características, para una imagen específica de *img\_query* en que se presentó en general una clasificación adecuada. En las figuras 3.1,3.2, 3.3 y 3.4. Se presentan los resultados obtenidos para la sugerencia de imágenes al utilizar como imagen de consulta la cuarta imagen de la carpeta *img\_query*, para los métodos de división en 9,16 y 36 celdas respectivamente. En las figuras se observa que si bien en los tres primeros lugares se sugirieron imágenes de la misma clase, a excepción de la división 6x6, que sugirió dos, dentro de los primeros lugares, el desempeño general varía al observar las otras imágenes sugeridas, puesto que si mientras en la división no rectangular se sugieren imágenes de clases bastante cercanas a la imagen original, los otros métodos se alejan más de esta, presentando algunas imágenes que cualitativamente podría definirse como no correspondientes.

### 3. Resultados.



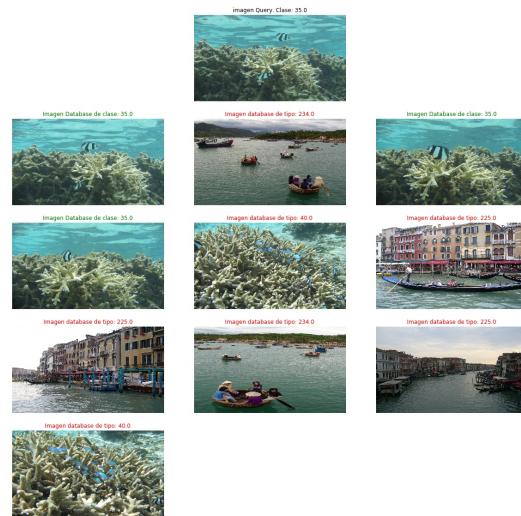
**Figura 3.1:** Imágenes sugeridas, 5 divisiones



**Figura 3.2:** Imágenes sugeridas, división 3x3



**Figura 3.3:** Imágenes sugeridas, división 4x4



**Figura 3.4:** Imágenes sugeridas, división 6x6

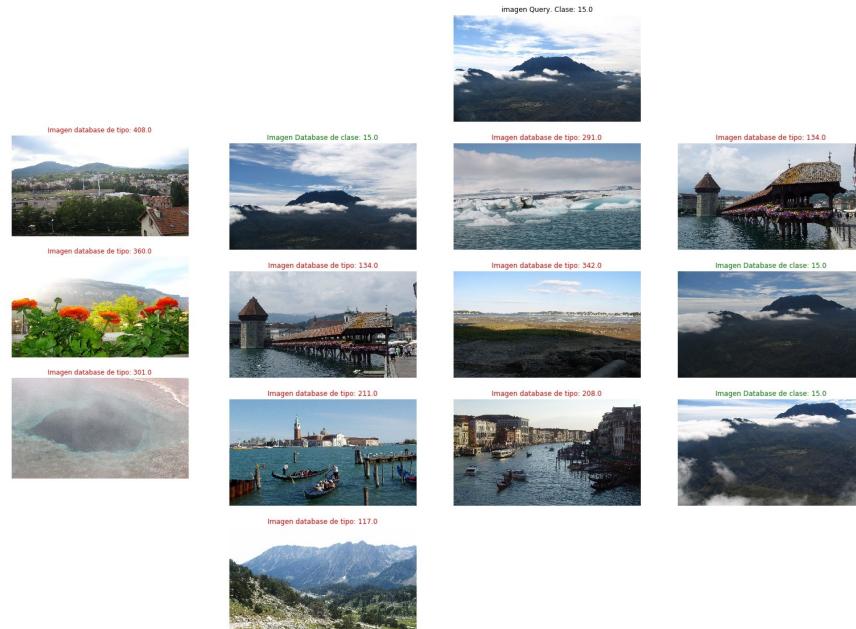
A continuación, se presentarán una serie de consultas para diferentes imágenes de la carpeta *img\_query* utilizando el método de división no rectangular, puesto que este presentó el mejor desempeño en relación las métricas presentadas en la tabla 3.1. Es posible observar diferentes desempeños para el método dependiendo

### 3. Resultados.

de la imagen presentada, si bien en general, la presentación de imágenes entrega un conjunto de figuras de colores similares, el que estas correspondan a la clase deseada depende de la imagen presentada, por ejemplo, en la figura 3.8 se observa una desempeño adecuado, puesto que se presentan tres imágenes de la misma clase como tres primeras sugerencia, y luego se presentan figuras que tienen una gran cercanía en color. Con un desempeño menor se presenta la figura 3.7, mientras que las figura 3.5 y 3.6 presentan un desempeño intermedio, presentando imágenes similares en color a la imagen consultada.



**Figura 3.5:** Imágenes sugeridas, query1



**Figura 3.6:** Imágenes sugeridas, query2

**Figura 3.7:** Imágenes sugeridas, query3**Figura 3.8:** Imágenes sugeridas, query11

### 3.2. Método propuesto: HOG

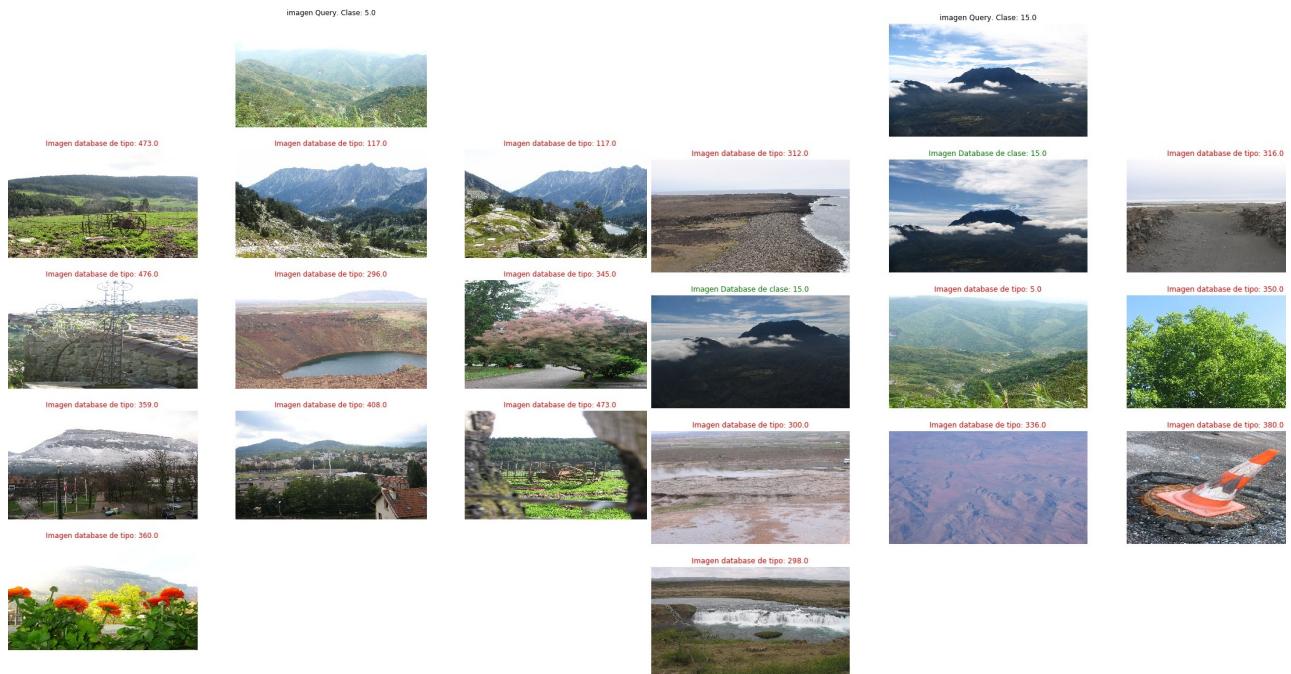
En primer lugar, al igual que en el caso de los métodos propuestos en enunciado se calcula la métrica *Rank normalizado promedio*, con el fin de evaluar el método. En la tabla 3.2 se presentan los valores obtenidos para la métrica en el caso del mecanismo basado en histogramas de gradientes, adicionalmente, en la misma tabla se incluyen los valores del mejor y peor mecanismo encontrados anteriormente. A partir de los resultados obtenidos para el valor de *Rank Normalizado promedio* es posible observar que el mecanismo de selección de características mediante el uso de histogramas de gradiente mejora el desempeño en comparación a los algoritmos de división en celdas cuadradas, sin embargo, se encuentra por debajo del desempeño presentado por la división no rectangular de imagen.

Método	Rank Normalizado Promedio
3x3	0.4559
celdas no rectangulares	0.0792
HOG	0.3312

**Cuadro 3.2:** Métricas métodos de división indicados.

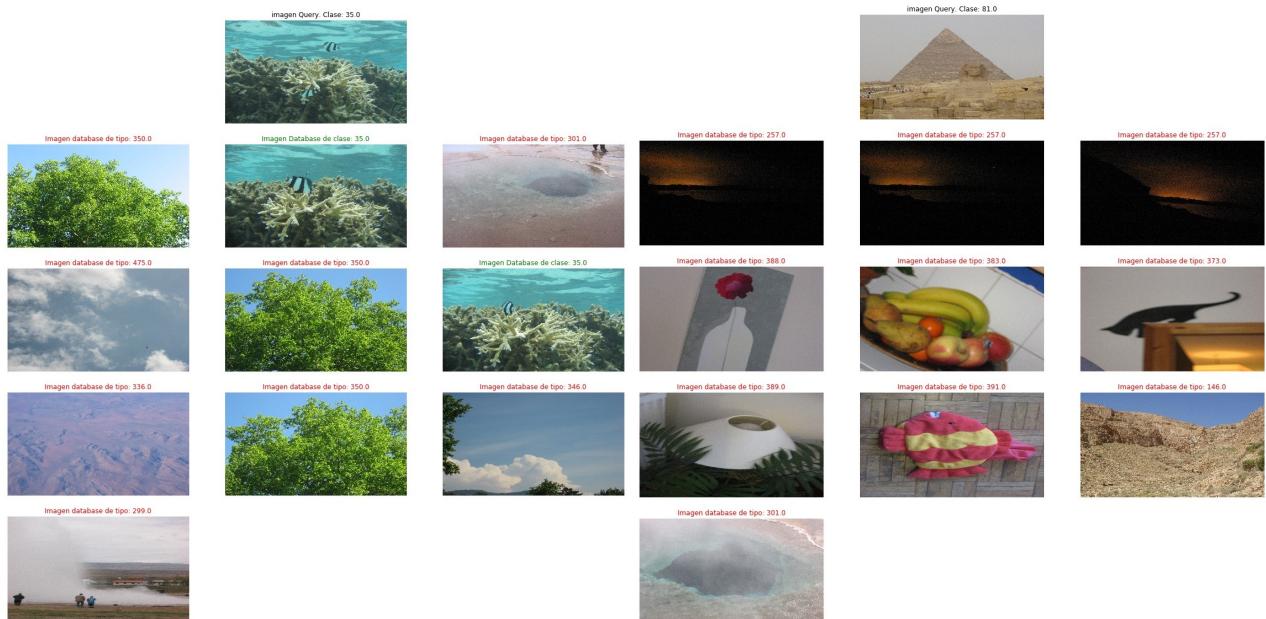
A continuación se presentan ejemplos de resultados cuantitativos, esto corresponde a ejemplo de resultados arrojados mediante esta metodología en la búsqueda de imágenes de *query* utilizadas para los ejemplos. Para mostrar los resultados se presentan las imágenes determinadas como similares para las mismas figuras de *query* utilizadas en la sección anterior con los métodos definidos en el enunciado del proyecto.

### 3. Resultados.



**Figura 3.9:** Imágenes sugeridas, query1, método HOG

**Figura 3.10:** Imágenes sugeridas, query2, método HOG

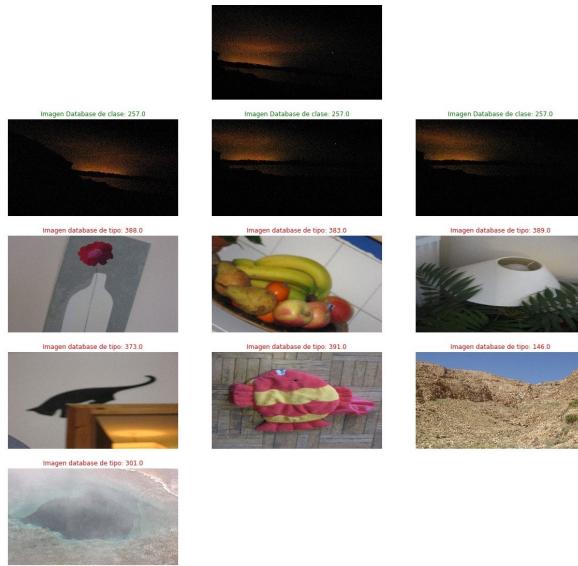


**Figura 3.11:** Imágenes sugeridas, query3, método HOG

**Figura 3.12:** Imágenes sugeridas, query11, método HOG

### 3. Resultados.

Como es posible observar en los ejemplos presentados, el desempeño de la selección de características mediante HOG, es cualitativamente peor en comparación con el método de división por colores desarrollado previamente. Cabe notar, que en este caso el color de la imagen no es relevante en la decisión, por lo que no es extraño que algunas de las sugerencias presentadas no tengan una cercanía en color a la imagen presentada como consulta. Dado que el método centra su funcionamiento en el cálculo de gradientes se busca una imagen que tenga más textura, con el fin de evaluar el desempeño en este caso, para esto se presenta la figura 3.13, correspondiente a la figura 53 de las figuras de consulta. A partir de esta se puede observar aciertos de comparación en los tres primeros casos, por lo que es de presumir que el mejor desempeño del método con respecto al caso de cortes rectangulares se debe a la mejor clasificación en este tipo de imágenes.



**Figura 3.13:** Imágenes sugeridas, query53, método HOG

### 3.3. Resultados utilizando IRP.

En primer lugar, al igual que en los casos anteriores, se calcula la métrica *Rank normalizado promedio*, con el fin de evaluar el método. En la tabla 3.3 se presentan los valores obtenidos para la métrica en el caso del mecanismo IRP combinando HOG y la división en 5 zonas no rectangulares, adicionalmente, en la misma tabla se incluyen los valores *Rank Normalizado Promedio*, para cada uno de estos métodos por separado. A partir de los resultados obtenidos para el valor de *Rank Normalizado promedio* es posible observar que el mecanismo de selección de características mediante IRP mejora el desempeño en comparación al algoritmo HOG, sin embargo, se encuentra por debajo del desempeño presentado por la división no rectangular de imagen.

Método	Rank Normalizado Promedio
<b>IRP</b>	0.1027
<b>celdas no rectangulares</b>	0.0792
<b>HOG</b>	0.3312

**Cuadro 3.3:** Métricas métodos de división indicados.

A continuación, tal como se realizó previamente, se presentan resultados cuantitativos de las búsquedas

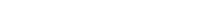
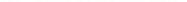
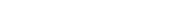
de diferentes figura existentes en la carpeta *img\_query* a modo de ejemplo del funcionamiento del sistema de selección de características utilizando la integración mediante IRP, donde es posible observar la influencia de ambos métodos, debido a que se presentan imágenes incluidos previamente en uno o en otro. Adicionalmente, si bien en la figura 3.14 se presenta un peor caso que al utilizar directamente la segmentación en sectores no rectangulares, en otras imágenes tales como las figuras 3.15, 3.17 y 3.19



**Figura 3.14:** Imágenes sugeridas, query1, método IRP

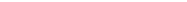
**Figura 3.15:** Imágenes sugeridas, query2, método IRP

### 3. Resultados.



**Figura 3.16:** Imágenes sugeridas, query3, método IRP

**Figura 3.17:** Imágenes sugeridas, query11, método IRP



**Figura 3.18:** Imágenes sugeridas, query3, método IRP

**Figura 3.19:** Imágenes sugeridas, query11, método IRP

## 4. Conclusión

A partir del trabajo realizador, es posible concluir que fue posible cumplir el objetivo propuesto como proyecto de final de curso, correspondiente al desarrollo de un sistema CBIR (*Content Based Image Retrieval*), dado que a partir de los resultados obtenidos tanto cuantitativa como cualitativamente, se observó una mejora en comparación a la elección azarosa de imágenes y de manera adicional, se observó que las imágenes que no correspondían a la misma clase, fueron en general similares ya sea en color o forma a la imagen de búsqueda (*query*).

Es posible observar la utilidad que presenta en el cálculo de vectores de características la subdivisión no rectangular de la imagen, puesto que utilizando esta metodología, considerando las cinco divisiones presentadas en este informe, fue que se obtuvieron los mejores resultados en relación a la presentación de imágenes similares.

Adicionalmente, es posible concluir la utilidad de la metodología de *Inverse Rank Position* con el fin de lograr un mejor desempeño del sistema. Este punto fue observado principalmente al realizar una comparación de desempeño entre utilizar el método HOG puro y la realización del método IRP para integrar este método junto a la división en 5 celdas no rectangulares.

Finalmente, en relación a los aprendizajes logrados al realizar este proyecto, en primer lugar, cabe destacar el aprendizaje en el uso de herramientas en Python para el procesamiento de imágenes, principalmente las librerías *openCV* y *scikit-image*. Adicionalmente, la familiarización con métodos de CBIR, específicamente las maneras de obtener vectores de características a partir de imágenes, donde se exploraron diferentes métodos durante el desarrollo del proyecto. Finalmente, cabe destacar la importancia del uso de métricas para la evaluación del desempeño de algoritmos, esto puesto que dado el gran volumen de imágenes es engorroso realizar todas las visualizaciones, por lo que contar con una métrica permite obtener un valor para comparación directa.

## Bibliografía

- [1] Proyecto Final Área Imágenes. Publicación en Material Docente, curso EL5206 - Laboratorio de Inteligencia Computacional y Robótica.
- [2] A content-based image retrieval (CBIR) system, pochic [Online]. Disponible en: <https://github.com/pochih/CBIR>
- [3] Satya Mallick, *Histogram of Oriented Gradients*, Learn OpenCV [Online]. Disponible en: <https://www.learnopencv.com/histogram-of-oriented-gradients/#comments>
- [4] scikit-image [Online]. Disponible en: <http://scikit-image.org/docs/dev/api/skimage.feature.html#skimage.feature.hog>
- [5] Jovic.M, Hatakeyama.Y, Dong.F, Hirota.K, *Image Retrieval Based on Similarity Score Fusion from Feature Similarity Ranking List* In International conference on Fuzzy Systems and Knowledge Discovery (pp. 461-470). Springer, Berlin, Heidelberg.
- [6] Shukla.J, Vania.Jignesh, *A Survey on CBIR Features Extraction Techniques*, International Journal Of Engineering And Computer Science ISSN:2319-7242. Dec.2014.