

**INFORME:**

**“Búsqueda por similitud usando HOG *(Histogram of oriented gradients)”***

***Curso: Visión por Computadora***

***Profesor: José M. Saavedra***

***Estudiantes: María José Briceño***

***Guillermo Otárola***

1. **Objetivo**

El objetivo de esta tarea es familiarizarse con métodos de extracción de características de imágenes en el contexto particular de búsqueda por similitud.

1. **Descripción**

El histograma de gradientes orientados (HOG) fue propuesto por Dalal y Triggs y originalmente fue utilizado para detección de peatones. Este algoritmo de detección de objetos consiste en generar una descripción basada en el gradiente de los píxeles. Esto se realiza creando un histograma que representa los ángulos de los píxeles ponderados por sus magnitudes.

En esta tarea se pide evaluar el desempeño del método HOG como extractor de características de imágenes. Pare ello se dispone de un conjunto de datos que contiene 14.000 imágenes y 6.000 imágenes de test.

El desempeño del método será evaluado a través de la métrica Mean Average Precisión (mAP) que consiste en una medida para evaluar la precisión de los detectores de objetos. La precisión promedio calcula el valor de precisión promedio para el valor de recuperación de 0 a 1. La precisión mide cuán precisas son sus predicciones, es decir, el porcentaje de sus predicciones son correctas.

1. **Implementación y Resultados**

**3.1 Cálculo del descriptor HOG y evaluación de detección de objetos mediante Mean Average Precisión (mAP)**

**HOG :** El descriptor HOG permite aprovechar de forma bastante eficiente la información del gradiente a partir de combinar esta información en forma de histogramas locales, que se calculan en celdas de pequeño tamaño y se distribuyen de forma uniforme por toda la imagen.

**Evaluación de detección de objetos:** En la detección de objetos, la evaluación no es trivial, porque hay dos tareas distintas para medir:

1. Determinar si existe un objeto en la imagen (clasificación)
2. Determinar la ubicación del objeto (localización, una tarea de regresión).

Además, en un conjunto de datos típico habrá muchas clases y su distribución no será uniforme. Entonces, una métrica simple basada en la precisión introducirá sesgos. También es importante evaluar el riesgo de clasificaciones erróneas. Por lo tanto, existe la necesidad de asociar un "puntaje de confianza" o puntaje de modelo con cada cuadro delimitador detectado y evaluar el modelo en varios niveles de confianza.

Para abordar estas necesidades, se introdujo la precisión promedio (AP). Para comprender el AP, es necesario comprender el Presicion y el Recall de un clasificador, Brevemente, en este contexto, la precisión mide la "tasa de falsos positivos" o la relación entre las detecciones de objetos verdaderos y el número total de objetos que predijo el clasificador. Si tiene un puntaje de precisión cercano a 1.0, existe una alta probabilidad de que lo que el clasificador predice como detección positiva sea, de hecho, una predicción correcta y el Recall mide la "tasa de falsos negativos" o la relación entre las detecciones de objetos verdaderos y el número total de objetos en el conjunto de datos. Si tiene un puntaje de Recall cercano a 1.0, el modelo detectará positivamente casi todos los objetos que están en su conjunto de datos.

Para visualizar la evaluación imprimimos 2 tipos de comparación 3 de 7 aciertos y cuando acierta a las 7 imágenes, es decir, para un AP de 0.42 y un AP de 1.0

A continuación se visualiza el código de implementación:

def calculate\_distance\_to\_all\_images(str\_datadir, original\_hogs, original\_labels, original\_filenames):

str\_file = os.path.join(str\_datadir, "test.list")

test\_files\_ = data.read\_data(str\_file)

filenames, labels = zip(\*test\_files\_)

image\_size = (128, 128)

grid\_size = (8, 8)

orientations = 8

hog = hog\_features.HOG(image\_size, orientations, grid\_size)

dim = hog.get\_lenght()

n\_images = len(filenames)

features = np.zeros((n\_images, dim), np.float32)

print('feat shape {}'.format(features.shape))

sum\_of\_aps = 0

for i, (filename, current\_label) in enumerate(zip(filenames, labels)) :

distances, filename = distancias\_ordenadas\_a\_imagen\_de\_test(current\_label, filename, hog, original\_filenames, original\_hogs,

original\_labels, str\_datadir)

**#CacularAP y mAP**

relevant\_counter = 0

AP = 0

for index, d in enumerate(distances):

dividendo = 0

if d['is\_relevant']:

relevant\_counter += 1

dividendo = relevant\_counter

AP += dividendo / (index + 1)

if relevant\_counter:

AP = AP/relevant\_counter

sum\_of\_aps += AP

imprimir\_imagenes(distances, filename, i, relevant\_counter, str\_datadir)

mAP = (sum\_of\_aps/ len(filenames))

print(mAP)

def distancias\_ordenadas\_a\_imagen\_de\_test(current\_label, filename, hog, original\_filenames, original\_hogs, original\_labels, str\_datadir):

filename = os.path.join(str\_datadir, 'png\_w256', filename)

image = pai.imread(filename, as\_gray=True)

this\_hog = hog.get\_hog(image)

distances = []

for index, (other\_hog, other\_label, other\_filename) in enumerate(zip(original\_hogs, original\_labels, original\_filenames)):

dist = np.linalg.norm(

this\_hog - other\_hog) ## calculando la distancia euclidiana entre el hog de la imagen de test y la previamente calculada

is\_relevant = current\_label == other\_label ## Es relevante SSI el label del test es igual a la de la imagen cercana.

datos\_de\_comparacion = {'is\_relevant': is\_relevant,

'dist': dist,

'other\_filename': other\_filename,

}

sorted\_distances = sorted(distances, key=lambda d: d['dist'])

# y seleccionamos las 7 más cercanas.

distances = sorted\_distances[:7]

return distances, filename

def imprimir\_imagenes(distances, filename, i, relevant\_counter, str\_datadir):

r = np.random.random()

imprimir = False

if (relevant\_counter == 3 and r > 0.9) or relevant\_counter == 6:

imprimir = True

## Imprimiremos los que tengan 3 aciertos y todos los que tengan 6 aciertos

if imprimir:

para\_mostrar = [filename]

for d in distances[:7]:

para\_mostrar.append(os.path.join(str\_datadir, 'png\_w256', d['other\_filename']))

final\_dim = (256 \* len(para\_mostrar), 256)

new\_im = Image.new('L', final\_dim)

for j, im\_file in enumerate(para\_mostrar):

im = Image.open(im\_file)

new\_im.paste(im, (j \* 256, 0))

new\_im.save('comparado\_{}\_{}.jpg'.format(relevant\_counter, i))

print('comparado\_{}\_{}.jpg'.format(relevant\_counter, i))

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser(description = '')

parser.add\_argument('-datadir', type = str, required = True)

args = parser.parse\_args()

str\_datadir = args.datadir

str\_file = os.path.join(str\_datadir, "data.list")

data\_ = data.read\_data(str\_file)

filenames, labels = zip(\*data\_)

np\_labels = np.array(labels, np.int)

image\_size = (128,128)

grid\_size = (8,8)

orientations = 8

hog = hog\_features.HOG(image\_size, orientations, grid\_size)

dim = hog.get\_lenght()

n\_images = len(filenames)

features = np.zeros((n\_images, dim), np.float32)

print('feat shape {}'.format(features.shape))

for i, filename in enumerate(filenames) :

filename = os.path.join(str\_datadir,'png\_w256', filename)

image = pai.imread(filename, as\_gray = True)

fv = hog.get\_hog(image)

features[i,:] = fv

str\_outfile = os.path.join(str\_datadir, 'features.np')

str\_labels = os.path.join(str\_datadir, 'labels.np')

features.tofile(str\_outfile)

print('feature saved at {}'.format(str\_outfile))

np\_labels.tofile(str\_labels)

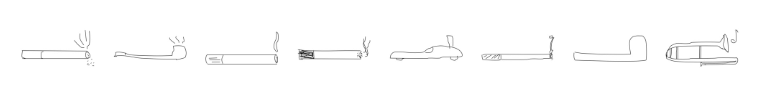
print('labels saved at {}'.format(str\_labels))

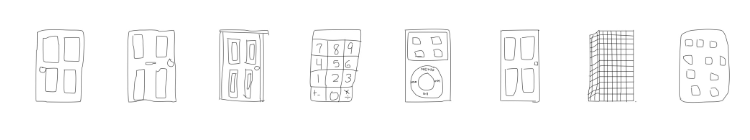
calculate\_distance\_to\_all\_images(str\_datadir, features, labels, filenames)

1. **Conclusiones y Resultados**

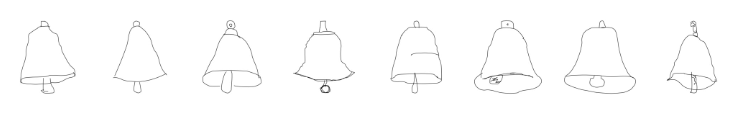
A continuación de muestran algunos ejemplos de comparación con AP = 0,42 ES DECIR 3 Aciertos de 7

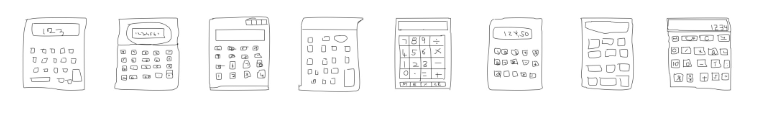


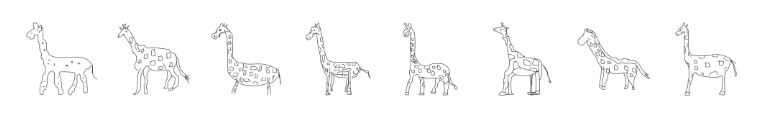




A continuación de muestran algunos ejemplos de comparación con AP = 1 ES DECIR 7 Aciertos de 7







Para cada Query se calculó su AP correspondiente. Un usuario puede tener tantas consultas como quiera en esta base de datos etiquetada. El mAP es la media de todas las consultas que hizo el usuario, en este caso:

**Mean Average Precisión (mAP) = 0.35993837962962955**