INFORME DEL TRABAJO # 2:
DETECCIÓN DE IMÁGENES

Trabajo realizado por

MAICOL LLANO MONCADA

CRISTINA GÓMEZ SANTAMARIA

Docente

UNIVERSIDAD PONTIFICIA BOLIVARIANA

CURSO DE SISTEMA Y SEÑALES

MEDELLÍN

26/05/2017

INFORME TRABAJO 2 - SISTEMAS Y SEÑALES –

DETECCIÓN DE IMÁGENES

Para el desarrollo de este trabajo, se decidió implementar un detector en cascada. Dicha implementación, se llevó a cabo en el software Matlab. A continuación se explicará cada uno de los pasos y etapas que se llevaron a cabo para su desarrollo:

1. DETECTOR EN CASCADA (BREVE EXPLICACIÓN):

A grandes rasgos, un detector en cascada funciona a partir del principio de ventanas deslizantes. Una ventana deslizante, representa un fragmento de la imagen que es seleccionado para pasar a través del detector y determinar si en dicho fragmento se encuentra el objeto que se debe identificar. Dichas ventanas como su nombre lo dice, son deslizantes, es decir, que realizan un barrido por toda la imagen con el fin de llevar a cabo el proceso de detección. Una de las principales características de estas ventanas, es que una vez se completa el barrido completo en las dos dimensiones de la imagen, cambia la escala de la misma, para dar lugar a un nuevo barrido completo. Es decir, se analiza la imagen en diferentes escalas (tamaños), lo cual, hace que el proceso de detección sea más completo.

El proceso de detección, funciona de la siguiente manera:

Una vez seleccionada una ventana de la imagen, se realiza un proceso de extracción de rasgos o características. Luego de dicha extracción, se pasa a través de una serie de clasificadores que se encargan de analizar las características extraídas previamente para determinar si concuerdan con las características del objeto a detectar. Estos clasificadores se encargan de entregar una respuesta a la detección, es decir, entregan un ‘sí’ en el caso de que el objeto sea detectado y un ‘no’ en caso de que no se encuentren coincidencias significativas. Una de las ventajas que tiene esta etapa del proceso, es que sólo basta que uno de los clasificadores determine que allí no se encuentra el objeto a detectar, para dar por descartada la ventana que se esté analizando. Esto, hace que el proceso de detección sea veloz, ya que si por ejemplo desde el primer clasificador se detectó que la ventana no contiene el objeto, no es necesario que pase por las demás etapas, debido a que la ventana se descarta inmediatamente. Para que se determine que el objeto se encuentra en la ventana que se está analizando, es necesario que todas las etapas de clasificación entreguen como respuesta un ‘sí’ en el proceso de detección.

Definiciones importantes:

Imagen positiva: Imagen que contiene una o más veces el objeto que se desea detectar.

Imagen negativa: Imagen que no contiene el objeto que se desea detectar.

Falsos positivos: Detección de objetos o fragmentos de la imagen que no corresponden con el objeto que se debe detectar.

Falso negativo: No detección de zonas donde se encuentran objetos que debieron ser detectados

Verdadero positivo: Detección de imágenes o fragmentos que de la imagen que sí corresponden con el objeto que se debe detectar.

Verdadero Negativo: No detección de objetos en una imagen que no contiene objetos que debían ser detectados.

Verdadero positivo (TP)

Verdadero negativo (TN)

Falso positivo (FP)

Falso negativo (FN)

Entrenamiento: Proceso de preparación que realiza el detector, a partir de cierta información que el diseñador recolecta (imágenes positivas, imágenes negativas). Cuanto mejor sea el entrenamiento, mejores serán los resultados que se obtengan en los procesos de detección.

False Alarm Rate (Tasa de falsos positivos): Es un parámetro, que nos dice que tan grande o pequeño es el número de falsos positivos con respecto a la cantidad de imágenes negativas que se utilizaron para el entrenamiento.

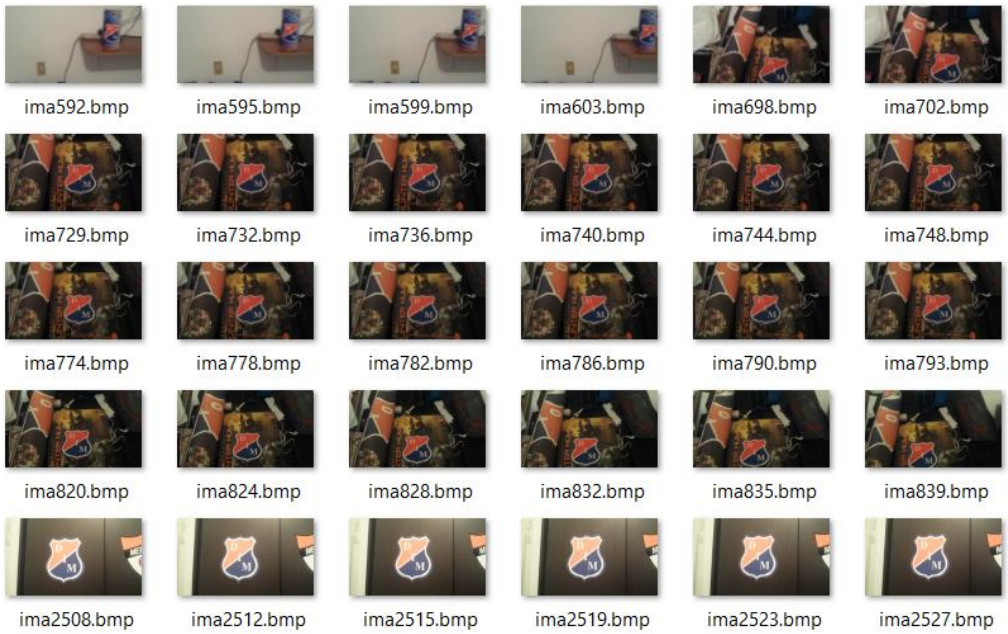
True Positive Rate (Tasa de verdaderos positivos): Es un parámetro, que nos indica que tan grande o pequeño es el número de detecciones correctas con respecto a la cantidad de imágenes positivas que se utilizaron para el entrenamiento.

2. PROCESO DE RECOLECCIÓN DE IMÁGENES Y ENTRENAMIENTO

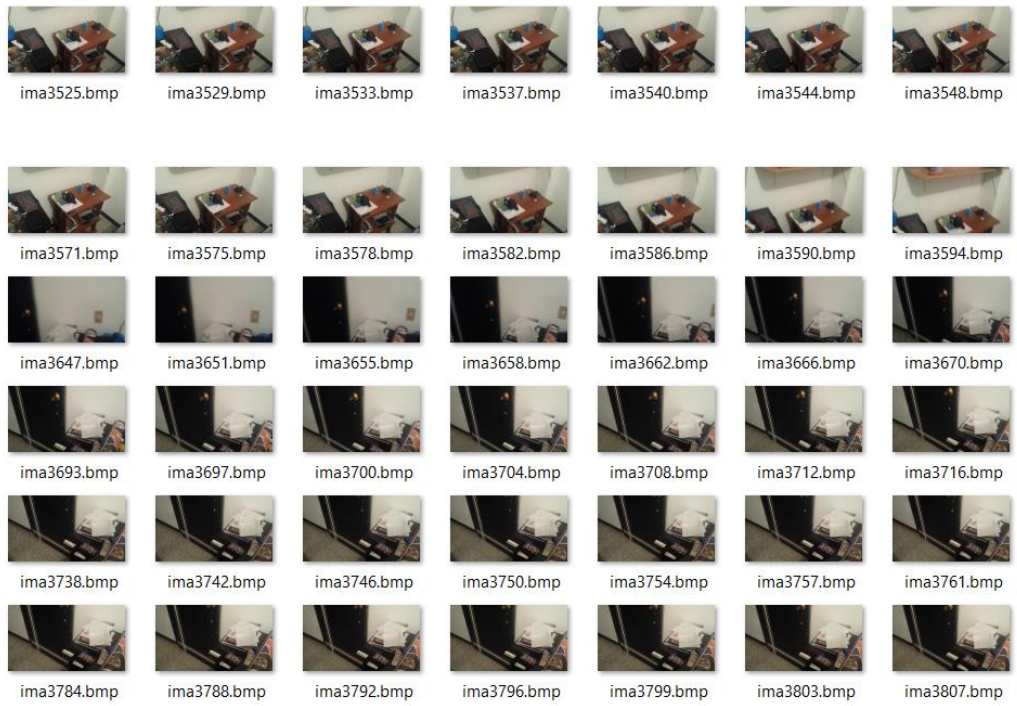
El detector que se construyó, tiene como propósito detectar escudos del Deportivo Independiente Medellín.

Para la recolección de las imágenes, se tomó como alternativa la grabación de un vídeo y a partir de este se extrajo la colección de imágenes positivas y negativas necesarias para el entrenamiento del detector. Durante la grabación de vídeo se filmaron diferentes escudos para aportar distintas fuentes al detector, y además se filmaron diferentes escenarios que no contenían ningún escudo para extraer las imágenes negativas.

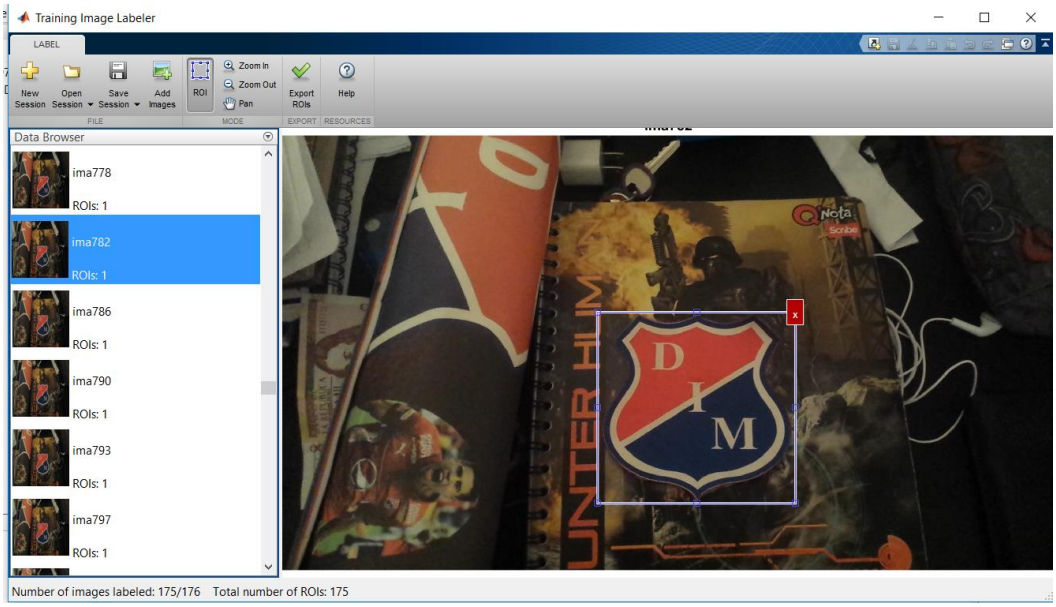
Muestra de imágenes positivas:



Muestra de imágenes negativas:



Una vez se clasificaron las imágenes en positivas y negativas, se procedió a realizar un proceso de etiquetado de las positivas mediante la función Training Image Labeler. Allí, se crea una base de datos que contiene todas las imágenes positivas, y para cada una de ellas se selecciona el área de interés (área donde se encuentra el objeto que se quiere detectar. A continuación se muestra un ejemplo de etiquetado de una de las imágenes.



Una vez se tienen todas las imágenes etiquetadas, se genera una estructura que contiene para cada imagen su ubicación de almacenamiento y la información de las áreas de interés (coordenadas), es decir, donde se encuentra el objeto a detectar La estructura resultante tiene el siguiente formato:

Gtrain

1x175 struct with 2 fields

Fields	imageFilename	objectBoundingBoxes
1	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[300 18 498 591]
2	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[306 85 496 582]
3	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[337 118 481 569]
4	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[371 112 496 575]
5	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[412 112 476 571]
6	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[426 99 484 580]
7	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[431 101 479 574]
8	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[412 83 486 588]
9	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[396 85 487 584]
10	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[376 65 505 591]
11	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[355 71 508 596]
12	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[347 79 500 590]
13	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[335 83 508 590]
14	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[345 81 506 598]
15	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[341 83 506 600]
16	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[359 101 504 580]
17	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[358 105 505 592]
18	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[371 91 502 592]
19	'C:\Users\Asus\Desktop\TRABAJO DETECCIÓN ...	[357 79 524 614]

Después de tener todos estos parámetros, se procedió a la fase de entrenamiento y pruebas del detector. Esto, se hizo mediante un código en Matlab que será explicado en la siguiente sección.

3. CÓDIGOS

3.1 CÓDIGO PARA EXTRAER LAS IMÁGENES

```
clc, clear all, close all

video = '20170526_004113.mp4';    % Nombre del video
carpeta = 'imagenes';  % Nombre de la carpeta para almacenar las imágenes
NumeroImagenes = 1000; %Número de imágenes a extraer

videol = VideoReader(video);
vectorframes = round(linspace(1,videoobj.NumberOfFrames,NumeroImagenes));
for i=vectorframes
    frame = read(videol,i);
    namefile = [carpeta, '\ima', num2str(i), '.bmp'];
    imwrite(frame,namefile)
end
```

Definición de las variables:

Video: Es el nombre (string) del vídeo del cual se desean extraer la imágenes

Carpeta: Nombre de la carpeta donde se almacenarán las imágenes

NumeroImagenes: Número de imágenes que se quieren extraer a partir del video.

Videol: Variable en la cual se lee la información contenida en el video y que almacena algunos parámetros como nombre, ubicación, framerate, duración, entre otros. (Para realizar la lectura del video, se hace uso de la función VideoReader)

Vectorframes: Es un vector que contiene tantos elementos como imágenes se quieran extraer, es decir, toma su longitud a partir de la variable Numeroimagenes. Cada una de las posiciones de este vector, representa el número de un cuadro (frame) que pertenece al vídeo. Básicamente, lo que se hace es usar la función linspace, para generar un vector que distribuya uniformemente el número de cuadros que contiene el video entre el número de imágenes que se desean extraer.

NOTA: Se usó la función round, ya que es posible que al generar el vector mediante la función linspace, algunos de sus elementos sean números no enteros. Teniendo en cuenta que lo que representan dichos elementos es el número de un cuadro perteneciente al vídeo, no es posible que este número no sea entero.

Una vez definidas estas variables, se procede a ejecutar un ciclo for, encargado de recorrer cada una de las posiciones almacenadas en la variable vectorframes, leer el cuadro correspondiente, y almacenarlo como una imagen, dentro de una ubicación específica.

3.2 CÓDIGO DEL DETECTOR

```
clc, clear all, close all

%% Entrenamiento de la Adaboost

%% Se carga la estructura positivas.mat . Dicha estructura contiene la
información generada después del proceso de etiquetado de las imágenes.
load('positivas.mat');

%% se especifica la ubicación de la carpeta que contiene las imágenes
positivas
positiveFolder = 'imagenes/positivas';

%% se especifica la ubicación de la carpeta que contiene las imágenes
negativas
negativeFolder = 'imagenes/negativas';


%% se asigna un nombre para el detector
detectorName = 'emergencySignsDetector1.xml';
```

```
%% A continuación se llama a la función trainCascadeObjectDetector,
encargado de realizar el entrenamiento del detector. Los parámetros que
se le entregan son, respectivamente: Nombre asignado al detector, Gtrain
(estructura incluida dentro de positivas.mat), carpeta de imágenes
negativas, tasa de falsos positivos

trainCascadeObjectDetector(detectorName, Gtrain, negativeFolder,...
     'FalseAlarmRate', 0.1, 'NumCascadeStages', 5);

%% Prueba

%% Se carga un detector en cascada, ajustándolo a los parámetros que se
generaron en el archivo detectorName, que fue obtenido luego del
entrenamiento

detector = vision.CascadeObjectDetector(detectorName);

%% Se lee la imagen que se va a utilizar para probar el detector

img = imread('DIM5.jpg');

%% se aplica el detector a la imagen seleccionada, y mediante la función
step, se obtienen las coordenadas donde se detectaron objetos.

bbox = step(detector, img);

%% Se inserta un rectángulo en las coordenadas que resultaron del paso
anterior, y se añade un label.

detectedImg = insertObjectAnnotation(img, 'rectangle', bbox, 'ESCUDO DEL
MEDALLO');

%% Finalmente, se muestra la imagen con los resultados obtenidos

figure; imshow(detectedImg);
```

4. MUESTRA DE FUNCIONAMIENTO

Luego de probar el detector, se observó que funciona perfectamente cuando la imagen de prueba es una de las que se utilizó para su entrenamiento, sin embargo, si la imagen es diferente, los resultados contienen algunos falsos positivos y falsos negativos. Dicha situación es entendible, ya que la cantidad de imágenes que se utilizó para el entrenamiento es muy poca en comparación con los valores recomendados para esta clase de detectores. A continuación se muestra algunos de los resultados obtenidos:

*El primer resultado es con una de las imágenes utilizadas para el entrenamiento, las otras dos son imágenes que no estaban incluidas en el entrenamiento del detector:

5. CONCLUSIONES

- El detector en cascada, es un detector que funciona velozmente, debido a que en las etapas de clasificación por medio de las ventanas deslizantes, sólo requiere de una respuesta negativa para que la ventana sea descartada.

- Para que el detector funcione de manera más eficiente y correcta, requiere de una cantidad de imágenes (negativas y positivas) bastante grande, para que el entrenamiento sea diverso y dé como resultado un detector lo suficientemente preciso.

- El hecho de que se aumente la cantidad de etapas de entrenamiento no necesariamente da como resultad un mejor detector, ya que a medida que se aumenta este número de etapas, el índice de falsos positivos y de verdaderos positivos, tiende a cero.

- El modelo de detección de imágenes implementado, no es sensible a cambios de posición u orientación. Por tanto, un pequeño cambio en la orientación del objeto resulta en una no detección o detección incorrecta

- En la implementación que se hizo, se observó que a pesar de que se logran muchas detecciones correctas, hay también un grado alto de falsos positivos. Se concluye que esto se debe a que la cantidad de imágenes con las cuales se entrenó el detector, está muy por debajo de los valores recomendados para este tipo de detectores.