UADY

UNIVERSIDAD AUTÓNOMA DE YUCATÁN

FACULTAD DE MATEMATICAS

INTELIGENCIA ARTIFICIAL

PROYECTO:

USO DEL APRENDIZAJE SUPERVISADO PARA LA DETECCION DE OBJETOS EN VIDEOS

Este Articulo fue realizado por:
Aldair Cauich Chiu - Ing. en Computacion
Jesus Iran Mena Garcia - Ing. en Computacion

Maestro:
Alejandro Pasos Ruiz

Resumen

La investigacion que aqui se presenta trata de que tan eficas puede ser la implementacion del aprendizaje supervisado para la deteccion de objetos en videos, ya sean grabados anteriormente o en vivo mediante camaras o cualquier dispositivo digital.
Como metodo para demostrar si nuestra hipotesis es correcta o no, se realizaron varios video-pruebas donde el objetivo es identificar un balon de basketball en diferentes entornos que podrian afectar a la deteccion.
Los resultados dependieron de la muestra ya que en cierto videos con ambientes abiertos tuvieron mas problemas que en un ambiente controlado. Al observar todos los casos posibles se llego a la conclusion de que el aprendizaje supervisado es eficaz para la deteccion de objetos en entornos controlados.

1. Introducción

La inteligencia artificial es una área multidisciplinaria que a través de ciencias como las ciencias de la computación, la matemática, la lógica y la filosofía, estudia la creación y diseño de sistemas capaces de resolver problemas cotidianos por si mismos utilizando como paradigma la inteligencia humana.
En nuestro caso queremos utilizar el método de aprendizaje supervisado apoyándonos de frames (imágenes), le enseñamos al programa lo que queremos ubicar en un video previamente grabado y así que aprenda a detectar por medio de una referencia previa en este caso la imagen, ese objeto en el video que en nuestro experimento será una pelota de basketball, este tipo de ejemplos podría ser utilizado en el aspecto de seguridad para las empresas así como en muchos otros.
Utilizamos el método de aprendizaje supervisado ya que lo que queremos es que el programa aprenda a distinguir un objeto dentro un entorno en base a una referencia previa y no necesitar de la presencia humana para detectar cosas en específico. El programa aprende por medio de datos de entrenamiento, en nuestro caso una o varias imágenes que hacen referencia al objeto que queremos encontrar.

2. Marco Teorico

En ciencias de la computación el aprendizaje automático o aprendizaje de máquinas es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. En muchas ocasiones el campo de actuación del aprendizaje automático se solapa con el de la estadística, ya que las dos disciplinas se basan en el análisis de datos.

Sin embargo, el aprendizaje automático se centra más en el estudio de la complejidad computacional de los problemas. Muchos problemas son de clase NP-hard, por lo que gran parte de la investigación realizada en aprendizaje automático está enfocada al diseño de soluciones factibles a esos problemas. El aprendizaje automático puede ser visto como un intento de automatizar algunas partes del método científico mediante métodos matemáticos.

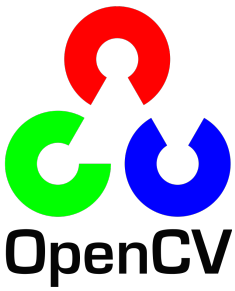
El aprendizaje automático tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica.

Aprendizaje supervisado En aprendizaje automático y minería de datos, el aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares de objetos (normalmente vectores): una componente del par son los datos de entrada y el otro, los resultados deseados. La salida de la función puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación). El objetivo del aprendizaje supervisado es el de crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos, los datos de entrenamiento. Para ello, tiene que generalizar a partir de los datos presentados a las situaciones no vistas previamente. [1]

El aprendizaje supervisado puede generar modelos de dos tipos. Por lo general, genera una función que transforma los datos de entrada en los resul-

tados deseados. Con el fin de resolver un determinado problema de aprendizaje supervisado (por ejemplo, aprender a reconocer la escritura) uno tiene que considerar varios pasos:
1- Determinar el tipo de ejemplos de entrenamiento.
2- Reunir un conjunto de entrenamiento que consiste en la formación de las características del mundo real.
3- Determinar la función de ingreso de la representación de la función aprendida.
4- Determinar la estructura de la función adecuada para resolver el problema y la técnica de aprendizaje.
5- Completar el diseño.

OPENCV



Lanzado oficialmente en 1999, el proyecto OpenCV fue inicialmente una iniciativa Intel Research para avanzar en las aplicaciones intensivas de la CPU, parte de una serie de proyectos, incluyendo el trazado de rayos en tiempo real y las paredes de visualización 3D. Los principales contribuyentes al proyecto incluyeron una serie de expertos en optimización de Intel Rusia, asícomo la biblioteca de rendimiento del equipo de Intel. En los primeros días de OpenCV, se describen los objetivos del proyecto como:

Avanzar en la investigación de visión proporcionando código abierto no sólo, sino también optimizada para la infraestructura básica de la visión. No más reinventar la rueda.
Difundir el conocimiento de visión, proporcionando una infraestructura común que los desarrolladores pueden construir, por lo que el código será más fácilmente legibles y transferibles.

Visión basada en aplicaciones comerciales anticipadas haciendo código portable, rendimiento optimizado disponible de forma gratuita, con una licencia que no requerÃa ser abierto o liberarse.
Areas de aplicación de OpenCV incluyen:

- Juegos de herramientas de funciones en 2D y 3D
- Estimación Egomotion
- Sistema de reconocimiento facial
- Reconocimiento de gestos
- La interacció persona-ordenador(HCI)
- Robótica móvil
- Comprensión del movimiento
- Identificación de objetos
- Estereopsis estéreo visión: la percepción de profundidad de 2 cámaras
- Seguimiento del movimiento
- Realidad aumentada

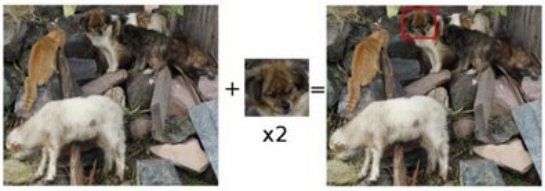
Para apoyar algunas de las áreas antes mencionadas, OpenCV incluye una biblioteca de aprendizaje automático estadístico que contiene:

- Decisión de aprendizaje de árbol
- Algoritmo k-vecino más cercano
- Clasificador de Bayes
- Redes neuronales artificiales

Template Matching

El emparejamiento de plantilla es un método para buscar y encontrar la ubicación de una imagen de plantilla en una imagen más grande. OpenCV viene con una función cv2.matchTemplate() para este propósito. Simplemente desliza la imagen de la plantilla sobre la imagen de entrada (como en convolución 2D) y compara la plantilla y la imagen de entrada de donde viene la plantilla. Varios métodos de comparación se implementan en OpenCV. Devuelve una imagen en escala de grises, donde cada píxel indica cuánto hace la vecindad de esa comparacion de píxeles con la plantilla.

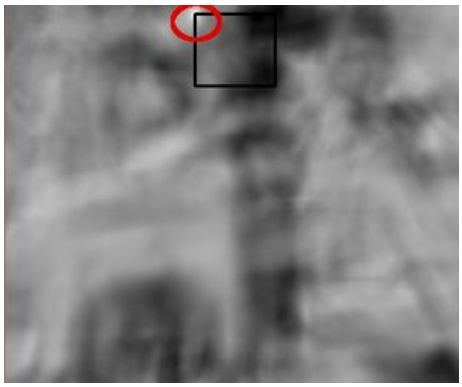
¿Cómo funciona?
Necesitamos dos componentes principales:
Image Fuente (I): La imagen en la que se espera encontrar una coincidencia con la imagen de la plantilla.
Imagen Modelo (T): La imagen de parche que se compara con la imagen de plantilla nuestro objetivo es detectar la zona más alta coincidencia:



Para identificar el área de coincidencia, tenemos que comparar la imagen de la plantilla en contra de la imagen de origen deslizándolo.

Al deslizar, nos referimos a mover el parche de un píxel a la vez (de izquierda a derecha, de arriba a abajo). En cada lugar, con una métrica se calcula por lo que representa como "bueno"ó "malo"la coincidencia en ese lugar es (o lo similar que el parche es esa área en particular de la imagen de origen).

Para cada localización de T sobre I, que almacena la métrica en la matriz resultado R. Cada ubicación (x, y) en I contiene la métrica de coincidencia:



La imagen de arriba es el resultado R de deslizar el parche con una metrica TM CCORR NORMED. Los lugares más brillantes indican las mayores coincidencias. Como puede ver,la ubicación marcada por el círculo rojo es probablemente el que tiene el valor más alto, por lo que la ubicación (el rectángulo formado por ese punto como una esquina y anchura y altura igual a la imagen parche) se considera la coincidencia.

En la práctica, se utiliza el minMaxLoc función para localizar el valor más alto (o más bajo, dependiendo del tipo de método de coindidencia) en la matriz R. [2]

En la siguiente imagen se puede apreciar las diferentes métricas de búsqueda que contiene esta función para el emparejamiento de plantilla.

a. **method=CV_TM_SQDIFF**

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

b. **method=CV_TM_SQDIFF_NORMED**

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

c. **method=CV_TM_CCORR**

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

d. **method=CV_TM_CCORR_NORMED**

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

e. **method=CV_TM_CCOEFF**

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

f. **method=CV_TM_CCOEFF_NORMED**

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

3. Diseño del Experimento

Al momento de realizar el experimento, lo que hicimos fue grabar un video donde el objeto que queríamos encontrar, en nuestro caso la pelota, después de grabar el video, por medio de la librería "opencv" lo que hicimos fue tomar el primer frame del video y guardar la imagen del balón para usarla como plantilla para luego encontrarlo dentro del video. Para esa parte lo que hicimos fue leer el video frame por frame y hacer la comparación de la plantilla con el resto de las imágenes.[3]

Primeramente habíamos grabado el video en un entorno con árboles y plantas, las cuales descubrimos que nos desfavorecían al momento de hacer la búsqueda del objeto, igualmente el cambio de sombras en el transcurso del movimiento del balón hacían que su búsqueda fuese más difícil, ya que aunque tuviera la misma forma a nosotros nos importaba encontrar el balón como en el estado que se encontraba en el primer frame tomado, igualmente la ropa de la persona que participaba en el primer video con el balón, al momento de hacer el cambio a escala de grises del video, la tono de la ropa se confundía con el tono del balón lo cual hacia complicado el análisis.

En el segundo experimento se llevó a cabo con un fondo blanco al igual que la camisa de la persona que sostenía el balón en el video, para que al convertirlo a la escala de grises no hubiese tanta diferencia al buscar el balón, y resulto que en este video fue mucho más fácil encontrar el balón gracias a los cambios de color tanto del fondo como la camisa ya que únicamente tuvimos que utilizar un frame y no varios.

El lenguaje de programación que utilizamos para este experimento es Python y se implementó la librería opencv2 la cual tiene las funciones que necesitaremos para realizar las comparaciones, se describirá brevemente, en pasos, que es lo que el código realiza:

- Primero se llama un video el cual se le aplicara una plantilla que pudo ser obtenida previamente, por aclarar, la plantilla de preferencia tiene que venir del video mismo, ya sea una captura de pantalla o buscar un programa que guarde frame por frame.

- Para la comparación en el video, se debe tomar frame por frame, entonces cuando tenga un frame se convertirá en una imagen gris, porque es más fácil encontrar una coincidencia en una escala en grises.

- Se sube la plantilla, se convierte a gris y se utiliza una Métrica, la cual dependiendo el algoritmo será eficacia al momento de buscar la coincidencia, en nuestro caso utilizamos la métrica TM CCOEFF NORMED ya que fue la que mejor resultados nos dio.

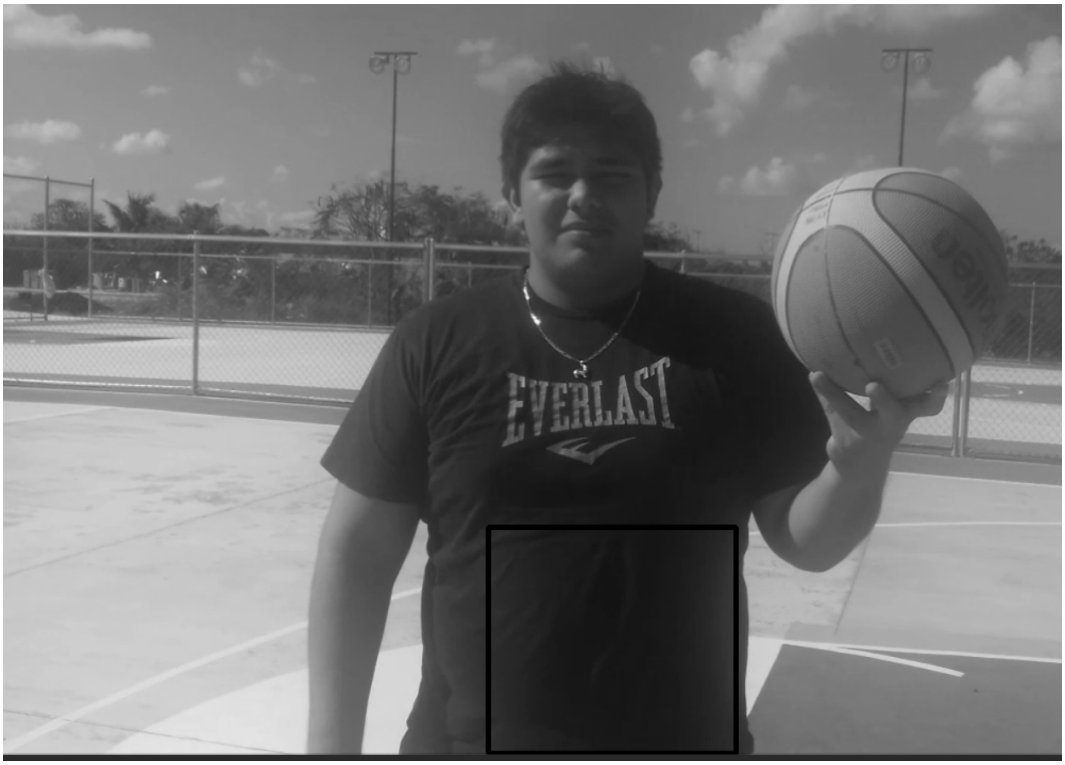
- Ya que logra una coincidencia, puede que no necesariamente sea lo que buscamos, el programa hace la comparación y regresa la zona con mayor coincidencia.

- Lo restante es solo crear un rectángulo alrededor de la coincidencia del tamaño de la plantilla y regresar la imagen la cual se le hizo la comparación para ver el resultado.

```python
import numpy as np
import cv2

cap = cv2.VideoCapture('tronco3.mp4') #Se carga un video

while(True):
    # Captura frame por frame
    ret, frame = cap.read()

    # Aquie empieza la manipulacion de los frames
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #convierte el frame a escala de grises
    template = cv2.imread('mol13.jpg') #se carga la plantilla
    template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY) # convierte la plantilla a gris
    result = cv2.matchTemplate(image,template, cv2.TM_CCOEFF_NORMED) #utiliza una metrica para hacer la comparacion
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)# se ajusta al tamaño de la comparacion
    top_left = max_loc
    h,w = template.shape
    bottom_right = (top_left[0] + w, top_left[1] + h)
    cv2.rectangle(image,top_left, bottom_right,(0,0,255),3) #Dibuja un rectangulo al rededor de la comparacion

    # Imprime la imagen con el rectangulo que muestr la coincidencia
    cv2.imshow('result',image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Cuando todo se termine, procede a cerrarse la ventana y el codigo.
cap.release()
cv2.destroyAllWindows()
```

4. Resultados

Del primer experimento, los resultados preliminares fueron favorables ya que aun teniendo la pelota de frente, detectaba algo distinto, como se puede ver en la siguiente imagen.



Como se puede observar el programa marca una parte del cuerpo en vez la pelota teniéndola claramente, haciendo que durante el video haya pocas coincidencias.

Debido que atreves del video el balón cambia tanto de perspectiva asá como de sombra, para que el código fuese capaz de encontrar el balón durante más tiempo, se tomaron en cuanta más imágenes:

Por lo que el código igual cambia, esto significa que en un rango de frames determinado se utilizara una plantilla, y en otro rango otra y así hasta terminar el video.

```
if (count >= 258 and count < 278):
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    template = cv2.imread('mol10.jpg')
    template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)
    result = cv2.matchTemplate(image,template, cv2.TM_CCOEFF_NORMED)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
    top_left = max_loc
    h,w = template.shape
    bottom_right = (top_left[0] + w, top_left[1] + h)
    cv2.rectangle(image,top_left, bottom_right,(0,0,255),3)
```

Uno de los principales problemas en este código es que se tenían que determinar de qué número de frame a otro se utilizaba una imagen para reconocer el objeto y esto nos desvía del aprendizaje supervisado, ya que consideramos que se tiende a utilizar fuerza bruta. Otra cosa que notÃ¡bamos es que al momento de transformarlo a la escala de grises, el color que tomaba el balón se confundía con el color de la playera y dificultaba encontrarlo, así como el entorno que tienen las esquinas de las diferentes imágenes de referencia.

Para el segundo experimento, se optó por usar un entorno mÃ¡s controlado, buscamos un lugar donde no haya muchas sombras, al igual que la persona de la prueba utilizaría una playera blanca para evitar que lo oscuro nos dé una coincidencia errónea.

Cuando se hizo la prueba con el código, los resultados fueron mucho más favorables que la del primer experimento con una sola plantilla, sigue teniendo sus coincidencias erróneas pero tiene más acertadas y por más tiempo que el primer experimento, al igual que se evita utilizar otro código donde se utilicen varias plantillas.

5. Conclusiones

El objetivo de este proyecto era identificar un objeto predeterminado en un video, en este caso, era poder identificar una pelota de basketball mientras que una persona movía el balón o hacia acciones diferentes, en esto implementamos el aprendizaje supervisado, el cual consiste poder identificar algo con una información previa guardada.

En los resultados nos dimos cuenta que esto es posible, mediante un ejemplo previo del video, pero no es el óptimo y que en un video las cosas cambian, posición, sombras, el fondo, cosas así influyen en la detección pero un así se puede llegar a tener resultados favorables

Como conclusión para este reporte , no nos queda mas que mencionar que a través de las investigaciones realizados, el método empleado no es el mas optimo para el reconocimiento de un objeto en un video, ya que en un video existen varias cosas que pueden hacer que el objeto que queremos encontrar cambie y asi se dificulte se detección, ya que si quisiéramos poder hacer el reconocimiento de manera optima, tendríamos que delimitar por secciones los frames que utilizaría el programa para detectar el objeto y entraría de cierta manera la implementación de fuerza bruta que hace que el sentido del proyecto pierda sentido. Si exiten formas para mejorar este tipo de códigos de reconocimiento de objetos, pero implementaría utilizar librerias mas avanzadas que igualmente hacen que se desvie el propósito de implementar el aprendizaje supervisado. Como se menciono antes, si quisiéramos no utilizar fuerza bruta, las pruebas tendrían que ser en un ambiente ideal, el cual no se puede representar en la vida real, sabemos que el ambiente puede cambiar constantemente y es por eso que este tipo de proyectos pierden un poco su sentido.

Como trabajo a futuro, este programa puede ser modificado, obviamente dejando a un lado el tema del aprendizaje supervisado, como la identificación del color naranja del balón, extrayendo esa parte y crear una plantilla, de igual manera detectando circunferencias se podría detectar el balón. Esto solo aplica para la detección de un objeto, igual se encuentra una base de datos lleno de imágenes, y por detección de tamaño o forma, se extrae y se compara con las plantillas, así pudiendo detectar el objeto.

Referencias

[1] Bradski, Gary; Kaehler, Adrian (2008)
Learning OpenCV: Computer vision with the OpenCV library.
https://en.wikipedia.org/wiki/OpenCV

[2] Open CV Tutorials
Template Matching
http://docs.opencv.org/2.4/doc/tutorials/
imgproc/histograms/template_matching/template_matching.html

[3] Open CV and Python Tutorials
Template Matching
http://opencv-python-tutroals.readthedocs.org/en/latest/
py_tutorials/py_imgproc/py_template_matching/py_template_matching.html