universidad de león

Escuela de Ingenierías

Visión Artificial - Conteo y aforo

Prácticas externas

Alumno: Sergio Chimeno Alegre

Grado en ingeniería informática

Índice

Datos de la práctica

Título del proyecto: Visión Artificial - Conteo y aforo
Empresa: Proconsi
Tutores:

- David García Prieto - david.garcia@proconsi.com
- David Jáñez Gonzalez – david.janez@proconsi.com

Objetivos

Familiarizarse con el lenguaje Python y la librería OpenCV

Decidir la estrategia a seguir para realizar el conteo de personas, tras haber analizando pros y contras de cada una de ellas.

Implementar un sistema capaz de contar personas en una determinada zona mediante cámara de vídeo preferentemente cenital (por ejemplo desde un drone). Debe funcionar en diferentes escenarios (playas, calles, conciertos.....)

Análisis de estrategias

Estrategia #1:Haar Cascade Classifier

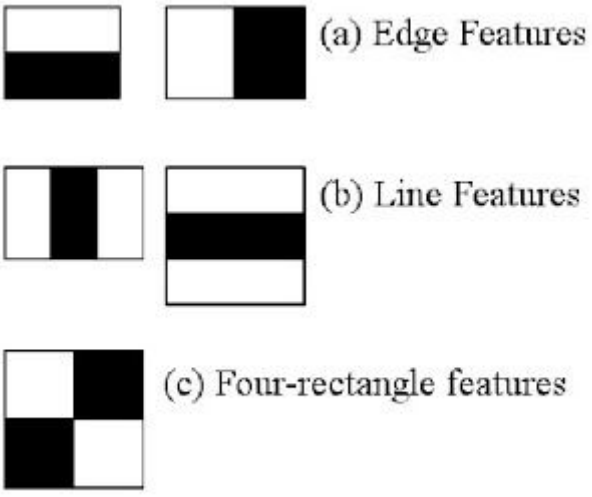
Para esta estrategia usaremos Haar feature-based cascade classifier para detectar objetos, en este caso personas.

Este es un metodo basado en machine learning, donde una función cascada es entrenada a partir de un dataset de imagenes. Una vez entrenado el clasificador se puede utilizar para detectar objetos.

Opencv ya nos provee una serie de modelos preentrenados: https://github.com/opencv/opencv/tree/master/data/haarcascades ; uno de estos modelos(haarcascade_fullbody.xml) puede ser útil para la detección de personas.

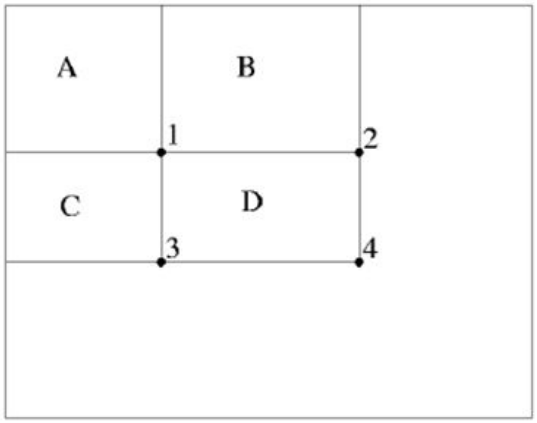
Este clasificador funciona con imagenes en escala de grises. Este algoritmo hace uso de: Haar Features, Integral images, adaboost y cascading.

Haar features: Son equivalentes a aplicar convolutional kernels, como los de la imagen de abajo, donde la región negra de la matriz son -1 y la región blanca son 1; es decir, es como restar la parte negra de la parte blanca. Estos kernels se aplicarán en múltiples tamaños a lo largo de la imagen.



Puesto que aplicar estos kernels requieren mucho coste computacional(muchas operaciones) se introduce el uso de las **integrales de imágenes**, que son un método que permite encontrar la suma de todos los píxeles bajo un rectángulo, con solo saber los puntos de las esquinas del rectángulo.

En una integral image el valor del pixel(x,y) es la suma de todos los pixeles que están por encima y a la izquierda.
La integral de una imagen solo es necesario calcularla una vez.
Una vez se tiene la integral de una imagen la suma de pixeles se realiza así:

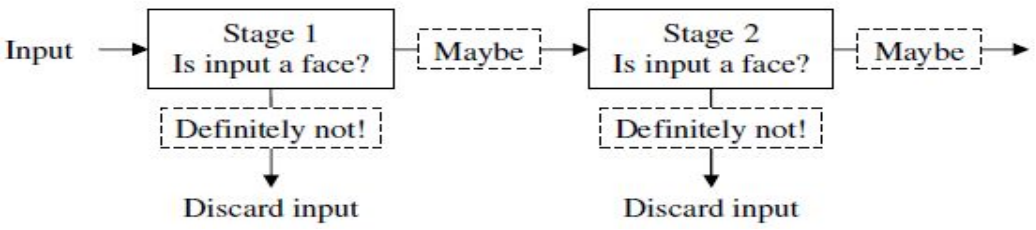


$$D = 1+4-(2+3)$$
$$= A+(A+B+C+D)-(A+C+A+B)$$
$$= D$$

En una imagen de 24+24 habría 160000 Haar features. Pero la mayoría de estas características son irrelevantes. Por lo tanto se hace una subselección de las mejores Haar features con **Adaboost,** que hace lo siguiente:

1. Entrena una serie de clasificadores débiles(un clasificador débil es aplicar una Haar feature y decidir a partir de una treshold) y elige los mejores.
2. Crea una clasificador fuerte como una weighted sum de clasificadores débiles

Por último, se usa **cascading,** que lo que hace es aplicar los clasificadores débiles en grupos, de manera que si no se cumple con lo mínimo requerido por un grupo de clasificadores débiles se pueda descartar esta clasificación y continuar con otra, ahorrando así coste computacional.



Ventajas:

- Nos permite saber la posicion excata de las personas

- Puede adaptarse a múltiples escenarios si el clasificador está bien entrenado
- Es un métedo de detección de objetos rápido(ya era fluido con los dispositivos que había en 2004)
- Ya hay modelos entrenados

Desventajas:

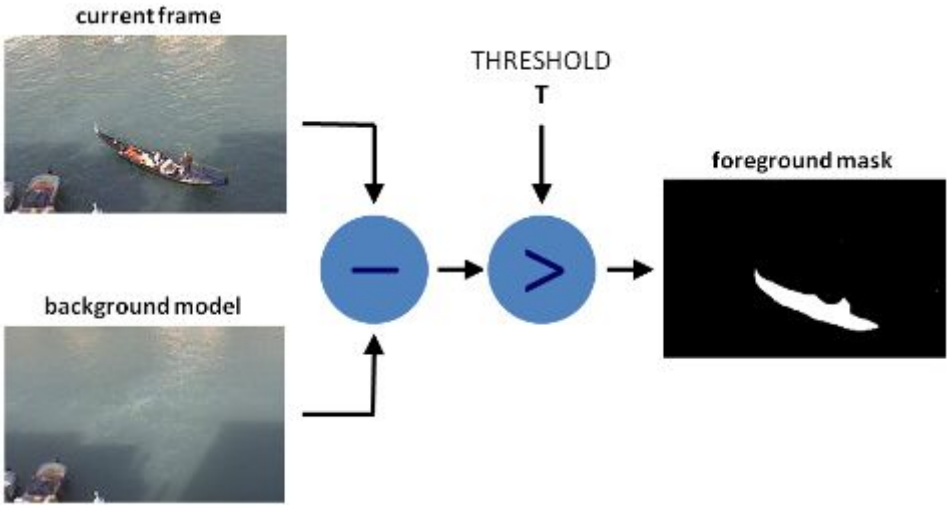
- Puede que las Haar features no se adapten a todos los escenarios

Estrategia #2: Background subtraction method

Background substraction es un metodo que nos permite generar una máscara, de los objetos en primer plano(foreground mask)

Algoritmo de background substraction:

1. Estimar el fondo(background)
2. Obtener el valor absoluto de la resta entre el fondo estimado y el frame actual
3. Aplicar la función threshold con un valor T, para obtener la máscara como una imagen binaria



Puede haber multiples formas de estimar el fondo:

- El frame anterior al actual
- La media de los n ultimos frames
- …

Opencv nos ofrece MOG2 y KNN como background subtractors

Algunos parámetros que podemos modificar son:

- varThreshold: (T) el valor del treshold
- detectShadow: si queremos que detecte las sombras

Además de esto habría que limpiar algo de ruido con transformaciones morfológicas o blurs.

Y por último decidir cómo diferenciar una persona de cualquier otro objeto en movimiento. Es decir, como diferenciar la marscara de un coche de la de una persona.
Un enfoque simple sería considerar como persona cuando hay un cierto rango de pixeles blancos juntos.
Otro enfoque sería usar un clasificador cascada y entrenarlo con máscaras de personas

Ventajas:

- Requiere poco procesamiento
- Funciona bien para detectar objetos que se mueven
- Puede funcionar en cámaras en movimiento(dron) si se actualiza la estimación del background

Desventajas:

- Con objetos estáticos, puede que funcione peor, por ejemplo si una persona no se mueve de un sitio será más dificil detectarla, ya que esta persona será detectada como parte del fondo
- Si dos personas están juntas puede ser dificil diferenciarlas
- Se pueden confundir otros objetos que se mueven con personas

Estrategia #3: HOG + SVM

Ventajas:

- Opencv ya viene con modelos preentrenados para detectar peatones en imágenes

Desventajas:

Bibliografía

- https://docs.opencv.org/master/d9/df8/tutorial_root.html
- https://www.youtube.com/watch?v=WfdYYNamHZ8