Universidad de Monterrey
Facultad: Ingeniería y Tecnologías
Visión Computacional

Contador de objetos y círculos

Alejandro Acosta Murillo #616929

“Doy mi palabra de que he realizado esta actividad con integridad académica”

Introducción:

Dentro de la industria, la detección de formas geométricas es una técnica ampliamente utilizada en el procesamiento de imágenes y visión por computadora. Para la detección de círculos se utilizó el método de **Hough Circles**, esta técnica permite identificar presencia y la ubicación de círculos, más específicamente este proceso convierte las coordenadas espaciales de una imagen en un espacio de parámetros, facilitando la detección de patrones geométricos.

Algunas de las principales aplicaciones prácticas de este tema se encuentran:

- Detección de objetos circulares en imágenes industriales para inspección de calidad.
- Reconocimiento de señales de tráfico en sistemas de vehículos autónomos.
- Seguimiento de objetos en deportes o videovigilancia.

A continuación, se muestra el código utilizado, así como una breve explicación del mismo.

Descripción del método:

Dentro del siguiente código se tienen diferentes etapas para preparar para la detección de los círculos, así como la impresión de los resultados una vez se hizo el correcto procedimiento. Primero que nada, se carga y procesa la imagen, por temas de tamaños de imagen se tiene que realizar una reducción a escala, así como después para procesarla se convierte a escalas de grises.

Después de esto se aplica un desenfoque gaussiano, el cuál ayuda a suavizar la imagen y reducir el ruido, haciendo que la detección de los círculos sea más precisa, evitando falsos positivos debido a pequeñas imperfecciones que se encuentran en las imágenes.

El paso clave de este código es la detección de círculos con la transformada de Hough, en la cual se tienen que entregar datos como la relación de resolución, la distancia mínima entre los centros de los círculos detectados, parámetros de los umbrales para la detección de bordes y el radio mínimo y máximo de los círculos que se desean detectar.

Por último, se detectan los círculos y se imprime en la imagen la cantidad de los círculos detectados.

Código para detección de círculos:

```
import cv2

import numpy as np


# Cargamos la imagen

image = cv2.imread('letrero.jpeg',cv2.IMREAD_COLOR)
```

```
image = cv2.resize(image,None,fx=0.75,fy=0.75)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Aplicamos desenfoque gaussiano para reducir el ruido y realizar una mejor detección
gray_blurred = cv2.GaussianBlur(gray,(9,9),2)

# Detectamos los círculos utilizando la transformada de Hough
circles = cv2.HoughCircles(gray_blurred, cv2.HOUGH_GRADIENT, dp =1.2,
minDist=30,param1=100, param2=30,
                    minRadius=15,maxRadius=50)

# Si se detectan circulos, dibujarlo
if circles is not None:
  circles = np.round(circles[0,:]).astype("int")

  for(x,y,r) in circles:
    # Dibujamos el círculo
    cv2.circle(image, (x,y),r,(0,255,0),4)

    # Dibujamos el centro del círculo
    cv2.circle(image, (x,y),2,(0,128,255),3)

# Obtener la dimensión del arreglo
dimensiones = circles.shape[1]
cv2.putText(image, f"La cantidad de circulos detectados es: {dimensiones}", (10, image.shape[0] -
10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

# Mostramos la imagen con los circulos detectados
cv2.imshow("Círculos detectados", image)
```

```
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Código para detección de personas:

```
import cv2

# Inicializamos el detector HOG con un clasificador preentrenado para personas

hog = cv2.HOGDescriptor()

hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())


def detectar_y_contar_personas(imagen):

  # Redimensionamos la imagen para acelerar el proceso de detección (opcional)

  imagen_redimensionada = cv2.resize(imagen, (min(800, imagen.shape[1]), min(600, imagen.shape[0])))


  # Detectamos personas en la imagen

  (rectangulos, _) = hog.detectMultiScale(imagen_redimensionada, winStride=(4, 4), padding=(8, 8), scale=1.05)


  # Dibujamos los rectángulos alrededor de las personas detectadas

  for (x, y, w, h) in rectangulos:

    cv2.rectangle(imagen_redimensionada, (x, y), (x + w, y + h), (0, 255, 0), 2)


  # Contamos la cantidad de personas detectadas

  cantidad_personas = len(rectangulos)


  # Escribimos la cantidad de personas detectadas en la imagen

  cv2.putText(imagen_redimensionada, f"Personas detectadas: {cantidad_personas}", (10, imagen_redimensionada.shape[0] - 10),

        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
```

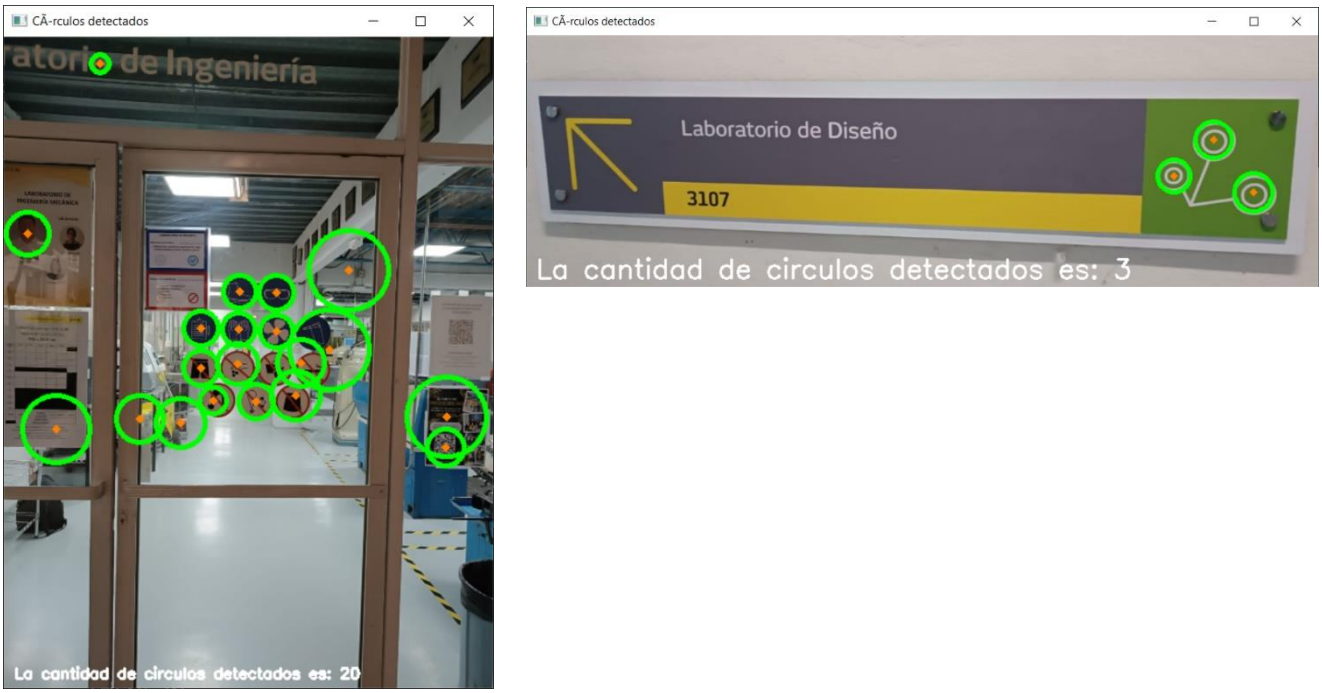
```
    return imagen_redimensionada, cantidad_personas


# Cargar la imagen donde se desea detectar personas
imagen = cv2.imread('ruta/a/tu/imagen.jpg')


# Detectar y contar personas en la imagen
imagen_resultante, cantidad_personas = detectar_y_contar_personas(imagen)


# Mostrar la imagen con las personas detectadas y el conteo
cv2.imshow("Detección de Personas", imagen_resultante)
print(f"Cantidad de personas detectadas: {cantidad_personas}")
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Descripción del método:

Como primera parte importamos la librería de OpenCV, el cuál funciona para el procesamiento de imágenes y visión computacional, además, se utilizó un detector HOG, creando objetos, que es un descriptor de características basadas en el Histograma de Gradientes Orientados, además se configura el detector HOG para usar un SVM pre-entrenado que está optimizado para detectar persona en imágenes.

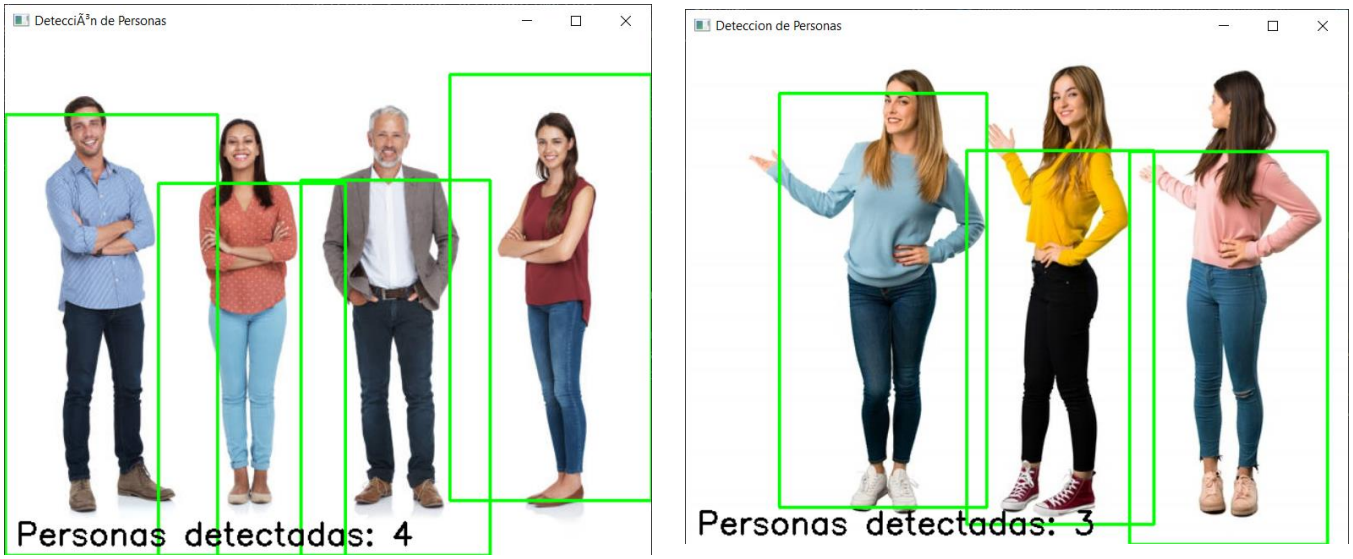
Después se define la función, en la cual se redimensiona la imagen, se detecta la persona utilizando HOG, en donde se agregan datos como el tamaño del paso de la ventana deslizante durante la detección, añade un margen alrededor de las ventanas de detección para evitar falsos positivos, y controla la escala de la imagen en cada iteración de la detección, lo que ayuda a identificar personas de diferentes tamaños. Además, dibuja rectángulos de las personas detectadas, y cuenta la cantidad de personas, añadiendo el texto con el número de personas detectadas, después llamas a la función, cargando la imagen que gustes detectar, además muestra la imagen con estas personas detectadas.

Resultados detección de círculos:

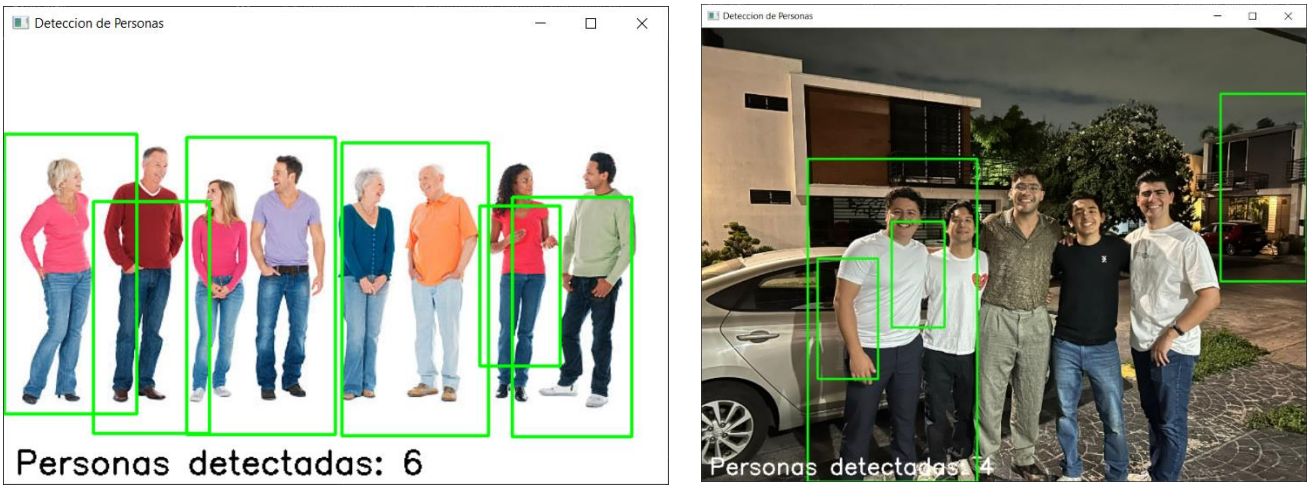


Figuras 1.1. Resultados de la detección de círculos

Resultados detección de personas:



Figuras 1.2. Resultados favorables de la detección de personas



Figuras 1.3. Resultados poco favorables de la detección de personas

Como se puede observar en los resultados, es necesario que las imágenes estén bien definidas, así como un fondo y separación notables, tanto para la detección de los círculos como para las personas. En las figuras anteriores (*Figuras 1.1*) se encuentran dos de las pruebas en las que se utilizó el código, así como en las *Figuras 1.2* en donde se muestran unas pruebas que se realizaron con imágenes de personas con fondos unicolor, aunque para probar el punto que se comentaba *en las Figuras 1.3,* se muestran varias pruebas que se realizaron en entornos no favorables y con muchas personas, mostrando falsos positivos y detecciones que no se realizaron.

Conclusiones:

En los últimos dos programas que se mostraron, se destacan enfoques clave en visión por computadora utilizando técnicas avanzadas de procesamiento de imágenes. El primer programa se centra en la detección y conteo de personas mediante un modelo preentrenado HOG+SVM, lo que demuestra cómo los modelos tradicionales pueden ser efectivos para identificar objetos en imágenes de manera precisa y rápida. En este caso, se optimiza el uso de ventanas deslizantes, redimensionado de imágenes y parámetros para mejorar la eficiencia y precisión.

El segundo programa se enfoca en la detección de círculos usando la transformada de Hough, un método clásico de visión computacional. Ambos códigos, aunque usan diferentes enfoques y técnicas, destacan la importancia de adaptar los parámetros y algoritmos a las necesidades específicas del problema a resolver, lo que subraya la versatilidad de OpenCV para el procesamiento de imágenes en aplicaciones tanto básicas como avanzadas.