Centro Universitario de Ciencias Exactas e Ingenierías

Asignatura: **Seminario Inteligencia Artificial**

Sección: D04

Proyecto Correlación Cruzada Normalizada

Alumno: Luis Jaime Portillo Correa

Código: 217546155

Profesor: **Javier Enrique Gómez Ávila**

Fecha: **30/11/2023**

Objetivo

Esta propuesta consiste en resolver la detección de plantillas utilizando la ecuación de NCC. Las recomendaciones para esta propuesta son:

- Puedes utilizar las imágenes y plantillas proporcionadas en el ejemplo o utilizar tus propias imágenes. En el caso de utilizar tus propias imágenes, procura extraer la plantilla de la imagen original. Si se requiere cambiar la resolución de alguna imagen, se debe cambiar para ambas, la imagen y la plantilla en la misma proporción.
- Las restricciones del espacio de búsqueda son las siguientes:

$$\begin{aligned} \mathbf{x}_l &= \begin{bmatrix} 1 & 1 \end{bmatrix}^T \\ \mathbf{x}_u &= \begin{bmatrix} W-w & H-h \end{bmatrix}^T \end{aligned}$$

donde W y H son el ancho y alto de la Imagen, w y h son el ancho y alto de la plantilla, respectivamente.

- Es necesario convertir la imagen rgb a escala de grises para aplicar la ecuación de NCC.

Desarrollo

Para la implementación de esta última práctica, o mejor dicho proyecto, fue necesario completar un ejercicio de Correlación cruzada normalizada. En el que la idea principal era conseguir encontrar puntos coincidentes de una imagen_1 con un fondo denominado Template, esto fue posible tras completar el algoritmo de Evolución Diferencial que el profe nos proporcionó.

Tras varias horas de intentos con distintos parámetros, pude identificar un conjunto de valores que me dieron resultados correctos, al momento de la ejecución en tiempo real parece ser tardado pero es el equilibrio que logré conseguir pues con otras configuraciones el algoritmo no era capaz de encontrar los resultados correctos.

```
[7]  def DE(img, temp, animacion):
         # Parámetros del algoritmo DE
         n_Gen = 80
         n_Pop = 50
         dim = 2
         F = 0.9  # Factor de escala para la mutación
         Cr = 0.9  # Tasa de recombinación
```

En el bucle principal , el código se encarga de mostrar en pantalla los posibles resultados que encuentra con las distintas generaciones que están en búsqueda de la solución.

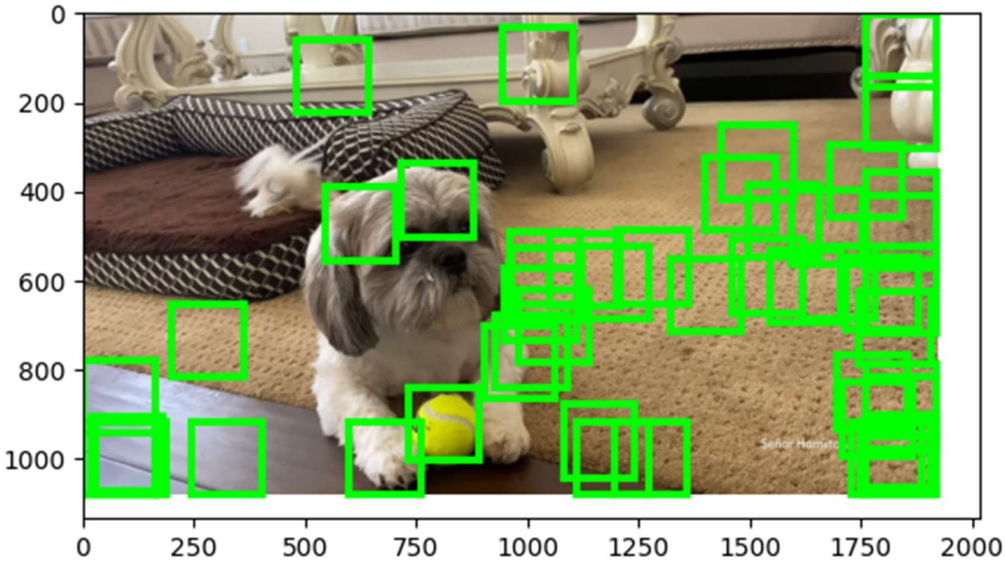
```
# Bucle principal de DE
for n in range(n_Gen):
    if animacion:
        img_2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.clf()
        display.display(plt.gcf())
        display.clear_output(wait=True)
        plt.imshow(img_2)

        for i in range(n_Pop):
            # Dibujar rectángulos en la animación
            plt.plot([x[0, i], x[0, i] + temp_W], [x[1, i], x[1, i]], color=(0,1,0), linewidth=3)
            plt.plot([x[0, i], x[0, i]], [x[1, i], x[1, i] + temp_H], color=(0,1,0), linewidth=3)
            plt.plot([x[0, i] + temp_W, x[0, i] + temp_W], [x[1, i], x[1, i] + temp_H], color=(0,1,0), linewidth=3)
            plt.plot([x[0, i], x[0, i] + temp_W], [x[1, i] + temp_H, x[1, i] + temp_H], color=(0,1,0), linewidth=3)
        plt.show(block=False)
        plt.pause(.05)
```

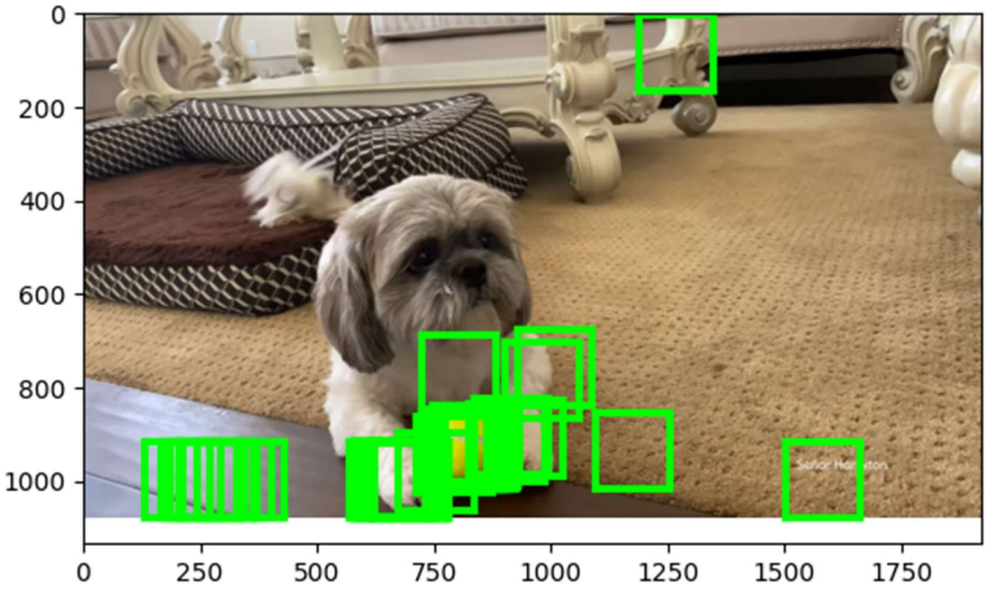
En esta parte del algoritmo implementé los pasos de la mutación, recombinación y selección en base a la dimensión del problema.

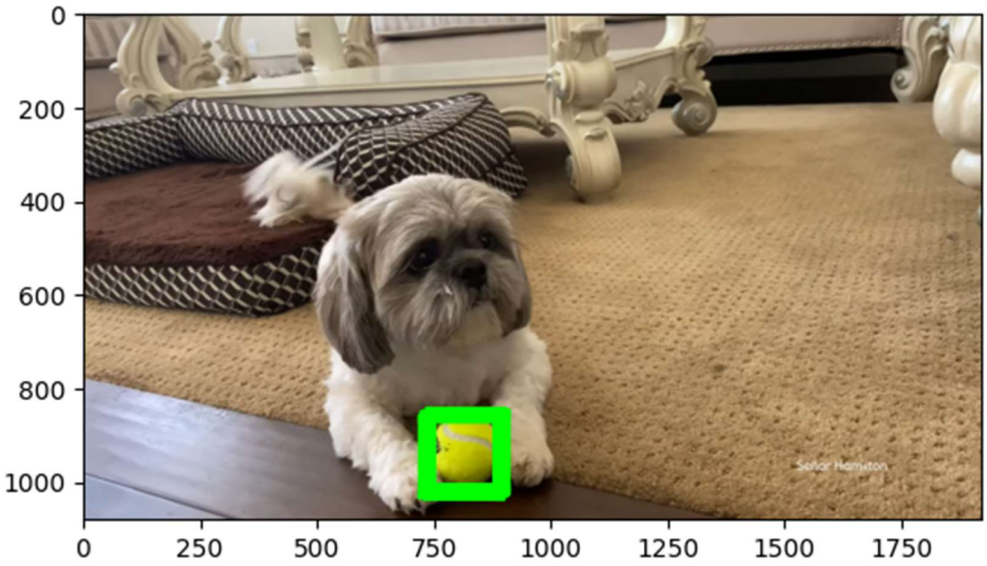
```
  for i in range(n_Pop):
      # Mutación
      r1, r2, r3 = np.random.choice(n_Pop, 3, replace=False)
      mutant = x[:, r1] + F * (x[:, r2] - x[:, r3])

      # Garantizar que los valores estén dentro de los límites
      mutant = np.maximum(lb, np.minimum(ub, mutant))

      # Recombinación
      crossover = np.random.rand(dim) < Cr
      if not np.any(crossover):
          crossover[np.random.randint(dim)] = True

      # Selección
      trial = np.where(crossover, mutant, x[:, i])
      trial_fitness = NCC(img_g, temp_g, int(trial[0]), int(trial[1]))

      # Actualizar la población si el nuevo individuo es mejor
      if trial_fitness > fitness[i]:
          x[:, i] = trial
          fitness[i] = trial_fitness
```

Resultados

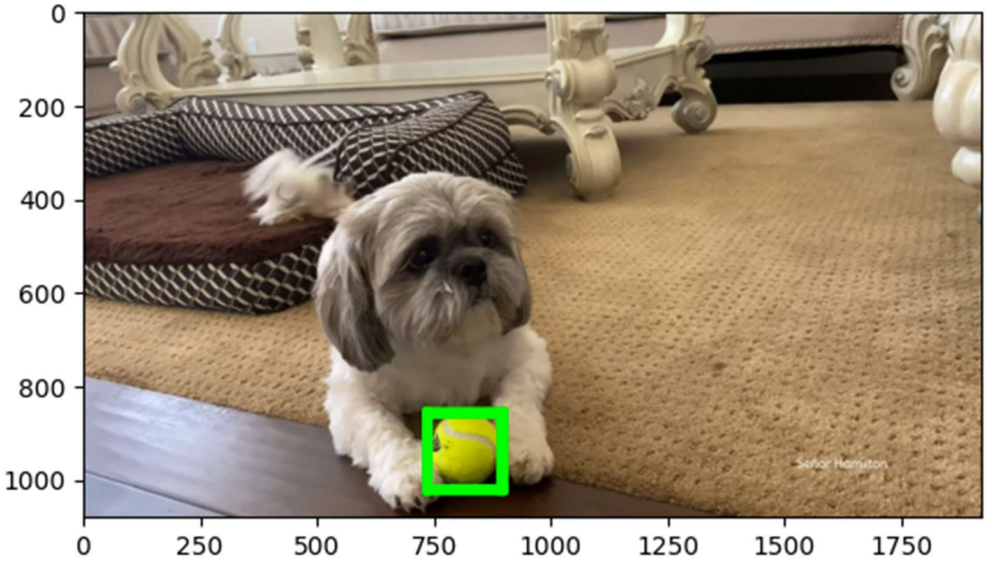
Capturas de la animación:

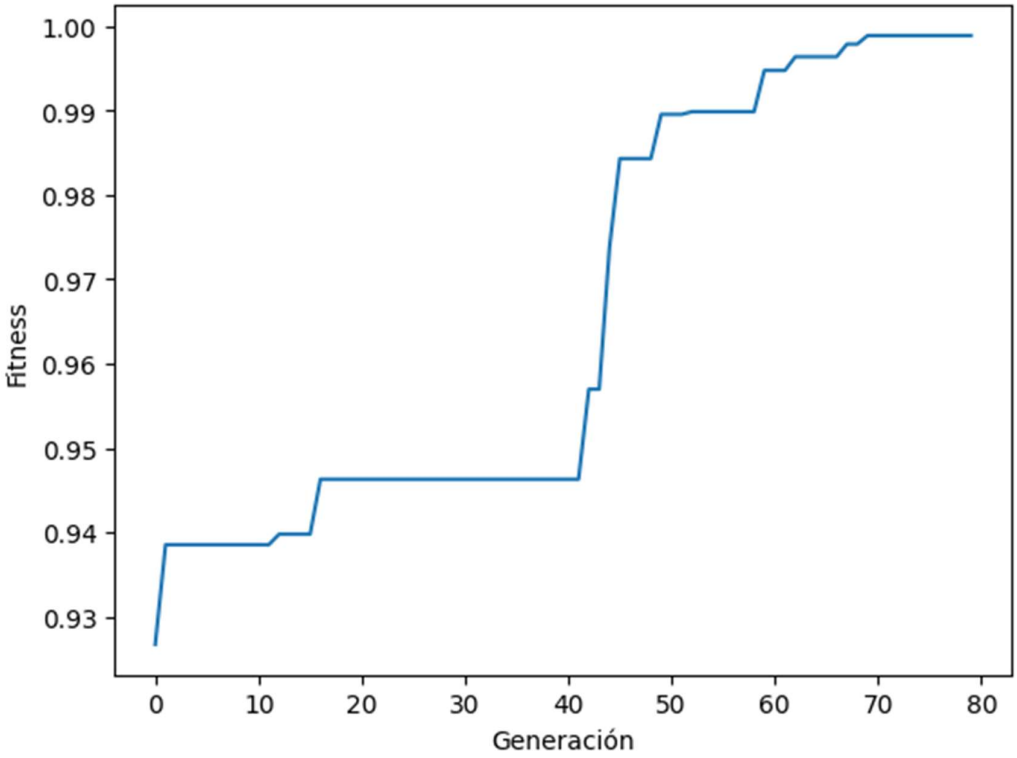


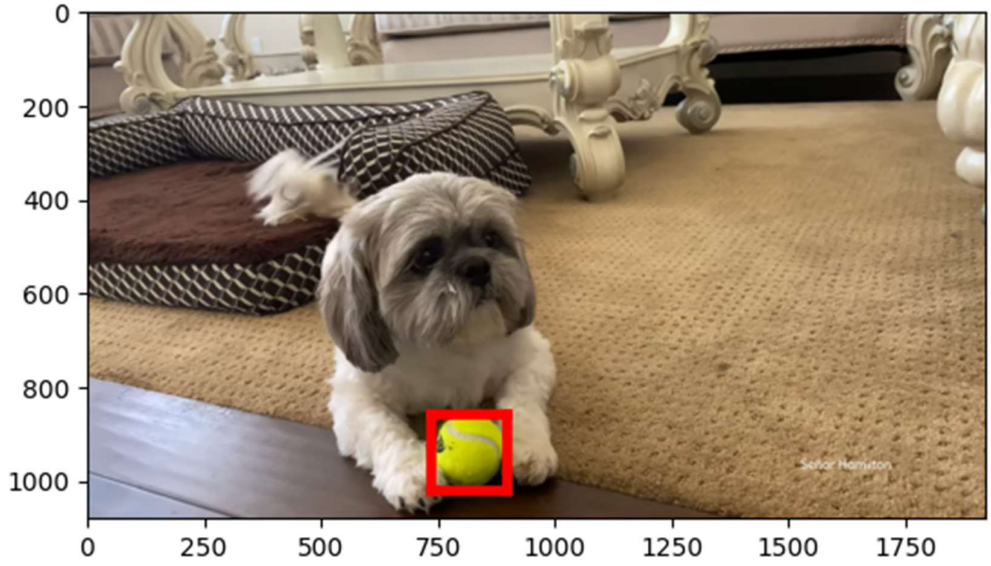




Resultados finales:




1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
Fitness
0
10
20
30
40
50
60
70
80
Generación


0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750