Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Inteligencia Artificial

Tarea Corta 02 - Del pincel al píxel

Realizado por:

Jennifer Alvarado Brenes c. 2020124171

Evelyn Cruz Solís c. 201251606

Profesor:

Kenneth Roberto Obando Rodriguez

II Semestre 2024

Contenido

1. Implementación

Para poder realizar el algoritmo genetico fue necesario entender el flujo y funcionamiento del mismo, por lo cual nos es importante mantener una estructura acorde al mismo. Como equipo se trabajo el siguiente flujo se para el algoritmo, que se explicará brevemente:

Es necesario una población, se crea una población inicial con TAM_POBLACION individuos, cada uno compuesto por NUM_POLIGONO polígonos aleatorios. Luego llevar a cabo una evaluación de la Población Inicial, donde cada uno de los individuos es comparado con la imagen que generan con la imagen de referencia, de esta manera se obtiene el fitness de un individuo siendo la suma de las diferencias absolutas entre las dos imágenes. Entre menor sea la diferencia mejor será el fitness.
Para un adecuado algoritmo genético, es importante el avance de las generaciones, para su "evolución" con el fin de acercarse al objetivo. Durante cada generación se realiza:

a. Selección de padres usando un torneo aleatorio para seleccionar los que tienen mejor fitness.
b. Se realiza un cruce entre los padres más aptos para crear los nuevos individuos de la población para la siguiente generación.
c. A los individuos hijos, se les aplica una mutación en función del tamaño de los polígonos que es el ADN.
d. Se evalúan los nuevos individuos y se seleccionan los mejores para formar la nueva población.

Con el algoritmo genético se busca que en cada generación, la población vaya mejorando gradualmente al seleccionar los mejores individuos, cruzarlos y mutarlos. Los polígonos se van ajustando en forma, tamaño y color para parecerse más a la imagen de referencia.

1.1 Parámetros del Algoritmo

- POPULATION_SIZE: Número de individuos en la población.
- NUM_POLYGONS: Cantidad de polígonos que componen cada individuo.
- VERTICES_PER_POLYGON: Número de vértices por polígono.
- MUTATION_RATE: Tasa de mutación de los individuos.
- GENERATIONS: Número máximo de generaciones.
- IMAGE_SIZE: Dimensiones de la imagen procesada.
- INITIAL_POLYGON_SIZE y FINAL_POLYGON_SIZE: Tamaños inicial y final de los polígonos que se ajustan a lo largo de las generaciones.

1.2 Preprocesamiento de la Imagen

```
# Cargar imagen de referencia en escala de grises y ajustar tamaño
reference_image = Image.open("imagen.png").convert("L")
reference_image = reference_image.resize(IMAGE_SIZE)
reference_array = np.array(reference_image)
```

Inicialmente, la imagen de referencia se convierte a una escala de grises. Además, se hace pequeña para que el tiempo de procesamiento no sea muy grande.

1.3 Generación de Individuos y Evaluación

La generación de individuos se realiza componiendo un número dado de polígonos generados aleatoriamente. La función de evaluación calcula la diferencia entre la imagen generada y la imagen de referencia:

```
def create_individual(size):
    return [(random_polygon(size), random.randint(0, 255)) for _ in
range(NUM_POLYGONS)]

def evaluate_individual(individual):
    drawn_image = draw_individual(individual)
    return np.sum(np.abs(reference_array - drawn_image))
```

1.4 Selección, Cruza y Mutación

El proceso de selección se realiza cruzando dos progenitores para formar un nuevo hijo. Los polígonos se modifican mutando gradualmente para fomentar la diversidad en la población.

```
def tournament_selection(population, k=3):
    return min(random.sample(population, k), key=lambda ind: ind[1])

def crossover(parent1, parent2):
    point = random.randint(1, len(parent1) - 1)
    child = parent1[:point] + parent2[point:]
    return child

def mutate(individual, size, generation):
    for i in range(len(individual)):
        if random.random() < MUTATION_RATE:
            new_size = calculate_polygon_size(generation) + random.randint(-5, 5)
            if random.random() < 0.5:
                individual[i] = (random_polygon(new_size), random.randint(0, 255))
            else:
                duplicated_polygon = individual[i][0][:]
                individual.append((duplicated_polygon, random.randint(0, 255)))
                if len(individual) > NUM_POLYGONS:
                    individual.pop(random.randint(0, len(individual) - 1))
    return individual
```

1.5 Visualización y Animación

La animación se crea utilizando **matplotlib**, mostrando la evolución de la mejor aproximación de la imagen en cada generación.

```
# Preparar la animación
fig, ax = plt.subplots()
ax.axis('off')
img_display = ax.imshow(draw_individual(population[0][0]), cmap='gray', vmin=0,
vmax=255)

def update(frame):
    global population
    current_polygon_size = calculate_polygon_size(frame)
    new_population = []
    for _ in range(POPULATION_SIZE):
        parent1 = tournament_selection(population)
        parent2 = tournament_selection(population)
        child = crossover(parent1[0], parent2[0])
        new_population.append((child, 0))
    new_population = [(mutate(individual, current_polygon_size, frame), 0) for
individual, _ in new_population]
    new_population = [(individual, evaluate_individual(individual)) for individual, _ in
new_population]
    population = sorted(new_population + population, key=lambda ind:
ind[1])[:POPULATION_SIZE]
    best_individual, _ = population[0]
    img_display.set_data(draw_individual(best_individual))
    ax.set_title(f'Generación {frame + 1}')
ani = FuncAnimation(fig, update, frames=GENERATIONS, repeat=False)
plt.show()
```

2. Experimentación y ajuste de parámetros

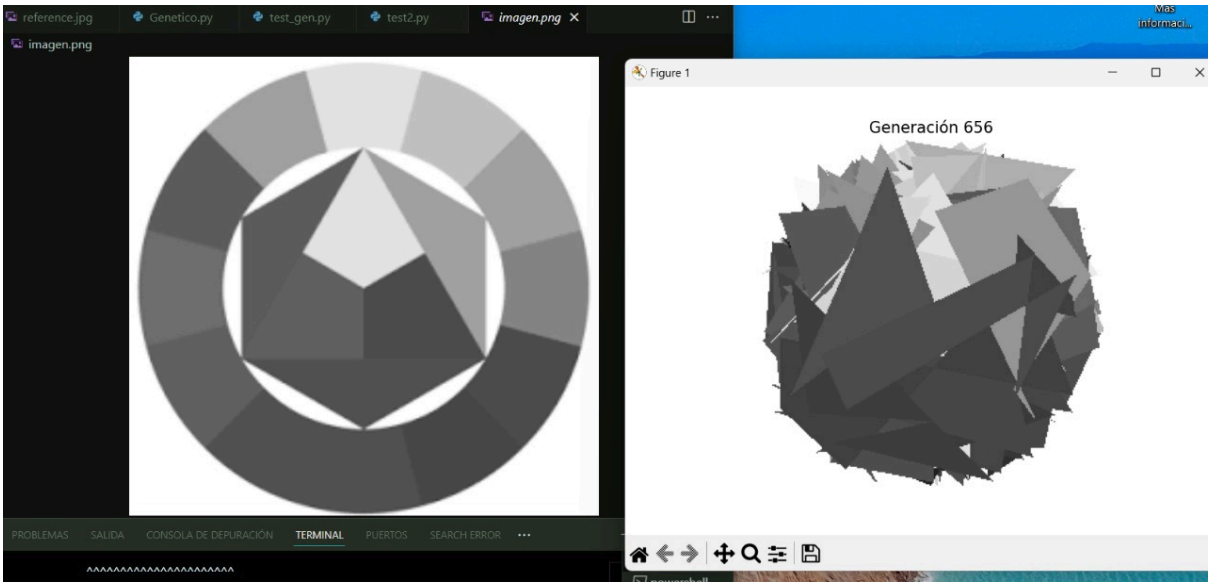
Se realizaron experimentos ajustando los parámetros como la tasa de mutación, el número de generaciones, y la cantidad de individuos por población. Se analizaron los resultados utilizando estos parámetros en la calidad final de la imagen, utilizando métricas de similitud para evaluar los resultados.

3. Resultados obtenidos

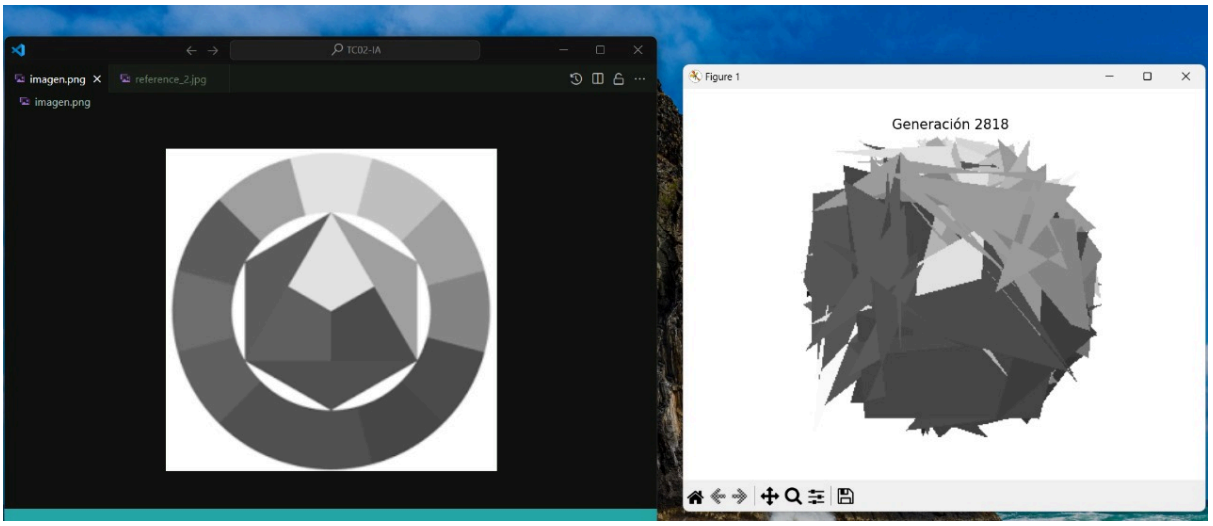
En las imágenes presentadas, se observa el desempeño del algoritmo genético diseñado para transformar una imagen en una representación artística utilizando polígonos en escala de grises. A continuación, se detallan los puntos clave del análisis de resultados basados en las capturas de las generaciones y la comparación con la imagen de referencia, además de algunos de los resultados obtenidos:

Imagen 1

- Generación 656:



- Generación 2818:



- Generación 5000:

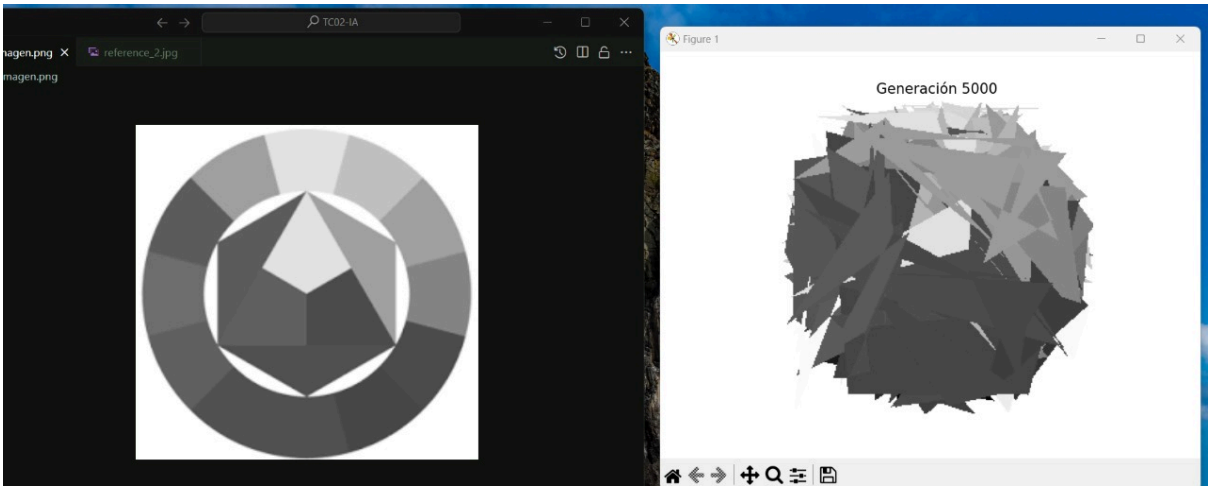


Imagen 2

- Generación 345:

- Generación 2057:

- Generación 5000:

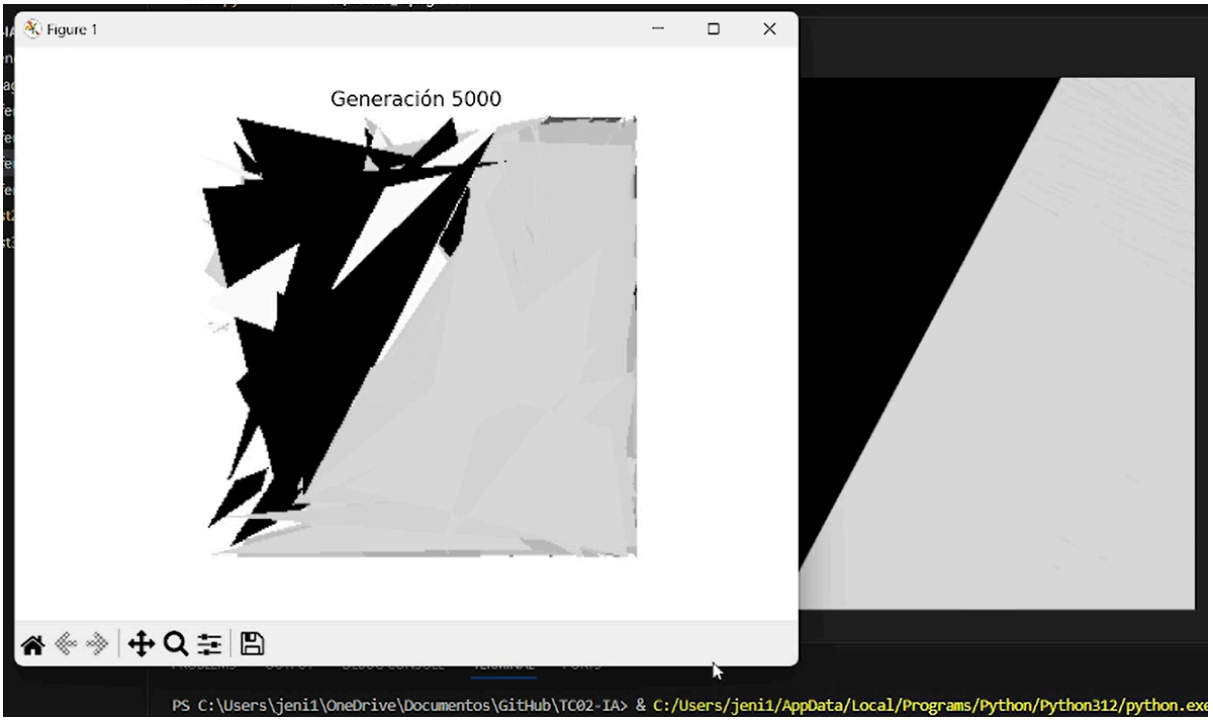
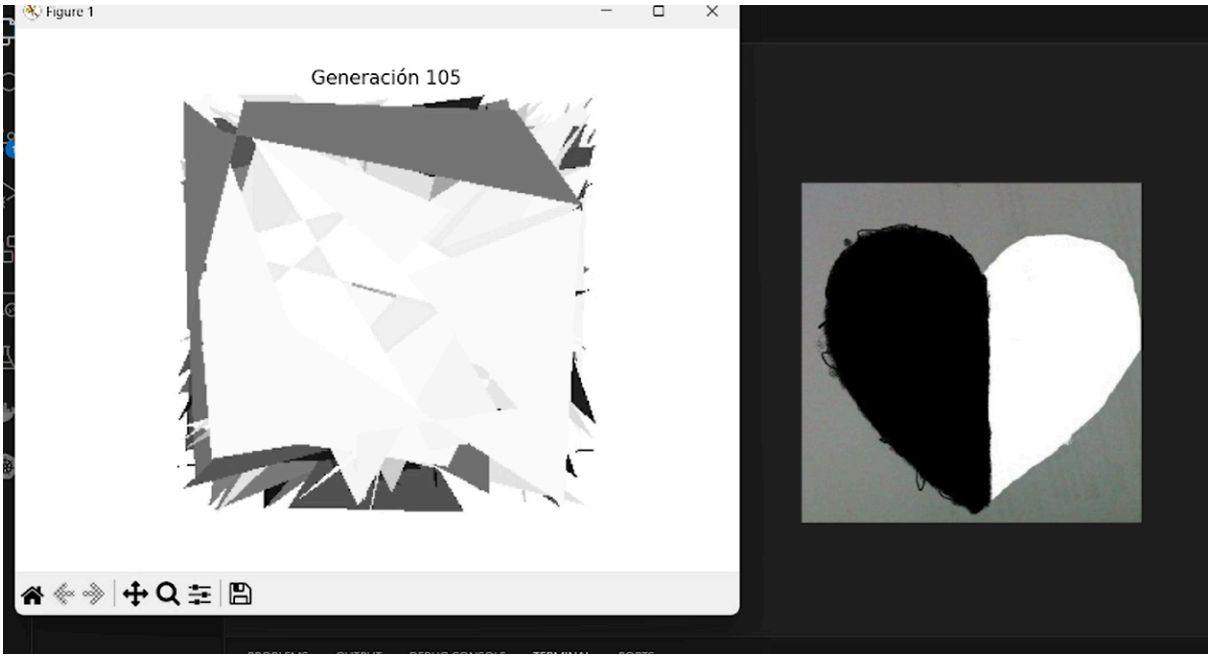


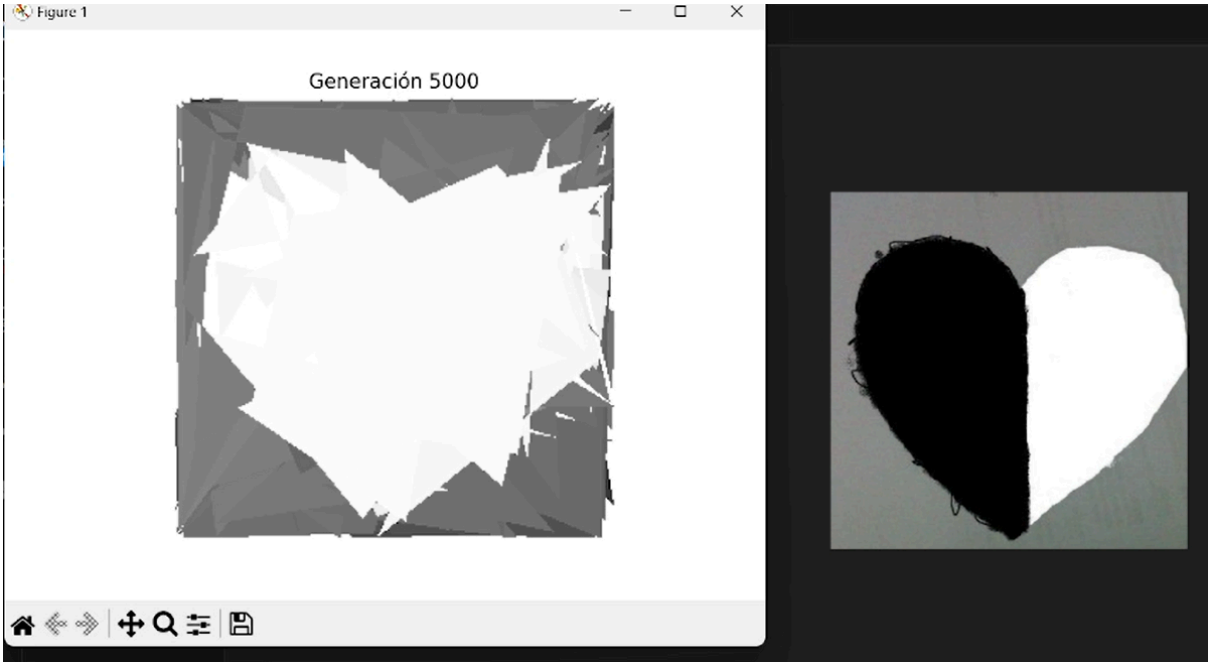
Imagen 3

- Generación 105:



- Generación 2051:

- Generación 5000:



- **Imagen de Referencia vs. Resultados Generados**:
    - La imagen de referencia muestra una figura geométrica compuesta por varios segmentos de diferentes tonos de gris, organizados de manera ordenada y simétrica.
    - En contraste, las imágenes generadas por el algoritmo muestran una representación caótica y menos estructurada, evidenciando que el algoritmo aún no logra capturar los patrones exactos y la simetría de la imagen de referencia.

- **Progresión a lo Largo de las Generaciones**:
    - **Generación inicial**: En esta etapa inicial, se observan polígonos con tamaños y orientaciones variadas, pero sin una estructura clara. La imagen muestra un grado de similitud con la referencia en términos de tonos de gris, pero los polígonos están dispersos sin una organización definida, lo que contribuye a una representación desordenada.
    - **Generación media**: En esta etapa intermedia, se observa más variedad en los polígonos tanto en sus formas y tamaños, como en sus tonos de la escala de gris. Si bien, aún no está cerca de parecerse a la imagen original, ya existe una forma más organizada de los polígonos.
    - **Generación 5000**: Aunque se evidencia una mejora en la densidad de los polígonos y una mayor cobertura del área central, la disposición sigue siendo desorganizada. Los operadores genéticos (selección, cruce, mutación) aún no están optimizados para reproducir patrones ordenados y geométricos. En

algunos casos, como en la imagen 2 y en la imagen 3, se nota una pérdida en las tonalidades más oscuras en zonas en donde deberían existir.

- **Similitudes y Diferencias Notables**:
    - **Similitudes**: El algoritmo ha logrado captar algunos tonos de gris similares a los de las imágenes de referencia, y la densidad de los polígonos aumenta progresivamente para cubrir más el espacio.
    - **Diferencias**: Las mayores diferencias radican en la organización y forma de los polígonos. La imagen generada no logra replicar la estructura simétrica y los ángulos definidos de la referencia, lo cual es muy importante para una evaluación de similitud visual. Además, la dispersión y superposición irregular de los polígonos crean una percepción visual desalineada con la imagen objetivo.

- **Evaluación del Fitness y Ajustes Potenciales**:
    - La función de fitness, basada en la diferencia absoluta entre la imagen generada y la de referencia, puede no estar capturando adecuadamente la necesidad de orden y estructura. Un ajuste podría ocasionar superposiciones excesivas o falta de alineación.

4. Conclusión

Aunque el algoritmo genético muestra una progresión en la similitud tonal, aún enfrenta desafíos significativos en la reproducción de la estructura y simetría de la imagen de referencia. Se requiere una mayor exactitud en los parámetros y en la función de fitness para alcanzar resultados más cercanos a los objetivos artísticos establecidos.