Align two or more images by optimizing the transformation parameters , such as scaling , rotation , and translation , to achieve the best match between them

```
import numpy as np
import matplotlib.pyplot as plt

# --- Image generation ---
def create_image(size=128):
    img = np.zeros((size, size))
    s1, s2 = size // 4, 3 * size // 4
    img[s1:s2, s1:s2] = 1.0
    return img

I_ref = create_image()

def transform_image(img, angle_deg=10, scale=1.1, tx=5, ty=-5):
    angle = np.deg2rad(angle_deg)
    h, w = img.shape
    cy, cx = h / 2, w / 2
    y, x = np.indices((h, w))
    x = x - cx
    y = y - cy
    x_new = (x - tx) / scale
    y_new = (y - ty) / scale
    xr = np.cos(-angle) * x_new - np.sin(-angle) * y_new
    yr = np.sin(-angle) * x_new + np.cos(-angle) * y_new
    xr += cx
    yr += cy
    x0 = np.floor(xr).astype(int)
    y0 = np.floor(yr).astype(int)
    x1 = x0 + 1
    y1 = y0 + 1
    x0 = np.clip(x0, 0, w - 1)
    x1 = np.clip(x1, 0, w - 1)
    y0 = np.clip(y0, 0, h - 1)
    y1 = np.clip(y1, 0, h - 1)
    Ia = img[y0, x0]; Ib = img[y1, x0]; Ic = img[y0, x1]; Id = img[y1, x1]
    wa = (x1 - xr) * (y1 - yr); wb = (x1 - xr) * (yr - y0)
    wc = (xr - x0) * (y1 - yr); wd = (xr - x0) * (yr - y0)
    warped = wa*Ia + wb*Ib + wc*Ic + wd*Id
    return warped

# --- True transform ---
I_mov = transform_image(I_ref, angle_deg=12, scale=1.05, tx=8, ty=-5)
```

```
true_params = dict(angle_deg=12, scale=1.05, tx=8, ty=-5)

# --- Fitness ---
def fitness(I_ref, I_mov):
    return -np.mean((I_ref - I_mov) ** 2)

# --- Cellular setup ---
n_cells = 100
cells = []
for _ in range(n_cells):
    params = {
        "angle_deg": np.random.uniform(-20, 20),
        "scale": 1 + np.random.uniform(-0.2, 0.2),
        "tx": np.random.uniform(-15, 15),
        "ty": np.random.uniform(-15, 15),
    }
    warped = transform_image(I_mov, **params)
    f = fitness(I_ref, warped)
    cells.append({"params": params, "fitness": f})

# --- Optimization loop ---
n_iter = 150
alpha = 0.3
mutation_rate = 0.1
fitness_history = []

for it in range(n_iter):
    new_cells = []
    for i, cell in enumerate(cells):

        neighbors = [cells[np.random.randint(0, n_cells)] for _ in range(2)]
        best_neighbor = max(neighbors, key=lambda c: c["fitness"])

        new_params = cell["params"].copy()
        if best_neighbor["fitness"] > cell["fitness"]:
            for key in new_params:
                new_params[key] += alpha * (best_neighbor["params"][key] - cell["params"][key])

        # small random mutation
        if np.random.rand() < mutation_rate:
            new_params["angle_deg"] += np.random.uniform(-1, 1)
            new_params["tx"] += np.random.uniform(-0.5, 0.5)
            new_params["ty"] += np.random.uniform(-0.5, 0.5)
            new_params["scale"] *= (1 + np.random.uniform(-0.01, 0.01))

        warped = transform_image(I_mov, **new_params)
        f_new = fitness(I_ref, warped)
        new_cells.append({"params": new_params, "fitness": f_new})
```

```
    cells = new_cells
    best = max(cells, key=lambda c: c["fitness"])
    fitness_history.append(best["fitness"])
    if it % 20 == 0:
        print(f"Iter {it:03d} | Best fitness: {best['fitness']:.6f}")

# --- Show result ---
best = max(cells, key=lambda c: c["fitness"])
aligned = transform_image(I_mov, **best["params"])

plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1); plt.title("Reference"); plt.imshow(I_ref, cmap="gray"); plt.axis("off")
plt.subplot(1, 3, 2); plt.title("Moving"); plt.imshow(I_mov, cmap="gray"); plt.axis("off")
plt.subplot(1, 3, 3); plt.title("Aligned (Recovered)"); plt.imshow(aligned, cmap="gray"); plt.axis("off")
plt.figure(); plt.title("Fitness over iterations"); plt.plot(fitness_history); plt.xlabel("Iteration");
plt.ylabel("Best fitness"); plt.show()

print("\nTrue transform:", true_params)
print("\nRecovered transform:", best["params"])
```

```
Iter 000 | Best fitness: -0.035628
Iter 020 | Best fitness: -0.012862
Iter 040 | Best fitness: -0.006192
Iter 060 | Best fitness: -0.001687
Iter 080 | Best fitness: -0.001097
Iter 100 | Best fitness: -0.000968
Iter 120 | Best fitness: -0.000923
Iter 140 | Best fitness: -0.000921
```