Primena genetskog algoritma u kompresiji slika

Projekat na kursu „Naučno izračunavanje“

Matematički fakultet, Univerzitet u Beogradu

Nemanja Antić, 1100/2017

Filip Lazić, 1101/2017

Profesor: dr. Mladen Nikolić

Asistent: dr. Stefan Mišković

Uvod

- Genetski algoritam kao algoritam pretrage inspirisan procesom prirodne selekcije koji svojim radom akumulira znanje o prostoru pretrage kako bi došao do opšteg optimalnog rešenja
- Inicijalizacija
- Fitness funkcija
- Selekcija
- Crossover
- Mutacija

```
START
Generate the initial population
Compute fitness
REPEAT
    Selection
    Crossover
    Mutation
    Compute fitness
UNTIL population has converged
STOP
```

SSGA - Steady–state Genetic Algorithm

Genetski algoritam za kompresiju slika.

1. Ulazni parametri

- Veličina populacije - 65
- Maksimalni broj generacije - 500
- Dužina hromozoma (veličina bloka) - 256
- Blokovi piksela (m x n) - 4 x 4
- Verovatnoća mutacije - 0.1
- Datoteka koju treba kompresovati

```
img = cv2.imread("img.bmp")
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

n = 4
m = 4
codebook_size = 256
pop_size = 65
max_gen = 500
mutation_prob = 0.1
```

2. Deljenje slike u blokove

- Delimo sliku u blokove piksela (n x m)
- Povećavanjem blokova dobijamo veći stepen kompresije, ali gubimo na kvalitetu i obrnuto.

```
def divide_image(img, n, m):
    image_vectors = []
    for i in range(0, img.shape[0] - n + 1, n):
        for j in range(0, img.shape[1] -m + 1, m):
            image_vectors.append(img[i : i + m, j : j + m])
    image_vectors = np.asarray(image_vectors).astype(int)
    return image_vectors
```

3. Generisanje početnog „codebook“-a

- Generišemo „codebook“ tako što uzmemo **codebook_size** nasumično izabranih blokova piksela. Neka je to skup C.

```
def build_codebook(codebook_size, vector_dimension, image_vectors):
    codebook = np.zeros((codebook_size, vector_dimension)).astype(int)
    for i in range(0, codebook_size):
        m = np.random.randint(0, image_vectors.shape[0])
        codebook[i] = image_vectors[m].reshape(1, vector_dimension)

    return codebook
```

- Za svaki blok (n x m) piksela nalazimo najbliži element iz C i klasifikujemo ga na osnovu toga. Tada za svaki element iz C imamo klasu najbližih image blokova.

```
def image_block_classifier(image_vectors, codebook):
    index_vector = np.zeros((image_vectors.shape[0],1)).astype(int)
    for i in range(0, image_vectors.shape[0]):
        hpsnr = 0
        best_unit = 0
        for j in range(0, codebook.shape[0]):
            tpsnr = psnr(image_vectors[i].reshape(1, codebook.shape[1]), codebook[j])
            if (tpsnr > hpsnr):
                best_unit = j
                hpsnr = tpsnr
        index_vector[i] = best_unit
    return index_vector
```

- Funkcija koja odredjuje bliskost:

$$\text{PSNR} = 10 \log_{10}\left(\frac{255^2}{\text{MSE}}\right) \quad \text{MSE} = \frac{1}{k \times k}\sum_{i=1}^{p}\sum_{j=1}^{p}\left(X_{i,j} - Y_{i,j}\right)^2$$

```
def psnr(x, y):
    mse = np.square(x - y).mean()
    if mse == 0:
        return -1
    return int (10 * np.log10(255 * 255 / mse))
```