Diskretna kosinusna transformacija

Seminarski rad u okviru kursa
Naučno izračunavanje
Matematički fakultet

Bajić Ana

jun 2019.

Sažetak

U ovom radu prikazan je algoritam diskretne kosinusne transformacije na primerima JPEG kompresije *greyscale* slika i slika u boji, kao i audio kompresije.

Ključne reči — DCT, JPEG kompresija, Python

Sadržaj

1 Diskretna kosinusna transformacija

Postoji više različitih formula kojima je diskretna kosinusna transformacija (u daljem tekstu: DKT) predstavljena (od DKT-I do DKT-VIII). Najčešće, kada se kaže **DKT** misli se na DKT-II. Ta transformacija je data sledećom formulom:

$$X_k = \sum_{n=0}^{N_1} x_n \cos\left[\frac{\pi}{N}(n+\frac{1}{2})k\right] \qquad k = 0, \ldots, N_1$$

DKT-II transformacija je ekvivalentna (do na faktor dvojke) diskretnoj furijeovoj transfomaciji 4N realnih ulaza parne simetrije, pri čemu su elmenti na parnim indeksima jednaki nuli.

Inverzna diskretna kosinusna transformacija (u daljem tekstu IDKT) takođe ima više varijanti (u zavisnosti od toga koja se DKT koristila). Na primer, inverz DKT-I je DKT-I pomnožen sa $\frac{2}{N-1}$, inverz DKT-IV je DKT-IV pomnožen sa $\frac{2}{N}$. Inverz DKT-II (tj. standardne DKT) je transformacija DKT-III. Ona je data formulom:

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N_1} x_n \cos\left[\frac{\pi}{N}n(k+\frac{1}{2})\right] \qquad k = 0, \ldots, N_1$$

DKT je moguće vršiti i u više dimenzija. Višedimenziona DKT je kompozicija jednodimenzionih DKT primenjenih na svaku dimenziju.

2 Implementacija u jeziku *Python*

Biblioteka za *Python SciPy* u svom paketu *fftpack* ima implementacije za jednodimenzionalni DKT i IDKT. Za kompresiju zvuka 3.1 su korišćene ove funkcije.

Pomoću njih su implementirani 2D i 3D DKT i IDKT algoritmi korišćeni u kompresiji *greyscale* slika i slika u boji. Implementacije se mogu videti na slici 1 i slici 2. 3D transformacija je izvršena pomoću 2D transformacije po svakoj dimenziji.

```
def discrete_cosine_transform_2D(img):
    return dct(dct(img.T, norm='ortho').T, norm='ortho')

def inverse_discrete_cosine_transform_2D(coefficients):
    return idct(idct(coefficients.T, norm='ortho').T, norm='ortho')
```

Slika 1: Implementacija 2D transformacija.

```
def dct3d(img):
    out = np.empty(img.shape)

    out[:,:,0] = discrete_cosine_transform_2D(img[:,:,0])
    out[:,:,1] = discrete_cosine_transform_2D(img[:,:,1])
    out[:,:,2] = discrete_cosine_transform_2D(img[:,:,2])

    return out
```

```
def idct3d(img):
    out = np.empty(img.shape)

    out[:,:,0] = inverse_discrete_cosine_transform_2D(img[:,:,0])
    out[:,:,1] = inverse_discrete_cosine_transform_2D(img[:,:,1])
    out[:,:,2] = inverse_discrete_cosine_transform_2D(img[:,:,2])

    return out
```

Slika 2: Implementacija 3D transformacija.

3 Kompresije

3.1 Kompresija zvuka

3.2 Kompresija *greyscale* JPEG slika

Kada sliku u boji konvertujemo u *Numpy* niz, zapravo dobijamo trodimenzionalni objekat - matricu dimenzija *width* × *height* gde svaki piksel sadrži tri kanala za boju - *R, B, G*.

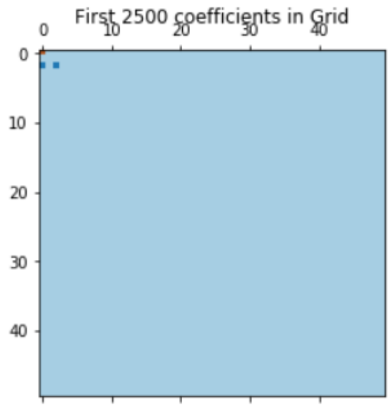
```
image = Image.open('slika.jpg')
image = image.resize((256, 256), 1)
image = np.array(image)

image.shape
```

```
(256, 256, 3)
```

Slika 3: Slika u boji

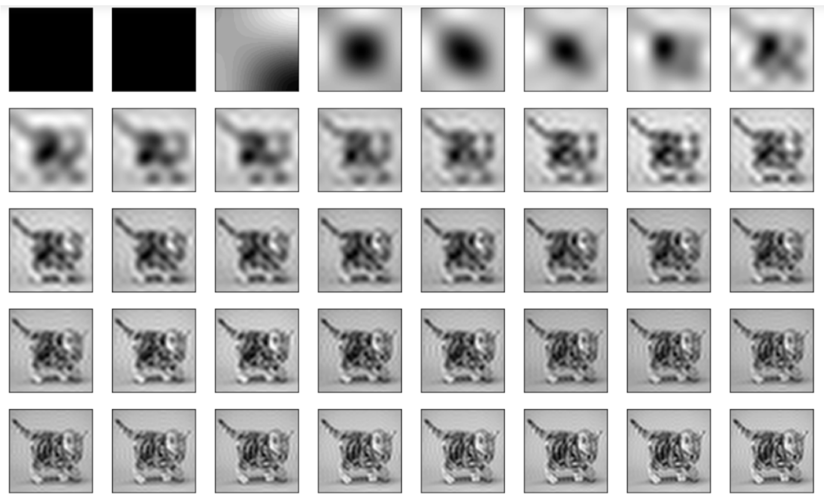
Python biblioteka *PIL* nam omogućava da učitamo sliku i konvertujemo je u *greyscale* sliku. Tim procesom za svaki piksel gubimo informacije o boji, tako da prebacivanjem slike u *Numpy* niz dobijamo običnu 2D matricu.

Sledeći korak u kompresiji je primena 2D diskretne kosinusne transformacije na dobijenu matricu (slika 1). Ako iscrtamo grafik ovako dobijenih koeficijenata, videćemo da samo prvih nekoliko nosi značajnu informaciju - ovo se može videti na slici 4). Dakle, sve dok njih ne odsečemo imaćemo dovoljno informacija da rekonstruišemo sliku dovoljno da možemo da raspoznamo šta je na njoj. Svi ostali koeficijenti predstavljaju fine detalje slike.



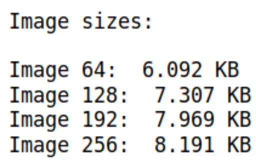
Slika 4: Prvih 2500 koeficijenata

Na slici 5 je prikazano nekoliko rekonstruisanih slika (dakle, na koeficijente je primenjena inverzna 2D diskretna kosinusna transfromacija sa slike 1). Za prvu, svi koeficijenti su postavljeni na 0 - zato je i crna. Zatim smo sačuvali 1×1 kvadrat a sve ostalo postavili na nule, za sledeću 3×3 itd. Već od trećeg reda možemo jasno videti šta je na slici - a prikazali smo samo 40 od 256 slika.



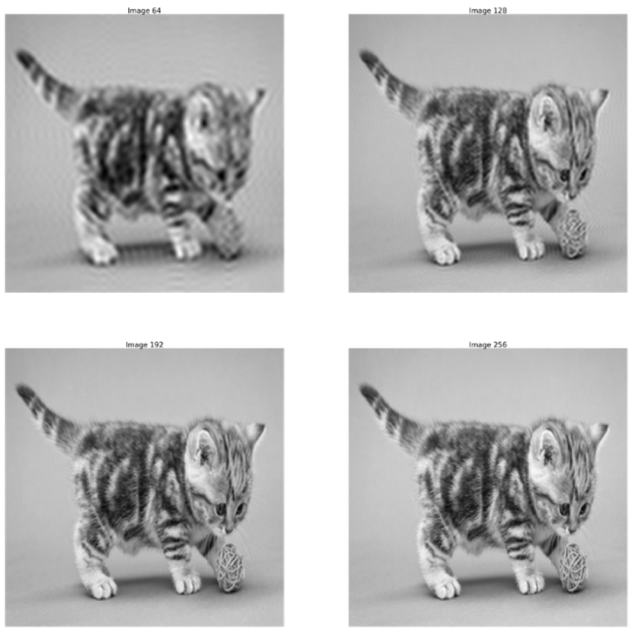
Slika 5: Prvih 40 kompresovanih slika

Na slici 6 su priakazane veličine kompresovanih slika kad je sačuvano redom prvih 64, 128, 192 i 256 koeficijenata a na slici 7 upravo te četiri slike.



```
Image sizes:

Image 64:  6.092 KB
Image 128:  7.307 KB
Image 192:  7.969 KB
Image 256:  8.191 KB
```

Slika 6: Veličine kompresovanih slika



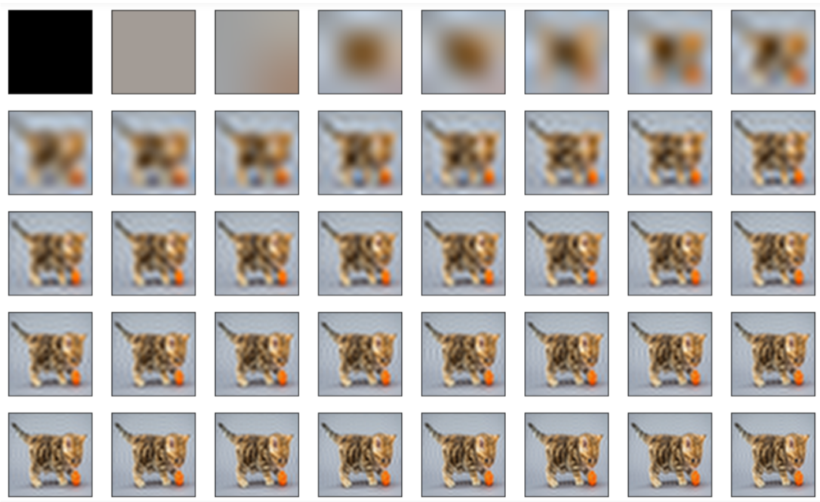
Slika 7: Kompresovane slike

3.3 Kompresija JPEG slika u boji

Kompresiju JPEG slika u boji od kompresije *greyscale* JPEG slika razlikuje samo to *na šta* primenjujemo diskretnu kosinusnu transformaciju. Kao što je već prikazano na slici 3, slika u boji za svaki piksel čuva informaciju o boji, tj. čuva R, B, G komponente.

Diskretnu kosinusnu transformaciju na trodimenzionalnu matricu ćemo primeniti tako što ćemo primetiti dvodimenzionalnu DKT na svaki od kanala, kao što je prikazano na slici 2.

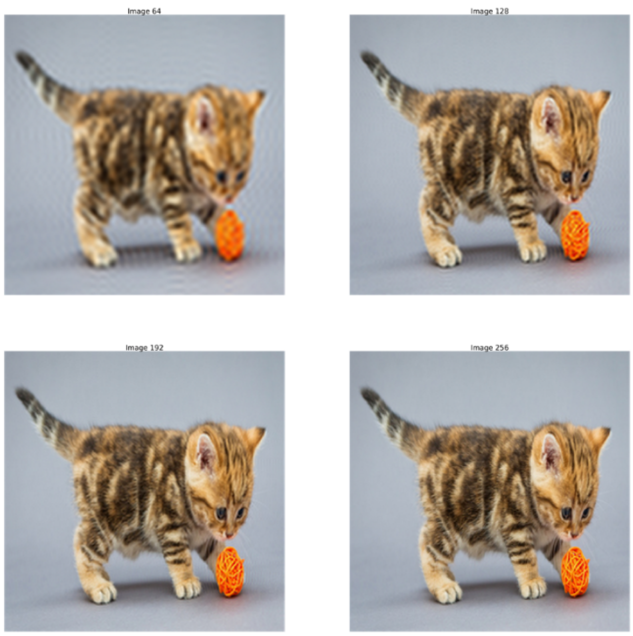
Kao i u odeljku 3.2, prikazaćemo na slici 8 prvih 40 slika (dobijenih na isti način, čuvanjem prvih $k \times k$ piksela), zatim na slici 9 veličine dobijenih slika a na slici 10 i same slike radi poređenja.



Slika 8: Prvih 40 kompresovanih slika

```
Image sizes:

Image 64:  7.564 KB
Image 128:  8.755 KB
Image 192:  9.432 KB
Image 256:  9.648 KB
```

Slika 9: Veličine kompresovanih slika



Slika 10: Kompresovane slike

4 Vremenska složenost

Složenost jednodimenzionalne DKT je $\mathcal{O}(n^2)$.

Naivna implementacija dvodimenzionalne DKT (sa 4 ugnježdene *for* petlje) je složenosti $\mathcal{O}(n^4)$.

Malo pametnija implementacija koja poziva jednodimenzionalni DKT prvo nad redovima pa onda nad kolonama tako dobijene matrice je složenosti $\mathcal{O}(n^3)$ - n puta se poziva jednodimenzionalna DKT po redovima i n puta se poziva nad kolonama.

Ako se pozivi jednodimenzionalne DKT zamene pozivima funkcije FKT (dobijena na način na koji se dobija FFT), može se postići vremenska složenost $\mathcal{O}(n \log n)$.

Literatura

[1] Luke's blog, Simple audio compression, 2011. http://www.lukedodd.com/really-simple-audio-compression/

[2] SciPy dokumentacija https://docs.scipy.org/doc/scipy/reference/index.html

[3] Wikipedia, Diskretna kosinusna transformacija https://en.wikipedia.org/wiki/Discrete_cosine_transform

[4] Unix4Lyfe, Diskretna kosinusna transformacija https://unix4lyfe.org/dct/