A picture containing diagram

Description automatically generated

JPEG algoritem

Timen Bobnar

Ljubljana, 2024

## Povzetek

V tej seminarski nalogi si bomo ogledali delovanje algoritma JPEG. Algoritem JPEG je metoda za stiskanje digitalnih slik, ki temelji na Diskretni kosinusni transformaciji (DCT). Ta omogoča, da se zmanjša velikost datoteke s sliko brez bistvene izgube vizualne kakovosti slike.

Začeli bomo z motivacijo, kjer si bomo ogledali, zakaj bi sploh želeli uporabljati algoritem JPEG in kaj so njegove lastnosti. Sledi razdelek o zajemanju slik in njihovi pretvorbi v format RGB. Nadaljevali bomo z opisom algoritma, s katerim zmanjšamo velikost datoteke brez izgube vizualne kakovosti slike.

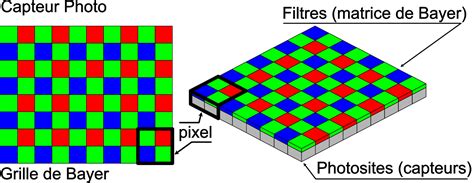
## Motivacija

Slike, ki jih posnamemo s telefonom ali fotoaparatom, so na začetku v formatu RAW. Ta shrani vse informacije, ki jih zajamejo senzorji. Zato je velikost datoteke v tem formatu zelo velika. Tako je v gradivih primerek take slike (SLIKA\_RAW.CR2, v gradivih), kjer je velikost datoteke 21MB. Želimo zmanjšati velikost datoteke tako, da razlike med originalno in predelano sliko niso opazne. Pri tem izrabimo dejstvo, da določenih razlik naše oči enostavno ne morejo zaznati. Tako smo iz prej ustvarjene datoteke izdelali datoteko (Predelana\_slika.jpg , v gradivih), kjer je velikost le 1,6MB. Datoteki sta po velikosti precej različni, vendar se sliki za naše oči ne razlikujeta veliko ali celo nič.

V nalogi bomo predstavili delovanja tega algoritma, saj se dandanes uporablja v veliko elektronskih napravah, ki zajemajo in prikazujejo slike.

## Kako nastane slika

V kameri se nahaja senzor. Senzor je plošča, sestavljena iz dveh plasti. Prva plast je barvni filter, ki prepusti le eno izmed komponente svetlobe, torej le rdečo, modro ali zeleno. Druga plast je sestavljena iz velikega števila detektorjev svetlobe. Te detektorji prepuščeno svetlobo spremenijo v električno energijo.



Slika Senzor v kameri

Kot opazimo na sliki, je en piksel sestavljen iz štirih detektorjev: po en za modro in rdečo barvo ter dva za zeleno. Tako razporeditev barv uporabimo, saj je naše oko veliko bolj občutljivo na zeleno barvo, saj ima naše oko več receptorjev za zeleno barvo kot za rdečo in modro barvo.

Če podatke od vsakega piksla direktno shranimo, temu rečemo format RAW. Več si lahko preberemo na spletnih straneh (Digital 2023) in (Richter brez datuma).

Z nadaljnjo obdelavo slike lahko format RAW pretvorimo v format RGB. Temu postopku pravimo barvna rekonstrukcija (v angleščini demosaicing oziroma debayering). Več o tem si lahko preberete na spletni strani (Wikipedija-Demosaicing 2024).

Na koncu postopka pretvorbe formata je naša slika matrika, sestavljena iz pikslov. Vsak piksel ima tri komponente (rdeča, zelena in modra). Vsaka od komponent ima vrednost med 0 in 255, kjer to število predstavlja intenziteto barve.

## JPEG algoritem

Naši vhodni podatek je matrika pikslov, kjer je posamezen piksel podan v formatu RGB.

### Obdelava barve

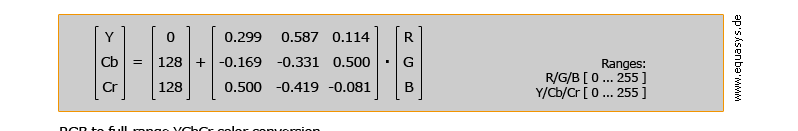
Naše oči so bolj občutljive na spremembo svetlosti kot na spremembo barve. Zato je smiselno naše slike za nadaljnjo obdelavo pretvoriti iz formata RGB v format YCrCb. Format YCrCb sliko razdeli ponovno na tri komponente podobno kot RGB, vendar so v tem primeru komponente svetlost (Y) ter kromatičnost(intenziteta ali nasičenost barve) modre (Cb) ter rdeče barve (Cr). Kromatični komponenti skupaj predstavljata barve tako, kot je videti na sliki 2.

Slika, ki vsebuje besede posnetek zaslona, besedilo, barvitost

Opis je samodejno ustvarjen

Slika Barve v kromatičnem sistemu

Te tri komponente izračunamo za vsak piksel posebej po formuli s slike 3.



Slika formule RGB v YCrCb

Za vsak piksel smo pridobili komponente Y, Cr in Cb. Naše oko je veliko bolj občutljivo na komponento svetlosti in veliko manj na kromatične komponente. To lahko opazimo če pogledamo sliko razcep.png v mapi Porocilo\_gradiva, kjer lahko vidimo, kako izgledajo posamezne komponente slike.

Namesto, da naše komponente obdelujemo kot eno matriko pikslov, kjer je vsak piksel sestavljen iz treh komponent, to matriko razdelimo na tri matrike, kjer vsaka vsebuje vrednosti le ene izmed komponent.

Sledi postopek, kjer zmanjšamo velikost obeh kromatičnih matrik (matriko svetlosti ohranimo v prvotni velikosti). Štiri sosednje vrednosti kromatičnih komponent povprečimo in jih shranimo v novo matriko. Torej sta matriki kromatičnih komponent veliki le četrtino osnovne matrike.

Slika, ki vsebuje besede besedilo, pisava, posnetek zaslona, samolepilni listič

Opis je samodejno ustvarjen

Slika zmanjševanje velikosti matrik kromatičnih komponent

Komponento 119 so dobili tako, da smo sešteli 96+189+71+120 ter delili s 4. Komponento 123 pa tako, da smo sešteli 64+251+12+52 ter delili s 4.

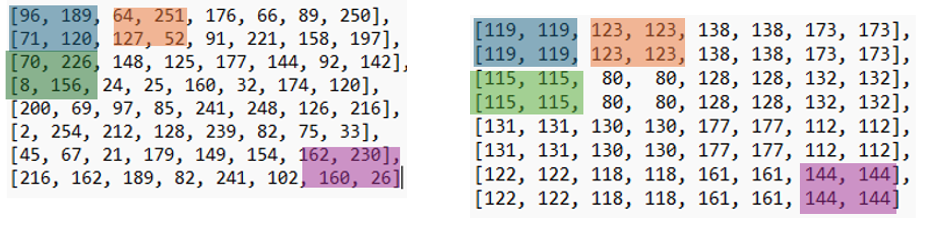
Na tak način smo že nekoliko zmanjšali velikost datoteke. Ko želimo našo sliko spet sestaviti, sta kromatični komponenti premajhnih dimenzij. Da dimenzije uskladimo, vsako komponento manjše matrike zapišemo štirikrat na sosednja polja.

Slika, ki vsebuje besede besedilo, pisava, samolepilni listič, posnetek zaslona

Opis je samodejno ustvarjen

Slika večanje velikosti matrik kromatičnih komponent

Dobljeni matriki sta seveda drugačni kot prvotni, a ustrezni sliki naše oči ne razlikujejo.



Slika Spreminja matrika za kromatično komponento

### Kosanje slike

Pri nadaljnji obdelavi slike se bomo osredotočili le na komponento svetlosti. Tudi ostali dve komponenti obdelamo na skoraj isti način. Na morebitne razlike bomo sproti opozarjali.

Matriko za komponento svetlosti razkosamo na bloke 8x8. To naredimo, ker so postopki, ki sledijo, optimizirani za bloke velikosti 8x8. V večini primerov sta dimenziji slike že deljivi z 8. Če pa temu ni tako, dodamo ustrezno število stolpcev ali vrstic. Pri tem lahko uporabimo več načinov, ki so omenjeni na sliki 7.

Slika, ki vsebuje besede posnetek zaslona, črno in belo, narava, voda

Opis je samodejno ustvarjen

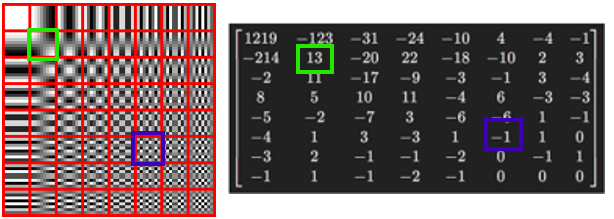
Slika Popravki dimenzij

### Diskretna kosinusna transformacija (DCT)

Kot vhodne podatke za algoritem DCT dobimo matriko dimenzije 8x8. Na vsakem 8x8 bloku imamo 64 komponent svetlosti (Y), ki so predstavljene z vrednostmi med 0 in 255. Postopek DCT posebej izvedemo za vsak 8x8 blok.

Kosinusna transformacija je matematična tehnika, ki se uporablja za predstavitev dvodimenzionalnih podatkov, kot so slike, v obliki linearne kombinacije diskretnih kosinusnih funkcij različnih frekvenc v obeh dimenzijah. Glavni cilj postopka je izraziti kompleksne vzorce podatkov z uporabo koeficientov, ki predstavljajo prispevke posameznih frekvenc v originalnih podatkih.

Kot izhodni podatek dobimo novo 8x8 matriko. Elementom v tej matriki rečemo DCT koeficienti. Te koeficiente lahko razumemo kot kako pogosto in s kakšno intenziteto se določeni vzorci (frekvenčne komponente) pojavijo v vhodnih matriki. Na razpolago imamo 64 različnih frekvenčnih komponent, ki skupaj sestavljajo frekvenčno bazo. Vse komponente baze lahko vidimo na spodnji sliki.



Slika frekvenčna baza ter primer matrike po izvedbi DCT algoritma.

Na sliki 8 lahko sedaj opazimo, da se vzorec svetlosti (frekvenčna komponenta) na mestu (2,2) v sliki pojavi s frekvenco oziroma pogostostjo 13. Frekvenčna komponenta obkrožena z modro pa se pojavi -1-krat.

Za podrobnejše razumevanje diskretne kosinusne transformacije si lahko ogledate video (Computerphile 2015) in (College 2021). V teh videih je lepo razloženo, kako sama transformacija deluje.

Če pa nas ozadje postopka ne zanima, je pomembna le končna formula:

Slika, ki vsebuje besede besedilo, pisava, bela, vrstica

Opis je samodejno ustvarjen

Slika Formula za izračun DCT

V formuli N predstavlja dimenzijo našega bloka (v našem primeru je N enak 8). Pixel (x, y) predstavlja vrednost v naši 8x8 matriki na poziciji x,y. DCT (i, j) predstavlja i, j-to komponento v naši izhodni matriki.

Pomembno je poudariti, da poznamo tudi inverzno transformacijo , ki jo bomo potrebovali kasneje.

### Kvantizacija

Kvantizacija je glavni postopek, pri katerem zmanjšujemo velikost datoteke, a s tem sočasno tudi kakovost slike. Kakovost slike ocenjujemo tako, da primerjamo, koliko podrobnosti smo izgubili in koliko je slika postala bolj 'zrnata' ali zamegljena. Višje kakovost pomenijo manj izgub in manjšo 'zrnatost' oziroma zamegljenost slike.

#### Kvantizacijska matrika

Za kvantizacijo uporabimo kvantizacijsko matriko, ki ima enake dimenzije kot matrika po DCT postopku. Kvantizacijska matrika določa, s kakšno natančnostjo bomo shranili posamezne frekvenčne komponente. Pri postopku kavtizacije delimo istoležne komponente v matriki po postopku DCT z elementi iz kvantizacijske matrike. Nove vrednosti zaokrožimo. Če je nova zaokrožena vrednost enaka nič, ob sestavljanju slike nazaj, tistega frekvenčnega vzorca ne bomo imeli in bo slika manj ostra (imela bo slabšo kvaliteto). Torej večja kot je vrednosti v kvantizacijski matriki, večja je verjetnost, da se bomo znebili frekvenčne komponente na tistem mestu.

Slika, ki vsebuje besede besedilo, pisava, posnetek zaslona, bela

Opis je samodejno ustvarjen

Slika Primer kvantizacijske matrike

#### Postopek kvantizacije

Pri postopku kvantizacije delimo elemente matrike, ki smo jo dobili po DCT postopku, z istoležnimi elementi v kvantizacijski matriki in rezultate zaokrožimo na najbližje celo število. S tem postopkom dobimo veliko ničel v naši matriki, kar pomaga pri zmanjševanju količine informacij, ki jih je treba shraniti. S tem posledično vpliva na velikost datoteke.

Slika, ki vsebuje besede vhodna naprava, periferen, kvadrat, posnetek zaslona

Opis je samodejno ustvarjen

Slika Primer uporabe kvantizacijske matrike na matriki po postopku DCT

Za boljše razumevanje si lahko ogledate posnetek (ENGEGY 2016)[2:15,5:00]

#### Kako naredimo kvantizacijsko matriko?

Kvantizacijska matrika je tesno povezana z matriko, ki jo dobimo na koncu DCT algoritma. Na sliki 8 se lahko vizualni prepričamo, kako naše oko zazna različne frekvence diskretne kosinusne transformacije. Opazimo lahko, da so vzorci dvodimenzionalne kosinusne transformacije iz slike 8 v zgornji desni polovice na pogled veliko bolj različni med sabo kot vzorci levo spodaj. Zaradi tega pojava uvedemo našo kvantizacijsko matriko, ki ima večje vrednosti desno spodaj ter manjše vrednosti levo zgoraj. S tem bomo povečali verjetnost, da odstranimo komponente levo spodaj, saj niso toliko pomembne kot tiste levo zgoraj.

Obstaja veliko različnih kvantizacijskih matrik (lahko generiramo tudi svoje). Kot smo že omenili, je naše oko veliko bolj občutljivo na svetlobo kot na kromatičnost, to znanje bomo uporabili tudi sedaj in to tako, da bomo za kromatičnosti in svetlost generirali različne kvantizacijske matrike.Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, številka

Opis je samodejno ustvarjen

Slika Kvantizacijske matrike

### Pretvorbo v linearno zaporedje

Sedaj smo na točki, ko moramo naše informacije le še shraniti. Problem nastane, saj v računalnik ne moremo neposredno shraniti matrik, saj je spomin v računalniku linearen. Zato moramo naše matrike, pridobljene po kvantizaciji, na pameten način obdelati in jih zapisati v obliki zaporedja.

Oglejmo si našo matriko po kvantizaciji.

Slika, ki vsebuje besede posnetek zaslona, besedilo

Opis je samodejno ustvarjen

Slika Matrika po kvantizaciji

Opazimo, da je v zgornjem levem kotu zelo velika števka. Če bi pogledali ostale matrike po kvantizaciji, bi opazili dve stvari:

1. V zgornjem levem kotu je zmeraj zelo velika števka
2. Matrike 8x8 blokov, ki so si sosednje, imajo vrednost v zgornjem levem kotu podobno(absolutna razlika je majhna)

Zaradi teh dveh dejstev bomo to zgornjo levo komponento, ki jo včasih imenujemo tudi DC (ostale komponente imenujemo AC) obdelali na poseben način.

#### Obdelava DC-ja

Po postopku kvantizacije imamo sedaj zelo veliko število matrik. Vse te matrike imajo po eno DC komponento. Vse te komponente sedaj postavimo v eno matriko.

Za shranjevanje manjših števil lahko uporabimo manj bitov (za 1200 potrebujemo 11 bitov (1200 = 100101100002), za 105 pa le 7 (105 = 11010012). Zato bomo naša števila v matriki DC zmanjšali na naslednji način.

Kjer DC0 predstavlja prvi stolpec matrike DC in d0,ki predstavlja prvi stolpec spremenjene matrike.

Slika, ki vsebuje besede besedilo, rokopis, vrstica, grafični prikaz

Opis je samodejno ustvarjen

Slika Primer predelave DC matrike

#### Cikcak sprehod

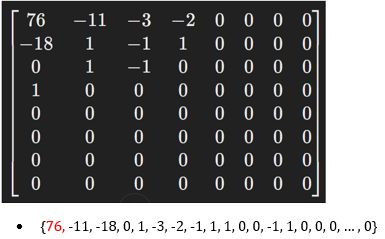
Ko želimo zapisati podatke iz matrike zaporedoma, si pomagamo s cikcak sprehodom. Najlažje ga bomo ponazorili s sliko

A diagram of a square with arrows and lines

Description automatically generated

Slika cikcak sprehod

Če izvedemo cikcak sprehod na matriki, pridobljeni s kvantizacijo



Slika Matrika po kvantizaciji

bomo dobili sledeče zaporedje *A black and white screen with numbers

Description automatically generated*

Opazimo, da po štirinajstem elementu nastopijo same ničle. Cikcak sprehod uporabimo ravno zato, da dosežemo, da bo na koncu zaporedja večinoma 0.

Element obarvan z rdečo barvo je DC komponenta. To DC komponento zamenjamo z elementom iz matrike **d** na **sliki 14.**

### Kodiranje po dolžini(RLE)

Po cikcak sprehodu imamo 8x8 matriko predstavljeno v linearni obliki. Da skrajšamo dolžino zapisa, uvedemo nov način zapisovanja. Rečemo mu kodiranje po dolžini.

V tem novem načinu so elementi dvojice, kjer prvi člen predstavlja ponovitev elementa, ki je drugi(drugi člen)

Oglejmo si to na dveh primerih:

* {76, - 11, - 18, 0,1, -3, -2, -1, 1, 1, 0, 0, -1, 1, 0,...., 0}
  + (1, -11), (1, - 18), (1, 0), (1, -3), (1, -2), (1, -1), (2, 1), (2, 0), (1, -1), (1, 1), (50, 0)
* {100, 5, -3, 5, 0, 0, 1, 8, -3, 2, -1, -1, -1, 0, 0, 0, 1, 0,..., 0}
  + (1, 5), (1, -3), (1, 5), (0, 2), (1, 1), (1, 8), (1,-3), (1, 2), (3, -1), (3, 0), (1, 1), (47,0)

Pri tem opozorimo, da različne oblike JPEG algoritma opravijo kodiranje po dolžini na različne načine. En način smo že omenili. Pri njem shranjujemo pare (ponovitve elementa X, element X). Ostale možnosti so še na primer (število bitov za zapis elementa, (število ponovitev X, element X)), možno je tudi, da števila le zapisujemo v vrsto.

### Huffmanovo kodiranje

Ko izvedemo kodiranje po dolžini, smo dobili obliko, ki jo lahko shranimo v računalnik. Te informacije bomo shranili na malo drugačen način, saj bomo tako zmanjšali število bitov, ki si jih je potrebno zapomniti. Uporabljamo Huffmanovo kodiranje, saj bomo na tak način porabili minimalno število bitov, da si zapomnimo vse potrebne informacije.

Delovanje Huffmanovega kodiranja si oglejte na posnetku (Technology 2019)

V primeru, da vas zanima ,več si oglejte: (Reducible 2021)

V obeh primerih generiramo Huffmanovo drevo za primere črk. Sedaj bomo to prilagodili tako, da bomo namesto črk uporabili dvojice kot so (1,5), (50,0),… Te dvojice predstavljajo komponente v našem kodiranju po dolžini. Na tak način precej zmanjšamo količino bitov, ki jih potrebujemo za shranjevanje naših informacij. Podobno idejo uporabimo tudi, ko kodiramo DC komponente naše slike.

Določeni JPEG algoritmi uporabljajo že standardizirana Huffmanova drevesa ter tako prihranimo dodatni prostor, saj nam ni potrebno shraniti še samih dreves, ki jih potrebujemo za dekodiranje.

### Zapis in shranjevanje podatkov

Iz oblike RAW smo torej dobili podatke, izražene s Huffmanovim kodiranjem A večkrat smo omenili, da imamo na določenem koraku več možnih načinov izvedbe postopkov. Seveda vsak tak postopek prinese nekoliko drugačen zapis. Zato informacijo o uporabljenih postopkih tudi na določen način zapišemo na sam začetek datoteke. Ustrezne načine shranjevanja si lahko ogledate na spletnem viru (Charette 2017), kjer najdete celotno shemo shranjevanja JPEG datotek.

# Viri:

Adams, Richars. 2020. *Silentpeakphoto.* 19. Juni. Poskus dostopa 24. November 2023. https://www.google.com/imgres?imgurl=http%3A%2F%2Fwww.silentpeakphoto.com%2Fwp-content%2Fuploads%2F2020%2F06%2FHow-Image-Sensors-Work.jpg&tbnid=KVe1zUZJj5ZURM&vet=10CBkQMyhzahcKEwiA0KGtiKGFAxUAAAAAHQAAAAAQAw..i&imgrefurl=https%3A%2F%2Fsilentpeakphoto.com%.

Charette, Stéphane. 2017. *Anatomy of a JPEG.* 1. februar. Poskus dostopa 24. februar 2024. https://www.ccoderun.ca/programming/2017-01-31\_jpeg/.

brez datuma. *cmlab-Encoding Process.* Poskus dostopa 2024. Februar 2024. https://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/jpeg/jpeg/encoder.htm.

College, Friendly. 2021. *YouTube-Discrete Cosine Transform and Haar transform with Examples.* 2. april. Poskus dostopa 24. Februar 2024. https://www.youtube.com/watch?v=lQnUex31f5Q.

Computerphile. 2015. *YouTube-JPEG DCT, Discrete Cosine Transform (JPEG Pt2)- Computerphile.* 22. maj. Poskus dostopa 24. Februar 2024. https://www.youtube.com/watch?v=Q2aEzeMDHMA.

Digital, ProGrade. 2023. *ProGrade Digital.* 3. julij. Poskus dostopa 26. junij 2024. https://progradedigital.com/understanding-camera-sensors-a-comprehensive-guide/.

ENGEGY. 2016. *YouTube-54- The JPEG compression algorithm.* 22. maj. Poskus dostopa 24. Februar 2024. https://www.youtube.com/watch?v=aFbGqXFT0Nw.

2022. *geeksforgeeks-JPEG full form.* 30. maj. Poskus dostopa 24. februar 2024. https://www.geeksforgeeks.org/jpeg-full-form/.

Griffin, Jonathan. 2023. *the webmaster.* 23. januar. Poskus dostopa 24. november 2023. https://dsp.stackexchange.com/questions/35339/jpeg-dct-padding.

Reducible. 2021. *YouTube-Huffman Codes: An Information Theory Perspective.* 30. julij. Poskus dostopa 24. februar 2024. https://www.youtube.com/watch?v=B3y0RsVCyrw.

Richter, Mr. brez datuma. *Student suport Mr. Richter's courses and materials.* Poskus dostopa 26. junij 2024. https://www.cours.jlrichter.fr/lycee/snt-sciences-numeriques-et-technologie/2snt-g-la-photographie-numerique/.

Technology, Pizzey. 2019. *YouTube-Huffman coding step-by-step example.* 13. januar. Poskus dostopa 24. februar 2024. https://www.youtube.com/watch?v=iEm1NRyEe5c.

Trauth, Martin H. 2018. *Matlab and python recipes for earth sciences.* 10. Julij. Poskus dostopa 24. Februar 2024. http://mres.uni-potsdam.de/index.php/2018/07/10/multispectral-cameras-bayer-mosaics-and-image-processing-with-matlab/.

brez datuma. *Wikipedija.* Poskus dostopa 24. november 2023. https://en.wikipedia.org/wiki/YCbCr.

brez datuma. *Wikipedija.* Poskus dostopa 24. februar 2024. https://en.wikipedia.org/wiki/Bayer\_filter.

Wikipedija-Demosaicing. 2024. *Wikipedija-Demosaicing.* 18. Junij. Poskus dostopa 26. junij 2024. https://en.wikipedia.org/wiki/Demosaicing.

brez datuma. *Wikipedija-JPEG.* Poskus dostopa 24. Februar 2024. https://en.wikipedia.org/wiki/JPEG.