

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – 1. stopnja

Klara Golob
Stiskanje slik

Delo diplomskega seminarja

Mentor: izr. prof. dr. Emil Žagar

Ljubljana, 2020

KAZALO

1. Uvod	4
2. Digitalna slika	4
2.1. Binarne, črno-bele in barvne slike	4
2.2. Obdelava slik in operacije z matrikami	5
3. Stiskanje slik	8
4. JPEG: standard za stiskanje slik	9
5. Osnovni JPEG algoritem	9
6. Pretvorba barvnega sistema	9
7. Razbitje slike na bloke velikosti 8×8	10
8. Transformacija	10
8.1. Diskretna kosinusna transformacija	11
8.2. Druge transformacije	12
9. Kvantizacija	13
10. Kodiranje	14
10.1. Huffmanovo kodiranje	15
11. Razširitev	18
11.1. Rezultati	19
12. Algoritem	21
13. Drugi formati za stiskanje slik	21
14. Konec dela	22
Slovar strokovnih izrazov	22
Literatura	23

Stiskanje slik

POVZETEK

Stiskanje slik je danes nepogrešljiv del vsakodnevnega življenja ljudi. Ločimo dva postopka, stiskanje z izgubami in stiskanje brez izgub. Eden izmed najbolj uporabnih algoritmov za stiskanje je JPEG (Joint Photographic Expert Group) algoritmom, ki temelji na stiskanju z izgubami. Za učinkovito stiskanje uporabi diskretno kosinusno transformacijo, ki pretvori sliko v frekvenčno domeno, kjer lažje odstrani nepomembne dele. Kljub temu, da se ponavadi nekaj podatkov pri poteku JPEG algoritma izgubi, lahko z inverznim algoritmom rekonstruiramo približek originalne slike. Rekonstruirana slika ni čisto enaka prvotni, ji je pa toliko podobna, da ponavadi s človeškim očesom ne zaznamo razlike.

Image compression

ABSTRACT

Image compression is nowadays an indispensable part of our daily lives. It falls into two groups, lossy compression and lossless compression. One of the most useful image compression algorithms is JPEG algorithm, which belongs to the lossy compression group. For efficient compression, JPEG algorithm uses the discrete cosine transform, which transforms the image into a frequency domain, that we can more easily remove insignificant parts of the image. Despite the fact that the data is lost during the JPEG algorithm, we can retrieve this data in the reverse order and reconstruct the original image. The reconstructed image is not exactly the same as the original, but it is so similar that we usually do not notice the difference with our human eye.

Math. Subj. Class. (2010): (94A8, 65D18, 62M40)

Ključne besede: stiskanje slik, JPEG standard, osnovni JPEG algoritmom, diskretna kosinusna transformacija, kvantizacija, cik-cak skeniranje, Huffmanovo kodiranje, razširjanje slik

Keywords: image compression, JPEG standard, main JPEG algorithm, discrete cosine transform, quantization, cik-cak scanning, Huffman coding, image decompression

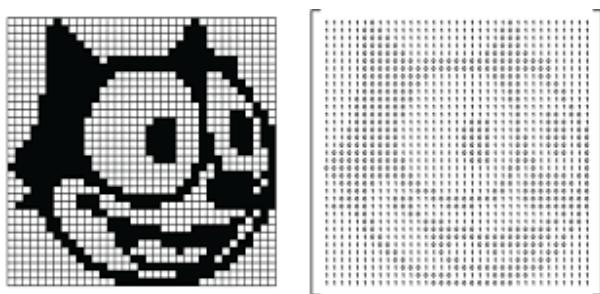
1. UVOD

Dandanes je tehnologija nepogrešljiv del življenja. Dnevno na miljone slik (velikokrat bomo enačili izraza slika in fotografija) in video posnetkov kroži po internetu, zato moramo znati učinkovito shranjevati, prenašati, analizirati in procesirati digitalne informacije, pri čemur si pomagamo s stiskanjem podatkov. V zadnjih desetletjih se je digitana tehnologija zelo razvila, še posebej na področju fotografije, videoposnetkov in analize podatkov. Stiskanje slik je precej specifični, saj imajo določene lastnosti, ki jih lahko izkoristimo in s tem pridemo do boljših rezultatov, kot če bi uporabili splošne programe za stiskanje podatkov. Stiskanje slik ima zelo pomembno vlogo v učinkovitem shranjevanju in prenašanju fotografij in je neobhodno potrebno. Cilj stiskanja je, da iz slike odstrani odvečne informacije, ki za človeško oko niso tako pomembne in shrani le tiste najbolj potrebne. Iz stisnjениh podatkov potem lahko rekonstruira približek originalne slike, v skladu s človeško vizualno percepциjo.

2. DIGITALNA SLIKA

Digitalna slika je elektronska predstravitev slike, ki je shranjena v računalniku. Ne moremo je videti in nima fizične velikosti, dokler je ne prikažemo na zaslonu ali jo natisnemo. Do takrat je le zbirka števil na trdem disku računalnika, ki opisuje posamezne elemente slike, piksele (tudi slikovne pike ali slikovne točke). Formalno je piksel najmanjša enota slike, ki vsebuje informacijo o intenziteti barve določenega koščka slike. Na zaslonu ima določeno velikost in predstavlja mersko enoto za izražanje ločljivosti. En piksel meri $\frac{1}{96}$ inča. Ločljivost slike pove, kako podrobna je slika. Višja ločljivost slike pomeni več pikslov na določeni površini, nižja ločljivost manj pikslov na določeni površini. Slike z višjo ločljivostjo so bolj podrobne in kvalitetne. Zbirka pikslov je urejena v vrstice in stolpce v obliki matrike, katere dimenzija pove, kako velika je slika in kakšno ločljivost ima. Če ima matrika slike velikost $m \times n$, je širina slike enaka n pikslov, višina m pikslov in ločljivost produkt $m \times n$.

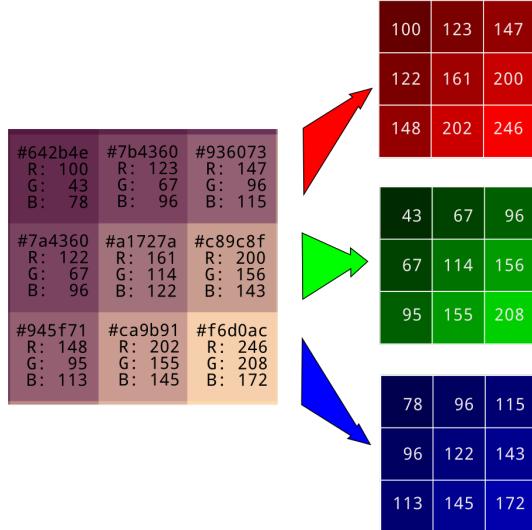
2.1. Binarne, črno-bele in barvne slike. Barvna ali bitna globina nam pove, koliko različnih barv je lahko na sliki. V kolikor za zapis piksla porabimo več bitov, lahko z njim prikažemo več različnih barv. Če piksel zapišemo z enim bitom, lahko zavzame $2^1 = 2$ vrednosti, se pravi 0 ali 1. Z omenjenim zapisom predstavimo binarne slike, ki so sestavljeni le iz črne in bele barve. Na sliki 1 je figura mačke,



SLIKA 1. Binarna slika mačke kot matrika velikosti 35×35 .

predstavljena je z matriko velikosti 35×35 , katera vsebuje bodisi število 0, ki določa črno barvo, bodisi število 1, ki določa belo barvo. Najpogosteje za zapis piksla potrebujemo 8 bitov, s čimer lahko predstavimo $2^8 = 256$ različnih odtenkov barv.

Tako večino črno-belih slik predstavljajo matrike s celimi števili med 0 in 255, ki lahko dosežejo 256 različnih odtenkov sive. Barvne slike so ponavadi predstavljene s tremi matrikami, ki določajo količino rdeče, zelene in modre barve. Takšen barvni sistem je znan kot RGB (Red, Green, Blue) in je prikazan na sliki 2. Posamezen piksel je tako zapisan s 24 biti, na sliki pa lahko prikažemo $2^{24} = 16777216$ različnih odtenkov barv.



SLIKA 2. Predstavitev barvnih fotografij s sistemom RGB.

2.2. Obdelava slik in operacije z matrikami. Ko digitalno sliko predstavimo z matriko, se lahko vprašamo, kako bi operacije na elementih matrike vplivale nanjo. Vzemimo na primer binarno fotografijo s slike 1. Opišimo jo z matriko $A = (a_{i,j})_{i,j=1}^{35}$ in na njej izvedimo naslednji dve operaciji:

$$(1) \quad B = (b_{i,j})_{i,j=1}^{35} = A^T,$$

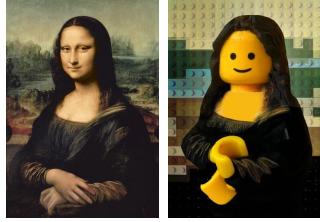
$$(2) \quad C = (c_{i,j})_{i,j=1}^{35} = (a_{j,35-i+1})_{i,j=1}^{35}.$$

S tem, ko na matriki izvedemo različne operacije, se spremeni tudi slika, ki jo ta predstavlja. Transformaciji (1) in (2) sta prikazani na sliki 3.



SLIKA 3. Na levi je originalna slika, v sredini slika predstavljena z matriko B , ki je definirana z (1) in na desni slika, predstavljena z matriko C , ki je definirana z (2).

Prav tako lahko na sliki uporabimo transformacije, ki spremenijo njeno vsebino. Poglejmo fotografiji na sliki 4. Levo, s podobo Mona Lise, opišimo z matriko D in

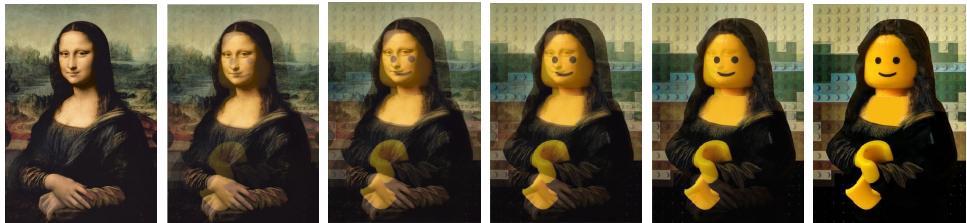


SLIKA 4. Na levi je slika Mona Lise, ki jo opišemo z matriko D na desni slika Mona Lise iz Lego kock, ki jo opišemo z matriko E .

desno, s podobo Mona Lise iz Lego kock, opišimo z matriko E . Za vsak skalar t z intervala $[0, 1]$ definirajmo matriko

$$(3) \quad M(t) = (1 - t)D + tE.$$

Opazimo, da je $M(0) = D$ in $M(1) = E$. Za $t \in (0,1)$ matrika $M(t)$ predstavlja sliko, ki ima nekaj lastnosti prve slike Mona Lise in nekaj lastnosti druge slike Mona Lise iz Lego kock.



SLIKA 5. Slike, predstavljene z matriko $M(t)$, ki je definirana s (3), za $t = 0; 0,2; 0,4; 0,6; 0,8$ in 1.

Ker ponavadi slike vsebujejo veliko število pikslov, je ena izmed pomembnih operacij, ki jih izvajamo na matrikah slik, stiskanje. Cilj tega je zmanjšati število potrebnih informacij za shranjevanje in prenašanje. Za učinkovito prenanašanje slik v elektronski obliki (preko satelita, interneta, faksa, ipd.), lahko slike stisnemo s pomočjo singularnega razcepja, da se izognemo prenašanju vseh podatkov.

Izrek 2.1 (Singularni razcep). *Za vsako matriko $A \in \mathbb{R}^{m \times n}$, $m \geq n$, obstaja singularni razcep (SVD)*

$$A = U\Sigma V^T,$$

kjer sta $U \in \mathbb{R}^{m \times m}$ in $V \in \mathbb{R}^{n \times n}$ ortogonalni matriki in je $\Sigma \in \mathbb{R}^{m \times n}$ oblike

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix},$$

kjer so $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ singularne vrednosti matrike A . Stolpci $U = [\mathbf{u}_1, \dots, \mathbf{u}_m]$ so levi, $V = [\mathbf{v}_1, \dots, \mathbf{v}_n]$ pa desni singularni vektorji.

Slike so sestavljene iz bolj in manj razgibanih delov. Obraz osebe na neki sliki vsebuje veliko pomembnih podrobnosti, medtem ko je na primer nebo v ozadju bolj enolično in zato manj kompleksno. Izkaže se, da manjše vrednosti v singularnem

razcepu matrike predstavljajo tisti „bolj dolgočasen“ del slike in jih zato lažje zanemarimo, brez da bi s tem pokvarili njeno podobo. Poglejmo si to podrobneje. Imamo sliko predstavljenou z matriko A in njen singularni razcep $A = U\Sigma V^T$, kar lahko zapišem kot

$$A = \sum_{i=1}^n \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$

Naj bo $k \leq n$. Definiramo

$$(4) \quad A_k = \sum_{i=1}^k \sigma_i \mathbf{u}_i \mathbf{v}_i^T,$$

ki predstavlja aproksimacijo matrike A , ki ohrani le prvih k singularnih vrednosti in vektorjev.

Izrek 2.2 (Eckart-Young-Mirsky). *Naj bo $A = U\Sigma V^T$ singularni razcep matrike $A \in \mathbb{R}^{m \times n}$, $m \geq n$, in $\text{rang}(A) > k$. Naj bo*

$$A_k = \sum_{i=1}^k \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

Potem velja

$$\min_{\text{rang}(B)=k} \|B - A\|_2 = \|A_k - A\|_2 = \sigma_{k+1}$$

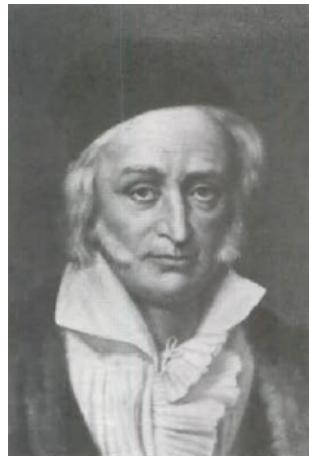
in

$$\min_{\text{rang}(B)=k} \|B - A\|_F = \|A_k - A\|_F = \left(\sum_{i=k+1}^n \sigma_i^2 \right)^{\frac{1}{2}}.$$

Iz izreka 2.2 sledi, da je A_k najboljša aproksimacija matrike A z matriko ranga k , σ_{k+1} pa pove, kakšno je odstopanje A od prostora matrik ranga k . Manjše kot so singularne vrednosti, ki jih odstranimo, manjše je odstopanje matrike A_k od A . Zato lahko ravno te majhne singularne vrednosti odstranimo iz matrike neke slik in s tem običajno ne poškodujemo njene kvalitete.

Na črno-beli sliki 6 je podoba matematika Carla Fridericha Gaussa, predstavljena z matriko A velikosti 340×280 . Če aproksimiramo matriko A z A_k , kot je opisano v (4), dobimo sliko, ki ustrezava prvim k singularnim vrednostim matrike A . Zadošča že, da ohranimo le prvih 32 singularnih vrednosti. Tako lahko namesto vseh $340 \times 280 = 95200$ števil prenesemo le 32 singularnih vrednosti $\sigma_1, \sigma_2, \dots, \sigma_{32}$, 32 levih singularnih vektorjev $\mathbf{u}_1, \dots, \mathbf{u}_{32}$ v \mathbb{R}^{340} in 32 desnih singularnih vektorjev $\mathbf{v}_1, \dots, \mathbf{v}_{32}$ v \mathbb{R}^{280} , kar je le $32 + 32 \times 340 + 32 \times 280 = 19872$ števil. Prejemnik fotografije nato izračuna A_{32} po (4), kar predstavlja aproksimacijo originalne slike. Na sliki 7 je prikazano nekaj primerov fotografij predstavljenih z matriko A_k za vrednosti k med 2 in 256. Na začetku je slika zelo zabrisana, a kmalu dobi lepo, čisto obliko. Slika z matriko A_{32} je že precej razločna in zelo podobna originalu. Za lažjo predstavo zapišimo nekaj singularnih vrednosti matrike A : $s_1 = 49096, s_{16} = 22589, s_{64} = 484, s_{128} = 182, s_{256} = 5$ in $s_{280} = 0,5$. Kljub temu da takšen pristop za stiskanje slik dobro deluje, se v praksi uporabljam še boljše metode, ki časovno niso tako zahtevne.

Obdelava slik ima mnogo aplikacij na različnih področjih, kot so medicina, računalništvo, robotika, filmska industrija, itd. Ker digitalne fotografije uporabljam



SLIKA 6. Matematik Carl Friderich Gauss



SLIKA 7. Aproksimacije fotografije s slike 6 z matrikami A_k , dobljenimi s singularnim razcepom, za vrednosti $k = 2, 8, 16, 32, 64, 256$.

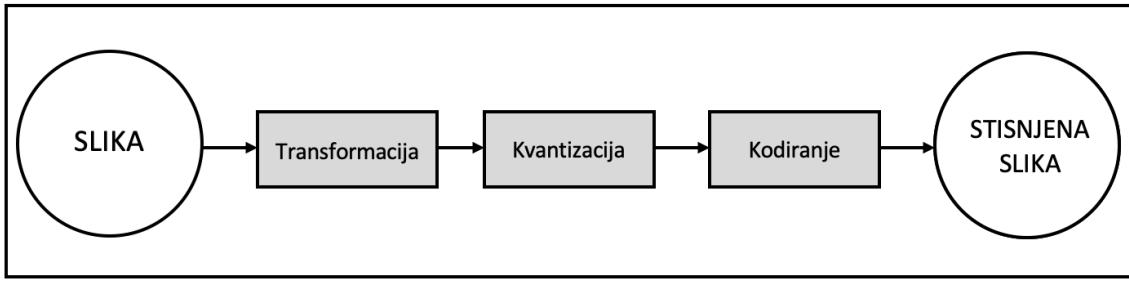
skoraj na vsakem koraku in je vsaka slika predstavljena z veliko podatki, je zelo pomembno, da znamo slike pri shranjevanju in prenašanju stisniti. Tako zavzamejo veliko manj prostora, shranimo lahko več slik in jih prenesemo hitreje, kar pa ima posledično več dobrih lastnosti. Za velika podatkovna skladišča, na primer tista, ki jih imata Google in Facebook, lahko prepeljitev količine podatkov predstavlja veliko zmanjšanje prostora in zahtevanih računalnikov ter posledično velik prihranek energije pri porabi in hlajenju ter ogromno zmanjšanje negativnih vplivov na okolje.

3. STISKANJE SLIK

Kot smo omenili, poznamo dva postopka stiskanja:

- (1) stiskanje brez izgub,
- (2) stiskanje z izgubami.

Že sami imeni povesta, da se pri stiskanju brez izgub med procesom stiskanja vsi podatki ohranijo, medtem ko se pri stiskanju z izgubami v procesu nekaj podatkov izgubi. Tipični sistem za stiskanje slik je ilustriran na sliki 8. Originalna slika je ponavadi transformirana v drugo domeno, v kateri se lahko bolj osredotočimo na pomembnejše informacije, tiste manj pomembne pa v procesu kvantizacije združimo ali odstranimo. Procesor ali kodirnik nato podatke zakodira in jih shrani v stisnjeno datoteko. Obnovljena slika, ki jo rekonstruiramo v obratnem vrstnem redu stiskanja, je med postopkom mogoče izgubila nekaj informacij in zato lahko vsebuje kakšno napako ali popačenje. Obstaja več algoritmov za stiskanje slik, ki temeljijo na različnih metodah. Eden izmed najbolj poznanih in uporabljenih je JPEG algoritmom.



SLIKA 8. Tipični sistem za stiskanje slik.

4. JPEG: STANDARD ZA STISKANJE SLIK

Standard JPEG je prvi nacionalni standard za stiskanje mirujočih barvnih in črno-belih slik do velikosti 65535×65535 pikslov. Ta standard je rezultat skupnega prizadevanja Mednarodne komunikacijske zveze (ITU), Mednarodne organizacije za standardizacijo (ISO) in Mednarodne komisije za elektrotehniko (IEC). Razvili so ga že v letu 1992, danes pa je eden izmed najbolj uporabljenih standardov za stiskanje slik. Slike, stisnjene z uporabo JPEG algoritma, se ponavadi shranijo v datoteke s končnico .jpg ali .jpeg, med katerima ni razlike. Krajši izraz obstaja, zato ker je starejša različica operacijskih sistemov Windows, natančneje MS-DOS 8.3 in FAT-16, podpirala imena datotek s končnicami, ki vsebujejo le 3 črke. Medtem ko UNIX-ovi operacijski sistemi, kot sta macOS ali Linux, niso imeli omejitve. Danes operacijski sistemi Windows podpirajo tudi štiričrkovne kratice, zato je možna uporaba obeh. Obstajajo širje različni modeli JPEG:

- (1) Zaporeden način, pri katerem je vsaka barvna komponenta zakodirana z enim skeniranjem zaporedno od vrha do dna.
- (2) Brezizgubni način, ki ohranja točno originalno sliko, omogoča manjše stiskanje in je manj uporabljen.
- (3) Progresiven način, pri katerem je slika zakodirana z več skeniranjami.
- (4) Hierarhični način, ki uporablja zelo progresivno metodo, pri katerem je slika razdeljena na več manjših delov, ki so zakodirani z enim ali več skeniranjimi.

V praksi se največ uporablja osnovni JPEG algoritem, ki temelji na stiskanju z izgubami in hierarhičnem načinu delovanja.

5. OSNOVNI JPEG ALGORITEM

Osnovni JPEG algoritem je sestavljen iz naslednjih korakov:

- (1) Pretvorba barvnega sistema ($\text{RGB} \rightarrow \text{YCbCr}$).
- (2) Razdelitev na bloke velikosti 8×8 .
- (3) Diskretna kosinusna transformacija (DCT).
- (4) Proses kvantizacije.
- (5) Cik-cak skeniranje in Huffmanovo kodiranje.
- (6) Razširitev z uporabo inverzne diskretne kosinusne transformacije.

6. PRETVORBA BARVNEGA SISTEMA

Da bi dosegli dobre rezultate stiskanja, najprej zmanjšamo korelacijo med barvnimi komponentami s pretvorbo barvnega sistema RGB v barvno-svetlostni sistem.

Ta pretvorba pri stiskanju prinese prednost, saj sta barvost in svetlost močno nepovezani med seboj. Poleg tega barvni kanali vsebujejo veliko odvečnih informacij, ki jih lahko enostavno odstranimo, brez da bi pri tem žrtvovali kvaliteto rekonstruirane slike. V človeškem očesu sta dve vrsti receptorjev, paličnice in čepnice. Paličnice zaznavajo svetlobo in so bolj občutljive, med tem ko čepnice zaznavajo barve in so manj občutljive, kar pomeni da je človeško oko bolj občutljivo na svetlost kakor na barvost. Osnovni JPEG algoritm uporablja pretvorbo v barvni sistem YCbCr, kjer kanal Y predstavlja svetlost, kanala Cb in Cr pa barvost. Pretvorbo barvnega sistema dosežemo s transformacijo

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0,299000 & 0,587000 & 0,114000 \\ -0,168736 & -0,331264 & 0,500002 \\ 0,500000 & -0,418688 & -0,081312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}.$$

Po pretvorbi je večino prostorske informacije o sliki shranjene v svetlostni komponenti Y, barvni komponenti Cb in Cr pa vsebujeta manj pomembne podatke. Zato lahko enostavno odstranimo vsako drugo vrednost v vsakem stolpcu in vsaki vrstici komponent Cb in Cr in s tem dobimo barvni format, ki se imenuje 4:2:0. JPEG podpira tudi formata 4:2:2 in 4:4:4. V formatu 4:2:2 ima vsaka barvna komponenta enako stolpcev kot svetlostna komponenta, v vrsticah barvnih komponent pa odstranimo vsako drugo vrednost. V formatu 4:4:4 ne odstranimo nobene informacije, zato imata barvni komponenti enako število vrstic in stolpcev, kot svetlostna komponenta. Seveda se informacije stisnejo in izgubijo samo v formatih 4:2:0 in 4:2:2. Ta transformacija je potrebna le za barvne slike, črno-bele slike imajo en kanal in očitno ne potrebujejo pretvorbe v drug barvni sistem.

7. RAZBITJE SLIKE NA BLOKE VELIKOSTI 8×8

S pretvorbo barvnega sistema lahko dobimo različne dimenzijske komponente Y, Cb in Cr. Vsaka komponenta je razdeljena v neprekrijoče bloke velikosti 8×8 . Če velikosti komponent niso celoštevilsko deljive z 8, se par piksov z roba odstrani ali doda. Zakaj ravno bloke velikosti 8×8 ? Z izbiro večjih blokov bi bile operacije na njih bolj zahtevne in zato časovno bolj potratne. Poleg tega pa z manjšimi bloki dosežemo, da so piksli bolj povezani med seboj. Če pogledamo na primer kos neke slike v levem spodnjem kotu in kos v desnem zgornjem kotu, piksli niso odvisni drug od drugega, medtem ko so piksli blizu skupaj bolj povezani med seboj. Logično bi potem lahko izbrali čim manjše bloke, a se izkaže, da manjši bloki ne vsebujejo dovolj informacije, da bi jih lahko dobro stisnili. Zato je smiselno, če uporabimo bloke velikosti 8×8 , primerno pa bi bilo uporabiti tudi bloke velikosti 16×16 . Razlike, ki se pokažejo na sliki, pri uporabi blokov velikosti 2×2 , 4×4 in 8×8 , so prikazane na sliki 9.

8. TRANSFORMACIJA

Vsak blok velikosti 8×8 najprej transformiramo. Če bi stiskanje opravili direktno na vrednostih trenutne matrike, ki predstavlja sliko, bi pri rekonstrukciji prišlo do ogromne napake, saj ne bi vedeli kateri elementi matrike so pomembni in kateri ne. Ideja transformacije je, da matriko transformiramo v drugo domeno, kjer bomo vedeli, katere vrednosti so bolj pomembne. Take bomo ohranili in ostale odstranili,



SLIKA 9. Na levi je originalna slika, desno pa stisnjene slike z uporabo blokov velikosti 2×2 , 4×4 in 8×8

nato pa izvedli inverzno transformacijo in s tem dobili rekonstruirano sliko z majhno napako, ki jo lahko opišemo s korenom povprečne kvadratne napake.

Definicija 8.1. Naj bo originalna slika predstavljena z matriko $A \in \mathbb{R}^{m \times n}$ in rekonstruirana slika z matriko $\bar{A} \in \mathbb{R}^{m \times n}$. Potem je koren povprečne kvadratne napake (RMSE), s katerim lahko opišemo kvaliteto stiskanja slik, enak

$$RMSE = \left[\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (\bar{a}_{ij} - a_{ij})^2 \right]^{\frac{1}{2}}$$

Algoritem JPEG uporablja diskretno kosinusno transformacijo.

8.1. Diskretna kosinusna transformacija. Diskretna kosinusna transformacija je osnova mnogih algoritmov za stiskanje fotografij in video posnetkov. Razvil jo je Nasir Ahmed leta 1974, v proces stiskanja slik pa sta jo vpeljala W. H. Chen in W.K. Pratt leta 1984. S procesom diskretne kosinusne transformacije pretvorimo sliko iz prostorske domene v frekvenčno domeno, s katero dobimo obliko slike, v kateri se lažje skoncentriramo na bolj in manj pomembne informacije. Človeško oko je bolj občutljivo na nižje frekvence in manj občutljivo na višje frekvence, zato lahko višje frekvence enostavno odstranimo in s tem zelo malo vplivamo na kvaliteto slike.

8.1.1. Enodimensionalna diskretna kosinusna transformacija. Diskretna kosinusna transformacija je funkcija $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$, ki pretvori vektor $\mathbf{a} = [a_0, a_1, \dots, a_{N-1}]$ v vektor $\mathbf{b} = [b_0, b_1, \dots, b_{N-1}]$ po predpisu

$$b_n = \sqrt{\frac{2}{N}} c_n \sum_{i=0}^{N-1} a_i \cos \left[\frac{(2i+1)n\pi}{2N} \right], \quad \text{za } n = 0, \dots, N-1,$$

kjer je

$$c_n = \begin{cases} \frac{1}{\sqrt{2}}; & n = 0, \\ 1; & \text{sicer.} \end{cases}$$

8.1.2. Dvodimensionalna diskretna kosinusna transformacija. Ker je slika dvodimensionalni objekt, je relevantna dvodimensionalna diskretna kosinusna transformacija. Matriko $A \in \mathbb{R}^{N \times N}$ transformiramo tako, da nejene elemente a_{ij} , za $i, j = 0, 1, \dots, N-1$, preslikamo v

$$(5) \quad b_{mn} = \frac{1}{\sqrt{2N}} c_m c_n \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} a_{ij} \cos \left[\frac{(2i+1)m\pi}{2N} \right] \cos \left[\frac{(2j+1)n\pi}{2N} \right],$$

$$c_n = \begin{cases} \frac{1}{\sqrt{2}}; & n = 0, \\ 1; & \text{sicer.} \end{cases}$$

Pri JPEG algoritmu se diskretna kosinusna transformacija izvaja na blokih velkosti 8×8 , zato je $N = 8$, i in j pa med 0 in 7. Enačba za diskretno kosinusno transformacijo, ki se uporablja pri stiskanju slik z osnovnim JPEG algoritmom na blokih velikosti 8×8 , je torej

$$b_{mn} = \frac{1}{4} c_m c_n \sum_{i=0}^7 \sum_{j=0}^7 a_{ij} \cos \left[\frac{(2i+1)m\pi}{16} \right] \cos \left[\frac{(2j+1)n\pi}{16} \right].$$

Transformacijo na matriki A lahko izvedemo tudi z množenjem matrik

$$B = TAT^T,$$

kjer je T matrika, ki jo lahko dobimo iz (5) po formuli

$$(6) \quad T_{mn} = \begin{cases} \frac{1}{\sqrt{N}}; & m = 0, \\ \sqrt{\frac{2}{N}} \cos \left[\frac{(2n+1)m\pi}{2N} \right]; & \text{sicer.} \end{cases}$$

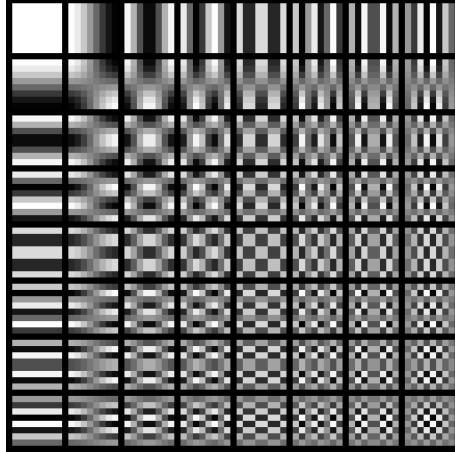
Za bloke velikosti 8×8 je matrika T enaka

$$T = \begin{bmatrix} 0,3536 & 0,3536 & 0,3536 & 0,3536 & 0,3536 & 0,3536 & 0,3536 & 0,3536 \\ 0,4904 & 0,4157 & 0,2778 & 0,0975 & -0,0975 & -0,2778 & -0,4157 & -0,4094 \\ 0,4619 & 0,1913 & -0,1913 & -0,4619 & -0,4619 & -0,4619 & 0,1913 & 0,4619 \\ 0,4157 & -0,0975 & -0,4904 & -0,2778 & 0,2778 & 0,4904 & 0,0975 & -0,4157 \\ 0,3536 & -0,3536 & -0,3536 & 0,3536 & 0,3536 & -0,3536 & -0,3536 & 0,3536 \\ 0,2778 & -0,4904 & 0,0975 & 0,4157 & -0,4157 & -0,0975 & 0,4904 & -0,2778 \\ 0,1913 & -0,4619 & 0,4619 & -0,1913 & -0,1913 & 0,4619 & -0,4619 & 0,1913 \\ 0,0975 & -0,2778 & 0,4157 & -0,4904 & 0,4904 & -0,4157 & 0,2778 & -0,0975 \end{bmatrix}.$$

V prvi vrstici matrike, kjer je $k = 0$, so vsa števila enaka $1/\sqrt{8}$, kot je pričakovano po (6). Stolpci matrike T tvorijo ortonormirano množico, zato je T ortogonalna matrika. Ker so slike sestavljene iz stotih ali tisočih blokov pikslov velikosti 8×8 , je postopek transformacije, ki se izvede na enem bloku, samo mikroskopen del JPEG algoritma. Transformiran blok, ki ga dobimo, je sestavljen iz 64 koeficientov. Koeficient v zgornjem levem kotu c_{00} predstavlja najnižjo frekvenco originalnega bloka slike. Ko se pomikamo po matriki stran od koeficiente c_{00} , desno in navzdol, koeficienti predstavljajo vedno višje frekvence. Koeficient c_{77} predstavlja najvišjo frekvenco. To se lepo vidi na sliki 10. Človeško oko je najbolj občutljivo na nižje frekvence, zato je vedno prvi koeficient c_{00} , ki predstavlja najnižjo frekvenco, največji in najpomembnejši. Vrednosti ostalih koeficientov pa se postopoma zmanjšujejo.

8.2. Druge transformacije. Poznamo tudi druge oblike transformacij, ki se lahko uporabljajo pri stiskanju fotografij. Še boljša transformacija od diskretno kosinusne transformacije, v smislu da minimizira koren povprečne kvadratne napake, je Karhunen-Loèeve transformacija. Razvila sta jo finski aktuar Kari Karhunen (1915-1992) in francoz Michel Loèeve (1907-1979). Glavna razlika med diskretno kosinusno transformacijo in Karhunen-Loèeve transformacijo je v tem, da pri prvi že imamo transformacijsko matriko, ki jo uporabimo pri transformaciji katerekoli slike podane z matriko iste velikosti, pri drugi pa je transformacijska matrika za vsako sliko drugačna. Tako moramo pred transformacijo vedno posebej izračunati transformacijsko matriko, ki jo dobimo po naslednjem postopku. Recimo da imamo set vektorjev $X = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots]$, ki ustreza vrsticam matrike neke slike. Najprej izračunamo povprečni vektor

$$m_X = E[X],$$



SLIKA 10. Frekvence, ki jih predstavljajo koeficienti transformiranega bloka velikosti 8×8 .

kjer je E pričakovana vrednost. Nato definiramo kovariančno matriko

$$C_X = E[(X - m_X)(X - m_X)^T].$$

Poščemo lastne vrednosti in lastne vektorje matrike C_X . Transformacijska matrika je

$$T = [e_1, e_2, e_3, \dots],$$

kjer so e_1, e_2, e_3, \dots lastni vektorji kovariančne matrike C_X , ki pripadajo lastnim vrednostim $\lambda_1, \lambda_2, \lambda_3, \dots$, za katere valja $\lambda_1 > \lambda_2 > \lambda_3 > \dots$. Kljub temu da je Karhunen-Loëve transformacija boljša od diskretno kosinusne transformacije, se v JPEG algoritmu ne uporablja, saj je zamudna za računanje.

9. KVANTIZACIJA

V koraku kvantizacije pride do dejanskega stiskanja. Z uporabo različnih kvantizacijskih matrik lahko dosežemo različne stopnje stiskanj na območju od 1 do 100, kjer 1 poda najslabšo kvaliteto slike in najvišjo stopnjo stiskanja, med tem ko 100 poda najboljšo kvaliteto slike in najnižjo stopnjo stiskanja. Subjektivni eksperimenti, narejeni na podlagi človeškega vida, so pokazali, da je najbolje uporabiti stopnjo stiskanja 50. S tem dobimo oboje, visoko stopnjo stiskanja in odlično kvalitetno končne rekonstruirane slike. Kvantizacijsko matriko stopnje 50 zato imanujemo standardna kvantizacijska matrika in jo označimo s Q_{50} .

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Kvantizacijo dostežemo tako, da elemene transformirane matrike, ki jo dobimo z diskretno kosinusno transformacijo, delimo z elementi kvantizacijske matrike in jih zaokrožimo na najbližje celo število. Naj bo Q poljubna kvantizacijska matrika in B

transformirana matrika, ki jo dobimo z uporabo diskretne kosinusne transformacije iz matrike, ki predstavlja neko sliko. Kvantizirano matriko označimo z C in jo izračunamo po formuli

$$c_{ij} = \left\lfloor \frac{b_{ij}}{q_{ij}} \right\rfloor.$$

Za drugačno stopno stiskanja uporabimo kvantizacijsko matriko, ki jo dobimo tako, da standardno kvantizacijsko matriko pomnožimo z določenim skalarjem. Naj bo željena stopnja stiskanja enaka n . Za stopnjo večjo od 50 (manjše stiskanje in boljša kvaliteta slike) standardno kvantizacijsko matriko pomnožimo s kvocientom $\frac{100-n}{50}$, med tem ko za stopnjo manjšo od 50 (večje stiskanje in slabša kvaliteta slike) standardno kvantizacijsko matriko pomnožimo s kvocientom $\frac{50}{n}$. Vse elemente matrike nato še zaokrožimo na cela števila. Kvantizacijsko matriko s stopnjo stiskanja n označimo s Q_n . Matriki za $n = 10$ in $n = 90$ sta:

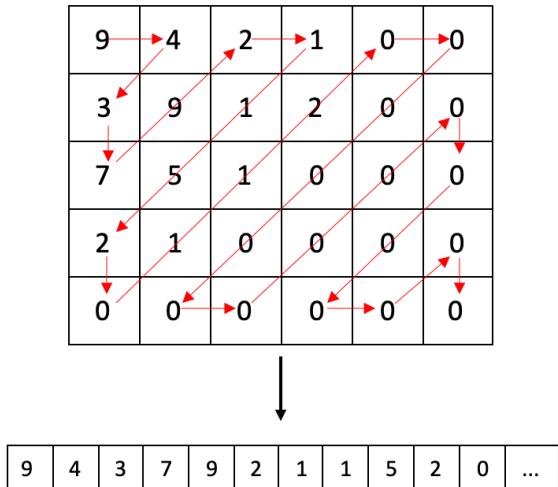
$$Q_{10} = \begin{bmatrix} 80 & 66 & 50 & 80 & 120 & 200 & 255 & 305 \\ 60 & 60 & 70 & 95 & 130 & 290 & 300 & 275 \\ 70 & 65 & 80 & 120 & 200 & 285 & 345 & 280 \\ 70 & 85 & 110 & 145 & 255 & 435 & 400 & 310 \\ 90 & 110 & 185 & 280 & 340 & 545 & 515 & 385 \\ 120 & 175 & 275 & 320 & 405 & 520 & 565 & 460 \\ 245 & 320 & 390 & 435 & 515 & 605 & 600 & 505 \\ 360 & 460 & 295 & 490 & 560 & 500 & 515 & 495 \end{bmatrix}$$

$$Q_{90} = \begin{bmatrix} 3 & 2 & 2 & 3 & 5 & 8 & 10 & 12 \\ 2 & 2 & 3 & 4 & 5 & 12 & 12 & 11 \\ 3 & 3 & 3 & 5 & 8 & 11 & 14 & 11 \\ 3 & 3 & 4 & 6 & 10 & 17 & 16 & 12 \\ 4 & 4 & 7 & 11 & 14 & 22 & 21 & 15 \\ 5 & 7 & 11 & 13 & 16 & 12 & 23 & 18 \\ 10 & 13 & 16 & 17 & 21 & 24 & 24 & 21 \\ 14 & 18 & 19 & 20 & 22 & 20 & 20 & 20 \end{bmatrix}.$$

Po procesu kvantizacije dobimo matriko, ki ima v zgornjem levem kotu največja števila, ki se v smeri desno in navzdolj zmanjšujejo, v desnem spodnjem kotu pa je veliko elementov enakih 0. Kot že prej omenjeno, koeficienti v levem zgornjem kotu, ki imajo višje vrednosti, predstavljajo nižje frekvence na katere je človeško oko najbolj občutljivo, med tem ko koeficienti v desnem spodnjem kotu, ki so manjši ali enaki 0, predstavljajo višje frekvence, na katere človeško oko ni tako občutljivo. Če bi pri procesu kvantizacije namesto standardne kvantizacijske matrike uporabili matriko Q_{10} , bi v kvantizirani matriki C desno spodaj dobili več ničel in manj ničel, če bi uporabili matriko Q_{90} .

10. KODIRANJE

Sledi še zadnji korak stiskanja. Na vsaki kvantizirani matriki izvedemo cik-cak skeniranje, kot je prikazano na sliki 11. Kot rezultat cik-cak skeniranja posameznega bloka velikosti 8×8 dobimo zaporedje 64 števil. Prvo število bo največje, nato sledi par manjših števil in na koncu veliko ničel. Ker se pojavi veliko število ničel, ne



SLIKA 11. Cik-cak skeniranje.

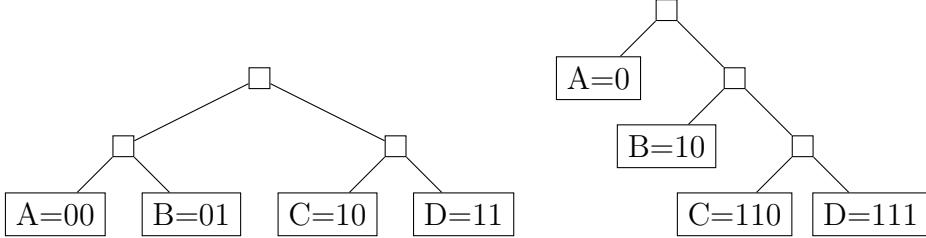
rabimo shraniti vsake ničle posebej, namesto tega lahko shranimo le en znak, ki bo predstavljal vse ničle naenkrat. Na ta način podatke zakodiramo in shranimo s Huffmannovim kodiranjem.

10.1. Huffmanovo kodiranje. S Huffmannovim kodiranjem na enostaven način stisnemo podatke z uporabo Huffmannovega algoritma. Algoritem je razvil David A. Huffman in ga objavl leta 1952, še danes pa ga uporablja veliko različnih formatov za stiskanje podatkov. Ideja Huffmannovega kodiranja je naslednja. V besedilih z določeno abecedo se vedno kakšni znaki ali zaporedja znakov pojavijo večkrat, zato si lahko računalnik namesto vseh teh znakov zapolni le en znak in njegovo frekvenco. Poglejmo si podrobneje na naslednjem primeru. Recimo da imamo besedilo, v katerem se simboli A, B, C in D pojavijo z naslednjimi frekvencami:

Simbol	Frekvenca
A	80
B	30
C	22
D	5

Simbol A bi lahko predstavili na primer z 00, B z 01, C z 10 in D z 11. S tako predstavljivo bi za posamezni simbol potrebovali 2 bita za zapis in za vse skupaj $80 \times 2b + 30 \times 2b + 22 \times 2b + 5 \times 2b = 274b$. Ali slučajno obstaja boljše kodiranje kot to? Opazimo da se simbol A pojavi večkrat kot simbol B, zato bi za njegov zapis lahko uporabili manj bitov in za zapis simbola B več bitov. Recimo da bi za zapis simbola A izbrali 0. Potem si za simbol, ki ima naslednjo frekvenco po velikosti, se pravi B, lahko izberemo 10, nato za C 110 in za D 111. Pri izbiri takšnih kod moramo paziti na enoličnost, zato si vedno izberemo kode, ki jim pravimo brezpredponske kode, kar pomeni, da nobena koda ni predpona kakšne druge kode. Z izbiro takšnih kod pa bi za zapis potrebovali $80 \times 1b + 30 \times 2b + 22 \times 3b + 5 \times 3b = 221b$, kar je že boljše. Da pridemo do najbolj optimalne izbire kodiranja, si pomagamo z odločitvenim drevesom. Vsaka izbira brezpredponskih kod ustrezava Huffmannovem

drevesu. Drevesi za prej omenjeni kodiranji sta:



Globina nekega elementa v drevesu predstavlja ravno število bitov, ki jih potrebujemo za njegovo predstavitev. Če to pomnožimo še z njegovo frekvenco, dobimo število, ki ga porabimo za predstavitev vseh pojavitev tega simbola. Na ta način lahko definiramo mero, ki predstavlja kako dobro je drevo. To mero imenujemo cena drevesa.

Definicija 10.1. Za poljubno drevo H in število pojavitev oziroma frekvenco $f : \Gamma \rightarrow [0, \infty)$, kjer je Γ abeceda, je cena drevesa $E(f, H)$ enaka:

$$E(f, H) = \sum_{x \in \Gamma} (\text{globina } x \text{ v } H) \times f(x)$$

Za dani f , iščemo takšen H , da bo cena drevesa $E(f, H)$ najmanša.

Trditev 10.2. Naj bo H optimalno drevo, Γ abeceda in $f : \Gamma \rightarrow [0, \infty)$ število pojavitev oz. frekvence. Če sta $x \in \Gamma$ in $y \in \Gamma$ najglobiji vozlišči v optimalnem drevesu H , sta $f(x)$ in $f(y)$ najmanjši vrednosti f .

Dokaz. To lahko dokažemo s protislovjem. Recimo, da x in y nista najglobiji vozlišči optimalnega drevesa H . Zato lahko enega od njiju brez škode zamenjamo z bolj redkim simbolom in dobimo učinkovitejše drevo. Kar pa je v nasprotju s predpostavko, ki pravi, da je H že optimalno drevo. \square

Vozlišča v drevesu H lahko združimo na naslednji način:

- vozlišče z otroki x in y označimo z xy ,
- nastavimo frekvenco tega vozlišča kot vsoto frekvenc vozlišč x in y , $f(xy) = f(x) + f(y)$.

Če sta x in y najglobiji vozlišči v optimalnem drevesu H in $f(x)$ frekvanca simbola x , je

$$\sum_{v \in V(H)} f(v) = f(x) + f(y) + \sum_{v \in V(H), v \notin \{x, y\}} f(v)$$

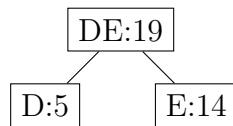
Z združevanjem vozlišč poiščemo optimalno Huffmannovo drevo, kot je prikazano v primeru 10.3 in algoritmu 1. Časovna zahtevnost Huffmannovega algoritma je $O(nL)$, kjer je L maksimalna dolžina kodirane besede.

Primer 10.3. Poiščimo optimalno drevo za naslednje simbole in njihove frekvence:

Simbol	Frekvanca
A	80
B	30
C	22
D	5
E	14

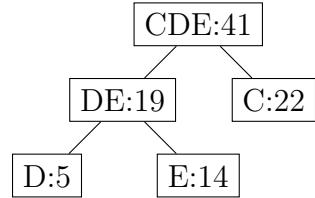
V drevesu bosta najglobje simbola z najnižjo frekvenco, v našem primeru sta to D in E, zato ju najprej dodamo v drevo in združimo v eno vozlišče, tako da seštejemo njuni frekvenci. Dobimo novo tabelo vozlišč in drevo:

Simbol	Frekvenca
A	80
B	30
C	22
DE	19



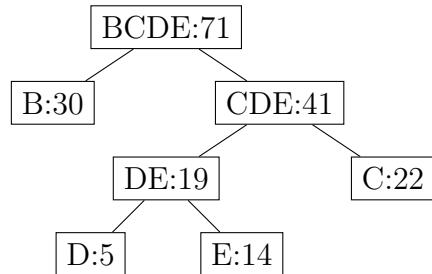
V novi tabeli pogledamo katera simbola imata najnižjo frekvanco, to sta DE in C. Ta dva ponovno dodamo v drevo in ju združimo skupaj s seštevanjem frekvenc:

Simbol	Frekvenca
A	80
B	30
CDE	41

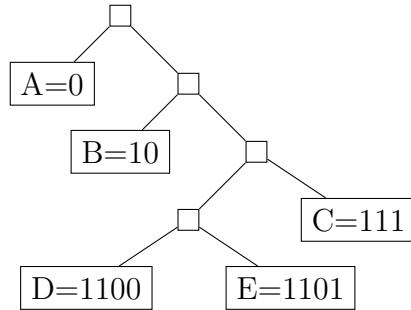


Nadaljujemo enako in združimo vozlišči B in CDE:

Simbol	Frekvenca
A	80
BCDE	71



Na koncu združimo še preostali dve vozlišči in dobimo optimalno Huffmanovo drevo z dodanimi brezpredponskimi kodami:



◇

Algoritem 1 Huffmanov algoritem

Vhod: A , seznam znakov ali informacij skupaj z njihovimi frekvencami, ki jih želimo zakodirati.

Izhod: Optimalno Hufmanovo drevo, predstavljeno z vrsto Q .

```

1: function HUFFMANOV ALGORITEM( $A$ )
2:    $n \leftarrow$  dolžina  $A$                                      ▷  $n$  naj bo dolžina seznama  $A$ .
3:    $Q \leftarrow$  prioritetna vrsta                                ▷  $Q$  naj bo prazna prioritetna vrsta.
4:   for  $i \leftarrow 1$  to  $n$  do
5:      $v \leftarrow$  vozlišče( $A[i]$ )                               ▷ Za vsak znak ustvarimo vozlišče.
6:      $Q.\text{push}(v)$                                          ▷ Znake kot vozlišča dodamo v prioritetno vrsto.
7:   end for
8:   while dolžina  $Q$  ni enaka 1 do
9:      $x \leftarrow Q.\text{pop}$                                      ▷ Iz vrste odstranimo dva elementa z najnižjo frekvenco.
10:     $y \leftarrow Q.\text{pop}$ 
11:     $V \leftarrow$  novo vozlišče                                ▷ Ustvarimo novo vozlišče.
12:     $V.\text{levo} \leftarrow x$ 
13:     $V.\text{desno} \leftarrow y$ 
14:     $V.\text{frekvenca} \leftarrow x.\text{frakvenca} + y.\text{frakvenca}$  ▷ Nastavimo novo frekvenco.
15:     $Q.\text{push}(V)$                                          ▷ Novo vozlišče dodamo v vrsto s prednostjo.
16:   end while
17:   return  $Q$ 
18: end function

```

11. RAZŠIRITEV

Ko sliko shranimo z JPEG algoritmom, se shrani to, kar smo dobili kot rezultat Huffmanovega kodiranja. Če želimo sliko nazaj odpreti, moramo vse operacije izvesti v obratnem vrstnem redu stiskanja:

- (1) Dekodiranje stisnjениh podatkov z uporabo Huffmanovega slovarja, definiranega pri Huffmanovem algoritmu.
- (2) Obrat cik-cak skeniranja.
- (3) Dekvantizacija.
- (4) Inverzna diskretna kosinusna transformacija.
- (5) Združevanje blokov velikosti 8×8 v matriko.
- (6) Pretvorba barvnega sistema ($YCbCr \rightarrow RGB$).

Ko dobimo rekonstruirane bloke velikosti 8×8 izvedemo proces dekvantizacije, tako da jih pomnožimo z kvantizacijsko matriko, ki smo jo uporabili v procesu kvantizacije. Nato bloke transformiramo nazaj v prvotno domeno z uporabo inverzne diskretne kosinusne transformacije:

$$\bar{a}_{mn} = \frac{2}{\sqrt{N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c_i c_j \bar{b}_{ij} \cos \left[\frac{(2i+1)m\pi}{2N} \right] \cos \left[\frac{(2j+1)n\pi}{2N} \right],$$

kjer so \bar{b}_{ij} elementi dekvantizirane matrike \bar{B} in

$$c_i = \begin{cases} \frac{1}{\sqrt{2}}; & i = 0, \\ 1; & \text{sic er.} \end{cases}$$

Inverzno diskretno kosinusno transformacijo izvedemo na vsakem bloku velikosti 8×8 in bloke nazaj sestavimo v matriko. Če smo pri razdelitvi matrike na bloke velikosti 8×8 odstranili ali dodali nekaj elementov, jih tukaj enostavno dodamo, toliko kot smo jih odstranili, bodisi odstranimo, toliko kot smo jih dodali. Na koncu še pretvorimo barvni sistem, iz YCbCr nazaj v RGB s transformacijo

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 & 1.40210 \\ 1.0 & -0.34414 & -0.71414 \\ 1.0 & 1.77180 & 0.0 \end{pmatrix} \begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} - \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}.$$

11.1. Rezultati. S procesom razširitve dobimo rekonstruirano matriko, ki ponavadi ni enaka originalni, saj smo nekaj podatkov med procesom stiskanja odstranili. Skoraj nemogoče je, da bi jih v celoti dobili nazaj. Kjub temu pa sta si originalna in rekonstruirana matrika še vedno zelo podobni. Poglejmo si potek algoritma in primerjavo med originalno in rekonstruirano matriko na naslednjem primeru.

Primer 11.1. Naj bo A blok velikosti 8×8 , ki predstavlja košček neke slike.

$$A = \begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 154 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 154 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{bmatrix}$$

Najprej izvedemo diskretno kosinusno transformacijo in dobimo:

$$B = \begin{bmatrix} 162.3 & 40.6 & 20.0 & 72.3 & 30.3 & 12.5 & -19.7 & -11.5 \\ 30.5 & 108.4 & 10.5 & 32.3 & 27.7 & -15.5 & 18.4 & -2.0 \\ -94.1 & -60.1 & 12.3 & -43.4 & -31.3 & 6.1 & -3.3 & 7.1 \\ -38.6 & -83.4 & -5.4 & -22.2 & -13.5 & 15.5 & -1.3 & 3.5 \\ -31.3 & 17.9 & -5.5 & -12.4 & 14.3 & -6.0 & 11.5 & -6.0 \\ -0.9 & -11.8 & 12.8 & 0.2 & 28.1 & 12.6 & 8.4 & 2.9 \\ 4.6 & -2.4 & 12.2 & 6.6 & -18.7 & -12.8 & 7.7 & 12.0 \\ -10.0 & 11.2 & 7.8 & -16.3 & 21.5 & 0.0 & 5.9 & 10.7 \end{bmatrix}$$

Nato izvedemo proces kvantizacije:

$$C = \begin{bmatrix} 10 & 4 & 2 & 5 & 1 & 0 & 0 & 0 \\ 3 & 9 & 1 & 2 & 1 & 0 & 0 & 0 \\ -7 & -5 & 1 & -2 & -1 & 0 & 0 & 0 \\ -3 & -5 & 0 & -1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Na koncu matriko cik-cak poskeniramo in izvedemo Huffmanovo kodiranje. Ko matriko nazaj razširimo, dobimo matriko \bar{A} :

$$\bar{A} = \begin{bmatrix} 149 & 134 & 119 & 116 & 121 & 126 & 127 & 128 \\ 204 & 168 & 140 & 144 & 155 & 150 & 135 & 125 \\ 253 & 195 & 155 & 166 & 183 & 165 & 131 & 111 \\ 245 & 185 & 148 & 166 & 184 & 160 & 124 & 107 \\ 188 & 149 & 132 & 155 & 172 & 159 & 141 & 136 \\ 132 & 123 & 125 & 143 & 160 & 166 & 168 & 171 \\ 109 & 119 & 126 & 128 & 139 & 158 & 168 & 166 \\ 111 & 127 & 127 & 114 & 118 & 141 & 147 & 135 \end{bmatrix}$$

◊

Razlika se pokaže na vsakem bloku velikosti 8×8 , zato je končna slika malo drugačna od prvotne. Ker črno-bele slike vsebujejo do 256 različnih odtenkov, je razlika med elementi originalne in rekonstruirane matrike komaj opazna s človeškim očesom. Pri barvnih slikah, ki vsebujejo tri barvne kanale, pa je ta razlika še manjša. Na sliki 12 je fotografija psa, na sliki 13 pa stisnjana fotografij iz slike 12, za stopnjo stiskanja 50, 10 in 5, z izbranim formatom 4:2:0. Na sliki, kjer je uporabljena stopnja kompresije 50, je s prostim očesom nemogoče opaziti razliko. Pri slikah s stopnjo kompresije 10 in 5 pa se opazi manjša razlika. Glede na to da smo toliko podatkov iz slike odstranili, je rezultat izjemno dober.



SLIKA 12. Slika psa.



SLIKA 13. Apriksimacija slike 12 pri stopnji kompresije 50, 10 in 5 s formatom 4:2:0

12. ALGORITEM

Algoritem 2 JPEG algoritem

Vhod: Slika, predstavljena z matriko A in stopnja stiskanja n .

Izhod: Rekonstruirana slika, predstavljena z matriko \bar{A}

```
1: function JPEG_ALGORITEM( $A, n$ )
2: end function
```

13. DRUGI FORMATI ZA STISKANJE SLIK

Dva bolj pogosta formata za stiskanje slik sta tudi GIF (Graphic Interchange Format) in PNG (Portable Network Graphics). Prvega so razvili leta 1987 z namenom učinkovitega prenašanja slik preko interneta, zato ga danes podpirajo vsi brskalniki in se ga veliko uporablja. Temelji na stiskanju braz izgub in zato, za razliko od formata JPEG, rekonstruira sliko, ki je enaka originalni, ampak stisnjena datoteka zavzame več prostora. Format podpira le 8 bitni zapis pikslov, kar posamezni sliki omogoča prikaz 256 različnih barv iz celotnega barvnega modela RGB. Podpira kombiniranje več slik v eno in ustvarjanje animacij. Stiskanje slik doseže z naslednjim postopkom. Najprej zmanjša število različnih barv v sliki, tako da podobne barve združi v bloke. Nato zakodira sliko z uporabo Lempel-Ziv-Welch algoritma, ki skenira sliko po vrsticah in zamenja večkratne ponovitve pikslov z njihovimi vrednostmi in številom ponovitev. Zaradi določenih slabosti tega formata, so iz njega razvili format PNG. Čeprav format GIF podpira tudi animacije, je PNG namenjen le shranjevanju mirujočih črno-belih in barvnih fotografij. Razvil se je kasneje, leta 1999 in deluje podobno kot format GIF. Prav tako kot GIF temelji na stiskanju brez izgub. Poleg 8 bitov na piksel podpira tudi 24 bitni barvni sistem RGB. Vsak od teh formatov je dober za stiskanje določenih fotografij. PNG ohrani dobro sliko, saj med stiskanjem obdrži vse informacije in je zato bolj primeren za shranjevanje manjših slik. GIF se danes uporablja le za animacije, vendar pokaže lepe rezultate tudi pri manjših 8 bitnih slikah. JPEG pa je še vedno glavni med vsemi formati

za stiskanje slik, a se, zaradi stiskanja z izgubami, slika lahko s postopnim shranjevanjem razgradi. Primerjava med kakovostjo slik in velikostjo datotek med formati JPEG, GIF in PNG je prikazana na sliki 14.



SLIKA 14. Leva slika je shranjena s formatom PNG z 92KB, srednja s formatom JPEG z 38KB in desna s formatom GIF s 65 KB.

14. KONEC DELA

SLOVAR STROKOVNIH IZRAZOV

compression stiskanje, kompresija

decompression razširjanje, dekompresija

lossy compression stiskanje z izgubami

lossless compression stiskanje brez izgub

luminance svetilnost

crominance barvnost

Joint photographic expert group Skupna skupina fotografskih strokovnjakov
discrete cosine transform diskretna kosinusna transformacija

singular value decomposition singularni razcep

United nations specialized agency for information and communication technologies Mednarodna telekomunikacijska zveza

International organization for standardization Mednarodna organizacija za standardizacijo

International electrotechnical commission Mednarodna komisija za elektrotehniko

eigenvalues lastne vrednosti

eigenvectors lastni vektorji

root mean square error koren povprečne kvadratne napake

quantization kvantizacija

quantization matrix kvantizacijska matrika

zig-zag scanning cik-cak skeniranje

Portable network graphics Prenosna mrežna grafika

Graphics interchange format Format grafične izmenjave

LITERATURA

- [1] T. Acharya in P. Tsai, *JPEG2000 standard for image compression:concepts algorithms and VLSI architectures*, **1**, Wiley-interscience, New Jersey, 2005.
- [2] M. Barni, *Document and image compression*, **1**, Taylor & Francis Group, Florida, 2006.
- [3] S.T. Acton in G. Aggarwal, *Handbook of Image and Video Processing*, **2**, Academic Press, Burlington, 2005.
- [4] K.R. Acton in G. Aggarwal, *Handbook of Image and Video Processing*, **2**, Academic Press, Burlington, 2005.
- [5] B. Plestenjak, *Numerične metode, delovna verzija*, 2010.
- [6] A.M. Rao in P. Yip, *Discrete Cosine Transform Algorithms, Advantages, Applications*, **1**, Academic Press, Boston, 1990.
- [7] E. Prathibha, A. Manjunath in R. El-Likitha, *RGB to YCbCr Color Conversion using VHDL approach*, International Journal of Engineering Research and Development **1** (2012) 15–22.
- [8] K. Cabeen in P. Gent, *Image compression and the discrete cosine transform*, Geom. **45** (2011) 88–140.
- [9] R. Cogranne, *Determining JPEG Image Standard Quality Factor from the Quantization Tables*, Troyes University of Technology **4** (2018) 1–6.
- [10] A. Shawahna, E. Haque in A. Amin, *JPEG Image Compression using the Discrete Cosine Transform: An Overview, Applications, and Hardware Implementation.*, Department of Computer Engineering King Fahd University of Petroleum and Minerals **1** (2019) 1–7.
- [11] A. Mazhir in H.A. Fouad, *The jpeg image compression algorithm*, IJAET **6** (2013) 1055–1062.
- [12] S. Dumas, *Karhunen-Loeve Transform and Digital Signal Processing*, (2016) 1–33.
- [13] Visual Resource Centre School of Humanities, *Understanding digital images*, University of Reading (2008) 1–8.
- [14] S. Kumar, *Image Compression*, Oktober 2001, [ogled 30. 07. 2020], dostopno na <https://www.debugmode.com/imagecmp/index.htm>.
- [15] M. Kozmus, *Barvna globina*, [ogled 30. 07. 2020], dostopno na http://www.egradiva.net/moduli/racunalnisko_oblikovanje/91_barvni_modeli/06_datoteka.html.
- [16] A.B. Watson, *Image Compression Using the Discrete Cosine Transform*, Avgust 1994, [ogled 12. 01. 2020], dostopno na https://www.researchgate.net/publication/2800608_Image_Compression_Using_the_Discrete_Cosine_Transform.
- [17] *Image compression*, v: Wikipedia, The Free Encyclopedia, [ogled 12. 01. 2020], dostopno na https://en.wikipedia.org/wiki/Image_compression.
- [18] *Discrete cosine transform*, v: Wikipedia, The Free Encyclopedia, [ogled 22. 07. 2020], dostopno na https://en.wikipedia.org/wiki/Discrete_cosine_transform.
- [19] *JPEG*, v: Wikipedia, The Free Encyclopedia, [ogled 22. 07. 2020], dostopno na <https://en.wikipedia.org/wiki/JPEG>.
- [20] *Pixel*, v: Wikipedia, The Free Encyclopedia, [ogled 22. 07. 2020], dostopno na <https://en.wikipedia.org/wiki/Pixel>.