**Poročilo izbirnega projekta 7B – Implementacija kodiranja in dekodiranja slik po postopku QOI**

Informacije in Kodi

Študent: Luka Melinc



Mentorja: asist. Dr. Klemen Grm, izr. prof. dr. Simon Dobrišek

Datum: 1.5.2023

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Uvod

Za izbirni projekt oziroma četrto laboratorijsko vajo pri predmetu sem si izbral projekt implementacije kodiranja in dekodiranja slik po postopku QOI(F) ali Quite OK Image (Format). Gre za brezizgubno kompresijo slik, ki deluje hitreje kot kompresija pri formatu PNG, zaradi česar se tak format kompresije uporablja v primerih, ko je potrebno hitro procesiranje slik. Eden izmed primerov uporabe je v videoigrah za rendering grafik v realnem času. V prihodnosti je napovedan tudi razvoj kodirnega sistema z podobnim pristopom za kodiranje audio datotek. Prednost tega pristopa je tudi to, da QOI ne potrebuje kakšnih posebnih knjižnic, ampak se ga lahko poganja z zelo osnovnimi knjižnicami, kar ni možno za PNG ali JPEG, ki potrebujeta obsežne dodatne knjižnice.

Koncept QOI formata je preprost in podpira tako RGB kot RGBA barvne modele. Pomemben del formata je t.i. *header* struktura z meta podatki, ki vsebuje bistvene informacije o datoteki. *Magic* bajti identificirajo datoteko kot QOIF datoteko, navedena je širina in višina datoteke, koliko barvnih kanalov vsebuje slika (3 ali 4).

Sam format je sestavljen iz treh načinov kodiranje vrednosti pikslov. Prvi je tako imenovan RLE pristop, ki kompresira zaporedja sekvenc istih pikslov. Ta pristop je zelo uspešen, kadar se zgodi, da so na sliki, ki jo kompresiramo, večja območja enakih pikslov. Drugi pristop je t.i. *indexing*, kjer QUI uporablja lookup tabelo vrednosti pikslov, ki so se zgodili pred dotičnim pikslom, tako se lahko ponovno uporabijo vrednosti pikslov, brez da bi se shranile njihove vrednosti. Tretji način, ki ga format uporablja, temelji na malih razlikah med vrednostmi zaporednih pikslov.

Pomemben koncept, ki je implementiran v QOI formatu, so tako imenovane »*opcodes«,* gre za nabor nizov, ki sporočajo, na kakšen način je določeno zaporedje pikslov kompresirano. Vse skupaj obstaja 6 opcod. Gre za dva bajta informacij, pri čemer dva najpomembnejša bita kode povesta, za kakšen način enkodiranja gre. V kolikor sta najpomembnejša bita 00, gre za enkodiranje z lookup tabelo, 01 pomeni enkodiranje z majhnimi razlikami, 10 pomeni podobno enkodiranje kot 01, le da gre v temu primeru za malce večje razlike med piksli, 11 pa pomeni ponavljanje trenutne vrednosti bita. Preostalih šest manj pomembnih bitov nato poda preostale pomembne inforamcije glede kodiranja napreimer dolžina ponavljanja piksla, vrednost male/večje razlike med pikslom, vrednost v lookup tabeli. Posebna primera sta, ko je opcoda 0b11111110 ali 0b11111111 (odvisno, ali ima slika tri ali štiri barvne kanale), ko pikslov ne moremo enkodirati na drugačen način.

Vedno se QOI formatirana slika zaključi s *footerjem*, ki vsebuje sedemkrat po 0x00 ter na koncu 0x01, kar zapre pretok informacij.

Pri temu projektu je bilo potrebno opraviti tri glavne naloge. V prvi je bilo potrebno napisati program, ki vhodno sliko zakodira v format QOI. V drugi nalogi je bilo potrebno to sliko nazaj dekodirati iz QOI v format PNG. V tretji nalogi pa smo vzeli nabor 24 referenčnih slik, jih kodrali v QOI ter jih dekodirali nazaj v PNG ter preverjali, ali so ob procesiranju nastale kakšne napake. Posikali smo tudi srednjo napako med posameznimi piksli slik, primerjali povprečne velikosti datotek med različnimi zapisi ter primerjali čas kodiranja in dekodiranja.

Pri prvi nalogi smo se torej lotili implementacije kode za kompresiranje v QOI format. Za test smo vzeli sliko edgecase.png iz nabora referenčnih slik. Ugotovili smo, da se z našim programom sliko ustrezno zakodira v qoi format, kompresijo smo preverjali z referenčnim kodirnikom, saj smo zakodirano datoteko odkodirali nazaj in ugotovili, da je bil dekompresijski postopek delujoč.

Najprej smo konstruirali header datoteke, identifikacijo formata datoteke, višino in širino ter število barvnih kanalov datoteke ter header zaključili z x00. Postavili smo se na prvi piksel ter zajeli bajte prvega piksla (3 oziroma 4, odvisno od števila barvnih kanalov). Skozi celotno sliko smo iterirali s for zanko, ki ze je začela na začetku z ničtim pikslom, trajal je dolžino vsej pikslov ter se pomikal za toliko bajtov, kolikor je barvnih kanalov v sliki.

Pri drugi nalogi smo se lotili dekodiranja qoi datoteke. V bistvu smo izvedli prvo nalogo v obratnem vrstnem redu. Najprej odpremo glavo programa ter pridobimo informacije o velikosti slike, kanalih, barvnem prostoru. Nato se slikovne podatke dekodira glede na QOI ukaze, ki obravnavajo barvne razlike in kompresijo podatkov. Program napolni buffer podatkov, ki ga nakoncu uporabi za tvorjenje slike.

Kodo sem preizkusil na podanih .qoi slikah. Uspešno sem pretvoril vse slike nazaj v png, razen pri slikah dice vedno dobim na vrhu slike črn pas pikslov, ki ga ne znam odstraniti, pri sliko qoi\_logo pa se pojavi kratek črni pas pikslov, česar tudi nisem znal razrešiti.

Pri tretji nalogi smo najprej morali preveriti, ali lahko uspešno pretvorimo vse izmed 24 referenčnih png slik najprej v qoi format in nato še nazaj v png format. To sem uspešno izvedel, ob temu pa sem ugotovil pomembno stvar glede slik iz kaggla. Ob ponovni pretvorbi nazaj v png format sem ugotovil, da je imela slika po pretvorbi večjo velikost kot originalna slika. To sem preveril z referenčnim pretvornikom, originalno sliko sem pretvoril v qoi format ter nazaj v png, tudi v temu primeru je bila velikost originalne in nazaj-pretvorjene datoteke različna. Predpostavljam, da so v kaggle datotekah določeni meta podatki, ki vplivajo na velikosti datoteke po pretvorbi.

Zato sem se odločil, da bom tretjo vajo izpeljal na referenčnih slikah iz druge naloge, pri je velikost datoteke ostala ista po kompresiji in dekompresiji. Postopek sem izvedel na vseh slikah, le pri zgoraj omenjenih dveh slikah je bila razlika pri pikslih različna od nič, kljub veliko vloženega dela nisem uspel razrešiti napake.

A close-up of several dice

Description automatically generated

Slika 1: Črni pas na vrhu dekodirane png slike

Lotili smo se tudi računanja povprečne velikosti datotek. Za originalne referenčne png slike iz spletne učilnice (8 vzorcev) je bila povprečna velikost slike okoli 362314 bajta, povprečna velikost slike v qoi formatu je bila 429130 bajtov in povprečna velikost slike v formatu bmp je bila 993250 bajtov. Png format je torej najbolj gospodaren, vendar pa je precej bolj kompleksen kot qoi format. BMP se je izkazal kot najmanj efektiven format za kompresijo.



Slika 2: Primer izvajanja programa s funkcijo time

Poleg tega sem tudi izmeril čas trajanja posamezne pretvorbe med dvema formatoma ter primerjal čase pretvorbe z referenčnim pretvornikom in z mojima programoma. V tem primeru sem uporabil funkcijo time, ki izmeri skupni čas izvajanja programa, uporabniški čas, ki ga proces preživi v uporabniškem načinu in sistemski čas, ki ga proces preživi v jedru, vendar pa nas je zanimal samo prvi, skupni čas izvedbe. Rezultati so sledeči, referenčni pretvornik je pretvorbo iz png v qoi format izvedel v 0.036s ter pretvorbo iz qoi v png format v 0.129 sekunde. Moja programa sta pretvorbo iz png v qoi format izvedla v 0.753 sekunde ter v obratni strani v 0.854 sekunde. Pretvorbo smo izvajali na sliki kodim02 petkrat za vsako pretvorbo. Zaostanek mojega programa za referenčnim lahko pripišemo v glavnem časovno neoptimizirani kodi ter tudi temu, da je program napisan v jeziku Python, ki ni najhitrejši programski jezik.

Zaključek

V nalogi smo bolj kot ne uspešno implementirali novi format kodiranja qoif za kodiranje slik. Za legitimnost rezultatov sem raje pri tretji nalogi uporabil manjši nabor slik, vendar pa je po mojem mnenju ta še vedno dovolj velik za legitimnost rezultatov. Implementiran format kodiranja in dekodiranja na precejšnjem delu slik dela brez težav, vendar pa obstajajo še robni pogoji, za katere potrebuje program še malce dodelave.

Implementacija formata je bila sicer normalno zahtevna, dokumentacija je kratka in jedernata, kar vsekakor olajša delo na projektu.