SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 539

FPGA implementacija metode povećanja originalne rezolucije slike korištenjem više snimki

Igor Sočec

Zagreb, lipanj 2012.

Zahvaljujem prof.dr.sc. Davoru Petrinoviću na korisnim savjetima koje mi je pružio tijekom izrade ovog rada.

Sadržaj

1. Uvod

Povećanje originalne rezolucije slike korištenjem više snimki je metoda poznata pod nazivom super rezolucija te ima široku primjenu u obradi medicinskih, vojnih, nadzornih, ali i komercijalnih slika. Metoda je temeljena na činjenici da uzastopne snimke iste scene pod pretpostavkom međusobnih sitnih pomaka sadrže drugačije informacije o snimanoj sceni. Određenim postupcima obrade moguće je iskorištavanjem dodatnih informacija iz više snimki sastaviti sliku koja ima višu rezoluciju od početnih snimki. Postoje razvijeni algoritmi za postizanje super rezolucije koji ovise o konkretnoj primjeni, a cilj ovog rada bio je pronalazak i implementacija algoritma koji može raditi u stvarnom vremenu, s potencijalnim korištenjem u obradi video zapisa

Zadatak je veoma opsežan te uključuje analizu postojećih algoritama super rezolucije, odabir najprikladnijeg za rad u stvarnom vremenu te njegovu implementaciju. Slike korištene za ispitivanje algoritma nastale su digitalnim fotoaparatom pa se faza analize i odabira algoritma pokazala ključnom jer je bilo potrebno sastaviti vlastiti algoritam super rezolucije za konkretnu primjenu. Implementacijska faza uključuje implementaciju odabranog algoritma u višem programskom jeziku MATLAB, te nakon postizanja zadovoljavajućih rezultata i diskusije krajnju implementaciju u jeziku za opis sklopovlja VHDL. Implementacija u jeziku za opis sklopovlja VHDL je zbog opsežnosti ograničena na opis ponašajnog modela.

2. Povećanje izvorne rezolucije slike

2.1. Rezolucija slike

Kad se općenito govori o rezoluciji slike pod tim se pojmom podrazumijeva razina detalja vidljiva na slici. Jedna od mjera rezolucije slike je najmanja udaljenost između dvije linije u stvarnosti pri kojoj ih je moguće razlučiti na slici. Kako bi se shvatio problem povećanja rezolucije važno je razumjeti proces nastanka digitalne slike.

Digitalna slika nastaje prikupljanjem informacija senzorom slike, dvodimenzionalnim poljem fotodioda koje pretvaraju upadnu svjetlost u električne signale. Najmanja jedinica tako nastale slike naziva se piksel pa se digitalna slika sastoji od velikog broja piksela. Kvalitetu informacija prikupljenih senzorom određuju prostorna rezolucija i osvjetljenje. Prostorna rezolucija ovisi o gustoći fotodioda na senzoru i količini zamućenja koje unose fotodiode. Najjednostavnije povećanje prostorne rezolucije na jednakoj promatranoj površini postiže se smanjenjem veličine piksela i time povećanjem njihove gustoće. No što su fotodiode manje to je manja i količina svjetla koja na njih pada te je potrebno duže vrijeme integracije svjetlosnog signala na fotodiodi kako bi se postigao zadovoljavajući omjer signala i šuma. Ako nema pomaka između kamere i snimane scene tada se smanjenje osvjetljenja može nadoknaditi povećanjem vremena ekspozicije odnosno vremena integracije svjetlosti na fotodiodama. U stvarnosti se za vrijeme snimanja kamera trese ili se objekti u sceni pomiču za vrijeme ekspozicije. U tom slučaju vrijeme ekspozicije je zahvatilo nekoliko različitih „kadrova“ što dovodi do zamućenja i ponovnog smanjenja prostorne rezolucije. Rješenja poput povećanja veličine senzora kako bi na njega stalo više fotodioda ili korištenja fotodioda koje su dovoljno male i daju dovoljno velik omjer signala i šuma često nisu prihvatljiva zbog velikih troškova ili tehnoloških ograničenja. Optimalna veličina jednog piksela ima donju granicu i trenutno korištena tehnologija je došla veoma blizu te granice, stoga je razumno poslužiti se nekim drugim tehnikama za povećanje rezolucije slike.

2.2. Super rezolucija

Postoje različite metode kojima se pokušava povećati prostorna rezolucija digitalne slike. Neke od tih metoda koriste napredne tehnike uklanjanja zamućenja i statističke analize kako bi povećale rezoluciju jedne zadane slike, no najuspješnije su metode koje za povećanje rezolucije koriste više uzastopnih slika iste scene. To znači da se na temelju niza slika niske rezolucije sastavlja jedna slika visoke rezolucije. Te tehnike obuhvaćene su jedinstvenim nazivom super rezolucija.

Korištenje više uzastopnih slika iste scene pri povećanju rezolucije slike temelji se na činjenici da je svaka od uzastopnih slika pomaknuta u odnosu na ostale te stoga sadrži drugačiju informaciju o sceni od ostalih slika. Dodatne informacije moguće je iskoristiti pri stvaranju slike veće rezolucije. Kako bi dvije slike iste scene sadržavale različite informacije o toj sceni nužno je da one budu međusobno pomaknute za iznos manji od veličine jednog piksela (pod-pikselni pomak). U tom slučaju pikseli na slikama nisu bili jednako osvijetljeni pa to stvara nove i različite informacije. Ako pomak između dviju slika iznosi cijeli broj piksela, tada su pikseli na slikama bili jednako osvijetljeni pa nema novih informacija po pikselima, već oni sadrže samo pomak scene. Ako pomak iznosi nekoliko piksela tada postupak iskorištavanja novih informacija pri stvaranju slike veće rezolucije postaje iznimno kompliciran te se dodatne informacije ne mogu kvalitetno iskoristiti.

Pomaci između uzastopnih slika pri snimanju mogu biti izazvani namjerno ili se dogoditi slučajno. Kad su pomaci izazvani namjerno, u kontroliranim uvjetima, tada se točno zna koliko oni iznose, odnosno točno se zna koja slika je snimljena s kojim pomakom. U tom slučaju se inzistira da svaka slika ima drukčiji pomak jer se na taj način prikupi najviše novih informacija te je moguće kvalitetnije povećati rezoluciju slike. Izvedba ove metode uključuje postavljanje kamere na platformu koja kontrolirano vibrira. Kod slučajnih pomaka potrebno je na neki način procijeniti koliko iznose pomaci pojedinih slika i to je vrlo važan korak. Bez točnih procjena slučajnih pomaka nemoguće je iskoristiti dodatne informacije pohranjene u nizu uzastopnih slika. Slučajni pomaci statistički pokrivaju većinu pomaka u svim smjerovima, stoga je količina prikupljenih informacija za povećanje rezolucije izravno ovisna o točnosti metode kojom se procjenjuju slučajni pomaci.

2.3. Primjena super rezolucije

Super rezolucija je primjenjiva u mnogo područja, kao što su medicina, satelitske snimke, nadzorne snimke, forenzika i slično.

U medicini je veoma važno koristiti što točnije i detaljnije slike te je super rezolucija od velike koristi kad korišteni optički sustav ne pruža zadovoljavajuću rezoluciju. Kod procesa snimanja u medicini često je lako generirati kontrolirane pomake između snimaka. To omogućava maksimalno iskorištavanje informacije kod metode super rezolucije jer su pomaci unaprijed poznati te su pokriveni svi kutevi pogleda na scenu s pod-pikselnim pomacima..

Super rezolucija je dobar izbor kad su potrebne satelitske snimke veće rezolucije nego što ih pružaju teleskopi korišteni na satelitu. Koristi se često u vojne svrhe radi pregleda terena ili traženja meta, no koristi se i u komercijalne svrhe. Ovdje je također moguće maksimalno iskoristiti metodu super rezolucije jer se pod-pikselni pomaci između snimaka mogu dobiti pomicanjem teleskopa.

Nadzorne kamere često snimaju incidente u kojima je važno identificirati osobe, registarske tablice ili bilo kakve oblike koji mogu dovesti do sudionika incidenta. Visoka rezolucija ne koristi se često kod nadzornih kamera zbog visoke cijene senzora i memorije za dugoročno snimanje, stoga metoda super rezolucije iz niza snimaka može dati sliku potrebne visoke rezolucije. Isti problem javlja se u forenzici gdje detalji koji se traže za rješavanje slučaja često nisu bili od važnosti tijekom nastanka slike ili snimke, pa nije korištena odgovarajuća oprema. Ovdje je važno spomenuti korištenje super rezolucije kod video zapisa. Osim uzastopnih slika s međusobnim pod-pikselnim pomacima, metoda super rezolucije može raditi i s kadrovima video zapisa jer oni također sadrže pomake između kadrova. Za to je potrebna brza implementacija super rezolucije koja može obrađivati kadrove video zapisa u realnom vremenu.

2.4. Koraci super rezolucije

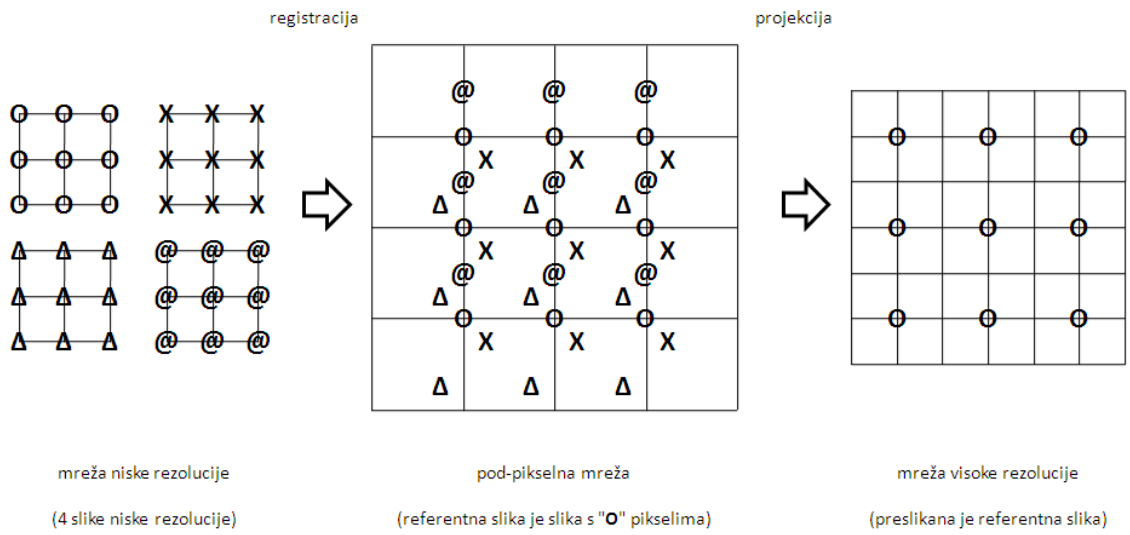
Postoji više algoritama za postizanje super rezolucije, a ono što im je zajedničko je da na temelju niza uzastopnih slika niske rezolucije, koje predstavljaju istu scenu i za svaku sliku u nizu su poznati pod-pikselni pomaci u odnosu na referentnu sliku,

sastavljaju jednu sliku visoke rezolucije. Metodu super rezolucije moguće je rastaviti na dva koraka:

1. Registracija slika niske rezolucije
2. Rekonstrukcija slike visoke rezolucije

Registracija slika niske rezolucije iznimno je važan korak za metodu super rezolucije tokom kojeg se određuju međusobni pomaci slika niske rezolucije u odnosu na referentnu sliku, nakon čega se slike međusobno preklapaju. Kao referentna slika obično se bira neka od slika iz promatranog niza. Međusobni pomaci se tada izražavaju u odnosu na referentnu sliku. Posao iznimno olakšan ako su međusobni pomaci slika niske rezolucije poznati. Ako međusobni pomaci nisu poznati potrebno je primijeniti metodu procjene odnosno detekcije međusobnih pomaka. Točnost konačne slike visoke rezolucije izravno ovisi o poznavanju međusobnih pomaka, stoga u slučaju kada oni nisu poznati izravno ovisi o točnosti i pouzdanosti metode procjene međusobnih pomaka. Iz tog je razloga kod slučajnih pomaka iznimno važno odabrati kvalitetnu metodu za procjenu pomaka. Ovdje je također važno napomenuti da međusobni pomaci slika niske rezolucije ne moraju nužno biti samo translatorni u smjeru horizontalne i vertikalne osi, već može doći i do rotacije te promjene udaljenosti ili perspektive između scene i kamere. Svi ti parametri moraju se uzeti u obzir prilikom odabira metode za procjenu pomaka. Nakon što su poznati svi pomaci pristupa se samom postupku registracije slika niske rezolucije.

Postupak registracije slika niske rezolucije s poznatim međusobnim pomacima prikazan je na slici 2-1.



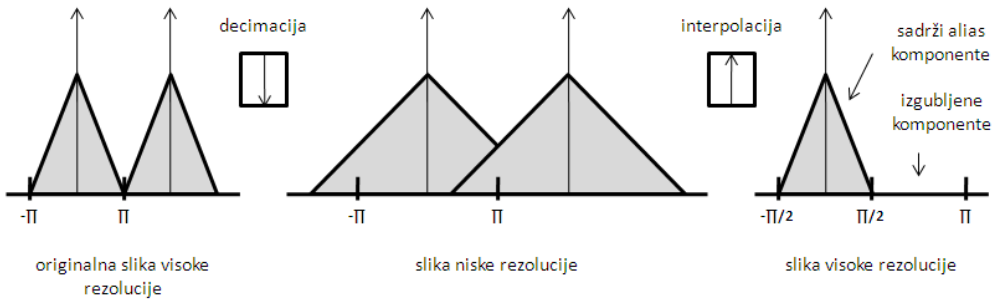
Slika 2-1: Registracija slika niske rezolucije

Slika niske rezolucije je mreža pravilno raspoređenih piksela. Pošto su međusobni pomaci pod-pikselni, registracijom slika niske rezolucije, odnosno njihovim preklapanjem, nastaje nepravilno raspoređena pod-pikselna mreža. Pikseli pomaknutih slika nalaze se na lokacijama koje nisu cjelobrojni višekratnici lokacija piksela referentne slike.

Kako je slika visoke rezolucije također mreža pravilno raspoređenih piksela, postavljanje slika niske rezolucije na mrežu visoke rezolucije izvodi se postavljanjem piksela niske rezolucije na pozicije piksela visoke rezolucije s razmakom koji je jednak faktoru povećanja. Problem nastaje kad pod-pikselni pomaci u niskoj rezoluciji ne mogu biti jednostavno preslikani na mrežu visoke rezolucije, odnosno na lokacije piksela na mreži visoke rezolucije. Na gornjoj slici je mreža visoke rezolucije dvostruko gušća od mreže niske rezolucije, što znači dvostruko povećanje slike. Referentna slika sastavljena je od „O“ piksela. Vidljivo je da pod-pikselni pomak slike sastavljene od „@“ piksela iznosi 0 piksela po horizontalnoj osi te -0,5 piksela po vertikalnoj osi. Pošto je mreža visoke rezolucije dvostruko gušća, „@“ piksele je moguće preslikati na mrežu visoke rezolucije, postavljanjem između dva „O“ piksela na vertikalnoj osi. Pomak „X“ piksela u odnosu na referentne piksele iznosi 0,25 piksela po obje osi pa ih nije moguće izravno preslikati na mrežu visoke rezolucije. Svaki piksel slike niske rezolucije doprinosi stvaranju slike visoke rezolucije, stoga je potrebno odrediti način na koji će se izvesti

projekcija svih piksela registriranih slika niske rezolucije na piksele visoke rezolucije. Tim problemom bavi se sljedeći korak metode super rezolucije.

Rekonstrukcija slike visoke rezolucije korak je u kojem se iz pod-pikselne mreže nastale registriranjem slika niske rezolucije izvlače dodatne informacije te se svakom pikselu visoke rezolucije pokušava dodijeliti najtočnija moguća vrijednost. Prostorna je rezolucija u direktnoj vezi s prostornom frekvencijom. Cilj ovog koraka je procjena izgubljenih odnosno nepostojećih komponenti visoke frekvencije koje su zaslužne za detalje u visokoj rezoluciji. Objašnjenje kako se mijenja spektar nakon decimacije (smanjenja slike) i interpolacije (povećanja slike) ilustrirano je na slici 2-2.



Slika 2-2: Promjene u spektru nakon decimacije i interpolacije

Počnimo od slike visoke rezolucije koja ima svoj spektar. Smanjenje dimenzija slike, a samim time i rezolucije, nastaje decimacijom, odnosno otipkavanjem slike nižom frekvencijom od one kojom je trenutno otipkana. Decimacijom dolazi do širenja spektra signala koji je decimiran. Ako se decimacija provede bez prethodnog filtriranja niskopropusnim filtrom doći će do pojave aliasinga, odnosno preklapanja spektra. Ako se takva slika želi povećati, odnosno ako se želi vratiti početnoj slici visoke rezolucije, potrebno je napraviti interpolaciju. Interpolacija je otipkavanje slike višom frekvencijom od one kojom je trenutno otipkana. Interpolacijom dolazi do sužavanja spektra signala koji je interpoliran. Sužavanjem spektra interpoliranog signala nastaje novi signal čije visoke frekvencije nisu poznate. Iz tog razloga nastala slika uvećanih dimenzija nema očekivanu veću rezoluciju. No visoke frekvencije originalne slike prije decimacije sadržane su u postojećim nižim frekvencijama kao alias komponente, pa je moguće doći do njih, iako taj posao nije sasvim trivijalan. Ono što ipak otežava ovaj

korak je činjenica kako u stvarnosti većina kamera sadrži anti-aliasing filtar koji uklanja visoke frekvencije prilikom nastajanja slike. Samim time slike čiju rezoluciju u stvarnosti pokušavamo povećati su slike niske rezolucije koje ne sadrže alias komponente viših frekvencija ili i sadrže u iznimno maloj količini. Visoke frekvencije zaslužne su za detalje vidljive u visokoj rezoluciji, stoga njihov gubitak znači da uvećana slika zapravo neće imati višu rezoluciju od umanjene slike. Kako bi se postigla visoka rezolucija potrebno je odrediti način na koji će se procijeniti izgubljene visoke frekvencije.

3. Pregled postojećih metoda

3.1. Metode registracije slika

Metode registracije slika koriste se u slučajevima kada više slika prikazuje istu scenu ili sadrže iste dijelove neke scene. Tada se registracijom slika pronalaze međusobni pomaci slika te se zajednički dijelovi međusobno povezuju i preklapaju. Rezultat je veća slika u slučaju panoramskih snimaka ili kvalitetnija slika naknadnom obradom kao što je super rezolucija. Metode registracije slika mogu se podijeliti na metode u prostornoj i metode u frekvencijskoj domeni. U nastavku slijedi kratki pregled najvažnijih postojećih metoda.

U prostornoj domeni se registracija slika često vrši traženjem specifičnih točaka slike koje odudaraju svojim intenzitetom od ostalih, te praćenjem promjene lokacija tih točaka na slikama koje se žele registrirati. Ovakav pristup zahtjeva a priori znanje o slikama i njihovim specifičnim točkama jer se samo objektivno gledano lako mogu pomiješati slične točke. Zbog toga ovakav pristup nije baš prikladan za korištenje kod super rezolucije u stvarnom vremenu. Nakon određivanja točaka koje se promatraju za detekciju njihovih pomaka u odnosu na referentnu sliku koriste se matematičke metode poput korelacije.

Pristup koji ne zahtjeva a priori znanje o slikama fokusira se na pronalaženje pomaka koji smanjuje grešku između dviju promatranih slika. Ovakav pristup pokazao se kao najbolji i najrobusniji te ga stoga koristi većina algoritama super rezolucije. Najčešće korištena verzija potječe iz znanstvenog članka koji su objavili Irani i Peleg [1]. Ona uzima u obzir horizontalni pomak *a*, vertikalni pomak *b* te rotaciju slike za kut θ. Ako je g_1 referentna slika, a g_2 slika čiji pomak želimo pronaći, problem možemo formulirati na sljedeći način:

$$g_2(x,y) = g_1(x\cos\theta - y\sin\theta + a, y\cos\theta + x\sin\theta + b).$$

Razvoj $sin\theta$ i $cos\theta$ u Taylorov red i uzimanje prva dva člana daje:

$$g_2(x,y) \approx g_1\left(x + a - y\theta - \frac{x\theta^2}{2}, y + b + x\theta - \frac{y\theta^2}{2}\right).$$

Razvoj g_1 u Taylorov red i uzimanje prvog člana daje:

$$g_2(x,y) \approx g_1(x,y) + \left(a - y\theta - \frac{x\theta^2}{2}\right)\frac{\partial g_1}{\partial x} + \left(b + x\theta - \frac{y\theta^2}{2}\right)\frac{\partial g_1}{\partial y}.$$

Greška između g_1 i g_2 nakon pomaka za a i b te rotacije za kut θ iznosi:

$$E(a,b,\theta) = \sum\left[g_1\left(x + a - y\theta - \frac{x\theta^2}{2}, y + b + x\theta - \frac{y\theta^2}{2}\right) - g_2(x,y)\right]^2,$$

gdje sumacija ide preko preklapajućih dijelova g_1 i g_2. Minimum funkcije $E(a,b,\theta)$ moguće je dobiti deriviranjem po a, b i θ te izjednačavanjem rezultata s nulom. To nas dovodi do konačnih jednadžbi:

$$\sum g_x^2 a + \sum g_x g_y b + \sum A g_x \theta = g_x g_t,$$

$$\sum g_y^2 b + \sum g_x g_y a + \sum A g_y \theta = g_y g_t,$$

$$\sum A g_x a + \sum A g_y b + \sum A^2 \theta = A g_t,$$

gdje je $g_x = \frac{\partial g_1}{\partial x}$, $g_y = \frac{\partial g_1}{\partial y}$, $g_t = g_2 - g_1$ i $A = x g_y - y g_x$. Rješavanjem ovih jednadžbi dobit ćemo horizontalni pomak a, vertikalni pomak b te kut rotacije θ. Ove jednadžbe vrijede pod pretpostavkom da su pomaci vrlo mali. U slučaju većih pomaka koristimo iterativni pristup. To znači da nakon izračuna translatornih pomaka i kuta rotacije sliku g_2 pomičemo sukladno izračunatim parametrima te ponavljamo postupak i računamo nove translatorne pomake i kut rotacije. Iterativni proces ponavljamo dok se translatorni pomaci i kut rotacije ne približe nuli, a konačne parametre pomaka dobijemo zbrajanjem rezultata u svakom koraku iteracije.

U frekvencijskoj domeni se za registraciju slika koristi svojstvo pomaka Fourierove transformacije [8, 9]. Za referentnu sliku I_1 i pomaknutu sliku I_2 vrijedi jednadžba:

$$I_2(\vec{x}) = I_1(\vec{x} + \vec{a}).$$

Svojstvo pomaka Fourierove transformacije kaže da u frekvencijskoj domeni ta jednadžba izgleda kao:

$$F_2(\vec{w}) = F_1(\vec{w}) e^{i\vec{w}\vec{a}}.$$

Kompleksni broj $e^{i\vec{w}\vec{a}}$ ima apsolutnu vrijednost jednaku 1, stoga je potrebno proći kroz sve frekvencije $\vec{w}$ i pronaći rezultat aproksimacijske pogreške $\left|F_2(\vec{w}) - e^{i\vec{w}\vec{a}} F_1(\vec{w})\right|^2$. Tražimo frekvenciju na kojoj je aproksimacijska pogreška jednaka nuli. Koristeći Lagrangeov multiplikator rješenje ovog problema je:

$$e^{i\vec{w}\vec{a}} = \frac{F_1(\vec{w})F_2(\vec{w})^*}{|F_1(\vec{w})F_2(\vec{w})^*|}.$$

U idealnom slučaju $e^{i\vec{w}\vec{a}}$ je sinusoida, a njena inverzna Fourierova transformacija je $\delta(\vec{x} - \vec{a})$, što je funkcija koja je jednaka nuli svugdje osim u točki $\vec{x} = \vec{a}$. Algoritam za pronalaženje vektora pomaka $\vec{a}$ je dakle:

1. Izračunaj Fourierovu transformaciju slika I1 i I2.
2. Izračunaj za svaku frekvenciju omjer $e^{i\vec{w}\vec{a}} = \frac{F_1(\vec{w})F_2(\vec{w})^*}{|F_1(\vec{w})F_2(\vec{w})^*|}$ kao funkciju $r(\vec{w})$.
3. Izračunaj inverznu Fourierovu transformaciju funkcije $r(\vec{w})$.
4. Nađi vektor pomaka $\vec{a}$ kao točku u kojoj funkcija $r(\vec{w})$ ima maksimum.

Prezentirana metoda računa samo translatorne pomake, no u literaturi postoje i složenije verzije koje prelaskom na polarne koordinate također računaju i rotaciju slika.

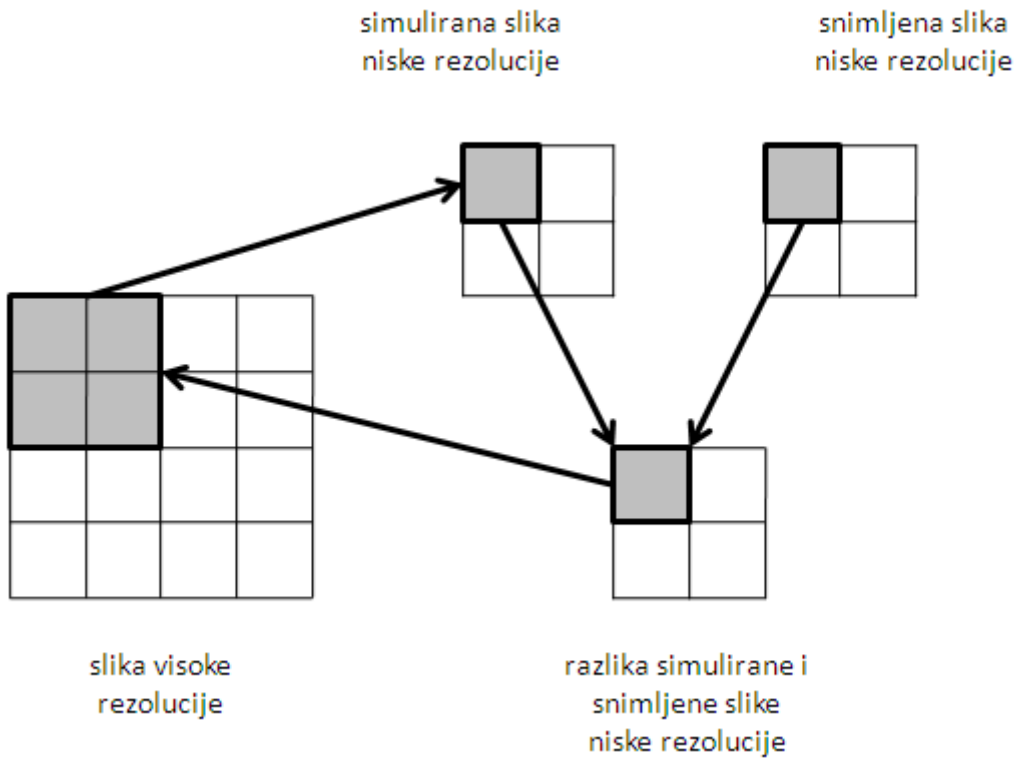
3.2. Metode rekonstrukcije slike visoke rezolucije

Rekonstrukcija slike visoke rezolucije iz niza slika niske rezolucije je problem s više rješenja. Kao što je ranije navedeno, ovaj korak metode super rezolucije pokušava procijeniti izgubljene visoke frekvencije u slici visoke rezolucije nastaloj nakon registracije slika niske rezolucije. Postojeće metode pokušavaju riješiti ovaj problem u prostornoj ili frekvencijskoj domeni. U nastavku slijedi kratki pregled najvažnijih postojećih metoda, s fokusom na one koje su utjecale na ovaj rad.

U prostornoj domeni najjednostavnija i najintuitivnija metoda rekonstrukcije slike visoke rezolucije nakon registracije niza slika niske rezolucije je interpolacija. Zbog nejednolikog rasporeda piksela na mreži visoke rezolucije nakon registracije korištena interpolacija je također nejednolika. To znači da se prilikom interpolacije uzima u obzir poznata vrijednost piksela nastalih registracijom slika niske rezolucije. Nakon nejednolike interpolacije slika je zamućena te je potrebno ukloniti to zamućenje. Kao najčešće i najkvalitetnije rješenje tog problema koristi se Wienerov filtar.

Popularna metoda je i projekcija na konveksne skupove. Ona koristi a priori znanje kako bi rješenje ograničila na zatvoreni konveksan skup rješenja. Istovremeno vrši interpolaciju i rekonstrukciju slike, a računalno je vrlo zahtjevna. Potreba za a priori znanjem i računalna zahtjevnost čine ju lošim izborom za rad u stvarnom vremenu.

Većina uspješno korištenih metoda u prostornoj domeni temelji se na iterativnoj projekciji unazad, koja potječe iz znanstvenog članka koji su objavili Irani i Peleg [1]. Ta metoda počinje s inicijalnom aproksimacijom slike visoke rezolucije, najčešće interpolacijom. Zatim se simulira proces nastanka slike pa od aproksimacije slike visoke rezolucije nastaje simulirana slika niske rezolucije. Ona se uspoređuje sa stvarnom snimljenom slikom niske rezolucije jer bi te dvije slike morale biti identične ako je aproksimacija slike visoke rezolucije točna. Na sliku visoke rezolucije vrši se „projekcija unazad“ razlike između simulirane i snimljene slike niske rezolucije jer ona zapravo predstavlja grešku u aproksimaciji slike visoke rezolucije. Proces se ponavlja iterativno sa svim snimljenim slikama niske rezolucije dok greška aproksimacije ne postane zadovoljavajuće mala. Algoritam je ilustriran na slici 3-1.



Slika 3-1: Ilustracija algoritma iterativne projekcije unazad

Kod ove metode proces nastanka snimljene slike modeliran je jednadžbom:

$$g_k(m,n) = \alpha_k \left(h\left(T_k\left(f(x,y)\right)\right) + \mu_k(x,y) \right),$$

gdje

- g_k predstavlja *k*-tu snimljenu sliku,
- f predstavlja sliku visoke rezolucije koju algoritam pokušava pronaći,
- T_k je 2D transformacija koja preslikava f u g_k,
- h je funkcija zamućenja koja ovisi o funkciji razmazivanja (*eng*. Point Spread Function) senzora kamere,
- μ_k je dodani šum,
- α_k je operator decimacije.

Algoritam stvara inicijalnu aproksimaciju slike visoke rezolucije $f^{(0)}$ te stvara niz simuliranih slika niske rezolucije $\left\{g_k^{(0)}\right\}_{k=1}^{K}$ koje odgovaraju nizu snimljenih slika niske rezolucije $\{g_k\}_{k=1}^{K}$. Proces stvaranja simuliranih slika niske rezolucije modeliran je jednadžbom:

$$g_k^{(n)} = T_k\left(f^{(n)}\right) * h,$$

gdje

- n predstavlja *n*-tu iteracija,
- $*$ je operator konvolucije.

Računa se razlika između simulirane i snimljene slike niske rezolucije kao $g_k - g_k^{(n)}$ te se vrši „projekcija unazad“ na sliku visoke rezolucije. Konačna iterativna jednadžba koja modelira sliku visoke rezolucije može se napisati kao:

$$f^{(n+1)} = f^{(n)} + \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}\left(g_k - g_k^{(n)}\right) * p,$$

gdje

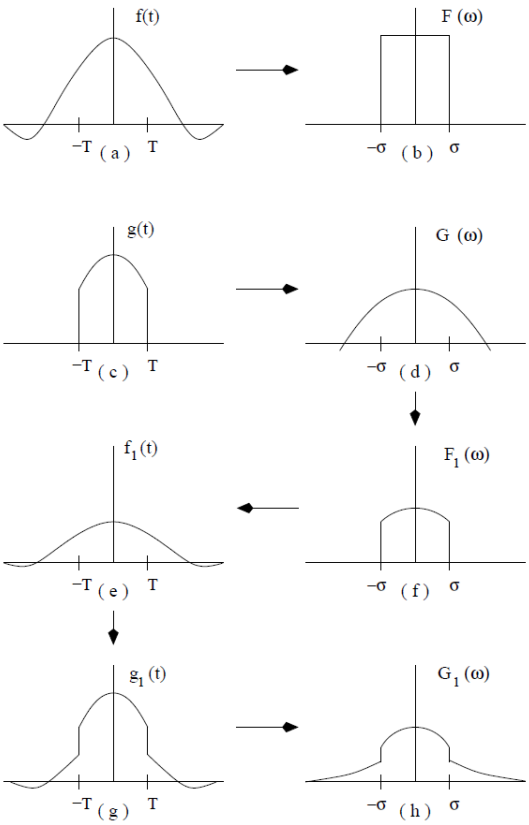
- *K* predstavlja broj slika niske rezolucije,
- T_k je 2D transformacija koja preslikava g_k u f,
- *p* je jezgra projekcije unazad, povezana je s funkcijom zamućenja *h*.

Ova metoda je jedna od temeljnih i najčešće korištenih u rekonstrukciji slike kod super rezolucije. Irani i Peleg svojim su radovima iznimno utjecali na metodu super rezolucije. Njihovi su radovi iz tog područja redovito citirani i predstavljaju temelj kod većine ostalih radova koji se bave super rezolucijom.

Kod rekonstrukcije slike visoke rezolucije u frekvencijskoj domeni dominantna je Papoulis-Gerchberg metoda [4, 5]. To je iterativna metoda koja se temelji na frekvencijskim karakteristikama procesa interpolacije i decimacije, odnosno promjenama koje se događaju u spektru signala za vrijeme tih procesa. Počinje tako da se registracijom slika niske rezolucije na mrežu visoke rezolucije dobije slika visoke rezolucije s nepravilno raspoređenim pikselima. Ne provodi se nikakva interpolacija već se nepoznatim pikselima dodjeljuje vrijednost nula. Takva slika se zapamti, te se izračunava njezina Fourierova transformacija. U dobivenom spektru sve se visoke frekvencije postavljaju na nulu, te se izračunava inverzna Fourierova transformacija. Dobije se nova slika visoke rezolucije koja umjesto nula na inicijalno nepoznatim pikselima ima interpolirane vrijednosti, dok na inicijalno poznatim pikselima ima promijenjene vrijednosti. Vrijednosti na inicijalno poznatim pikselima smanjene su zbog procesa interpolacije koji se dogodio. Sljedeći korak je vraćanje inicijalnih vrijednosti na inicijalno poznate piksele, pošto su to sigurno točne vrijednosti dobivene iz slika niske rezolucije. Time je završen jedan krug ove metode. Idući krug započinje računanjem Fourierove transformacije trenutne slike visoke rezolucije. Proces se iterativno ponavlja dok se ne postigne zadovoljavajući rezultat. Iterativni algoritam na slici visoke rezolucije stoga glasi:

1. Pikselima s inicijalno poznatim vrijednostima pridružiti te vrijednosti
2. Izračunati Fourierovu transformaciju
3. Visoke frekvencije postaviti u nulu
4. Izračunati inverznu Fourierovu transformaciju
5. Povratak na korak 1.

Ono što se zapravo događa je da Papoulis-Gerchberg metoda pokušava interpolirati nepoznate vrijednosti i time popraviti aliasing u niskim frekvencijama, dok konstantno vraćanje vrijednosti poznatim pikselima generira komponente visokih frekvencija na temelju informacija sadržanih u ispravno registriranim slikama niske rezolucije. Ilustracija Papoulis-Gerchberg metode nalazi se na slici 3-2.



Slika 3-2: Ilustracija Papoulis-Gerchberg metode (preuzeto iz [4])

Na slici 3-2 vidimo signal koji želimo rekonstruirati pod (a) te njegov spektar pod (b). Signal koji nam je dostupan je pojasno ograničen jer mu nedostaju visoke frekvencije i nalazi se pod (c). Njegov spektar koji se nalazi pod (d) dobijemo Fourierovom transformacijom, a sadrži niske i neke netočne visoke frekvencije. Pojasnim ograničavanjem tog spektra postavljanjem visokih frekvencija u nulu dobivamo spektar pod (f). Inverznom Fourierovom transformacijom dobivamo interpolirani signal koji se nalazi pod (e). Vidljivo je da je postignut oblika signala pod (a) koji želimo rekonstruirati. Vraćanjem inicijalno poznatih vrijednosti pikselima dobivamo signal pod (g), čiji spektar ponovno sadrži visoke frekvencije, kao što je vidljivo pod (h). No također je vidljivo da signal pod (e) i njegov spektar pod (f) imaju oblik signala koji želimo rekonstruirati. Iteracijom ovog postupka taj oblik će postati sve sličniji željenom signalu, do neke mjere određene šumom u procesu snimanja te točnosti korištene metode registracije slika niske rezolucije.

4. Korišteni algoritam za povećanje izvorne rezolucije slike

4.1. Nastanak algoritma

Algoritam koji sam koristio razvio sam kao kombinaciju postojećih metoda. Kriteriji pri odabiru korištenih metoda bili su računalna jednostavnost i mogućnost brze implementacije na FPGA sklopu te pouzdanost u radu s različitim tipovima slika i kamera. Nakon proučavanja literature smatrao sam da bi za rekonstrukciju slike visoke rezolucije dobre rezultate mogao dobiti korištenjem metode koju su predložili Irani i Peleg jer je najčešće korištena i analizirana. Iako je računalno jednostavna zahtjeva određeno vrijeme da se izvrši te sam bio skeptičan oko zadovoljavanja vremenskih zahtjeva u stvarnom vremenu. Za registraciju slika niske rezolucije nisam bio siguran hoće li bolje rezultate davati metoda koju su predložili Irani i Peleg ili metoda u frekvencijskoj domeni, stoga sam odlučio implementirati obje i vidjeti koja zaista daje bolje rezultate. Implementirao sam spomenute metode u MATLAB-u te pristupio analizi dobivenih rezultata.

Algoritam sam inicijalno ispitivao na simuliranim slikama dobivenim smanjivanjem originalnih slika visoke rezolucije. Koristio sam ugrađene MATLAB funkcije *imresize* i *interp2* kako bi decimirao originalnu sliku visoke rezolucije. Pod-pikselne pomake sam s funkcijom *imresize* izvodio korištenjem funkcije *circshift* prije decimacije tako da sam piksele slike visoke rezolucije pomicao horizontalno i vertikalno za broj piksela manji od faktora smanjenja, čime se nakon decimacije dobiju slike niske rezolucije koje međusobno imaju pod-pikselni pomak. Kod funkcije *interp2* moguće je birati realne točke u kojima će se cjelobrojna mreža otipkati, pa se slike niske rezolucije s pod-pikselnim pomacima izvode otipkavanjem nižom frekvencijom u realnim točkama umjesto u cjelobrojnim točkama. Svi pomaci bili su slučajno generirani. Slike je također bilo potrebno propustiti kroz nisko propusni anti-alias filtar te unijeti određeno zamućenje kao posljedicu senzora kamere. Nisam bio zadovoljan generiranjem slika za ispitivanje na ovakav način jer sam tijekom ispitivanja shvatio da pretpostavljam poznavanje karakteristika senzora kamere i modeliram ih na relativno jednostavan

način. Bila je upitna sličnost karakteristika ovako generiranih slika i slika koje su zaista snimljene kamerom. Pošto sam želio da moj algoritam bude primjenjiv u stvarnosti odlučio sam ga ispitivati na realnim slikama dobivenim digitalnim fotoaparatom.

Korišteni digitalni fotoaparat je Sony DSC-W12 u Multi Burst načinu snimanja. U tom načinu snimanja pritiskom na tipku za okidanje automatski se snimi 16 uzastopnih slika. Vremenski razmak između dva okidanja je 1/30 sekundi. Veličina nastalih slika u pikselima iznosi 320 x 240 piksela. Radi jednostavnosti sam koristio crno-bijele slike s 8 bitova po pikselu, odnosno svaki piksel je u memoriji bio predstavljen jednim bajtom, kao cijeli broj bez predznaka (*unsigned integer*). Nakon detaljne analize nekoliko nizova ispitnih slika shvatio sam da za korišteni algoritam mogu pretpostaviti neke činjenice. Naime, pokazalo se da mirno držanje kamere u ruci tijekom snimanja uzastopnih slika zapravo i nije toliko mirno. Prirodno dolazi do drhtanja ruke što uzrokuje slučajne pomake koji su dovoljno mali i statistički dobro pokrivaju snimanu scenu, stoga je takav niz slika moguće iskoristiti u metodi super rezolucije. Također, pomaci koji nastaju „mirnim“ držanjem kamere u kratkom vremenu translatorne su prirode, odnosno dolazi do pomaka u horizontalnom i vertikalnom smjeru pa je moguće zanemariti pojavu rotacije pri registraciji slika.

4.2. Registracija slika

Zanemarivanje pojave međusobne rotacije među slikama niske rezolucije iznimno je pojednostavilo metode za registraciju slika. Metodu u frekvencijskoj domeni prezentiranu u poglavlju 3.1. bilo je moguće izravno implementirati jer nije uzimala u obzir rotaciju slika. Metodu koju su predložili Irani i Peleg također prezentiranu u poglavlju 3.1. bilo je potrebno pojednostaviti, odnosno modificirati kako bi se maknuo utjecaj rotacije, po uzoru na [2]. Irani i Peleg metoda računa horizontalni pomak *a*, vertikalni pomak *b* te kut rotacije θ kao rješenje jednadžbi:

$$\sum g_x^2 a + \sum g_x g_y b + \sum A g_x \theta = g_x g_t,$$

$$\sum g_y^2 b + \sum g_x g_y a + \sum A g_y \theta = g_y g_t,$$

$$\sum A g_x a + \sum A g_y b + \sum A^2 \theta = A g_t,$$

gdje je $g_x = \frac{\partial g_1}{\partial x}$, $g_y = \frac{\partial g_1}{\partial y}$, $g_t = g_2 - g_1$ i $A = x g_y - y g_x$. Ako za kut rotacije θ pretpostavimo da je jednak nuli, tada su nam preostaju samo dvije nepoznanice *a* i *b*.

Sad su nam od ponuđene tri jednadžbe dovoljne samo dvije. Ako odaberemo prve dvije jednadžbe zbog svoje sličnosti i jednostavnosti, dobili smo sustav jednadžbi:

$$\sum g_x^2 a + \sum g_x g_y b = g_x g_t,$$

$$\sum g_y^2 b + \sum g_x g_y a = g_y g_t.$$

To je zapravo linearni sustav koji se može u matričnom obliku zapisati kao:

$$\begin{bmatrix} \sum g_x^2 & \sum g_x g_y \\ \sum g_x g_y & \sum g_y^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g_x g_t \\ g_y g_t \end{bmatrix},$$

odnosno:

$$\begin{bmatrix} \sum \left(\frac{\partial g_1}{\partial x}\right)^2 & \sum \frac{\partial g_1}{\partial x}\frac{\partial g_1}{\partial y} \\ \sum \frac{\partial g_1}{\partial x}\frac{\partial g_1}{\partial y} & \sum \left(\frac{\partial g_1}{\partial y}\right)^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x}(g_2 - g_1) \\ \frac{\partial g_1}{\partial y}(g_2 - g_1) \end{bmatrix}.$$

Ako navedenu matričnu jednadžbu zapišemo kao:

$$M \begin{bmatrix} a \\ b \end{bmatrix} = V,$$

tada horizontalni pomak a i vertikalni pomak b dobijemo kao rješenje matrične jednadžbe:

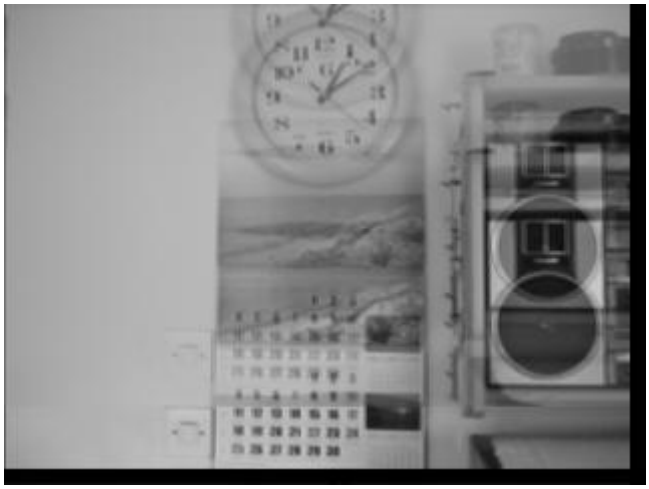
$$\begin{bmatrix} a \\ b \end{bmatrix} = M^{-1} V.$$

Rješenja su realni brojevi čija apsolutna vrijednost u idealnom slučaju ne prelazi 1, a predstavlja pod-pikselni pomak slike niske rezolucije u odnosu na referentnu sliku niske rezolucije. Ova metoda u stvarnosti dobro radi kad su pomaci zaista pod-pikselni ili manji od 1,5 piksela, dok bi za veće pomake bilo potrebno implementirati iterativnu strukturu koja zbraja dobivene pomake i ponavlja se sve dok detektirani pomaci u nekom koraku ne iznose nula. U svom algoritmu nisam implementirao takvu strukturu iz dva razloga. Prvi je vezan za početnu pretpostavku o snimanju slika „mirnim“ držanjem, čime nastaju dovoljno mali pomaci. Drugi razlog je gubitak preciznosti registracije slika kroz iterativni postupak.

Ispitivanje točnosti implementiranih metoda registracije slika izvedeno je vizualnom procjenom, pošto nakon snimanja kamerom ne postoje numerički podaci o stvarnim pomacima između nastalih slika. Osim ručne analize promatranjem ponašanja nekih

značajnih piksela promatrao sam i rezultat registracije slika. Slike su nakon pod-pikselnih pomaka preklopljene jedna preko druge te se promatra poklapaju li se obrisi objekata u sceni na preklopljenim slikama. Poklapanje obrisa objekata, odnosno izostanak pojave obrisa objekata na neočekivanim mjestima znak je da su pomaci uspješno detektirani.

Rezultati preklapanja pomaknutih slika nalaze se na slikama 4-1 i 4-2.



Slika 4-1: Detekcija pomaka u frekvencijskoj domeni

Na slici 4-1 nalaze se rezultati detekcije pomaka u frekvencijskoj domeni. Vidljivo je da su pomaci barem jedne slike u nizu pogrešno detektirani za iznos reda veličine 10 piksela, što veoma narušava izgled slike i utječe na rezultate metode rekonstrukcije slike visoke rezolucije koja slijedi nakon registracije slika. Ovaj problem se pojavio kod svih ispitanih nizova slika. Uvijek bi pomaci jedne ili dviju slika bili pogrešno izračunati, pa je jedno od rješenja tog problema bilo ignoriranje slika čiji pomaci u velikoj mjeri odstupaju od pomaka ostalih slika. Takvo rješenje nije bilo zadovoljavajuće jer se tada ne koriste slike s pogrešno detektiranim pomacima, što je nepotreban gubitak informacije. Metoda je vidljivo osjetljiva na smetnje.

Na slici 4-2 nalaze se rezultati detekcije pomaka Irani i Peleg metodom:

Slika 4-2: Detekcija pomaka Irani i Peleg metodom

Vidljivo je da ovdje nema obrisa objekata na neočekivanim mjestima što znači da su pomaci dobro detektirani. Kod nijednog ispitanog niza slika nije se pojavio problem u detekciji pomaka jer je metoda dovoljno robusna. Iz tog je razloga Irani i Peleg metoda odabrana kao metoda kojom će se registrirati slike niske rezolucije u korištenom algoritmu.

4.3. Rekonstrukcija slike visoke rezolucije

Za rekonstrukciju slike visoke rezolucije odlučio sam koristiti metodu iterativne projekcije unazad koju su predložili Irani i Peleg. Nakon implementacije u MATLAB-u naišao sam na neke probleme. U jednadžbama koje opisuju tu metodu važan faktor igraju funkcija zamućenja i jezgra projekcije unazad. Funkcija zamućenja ovisi o parametrima senzora kamere koji mi nisu bili poznati. Jezgra projekcije unazad ovisna je o funkciji zamućenja. Vodeći se korištenom literaturom pretpostavio sam da funkcija zamućenja ima oblik Gaussove krivulje, no nisam bio zadovoljan rezultatima.

Na slici 4-3 nalazi se jedna od slika niske rezolucije.

Slika 4-3: Jedna od slika niske rezolucije

Na slici 4-4 nalazi se rezultat rekonstrukcije slike 4 puta veće rezolucije od originalne koji nije zadovoljavajuće kvalitete.

Slika 4-4: Rezultat povećanja rezolucije 4 puta iterativnom projekcijom unazad

Vidljivo preklapanje projekcija susjednih piksela, što je posljedica loše procjene funkcije zamućivanja i jezgre projekcije unazad. Unatoč podešavanju tih parametara nisam dobivao bitno bolji rezultat. Problem bi možda bilo moguće riješiti dodatnim filtriranjem no takvo rješenje mi se nije sviđalo iz dva razloga. Prvi razlog je dovoljna složenost samog algoritma, odnosno vrijeme njegovog izvođenja kojim nisam bio sasvim zadovoljan. Smatrao sam da dodavanje novih izračuna u algoritam ne pogoduje njegovoj brzini. Drugi razlog vezan je uz činjenicu da algoritam nije robustan u smislu da je nužno a priori poznavanje parametara senzora kamere. Stoga sam se vratio

literaturi u potrazi za drugim metodama rekonstrukcije slike visoke rezolucije koje bih mogao koristiti.

Metoda koja me zainteresirala je Papoulis-Gerchberg metoda, prezentirana u poglavlju 3.2., koja rekonstrukciju vrši u frekvencijskoj domeni. Imajući u vidu želju za radom algoritma u stvarnom vremenu odlučio sam pretvoriti korake Papoulis-Gerchberg metode koji se odvijaju u frekvencijskoj domeni u korake koji se odvijaju u prostornoj domeni. Na taj korak sam se odlučio iz dva razloga. Prvi je činjenica da se koraci Papoulis-Gerchberg metode u frekvencijskoj domeni mogu u prostornoj domeni prikazati kao jednostavno nisko propusno filtriranje. Drugi razlog je činjenica da je računanje Fourierove i inverzne Fourierove transformacije računalno zahtjevnije od filtriranja pa je razumnije posegnuti za filtriranjem u prostornoj domeni kako bi se postigla veća brzina algoritma. Koraci jedne iteracije Papoulis-Gerchberg metode u frekvencijskoj domeni glase:

1. Pikselima s inicijalno poznatim vrijednostima pridružiti te vrijednosti
2. Izračunati Fourierovu transformaciju
3. Visoke frekvencije postaviti u nulu
4. Izračunati inverznu Fourierovu transformaciju
5. Povratak na korak 1.

Možemo primijetiti da je treći korak zapravo nisko propusno filtriranje. Sukladno tome je navedene korake moguće prebaciti u prostornu domenu kao što slijedi:

1. Pikselima s inicijalno poznatim vrijednostima pridružiti te vrijednosti
2. Izvršiti filtriranje nisko propusnim filtrom
3. Povratak na korak 1.

Algoritam mora završiti s korakom filtracije, a ne vraćanjem inicijalno poznatih vrijednosti jer će u protivnom rezultirajuća slika imati skokove u intenzitetu na lokacijama inicijalno poznatih vrijednosti. Ovu modifikaciju Papoulis-Gerchberg metode implementirao sam u MATLAB-u, a rezultat rekonstrukcije slike 4 puta veće rezolucije od originalne moguće je vidjeti na slici 4-5:

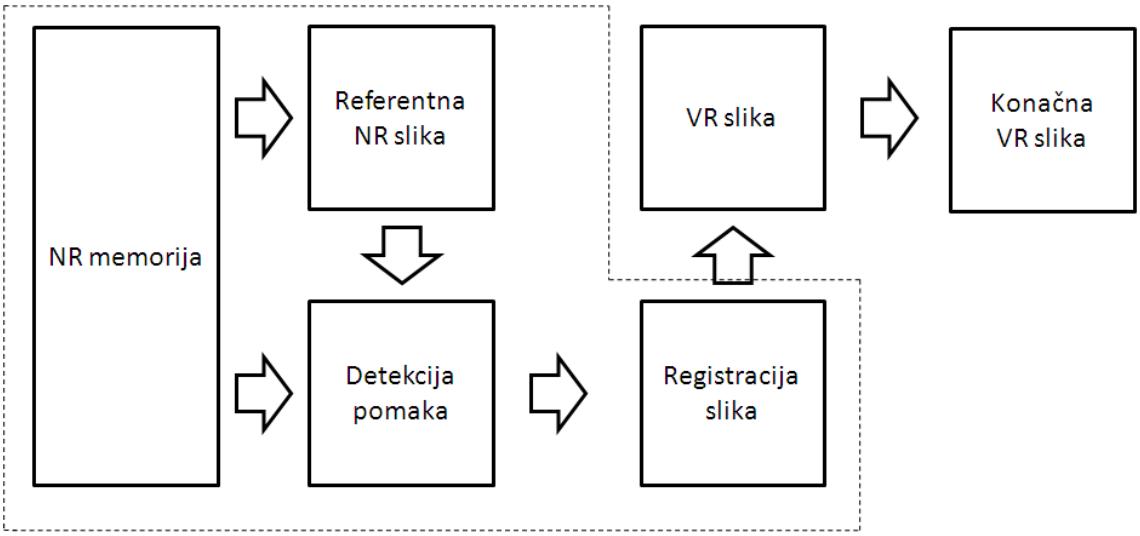
Slika 4-5: Rezultat povećanja rezolucije 4 puta modifikacijom Papoulis-Gerchberg metode

Slika je iznimno kvalitetno uvećana i rezultati su veoma zadovoljavajući. Također, brzina izvođenja algoritma je višestruko poboljšana u odnosu na iterativnu projekciju unazad. Moguće je zaključiti kako je korištena modifikacija Papoulis-Gerchberg metode veoma uspješna u rekonstrukciji slike visoke rezolucije, čak pri faktoru povećanja 4. Također je brza i ne zahtjeva nikakvo a priori znanje o senzoru kamere ili slici. Iz tog je razloga ova modifikacija Papoulis-Gerchberg metode odabrana kao metoda kojom će se rekonstruirati slika visoke rezolucije u korištenom algoritmu.

4.4. Opis algoritma

Nakon što su odabrane ključne metode algoritma za registraciju slika niske rezolucije i rekonstrukciju slike visoke rezolucije potrebno je kreirati čitav algoritam. Algoritam kao ulazne parametre prima niza slika niske rezolucije te faktor povećanja za koji želimo povećati rezoluciju slike, dok na izlazu daje sliku visoke rezolucije.

Blok shema algoritma dana je na slici 4-6:



Slika 4-6: Blok shema algoritma za povećanje izvorne rezolucije slike

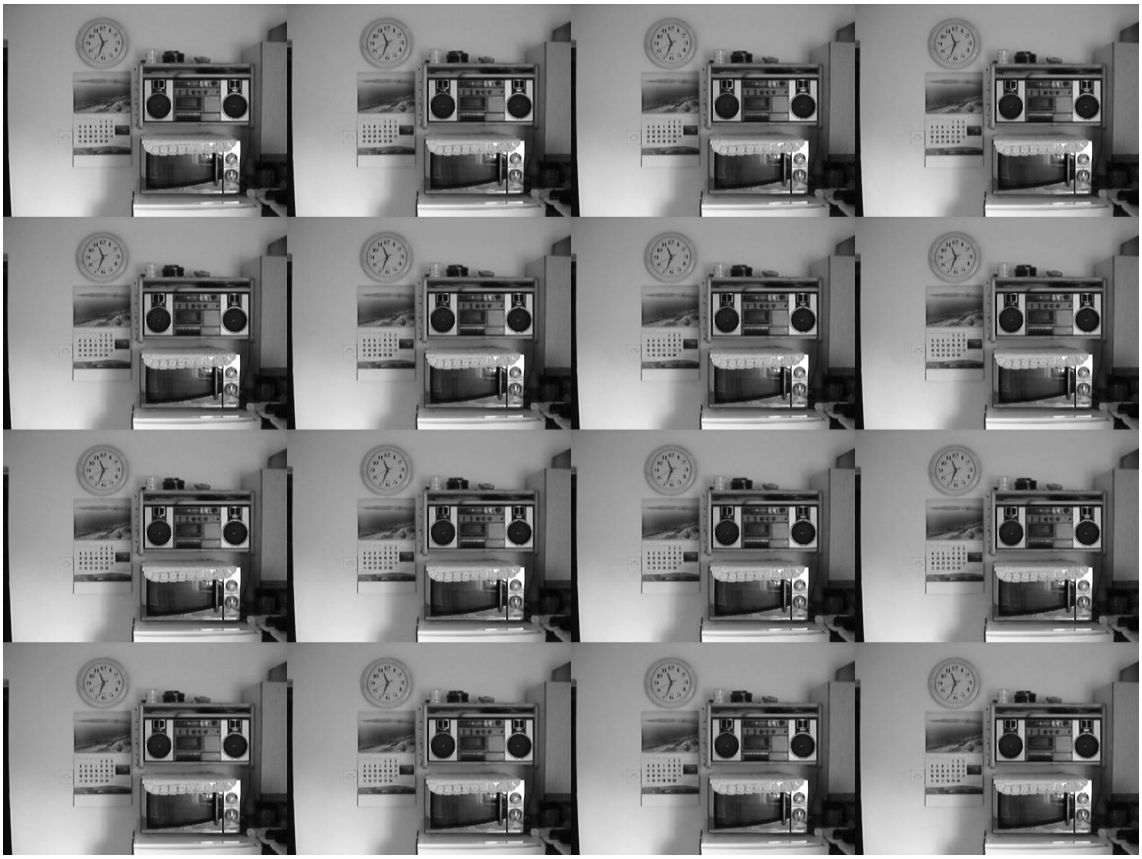
Slike niske rezolucije nalaze se u memorijskom bloku „NR memorija“. Odabrao sam da prva u nizu slika niske rezolucije postane referentna slika niske rezolucije prema kojoj će se računati pomaci ostalih slika. Ona se prosljeđuje u blok „Referentna NR slika“. Kad pristigne iduća slika niske rezolucije, u bloku „Detekcija pomaka“ računa se njezin pomak po horizontalnoj i vertikalnoj osi u odnosu na referentnu sliku. Pomaci se računaju korištenjem spomenute modificirane Irani i Peleg metode. S tim podacima se u bloku „Registracija slika“ ta slika niske rezolucije registrira na mrežu visoke rezolucije. Registrirana se slika kombinira s prijašnjim registriranim vrijednostima, a taj se proces ponavlja za sve slike niske rezolucije na raspolaganju. Kad su sve slike niske rezolucije prošle kroz ovaj ciklus, čime je sastavljena slika visoke rezolucije s nejednoliko raspodijeljenim pikselima, u bloku „VR slika“ rekonstruira se slika visoke rezolucije. Rekonstrukcija se vrši korištenjem spomenute modificirane Papoulis-Gerchberg metode. Po završetku procesa rekonstrukcije slike visoke rezolucije dobili smo konačnu sliku visoke rezolucije.

5. Implementacija algoritma u programskom jeziku MATLAB

U sljedećim poglavljima opisan je način implementacije ključnih dijelova algoritma u programskom jeziku MATLAB. Kreće se od izravne implementacije te se ona zatim pretvara u pojednostavljenu koja je bliža izvedbi na FPGA sklopu. Na kraju su prezentirani i diskutirani rezultati implementacije u vidu slika visoke rezolucije.

5.1. Detalji implementacije

Implementacija algoritma počinje učitavanjem slika niske rezolucije snimljenih kamerom. Kao što je već navedeno, korištenjem digitalnog fotoaparata u Multi Burst načinu snimanja jednim okidanjem nastane 16 slika niske rezolucije, veličine 320 x 240 piksela, koje su međusobno snimljene s vremenskim razmakom od 1/30 sekundi. Slike su zapisane kao jedna slika veličine 1280 x 960 piksela na kojoj su u pravilnoj mreži raspoređene slike niske rezolucije. Primjer je moguće vidjeti na slici 5-1:



Slika 5-1: Niz slika niske rezolucije snimljenih u Multi Burst načinu snimanja. Originalna slika u boji pretvorena je u crno-bijelu sliku s 8 bitova po pikselu.

Slika je izvorno u boji i pohranjena u JPEG formatu. Korištenjem besplatnog programa za obradu slika Paint.NET pretvorena je u crno-bijelu sliku zapisanu s 8 bitova po pikselu (*unsigned integer*) te pohranjena u PNG formatu. Besplatan program za obradu slika Paint.NET moguće je pronaći na internetskoj stranici http://www.getpaint.net/. Korišteni kôdovi priloženi su ovom radu u digitalnom obliku, a u nastavku će biti prokomentirani neki ključni dijelovi.

Za izvlačenje pojedinačnih slika niske rezolucije iz velike slike nastale snimanjem u Multi Burst načinu koristio sam samostalno napisanu funkciju `extractBurst`:

```
inLR = extractBurst(inputImage, width, height),
```

koja u varijablu `inLR` posprema niz od 16 slika niske rezolucije jednu pored druge. Ukupna veličina te varijable je dakle 5120 x 240 piksela (16 puta po 320 piksela u širinu). Za dohvat određenog manjeg segmenta slike koristio sam samostalno napisanu funkciju `getPart`:

```
inLR =
getPart(inLR,widthLR,heightLR,oWidth,oHeight,offX,offY),
```

koja kao parametre prima niz od 16 slika niske rezolucije, veličinu željenog manjeg segmenta slike te njegovu udaljenost od ishodišta slike. U varijablu `inLR` vraća željene segmente iz svih 16 slika poredane jedan uz drugog kao što su početne slike niske rezolucije bile poredane prije pozivanja ove funkcije. Ovakva organizacija slika niske rezolucije omogućava lak pristup željenoj slici niske rezolucije jednostavnim indeksiranjem. Primjerice, referentna slika niske rezolucije posprema se u posebnu varijablu naredbom:

```
reference = inLR(:, 1 : widthLR);
```

Za postavljanje slike niske rezolucije na mrežu visoke rezolucije koristio sam samostalno napisanu funkciju `doHR`:

```
function frameHR = doHR(frameLR, factor)
% Allocate a HR grid.
frameHR = -ones(factor*size(frameLR, 1), factor*size(frameLR, 2));
% Go through LR pixels.
for x = 1 : size(frameLR, 2)
    for y = 1 : size(frameLR, 1)
        % Fill influenced HR pixels with current LR pixel.
        frameHR(y * factor, x * factor) = frameLR(y, x);
    end
end
frameHR = round(frameHR);
end
```

Funkcija postavlja piksele slike niske rezolucije na mrežu visoke rezolucije na lokacije koje su cjelobrojni višekratnici faktora povećanja F. To je iz razloga što, primjerice, povećanje 2 puta znači da je jedan piksel niske rezolucije predstavljen s 2^2 piksela visoke rezolucije, odnosno F^2 u općenitom slučaju. Postavljanje piksela niske rezolucije samo na piksel koji je cjelobrojni višekratnik faktora povećanja omogućava da se tada pod-pikselni pomaci prisutni u mreži niske rezolucije izvode na mreži visoke rezolucije jednostavnim pomicanjem piksela po području veličine F^2 koje je rezervirano za taj piksel niske rezolucije. Pikseli kojima nije dodijeljena vrijednost jednaki su -1.

Glavna petlja algoritma koja učitava slike niske rezolucije i vrši njihovu registraciju prikazana je u nastavku:

```
% Allocate frame that counts number of overlapping frames on a pixel.
overlapHR = ones(heightHR, widthHR);

% Start the main loop.
```

```
for i = 2:numFrames % skip 1 step because of reference frame

    % Fetch LR frame.
    frameLR = inLR(:, (i-1)*widthLR+1 : i*widthLR);
    % Detect shifts of LR frame.
    [shiftX, shiftY] = getShifts(reference, frameLR);
    shiftX = round(shiftX*factor);
    shiftY = round(shiftY*factor);
    % Create HR frame.
    frameHR = doHR(frameLR, factor);
    % Shift HR frame and assemble it on the resulting SR frame.
    [SR, overlapHR] = adjustHR2(SR, overlapHR,frameHR,shiftX,shiftY);

end
```

Detektirani pomaci na mreži niske rezolucije realni su brojevi, u idealnom slučaju apsolutne vrijednosti manje od 1, koji se pretvaraju u cjelobrojne pomake na mreži visoke rezolucije množenjem s faktorom povećanja i zaokruživanjem dobivenog rezultata. Time je projekcija piksela niske rezolucije na mrežu visoke rezolucije određena kao zaokruživanje na najbližu cjelobrojnu vrijednost.

Glavne funkcije korištene u ovoj petlji su samostalno napisane `getShifts` i `adjustHR2`. Funkcija `getShifts` izravno implementira matričnu jednadžbu modificirane Irani i Peleg metode detekcije pomaka prezentirane u poglavlju 4.2.:

```
function [shiftX, shiftY] = getShifts(referenceFrame,observationFrame)

% Convert input images to double because of command compatibility.
o1 = double(referenceFrame);
ok = double(observationFrame);

%% The reference frame part of equation.
% Same for all observation frames.
d1m = diff(o1, 1, 2); % horizontal diff
d1m = d1m(2:end, :); % fix size
d1m2 = d1m.^2; % square
d1n = diff(o1, 1, 1); % vertical diff
d1n = d1n(:, 2:end); % fix size
d1n2 = d1n.^2; % square

% Creating elements for M matrix of the equation.
Me1 = sum(sum(d1m2, 1), 2);
Me2 = sum(sum(d1m.*d1n, 1), 2);
Me3 = Me2;
Me4 = sum(sum(d1n2, 1), 2);

%% The observation frame part of the equation.
dk1 = ok-o1; % subtraction
dk1 = dk1(2:end, 2:end); % fix size
dkm = dk1.*d1m;
dkn = dk1.*d1n;
```

```
% Creating elements for W matrix of the equation.
We1 = sum(sum(dkm, 1), 2);
We2 = sum(sum(dkn, 1), 2);

%% Solving the equation
% Construct M and W matrices.
M = [Me1 Me2; Me3 Me4];
W = [We1; We2];
% Solve for horizontal and vertical shift, S = [h,v]'.
%S = inv(M)*W;
S = M\W;

shiftX = S(1);
shiftY = S(2);

end
```

Funkcija `adjustHR2` vrši registraciju slika niske rezolucije preslikanih na mrežu visoke rezolucije funkcijom `doHR`, uzimajući u obzir detektirani pomak. Rezultat se za sve slike niske rezolucije pohranjuje na zajedničkoj slici visoke rezolucije koja će na kraju sadržavati nepravilno raspoređene piksele nastale registracijom slika niske rezolucije. Funkcija također vodi računa o preklapanju više slika na iste piksele prilikom registracije. Taj problem rješava tako da pamti broj slika koje se preklapaju na pojedinom pikselu u varijabli `overlapHR`, te se nakon izlaska iz glavne petlje svaki piksel mora podijeliti s brojem preklapajućih slika na njemu. Funkcija `adjustHR2` nalazi se u nastavku:

```
function [outputHR, overlapHR] =
adjustHR2(inputHR,overlapHR,frameHR,shiftX,shiftY)

% Shifting HR frame.
frameHR = circshift(frameHR, [shiftY shiftX]);
% Overlapping shifted frame over input HR frame
for x = 1 : size(frameHR, 2)
    for y = 1 : size(frameHR, 1)
        if (frameHR(y, x) ~= -1)
            if (inputHR(y, x) == -1)
                inputHR(y, x) = frameHR(y, x);
            else
                inputHR(y, x) = inputHR(y, x) + frameHR(y, x);
                overlapHR(y, x) = overlapHR(y, x) + 1;
            end
        end
    end
end
outputHR = inputHR;
end
```

Nakon što glavna petlja završi obavlja se spomenuto dijeljenje svakog piksela dobivene slike visoke rezolucije s brojem slika koje se preklapaju na svakom pikselu:

```
% Average overlapping frames.
SR = SR./overlapHR;
```

Slijedi korak rekonstrukcije slike visoke rezolucije korištenjem modificirane Papoulis-Gerchberg metode prezentirane u poglavlju 4.3 koja se sastoji od filtriranja dobivene slike visoke rezolucije. Koristi se samostalno napisana funkcija `filterSR22`:

```
function fHR = filterSR22(HR,LR,force,factor,order,cutoff,iterations)
%% Based on PG method.

width = size(HR, 2);
height = size(HR, 1);

know = HR;
HR(HR == -1) = 0;

b = fir1(order, cutoff);

psfSize = factor;
psfSig = 1;
PSF = fspecial('gaussian', [psfSize psfSize], psfSig);
c = sum(sum(PSF));

for j = 1:iterations

    % Forcing.

    % Regular PG.
    if force == 1
        for x = 1:width;
            for y = 1:height
                if know(y, x) ~= -1
                    HR(y, x) = know(y, x);
                end
            end
        end
    end
    % PG with back projection.
    if force == 2
        a = zeros(size(LR));
        e = zeros(size(LR));
        for x = 1:width/factor
            for y = 1:height/factor
                frame = HR( (y-1)*factor+1:y*factor,
                            (x-1)*factor+1:x*factor );
                a(y, x) = sum(sum(PSF.*frame))/c;
                e(y, x) = LR(y, x) -a(y, x);
                frame =  frame + PSF .* e(y, x);
                HR( (y-1)*factor+1:y*factor,
                    (x-1)*factor+1:x*factor ) = frame;
            end
        end
    end
```

```
    % Filtering.

    for i = 1:width
        HR(:, i) = filter(b, 1, HR(:, i));
    end

    for i = 1:height
        HR(i, :) = filter(b, 1, HR(i, :));
    end

    HR = abs(HR);
    HR = circshift(HR, [-order/2, -order/2]);

end

fHR = HR;

end
```

Funkcija vrši filtriranje po stupcima, a zatim po redovima slike. Ovakav pristup odabran je kako bi se, imajući na umu da se ovaj algoritam mora implementirati na FPGA sklopu, unaprijed ispitala učinkovitost ovakvog način jednodimenzionalnog filtriranja koje je računalno jednostavnije od dvodimenzionalnog filtriranja. Kao što će biti pokazano u idućim poglavljima rezultati su veoma zadovoljavajući. Ispitivanja su također pokazala da je zadovoljavajući broj iteracija modificirane Papoulis-Gerchberg metode jednak 7, odnosno poboljšanja koja nastaju s više od 7 iteracija nisu toliko značajna, a zahtijevaju određeno vrijeme dok se izvrše. Također broj iteracija moguće je smanjiti do samo 3 ako je potrebno, jer je i tada vidljivo poboljšanje u odnosu na standardne metode. U literaturi vezanoj za Papoulis-Gerchberg metodu spominje se modificirana metoda koja umjesto vraćanja inicijalnih vrijednosti poznatim pikselima nakon filtriranja zapravo vrši proces iterativne projekcije unazad [4]. Funkcija `filterSR22` ima ugrađenu i tu funkcionalnost preko ulaznog parametra `force`, ali su ispitivanja pokazala da rezultat nije zadovoljavajući, vjerojatno ponovno zbog lošeg odabira jezgre projekcije unazad koja je pretpostavljena kao Gaussova krivulja.

Ovdje je važno napomenuti da nakon filtriranja na donjem i desnom rubu slike postoje područja piksela, širine reda korištenog filtra, čija je vrijednost nula kao posljedica kašnjenja filtra. Također, gornji i lijevi rub slike mogu sadržavati par linija koje odudaraju od ostalih piksela, što je posljedica pomaka slika pri registraciji, jer je pomak izveden ciklički radi jednostavnosti. Općenito se u rezultatu mogu zanemariti rubovi slike jer su žrtvovani radi brzine i jednostavnosti algoritma.

Implementacija završava pohranjivanjem dobivene slike visoke rezolucije u posebnu datoteku:

```
imwrite(SRu8, fileName),
```

te generiranjem slike dobivene standardnim povećanjem radi usporedbe:

```
HR = imresize(reference, factor).
```

Sljedeći korak je pojednostavljenje implementacije algoritma na jednostavne operacije koje se mogu izvesti na FPGA sklopu.

5.2. Detalji pojednostavljene implementacije

Cilj ovog koraka je pretvoriti sve naredbe korištene u implementaciji algoritma u jednostavne naredbe izvedive na FPGA sklopu. Naime, MATLAB je viši programski jezik orijentiran na rad s matricama. Mnoge matematičke funkcije i operacije dostupne su kao gotova rješenja, a najbolja obrada podataka vrši se njihovom organizacijom u matrice čime je pristup pojedinim podacima brži, a moguća je i istovremena obrada velike količine podataka. FPGA sklop nema dostupna gotova rješenja za kompleksne matematičke operacije ili rad s matricama, stoga je promjena naredbi u implementiranom modelu nužna.

Funkcije koje služe učitavanju slika ili analizi rezultata nije potrebno mijenjati, jer se na FPGA implementira funkcionalni dio algoritma koji prima slike niske rezolucije a vraća sliku visoke rezolucije. Pojednostavljeni algoritam dan je u nastavku:

```
% Create reference LR frame.
reference = inLR(:, 1 : widthLR);
% Create starting SR frame from reference LR frame.
SR = makeHR_v(reference, factor);
% Allocate frame that counts number of overlapping frames on a pixel.
overlap = zeros(factor, factor);

% Start the main loop for shift detection.
for i = 2:numFrames % skip 1 step because of reference frame

    % Fetch LR frame.
    frameLR = inLR(:, (i-1)*widthLR+1 : i*widthLR);
    % Detect shifts of LR frame.
    [Me1, Me2, Me3, Me4, We1, We2] = getShiftsRO_v(reference,frameLR);
    [shiftX, shiftY] = getShifts_v(Me1, Me2, Me3, Me4, We1, We2);
    shiftX = round(shiftX*factor);
    shiftY = round(shiftY*factor);
    % Update overlapping frames.
    overlap = makeOverlap_v(overlap, factor, shiftX, shiftY);
    % Create HR frame.
    frameHR = makeHR_v(frameLR, factor);
```

```
    % Shift HR frame and assemble it on the resulting SR frame.
    SR = makeBaseSR_v(SR, frameHR, shiftX, shiftY);

end

% Average overlapping frames.
SR = removeOverlap_v(SR, overlap);

% PG method filter.
b1 = fir1(orderPG, cutoffPG);

known = zeros(size(SR));
known = SR;

% Start the main loop for SR refinement.
for j = 1:iterations
    SR = forcingPG_v(SR, known);
    filterDirection = 0; % filterDirection = 0 = filtration of columns
    SR = filter_v(SR, b1, filterDirection);
    filterDirection = 1; % filterDirection = 1 = filtration of rows
    SR = filter_v(SR, b1, filterDirection);
end

% Round the result.
SR = round(SR);
```

Prva velika promjena je rastavljanje funkcije za detekciju pomaka u dvije funkcije. Funkcija `getShiftsRO_v` na temelju referentne i ulazne slike niske rezolucije računa i vraća elemente matrica korištenih u jednadžbi za detekciju pomaka, dok funkcija `getShifts_v` provodi račun matrične jednadžbe. Glavna petlja funkcije `getShiftsRO_v` izgleda ovako:

```
for x = 2:width
    for y = 2:height

        d1m(y, x) = referenceFrame(y, x) - referenceFrame(y, x-1);
        d1m2(y, x) = d1m(y, x) * d1m(y, x);

        d1n(y, x) = referenceFrame(y, x) - referenceFrame(y-1, x);
        d1n2(y, x) = d1n(y, x) * d1n(y, x);

        Me1 = Me1 + d1m2(y, x);
        Me2 = Me2 + d1m(y, x) * d1n(y, x);
        Me3 = Me2;
        Me4 = Me4 + d1n2(y, x);
        dk1(y, x) = observationFrame(y, x) - referenceFrame(y, x);
        dkm(y, x) = dk1(y, x) * d1m(y, x);
        dkn(y, x) = dk1(y, x) * d1n(y, x);
        We1 = We1 + dkm(y, x);
        We2 = We2 + dkn(y, x);
    end
end
```

Operacije diferencije i instantnog sumiranja zamijenjene su operacijama po pikselima. Algoritam je bilo moguće rastaviti na operacije koje se mogu izvoditi piksel po piksel sa spremanjem rezultata u međuspremnike. Ovakva implementacija moguća je u FPGA sklopu. Zanimljiva je i implementacija funkcije `getShifts_v` koja glasi:

```
invMc = 1/(Me1*Me4 - Me2*Me3);
shiftX = invMc * (Me4 * We1 - Me2 * We2);
shiftY = invMc * (-Me3 * We1 + Me1 * We2);
```

Rješenje matrične jednadžbe koje usput zahtjeva i računanje inverza jedne matrice pretvoreno je u jednostavniji oblik. Iskorištena je činjenica da se radi o kvadratnoj matrici 2 puta 2, čiji se inverz može eksplicitno izraziti.

Uvedena promjena je i funkcija `makeOverlap_v` koja umjesto da pamti broj preklopljenih slika na svakom pikselu koristi činjenicu da se preklapanje može promatrati na jednom kvadratu s bazom veličine faktora povećanja. To je moguće jer su sve slike niske rezolucije preslikane na mrežu visoke rezolucije s razmakom između poznatih piksela iznosa faktora povećanja, pa se pikseli na kojima je došlo do preklapanja pravilno ponavljaju. Funkcija računa broj preklopljenih slika na promatranom kvadratu na temelju detektiranih pomaka za trenutnu sliku:

```
if shiftX < 0
    shiftX = factor + shiftX;
end
if shiftY < 0
    shiftY = factor + shiftY;
end
base(mod(shiftY, factor)+1, mod(shiftX, factor)+1) = ...
    base(mod(shiftY, factor)+1, mod(shiftX, factor)+1) + 1;
```

Promijenjen je i način provedbe filtriranja prilikom rekonstrukcije slike visoke rezolucije. Razvijen je jedan filtar koji kao parametar uzima smjer filtriranja, čime se određuje filtriraju li se stupci ili redci slike. Jednodimenzionalni filtar moguće je realizirati u FPGA kao konvoluciju. Algoritam također više ne koristi -1 kao vrijednost koja označava piksel s neodređenom vrijednošću, već je ta vrijednost postala nula. Svi pikseli ulaznih slika niske rezolucije koji imaju vrijednost nula na početku su promijenjeni u vrijednost 1. Razlika između vrijednosti 0 i 1 kod zapisa od 8 bitova po

pikselu nije vidljiva jer se radi o najtamnijim tonovima, a korištenje nule za oznaku piksela s neodređenom vrijednosti olakšava rad prelaskom na FPGA sklop. Ostale promjene uključuju zamjenu svakog indeksiranja matrice s *for* petljom te obraćanje pažnje da se iste varijable ne koriste za pohranu drugačijih tipova podataka, što je inače dozvoljeno u MATLAB-u.

5.3. Rezultati

Uspješnom implementacijom pojednostavljenog algoritma postignuto je da obje verzije algoritma daju jednake rezultate koje je stoga moguće prezentirati korištenjem bilo koje verzije algoritma.

Kao mjera kvalitete povećanja rezolucije korištena je vizualna kvaliteta nastale slike u odnosu na sliku nastalu standardnim povećanjem za jednaki faktor. U tu svrhu korištena je MATLAB-ova funkcija *imresize* koja pri povećanju slike koristi bikubičnu interpolaciju pri računanju nepoznatih vrijednosti, što je prilično dobra metoda interpolacije. U nastavku slijede referentna slika niske rezolucije, rezultat povećanja izdvojenog dijela slike faktorom 2 korištenjem bikubične interpolacije te zatim korištenjem implementirane metode super rezolucije:

Slika 5-2: Referentna slika niske rezolucije

Slika 5-3: Povećanje faktorom 2 korištenjem bikubične interpolacije

Slika 5-4: Povećanje faktorom 2 korištenjem super rezolucije

Na slici 5-4 nastaloj metodom super rezolucije vidljivi su ranije spomenuti efekti crnih rubova i linija koji se mogu zanemariti. Na kazaljci zidnog sata koja pokazuje broj 11 te okruglom rubu sata moguće je primijetiti poboljšanje u odnosu na bikubičnu interpolaciju. Pravi rezultati postaju vidljivi pri većim faktorima povećanja. U nastavku slijede rezultati povećanja faktorom 3:

Slika 5-5: Povećanje faktorom 3 korištenjem bikubične interpolacije

Slika 5-6: Povećanje faktorom 3 korištenjem super rezolucije

Poboljšanje metode super rezolucije u odnosu na bikubičnu interpolaciju je očito. Slika je manje „zrnasta“, a oblici su jasniji. Povećanje faktorom 4 također daje dobre rezultate:

Slika 5-7: Povećanje faktorom 4 korištenjem bikubične interpolacije

Slika 5-8: Povećanje faktorom 4 korištenjem super rezolucije

Vidljivo je kako implementirana metoda super rezolucije rezultira boljim uvećanjem slike, s manjom „zrnatošću“ i više detalja. Za povećanja faktorom većim od 4 rezultat super rezolucije postaje sve zamućeniji i neiskoristiv. Kao primjer će poslužiti povećanje faktorom 6:

Slika 5-9: Povećanje faktorom 6 korištenjem super rezolucije

Još jedan primjer povećanja detalja slike faktorom 4:

Slika 5-10: Povećanje faktorom 4: lijevo bikubična interpolacija, desno super rezolucija

Na slici 5-10 moguće je primijetiti bolju razinu detalja dobivenu metodom super rezolucije, poput jasnijeg teksta na boci te u gornjem lijevom kutu slike. Također oblik nogometne lopte u donjem desnom kutu slike je puno jasniji, kao i oblik i detalji znaka između spomenute nogometne lopte i teksta na boci.

Važno je prisjetiti se pretpostavke o „mirnom“ držanju kamere tijekom snimanja zbog koje su međusobni pomaci slika niske rezolucije dovoljno mali te je slike moguće uspješno registrirati. Što se dogodi kada tijekom snimanja niza slika previše pomičemo kameru, te stvaramo pomake koji su reda veličine nekoliko piksela, pokazuje nam iduća slika:

Slika 5-11: Povećanje faktorom 4 uz prevelike pomake između slika niske rezolucije

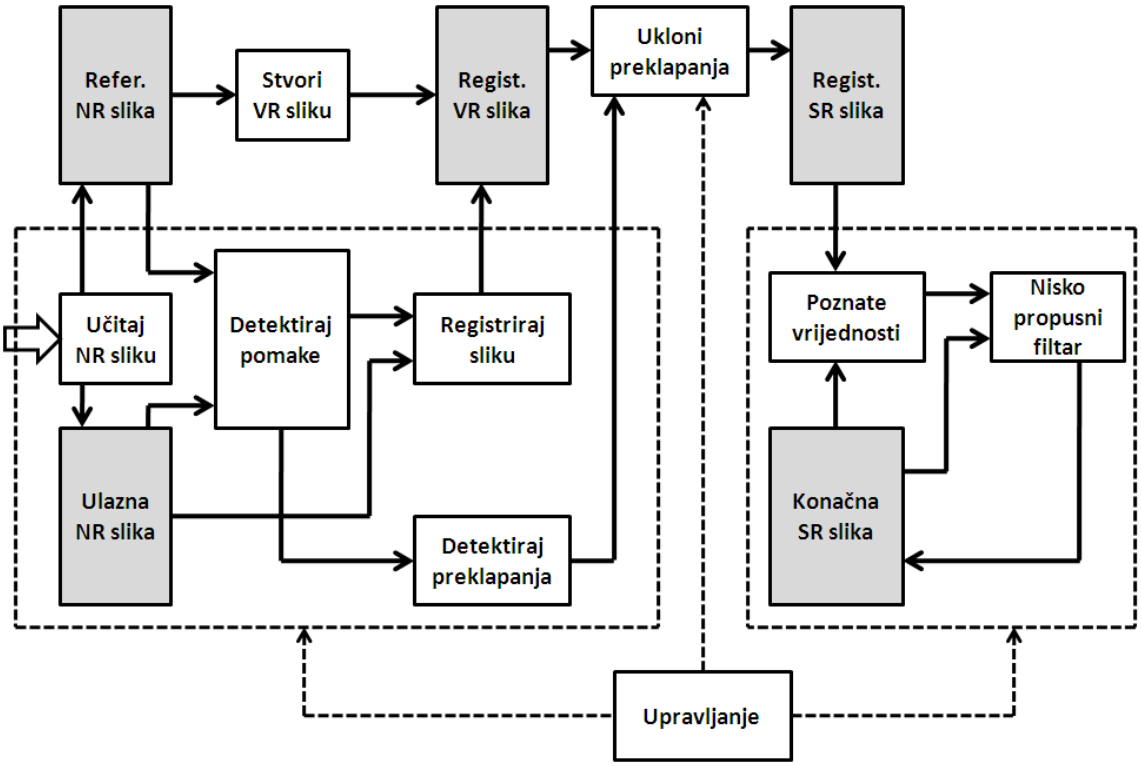
Vidljivo je kako rezultat nije zadovoljavajući jer metoda registracije ne uspijeva točno registrirati slike niske rezolucije te dolazi do zamućenja. Zbog toga je važno da međusobni pomaci slika niske rezolucije budu pod-pikselni. Ispitivanja su pokazala da je potreban broj slika niske rezolucije moguće smanjiti na 10 ako je to potrebno, uz očuvanje boljeg rezultata u odnosu na standardne metode povećanja. Na priloženom kompaktnom disku nalazi se još nekoliko slika na kojima je moguće pregledati uspješnost implementirane metode super rezolucije.

6. Implementacija algoritma korištenjem jezika za opis sklopovlja VHDL

6.1. Detalji implementacije

Predložena je arhitektura sustava koji ostvaruje razvijeni algoritam super rezolucije u stvarnom vremenu te je implementiran ponašajni VHDL model. Za razvoj je korišten Xilinx ISE Design Suite 13.4 s besplatnom studentskom Webpack licencom. Ta licenca uvodi neka ograničenja koja uključuju ograničen rad ISim(O.87xd) simulatora. Iz tog razloga je model razvijan sa slikama niske rezolucije veličine 32 x 24 piksela i maksimalnim povećanjem faktorom 4, što rezultira slikama veličine 128 x 96 piksela. Originalne slike korištene u razvoju algoritma u programskom jeziku MATLAB su veličine 320 x 240 piksela za nisku rezoluciju i 1280 x 960 za visoku rezoluciju uz povećanje faktorom 4. Uhodana implementacija se tada jednostavno memorijski proširi, uz prethodni izračun količine potrebnih operacija te memorijskih resursa. Sukladno tome potrebno je odabrati odgovarajući FPGA sklop.

Na slici 6-1 prikazana je blok shema predložene arhitekture koja je realizirana kao ponašajni VHDL model:



Slika 6-1: Arhitektura VHDL modela

Arhitektura se sastoji od dvije faze koje se mogu paralelno odvijati, a na blok shemi su vizualno odvojene pozicioniranjem s lijeve i desne strane bloka *Upravljanje*. U prvoj fazi vrši se registracija slika niske rezolucije i stvara mreža visoke rezolucije s nepravilnim rasporedom piksela. U drugoj fazi vrši se rekonstrukcija slike visoke rezolucije. VHDL kod moguće je pronaći na priloženom kompaktnom disku, a u nastavku slijedi detaljniji opis korištenih blokova. Implementirano povećanje iznosi 4 jer je tijekom ispitivanja sustava u MATLAB-u pokazano da je to najveći faktor za koji postoji poboljšanje.

Učitaj NR sliku je blok koji učitava sliku niske rezolucije iz vanjske memorije u radnu memoriju sustava. Radi kao automat s konačnim brojem stanja te su mu za obradu jednog piksela slike niske rezolucije potrebna 3 perioda takta.

Referentna NR slika je blok koji koristi dvopristupnu Block RAM memoriju FPGA sklopa i služi za pohranjivanje referentne slike niske rezolucije. Sadrži 768 lokacija veličine 8 bitova, jer su slike niske rezolucije veličine 32 x 24 piksela zapisane s po 8 bitova po pikselu. Tijekom obrade jednog niza slika sadržaj ovog bloka se ne mijenja.

Ulazna NR slika također je blok koji koristi dvopristupnu Block RAM memoriju FPGA sklopa, a služi za pohranjivanje ulazne slike niske rezolucije. Karakteristike su mu jednake kao karakteristike bloka *Referentna NR slika* samo što mu se sadržaj mijenja dolaskom svake nove slike niske rezolucije.

Detektiraj pomake je blok koji na temelju referentne i ulazne slike niske rezolucije iz blokova *Referentna NR slika* i *Ulazna NR slika* računa njihove međusobne pomake u horizontalnoj i vertikalnoj osi kao pod-pikselne pomake, te ih vraća zaokružene nakon množenja s faktorom povećanja. Radi kao automat s konačnim brojem stanja te mu je za obradu jednog piksela slike niske rezolucije potrebno 6 perioda takta.

Registrirana VR slika je blok koji koristi dvopristupnu Block RAM memoriju i veličine je 12288 lokacija po 8 bitova, što odgovara zapisu slike visoke rezolucije od 128 x 96 piksela. Služi za pohranu mreže visoke rezolucije nastale registracijom slika niske rezolucije. Njegov sadržaj se mijenja dolaskom svake nove slike niske rezolucije.

Registriraj sliku je blok koji vrši registraciju slike niske rezolucije, odnosno sliku niske rezolucije iz bloka *Ulazna NR slika* na temelju detektiranih pomaka iz bloka *Detektiraj pomake* postavlja na mrežu visoke rezolucije prisutnu u bloku *Registrirana VR slika*. Ako dolazi do preklapanja slika na određenim pikselima njihove se vrijednosti zbrajaju te se u posebnom bloku vodi evidencija o lokaciji preklapanje kako bi se na kraju registracije svih slika izvršilo izjednačavanje vrijednosti piksela na kojima je došlo do preklapanja. Blok radi kao automat s konačnim brojem stanja te su mu za obradu jednog piksela niske rezolucije potrebna 3 perioda takta.

Detektiraj preklapanja je blok koji koristi Block RAM memoriju veličine 16 piksela, što odgovara kvadratu s bazom veličine faktora povećanja. Na temelju detektiranih pomaka računa koliko se slika niske rezolucije međusobno preklapa na pojedinom pikselu tog kvadrata kao što je objašnjeno u poglavlju 5.2. Na kraju registracije svih slika na svakom će se pikselu izvršiti dijeljenje s brojem slika koje se na njemu preklapaju. Ovaj blok vraća broj slika koje se preklapaju na određenom pikselu visoke rezolucije zadanom njegovom adresom u bloku *Registrirana VR slika*. Radi kao automat s konačnim brojem stanja, a za pohranu lokacije preklapanja ili vraćanje broja preklapanih slika na pikselu visoke rezolucije potrebna su mu 2 perioda takta.

Stvori VR sliku je blok koji referentnu sliku niske rezolucije iz bloka *Referentna NR slika* stavlja na mrežu visoke rezolucije u bloku *Registrirana VR slika*, a razmaci

između piksela iznose faktor povećanja. Vrši inicijalizaciju bloka *Registrirana VR slika* postavljajući piksele s nepoznatim vrijednostima na nulu. Radi kao automat s konačnim brojem stanja te su mu za obradu jednog piksela slike visoke rezolucije potrebna 3 perioda takta.

Ukloni preklapanja je blok između dvije faze čija je zadaća na temelju rezultata iz bloka *Detektiraj preklapanja* odrediti vrijednosti piksela u bloku *Registrirana VR slika* na kojima je došlo do preklapanja nekoliko slika tako da se vrijednost piksela podijeli s brojem slika koje se preklapaju na tom pikselu. Rezultat se pohranjuje u blok *Registrirana SR slika*. Radi kao automat s konačnim brojem stanja te su mu za obradu jednog piksela visoke rezolucije potrebna 3 perioda takta.

Registrirana SR slika i *Konačna SR slika* su memorijski blokovi veličine 12288 lokacija po 8 bitova, što odgovara slici visoke rezolucije. Smatra se da su to vanjske memorije.

Registrirana SR slika sadrži rezultat faze registracije niskih slika, nakon što su određene vrijednosti svih piksela na kojima je došlo do preklapanja. Uloga ovog bloka je da pamti rezultat registracije slika niske rezolucije kako bi se vrijednosti poznatih piksela mogle ponovno vraćati tim pikselima u početnoj fazi Papoulis-Gerchberg metode. Također odvaja memorijske resurse faze rekonstrukcije od faze registracije čime je omogućeno paralelno izvođenje tih dviju faza.

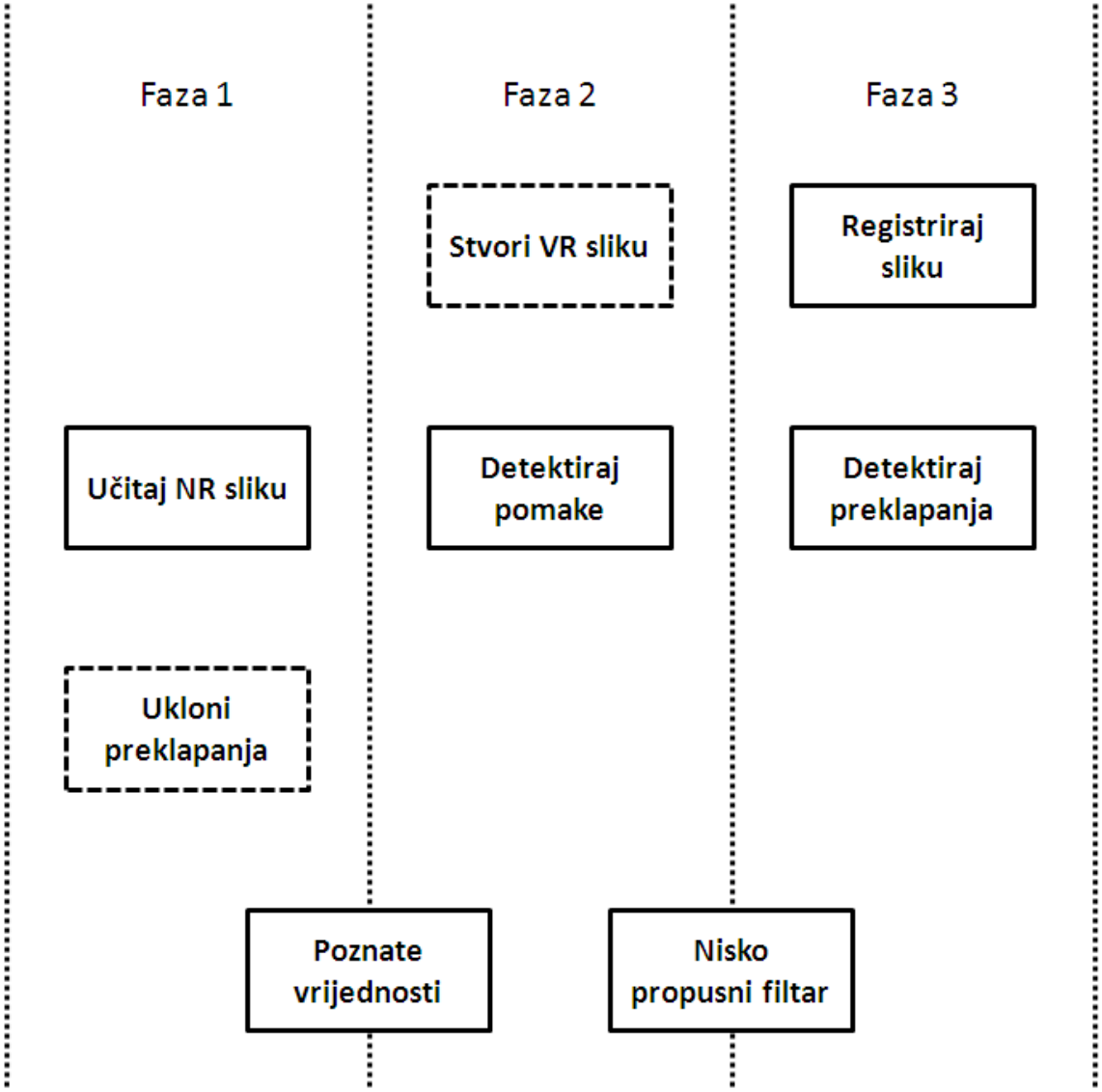
Poznate vrijednosti i *Nisko propusni filtar* blokovi su koji realiziraju metodu rekonstrukcije slike visoke rezolucije korištenjem Papoulis-Gerchberg metode. Blokom *Nisko propusni filtar* vrši se filtriranje podataka pohranjenih u bloku *Registrirana SR slika* u vertikalnom te zatim u horizontalnom smjeru. Blok *Poznate vrijednosti* brine o tome da se kod vertikalnog filtriranja na poznatim pikselima koriste poznate vrijednosti piksela nastale registracijom slika niske rezolucije, a prosljeđuje ih iz bloka *Registrirana SR slika*. Nakon filtriranja vrijednosti su pohranjene u blok *Konačna SR slika*. Sustav rekonstrukcije slike radi kao automat s konačnim brojem stanja te mu je za obradu jednog piksela visoke rezolucije potrebno 6 perioda takta.

Konačna SR slika je dakle blok koji pohranjuje konačan rezultat super rezolucije.

Upravljanje je blok čija je uloga sinkronizacija rada ostalih blokova. To uključuje ponavljanje faze registracije slika niske rezolucije dok sve slike nisu učitane, uklanjanje preklapanja između dviju faza, ponavljanje Papoulis-Gerchberg metode potrebni broj

iteracija u fazi rekonstrukcije slike visoke rezolucije te ostale upravljačke poslove unutar pojedinih blokova. Radi kao automat s konačnim brojem stanja.

Odvojenost memorijskih blokova nad kojima se vrše operacije u fazama registracije i rekonstrukcije slike omogućava da se te dvije faze istovremeno izvršavaju. Dok se izvršava rekonstrukcija slike visoke rezolucije novi niz slika niske rezolucije može se paralelno registrirati, a o pravilnoj sinkronizaciji ovakvog paralelnog rada brine se blok *Upravljanje*. Paralelno izvršavanje ilustrirano je na slici 6-2:



Slika 6-2: Ilustracija paralelnog izvršavanja

Blok *Ukloni preklapanja* koristi se samo kada su registrirane sve slike niske rezolucije te nakon njega počinje novi ciklus rekonstrukcije slike blokovima *Poznate vrijednosti* i *Nisko propusni filtar*. U novom ciklusu ti se blokovi počinju koristiti u fazi 2, no jednom kad su preklapanja uklonjena oni se koriste već u fazi 1. Početkom novog ciklusa učitava se nova referentna slika niske rezolucije i samo se tada koristi blok *Stvori VR sliku*. Ostali blokovi koriste se konstantno u odgovarajućim fazama.

6.2. Rezultati i daljnji rad

U jeziku za opis sklopovlja VHDL razvijen je ponašajni model predložene arhitekture sustava za realizaciju super rezolucije u stvarnom vremenu. Kao referentni rezultati smatrani su rezultati dobiveni realizacijom u MATLAB-u, te je pregledom sadržaja memorije u bloku *Konačna SR slika* moguće zaključiti da su rezultati zadovoljavajući. Postoje odstupanja koja su tolerantna i posljedica su razlike cjelobrojnog računanja na FPGA sklopu i računa s realnim brojevima u MATLAB-u.

Kako bi ponašajni model bio u potpunosti implementiran potrebno je dodatnu pažnju posvetiti blokovima *Ukloni preklapanja* i *Nisko propusni filtar*, odnosno učiniti ih implementabilnima. U bloku *Ukloni preklapanja* dolazi do cjelobrojnog dijeljenja koje je potrebno realizirati korištenjem ponuđenih specijaliziranih IP jezgri, čime bi za obradu jednog piksela visoke rezolucije bilo potrebno dodatnih 3 perioda takta na već postojeća potrebna 3 perioda takta. Blok *Nisko propusni filtar* nastao je korištenjem alata HDL Coder koji je sastavni dio MATLAB-a i generira VHDL kod za korištene MATLAB blokove koje je moguće prilagoditi i prevesti na jezik za opis sklopovlja VHDL. Nisko propusni filtar je tim alatom preveden, ali računa s realnim brojevima, stoga je potrebno prijeći na računanje s realnim brojevima u zapisu s nepomičnom decimalnom točkom. Moguća su dva rješenja od kojih prvo uključuje prebacivanje čitavog sustava na računanje s realnim brojevima u zapisu s nepomičnom decimalnom točkom, dok drugo uključuje samo prebacivanje faze rekonstrukcije slike na računanje s realnim brojevima u zapisu s nepomičnom decimalnom točkom.

Ostali blokovi uspješno su implementirani. Osim blokova koji koriste mnogo ugrađene Block RAM memorije vidljivo je da blokovi ne zauzimaju mnogo resursa jer su pisani tako da se prilikom implementacije prepoznaju gotovi obrasci poput automata s konačnim brojem stanja. Posebna je pažnja posvećena izbjegavanju gomilanja razina

korištene logike pa maksimalne frekvencije sklopova iznose preko 200 MHz. Arhitektura je temeljena na FPGA sklopu Virtex-5 XC5VLX330T jer uz brzu logiku nudi 11,664 Kb Block RAM memorije što je dovoljno za pohranu dvije slike niske rezolucije i jedne slike visoke rezolucije u radnu memoriju. Slika niske rezolucije zauzima 76,800 piksela, što je uz zapis od 8 bitova po pikselu ukupno 600 Kb. Slika visoke rezolucije zauzima 1,228,800 piksela, što je uz zapis od 8 bitova po pikselu ukupno 9,600 Kb. Ukupna potrebna memorija stoga iznosi 10,800 Kb. Sustav podrazumijeva vanjsku memoriju koja može pohraniti dvije slike visoke rezolucije u koju pohranjuje rezultat registracije slika i konačan rezultat.

Izradom ispitnog okruženja ispitana je brzina izvođenja ponašajnog modela. Zbog već spomenutog ograničenja simulatora ISim(O.87xd) ispitivanje je ograničeno na slike niske rezolucije 32 x 24 piksela, povećanje faktorom 4 koje daje slike visoke rezolucije 128 x 96 piksela, broj korištenih slika niske rezolucije je 4, a broj iteracija algoritma za rekonstrukciju slike je 2. Identični parametri odabrani su u MATLAB-u te su generirane datoteke koje sadrže ulazne slike niske rezolucije i konačnu sliku visoke rezolucije. Slike niske rezolucije učitane su u ispitno okruženje te je promatran jedan ciklus registracije četiriju slika niske rezolucije te proces rekonstrukcije slike visoke rezolucije kroz dvije iteracije. Konačnu brzinu izvođenja moguće je procijeniti izračunom pomoću potrebnog broja perioda takta za izvođenje pojedinih blokova. Funkcija koja opisuje kako broj potrebnih perioda takta za dobivanje konačne slike visoke rezolucije ovisi o broju piksela, broju ulaznih slika niske rezolucije te broju iteracija pri rekonstrukciji slike glasi:

$$num_{clk}(P, frames, iterations) = 3 \cdot P \cdot 16 + (3 + 6 + 3) \cdot frames \cdot P + (3 + 6 \cdot iterations) \cdot P \cdot 16,$$

gdje je *frames* broj ulaznih slika niske rezolucije, *iterations* broj iteracija pri rekonstrukciji slike, a *P* broj piksela slike niske rezolucije.

Podrazumijeva se povećanje faktorom 4. Uz poznati broj piksela slike niske rezolucije koji iznosi 76,800 te pretpostavljenih 16 ulaznih slika i 7 iteracija pri rekonstrukciji slike dolazimo do ukupne brojke od 73,728,000 potrebnih perioda takta. Uz pretpostavku da sklopovi koji čine sustav mogu raditi na frekvenciji od 200 MHz

vrijeme potrebno za nastanak slike visoke rezolucije je 0.36864 sekundi. Rezultat daje približnu sliku o brzini sustava jer se ipak ne radi o potpuno implementiranom sustavu te je konačna brzina procijenjena skaliranjem na više iteracija i ulaznih slika.

Ovdje je važno prilagoditi vrijeme izvođenja namjeni sustava jer su broj ulaznih slika i broj iteracija pri rekonstrukciji slike podesivi parametri. Ako se radi o stvaranju slike visoke rezolucije iz niza kadrova video zapisa, koji pristižu brzinom od 24 snimaka u sekundi, tada se uz potrebnih 16 snimaka za jednu sliku visoke rezolucije radi o nastanku slike visoke rezolucije svakih 0.6666 sekundi. To očito nije brzina prihvatljiva za reprodukciju video zapisa, ali je možda prihvatljiva za funkciju određenih nadzornih kamera. U slučaju korištenja satelitskih snimaka koje su statične prirode, nastanak slike svakih 0.36864 je prihvatljiv. Konkretan odabir parametara ovisi o primjeni, a uvijek je moguće smanjiti i faktor povećanja na 2 ili 3.

Daljnji rad uključuje potpunu implementaciju blokova *Ukloni preklapanja* i *Nisko propusni filtar* te ispitivanje utjecaja podesivih parametara na rad sustava. Ispitivanjem potpuno implementiranog sustava u stvarnom vremenu moguće je analizirati kritične dijelove te poboljšati protočnost sustava.

7. Zaključak

U ovom radu predstavljena je i implementirana metoda povećanja izvorne rezolucije slike korištenjem više snimaka. Analizirani su postojeći algoritmi te odabrani i prilagođeni oni koji mogu raditi u stvarnom vremenu. Algoritmi su odabrani na temelju performansi nakon implementacije u programskom jeziku MATLAB. Faza odabira zadovoljavajućeg algoritma trajala je veoma dugo jer su algoritmi opisani u literaturi često temeljeni na simuliranim slikama, dok je ovaj rad temeljen na stvarnim slikama dobivenim digitalnim fotoaparatom. Nakon odabira i prilagodbe korištenih algoritama slijedila je verifikacija rezultata gdje su ispitivani utjecaj raznih parametara te pronađene granice korištenih algoritama. Nakon optimiziranja rješenja predložena je arhitektura koja bi ostvarila algoritam na FPGA sklopu. Nakon analize arhitekture slijedila je implementacija u jeziku za opis sklopovlja VHDL. Zbog opsežnosti zadatka implementacija je ograničena na ponašajni model uz adekvatnu diskusiju o rezultatima i potrebnim koracima za potpunu implementaciju. Zbog ograničenja simulatora pod studentskom licencom ispitivanje ponašajnog modela bilo je ograničeno jer implementirani sustav bio prevelik. Zbog toga je korišten linearno umanjen model, a posebna je pažnja posvećena proširivanju rezultata ispitivanja na veličinu očekivanog sustava. Daljnji rad uključuje potpunu implementaciju te ispitivanje i poboljšanje stvarnog sustava

Kroz ovaj sam rad prošao sve faze rješavanja inženjerskog problema. Od definicije problema, pronalaska odgovarajućeg rješenja, implementacije rješenja na višoj razini pa sve do projektiranja i početka realizacije konkretnog sustava naučio sam mnogo. Zbog specifičnih zahtjeva zadatka neke od faza su ponavljane, što pokazuje kako je to iterativni proces. Zadatak je bio veoma opsežan te je u dogovoru s mentorom ponašajni model zadovoljavajući rezultat u zadanom roku, no sigurno ću nastaviti rad i potpuno implementirati sustav.

8. Literatura

[1] IRANI, M., PELEG, S.: *Improving Resolution by Image Registration*, CVGIP: Graphical Models and Image Processing, Vol. 53, No. 3, pp. 231 – 239, svibanj 1991.

[2] ALAM, S., BOGNAR, J., HARDIE, R., YASUDA, B.: *Infrared Image Registration and High-Resolution Reconstruction Using Multiple Translationally Shifted Aliased Video Frames*, IEEE-TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 49, NO. 5, listopad 2000.

[3] ANGELOPOULOU, M., BOUGANIS, C., CHEUNG, P., CONSTANTINIDES G.: *FPGA-Based Real-Time Super-Resolution on an Adaptive Image Sensor*, ARC 2008, LNCS 4943, pp. 125–136, 2008.

[4] CHATTERJEE, P., MUKHERJEE, S., CHAUDHURI S., SEETHARAMAN, G.: *Application of Papoulis-Gerchberg Method in Image Super-resolution and Inpainting*, The Computer Journal Vol. 00 No. 0, 2007.

[5] JORGE, P., FERREIRA, G.: *Interpolation and the Discrete Papoulis-Gerchberg Algorithm*, IEEE Trans. Signal Processing, Vol. 42, 1994.

[6] PARK, S.C., PARK, M.K., KANG, M.G.: *Super-Resolution Image Reconstruction: A Technical Overview*, IEEE SIGNAL PROCESSING MAGAZINE 1053-5888/03, svibanj 2003.

[7] AGUENA, M., MASCARENHAS, N.: *Generalization of Iterative RestorationTechniques for Super-Resolution*, SIBGRAPI '11 Proceedings of the 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images, pp. 258-265, 2011.

[8] ZITOVA, B., FLUSSER, J.: *Image registration methods: a survey*, Image and Vision Computing 21 (2003) 977–1000, lipanj 2003.

[9] ARAIZA, R., AVERILL M.G., KELLER, G.R., STARKS, S.A., BAJAJ, C.: *3-D Image Registration Using Fast Fourier Transform, with Potential Applications to Geoinformatics and Bioinformatics*, The University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-06-11, 2006.

[10] SCHÜLER, J., SCRIBNER, D., KRUER, M.: *Super Resolution Imagery From Multi-frame Sequences with Random Motion*, ADM201041, 1998 IRIS Proceedings, ožujak 1998.

[11] BOWEN, O., BOUGANIS, C.: *Real-Time Image Super Resolution Using an FPGA*, Field Programmable Logic and Applications, 2008. FPL 2008. International Conference, pp. 89-94, rujan 2008.

[12] LIN, Z., SHUM, H.-Y.: *Fundamental Limits of Reconstruction-Based Superresolution Algorithms under Local Translation*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 26, NO.1, siječanj 2004.

[13] WAGNERA, R., WAAGENB, D., CASSABAUMB, M.: *Image Super-Resolution for Improved Automatic Target Recognition*, SPIE Defense & Security Symposium, travanj 2004.

[14] VISHAL VISHWANATH ANAGIRE: *Reengineering of Super-Resolution*, International Journal of Computer Science And Technology, Vol. 3, Issue 1, ožujak 2012.

[15] FARRELL, J., XIAO, F., KAVUSI, S.: *Resolution and Light Sensitivity Tradeoff with Pixel Size*, SPIE Electronic Imaging 2006 Conference, vol. 6069, pp. 211–218, veljača 2006.

[16] MATLAB Product Documentation, http://www.mathworks.com/help/techdoc/

[17] VUČIĆ, M., MOLNAR, G.: *Alati za razvoj digitalnih sustava: Materijali za predavanja I, II, III*, Fakultet elektrotehnike i računarstva, Zavod za elektroničke sustave i obradbu informacija, Zagreb, 2010.

9. Sažetak

FPGA implementacija metode povećanja originalne rezolucije slike korištenjem više snimki

U ovom radu predstavljena je i implementirana metoda povećanja izvorne rezolucije slike korištenjem više snimaka koja može raditi u stvarnom vremenu, s potencijalnim korištenjem u obradi video zapisa.

Povećanje originalne rezolucije slike korištenjem više snimki metoda je poznata pod nazivom super rezolucija te ima široku primjenu u obradi medicinskih, vojnih, nadzornih, ali i komercijalnih slika. Temelji se na činjenici da uzastopne snimke iste scene pod pretpostavkom međusobnih sitnih pomaka manjih od veličine piksela sadrže drugačije informacije o snimanoj sceni. Obradom takvih snimki moguće je iskoristiti dodatne informacije i sastaviti sliku koja ima višu rezoluciju od početnih snimki.

Međusobni pomaci snimaka mogu biti kontrolirani i izazvani namjerno, ali mogu se dogoditi slučajno kao posljedica vibracija. Metoda super rezolucije sastoji se od koraka registracije slika niske rezolucije, koji uključuje procjenu međusobnih pomaka ako su oni slučajni, te koraka rekonstrukcije slike visoke rezolucije iz registriranih slika. Za registraciju slika niske rezolucije u ovom radu korišten je algoritam temeljen na diferenciji slika koji procjenjuje njihove međusobne translatorne pomake. Registrirane slike postavljene su na mrežu visoke rezolucije, a za rekonstrukciju slike visoke rezolucije korišten je algoritam koji se temelji na decimaciji i interpolaciji u frekvencijskoj domeni. Korišteni algoritam je modificiran i pretvoren u filtriranje kako bi bio lakše izvediv u stvarnom vremenu. Svi algoritmi su implementirani te ispitani u programskom jeziku MATLAB.

Predložena je arhitektura sustava koji ostvaruje metodu na FPGA sklopu te je razvijen ponašajni model u jeziku za opis sklopovlja VHDL. Na temelju dobivenih rezultata u ispitnom okruženju diskutirani su koraci za potpunu implementaciju sustava.

10. Summary

FPGA implementation of an image resolution enhancement method using multiple images

An image resolution enhancement method using multiple images is presented and implemented in this thesis. The method can work in real time with potential use in video processing.

Image resolution enhancement using multiple images is a method known as Super resolution. It has many applications in processing medical, military, surveillance but also commercial images. It is based on a fact that multiple images of the same scene contain different information about that scene, under assumption that there are small relative shifts between the images less than size of a pixel. Using the extra information by processing this kind of images it is possible to construct an image with higher resolution than the original images.

Relative shifts between the images can be controlled and intentional but also unintentional as a result of some vibrations. Super resolution method is made of two steps. The first one is registration of low resolution images which involves estimation of relative shifts if they are unknown. The second is high resolution image reconstruction from the registered images. In this thesis an algorithm based on image differentiation is used for registration of low resolution images. It estimates translator shifts between the images. Registered images are placed on a high resolution grid and a frequency domain algorithm based on decimation and interpolation is used for high resolution image reconstruction. The algorithm is modified and transformed into filtering so it can be easier to implement in real time. All algorithms are implemented and tested in MATLAB.

An architecture that implements the used method on FPGA is proposed and a behavioral model is designed in VHDL. Based on test bench results the steps for full implementation are discussed.

11. Skraćenice

FPGA *field-programmable gate array*

integrirani krug koji je moguće konfigurirati nakon proizvodnje

JPEG *Joint Photographic Experts Group*

format zapisa digitalnih slika

NR *niska rezolucija*

označava slike niske rezolucije

PNG *Portable Network Graphics*

format zapisa digitalnih slika

RAM *random-access memory*

memorija sa slučajnim pristupom, radna memorija

VHDL *very-high-speed integrated circuits hardware description language*

jezik za opis sklopovlja

VR *visoka rezolucija*

označava slike visoke rezolucije

12. Privitak

Privitak A

Kôd upravljačkog bloka ponašajnog modela u jeziku za opis sklopovlja VHDL korišten za ispitivanje jednog ciklusa implementiranog algoritma. Vrši se registracija četiriju slika niske rezolucije te rekonstrukcija slike visoke rezolucije kroz dvije iteracije.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
use work.myPackage.all;

entity control is
port (
      clk : in std_logic;
      rst : in std_logic;
      fin : out std_logic;
      newFrame : out std_logic;
      refRAM : out std_logic;
      inRAM : out std_logic;
      rst_overlap : out std_logic;
      addrLR : in std_logic_vector(addressLR downto 0);
      go_makeHR : out std_logic;
      fin_makeHR: in std_logic;
      go_shifts : out std_logic;
      fin_shifts: in std_logic;
      go_stack : out std_logic;
      fin_stack: in std_logic;
      go_overlap : out std_logic;
      fin_overlap: in std_logic;
      go_remove : out std_logic;
      fin_remove: in std_logic;
      go_ff : out std_logic;
      fin_ff : in std_logic
      );
end control;

architecture Behavioral of control is

      type states is (reset, refLR, inLR_makeHR, inLR, getShifts, makeStack_overlap,
remove, force_filter, force_filter_prep, idle);
      signal state : states;

      signal countFrames : std_logic_vector(3 downto 0);
      signal iterations : std_logic_vector(2 downto 0);
      signal clk_count : integer := 0;

begin

      process(clk)
      begin
            if rising_edge(clk) then
                  clk_count <= clk_count+1;
                  if rst = '1' then
                        fin <= '0';
                        newFrame <= '1';
                        refRAM <= '0';
                        inRAM <= '0';
                        rst_overlap <= '1';
                        go_makeHR <= '0';
                        go_shifts <= '0';
                        go_stack <= '0';
                        go_overlap <= '0';
                        go_remove <= '0';
                        go_ff <= '0';
```

```
		countFrames <= "0011";
		iterations <= "001";
		state <= reset;
	else
		case state is
			when reset =>
				state <= refLR;
				refRAM <= '1';
				rst_overlap <= '1';
			when refLR =>
				if addrLR+1 = 0 then
					state <= inLR_makeHR;
					refRAM <= '0';
					inRAM <= '1';
					go_makeHR <= '1';
				end if;
			when inLR_makeHR =>
				if fin_makeHR = '1' then
					state <= getShifts;
					inRAM <= '0';
					go_makeHR <= '0';
					go_shifts <= '1';
					newFrame <= '0';
				end if;
			when inLR =>
				if addrLR+1 = 0 then
					state <= getShifts;
					inRAM <= '0';
					go_shifts <= '1';
					newFrame <= '0';
				end if;
			when getShifts =>
				if fin_shifts = '1' then
					state <= makeStack_overlap;
					go_shifts <= '0';
					go_stack <= '1';
					go_overlap <= '1';
					rst_overlap <= '0';
				end if;
			when makeStack_overlap =>
				if fin_overlap = '1' then
					go_overlap <= '0';
				end if;
				if fin_stack = '1' then
					if countFrames = 1 then
						state <= remove;
						go_remove <= '1';
					else
						state <= inLR;
						inRAM <= '1';
						newFrame <= '1';
						countFrames <= countFrames - 1;
					end if;
					go_stack <= '0';
					go_overlap <= '0';
				end if;
			when remove =>
				if fin_remove = '1' then
					state <= force_filter;
					rst_overlap <= '0';
					go_remove <= '0';
					go_ff <= '1';
				end if;
			when force_filter =>
				if fin_ff = '1' then
					if iterations = 0 then
						state <= idle;
						go_ff <= '0';
					else
						state <= force_filter_prep;
						go_ff <= '0';
						iterations <= iterations - 1;
					end if;
				end if;
			when force_filter_prep =>
				go_ff <= '1';
				state <= force_filter;
```

```
                              when idle =>
                                    fin <= '1';
                                    if rst = '1' then
                                          state <= reset;
                                    end if;
                        end case;
                  end if;
            end if;
      end process;

end Behavioral;
```