Elektrotehnički fakultet
Univerzitet u Banjoj Luci

IZVJEŠTAJ PROJEKTNOG ZADATKA

iz predmeta

SISTEMI ZA DIGITALNU OBRADU SIGNALA

Mentori:

prof. dr Mladen Knežić
prof. dr Mitar Simić
ma Vedran Jovanović
dipl. ing. Dimitrije Obradović

Student:

Radislav Kosijer 1152/19

Februar 2025. godine

1. Opis projektnog zadatka

Prema projektnom zadatku, potrebno je realizirati sistem za generisanje empirijske vremensko-frekvencijske dekompozicije signala (Empirical Mode Decomposition - EMD) te primijeniti taj postupak na fuziju multifokusiranih slika.

Empirijski način vremensko-frekvencijske dekompozicije signala je pogodna tehnika za multirezolucionu dekompoziciju nelinearnih, nestacionarnih signala. EMD vrši adaptivno razlaganje signala na konačan skup amplitudno/frekvencijski modulisanih komponenti nazvanih unutrašnje mod funkcije (Intristic Mode Functions - IMFs). One predstavljaju skup oscilujućih funkcija ugrađenih u signal koji se razlaže, te tako ukazuju na frekvencijski sadržaj signala, pa se iste mogu tumačiti i kao skup osnovnih funkcija unutar posmatranog signala.

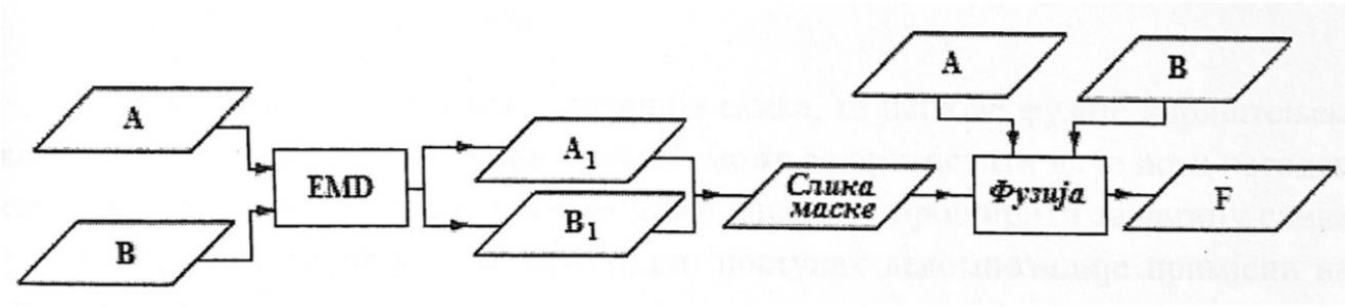
Da bi neka funkcija bila proglašena za IMF ona mora da zadovolji dva uslova, da broj prolazaka kroz nulu i broj ekstrema funkcije se može razlikovati za najviše jedan, i da na bilo kojoj lokaciji unutar signala srednja vrijednost anvelope definisana preko lokalnog minimuma i lokalnog maksimuma anvelope treba biti približno jednaka nula.

Postupak izdvajanja prvog IMF-a iz ulaznog 1D signala se može opisati na sljedeći način:

1. Prvo se detektuju lokalni minimumi i maksimumi ulaznog 1D signala.
2. Potom se od istih formiraju gornja i donja anvelopa.
3. Računa se srednja vrijednost zbira gornje i donje anvelope u svim tačkama i ona se oduzima od ulaznog signala formirajući kandidata za IMF.
4. Provjerava se da li su zadovoljeni uslovi da to bude IMF.
5. Ako je izdvojena srednja vrijednost zadovoljava uslove, ona se proglašava prvim nivoom EMD dekompozicije (prvi IMF), a ostatak se dalje razlaže ponavljajući postupak.

U sklopu zadatka, student je dužan:

- Učitati dvije multifokusirane slike, pretvoriti ih u grayscale format te formirati 1D vektore.
- Provesti EMD dekompoziciju signala izdvajajući prvi IMF.
- Izračunati lokalnu varijansu piksela (npr. koristeći 3x3 prozor) i generisati masku odlučivanja na temelju postavljenog praga ε.
- Napraviti fuziju dvije originalne slike na osnovu generisane maske.
- Izvršiti optimizaciju svakog koraka, mjerenje procesorskih ciklusa i analizu "bottle neck" dijelova koda.



Slika 1.1 – *Blok dijagram algoritma za fuziju multifokusiranih slika A i B*

2. Izrada projektnog zadatka

Sistem se sasatoji od nekoliko međusobno povezanih modula:

- Učitavanje i predobrada slike
- EMD dekompozicija
- Generisanje maske odlučivanja
- Fuzija slika i čuvanje binarne datoteke
- Upravljanje LE diodama
- Postobrada slike i čuvanje slike u željenom formatu

2.1. Učitavanje i predobrada slike

Za ovaj dio projektnog zadatka, korištena je *Python* skripta *generate_header.py*. Ova skripta prvo provjerava postoji li direktorijum *Images*, u kome se nalaze dvije multifokusirane slike. Zatim prolazi kroz sve datoteke unutar tog direktorija, obrađujući samo slike s ekstenzijama BMP, JPG ili JPEG. Svaka slika se otvara, konvertuje u grayscale, te se njeni piksel podaci pretvaraju u *NumPy* niz. Nakon toga se generiše C header datoteka koja sadrži dimenzije slike i niz piksela u heksadecimalnom zapisu, uz uključene *include guard*-ove i smještanje niza u specificifikovani memorijski segment, u ovom slučaju, seg*_sdram1*.

Ovakav pristup omogućava jednostavno učitavanje slika u C projektima.

2.2. EMD dekompozicija

Za dio projektnog zadatka koji je vezan za EMD dekompoziciju, prvo je potrebno da se svi privremeni nizovi smjeste u memorijski segment *seg_sdram1*, zbog tog što se radi sa velikim nizovima, koji ne mogu da se smjeste u internu memoriju. Takođe, zbog veličine ovih nizova i veličine *heap*-a, koji je dostupan, nije moguće dinamički alocirati nizove, tako da, jedini i najbolji način za rukovanje nizovima je zapravo postavljanje statičkih bafera u ovaj memorijski segment.

Bitna funckija za obradu signala u ovom dijelu programa je konvertovanje u format Q16.16. Ova funckija pretvara 8-bitne vrijednosti piksela u Q16.16 format pomoću šiftovanja za 16 bita. Ovo je bitno iz razloga što dobijamo dovoljno veliku preciznost u daljnjem toku algoritma

U algoritmu za emd, funckija detektuje lokalne maksimume i minimume signala, zatim računa gornju i donju anvelopu pomoću funkcije za linearnu interpolaciju, gdje se pro popunjavaju rubovi signala, a zatim se segmetni između interpoliraju i računa se nagib u Q16.16 formatu. Nakon interpolacije, signalu se oduzima srednja vrijednost envelope, čime se dobija prvi IMF, koji je i potreban za izradu ovog projektnog zadatka.

2.3. Generisanje maske odlučivanja

U ovom modulu, fokus je na generisanju maske odlučivanja na osnovu dobijenog prvog IMF-a. Prvo se računa lokalna varijansa pomoću kliznog prozora određene veličine. Za svaki piksel se računa suma i suma kvadrata vrijednosti u tom prozoru, a nakon tog se računa varijansa, koja se takođe računa u formatu Q16.16, sa odgovarajućim offset-om za pravilno zaokruživanje.

Nakon računanja lokalne varijanse, računa se i maska odlučivanja, koja, nad nizovima izračunate lokalne varijanse obe slike, vrši poređenje član po član da bi odredila na kojoj od slika je neki piksel više fokusiran, to jeste, da li će se koristiti piksel iz slike A, slike B ili prosjek oba piksela. Prag po kome se poredi, ε *(epsilon)*, računa se adaptivno, za svaku sliku pojedinačno, na osnovu ukupne prosječne varijanse.

2.4. Fuzija slika i čuvanje binarne datoteke

Nakon određivanja maske odlučivanja, može se pristupiti fuziji originalnih slika A i B u novu, fokusiranu sliku. Pomoću funckije za fuziju slika se pravi nova slika, a za svaki piksel se, pomoću maske odlučivanja određuje da li će se u niz piskela upisati piksel iz slike A, ili B, ili će se uzeti njihov prosjek. Nakon završetka ove funckije, dobija se nova, fokusirana slika.

Međutim, u procesu obrade može doći do smanjenja opsega vrijednosti piksela, samim tim, dolazi do gubitka kontrasta. Pomoću algoritma za linaerno histogramsko razvlačenje piksela, vrši se analiza slike i pronalaze maksimalne i minimalne vrijednosti piskela, te se linearno rasteže njihov raspod na [0,255], a ako su svi pikseli jednaki, ova operacija se preskače.

Nakon konačne obrade, pomoću funckije za čuvanje slike, slika se sprema u binarnu datoteku, gdje se prvo upisuju informacije o dimenzijama slike, a zatim se pikseli grupišu u 32-bitne podatke, jer ADSP-21489 zahtjeva da se podaci čitaju i upisuju u 32-bitnim riječima.

2.5. Upravljanje LE diodama

Cilj ovog modula je da se pomoću LE dioda prikaže napredak obrade, to jeste koliko obrade je završilo u određenom trenutku.

Ovaj modul osigurava centralizovanu kontrolu LE dioda na razvojnoj platformi. Inicijalizacijom SRU i postavljanjem flag pinova, osigurava se precizno upravljanje signalima, što je ključno za vizualnu indikaciju statusa različitih operacija unutar projektnog zadatka. Funkcije za uključivanje i isključivanje LE dioda omogućavaju jednostavno korištenje u ostalim dijelovima sistema, dok *Delay_Cycles* osigurava vremenski period potreban za stvaranje vidljivih efekata blinkanja LED dioda.

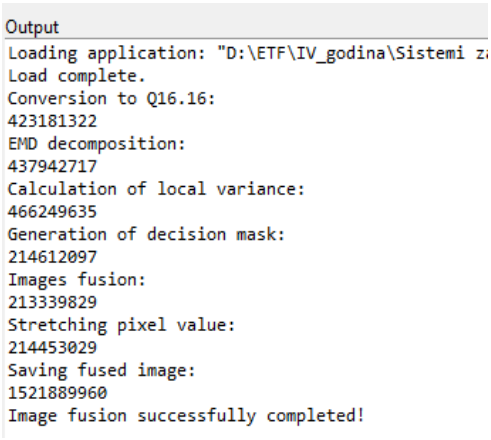
2.6. Postobrada slike i čuvanje slike u željenom formatu

Kada je generisana datoteka *fused_img.bin,* potrebno je pomoću Python skripte generisati sliku u *.bmp* ili *.jpg* formatu. Za to su napisane dvije skripte, *generate_bmp_image.py* i *generate_jpg_image.py*, zavisno od formata u kom se slika čuva. Obe skripte čitaju *fused_img.bin* datoteku u posebnoj funkciji, te čitaju prvih 8 bajtova, kako bi se izvukle širina i visina iz niza. Nakon toga se, zavisno od skripte, radi manipulacija nad bajtovima.

U skripti za pravljenje JPG datoteke, pomoću PIL biblioteke, kreira Image objekt koji od proslijeđenih argumenata (dimenzije i niz bajtova/piksela) pravi objekat slike u grayscale formatu, koji se dalje sprema u JPG datoteku.

U skripti za pravljenje BMP datoteke, prvo se formira BMP zaglavlje, nakon toga se formira grayscale paleta, te se pristupa upisivanju bajtova koju predstavljaju piksele i krajnjem formiranju BMP slike.

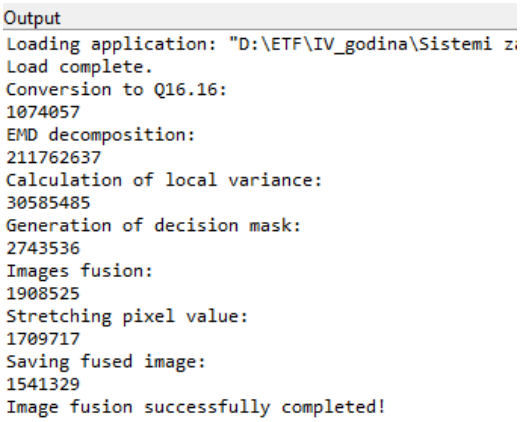
U sklopu izrade projekta, bilo je potrebno i detektovati "bottle neck", te uraditi određene optimizacije koda, pogotovo u tom dijelu. Nakon uključivanja pretprocesorske direktive za brojanje ciklusa i pokretanjem programa dolazi se do sljedećih rezultata *(Slika 2.1)*.



Slika 2.1 – *Ukupan broj procesorskih ciklusa po pozivu svake funckije, bez optimizacija*

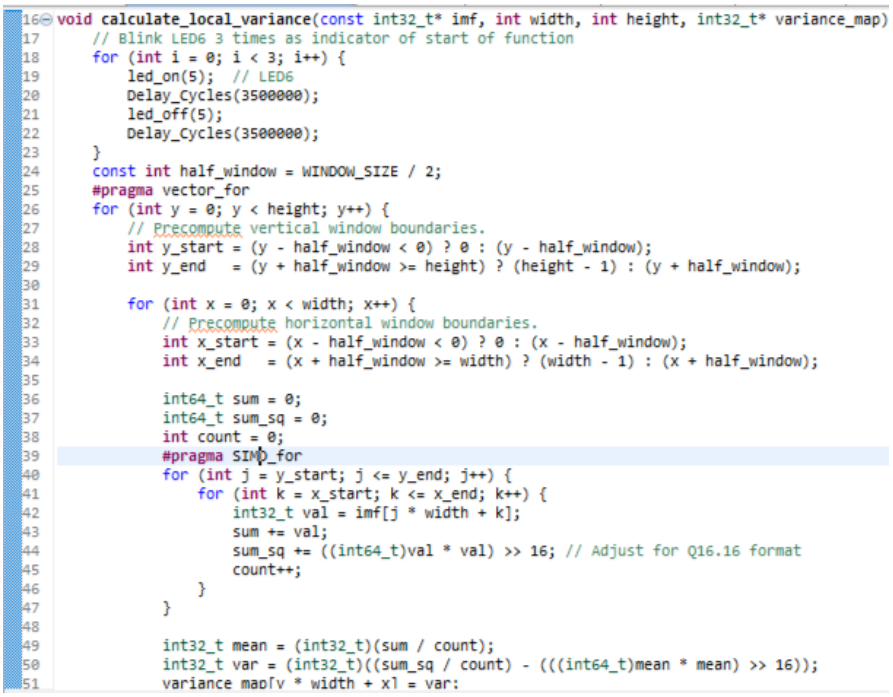
Na slici 2.1 jasno možemo da vidimo da je „bottle neck" zapravo računanje lokalne varijanse, a razlog tome je više ugnježdenih for petlji, dok takođe i funkcija za EMD dekompoziciju troši puno procesorskih ciklusa, jer se u njoj zapravo problem sa tim što koristi dvije for petlje koje prolaze kroz čitav niz pikselaa, i takođe poziva funkciju za interpolaciju, koja takođe koristi dvije for petlje koje prolaze kroz čitav niz.

Nakon uključivanja optimizacije u projektu, i upotrebom pragma direktiva za vektorizaciju petlji, dobijaju se sljedeći rezultati



Slika 2.2 – *Ukupan broj procesorskih ciklusa po pozivu svake funckije, sa optimizacijom*

Na slici 2.2 jasno se vidi da je uključivanje optimizacija i pragma direktiva zaista doprinijelo smanjenju procesorskih ciklusa potrebnih za izvršavanje svih funckija, a naručito funckije za izračunavanje lokalne varijanse i EMD algoritma.



```
16  void calculate_local_variance(const int32_t* imf, int width, int height, int32_t* variance_map)
17      // Blink LED6 3 times as indicator of start of function
18      for (int i = 0; i < 3; i++) {
19          led_on(5);  // LED6
20          Delay_Cycles(3500000);
21          led_off(5);
22          Delay_Cycles(3500000);
23      }
24      const int half_window = WINDOW_SIZE / 2;
25      #pragma vector_for
26      for (int y = 0; y < height; y++) {
27          // Precompute vertical window boundaries.
28          int y_start = (y - half_window < 0) ? 0 : (y - half_window);
29          int y_end   = (y + half_window >= height) ? (height - 1) : (y + half_window);
30
31          for (int x = 0; x < width; x++) {
32              // Precompute horizontal window boundaries.
33              int x_start = (x - half_window < 0) ? 0 : (x - half_window);
34              int x_end   = (x + half_window >= width) ? (width - 1) : (x + half_window);
35
36              int64_t sum = 0;
37              int64_t sum_sq = 0;
38              int count = 0;
39              #pragma SIMD_for
40              for (int j = y_start; j <= y_end; j++) {
41                  for (int k = x_start; k <= x_end; k++) {
42                      int32_t val = imf[j * width + k];
43                      sum += val;
44                      sum_sq += ((int64_t)val * val) >> 16; // Adjust for Q16.16 format
45                      count++;
46                  }
47              }
48
49              int32_t mean = (int32_t)(sum / count);
50              int32_t var = (int32_t)((sum_sq / count) - (((int64_t)mean * mean) >> 16));
51              variance_map[y * width + x] = var;
```

Slika 2.3 – *Funckija za računanje lokalne varijanse, sa uključenim pragma direktivama*

Takođe, u nekim dijelovima koda, upotrebljene su i dodatni načini optimizacije, na primjer, obradom specijalnih slučajeva van for petlje.

Još jedna stvar u sklopu izrade projektnog zadatka je mjerenje vremena izvršavanja značajnijih funkcija koje je prikazano na slici ispod.

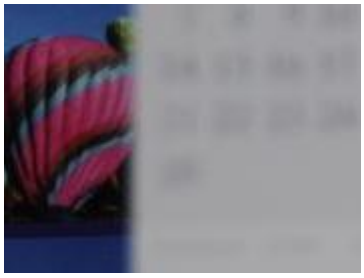
```
Loading application: "D:\ETF\IV_godina\Sistemi za digitalnu obradu signal
Load complete.
Conversion to Q16.16 time taken is           1.074062e+06 seconds
EMD decomposition time taken is              2.117626e+08 seconds
Calculation of local variance time taken is  3.058549e+07 seconds
Generation of decision mask time taken is    2.743537e+06 seconds
Images fusion time taken is                  1.908532e+06 seconds
Stretching pixel value time taken is         1.709717e+06 seconds
Saving fused image time taken is              1.541347e+06 seconds
Image fusion successfully completed!
```

Slika 2.4 – *Vremena izvršavanja svih bitnih funkcija*

Na osnovu ovih vrijednosti, može se vidjeti da EMD dekompozicija traje najduže, a računanje lokalne varijanse je također značajno duže od ostalih vremena. Ovi rezultati se odnose na kod nakon optimizacije, te se i po tome koliko koja funckija zahtjeva procesorskih ciklusa moglo zaključiti i koja će funckija trebati najduže vrijeme izvršavanja.

Nakon završetka programa, očekivano je da se od dvije multifokusirane slike, dobije jedna u potpunom fokusu, a primjer rada je prikazan na slikama ispod.

Slika 2.5 – *Slika A*



Slika 2.6 – Slika B

Slike 2.5 i 2.6 su originalne slike, koje se pretvaraju u grayscale, te prolaze kroz EMD algoritam, gdje se obrađuju, te se dobijaju sljedeći rezultati.

Slika 2.7 – *Slika F, slika nakon fuzije, bez histogramskog razvlačenja vrijednosti piksela*

Slika 2.8 – *Slika F, slika nakon fuzije, sa histogramskim razvlačenjem vrijednosti piksela*

Na osnovu originalnih slika i dobijenih rezultata, može se vidjeti da se kvalitet slike izgubio, ali da su uspješno pronađeni fokusirani dijelovi na obe slike. Slika 2.7 je slika bez obrade odmah nakon fuzije, gdje se vidi da je slika izgubila dio kontrasta koji su originalne slike imale. Na slici 2.8, razvlačenjem vrijednosti piksela, vidi se da slika posjeduje mnogo veći kontrast i da se brojevi vide jasnije nego na slici 2.7, ali da još uvijek ta slika nema potpuni kvalitet originalne slike.

3. Zaključak

U okviru ovog projektnog zadatka uspješno je realizovan sistem za generisanje empirijske vremensko-frekvencijske dekompozicije signala (EMD) i primjena istog na fuziju multifokusiranih slika. Kroz implementaciju EMD algoritma, izdvojen je prvi IMF, što je omogućilo precizno mapiranje frekvencijskog sadržaja ulaznih signala, a time i selektivno fuziranje slika na osnovu lokalne varijanse. Iako je inicijalna fuzija rezultovala određenim gubitkom kontrasta, primjenom histogramskog razvlačenja uspješno je obnovljen vizuelni kvalitet, čime je postignut kompromis između detekcije fokusiranih dijelova i očuvanja kontrasta.

Daljnja optimizacija sistema, posebno kroz vektorizaciju ključnih petlji pomoću pragma direktiva, značajno je smanjila procesorske cikluse, što potvrđuje da su identifikovani "bottle neck" dijelovi – naročito kod izračunavanja lokalne varijanse i EMD dekompozicije – uspješno unaprijeđeni. Time je ostvarena efikasnija implementacija koja omogućava bržu obradu velikih nizova podataka, što je ključno za real-time primjene.

Glavno ograničenje ovog koda je što je veličina slike ograničena na 200x200 piksela, što je realtivno mala slika. Ovo ograničenje je posljedica male interne, kao i eksterne memorije na samoj platformi, što onemogućuje rad sa većom količinom podataka, u ovom slučaju, većom količinom piskela.

4. Literatura

- **Analog Devices.** (Rev. 1.1). *ADSP-21489 ezBoard User Guide*. https://www.analog.com/media/en/technical-documentation/user-guides/ADSP-21489_ezboard_man_rev.1.1.pdf
- **Analog Devices.** (Rev. 1.5). *CCES2-2.0 SHARC Compiler Manual*. https://www.analog.com/media/en/dsp-documentation/software-manuals/cces2-2-0_SharcCompiler_mn_rev1-5.pdf#page=195&zoom=100,0,776
- **Analog Devices.** (n.d.). *CCES SHARC Library Manual*. https://www.analog.com/media/en/dsp-documentation/software-manuals/cces-SharcLibrary-manual.pdf
- **Analog Devices.** (Rev. 1.1). *ADSP-214xx Hardware Reference Manual*. https://www.analog.com/media/en/dsp-documentation/processor-manuals/adsp-214xx_hwr_rev1.1.pdf
- *emd for sale.* (n.d.). GitHub repozitorij. https://github.com/emdforsale/emd
- *Empirical Mode Decomposition – an introduction.* (n.d.). ResearchGate. https://www.researchgate.net/publication/221534245_Empirical_Mode_Decomposition_-_an_introduction
- *Multidimensional Empirical Mode Decomposition.* (n.d.). Wikipedia. https://en.wikipedia.org/wiki/Multidimensional_empirical_mode_decomposition