## mode\_pred.cpp

Manipuliranje vektorima pomaka. Sadrži sve potrebne funkcije za rekonstrukciju vektora pomaka za trenutni frame. Sastoji se od funkcija potrebnih za predviđanje vektora pomaka te funkcije za konačnu rekonstrukciju na temelju predviđenih vektora pomaka i predane razlike MVd.

### Globalne varijable i strukture podataka

int \*\*\*mvL0x;

int \*\*\*mvL0y;

int \*subMvCnt;

int \*refIdxL0;

**\*\*\*mvL0x i \*\*\*mvL0y**: rekonstruirani vektori pomaka za sve blokove trenutne slike. Varijable su globalne jer se koriste i za vrijeme određivanja predviđanja u koderu, gdje je moguće direktno postaviti određeni vektor pomaka.

**\*subMvCnt**: broj vektora pomaka za svaki makroblok.

**\*refIdxL0**: referentna slika za svaki makroblok. U baseline profilu H.264 je ova vrijednost 0 (prethodna slika) za sve blokove.

**void AllocateMemory()**

Funkcija priprema memoriju za globalne varijable mode\_pred-a. Potrebno ju je pozvati prilikom inicijalizacije koder i dekodera.

**void get\_neighbour\_mv(int mbAddrN, int mbPartIdx, int \* mvNx, int \* mvNy, int \* refIdxL0N)**

**mbAddrN**: makroblok od kojeg je potrebno dobiti vektor pomaka.

**mbPartIdx**: makroblok od kojeg je potrebno dobiti vektor pomaka.

Funkcija preko mvNx i mvNy parametara vraća vektor pomaka za podblok mbPartIdx makrobloka mbAddrN. Dodatno, vraća i koja je referentna slika (uvijek 0).

**void DeriveNeighbourLocation(int xN, int yN, int \* mbAddrN, int \* xW, int \* yW, bool \* validN)**

**xN**: udaljenost po x koordinati od gornjeg lijevog kuta trenutnog makrobloka.

**yN**: udaljenost po y koordinati od gornjeg lijevog kuta trenutnog makrobloka

**\*mbAddrN**: povratna vrijednost koja određuju rezultantni makroblok.

**\*xW, \*yW**: pozicija unutar rezultantnog makrobloka.

**\*validN**: povratna vrijednost koja određuju dali je rezultatni makroblok valjan.

Funkcija vraća unutar kojeg makrobloka (ako je valjan) i na kojem mjestu tog makrobloka se nalazi točka koja je udaljena za [xN, yN] od gornjeg lijevog kuta trenutnog makrobloka.

**void DeriveNeighbourPartitions(int mbPartIdx, int subMbPartIdx,**

**int \* mbAddrA, int \* mbPartIdxA, int \* subMbPartIdxA, bool \* validA,**

**int \* mbAddrB, int \* mbPartIdxB, int \* subMbPartIdxB, bool \* validB,**

**int \* mbAddrC, int \* mbPartIdxC, int \* subMbPartIdxC, bool \* validC,**

**int \* mbAddrD, int \* mbPartIdxD, int \* subMbPartIdxD, bool \* validD)**

**mbPartIdx:** redni broj podbloka unutar trenutnog makrobloka.

**subMbPartIdx**: redni broj dijela podbloka

**mbAddrN, mbPartIdxN, subMbPartIdxN, validN:** identifikacija susjednog dijela (N se zamjenjuje sa A, B, C ili D) s rednim brojem makrobloka unutar slike (mbAddr), rednim brojem podbloka (mbPartIdx), rednim brojem podjele na 4x4 (subMbPartIdx), jeli taj dio valjan (valid).

Funkcija identificira susjede za dio trenutnog bloka (koji može biti bilo koji 8x8 dio, ili 4x4). Ukoliko neki susjedni blok nije dostupan (nalazi se izvan okvira slike), označava se flag Valid sa false. Ova funkcija se koristi kod rekonstrukcije vektora pomaka jer se predviđeni vektori pomaka dobivaju na temelju vektora pomaka od susjeda. Kako je dekoder sastavni dio kodera, funkcija se koristi u oba.

**void PredictMV\_LumaSubMB(int mbPartIdx, int subMbPartIdx)**

**mbPartIdx:** određuje podblok trenutnog makrobloka (veličine 8x8).

**subMbPartIdx:** određuje dodatnu podjelu podbloka.

Funkcija predviđa vektor pomaka za određeni dio trenutnog makrobloka i zapisuje ga u polje mvL0x, odnosno mvL0y. Za predviđanje vektora pomaka koriste se vektori pomaka susjednih blokova dohvaćeni funkcijom **DeriveNeighbourPartitions**. Ukoliko nije dostupan susjed C, tada se umjesto njegovih vektora pomaka koriste vektori pomaka od susjeda D. Nadalje, ako je dostupno manje od tri vektora pomaka i nakon moguće potrebne zamjene susjeda C sa D, tada se svim nedostupnim susjedima postavljaju vektori pomaka prvog dostupnog (to je obično A ili B). Nakon ovog koraka, konačni predviđeni vektor pomaka za dio trenutnog makrobloka, određenog sa mbPartIdx i subMbPartIdx, se postavlja na srednju vrijednost vektora pomaka susjeda A, B i C.

**void PredictMV\_Luma(int mbPartIdx)**

**mbPartIdx:** dio trenutnog makrobloka.

Funkcija predviđa vektor pomaka za dio trenutnog makrobloka te se zapisuje u [mvL0x, mvL0y]. Ako je neki dio tipa P\_L0\_8x8, tada se dodatno za njegove dijelove poziva funkcija **PredictMV\_LumaSubMB** kako bi se predvidjeli vektori pomaka i za sitnije dijelove.

**void PredictMV()**

Funkcija predviđa vektore pomaka za trenutni makroblok pozivima funkcije **PredictMV\_Luma.** Ako je trenutni makroblok predviđan sa P\_Skip predikcijom, tada se dodatno postavlja i vrijednost razlike vektora pomaka na 0 (mvd\_l0 = 0). Kako se vektori pomaka računaju na temelju susjednih blokova, koji mogu biti i unutar trenutnog makrobloka, nakon predviđanja pokreta na pojedini blok, automatski se i pridoda razlika mvd\_l0.

**void DeriveMVs()**

Funkcija poziva funkciju za predviđanje vektora pomaka te ako je riječ o većoj podjeli trenutnog makrobloka, prepisuje vektore pomaka na podblokove (kako bi se kasnije olakšala rekonstrukcija slike i drugih vektora pomaka).

**mocomp.cpp**

Rekonstrukcija slike na temelju referentne i vektora pomaka. Ovdje se nalaze funkcije za interpoliranje vrijednosti na necjelobrojnim lokacijama slike, određivanje predviđenih makroblokova i konačne rekonstrukcije (predviđeno + ostatak).

**void FillTemp\_4x4\_refPart(frame\_type \*ref, int org\_Lx, int org\_Ly, int org\_Cx, int org\_Cy)**

**\*ref:** referentna slika.

**org\_Lx, org\_Ly:** lokacija luminancije u referentnoj slici.

**org\_Cx, org\_Cy:** lokacija boje u referentnoj slici.

Funkcija iz referentne slike dohvaća podatke potrebne za interpoliranje vrijednosti na necjelobrojnim pozicijama, a koji se nalaze unutar 4x4 bloka s gornjim lijevim kutom na poziciji org\_Lx, org\_Ly. Potrebni podaci se spremaju u nizove L\_Temp\_4x4\_refPart i C\_Temp\_4x4\_refPart.

**Tap6Filter(E,F,G,H,I,J) Border(((E)-5\*(F)+20\*(G)+20\*(H)-5\*(I)+(J)+16)>>5)**

Ova funkcija interpolira vrijednost na sredini pravca koji je redom definiran točkama E, F, G, H, I i J. Točnije, interpolira vrijednosti na sredini dužine GH.

**int L\_MC\_frac\_interpol(int \*data, int frac)**

**data:** pokazivač na niz L\_Temp\_4x4\_refPart.

**frac:** oznaka za pomak do četvrtine piksela po x i y koordinati.

Funkcija na temelju podataka L\_Temp\_4x4\_refPart i funkcije za interpolaciju (Tap6Filter) određuje vrijednost određene pozicije unutar bloka 4x4. Funkcija kad izračuna potrebnu vrijednost odmah prekida iz razloga što je kod rekonstruiranja za sve elemente bloka 4x4 potrebno odrediti samo određenu interpolaciju.

**void FillInterpolSubMBPart(int predL[16][16][16], int sx, int sy, int \*data)**

**predL:** parametar za povratnu vrijednost.

**sx, sy:** piksel kod kojeg je potrebno obaviti interpolaciju svih 16 vrijednosti.

**data:** pripremljeni podaci za moguću interpolaciju vrijednosti bloka 4x4.

Funkcija radi isto što i **L\_MC\_frac\_interpol**, samo za razliku od nje ne prekida pri dolasku do određene pozicije za interpoliranje, već redom interpolira vrijednost za 4x4 međupikselne pozicije kojima je gornji lijevi kut određen sa **sx, sy.**

**void FillInterpolatedMB(int predL[16][16][16], int predCr[16][8][8], int predCb[16][8][8], frame\_type \*refPic)**

**predL, predCr, predCb:** izlazni parametri za 16 interpolirajućih vrijednosti kod svakog piksela trenutnog makrobloka (veličine 16x16).

**\*refPic:** referentna slika.

Funkcija se koristi kod kodera za potrebe prvotnog punjenja niza svih interpolirajućih vrijednosti referentne slike, a različita je od standarne takve funkcije kod dekodera iz razloga što puni za sve vektore pomaka, tj. određuju se interpolirajuće vrijednosti na svim mogućim lokacijama unutar referentne slike.

**void MotionCompensateSubMBPart(int predL[16][16], int predCr[8][8], int predCb[8][8], frame\_type \*refPic)**

**predL, predCr, predCb:** izlazni parametri za predviđene vrijednosti kod svakog piksela trenutnog makrobloka (veličine 16x16) na temelju mvL0x i mvL0y.

**\*refPic:** referentna slika.

Funkcija koristi globalna polja mvL0x i mvL0y, te na temelju vektora pomaka iz tih polja predviđa vrijednosti trenutnog makrobloka. Ako je potrebno, obavlja interpolaciju luminancije prema funkciji **L\_MC\_frac\_interpol,** a za boju obavlja linearnu interpolaciju (linearna interpolacija između dva susjedna piksela).

**void Decode(int predL[16][16], int predCr[8][8], int predCb[8][8])**

**predL, predCr, predCb:** izlazni parametri za rezultantne predviđene vrijednosti kod svakog piksela trenutnog makrobloka (veličine 16x16).

Funkcija pozivima **MotionCompensateSubMBPart** za sve blokove veličine 4x4 unutar trenutnog makrobloka puni konačne predviđene vrijednosti za trenutni makroblok. Ova funkcija se poziva i kod kodera i dekodera, a namjerno je postavljeno posebno kako bi program kodiranja/dekodiranja bio čitljiviji pozivom samo funkcije Decode, a **MotionCompensateSubMBPart** za svaki dio. Stvarne vrijednosti piksela trenutnog makrobloka se dobivaju tek kasnije kad se predviđenim vrijednostima pridodan i ostatak (globalno **residual**).

**Moestimation.cpp**

Ovdje se nalazi skup funkcija kodera H.264 za međuslikovno predviđanje.

### Globalne varijable i strukture podataka

frame\_type refFrameInterpolated[16];

int MAXDIFF = 2;

int \*\*refFrameKar[5][16];

int \*sortedSuma0[5];

**refFrameInterpolated[16]**: sve potrebne interpolirajuće vrijednosti referentne slike za predviđanje trenutne. Po svakom pikselu postoji 16 interpolirajućih vrijednosti jer su mogući pomaci za četvrtinu piksela.

**MAXDIFF**: dopuštena razlika predviđenih piksela u odnosu na stvarne. Ova varijabla se uglavnom koristi za određivanje može li se trenutni makroblok predviđati P\_Skip tipom predikcije. Ova varijabla ima izuzetnu važnost jer dok povećava broj makroblokova koji su P\_Skip predviđeni i time povećava kompresiju, degradira kvalitetu kasnije rekonstrukcije.

**\*\*refFrameKar[5][16]**: pet karakteristika potrebnih za određivanje potencijalno najboljih vektora pomaka. Karakteristike se računaju za sve moguće referente pozicije, dakle ima ih 16 puta više nego piksela u slici.

**\*sortedSuma0[5]**: nizovi potrebni za prvo filtriranje velikih vektora pomaka za trenutni blok veličine 8x8. U ovom nizu se nalaze vektori pomaka sortirani prema sumi luminantnih vrijednosti.

**void InitializeInterpolatedRefFrame()**

Funkcija alocira potrebnu memoriju za globalna polja koja koristi koder za međuslikovno predviđanje. Koristi se samo u koderu i potrebnu ju je pozvati prilikom inicijalizacije kodera.

**void FillInterpolatedRefFrame()**

Ova funkcija se koristi za punjenje nizova **refFrameInterpolated, refFrameKar, sortedSuma0.** Poziva se nakon obrade pojedine slike kako bi se pripremili svi potrebni podaci za predviđanje iduće slike. **refFrameInterpolated** se puni pozivanjem **FillInterpolSubMBPart** za svaki makroblok slike. Unutar ove funkcije su implementirane dvije optimizacije – linearno punjenje **refFrameKar** polja te linearno sortiranje pozicija u polje **sortedSuma0**. Linearno punjenje **refFrameKar** polja je izvedeno dvama prolazima kroz polje **refFrameInterpolated** pri čemu se u prvom prolazu na svaku poziciju zapiše suma luminantnih komponenti svih piksela dolje-desno od trenutnog, a drugom prolazu se na temelju tih vrijednosti formulama određuju vrijednosti pojedinih karakteristika. Linearno sortiranje u polju **sortedSuma0** je napravljeno u odnosu na sumu luminantnih vrijednosti piksela u 8x8 bloku korištenjem counting sorta. Counting sort je moguć u ovom slučaju jer suma luminantnih vrijednosti za pojedini blok može poprimiti vrijednosti u intervalu [0, 16384].

**int sadLuma8x8MVs(int mvx, int mvy, int luma8x8BlkIdx)**

**mvx, mvy**: pomak za koji je potrebno izračunati SAD (*Sum of Absolute Differences*) u odnosu na trenutni blok veličine 8x8.

**luma8x8BlkIdx:** redni broj 8x8 bloka unutar trenutnog makrobloka za koji se traži SAD funkcija.

Funkcija vraća SAD između trenutnog bloka 8x8 i predviđenog vektorom pomaka **mvx, mvy**. Funkcija za određivanje SAD koristi za predviđeni blok ranije napunjen niz **refFrameInterpolated.**

**int ExactPixels(int predL[16][16])**

**predL:** predviđeni makroblok.

Funkcija vraća broj piksela koji su jednaki ili skoro jednaki u trenutnom makrobloku i predviđenom ukoliko se pozove **Decode()** s definiranim P\_Skip predviđanjem. Ova funkcija se koristi kod određivanja hoće li se za trenutni makroblok koristiti P\_Skip predviđanjem i na taj način maksimalno povećati kompresija.

**void PopraviPSkip(int predL[16][16])**

**predL:** predviđeni makroblok.

Funkcija popravlja trenutni blok na temelju predviđenog ukoliko se koristi P\_Skip predviđanje. To je nužno jer se kod P\_Skip predviđanja najčešće dopuštaju odmaci od stvarnog makrobloka, a kako se greška ne bi propagirala na naredne slike, potrebno je prepraviti trenutni makroblok kako bi izgledao identično onom predviđenom kasnije u dekoderu.

**void MEstimation(int sx, int sy, int granica, int stepMV, int stepFrac, int px, int py)**

**sx, sy:** početna pozicija bloka 8x8.

**px, py:** predviđeni vektor pomaka.

**granica:** mogući odmak vektora pomaka od predviđenog.

**stepMV:** svakih koliko piksela unutar granice provjeriti dobrotu vektora pomaka.

**stepFrac:** svakih koliko interpolirajućih vektora pomaka isprobati. Moguće isprobati samo 16 po svakom pikselu, pa se obično ovaj parametar postavlja na 1 ili na 16 (sve interpolirajuće isprobati, probati samo cjelobrojni pomak).

Ovo je funkcija koja u globalno polje potencijalno najboljih vektora pomaka (**bmin**, **bxs, bys**) postavlja N najboljih iz skupa određenog parametrima pri pozivu funkcije. Funkcija za odabir njih N koristi ranije definirane i izračunate karakteristike blokova 8x8.

**void basicInterEncoding(int predL[16][16], int predCr[8][8], int predCb[8][8])**

**predL, predCr, predCb:** povratne vrijednosti predstavljaju predviđeni makroblok.

Funkcija koja obavlja osnovno međuslikovno predviđanje korištenjem potpunog pretraživanja (isprobava redom sve pomake unutar zadanog prozora pretraživanja). Svaki od vektora pomaka se isprobava na način da se prvo on direktno postavi kao konačan u **mvL0x** i **mvL0y** te zatim pozove **Decode()** funkcija. Po izvršavanju **Decode()** funkcije, uspoređuje se **predL** sa trenutnim makroblokom i ono predviđanje koje ima najmanje razlike (najmanja **SAD** funkcija) se postavlja kao stvarno. Dodatno, na početku se provjerava može li trenutni blok proći kao P\_Skip predviđanje. Taj dio je jako bitan za međuslikovno predviđanje, pa se ni u osnovnoj izvedbi ne izostavlja.

**void interEncoding(int predL[16][16], int predCr[8][8], int predCb[8][8])**

**predL, predCr, predCb:** povratne vrijednosti predstavljaju predviđeni makroblok.

Kao i **basicInterEncoding** funkcija započinje provjerom može li trenutni blok proći kao P\_Skip predviđanje. Ako može, tada se tip predviđanja postavlja na P\_Skip i poziva **PopraviPSkip** čime završava određivanje međuslikovno predviđanje za trenutni makroblok. Ako to nije slučaj, nastavlja se odabir predviđanjima za svaki 8x8 podblok. Određuju se potencijalno najbolji pomaci po tri karakteristična tipa pomaka: jako kratki neovisni o susjedima, mali odmaci od susjednih pomaka, daleki pomaci različiti od susjednih. Prva dva tipa se rješavaju vrlo jednostavno pozivanjem **MEstimation** funkcije s time da se kao ciljni vektor pomaka postavlja MVp u jednom ili 0 drugom slučaju pri čemu se dobiva određen broj potencijalno najboljih pomaka. Zadnji tip pomaka se ne može efikasno riješiti jednostavnim pozivom **MEstimation** funkcije jer je potrebno provjeriti veliki broj pomaka (najbolji pomak može biti prema bilo kojoj poziciji na referentnoj slici). U tu svrhu se koristi polje **sortedSuma0** za prvotno filtriranje svih pozicija na referentnoj slici po ukupnoj sumi luminantnih vrijednosti, na način da se dopušta prosječno odstupanje za 3 po pikselima. Odnosno, dozvoljeno odstupanje ukupne sume luminantnih vrijednosti za trenutni blok je 180. Dodatno, ne uzimaju sve svi ti pomaci kao potencijalni nego samo oni kojima lijeva i desna polovica odnosno gornja i donja polovica imaju prosječno odstupanje po pikselu za luminantne komponente također do 3. Time se filtrira na predviđene blokove i po obliku slične trenutnom. Ovo filtriranje se radi po cjelobrojnim pomacima te se za određen broj tako najboljih cjelobrojnih pomaka poziva **MEstimation** funkcija malim prozorom pretraživanja za finije pomake. Od potencijalnih pomaka za sva tri tipa predviđanja se temeljem funkcije **sadLuma8x8MVs** određuje najbolji pomak. Jednom određeni pomaci za svaki 8x8 blok, se po potrebi spajaju u veće cjeline čime se dobiva i konačna podjela trenutnog makrobloka na 16x16, 16x8, 8x16 ili 8x8 veličine.