

Contents

1	Uvod	3
1.1	Općenito o video sekvencama	3
1.1.1	Video fajlovi	3
1.2	Definicija opsega rada	3
1.3	Primjene video interpolacije	3
2	Osnovne tehnike	4
2.1	Duplikacija frejmova	4
2.2	Linearna interpolacija	4
2.3	Područja primjene	4
3	Algoritmi uparivanja blokova	5
3.1	Uvod	5
3.2	Osnovni algoritmi	7
3.2.1	Potpuna pretraga	7
3.2.2	Trostepena pretraga	8
3.2.3	Četverostepena pretraga	9
3.2.4	Dijamantna pretraga	9
3.3	Adaptive Rood Pattern Search - ARPS	9
3.4	Fazna korelacija	9
4	Uklanjanje grešaka	10
5	Interpolacija frejmova korištenjem optičkog toka	11
6	Paralelno izvršavanje tehnika interpolacije	12
7	Implementacija interpolatora korištenjem OpenCV biblioteke i benchmark testovi	13

Uvod

1.1 Općenito o video sekvencama

1.1.1 Video fajlovi

1.2 Definicija opsega rada

1.3 Primjene video interpolacije

Osnovne tehnike

2.1 Duplikacija frejmova

2.2 Linearna interpolacija

2.3 Područja primjene

Algoritmi uparivanja blokova

3.1 Uvod

Prva klasa algoritama koje ćemo proučavati kreću od iste osnovne ideje: Podijeliti prvi frejm na blokove, za svaki blok pronaći vektor pomaka iz prvog u drugi frejm, te primijeniti jedan dio tog vektora pomaka na blok. Ako nam je cilj samo kreirati jedan novi frejm između svaka dva postojeća, svaki blok ćemo pomjeriti duž pola izračunatog vektora pomaka. Ako nam je cilj interpolirati dva frejma između dva postojeća, blok ćemo pomjeriti duž jedne trećine vektora pomaka za prvi, i dvije trećine za drugi interpolirani frejm, itd. Cilj slijedećih algoritama jeste uparivanje blokova prvog frejma sa blokom iste veličine u drugom frejmu. Međutim, postoji nekoliko pitanja na koja moramo odgovoriti prije nego što možemo primijeniti ove algoritme:

- Koju veličinu bloka ćemo koristiti?
- Koliki će biti prozor pretrage, odnosno koliko će se svaki blok moći maksimalno pomjeriti između prvog i drugog frejma?
- Koji je kriterij sličnosti dva bloka?
- Kako odrediti uspješnost uparivanja?

U praksi se koriste blokovi veličine 16x16 piksela, te prozor pretrage veličine 30x30 piksela. To znači da pretpostavljamo da se između dva susjedna frejma blokovi neće pomjeriti više od 7 piksela u bilo kojem od 4 kardinalna smjera. To nam daje 225 mogućih lokacija za svaki blok. Naravno, ne postoji definitivna, optimalna veličina bloka ili prozora pretrage za sve slučajeve. Manje blokove je brže uporediti, ali je njihov broj veći, te je veća vjerovatnoća da će dva bloka biti slučajno veoma slična. Veći prozor pretrage

nam omogućava pronalaženje ispravnih vektora pomaka i u slučaju kada se desi pomak veći od 7 piksela, ali značajno povećava vrijeme potrebno za izračunavanje te, slično kao u slučaju blokova, povećanjem prozora pretrage se povećava i vjerojatnoća uparivanja dva bloka koji su slični, ali zapravo ne predstavljaju isti blok. U svim slijedećim algoritmima ćemo koristiti blokove i prozore pretrage navedene veličine.

Slijedeće pitanje se tiče kriterija sličnosti dva bloka. Svaki blok je sastavljen od 256 piksela, koji se sastoje od 3 komponente: crvene, zelene, i plave. Svaka komponenta ima cjelobrojnu jačinu u rasponu od 0 do 255, uključivo. Za upoređivanje blokova se prvo pikseli pretvore u crno-bijele, sa jednom komponentom koja predstavlja jačinu bijele boje piksela. Korišteni kriteriji sličnosti blokova su veoma jednostavni. Jedan je *Mean Absolute Difference (MAD)*, odnosno *srednja apsolutna razlika*. Ova mjera nije ništa drugo nego suma apsolutnih vrijednosti razlika jačina odgovarajućih piksela u blokovima, podijeljena sa veličinom bloka. Drugim riječima, zadana je formulom

$$MAD = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |A_{ij} - B_{ij}|$$

Pri čemu N predstavlja visinu i širinu bloka (u našem slučaju 16), dok A_{ij} i B_{ij} predstavljaju vrijednosti piksela na koordinatama i, j (sa početkom u gornjem lijevom uglu bloka) prvog, odnosno drugog razmatranog bloka.

Druga, veoma slična mjera jeste *Mean Squared Error (MSE)*, odnosno *srednji kvadrat greške*. Umjesto uzimanja apsolutne vrijednosti razlika piksela, ova mjera kvadrira razlike piksela, čime se više kažnjavaju veće razlike. *MSE* je zadana formulom

$$MSE = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (A_{ij} - B_{ij})^2$$

Mi ćemo ove funkcije ubuduće zvati zajedničkim imenom *funkcije cijene*. Cilj svih algoritama u ovom poglavlju jeste minimizacija funkcije cijene, bilo *MAD* ili *MSE*.

Još jedno veoma važno područje primjene algoritama uparivanja blokova (i zapravo područje gdje se najviše primjenjuju) jeste kompresija video sekvenci. U dijelu vezanom za osnovne algoritme i ARPS, obrađuju se algoritmi koji su korišteni u standardima H.261, H.262 i H.263. U novijim standardima (pri čemu je najnoviji H.265, čija je prva verzija izašla 2013. godine) korišteni su napredniji algoritmi, koji u ovom radu neće biti obrađeni.

3.2 Osnovni algoritmi

Sada ćemo se osvrnuti na neke od osnovnih algoritama uparivanja blokova. Pretpostavit ćemo da su blokovi veličine 16×16 piksela, a prozori pretrage veličine 30×30 piksela, te da su svi pikseli monohromatski (crno-bijeli). Od svih slijedećih algoritama, samo prvi pronalazi optimalno uparivanje. Svi ostali ostali algoritmi su aproksimacije. Preciznije rečeno, postoji 225 mogućih lokacija gdje se početni blok može nalaziti u prozoru pretrage. Definišimo matricu $C_{15 \times 15}$, pri čemu svakom elementu matrice C odgovara vrijednost funkcije cijene (MAD ili MSE) koju dobijemo ako postavimo blok na to mjesto u prozoru pretrage (drugim riječima, elementu $1,1$ odgovara vrijednost funkcije cijene koju dobijemo ako blok postavimo u gornji lijevi ugao prozora pretrage, pomjeranjem bloka dobijamo druge vrijednosti matrice). Aproksimativni algoritmi pretpostavljaju da ova matrica ima jednu najmanju (optimalnu) vrijednost, i da vrijednosti elemenata matrice monotonno rastu kako se udaljavamo od ovog elementa. Kontinualni analogon ove osobine bi bila funkcija 2 realne promjenljive koja ima samo jedan lokalni minimum, koji je ujedno i globalni minimum. Za funkciju koja ima ovu osobinu kažemo da je *unimodalna*. U slučaju kad ova osobina postoji, možemo pronaći minimalnu vrijednost matrice jednostavnim spuštanjem po gradijentu, odnosno, počevši od proizvoljnog početnog elementa, u svakom koraku pređemo na bilo koji od manjih elemenata dok ne dođemo do nekog koji je manji od svih svojih susjednih elemenata.

Naravno, ne postoji ništa da nam garantuje ovu osobinu, što znači da će aproksimativni algoritmi samo u rijetkim slučajevima pronaći optimalno rješenje.

3.2.1 Potpuna pretraga

Ovaj algoritam se zasniva na potpunom pretraživanju svih mogućih lokacija za blok, te pronalaženju lokacije koja daje najmanju vrijednost funkcije cijene. Postoji 225 mogućih lokacija za blok unutar prozora pretrage, a za izračunavanje vrijednosti funkcije cijene moramo uporediti $16 \times 16 = 256$ piksela. To nam daje $225 \times 256 = 57600$ upoređivanja piksela po bloku. HD video (dimenzija 1280×720 piksela) sadrži frejmove koji se sastoje od 3600 blokova, što nam ukupno daje preko 200 miliona upoređivanja piksela za svaki frejm video sekvence. Tako da nije teško vidjeti zašto se ovaj algoritam ne koristi u praksi.

3.2.2 Trostepena pretraga

Trostepena pretraga se oslanja na osobinu unimodalnosti matrice vrijednosti funkcije cijene. Ovaj algoritam je ovdje korišten isključivo kao uvod, jer ostvaruje veoma slabe rezultate u praksi. Algoritam je mnogo brži od potpune pretrage te koristi slične tehnike kao bolji algoritmi koji će biti obrađeni u narednim dijelovima.

Neka je zadan prozor pretrage veličine 30×30 piksela i blok veličine 16×16 piksela. Trostepena pretraga radi tako što u svakom koraku izračuna funkciju cijene bloka na 9 lokacija u prozoru pretrage. Tih 9 lokacija su trenutni centar i sve kombinacije lokacija koje dobijemo ako blok pomjerimo za $-S$, 0 ili S piksela po x i y osama (postoji 9 kombinacija, od kojih je jedna $(0, 0)$, što predstavlja naš centar). S predstavlja trenutnu *veličinu koraka*. Nakon izračunavanja funkcije cijene na svih 9 lokacija, izaberemo najbolju (tj. lokaciju koja daje najmanju vrijednost funkcije cijene), nju odredimo za novi centar, i smanjimo S na jednu polovinu trenutne vrijednosti. U prvom koraku, S će imati vrijednost 4, u drugom 2, a u trećem 1, dok će centar u prvom koraku predstavljati stvarni centar prozora pretrage. S obzirom da se na početku blok nalazi 7 piksela daleko od ivica prozora pretrage, a u 3 koraka (sa pomakom od 4, 2 i 1 pikselom) se moguće pomjeriti najviše tačno 7 piksela, blok može zauzeti bilo koju poziciju u prozoru pretrage.

U svakom od 3 koraka, imamo 9 različitih mogućih pozicija bloka, tako da izgleda da moramo izračunati funkciju cijene bloka 27 puta. Međutim, centar u drugom i trećem koraku je već izračunat u prethodnom koraku, tako da je zapravo potrebno izračunati samo 25 različitih funkcija pretrage, što je 9 puta manje od 225 kod potpune pretrage, što naravno predstavlja veoma značajno ubrzanje. Međutim, kao što je prije rečeno, ovaj algoritam generalno ne izračunava zadovoljavajuće vektore pomaka, tako da se u praksi i ne koristi.

3.2.3 Četverostepena pretraga

3.2.4 Dijamantna pretraga

3.3 Adaptive Rood Pattern Search - ARPS

3.4 Fazna korelacija

Uklanjanje grešaka

Interpolacija frejmova korištenjem optičkog toka

Paralelno izvršavanje tehnika interpolacije

Implementacija interpolatora korištenjem OpenCV biblioteke i benchmark testovi

Zaključak