Morphing – interpolacija niza slika između slika dvaju osoba

Karlo Koščević, Lovro Magaš, Mihael Marović, Luka Papež, Josip Tomić

June 4, 2017

Sažetak

U radu se obrađuje metoda morphinga, tj. interpolacije između dviju slika lica. Kroz rad se opisuje ideja morphinga, njena povijest i uporabe u svijetu. Također, objašnjene su razne metode rješavanja ovog problema te je na kraju detaljno iznesena jedna od mogućih implementacija temeljena na Delaunay triangulaciji.

1. Uvod

Interpolacija digitalne slike općeniti je izraz za skupinu metoda kojima se žele procijeniti vrijednosti pojedinih elemenata na digitalnoj slici. Naravno, kao i u mnogim granama obrade slike, tehnike interpolacije uvelike ovise o kontekstu problema u kojem se koriste. Jedna od najčešćih upotreba interpolacije je prilikom promjene rezolucije slike. Tako je pri povećanju slike potrebno odrediti vrijednosti piksela koji na početnoj slici nisu postojali. Navedeno možemo postići metodama linearne ili bilinearne interpolacije i sličnima. Tehnika interpolacije slika koja se opisuje u ovome radu naziva se image morphing te je opisana u narednom poglavlju.

2. Opis područja i aktivnosti u svijetu

Morphing se u obradi slike koristi kao metoda za preobražaj s jedne slike u drugu. Ova metoda najčešće se primjenjuje kako bi se prikazala pretvorba jedne osobe u drugu. Primjer takve operacije prikazan je na slici 1.. Originalni autor prvog morphing algoritma je Douglas B. Smythe. U svome članku [1] opisuje tehniku koja se sastoji od niza istovremenih iskrivljenja i 'topljenja' slike. Morphing se prvi puta u velikoj mjeri koristio za snimanje filma Willow koristeći tehnike koje je razvila tvrtka Industrial Light and Magic. Morphing i dalje ima veliku ulogu u filmskoj industriji. Osim toga, izrazito se koristi u dizajnu modernih fontova. Dizajner je u mogućnosti upotrebom tehnologije *multiple master* odrediti kombinaciju dvaju stilova.

Morphing neprestano napreduje pa tako danas postoje programi koji automatski rade morphing slika uz minimalnu interakciju s korisnikom. Sve više se koristi i za izradu realnih efekata usporenih snimki koje u originalnom sadržaju ne postoje. U današnje vrijeme efekti morphinga dizajniraju se tako da su gotovo nevidljivi.

Figure 1.: Primjer morphinga lica

3. Pregled metoda za morphing slika

Ideja morphinga je relativno jednostavna. Sliku M želimo dobiti kombinacijom ulaznih sika I i J. Preobrazbu ulaznih slika kontroliramo pomoću parametra α koji može poprimiti vrijednosti od 0 do 1. Kada α ima vrijednost 0 tada je slika M identična ulaznoj slici I, dok je u suprotnom slučaju (kada α iznosi 1) slika M identična slici J. Najjednostavniju implementaciju morphinga postižemo ako ovaj opis pretvorimo u matematičku jednadžba (1) i direktno primijenimo na ulazne slike. Ovakva implementacija morphinga ne daje posebice kvalitete

rezultate. Razlog tome je što lokacije pojedinih dijelova lica nisu uzete u obzir, već se interpoliraju dijelovi slika neovisno o tome što prikazuju.

$$M(x,y) = (1-\alpha)I(x,y) + \alpha J(x,y) \quad (1)$$

Kako bi se izvršio ispravan morphing ulaznih slika I i J potrebno je prvo za svaki piksel (x_i, y_i) slike I odrediti odgovarajući piksel (x_j, y_j) na slici J. U ovom konkretnom slučaju sličnost za piksele p_i i p_j može se reći da odgovaraju jedan drugome ako se nalaze na jednakim dijelovima lica na slikama I i J (npr. oba piksela nalaze čine dio nosa). Kada su pronađeni poklapajući pikseli (p_i i p_j) na ulaznim slikama, prema formuli (2) određuje se lokacija piksela p_m na rezultantnoj slici.

$$\begin{aligned} x_m &= (1-\alpha)x_i + \alpha x_j \\ y_m &= (1-\alpha)y_i + \alpha y_j \end{aligned} \quad (2)$$

Konačno, vrijednosti piksela rezultantne slike M računamo na sljedeći način

$$M(x_m, y_m) = (1-\alpha)I(x_i, y_i) + \alpha J(x_j, y_j) \quad (3)$$

Potrebno je uočiti da je jednadžba (3) modificirani oblik početne jednadžba (1) tako da uzima u obzir sadržaj koji svaki piksel prenosi.

3.1. Brzina morphinga i ljudska percepcija istog

U sklopu projekta napravljen je kratak i zanimljiv eksperiment koji se tiče brzine morphinga. Kao što znamo, kod linearnog interpoliranja rezultantnu sliku generiramo koristeći jednadžbu $M(x,y) = (1-\alpha)I(x,y) + \alpha J(x,y)$ gdje je $I(x,y)$ početna, a $J(x,y)$ ciljna slika. Varijablu α u svakom koraku mijenjamo linearno u ovisnosti o broju koraka kojima želimo interpolirati sliku. Ako primjerice želimo usporiti morphing na početku, moramo nekako povećati utjecaj koji ima prva slika. Stoga se kao ideja nameće da α zamijenimo s $(\alpha)^k$. Dakle, imamo $M(x,y) = (1-(\alpha)^k)I(x,y) + (\alpha)^k J(x,y)$. Odavde se vidi da ako želimo usporiti morphing na početku moramo odabrati neki smisleni (ne prevelik po apsolutnoj vrijednosti) k, i to tako da je $k > 1$. U tome slučaju, vidimo da je utjecaj prve slike u svakom koraku interpolacije veći nego inicijalno jer iz $(\alpha)^k < \alpha$ slijedi $1-(\alpha)^k > 1-\alpha$. Pri čemu, jasno, ako odaberemo preveliki k, imamo prenagli prijelaz u zadnjem koraku interpolacije. Zato za α odabiremo neke smislene vrijednosti kao npr. 1.3 da interpolacija i dalje izgleda dobro, ali je početno nešto usporena. Potpuno analogno, ako želimo ubrzati interpolaciju, odnosno imati nagliji prijelaz na početku (što je zapravo i bila motivacija za ovaj postupak, naime razlika između prve dvijetri slike nam se činila premalena) treba odabrati k za koji vrijedi $0 < k < 1$. Također, taj k bi trebao biti smislen, primjerice $k = 0.8$. Usporedbom slika 2. i 3. doista možemo vidjeti golim okom da je uz $k = 0.8$ morphing ubrzan. Možemo konstatirati da je donekle diskutabilno koji je niz slika ljudskom oku prirodniji (tj. možda bi netko procijenio da je bolji morphing na slici 3., tj. bolji niz slika). U tome kontekstu, smisleno je promotriti brzinu morphinga. Ova dva primjera su napravljena algoritmom implementiranim u nastavku rada te ujedno prikazuju i rezultate algoritma. Možemo primijetiti da je moguće napraviti svakakve kombinacije pri mijenjanju brzine morphinga (možemo eksperimentirati sa svakakvim ad-hoc funkcijama od α), ali nismo toliko duboko obradili ovu tematiku.

Figure 2.: Morphing uz faktor $k = 1$ (standardno)

Figure 3.: Morphing uz faktor $k = 0.8$ (ubrzano)

4. Implementacijski detalji

Za potrebe demonstracije razvijen je jednostavan program koji ostvaruje jedan algoritam morphinga

lica. Osnovna ideja algoritma je detaljnije opisana u sljedećem odjeljku a apstraktna ideja je pronaći regije točaka zajedničke u objema slikama te ih mapirati jedne na drugu. Te regije su zapravo trokuti (poligoni) koji odgovaraju određenim dijelovima lica. Korišten je programski jezik *Python* uz knjižnice *numpy*, *dlib* te *OpenCV*.

Koraci algoritma su:

1. **Pronalazak zajedničkih regija** Karakteristične točke lica omogućuju pronalaženje sličnih elemenata na ulaznim slikama I i J. Osnova ideja je da se povežu točke koje odgovaraju istim dijelovima lica na ulaznim slikama. Za određivanje karakterističnih točaka lica korištena je biblioteka *dlib*. Navedena biblioteka pronalazi 68 karakterističnih točaka (slika 4.). Dodatno, na slikama se označavaju i točke u uglovima te točke koje se nalaze na polovici udaljenosti između uglova (inače se mogu dobiti čudne podjele na regije u idućem koraku). Ako se na ulaznim slikama označi i veći broj točaka dobivaju se bolji rezultati, ali cijeli postupak morphinga se usporava s povećanjem broja karakterističnih točaka.

2. **Delaunay triangulacija** U ovom koraku se iz pronađenih karakterističnih točaka računaju trokuti koji čine regije. Primjerice ako imamo točke t_1 do t_9 onda jedna se jedna regija može primjerice sastojati od trokuta (t_3, t_5, t_7) a druga od (t_2, t_5, t_9) itd. Ova podjela na regije se mora računati samo jednom i to za samo jednu sliku, nakon čega se ide u sljedeći korak algoritma. Dovoljno je računati podjelu na regije za samo jednu sliku jer ta podjela vraća indekse točaka u svakoj pojedinoj regiji. Ti indeksi se onda mogu iskoristiti za stvaranje regija u drugoj slici zbog činjenice da su točke koje vraća *dlib* točno određenog redoslijeda. Korišteni algoritam za podjelu točaka u trokute se zove Delaunayeva triangulacija [2] te je implementiran u *OpenCV*. Delaunayeva triangulacija karakterističnih točaka lica prikazana je na slici 5..

3. **Interpolacija područja** Nakon što su određena područja koja nas zanimaju, pristupa se interpolaciji područja. Interpolacija područja vrši se istovremeno i u prostoru i po bojama. Naime, u jednom koraku, jedno područje se istovremeno pomakne od početnog prema krajnjem te je u koraku to područje sadržajno linearna interpolacija početnog i krajnjeg područja.

Figure 4.: Karakteristične točke lica

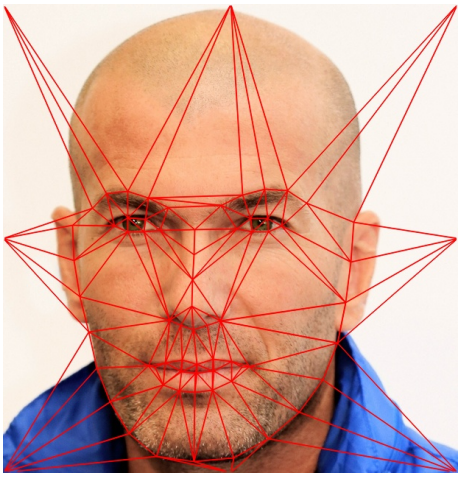

Figure 5.: Delaunayeva triangulacija karakterističnih točaka lica

4.1. Rezultati, problemi, budući rad

Razvijeni program omogućava dva načina rada: interaktivni i pasivni. U pasivnom načinu rada programu se predaju dvije slike i parametri te on stvara niz slika zadane duljine zadanim algoritmom (linearnom interpolacijom ili prethodno opisanim).

U interaktivnom načinu rada, na sučelju je vidljiv klizač kojim je moguće interaktivno u gotovo stvarnom vremenu vidjeti rezultat morphinga za zadani intenzitet. Detaljnije o korištenju programa se može pronaći u samom programu koji pri pokretanju bez parametara ispisuje pomoć o korištenju. Rezultati pokretanja programa su već prikazani na slikama 2. i 3..

Glavni problem kod trenutne implementacije je brzina izvođenja za opisani algoritam, u interaktivnom načinu rada nemoguće je dobiti glatko pomicanje klizača za interaktivni način rada. Osim toga, česti problem je i neuspjeh algoritma za pronalaženje karakterističnih točaka lica te u tom slučaju program javlja pogrešku.

S obzirom na to da je program napravljen modularno, vrlo je lako dodati nove algoritme u kod (primjerice za morphing automobila umjesto lica ili slično). To otvara mogućnost za daljnji rad i eksperimentiranje.

5. Zaključak

Image morphing je zanimljiva tehnika interpoliranja dvaju slika lica. Ideja iza image morphinga je dosta jednostavna i laka za implementirati. Ako ciljamo na bolje rezultate može se zakomplicirati ovisno koliko dobar rezultat želimo dobiti. Implementacija objašnjena u ovome radu dala je dosta dobre rezultate, uz manje nadogradnje osnovne ideje interpoliranja. Naravno, moguće su buduće nadogradnje. Posebnu pažnju trebalo bi obratiti na povećanje brzine u interaktivnom načinu rada.

References

[1] D. B. Smythe. A Two-Pass Mesh Warping Algorithm for Object Transformation and Image Interpolation, ILM Technical Memo 1030, Computer Graphics Department, Lucasfilm Ltd., 1990

[2] Robert Delaunay Delaunay's triangulation - https://www.mathworks.com/help/matlab/math/delaunay-triangulation.html