Animiranje obrazov v slikarskih delih

Janez Justin[1]*, Katja Kunej[1]*, Katarina Pikec[1]*, Tajda Urankar[1]*, as. Blaž Meden[1], izr. prof. dr. Narvika Bovcon[1]

[1]*Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana*
E-pošta: jj7084@student.uni-lj.si, kk9251@student.uni-lj.si, kp4446@student.uni-lj.si, tu5161@student.uni-lj.si, blaz.meden@fri.uni-lj.si, narvika.bovcon@fri.uni-lj.si

** These authors contributed equally to this work.*

Animating faces on paintings

Abstract. *In this paper, we introduce our approach of animating faces on famous paintings. The most important part was face detection which helped us find the critical points of the face in our video and our chosen painting. There are a lot of approaches and algorithms available on the internet. Some of them we described in section 2. For our version we used Python libraries dlib and skimage that were essential for face detection and affine transformation. Firstly we filmed a short video with our face expressing a certain emotion. With mentioned libraries we detected eyes, brows, lips, etc. Then we tracked all these points throughout the video and stored their positions. With the affine transformation of the tracked points we were able to transform the movement from the video onto the painting. We paid special attention to defining a safe border around the face on the painting where the movement is allowed. Otherwise we noticed some anomalies in the output video. Overall, we conclude that our algorithm is fast if the paintings are preprocessed, gives good results and is useful for some personal fun.*

1 Uvod

Kakšna čustva ali izraze na obrazu (ang. *facial expressions*) lahko razberemo iz slikarskih del? To vprašanje nas je spodbudilo k raziskovanju že uveljavljenih rešitev in algoritmov, ki se ukvarjajo z zaznavanjem obrazov (ang. *face detection*), animacij in raznih transformacij. Našli smo precej že obstoječih rešitev; na primer algoritem, ki z umetno inteligenco animira obraze ali pa algoritem, ki je sestavljen iz vhodne slike in izvornega videa in vrača obrazne točke, ki se premikajo. Slednji algoritem je bil tudi osnova za našo raziskavo, vendar vseeno na malenkost drugačen način. Raziskave smo se lotili precej matematično, in sicer z afinimi transformacijami točk iz videa na slikarsko delo, tako se slikarsko delo premika glede na video. Video smo že prej posneli, nato smo locirali kritične točke na obrazu (oči, obrvi, usta, obris) in spremljali njihove pozicije skozi video. Na slikarskem delu je program prav tako zaznal obrazne točke in jih nato, glede na točke iz videa, spreminjal, da smo dobili gibanje.

2 Obstoječe rešitve

Obstaja veliko načinov za interaktivno modifikacijo slik, ena možnost je uporaba umetne inteligence, druga pa npr. ročna modifikacija slik.

Ena izmed že obstoječih rešitev je projekt "Deep Nostalgia" [1]. Uporabnikom je na voljo kot spletna [2] ali mobilna aplikacija. Uporabnik si mora za uporabo spletnega vmesnika ustvariti račun. V spletni vmesnik naloži sliko in počaka na animirane rezultate. Vsaka naložena slika se animira na enak način.

Rešitev GANimation [3] na eni sliki predela izraz na obrazu z uporabo generativnih kontradiktornih omrežij (GAN), ki so sestavljena iz dveh sistemov. Prvi sistem, generator, ustvari kopijo slike z uporabo novih podatkov, drugi sistem, diskriminator, pa določa, ali podatki prehajajo kot resnični ali lažni. Sčasoma se bosta orodji naučili izdelovati čedalje bolj realistične slike oz. animacije.

Podobno uporabljajo GAN tudi druge rešitve, kot je npr. "SC-FEGAN" [4], ki uporabnikom daje možnost skiciranja na sliko in uporabe barv. Končna slika vsebuje, kar je uporabnik narisal (npr. očala, nasmešek). Še ena rešitev "X2Face" [5] omogoča vnos dveh videov, potem pa se originalni video premika skladno s premiki v drugem videu.

"First Order Motion Model for Image Animation" [6] je rešitev, ki omogoča animacijo ne samo obrazov, temveč deluje tudi npr. na telesu in likih. Model je potrebno najprej natrenirati na objektih iz iste kategorije, slabost pa je, da je časovno precej potraten.

3 Ideja

Naša ideja je izhajala iz tega, da na izbranih slikarskih delih prikažemo različne izraze na obrazu s pomočjo lastnih video posnetkov. Ideja je bila, da spremembe mimike iz videa apliciramo na slikarsko delo in ga tako animiramo. Obrazne točke slikarskega dela in posamezne sličice videa smo zaznali z uporabo strojno naučenih modelov. Premikanje točk obraza iz videa smo aplicirali na slikarsko delo in dobili animacijo z željenimi spremembami.

4 Baza slik

Slike, ki smo jih vključili v projekt, smo pridobili iz spletne enciklopedije vizualne umetnosti WikiArt [7]. Na

spletni strani smo najprej poiskali umetnike, ki so izdelovali portrete. Preizkusili smo nekaj portretov, kjer je bil obraz obrnjen levo, desno ali naravnost. Ugotovili smo, da so popačenja večja, če je obraz preveč zasukan v eno smer, najbolje je bilo, če je bil obraz obrnjen naravnost. Razlog za večje popačenje je tudi v tem, da so bile naše glave pri snemanju izvornih videov usmerjene direktno v kamero. Prav tako smo ugotovili, da so mimike izražene na obrazu s slike manj popačene, če je slika bolj čista. Za najboljši rezultat smo izbrali portrete, kjer glava ni preveč zasukana in slika deluje kar se da čisto.

5 Vhodni videi

Pri snemanju vhodnih videov smo najprej poskusili z bolj izrazitimi premiki ustnic, čela ter tudi zapiranjem oči, vendar je pri animaciji prišlo do prevelikega popačenja obraza. Spreminjala se je oblika glave ali pa so se ušesa neusklajeno premikala. Želeli smo čim bolj naravne obrazne premike, zato smo posneli videe z rahlimi spremembami v izrazih na obrazu. Obrazi na slikarskih delih so obrnjeni v različne smeri, imajo različne oblike obrazov, zato je bilo potrebno za vsako sliko posebej popraviti vse izraze. V nasprotnem primeru bi dobili preveč popačene rezultate. Izkazalo se je, da manjše spremembe obrazne mimike delujejo bolj prepričljivo in fotorealistično kot velike spremembe, ki delujejo groteskno oziroma kot pretiravanje v risankah, kjer se skladno s principi animacije mimiko in gibe pretirano poudarja. Ves čas snemanja je bilo potrebno paziti na mirujoč položaj glave, sicer je pri animaciji prihajalo do popačenj.

6 Implementacija

Izvorna koda je dostopna na GitHubu [8], projekt pa na GitHub Pages [9]. Rešitev je implementirana v jeziku Python, pri tem pa smo si pomagali s knjižnicama:

- `dlib` v kombinaciji z 68 točkovnim modelom za zaznavanje obraznih točk, dostopnim na dlib spletni strani [10],
- `skimage` za izvedbo afinih transformacij [11].

Velja omeniti, da se slikarsko delo in video ne ujemata nujno v višini in širini.

6.1 Določanje časovnih sprememb

Pri določanju sprememb smo se osredotočili na posamezne sličice v videu, rekli smo ji j-ta sličica in se posvečali procesiranju le-te.

Za določanje sprememb smo uporabili prvo sličico iz videa; rekli smo ji sidro. Za vsako sličico smo poiskali spremembo vsake izmed obraznih točk; to je razlika i-te točke sličice in i-te točke sidra. Določili smo še seznam $v_s = [(x_1, y_1), ..., (x_n, y_n)]$.

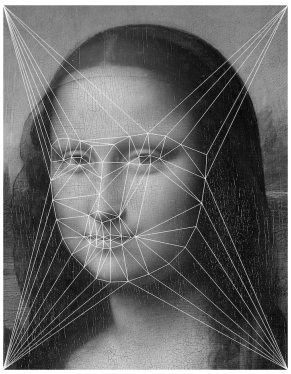
Označili smo $k = \frac{s_{size}}{d_{size}}$, kjer s_{size} predstavlja velikost obraza na sličici in d_{size} velikost obraza na slikarskem delu. $v_d := kv_s$ je potem seznam, katerega i-ti element ob seštevanju je popeljal i-to točko slikarskega dela na mesto, kjer smo to točko želeli. To skaliranje smo izvedli ločeno za os x in ločeno za os y.

Nato smo vzeli $d = [(x_1, y_1), ..., (x_n, y_n)]$ obrazne točke slikarskega dela. $d_n = d + v_d$ so obrazne točke, ki smo jih želeli za trenutno sličico končne animacije.

Za konec določanja sprememb smo želeli transformirati obraz označen s točkami d na slikarskem delu v obraz označen z d_n. Tako smo dobili j-to sličico naše končne animacje.

6.2 Afina transformacija

Z Delaunayevo triangulacijo [12] (slika 1) smo triangulirali seznam točk d_n in potem posamezni trikotnik afino transformirali s premikom oglišč v d. Seveda tako, da je i-ta točka iz d pristala na i-ti točki iz d_n. Tej transformaciji smo rekli A. Potem je bila transformacija A^{-1} ravno tista, ki smo jo želeli, da je slikarsko delo transformirala v j-to sličico naše animacije.



Slika 1: Delaunayeva triangulacija prikazana na Mona Lisi.

Akcije v razdelku 6.2 so v implementaciji izvedene z uporabo funkcij iz knjižnice `skimage.transform`. Za pridobitev A in `warp` za izvedbo A^{-1} smo uporabili `PiecewiseAffineTransform.estimate`.

7 Izboljšave implementacije

Na tej točki implementacije je bil rezultat že skoraj prepričljiv. Problem se je pojavil v naslednjih primerih:

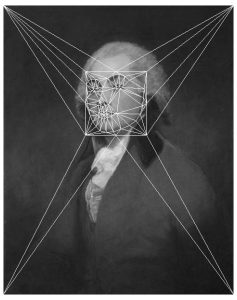
- če subjekt na videu ni ostajal centriran, se je rezultat pokvaril,
- ob premikanju npr. brade, je afina transformacija "sploščila" trikotnik pod brado, kar se je izrazilo v deformiranem trupu,
- je šum pri zaznavanju obraza iz videa povzročil tresočo sliko.

7.1 Normalizacija poravnanosti obraza

Namesto, da bi gledali spremembe (t.j. seznam v_s) direktno med točkami na trenutni sličici in med sidrom, smo gledali, do kakšne spremembe je prišlo med neko točko, ki smo jo izbrali za fiksno (v našem primeru je to točka med obrvema) na trenutni sličici in med to isto točko na sidru. Tako smo gledali samo spremembe na obrazu, ne pa tudi sprememb v prostoru.

7.2 Rešitev deformacije trupa

Ideja za rešitev tega problema je bila, da okrog obraza postavimo kvadrat, ki je igral vlogo varnega okvirja (slika 2), zunaj katerega ni prihajalo do sprememb.



Slika 2: Delaunayeva triangulacija z varnim okvirjem na George Romneyevem delu Captain John Taubman III.

Zopet smo si izbrali neko fiksno točko, od katere smo gledali odmike točk. Varni okvir smo določili kot največji možni odmik obraza gor, dol, levo in desno od fiksne točke na vhodnem videu. Ta odmik smo potem, podobno kot v_s, skalirali, da smo dobili pozicije na slikarskem delu.

7.3 Predprocesiranje posnetkov

Ker smo iste videoposnetke uporabili pri več slikarskih delih, je procesiranje posameznega posnetka izraženega čustva vzelo precej časa - za vsako procesirano slikarsko delo se je na videoposnetku izvedlo povsem enako zaznavanje točk. Zato smo se odločili, da vsak videoposnetek predprocesiramo in ustvarimo seznam obraznih točk, ki jih potem kot JSON datoteko posredujemo programu. Ta JSON datoteka vsebuje 68 značilnih obraznih točk in točke pravokotnika pozicije obraza (ki se uporablja za določanje faktorja skaliranja pri prevajanju točk iz videoposnetka v slikarsko delo) za vsako sličico v videu.

S to metodo smo hitrost procesiranja posameznega slikarskega dela skoraj razpolovili. Tudi uvedba metode opisane v 7.4 je bila lažje implementirana s predprocesiranimi videoposnetki.

7.4 Glajenje vhodnega videa

Čeprav je bil subjekt na vhodnem videu pri miru, je prihajalo do manjšega šuma pri zaznavanju točk obraza. To je pripeljalo do tresočega rezultata. Ta problem smo reševali s povprečenjem iste točke na zaporednih sličicah.

Za vsako izmed 68 značilnih točk (ang. *facial landmarks*) na poljubni sličici videa smo izračunali povprečno pozicijo v koordinatnem sistemu na podlagi n prejšnjih in naslednjih sličic. Opazili smo, da se metoda pri 30 sličicah na sekundo najbolje obnese pri $n = 5$. Ob izbiri večjega n se pojavijo preveč razvlečeni premiki. Ob manjšem n-ju pa je slika še vedno tresoča.

8 Rezultati

Za predstavitev rezultatov smo izbrali dve slikarski deli vidni na sliki 3. Na slikah 4 in 5 pa so predstavljena vsa izražena čustva.

Slika 3: Slikarsko delo Thomasa Gainsborougha, "Woman in Blue", brez animacije (levo). Slikarsko delo Alekseya Antropova, avtoportret, brez animacije (desno).

Slika 4: Izrazi na obrazu predstavljeni po vrsti od leve zgoraj do desne spodaj: zaspanost, posmehljivost, sumničavost, veselje, žalost, zaljubljenost.

9 Uporaba vmesnika

Pripravili smo tudi vmesnik (slika 6) za lažji pregled generiranih videov, pri katerem smo uporabili HTML in JavaScript. Vmesnik omogoča izbiro slikarskega dela in emotikona (ki prikazuje izraz na obrazu), potem pa predvaja video posnetek, ki smo ga generirali za ta par slike in izraza na obrazu. Uporabnik lahko po želji izbere poljubno drugo slikarsko delo ali emotikon in animacija se bo avtomatsko posodobila. Video posnetek je mogoče ustaviti, ponovno predvajati ali pa predvajati čez celoten zaslon. Pod generiranim video posnetkom se izpišejo podatki slikarskega dela – avtor, naslov, leto nastanka. S tem se uporabnik seznani z deli, ki jih prej ni poznal. Menimo, da je vmesnik uporabnikom prijazen, saj jih od začetka vodi z navodili za uporabo.

10 Predlogi za izboljšave

Ena izmed izboljšav, ki bi jo lahko implementirali, bi bila tudi možnost odpiranja ust likov, saj pri nas to ni mogoče. Če so bila usta v vhodnem videu odprta, se je to na končnem videu prikazalo kot črn prostor v ustih, torej ni bilo videti zob ali jezika. Prav tako bi lahko izboljšali tudi zapiranje vek, saj trenutno, če oseba na vhodnem videu preveč zapre oči, pride pri animaciji do popačenja delov obraza okoli oči. Če bi nam to uspelo, bi bila po našem mnenju uporabniška izkušnja še boljša, rezultati pa še bolj realistični. Dober primer zapiranja oči in odpiranja

Slika 5: Izrazi na obrazu predstavljeni po vrsti od leve zgoraj do desne spodaj: zaspanost, posmehljivost, sumničavost, veselje, žalost, zaljubljenost.

Slika 6: Prikazan uporabniški vmesnik prikazuje predogled animiranega videa (levo), možnost izbire različnih slikarskih del (desno zgoraj) in izbiro upodobljenega izraza na obrazu preko emotikonov (desno spodaj).

ust so dosegli z globokimi nevronskimi mrežami [13], kar pa bi precej povečalo časovno zahtevnost generiranja animacij.

Še ena možna izboljšava bi bila, da bi na slikarskem delu animirali več obrazov. Tako bi se lahko vsi obrazi na slikarskih delih premikali in bi lahko uporabili dela, na katerih se pojavi več oseb. Trenutna implementacija že zazna vse obraze, dodatno bi morali na vsaki sliki videa animirati vse obraze namesto samo prvega zaznanega. Čas generiranja videov za n obrazov bi se tako podaljšal za približno n-krat. Če bi želeli na obrazih uporabiti različne izraze, bi bila izboljšava še malce otežena. Zaradi različnih dolžin posnetkov izrazov bi se pojavil problem, kaj storiti z obrazi, katerih animacija se je končala. Možna rešitev bi bila, da animacijo, ki pride do konca, odvrtimo v vzvratno smer. Drug problem bi bil, kako izbrati, katera čustva želimo na danih obrazih. To bi bilo rešljivo z razširitvijo uporabniškega vmesnika.

Tudi pri uporabniškem vmesniku je še prostor za izboljšave. Uporabnik bi lahko posnel samega sebe za izvorni video, ki bi ga naš algoritem nato sprocesiral, liki pa bi se premikali, tako kot bi uporabnik želel. Sam postopek bi vzel več časa, kar bi po našem mnenju vodilo k slabši uporabniški izkušnji - zaradi tega smo v obstoječi verziji vmesnika to možnost izključili.

11 Zaključek

Glede na to, da smo opisano raziskavo delali v okviru predmeta in smo zato bili časovno omejeni, smo precej zadovoljni z rezultati. Sama spletna aplikacija je zabavna za uporabo, je pa še ogromno možnosti za izboljšavo (nekaj je omenjenih v prejšnjem poglavju), ki bi jih bilo potrebno preizkusiti.

Literatura

[1] Deep Nostalgia, https://www.smithsonianmag.com/smart-news/ai-program-deep-nostalgia-revives-old-portraits-180977173

[2] Uporabniški vmesnik Deep Nostaliga, https://www.myheritage.si/deep-nostalgia

[3] GANimation: Anatomically-aware facial animation from a single image, https://openaccess.thecvf.com/content_ECCV_2018/html/Albert_Pumarola_Anatomically_Coherent_Facial_ECCV_2018_paper.html

[4] SC-FEGAN (Face Editing Generative Adversarial Network), https://openaccess.thecvf.com/content_ICCV_2019/papers/Jo_SC-FEGAN_Face_Editing_Generative_Adversarial_Network_With_Users_Sketch_and_ICCV_2019_paper.pdf

[5] X2Face, https://arxiv.org/pdf/1807.10550.pdf

[6] First Order Motion Model for Image Animation, https://iris.unitn.it/retrieve/handle/11572/250831/300686/8935-first-order-motion-model-for-image-animation.pdf

[7] enciklopedija vizualne umetnosti WikiArt (baza slikarskih del), https://www.wikiart.org/

[8] GitHub repozitorij, https://github.com/jjustin/paintings-animator

[9] projekt izdan na GitHub Pages, http://jjustin.github.io/paintings-animator

[10] knjižnica dlib, http://dlib.net/files

[11] knjižnica skimage, https://scikit-image.org/docs/dev/api/skimage.html

[12] skimage PiecewiseAffineTransform dokumentacija, https://scikit-image.org/docs/stable/api/skimage.transform.html#skimage.transform.PiecewiseAffineTransform

[13] Interactive facial animation with deep neural networks, https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-cvi.2019.0790