Sledenje pogledu s spletno kamero

Luka Vranješ
Univerza v Ljubljani Fakulteta za računalništvo in informatiko
Večna pot 113
1000, Ljubljana
luka.vranjes5@gmail.com

Jure Žabkar
Univerza v Ljubljani Fakulteta za računalništvo in informatiko
Večna pot 113
1000, Ljubljana
jure.zabkar@fri.uni-lj.si

POVZETEK

Sledenje pogledu uporabnika med uporabo računalnika je najbolj priljubljeno v industriji iger in medicinskih aplikacijah. Čeprav so prvi sistemi za sledenje pogledu uporabljali spletne kamere, so se sčasoma v ta namen razvile posebne naprave, ki jih odlikuje precej večja natančnost in temu primerno višja cena. V naši raziskavi napravo za sledenje pogledu uporabljamo za odkrivanje disleksije pri osnovnošolcih. Zaradi želje po širši uporabi aplikacije, želimo napravo za sledenje pogledu nadomestiti s cenovno ugodnejšo spletno kamero. V pričujočem delu nas zanima, kakšno natančnost lahko dosežemo z uporabo spletne kamere in globokimi nevronskimi mrežami, ki jih uporabljamo pri učenju preslikave med položajem oči in točko na zaslonu.

Keywords

eye-tracking, webcam

1. UVOD

Sledenje očem pri uporabi računalnika postaja vse bolj priljubljeno. V industriji se najpogosteje uporablja za testiranje uporabniške izkušnje pri uporabi aplikacij in pri oglaševanju ter v industriji računalniških iger, kjer igralcu s prilagajanjem vmesnika omogoči večjo vživetost. Na raziskovalnem področju se zelo pogosto uporablja v medicinskih in psiholoških študijah, saj očesni gibi odražajo mnoge kognitivne in nevrološke motnje.

Aplikacija za odkrivanje disleksije, ki smo jo razvili v okviru projekta ŠIPK Disleksija [10], za sledenje pogledu uporablja sledilec očem proizvajalca Tobii. Čeprav je naprava relativno poceni, brez drage licence ni uporabna za naš namen. Testiranje otrok z aplikacijo je zato omejeno in dostopno le redkim. V želji po širši uporabi aplikacije želimo sledilec očem Tobii nadomestiti s cenovno ugodnejšimi in zelo razširjenimi spletnimi kamerami, kar bi uporabnikom omogočilo testiranje na domu.

V tej raziskavi poskušamo z uporabo globokega učenja in pristopov računalniškega vida doseči čim večjo natančnost sledenja pogledu s spletno kamero. Omejimo se na uporabo v aplikaciji za odkrivanje disleksije, ki je sestavljena iz šestih nalog. Štiri naloge uporabljajo večje grafične elemente in glede natančnosti niso zahtevne, dve bralni nalogi pa zahtevata visoko natančnost.

2. PREGLED PODROČJA

Sledenje pogledu s kamero je dobro raziskano; pregled metod je povzet v [2]. Problem sledenja tipično obravnavamo kot dva podproblema – zaznave očesa in napovedovanje točke pogleda. V splošnem se sorodna dela izogibajo uporabi specializiranih kamer. Tako so uporabljene kamere, kot tudi v našem primeru, povprečne kakovosti. Zaradi potrebe po delovanju v realnem času, sledenje pogledu onemogoča uporabo zahtevnih algoritmov. Komercialni sledilci očem ponujajo minimalno frekvenco 30Hz, kar je maksimalna hitrost zajemanja slik preprostih spletnih kamer.

Med pristopi, ki uporabljajo zgolj spletno kamero, so se za zelo uspešne izkazale metode, ki uporabljajo konvolucijske nevronske mreže, kar zahteva veliko količino podatkov [4, 12]. V prvem so uspeli zbrati podatkovno množico 2445504 slik in pripadajočih točk na zaslonu ter z njo dosegli odlične rezultate na pametnih telefonih in tabličnih računalnikih. V [12] so imeli bistveno manjšo učno množico – 213659 slik in prileţnih točk. Primer klasičnega pristopa je predstavljen v [8], kjer za napoved uporabljajo interpolacijo, kot vhod pa točki središča zenice in zunanji očesni kotiček. Zanimiva je rešitev v [5], kjer za napoved uporabljajo vektor pridobljen iz slike očesa. Podrobnosti tega koraka lahko najdemo v [11]. Presenetljivo redko raziskave upoštevajo obraz; enostaven primer sledenja očem pri upoštevanju obraza najdemo v [7]. V članku je tudi opisano, da upoštevanje obraza ni pogosto zaradi potrebe po čim hitrejšem delovanju predlaganih pristopov.

Naša rešitev preizkuša primernost uporabe globokega učenje brez predhodnega učenja, kar ni preizkušeno v nobeni od predstavljenih implementacij.

3. TEORETIčNO OZADJE

Naš algoritem je sestavljen iz štirih delov: zaznavanje obraza (algoritem Viola-Jones), poravnave obraza z ansamblom regresijskih dreves, ocenjevanja centra očesa z gradienti slike in globoko nevronsko mrežo za modeliranje preslikave med očesom in točko na zaslonu.

Zaznavo obraza na podlagi preprostih značilk smo implementirali po [9]. Metoda razdeli obraz na kvadratne bloke, ki podajo oceno glede na razliko vsot dveh izbranih pravokotnih skupin slikovnih pik. To je mogoče izračunati zelo hitro z uporabo integralne slike, ki jo z enim obhodom izračunamo iz vhodne slike. Ker je število možnih preprostih značilnosti veliko (preko 180000 v primeru, ko je detektor velikosti 24×24), se za izbiro najboljših uporablja različico algoritma

AdaBoost. S tem med vsemi značilnostmi izberemo nekaj 100 ali nekaj 1000 takih, ki skupaj dobro delujejo. Končni algoritem deluje po principu kaskade klasifikatorjev. Za vsak nivo se z uporabo algoritma AdaBoost izbere nekaj preprostih klasifikatorjev. Njihove uteži so prilagojene tako, da na testni množici prepoznajo čim več obrazov. Napačno zaznane poskušamo odstraniti v kasnejših nivojih kaskade. Takoj, ko v enem nivoju v interesnem območju ne zaznamo obraza, to območje zavržemo.

Za poravnavo obraza smo uporabili ansambel regresijskih dreves [3]. Za učenje enega regresijskega drevesa uporabimo trojček, ki je sestavljen iz slike obraza, začetne napovedi oblike in ciljnega popravnega vektorja. Iz tega se naučimo regresijske funkcije, z uporabo algoritma iz [3] in vsoto kvadratov napake kot funkcijo izgube.

Center očesa določimo z gradienti slike po metodi [6]. Za vsako točko na sliki izračunamo vsoto vektorskih produktov med vsemi gradienti g_i in normaliziranim vektorjem premika d_i, ki imajo enako orientacijo. Vektor premika izračunamo z:

$$d_i = \frac{x_i - c}{||x_i - c||}, \forall i : ||g_i|| = 1, \quad (1)$$

kjer x_i predstavlja lokacijo gradienta g_i na sliki. Za boljšo robustnost algoritma se priporoča normalizacija gradientov. Prav tako je normaliziran vektor premika, saj tako ohranimo enako utež za vse slikovne pike. Končni optimalni center nato dobimo po enačbi:

$$c^* = \arg\max_c \left\{ \frac{1}{N} \sum_{i=1}^{N} (d_i^T g_i)^2 \right\} \quad (2)$$

Hitrost izračuna lahko zmanjšamo tako, da upoštevamo le gradiente z dovolj visoko magnitudo.

4. IMPLEMENTACIJA

Algoritem smo implementirali v programskem jeziku Python. Razdelimo ga lahko na štiri korake: zaznavanje obraza, zaznava interesnih točk na obrazu, zaznava središča zenice in napovedovanje točke pogleda.

4.1 Zaznavanje obraza

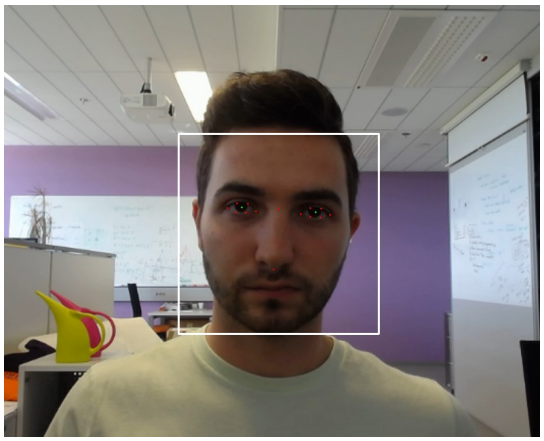
Za implementacijo algoritma smo uporabili že obstoječo implementacijo iz knjižnice openCV. Razred CascadeClassifier smo inicializirali s prednaučenimi kaskadami ("haarcascade_frontalface_default.xml"), ki so ravno tako del knjižnice. Za hitrejše delovanje smo sliko predhodno pomanjšali na velikost $(426, 240)$ in jo pretvorili v sivinsko sliko. Slednje od nas zahteva funkcija detectMultiScale, ki je del razreda CascadeClassifier. Ostala dva parametra, ki jih ta funkcija zahteva sta še skalirni faktor, ki smo ga nastavili na 1.3, kar je najmanjše število bližnjih mejnih pravokotnikov, katerega smo nastavili na 5. Rezultat funkcije so mejni pravokotniki vseh najdenih obrazov. Ker nas zanima le en obraz, med vsemi ohranimo le največjega. Razultat koraka je prikazan kot bel kvadrat na sliki 1.

4.2 Zaznava interesnih točk na obrazu

Za implementacijo tega koraka smo uporabili razred shape_predictor, ki je vključen v knjižnici dlib. Ta vsebuje tudi datoteko shape_predictor_68_face_landmarks.dat, ki jo uporabimo za inicializacijo razreda in vsebuje potrebne prednaučene parametre. Za pridobitev interesnih točk smo uporabili funkcijo predictor(), ki je del razreda shape_predictor in kot vhodna argumenta prejme sivinsko sliko in mejni pravokotnik. Sivinska slika je celotna slika, pridobljena iz spletne kamere, pretvorjena v sivinsko sliko. Mejni pravokotnik je rezultat prejšnjega koraka. Od 68 interesnih točk ohranimo le 14 izbranih. Rezultat koraka je prikazan kot množica rdečih točk na sliki 1.

4.3 Zaznava središča zenice

Za ocenjevanje centra zenice z gradienti slike smo uporabili [1]. Vhod v algoritem je sivinska slika očesa. Iz sivinske slike kamere iz prejšnjega koraka na podlagi interesnih točk izrežemo obe očesi. Nad dobljenima slikama uporabimo funkcijo iz knjižnice openCV, equalizeHist(). Obe sliki nato ločeno podamo v funkcijo locate(). Rezultat funkcije za vsako sliko sta x in y koordinati središč obeh zenic. Rezultat koraka je prikazan z zelenima točkama na sliki 1.



Slika 1: Prikaz dobljenih središč zenic (zelena barva), ohranjenih interesnih točk (rdeča barva) in mejnega pravokotnika okoli obraza.

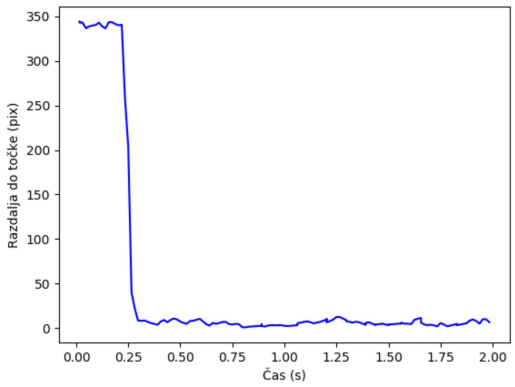
4.4 Napovedovanje točke pogleda

Iz knjižnice dlib smo uporabili razred MLPRegressor za implementacijo globokih nevronskih mrež. Tej smo pri inicializaciji nastavili 5 parametrov. Velikost skritih nivojev smo nastavili na $(23, 14, 6)$. Največje število epoh smo omejili na 500. Za aktivacijsko funkcijo smo uporabili ReLU. Za reševalnik smo nastavili adam. Toleranco smo nastavili na 10^{-5}. Vhodni nivo mreže prejme 48 parametrov sestavljenih iz 14 interesnih točk in 2 točk, ki predstavljata center zenice. Vhodne parametre pred učenjem s pomočjo razreda StandardScaler (del openCV knjižnice) s funkcijo transform() prilagodimo na interval, primeren za učenje.

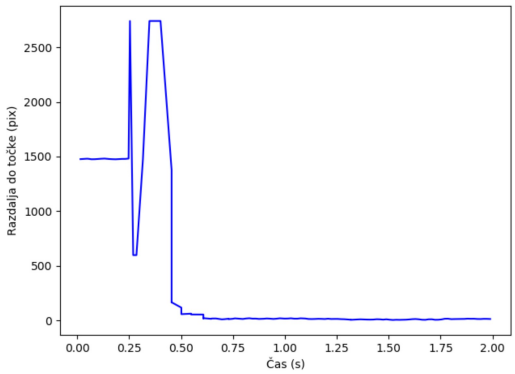
5. EKSPERIMENTI

Velik vpliv na točnost napovedi ima začetna kalibracija sistema: uporabnik na črnem zaslonu gleda v prikazujoče se rdeče točke. Na zaslonu je hkrati prikazana le ena točka na naključni lokaciji. Kalibracija je nujen korak za uporabo sledilca in je unikatna za vsakega uporabnika.

V sklopu delovanja in za potrebe eksperimentov definiramo



Slika 2: Prikaz razdalje od napovedane točke pogleda do dejanske točke v odvisnosti od časa.



Slika 3: Prikaz razdalje od napovedane točke pogleda do dejanske točke v odvisnosti od časa.

dodatne omejitve, vezane na aplikacijo za odkrivanje disleksije. Kamera mora biti fiksna, z resolucijo 1280 in hitrostjo vsaj 30 sličic na sekundo. Glava mora biti med uporabo od začetka kalibracije naprej v statičnem položaju, uporabnik pa v dobro osvetljenem okolju. Poleg tega pri vseh eksperimentih uporabljamo še nekaj statičnih parametrov. Zaslon z diagonalo 59.44 cm in ločljivostjo 1920×1080. Oddaljenost uporabnika od zaslona je 55 cm. Dolžina kalibracije je omejena na 5 minut. Predstavljeni eksperimenti so izvedeni na enem uporabniku.

5.1 Okno zajema podatkov in dolžina prikaza točke

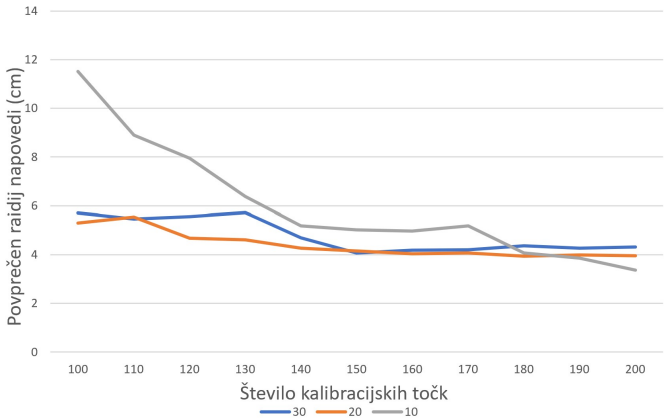
Za čim hitrejši proces kalibracije si želimo čim krajši čas prikaza točke. Ta sicer ne sme biti prekratek, saj si želimo, da uporabik udobno sledi menjavi točk. Pri oknu zajema je pomembno, da znotraj okna ne prihaja do večjih sprememb vhodnih parametrov (mirno oko). Hkrati si želimo, da je okno čim daljše, kar nam omogoča zajeti večje število podatkov.

Za nastavitev teh dveh parametrov bomo uporabili sledilec pogleda Tobii Pro nano. Na zaslonu prikazujemo točke in beležimo razdalje med dejansko točko na zaslonu in napovedano točko sledilca Tobii. Vsaka točka bo na zaslonu prikazana $2s$. Primer dobljenih rezultatov prikazuje graf na sliki 2. Ta graf prikazuje standardni primer preskoka očesa med kalibracijskima točkama. Po menjavi kalibracijskih točk potrebuje oko testiranca približno $0.25s$, da pogled preusmeri na novo točko. Med testom je v najslabšem primeru oko potrebovalo približno $0.5s$ za dokončno stabilizacijo, saj je preskosku sledil še manjši popravek. Graf na sliki 3 prikazuje primer preskoka, ki sta mu sledila kratek in malo daljši mežik očesa. Mežiki se lahko pojavijo kadarkoli v času prikaza točke, zato jih je potrebno upoštevati.

Na podlagi dobljenih rezultatov smo se odločili omejiti čas prikaza točke na $1.5s$, ter okno zajema na interval $[0.75, 1.5]$. Začetek okna zajema je nastavljen tako, ker želimo biti popolnoma prepričani, da je uporabnik pogled uspešno ustalil na točki, se pravi je dovolj oddaljen od točke preskoka. Konec okna je pomaknjen do konca prikaznega časa točke, saj iz pridobljenih podatkov ni videti, da bi kakovost fokusa v tem času padla. Za izbrano dolžino prikaza smo se odločili, ker nam omogoča zajem dovolj podatkov in uporabnik med kalibracijo ne dobi občutka čakanja.

5.2 Velikost in število kalibracijskih točk

Testirali smo tri velikosti kalibracijskih točk, s polmeri 30, 20 in 10 slikovnih pik. Za vsako velikost naredimo ločeno kalibracijo z 200 kalibracijskimi točkami. To je zgornja meja števila točk zaradi izbrane zgornje meje časa kalibracije in dožine prikaza ene točke. Takoj za kalibracijo zajamemo podatke novih 30 naključno generiranih točk, ki jih uporabimo kot testno množico. Za testiranje števila kalibracijskih točk razdelimo učno množico, ki smo jo pridobili med kalibracijo, na 20 podmnožic. Prva vsebuje 10 točk, vsaki naslednji pa dodajamo 10 točk tako, da zadnja vsebuje vse točke. Točke se dodajajo v enakem vrstnem redu, kot so bile zajete. Vpliv števila in velikosti kalibracijskih točk ocenimo iz grafa na sliki 4. Iz grafa je razvidno, da najboljši rezultat,



Slika 4: Graf povprečne napake napovedi v odvisnosti od števila kalibracijskih točk za 3 velikosti kalibracijskih točk.

$3,37$ cm, doseže kalibracija z 200 točkami, s polmerom 10

slikovnih pik. Graf je narisan le za učne podatke, ki vsebujejo med 100 in 200 kalibracijskih točk; rezultati množic z manj točkami so slabši in posledično neuporabni. Na grafu najbolj izstopa kalibracija s točkami polmera 10, katerih napaka je na začetku bistveno večja od ostalih, vendar hitro pade in da na koncu najboljše rezultate. Tukaj gre najverjetneje za visok delež šuma v zgodnjih podatkih, kar se nato z dodajanjem novih popravlja. Če primerjamo rezultate vseh treh velikosti kalibracijskih točk, opazimo, da z manjšanjem radija točke za 10 slikovnih pik napaka pade med 15 in 20 slikovnih pik. Manjšega polmera točke nismo testirali.

6. ZAKLJUčEK

Predstavili smo algoritem za oceno točke pogleda s spletno kamero. Z uporabo referenčnih točk in globoke nevronske mreže smo v najboljšem primeru dosegli napako $3,37$ cm. Na osnovi poskusov ugotavljamo, da naša metoda deluje pri različnih svetlobnih pogojih (različna jakost svetlobe, luč v ozadju), vendar bolje deluje pri močnejši osvetlitvi, brez ambientne osvetlitve pa metoda ni uporabna. Zaradi specifičnih omejitev predlaganega pristopa, bo tega težko direktno primerjati z drugimi. Za primerjavo bi morali omiliti omejitve ali testirati ostale pristope z enakimi omejitvami. V obeh primerih, bi za uspešno primerjavo potrebovali večje število ljudi. Predstavljen pristop je pogojno primeren za uporabo s ciljno aplikacijo, saj ne omogoča spremljanja branja.

7. REFERENCES

[1] J. Engelberts. PupilDetector. `https://github.com/jonnedtc/PupilDetector`, June 2019. Accessed: 2019-09-05, Commit: 3442026.

[2] D. W. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):478–500, March 2010.

[3] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874, 2014.

[4] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba. Eye tracking for everyone. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2176–2184, 2016.

[5] A. Papoutsaki, P. Sangkloy, J. Laskey, N. Daskalova, J. Huang, and J. Hays. Webgazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence-IJCAI 2016*, 2016.

[6] F. Timm and E. Barth. Accurate eye centre localisation by means of gradients. pages 125–130, 01 2011.

[7] R. Valenti, N. Sebe, and T. Gevers. Combining head pose and eye location information for gaze estimation. *IEEE Transactions on Image Processing*, 21(2):802–815, Feb 2012.

[8] R. Valenti, J. Staiano, N. Sebe, and T. Gevers. Webcam-based visual gaze estimation. In *International Conference on Image Analysis and Processing*, pages 662–671. Springer, 2009.

[9] P. Viola, M. Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3, 2001.

[10] J. Žabkar. Izdelava računalniškega sistema za prepoznavanje disleksije in pomoč dislektikom, ul fri, June 2019.

[11] P. Xu, K. A. Ehinger, Y. Zhang, A. Finkelstein, S. R. Kulkarni, and J. Xiao. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*, 2015.

[12] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Appearance-based gaze estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4511–4520, 2015.