POROCILO RESITVE NALOGE OPERACIJA PERISKOP

ANDREJ DOLENC

1. UVOD

Videi so eni najpomembnejših in najpopularnejših medijev v uporabi, tako za prenašanje pomembnih informacij kot tudi za zabavo. Z ogromnimi napredki na področju virtualne realnosti v zadnjih nekaj letih lahko pričakujemo, da bodo tudi tu videi igrali zelo pomembno vlogo, vendar pa novo področje s sabo prinese nove izzive. V virtualni realnosti namreč pogosto omogočimo uporabniku, da poljubno premika svojo glavo in ker mu ne želimo uničiti občutka imerzije, mu predvajamo video za vseh $360°$. Da pa mu to omogočimo, mu moramo pošiljati ogromno količino podatkov naenkrat če mu želimo zagotoviti dobro kvaliteto posnetka za vseh 360 stopinj. Zato je morda interesantno, da vnaprej predvidimo v katero smer bo uporabnik gledal ter mu pošljemo visoko kvaliteto le za predviden zorni kot.

V nalogi tako poskusimo na podlagi prejšnjih ogledov drugih uporabnikov nekega videa ter na podlagi nekaj sekundne zgodovine obračanja glave uporabnika napovedati, katere tokove se nam splača poslati, da mu zagotovimo čim boljšo izkušnjo.

V poročilu najprej predstavimo podane podatke ter omejitve naloge, nato opišemo in utemeljimo našo rešitev, za konec pa predstavimo še nekaj pomankljivosti naše naloge ki bi prišle v poštev v dejanski uporabi.

1.1. Opis podatkov ter omejitev naloge.

V nalogi smo morali najprej na nek pameten način definirati množico največ 100 tokov; t.j. terk $(\text{začetni kot}, \text{končni kot}, \text{kvaliteta})$. Nato je bilo potrebno za vsako (približno) desetinko sekunde za 4 sekunde vnaprej napovedati, katere izmed vnaprej pripravljenih tokov je potrebno uporabniku naložiti v predpomnilnik za zagotavljanje čim večje kvalitete videa znotraj uporabnikovega zornega kota. Ta je obsegal vsega skupaj 61 stopinj, torej po 30 stopinj na vsako stran od njegovega trenutnega pogleda. V ta namen smo dobili prejšnje 4 sekunde uporabnikovega dejanskega pogleda.

Poleg tega sta tu veljali še dve dodatni omejitvi: pasovna širina vseh prenesenih tokov v vsakem trenutku ne sme presegati vrednosti 10000, za vsakega ot 360 kotov pa moramo zagotoviti vsaj kvaliteto 1%.

Podani učni podatki so obsegali 86 t.i. sledi (množic terk (čas, kot uporabnika)), kjer je bila časovna resolucija podana na približno $1/10$ sekunde, kotna resolucija pa na 2 stopinji natančnosti.

Za testne podatke smo nato prejeli še 728 sledi po 8 sekund, pri čemer smo za prve 4 sekunde dobili podan eksakten kot uporabnika, naslednje 4 pa smo morali napovedati.

Pred začetkom snovanja algoritma se seveda izplača pregledati podatke. Graf 1.1 predstavlja relativne premike uporabnikov v kotih na desetinko sekunde. Iz tega grafa lahko razberemo, da največkrat uporabniki ne premikajo svoje glave, ali pa pogled premikajo relativno počasi. Razlog, da sta na histogramu stolpca $1°$ in $-1°$ tako nizka leži v tem, da je premik v podatkih podan na 2 stopinji natančno.

Date: July 31, 2018.

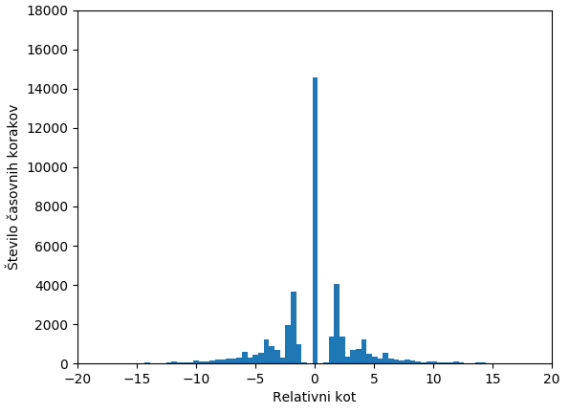

FIGURE 1.1. Histogram relativnih premikov v učni množici.

Grafi 1.2 prikazujejo vročinske slike (ang. *heat map*) pogledov uporabnikov iz učne množice. Pri tem so bile različne slike ustvarjene tako, da smo označili vseh $61°$, ki jih uporabniku prikažemo, ter kote spustili skozi okna. To lahko razložimo s tem, da v nekem trenutku uporabnika ne zanima le popolna sredina njegovega pogleda, temveč so pomembni tudi preostali koti, to pomembnost pa lahko določimo z različnimi okni; npr. Kronecker-delta okno predpostavi, da je pomembna zgolj sredina pogleda, pravokotno okno predvideva da je enako pomemben cel pogled, medtem ko Welch okno dodeli večjo pomembnost sredini pogleda in vedno manjšo pomembnost z oddaljenostjo od sredine.

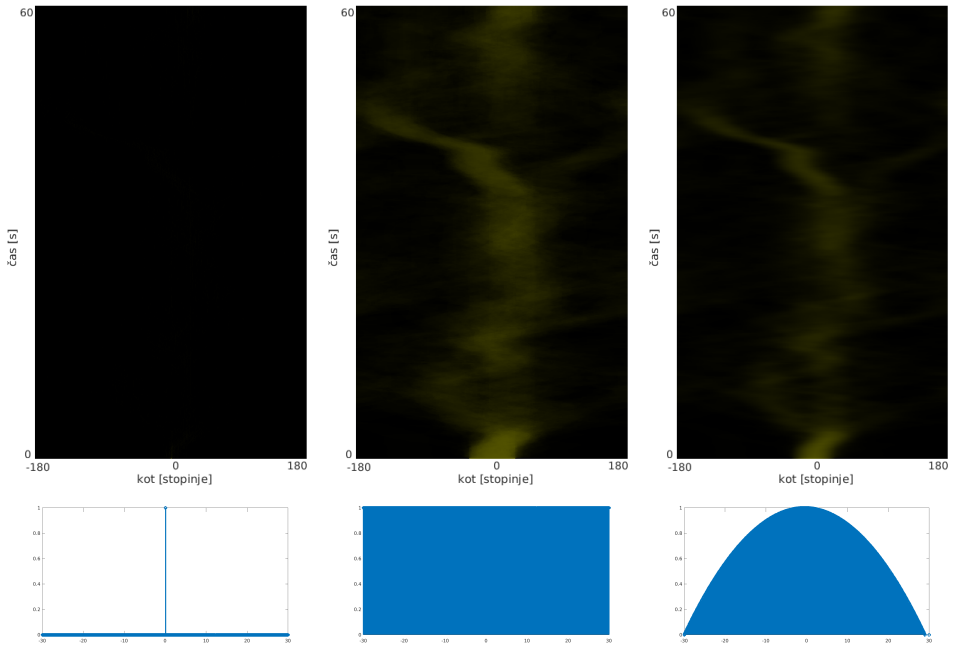

FIGURE 1.2. Vročinske slike izdelane z različnimi okni nad učno množico. Bolj rumena barva pomeni večjo vrednost. V zgornji vrstici od leve proti desni: izdelane s Kroneckerjevim oknom, s pravokotnim oknom in s Welchovim oknom. Pod njimi slike pripadajočih oken.

2. METODE IN REZULTATI

Nalogo smo razdelili na 2 dela: napovedovanje kota pogleda uporabnika, ter napovedovanje optimalnih uporabljenih tokov glede na kote. Glede na takšno delitev v tem poglavju tudi predstavimo našo rešitev. Začnemo pa z opisom metode za preverjanje kvalitete napovedi.

2.1. **Interno prečno preverjanje.**

Ker je bilo dovoljenih oddaj na lestvico le 10, smo morali implementirati interno prečno preverjanje (ang. *Cross validation*). Le tako smo lahko namreč učinkovito določili hiperparametre za uporabljene algoritme ter jih primerjali med seboj.

Prečno preverjanje je v tem primeru delovalo tako, da smo nekaj tokov iz učne množice odstranili iz nje ter jih razbili na odseke dolge po 8 sekund, pri čemer smo za zadnje 4 sekunde odstranili kote. Pri rezanju tokov smo zaradi količine podatkov dovolili tudi, da se izrezani tokovi prekrivajo. Poleg tega smo namesto računanja rezultata s formulo, ki je bila uporabljena za lestvico, raje računali povprečno kvaliteto uporabnikovega pogleda.

2.2. **Napovedovanje kota pogleda.**

Za napovedovanje kota pogleda smo preizkusili 3 različne strategije.

2.2.1. *Konstantni kot.*

Prva strategija je temeljila na podatkih pridobljenih iz vročinskih slik 1.2. Iz teh lahko vidimo, da uporabniki zelo radi gledajo posnetek v okolici kota $0°$. Tako smo za vsak kot napovedali kar konstantni kot, kjer je bila vsota stolpca vročinske slike največja. Iz slik lahko še opazimo, da se optimalen konstantni kot spreminja glede na uporabljeno okno, torej je izbira tega precej pomembna.

2.2.2. *Zadnji znani kot.*

Druga strategija je temeljila na histogramu 1.1, iz katerega lahko razberemo, da se pogledi uporabnikov večinoma ne spreminjajo veliko skozi čas. Na podlagi tega je bila ideja, da je napovedovanje zadnjega znanega kota pogleda uporabnika zelo dobra aproksimacija dejanskega pogleda za naslednjih nekaj desetink sekunde.

2.2.3. *Seam Carving.*

Navdih za zadnjo strategijo je prišel s strani fantastičnega algoritma uporabljenega za skaliranje slik, pri čemer na sliki pomembnejši deli ohranjajo pravo razmerje. Algoritem je originalno znan pod imenom *Seam Carving* [1], oziroma *Content Aware Scaling* v programskem orodju Photoshop. Deluje tako, da na sliki najprej vsakemu pikslu določi 'energijo' (npr. velikost gradienta), nato pa iz zgornje vrstice pikslov na sliki poišče pot ki minimizira to energijo do spodnje, pri čemer se lahko pomika iz posameznega piksla le v tri neposredno v vrstici pod njim. Nato to pot preprosto izreže iz slike in ta postane tako za en piksel ožja.

Drugače povedano za naše potrebe lahko algoritem iz vsakega para (θ, t) določi optimalno pot skozi video z začetkom v podanem paru ter koncem v $t' = 60s$.

V naši verziji namesto energije tako uporabimo kar vročinsko sliko, ki smo jo predstavili v 1.2, pri čemer smo mi zainteresirani v maksimizaciji energije. Algoritem iskanja optimalne poti nato dejansko temelji na zelo preprosti uporabi dinamičnega programiranja. Našo modificirano verzijo lahko opišemo z naslednjo enačbo:

$$sc(\theta, t) = heatmap(\theta, t) + \begin{cases} 0 & \text{če } t = 60, \\ \max\{mp(\theta' - \theta) \cdot sc(\theta', t+1)\}_{\theta' \in \theta + [-\alpha, \alpha]} & \text{sicer,} \end{cases}$$

kjer so:

- θ ... aktualni kot,
- t ... aktualni čas,
- $heatmap$ ... lookup tabela za izračunano vročinsko sliko (glej 1.2),
- mp ... funkcija ki glede na relativni kot pove 'verjetnost' premika (uporabljena kot utež premika), in
- α ... parameter ki določa, koliko kotov na vsako stran preverimo.

Racionalizacija za uporabljeno vročinsko sliko je ista, kot je bila v opisu prvega algoritma, podobno ideja za funkcijo mp stoji za analizo histogramov relativnih premikov uporabnikov (glej 1.1). S tem smo nekako združili dobri ideji obeh prvih algoritmov v enega samega. Smo pa se morali precej igrati z izbiro okna oz. parametrov. Naivna uporaba pravokotnega okna za vročinske slike ter parametri ki zagotovijo čim boljše prileganje Gaussove distribucije k dejanskim podatkom namreč niso dali najboljših rezultatov.

Na delovanje algoritma torej lahko vplivamo na nekaj načinov:

- s spremembo uporabljenega okna pri izdelavi vročinske slike,
- s spremembo resolucije vročinske slike,
- s spremembo funkcije mp, ter
- s spreminjanjem parametra α.

Na vse od teh parametrov smo gledali kot na hiperparametre ter iskali optimalne vrednosti s pomočjo postopka opisanega v naslednjem podpoglavju.

2.3. **Ocenjevanje napovedanih kotov.** Ker za ocenjevanje kvalitete z uporabo implementiranega prečnega preverjanja potrebujemo tudi tokove, smo kvaliteto napovedanih kotov interno preverjali malce drugače. Podobno kot pri prečnem preverjanju smo izdelali učno in validacijsko množico, vendar pa smo tokrat gledali povprečno $L1$ razdaljo med napovedanim ter dejanskim kotom v zadnjih štirih sekundah tokov. Graf 2.1 prikazuje povprečno odstopanje v stopinjah v odvisnosti od časa. Tu je lepo razvidno, da zadnja metoda dejansko rezultira v najpravilnejše napovedi kotov.

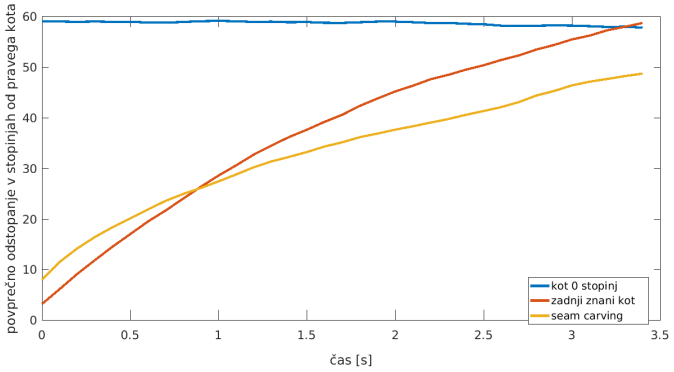

FIGURE 2.1. Graf odstopanja napovedanega kota od dejanskega kota v odvisnosti od časa za različne metode.

V našem primeru smo za zmagovalno rešitev torej uporabili algoritem, ki je temeljil na tretji strategiji. Pri tem podatke najprej prevzorčili na časovno resolucijo $1/10$ sekunde, ter kotno resolucijo $1°$. Za izdelavo vročinske slike smo uporabili Welchovo okno, za dinamično programiranje pa smo gledali kote 4 stopinje v vsako smer od trenutnega kota, z verjetnostjo premika definirano kot Gaussova funkcija s povprečjem 0 in standardnim odklonom $\sqrt{80}$. Izgled poti, ki jih algoritem ubere je razviden is slike 2.2

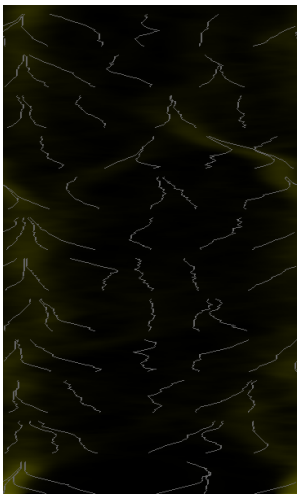

FIGURE 2.2. Graf izbranih poti končnega algoritma. Z belo barvo so izrisane poti na vročinski sliki.

2.4. **Napovedovanje optimalnih uporabljenih tokov.** Drugi del naloge je predstavljala specifikacija največ 100 tokov. Tu smo izhajali iz ideje, da želimo za podane kote pri prvih 4 sekundah tokov uporabniku zagotoviti optimalno izkušnjo: video kvalitete 100% za celoten pogled. Na podlagi tega smo izdelali 90 tokov kvalitete 100% ter kotnega razpona $4°$: $\{(4 \cdot \theta, 4 \cdot \theta + 4 - 1, 100\%) ; \theta \in [0, 89]\}$, pri čemer smo lahko nato za vsak kot izračunali optimalne tokove, ki so nam zagotovili čim večji razpon na obe smeri (pri omejitvi, da ne presežemo skupne pasovne širine).

Ker pa na ta način ne pokrijemo vseh $360°$ pa smo si tu ustvarili še pomožen "dummy" tok, ki je zgolj izpolnjeval zahtevo, da je kvaliteta na vsakem od 360 kotov vsaj ena stopinja: $(0, 359, 1\%)$.

Tokovi kvalitete 100% lahko pokrijejo le kot $96°$ (360 pasovne širine potrebujemo za dummy tok). Ker smo na sliki 2.1 odkrili da povprečno optimalen kot zgrešimo za precej več, smo tu poskusili tudi na pameten način uporabljati tokove z več različnimi kvalitetami. Če bi namreč uporabljeno kvaliteto postopoma zniževali bi lahko pokrili večji kot in tako ne bi za napačne napovedi prejeli le ene točke. Vendar pa se je vsak poskus nadgradnje uporabljene strategije izkazal za slabšega, tako da smo na koncu uporabili kar opisanih 90 100% tokov.

3. ZAKLJUČEK

V poročilu smo opisali pristop, ki se je na javni lestvici izkazal za najboljšega. Žal pa ne pride brez pomanjkljivosti: kjer pravilno napovemo uporabnikov zorni kot bo sicer uporabnik prejel idealno kvaliteto, kolikor hitro pa se odloči za preveč odtavati od naše napovedi, pa bo prejel kvaliteto 1%.

Naš pristop najverjetneje tudi ni najboljši kar se tiče racionalizacije. Kljub temu, da smo z mahanjem rok lahko na nek način upravičili njegovo zasnovo, v resnici deluje precej drugače kot pa izgleda dejanski ogled videa s strani uporabnika. S tem ko računamo poti, ki maksimizirajo energijo, namreč naš algoritem 'vidi v prihodnost' in lahko pove, kje se bodo zgodile najbolj zanimive scene. Po drugi strani uporabnik tega ne more nujno storiti in obrača glavo le na podlagi trenutne slike.

REFERENCE

[1] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Trans. Graph.*, vol. 26, July 2007.