

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Gregor Majcen

**Interaktivne računalniške umetniške  
instalacije v času hitrega tehnološkega  
razvoja**

MAGISTRSKO DELO  
ŠTUDIJSKI PROGRAM DRUGE STOPNJE  
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: prof. dr. Franc Solina

Ljubljana, 2015



AVTORSKE PRAVICE. Rezultati magistrskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov magistrskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

©2015 GREGOR MAJCEN



## IZJAVA O AVTORSTVU MAGISTRSKEGA DELA

Spodaj podpisani Gregor Majcen sem avtor magistrskega dela z naslovom:

*Interaktivne računalniške umetniške instalacije v času hitrega tehnološkega razvoja*

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod mentorstvom prof. dr. Franca Soline,
- so elektronska oblika magistrskega dela, naslov (slovenski, angleški), povzetek (slovenski, angleški) ter ključne besede (slovenske, angleške) identični s tiskano obliko magistrskega dela,
- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki “Dela FRI”.

V Ljubljani, 13. avgusta 2015

Podpis avtorja:



## ZAHVALA

*Hvala mentorju prof. dr. Francu Solini za vso strokovno pomoč in material za izdelavo magistrskega dela.*

*Hvala neuradnemu somentorju viš. pred. dr. Borutu Batagelju za pomoč tako pri izdelavi praktičnega dela magistrskega dela kot tudi pri dobavi strojne opreme.*

*Največja zahvala gre ženi Mateji za spodbudo, podporo in zabavo, ki je nastala med testiranjem praktičnega dela.*

*Hvala tudi moji in ženini družini za vso spodbudo, da zmorem.*

*Hvala prijatelju in sodelavcu Maticu Potočniku za komentarje, ideje in debate o tem, kaj vse bi se iz te naloge še dalo narediti.*

*Hvala tudi Teji, ki je to delo lektorirala, da branje lepše in lažje teče.*

*Gregor Majcen, 2015*



# Kazalo

<b>Povzetek</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>1 Uvod</b>	<b>1</b>
<b>2 Andy Warhol, popart umetnost in naša instalacija</b>	<b>5</b>
2.1 Princip delovanja instalacije . . . . .	6
<b>3 Prvotna izvedba: 15 sekund slave</b>	<b>9</b>
3.1 Strojna in programska oprema . . . . .	9
<b>4 Ohranjanje digitalne umetnosti</b>	<b>13</b>
<b>5 Nadgradnja: Naprava in pripomočki</b>	<b>15</b>
5.1 Testna naprava: Samsung Galaxy S5 . . . . .	15
5.2 Android . . . . .	16
<b>6 Nadgradnja: Zaznavanje obrazov</b>	<b>19</b>
6.1 Zaznavanje obrazov z uporabo OpenCV . . . . .	19
6.2 Zaznavanje obrazov z uporabo Android SDK . . . . .	23
<b>7 Nadgradnja: Obdelava slik</b>	<b>25</b>
7.1 GIMP . . . . .	25
7.2 Izbira filtrov . . . . .	25
7.3 Implementacija filtrov . . . . .	26

*KAZALO*

<b>8 Nove funkcionalne možnosti</b>	<b>43</b>
8.1 Dve sliki . . . . .	43
8.2 Obraz kot mozaik iz majhnih obrazov . . . . .	44
<b>9 Povezovanje s socialnimi omrežji</b>	<b>47</b>
9.1 Možnosti . . . . .	48
9.2 Implementacija s socialnim omrežjem Facebook . . . . .	48
9.3 Omejitve . . . . .	49
<b>10 Sklepne ugotovitve</b>	<b>51</b>





# Seznam uporabljenih kratic

kratica	angleško	slovensko
<b>LCD</b>	Liquid crystal display	Zaslon s tekočimi kristali
<b>SIM</b>	Subscriber identity module	Identifikacijska kartica za storitve mobilne telefonije
<b>HDMI</b>	High-Definition Multimedia Interface	Vmesnik za prenašanje multimedijalne vsebine v visoki ločljivosti
<b>API</b>	Application programming interface	Programski vmesnik
<b>SDK</b>	Software development kit	Komplet programskih orodij za razvijanje programske opreme
<b>JIT</b>	Just-in-time compilation	Dinamični prevajalnik, ki prevaja kodo med izvajanjem programa
<b>AOSP</b>	Android Open Source Project	Odprtokodni projekt Android
<b>JNI</b>	Java Native Interface	Izvorni javanski vmesnik
<b>OpenGL</b>	Open Graphics Library	Knjižnica za računalniško grafiko
<b>ES</b>	Embedded system	Namenski sistem
<b>GIMP</b>	GNU Image Manipulation Program	GNU program za predelovanje slik
<b>HSL</b>	Hue, saturation, lightness	Barvni odtenek, intenzivnost, svetlost



# Povzetek

V okviru magistrske naloge je bila ponovno implementirana in nadgrajena interaktivna umetniška instalacija ‐15 sekund slave‐. Motivacija za instalacijo je umetniško delovanje ameriškega popart umetnika Andyja Warhola. ‐15 sekund slave‐ izgleda kot klasična slika, a je dejansko računalniški zaslon, okvirjen kot umetniška slika. Nad zaslonom je v okviru vgrajen digitalni fotoaparat, ki je povezan z računalnikom v ozadju. Vsakih petnajst sekund fotoaparat slika obiskovalce galerije, ki stojijo pred sliko. Na sliki računalniški program poišče vse obraze in nato naključno izbere enega izmed njih. Ta obraz nato z grafičnimi filtri program obdela tako, da pridobi tako imenovani popart videz z manjšim številom živih barv, ki spominjajo na slike slavnih osebnosti, ki jih je iz fotografij delal Andy Warhol. Ker je prvotna instalacija nastala pred več kot desetimi leti in se je strojna oprema v tem času že zelo spremenila, se je pokazala potreba po prilagoditvi aplikacije novemu stanju tehnologije. V nalogi so opisani problemi pri vzdrževanju novomedijskih umetniških instalacij in kako smo le-te rešili v primeru opisane instalacije.

## Ključne besede

*računalnik, umetnost, pametni telefon, android, filter, digitalna umetnost, ohranjanje, vzdrževanje programske opreme*



# Abstract

This thesis describes the reimplementations of the interactive art installation entitled “15 seconds of fame”. The motivation for the installation was the famous American artist Andy Warhol. He modified simple photograph portraits into pop-art pieces. The production of such portraits is the goal of the installation “15 seconds of fame”. The installation looks like a picture on a wall, but it is in fact a computer monitor framed as a picture. On the top of the frame there is a small camera which is connected to a computer hidden behind the scene. Every fifteen seconds, the camera takes a picture of the scene in front of it and sends it to the computer, where all the magic happens. It searches for human faces in the picture, picks one face randomly and applies a combination of image filters which converts the face into a pop-art portrait. The idea is still great, even ten years later, but the original technology is outdated and increasingly difficult to maintain. We reimplemented the installation with new hardware and software and we discuss the problems of maintaining new media art installations in general.

## Keywords

*computer, art, smartphone, android, filter, digital art, conservation, software maintenance*



# Poglavlje 1

## Uvod

Magistrska naloga sodi na relativno novo področje vzdrževanja novomedijskih umetniških instalacij, ki za svoje delovanje potrebujejo računalniško tehnologijo [1]. Umetniki so se za ta način ustvarjanja odločili že kmalu po pojavu prvih računalnikov, vendar niso bili zadovoljni z osnovnimi aplikacijami, saj so želeli pri svojih umetninah prikazati nekaj več [5]. Tu pa se je pričelo sodelovanje z računalniškimi strokovnjaki [6].

V magistrski nalogi bomo posodobili interaktivno umetniško instalacijo ‐15 sekund slave‐. Najprej se bomo v 2. poglavju seznanili z motivacijo za to instalacijo in spoznali, kako instalacija deluje.

V 3. poglavju se seznanimo, kako je bila instalacija prvotno implementirana.

Da bi se naloge pravilno lotili, smo v 4. poglavju preučili širšo problematiko vzdrževanja novomedijskih umetniških del. Kako sploh vzdrževati umetniško instalacijo, ki temelji na računalniški tehnologiji. Kakšni so principi in strategije. Novejše v tem kontekstu ne pomeni vedno boljše. Zaradi hitrega napredka računalniške tehnologije je potrebno take aplikacije po eni strani prilagajati novi strojni in sistemski programski opremi, pa tudi novim funkcionalnim možnostim, ki jih nove tehnologije nudijo. Po drugi strani pa z umetniškega vidika običajno želimo, da se zunanjja pojava umetniškega dela ne spremeni.

V 5. poglavju smo po preučevanju teorije, kako se sploh pravilno lotiti nadgradnje umetniške instalacije ‐15 sekund slave‐, začeli razmišljati še s praktičnega vidika. V desetih letih, odkar je bila prvotna instalacija narejena, je računalniška tehnologija močno napredovala, še posebej v razvoju mobilnih naprav. Razmišljali smo, kako bi novo tehnologijo lahko prenesli v trenutno instalacijo. Odločili smo se za mobilni telefon. Pametni telefoni so enostavno in prosto dostopni na trgu in ker se masovno izdelujejo, tudi niso zelo dragi. Vsebujejo pa vse, kar potrebujemo: kamero, prikaz slike in programabilnost. Ostale lastnosti so le bonus. Sedaj je bilo potrebno izbrati le še operacijski sistem. Zaradi osebnih izkušenj smo se odločili za operacijski sistem Android (5.2). Po izbiri strojne opreme smo začeli z načrtovanjem, kako prenesti vso obstoječo programsko funkcionalnost.

V 6. poglavju smo se soočili z zaznavanjem obrazov. Načini, kako zaznavati obraze na sliki, so se v desetih letih močno spremojali. Obstaja tudi že veliko orodij, ki nam pomagajo pri tem. Naša izbira je na koncu pristala na dveh orodjih: OpenCV (6.1) in AndroidSDK (6.2). Ti dve orodji temeljita na popolnoma različnih strategijah, zato smo se odločili poskusiti obe.

V 7. poglavju smo obravnavali obdelavo slik. Že pri prvotni instalaciji je bilo izhodišče za vse filtre programsko orodje GIMP (7.1). V tej instalaciji smo se odločili, da si bomo tudi pomagali s tem programskim orodjem, vendar z različico iz današnjega časa. Iz programskega orodja GIMP smo si izbrali tri vrste filtrov:

- uravnovešanje barv (7.3.3),
- posteriziranje (7.3.4),
- uravnovešanje barv v prostoru HSL (7.3.5).

Različni parametri in mešanje teh filtrov med seboj so nas pripeljali do popart slik (7.3.6).

S tem je bila prestavitev iz originalnega sistema v mobilni sistem končana. Vendar pa smo želeli še več. V 8. poglavju opisujemo nove funkcionalne možnosti, ki smo jih dodali za popestritev instalacije.

V 9. poglavju pa smo se ukvarjali z najbolj pomembno lastnostjo, ki je pravi hit sedanjega časa, in sicer s povezavo s socialnimi omrežji. Mladim in vse bolj tudi starejšim postaja pregledovanje socialnega omrežja vsakdanje opravilo. In trenuten trend ‐selfijev‐ je kot nalašč za našo instalacijo.



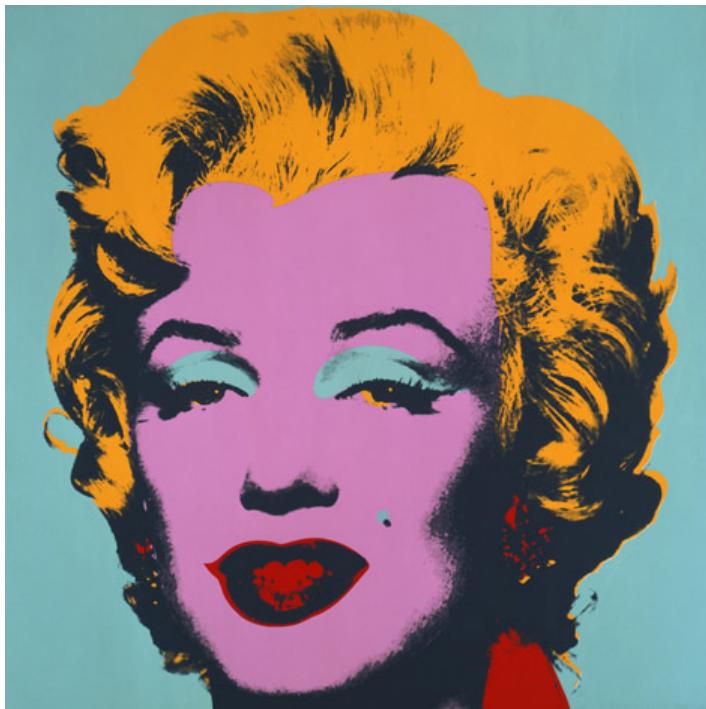
## Poglavlje 2

# Andy Warhol, popart umetnost in naša instalacija

Andy Warhol se je rodil 6. avgusta 1928 v Pittsburghu v Pensilvaniji staršema Slovakoma [21]. Njegovo otroštvo je bilo zelo težko, saj je v času velike gospodarske krize oče ostal brez službe. Kljub temu je Andy po osnovnem šolanju obiskoval tečaj komercialnega oblikovanja na Carnegijevem inštitutu za tehnologijo v Pittsburghu. Pri 21 letih se je preselil v New York, kjer je izdal serijo ilustracij za čevlje. Kmalu je postal uspešen in dobro plačan slikar, vendar pa je hrepenel po umetniški slavi. Po številnih neuspehih se mu je leta 1964 pri razstavi Ameriški supermarket le posrečilo. Ta razstava je bila ena izmed prvih, ki je javnost seznanila s pop artom. Nato se je njegova kariera le še vzpenjala.

Njegove najbolj znane slike so Dick Tracy, Plesna shema (tango), Ploče-vinka juhe Campbell, Zvitek bankovcev in serija portretov znanih ljudi, kot sta Elvis Presley in Marilyn Monroe. Popart portret Marilyn Monroe lahko vidimo na sliki 2.1.

Andy Warhol je umrl v spanju zaradi srčne kapi 22. februarja 1987, star 58 let.



Slika 2.1: Popart slika umetnika Andyja Warhola [18]

## 2.1 Princip delovanja instalacije

“15 sekund slave” je interaktivna umetniška instalacija, ki obraz naključno izbranega obiskovalca galerije povzdigne v umetniški objekt za petnajst sekund [10]. Instalacija je bila navdahnjena s slavnim citatom Andyja Warhola: “V prihodnosti bodo vsi ljudje doživeli svojih petnajst minut slave.”<sup>1</sup> pa tudi z njegovim načinom predelave obrazov v slogu popart [14].

Če pogledamo sliko 2.2, vidimo le “mojstrovino”, uokvirjeno v dragocen okvir. V resnici pa se za to podobo skriva še mnogo več.

Nad sliko obraza vidimo črno luknjo, to je prostor za digitalni fotoaparat, ki enkrat na vsakih petnajst sekund fotografira okolico in fotografijo pošlje računalniku, ki je skrit v ozadju. Fotoaparat ima širokokoten objektiv, da lahko zajame čim širšo okolico.

---

<sup>1</sup>Originalno v angleščini: “In the future everybody will be world-famous for fifteen minutes.”[19, 9].



**Slika 2.2:** Prikaz delovanja instalacije

Vsako novo fotografijo nato pošljemo računalniku, ki izvede algoritem za iskanje obrazov. Če je teh več, enega izberemo po principu naključnega izbora. Ta obraz izrežemo iz fotografije in ga pošljemo kot vhod naslednjemu algoritmu.

Izbran obraz se obdela z enim izmed vnaprej določenih barvnih filtrov. Ti filtri so sestavljeni iz različnih barvnih transformacij, da preoblikujejo fotografijo v slogu popart.

Slika obraza v slogu popart je končni rezultat, ki je prikazan takoj, ko prejšnjemu obrazu poteče njegovih petnajst sekund. In kako je prikazan? Pod luknjo za fotoaparat je izrezan še večji pravokotnik, za katerim se skriva računalniški zaslon LCD, povezan z računalnikom, ki je opravil transformacijo fotografije.

Ta postopek se ponavlja v neskončnost, posebnost instalacije pa je, da se vse dogaja v živo. Če bo torej nekdo dovolj dolgo gledal v okvir, ima veliko možnosti, da bo kmalu zagledal samega sebe kot "mojstrovino" v galeriji [4].

*POGLAVJE 2. ANDY WARHOL, POPART UMETNOST IN NAŠA  
INSTALACIJA*

---

# Poglavlje 3

## Prvotna izvedba: 15 sekund slave

Kot že omenjeno, je prvotna instalacija nastala pred več kot desetimi leti. Ker je težko nadomestiti le posamezne dele v sistemu in ker je tehnologija v teh letih močno napredovala, se je pokazala potreba po prilagoditvi aplikacije novemu stanju tehnologije [15].

### 3.1 Strojna in programska oprema

**Strojna oprema.** Prva razstavljena različica je imela bogato okrašen lesen okvir, ki pa ni bil prenosen. Za tem okvirjem se je skrival računalniški zaslon LCD velikosti 17 palcev, malo nad tem zaslonom pa je bil fiksiran digitalni fotoaparat Olympus C3020 ZOOM (objektiv tipa 32–96mm, maksimalna ločljivost je  $2048 \times 1536$ ) [13].

Ob selitvi instalacije se je pripetila prva nadgradnja. Fiksen lesen okvir se je zamenjal z razstavljivim in prenosljivim. V tem času je tudi tehnologija naredila korak naprej. Naprave so postajale vse manjše in zmogljivejše. 17-palčni računalniški zaslon je nadomestil 19-palčni, kljub temu pa se sama velikost naprave ni povečala. Zamenjal se je tudi digitalni fotoaparat. Zaradi programske opreme je bil najprimernejši kar Olympus, in ker je bil še vedno



Slika 3.1: 15 sekund slave slike umetnika Andyja Warhola [11]

željen široki kot objektiva, je to napravo nadomestil Olympus C40 ZOOM (objektiv tipa 35–98mm, maksimalna ločljivost  $2272 \times 1704$ ).

Omenjeno je bilo, da je bil Olympus primeren zaradi programske opreme. Ta digitalni fotoaparat je omogočal upravljanje same naprave programsko preko računalnika. Orodje SDK omogoča nastavljanje ISO, odprtosti zaslonke, hitrosti slikanja, fokusa, možnosti zajemanja slike ... S tem je omogočeno avtomatsko zajemanje nove slike vsakih petnajst sekund.

**Programska oprema.** Modul za detekcijo obrazov je bil napisan v programskem jeziku C++. Sprva je algoritem iskal obraze na podlagi barve kože. Zaradi hitrega in učinkovitega delovanja je bila sama slika pomanjšana na resolucijo  $160 \times 120$  pikslov. Najmanjši obraz, ki je bil lahko najden, je velikosti  $11 \times 12$  pikslov, največji pa  $96 \times 106$  pikslov. Zaznavanje je bilo zelo občutljivo na svetlobo, še posebej ob večjih spremembah med samim delovanjem. Za zmanjšanje tega problema so se uporabljale različne metode za kompozicijo

osvetlitve [3]. Kasneje se je celoten algoritmom za detekcijo obrazov zamenjal z algoritmom Viola-Jones [16], ki pa ni bil več odvisen od barve kože. Najmanjši obraz pri algoritmu Viola-Jones je bil velikosti  $24 \times 24$  pikslov.

Modul za filtre, s katerimi so se ustvarjali popart efekti, je bil prav tako napisan v programskem jeziku C++. Slika obraza oziroma vhod, ki ga je algoritmom dobil, je bila slika velikosti  $400 \times 400$  pikslov. V kolikor je bil rezultat detekcije obraza manjši, se je slika povečala na to velikost.

Komunikacija s samo strojno opremo in vsemi moduli je bila napisana v programskem jeziku Pascal/Delphi. Celotna instalacija je bila narejena in testirana na operacijskem sistemu Windows XP [13].



## Poglavlje 4

# Ohranjanje digitalne umetnosti

Ohranjanje digitalne umetnosti je zelo pomembna vrednota v svetu umetnosti [8, 13]. Navezuje se na umetnost, ki je sicer že v digitalni obliki, a mora zaradi zelo hitrega razvoja tehnologije vseeno ostati v koraku s časom. Novejša tehnologija nam omogoča zmogljivejše in hitrejše naprave, manjšo porabo energije, bolj točne algoritme in še veliko drugih pomembnih izboljšav. Vendar pa umetnost ni le tehnologija, umetnost je nekaj, kar naj bi se ohranjalo skozi leta, desetletja, stoletja in še dlje.

Pri digitalni umetnosti se pojavi kar nekaj težav, ki jih pri ohranjanju običajnih umetniških izdelkov ni. Če nismo v koraku s časom, je možnost, da strojna oprema, ki jo potrebujemo za instalacijo, ni več dobavljava na trgu ali pa je že dovolj stara, da ima zgodovinsko vrednost. Nova strojna oprema doprinese še programske posodobitve, nove principe, standarde in različna delovanja. Zato je res pomembno, da sledimo razvoju, obenem pa se še vedno trudimo, da zunanji obiskovalec kljub novi tehnologiji misli, da je to ista, prvotna instalacija.

Četudi želimo zamenjati le strojno opremo, to v zelo starih instalacijah ni mogoče. Stara programska oprema ne deluje vedno na novejših sistemih. V tem primeru smo prisiljeni ponovno implementirati algoritme, tako da so združljivi s strojno opremo. Tukaj obstaja več možnosti. Vedno lahko uporabljamo staro kodo z manjšimi ali pa tudi večjimi popravki, tako da

se ta prevede tudi na novejših sistemih. Vendar pa je tu treba biti pozoren na to, da si s tem ne povzročimo še več dela, kot če bi vse skupaj ponovno implementirali.

Ko že govorimo o novi strojni in programski opremi, omenimo še drugo težavo. Zaposleni v muzejih in galerijah običajno niso izobraženi za vzdrževanje digitalne umetnosti, saj ti ljudje načeloma niso strokovnjaki računalniške stroke. Posledično so možne težave pri rokovanju s tako umetnino, zato bi bilo priporočljivo, če bi bila instalacija podrobno dokumentirana in bi bilo tako pojasnjeno, kako jo postaviti, kako jo vzdrževati in kaj storiti v primeru težav.

Splošno gledano obstajata dve strategiji za ohranjanje digitalne umetnosti [8]:

- Ohraniti želimo vso strojno in programsko opremo, dokler je to le mogoče. S tem zagotovimo, da je uporabniška izkušnja ves čas enaka. Tukaj bi se večina lahko vprašala, zakaj ne bi strojne opreme s časom nadgrajevali. Vzemimo za primer računalniški zaslon. Res je, tehnologija LCD je zelo napredovala, slika je v primerjavi s sliko na zaslonih s katodno cevjo precej bolj ostra in čista. Vendar pa ravno ta ostrina lahko popolnoma spremeni končni produkt, saj je bil morda nekdo nad umetnino navdušen ravno zaradi teh zabrisanih robov, ki so ji dajali nekoliko starinski pridih.
- Nadgradimo strojno in programsko opremo. Pri tem moramo biti zelo pazljivi, da uporabnik še vedno dobi tisto, kar je dobil pred nadgradnjo. Umetniška instalacija mora ostati kar se da podobna prejšnji, vključno z vsemi funkcionalnostmi. Vendar pa se pogosto zgodi, da avtor skozi čas pridobi nove ideje za posodobitve, velikokrat gre tako za nekaj, kar prej s staro tehnologijo sploh še ni bilo mogoče. Lep primer take ideje je integracija s socialnimi omrežji.

## Poglavlje 5

# Nadgradnja: Naprava in pripomočki

Prvi motiv za nadgradnjo, ki smo si jo zamislili, je čim bolj učinkovito zmanjšati porabljen prostor za instalacijo in s tem olajšati njeno prenosljivost in povečati enostavnost postavitve.

Prvotna instalacija potrebuje osebni ali prenosni računalnik, zaslon LCD, digitalni fotoaparat, mrežno opremo in dostop do interneta.

Kot najboljša zamenjava se je izkazala naprava ‘pametni telefon’, saj ima vse prej naštete naprave združene v eno. Ima vgrajeno kamero, ki služi kot fotoaparat, dostop do interneta preko kartice SIM ter operacijski sistem, ki omogoča izdelavo programske opreme, s katero lahko dostopamo do prej naštetih funkcij.

Pametni telefoni višjega cenovnega razreda omogočajo tudi, da lahko sliko v živo predvajamo na večjem ekranu. Zato smo se odločili, da kot testno napravo vzamemo pametni telefon Samsung Galaxy S5.

### 5.1 Testna naprava: Samsung Galaxy S5

Samsung Galaxy S5 je pametni telefon, narejen v podjetju Samsung. Deluje z operacijskim sistemom Android verzije 4.4.4, imenovanim tudi Android



(a) Kamera z bliskavico

**Slika 5.1:** Testna naprava: Samsung Galaxy S5

KitKat. Trenutno najnovejša obstoječa verzija operacijskega sistema Android je 5.0, poimenovan Lizika (*angl. Lollipop*).

Samsung Galaxy S5 se ponaša s štirijedrnim procesorjem 2.5GHz Krait 400 na čipu Qualcomm MSM8974AC Snapdragon 801. Ta procesor je dovolj zmogljiv za preračunavanje vseh filtrov v zadovoljivem času.

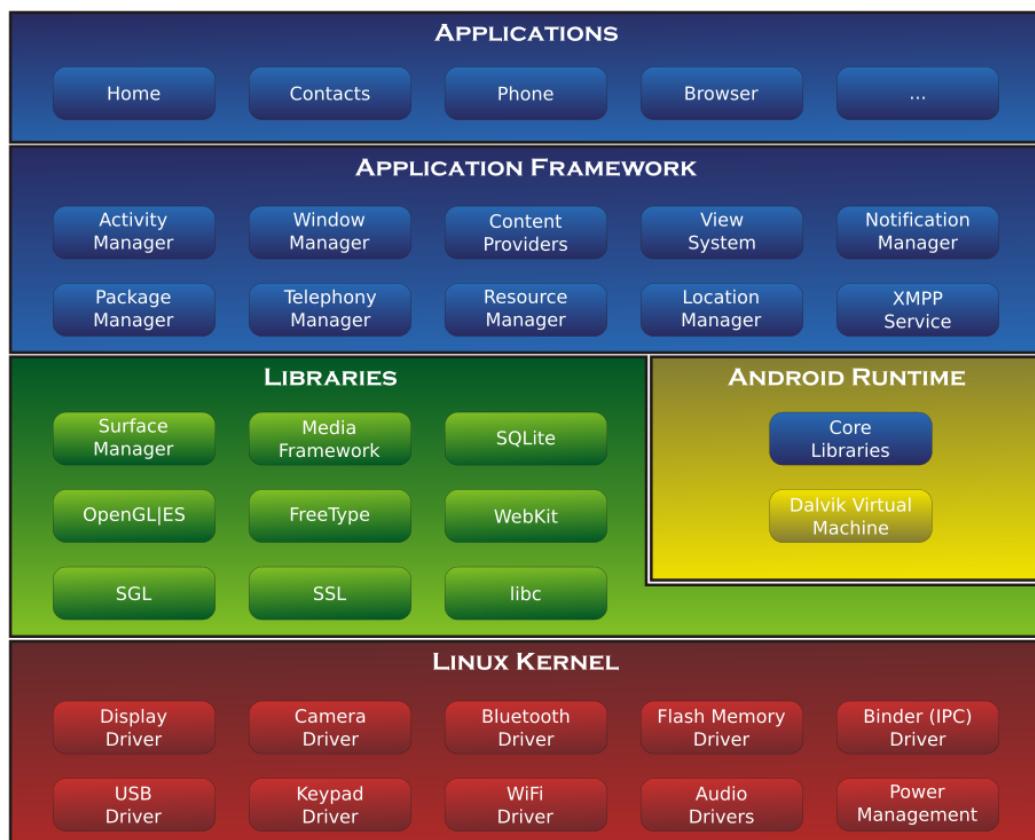
Grafična enota, Adreno 330, omogoča prikazovanje v ločljivosti  $1080 \times 1920$  pikslov. Ena izmed pomembnih lastnosti je tudi možnost povezave zunanje naprave preko priključka HDMI.

Tudi kamera je izjemno kvalitetna, je namreč tipa 1/2.6". Svetlobno tipalo ima 16 megapikslov in lahko torej naredi sliko v velikosti  $5312 \times 2988$  pikslov. Velikost in obliko kamere lahko vidimo kot majhen črn kvadrat, viden na desni sliki 5.1.

Pomembna lastnost je tudi operacijski sistem Android, za katerega je v programskem jeziku Java mogoče napisati programsko opremo, imenovano aplikacija. Vse, kar za to potrebujemo, je Android SDK in prevajalnik Java.

## 5.2 Android

Android je odprtokodni sistem za pametne telefone, ki ga je leta 2007 izdelal Open Handset Alliance pod vodstvom podjetja Google. Projekt se imenuje



Slika 5.2: Struktura operacijskega sistema Android [20]

AOSP<sup>1</sup> in se še danes zelo hitro razvija.

Temelji na jedru Linux, najbolj bistven del arhitekture pa je navidezni stroj (*angl. virtual machine*). Bolj podrobna zgradba sistema je prikazana na sliki 5.2. Navidezni stroj, imenovan Dalvik, vsebuje prevajalnik JIT, ki je zadolžen za zaganjanje že prevedene programske kode v javi. Prevedena koda je zapakirana v datoteke s končnico .apk. Tem datotekam pravimo aplikacije, izdelane za sistem Android.

Za izdelavo aplikacije Android potrebujemo znanje programskega jezika Java, Android SDK, ki je brezplačno na voljo na uradni spletni strani<sup>2</sup>.

<sup>1</sup>Celotno ime je Android Open Source Project

<sup>2</sup>Uradna spletna stran je dostopna na <http://developer.android.com/>

Priporočljivo je tudi branje dokumentacije SDK, ki je na voljo na spletu<sup>3</sup>. Dokumentacija je napisana zelo razumljivo, tako da se lahko znajdejo tudi začetniki. S pomočjo Android SDK na koncu nastane datoteka .apk.

---

<sup>3</sup>Dokumentacija SDK je dostopna na <http://developer.android.com/sdk/>

# Poglavlje 6

## Nadgradnja: Zaznavanje obrazov

Eden od pomembnih funkcionalnih elementov v instalaciji je avtomatska detekcija obrazov na slikah. V prvotni verziji instalacije je bila detekcija obrazov izvedena s pomočjo barve kože [12, 3], od takrat pa je razvoj na področju računalniškega vida prinesel veliko novosti in boljših metod.

Operacijski sistem Android nam preko vmesnika JNI omogoča vključevanje kode, napisane v programskem jeziku C ali C++. Posledično lahko uporabljam splošne knjižnice in programska ogrodja (*angl. framework*).

Ker smo v naši nadgradnji želeli, da bi se obrazi zaznavali v realnem času, smo se odločili za uporabo programskega ogrodja OpenCV.

### 6.1 Zaznavanje obrazov z uporabo OpenCV

OpenCV je programsko ogrodje, ki nam omogoča izvedbo raznih operacij, izvedenih nad slikami. Napisano je v programskem jeziku C, za potrebe operacijskega sistema Android pa so napisani tudi povezovalni javanski razredi (*angl. class*), ki nam omogočajo uporabo operacij OpenCV kar v programskem jeziku Java.

Najpomembnejša funkcionalnost programskega ogrodja OpenCV za nas je

bila implementacija zaznavanja obrazov. Ta uporablja algoritem kaskadnega klasificiranja (*angl. cascade classifier*), ki nam omogoča klasifikacijo različnih lastnosti.

Lastnosti so zapisane v datoteki XML. Ta datoteka vsebuje več sto ali tisoč različnih lastnosti, ki opisujejo določen predmet, v našem primeru obliko obraza.

Obstaja več različnih tipov, preko katerih so te lastnosti opisane. Za zaznavanje obrazov sta najpomembnejša tipa HAAR in LBP. Tipa se razlikujeta po točnosti in hitrosti.

Opisi HAAR so procesorsko bolj potratni in potrebujejo relativno veliko časa za prepoznavo obraza, vendar pa je zato veliko manj napačnih detekcij (*angl. false positive*).

LBP pa je, čeprav z nekaj več napakami pri klasifikaciji kot HAAR, za naše potrebe še vedno dovolj točen. Njegova glavna prednost pred prej opisanim je njegova hitrost, kar je pri detekciji obrazov v živo zelo pomembno, če želimo doseči zadovoljstvo uporabnika.

Ker so si obrazi med seboj zelo različni, je nemogoče, da bi bil najden prav vsak obraz, vseeno pa želimo, da je število zgrešenih obrazov čim manjše. Vendar pa zgrešen obraz ni največja težava, saj uporabnik tega najbrž ne bi opazil. Veliko bolj neprijetna težava je, če najdemo obraz tam, kjer ga ni, na primer nek predmet v okolini, ki ga zaznamo in obdelamo kot obraz.

Testirali smo oba tipa z različnimi tipi lastnosti. Pri tem smo uporabili le lastnosti, ki opisujejo sprednji del obraza. Vendar pa se je pri uporabi opisov LBP še vedno pokazalo preveč lažnih obrazov. Zato smo se odločili za uporabo lastnosti tipa HAAR. Veliko bolj se nam namreč zdi pomembno, da ima petnajst sekund slave obraz uporabnika, in ne kakšen predmet v okolini, pa čeprav je za to potrebno nekoliko več časa.

Vendar pa tudi opisi obrazov tipa HAAR niso vedno dosegli našega pričakovovanja. Zato smo se odločili, da naredimo dvojno klasifikacijo.

Najprej smo klasificirali z lastnostmi sprednjega dela obraza, opisanimi v datoteki *haarcascade\_frontalface\_default.xml* 6.1, s katerimi smo dobili re-

zultate obrazov. Ker se je že za samo klasifikacijo obraza porabilo kar nekaj časa, smo se odločili, da iskanje prekinemo takoj po najdenem prvem obrazu, ki je velik vsaj eno desetino celotne slike.

Izsek 6.1: Nekaj vrstic datoteke *harrcascade\_frontalface\_default.xml*

```
<maxWeakCount>9</maxWeakCount>
<stageThreshold>-5.0425500869750977e+00</stageThreshold>
<weakClassifiers>
  <_>
    <internalNodes>
      0 -1 0 -3.1511999666690826e-02
    </internalNodes>
    <leafValues>
      2.0875380039215088e+00 -2.2172100543975830e+00
    </leafValues>
  </_>
  <_>
    <internalNodes>
      0 -1 1 1.2396000325679779e-02
    </internalNodes>
    <leafValues>
      -1.8633940219879150e+00 1.3272049427032471e+00
    </leafValues>
  </_>
  ...
  ...
```

Kandidata obraza smo testirali še s klasifikacijo očesa. Vsak obraz ima dve očesi. Zavedali smo se, da v primeru očal tega ne bomo našli, vendar pa se je izkazalo, da težave nastopajo le pri zasenčenih očalih. Za to klasifikacijo smo uporabili lastnosti, zapisane v datoteki *harrcascade\_eye.xml* 6.2. V primeru, da smo našli točno dva rezultata, smo vrnili prejšnji rezultat, v nasprotnem primeru pa javili, da obraz ni najden.

Izsek 6.2: Nekaj vrstic iz datoteke harrcascade\_eye.xml

```

<maxWeakCount>6</maxWeakCount>
<stageThreshold>-1.4562760591506958e+00</stageThreshold>
<weakClassifiers>
  <_->
    <internalNodes>
      0 -1 0 1.2963959574699402e-01
    </internalNodes>
    <leafValues>
      -7.7304208278656006e-01 6.8350148200988770e-01
    </leafValues>
  </_>
  <_->
    <internalNodes>
      0 -1 1 -4.6326808631420135e-02
    </internalNodes>
    <leafValues>
      5.7352751493453979e-01 -4.9097689986228943e-01
    </leafValues>
  </_>
  ...

```

S temi nastavivami in omejitvami smo dosegli pričakovane rezultate. Polje z najdenim rezultatom smo povečali še za približno eno tretjino, tako da je končni produkt zajemal nekoliko večji predel osnovne slike in je bil kvadratne oblike. Zakaj ravno eno tretjino? Iskali smo prednji del obraza in ga tudi našli. Vendar si v okvirju želimo videti sliko celotne glave in ne samo obraza. Povečava polja za eno tretjino se je izkazala kot ravno pravšnja za večino testiranih primerov.

Detekcija obrazov s programskim ogrodjem OpenCV je le ena izmed možnosti, ki smo si jih podrobnejše ogledali. Kot prva implementacija je bila



(a) Original      (b) Zaznavanje obraza      (c) Zaznavanje oči

**Slika 6.1:** Prikaz zaznavanja obraza in oči s programskim ogrodjem OpenCV

izbrana zaradi velike stopnje nadzora nad vsakim korakom, kot so: detekcija obraza, prikazana kot zelen pravokotnik okrog obraza na sliki 6.1b; določitev regije, kjer je potrebno iskatiti; detekcija oči v določeni regiji, prikazana kot dva modra kroga okrog oči na sliki 6.1c in še bi lahko naštevali.

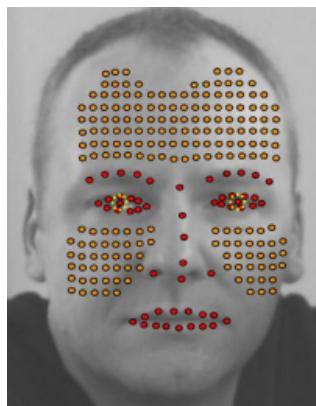
Kljub zadovoljivim rezultatom iskanja obrazov se je izkazalo, da slike velikokrat niso ostre. Prej smo omenili, da imamo veliko nadzora nad različnimi algoritmi za detekcijo, kar pa ne vključuje uporabe kamere, ki je v pametnem telefonu. OpenCV uporablja sliko iz kamere tako, kot je, brez predhodne izostritve. Povedano z drugimi besedami, uporablja jo kot kamero in ne kot digitalni fotoaparat.

## 6.2 Zaznavanje obrazov z uporabo Android SDK

Android SDK deluje povsem drugače kot OpenCV. Ne omogoča nobenega nazdora in parametrov, možna je uporaba le vnaprej določenega algoritma.

Ta algoritem temelji na algoritmu NV1-NORM [7], ki ga je razvilo podjetje Neven Vision 6.2. Leta 2006 je Neven Vision postalo last podjetja Google in od takrat se uporablja za zaznavo obrazov v mnogih programih, ki jih je izdelalo podjetje Google. Lep primer je program za prikaz slik Google Picasa.

Natančnejša implementacija algoritma za zaznavo obraza je zaprtoko-



**Slika 6.2:** Vizualizacija zaznavanja obraza, Neven Vision

dna. Če si ogledamo dokumentacijo Android SDK-ja, je opisan samo primer uporabe:

Izsek 6.3: Primer uporabe zaznavanja obraza z orodjem Android SDK

```
FaceDetector fc = new FaceDetector(  
    sirina_slike,  
    visina_slike,  
    maksimalno_stevilo_obrazov  
) .findFaces(slika, obazi);
```

Vendar pa kljub zaprtosti in nezmožnosti prilagajanja detektorja prinaša zelo dobre rezultate. Ker so vse funkcije že vključene v operacijski sistem, tudi ne potrebujemo dodatnih programskega ogrodja. S tem smo si prihranili kar nekaj časa in prostora.

Rezultat zaznavanja obraza pa tokrat ni obraz sam. Kot rezultat dobimo razdaljo med očmi, rotacijo obraza ter točko med očmi. Na podlagi teh podatkov lahko določimo regijo, na kateri je obraz na sliki.

Testirali smo razlike med rezultati obeh algoritmov in se na koncu odločili za uporabo Android SDK. Kot že opisano, poleg dobrih rezultatov pridobimo še na velikosti končne aplikacije in kompleksnosti celotnega programa.

# Poglavlje 7

## Nadgradnja: Obdelava slik

Instalacija ‐15 sekund slave‐ vsebuje 17 različnih filtrov [2, Poglavlje 5] za predelavo slik obrazov v popart portrete. Filtri so bili izdelani s pomočjo grafičnega programa GIMP.

### 7.1 GIMP

GIMP je odprtokodno programsko orodje za obdelavo slik.

Začetek projekta sega v leto 1995, kjer se je vse skupaj začelo kot semestrski projekt na Univerzi Kalifornije, Berkeley. Avtorja Spencer Kimball in Peter Mattis sta projekt poimenovala *angl. General Image Manipulation Program*, čez nekaj časa pa sta si premislila in Richarda Stallmana, ki je ravno takrat obiskal univerzo, prosila, če lahko spremenita ‐General‐ v ‐GNU‐. Od takrat naprej se program imenuje *angl. GNU Image Manipulation Program* ali na kratko še vedno GIMP [22].

### 7.2 Izbira filtrov

Že naša prvotna izbira filtrov je izhajala iz programskega orodja za predelavo slik GIMP. Zaradi lažjega dela se je najprej s pomočjo grafičnega vmesnika izbralo zaporedje različnih filtrov in njihove parametre. Izbrani so bili naslednji

filtrij:

- **uravnovešanje barv** *angl. color balance*

Prekrije celotno sliko z barvo v izbranem odtenku.

- **posteriziranje** *angl. posterize*

Zmanjšuje število barv na sliki.

- **uravnovešanje barv v prostoru HSL** *angl. hue saturation balance*

Spreminja odtenek, svetlost in nasičenost barv na sliki.

Iz teh treh filtrov je bilo narejenih mnogo kombinacij, v končni fazi pa izbranih le 17 tistih, ki so bile najprimernejše. Ostale so bile izločene, ker rezultati niso bili dovolj dobri, pri nekaterih pa je bila prevelika časovna zahtevnost [2].

### 7.3 Implementacija filtrov

Ker je GIMP odprtokoden, smo imeli dostop tudi do implementacij filtrov. Te so napisane v programskem jeziku C – na prvi pogled kot nalašč za enostaven prenos na platformo Android preko JNI. Vendar pa so se pokazale težave, saj so bili filtri močno povezani z GIMP-objekti in nekoliko bolj kompleksni, kot bi bilo potrebno za naš projekt. Zato smo se odločili, da filtre ponovno implementiramo in zavrzemo vse funkcionalnosti, ki za nas niso pomembne. Naredili smo tri različne implementacije in jih med seboj primerjali.

Prva implementacija je napisana v programskem jeziku Java. Izkazala se je za zelo počasno. Orodje za sproščanje pomnilnika GC (*angl. garbage collector*) se je klicalo prepogosto, kar je posledično porabilo več kot nekaj sekund za filter. Tega si nismo mogli privoščiti, zato smo iskali alternative.

Naslednja implementacija je napisana s pomočjo grafične kartice, in sicer z ogrodjem OpenGL ES verzije 2.0. Čas izvajanja se je občutno zmanjšal. Ko so bile tekture zapisane na grafični kartici, so se filtri izvajali v manj kot sekundi. Ta rešitev je bila že dovolj dobra, vendar pa se nam je zdelo, da je uporaba grafične kartice za tako lahke operacije potrata energije.

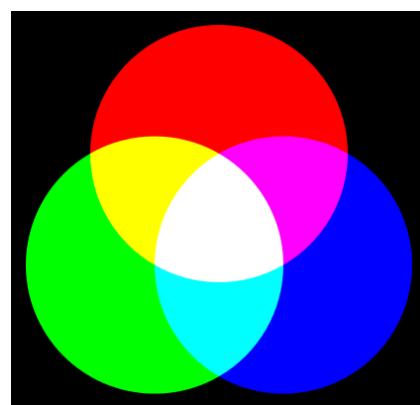
Čeprav so bili rezultati že dovolj dobri, smo se odločili še za implementacijo v programskem jeziku C. Rezultati so pokazali, da je ta rešitev nekoliko počasnejša kot prejšnja, vendar pa za to nismo potrebovali grafične kartice na telefonu. Porabljen čas je bil manj kot sekundo za filter.

### 7.3.1 Barvni prostor

Ko govorimo o filtroh, govorimo predvsem o spreminjanju barv. Tukaj se pojavi vprašanje: kako te barve matematično zapisati? Poznamo več barvnih prostorov, s katerimi lahko določeni barvi priredimo določeno številko.

#### Barvni prostor RGB

Najbolj znan in osnoven barvni prostor je RGB. Sestavljen je iz treh osnovnih barv, to so rdeča (R), zelena (G) in modra (B). Vsaka barva se lahko pojavi v 256 odtenkih, kar na koncu znaša 16777216 barv ( $256^3$ ).

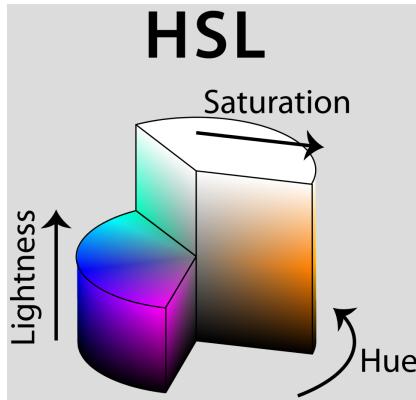


Slika 7.1: Grafični prikaz barvnega prostora RGB

#### Barvni prostor HSL

Drugi, malo manj znan, je barvni prostor HSL. Sestavljen je iz treh komponent: barvni odtenek (H), intenzivnost (S) in svetlost (L). Če primerjamo slike 7.1 in 7.2, vidimo, da HSL nima treh enakih komponent, ampak je

zgrajen v obliki valja. Barvni odtenek je zapisan s kotom, in sicer pri  $0^\circ$  se začne rdeča, pri  $120^\circ$  zelena in pri  $240^\circ$  modra. Drugi dve komponenti, nasičenost in svetlost, sta zapisani z odstotki, in sicer od 0% do 100%.



**Slika 7.2:** Grafični prikaz barvnega prostora HSL

Obstaja še veliko barvnih prostorov, vendar sta za našo instalacijo zadostovala ta dva. Pomembno pri barvnih modelih je tudi to, da so enostavno preračunljivi iz enega prostora v drugega.

### 7.3.2 Pretvorba iz barvnega modela RGB v HSL

$$R, G, B \in [0, 255]$$

$$C_{max} = \max(R, G, B) \quad (7.1)$$

$$C_{min} = \min(R, G, B) \quad (7.2)$$

$$\Delta = (C_{max} - C_{min}) \quad (7.3)$$

$$L = (max + min)/2 \quad (7.4)$$

$$S = \begin{cases} 0, & \Delta = 0 \\ 255 * \Delta / (C_{max} + C_{min}), & L < 128 \\ 255 * \Delta / (511 - C_{max} - C_{min}), & \text{ostalo} \end{cases} \quad (7.5)$$

$$H' = \begin{cases} 0, & \Delta = 0 \\ 0 + 42.5 * (G - B) / \Delta, & R = Cmax \\ 85 + 42.5 * (B - R) / \Delta, & G = Cmax \\ 170 + 42.5 * (R - G) / \Delta, & B = Cmax \end{cases} \quad (7.6)$$

$$H = \begin{cases} H' + 255, & H' < 0 \\ H' - 255, & H' > 255 \\ H', & \text{ostalo} \end{cases} \quad (7.7)$$

### 7.3.3 Uravnovešanje barv

$$\text{clamp}_{\min}^{\max}(x) = \begin{cases} \min, & x < \min \\ \max, & x > \max \\ x, & \text{ostalo} \end{cases} \quad (7.8)$$

$$\text{shadows}(x) = \text{clamp}_0^1\left(\frac{117 - x}{64}\right) * 1.785 \quad (7.9)$$

$$\text{color\_balance}(x, x') = \text{clamp}_0^{255}(x + x' * \text{shadows}(x)) \quad (7.10)$$

Uravnovešanje barv se računa v barvnem prostoru RGB. Računamo za vsako barvo posebej, in sicer z enačbo (7.10), kjer je  $x$  enak originalni barvi,  $x'$  pa barvi, s katero želimo uravnovesiti originalno barvo  $x$ .  $x$  in  $x'$  sta pozitivni celi števili med 0 in 255.

Po izračunanih vrednostih dane rezultate iz prostora RGB prenesemo v prostor HSL s pomočjo enačb, napisanih v poglavju 7.3.2. Tako dobimo  $H$ ,  $S$  in  $L$ . Iz originalne vrednosti, to je vrednost barve na sliki, izračunamo  $L$  (7.4) in ga nadomestimo s prejšnjim, zato da ohranimo podobno svetlost.  $H$ ,  $S$  in novo pridobljeni  $L$  prenesemo nazaj v prostor RGB.

### 7.3.4 Posteriziranje

$$\text{luts}(x, levels) = 255 * \frac{\lfloor \frac{x}{255} * (levels - 1) + 0.5 \rfloor}{levels - 1} + 0.5 \quad (7.11)$$

Posteriziranje se računa v barvnem prostoru RGB. Računamo za vsako barvo posebej, in sicer z enačbo (7.11), kjer je  $x$  enak originalni barvi,  $levels$  pa je stopnja posteriziranja.  $x$  in  $levels$  sta pozitivni celi števili med 0 in 255.

### 7.3.5 Uravnovešanje barv v prostoru HSL

$$htv(x, x') = x + 255 * \frac{x'}{360}$$

$$hue\_transfer(x, x') = \begin{cases} htv(x, x') + 255, & htv(x, x') < 0 \\ htv(x, x') - 255, & htv(x, x') > 255 \\ htv(x, x'), & \text{ostalo} \end{cases} \quad (7.12)$$

$$ltv(x, x') = clamp_{-255}^{255}(1.27 * x')$$

$$lightness\_transfer(x, x') = \begin{cases} x * \frac{255 + ltv(x, x')}{255}, & ltv(x, x') < 0 \\ x + \frac{(255 - x) * ltv(x, x')}{255}, & ltv(x, x') \geq 0 \end{cases} \quad (7.13)$$

$$stv(x, x') = clamp_{-255}^{255}(2.25 * x')$$

$$saturation\_transfer(x, x') = clamp_0^{255}\left(x + x * \frac{stv(x, x')}{255}\right) \quad (7.14)$$

Uravnovešanje barv v prostoru HSL se računa v barvnem prostoru HSL. Ker pa so prvotno barve zapisane v barvnem prostoru RGB, je potrebno te pretvoriti v barvni prostor HSL. To naredimo s pomočjo enačb, napisanih v poglavju 7.3.2. Tako dobimo  $H$ ,  $S$  in  $L$ .

Zdaj lahko vsako komponento uravnovesimo. Vsaka komponenta se izračuna po svoji enačbi. Za  $H$  uporabimo (7.12), za  $L$  (7.13) in za  $S$  (7.14).

Dobljene vrednosti pretvorimo nazaj v prostor RGB, kar je tudi naš rezultat.

### 7.3.6 Sestavljanje popart filtrov

#### Filter 1

Prvi filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = 30$ ,  $G = -32$  in  $B = 16$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 4$ . Rezultat testne slike lahko vidimo na sliki 7.3.



**Slika 7.3:** Primerjava originalne slike s sliko, obdelano s filtrom št. 1

#### Filter 2

Drugi filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = -36$ ,  $G = -37$  in  $B = 39$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 4$ . Rezultat testne slike lahko vidimo na sliki 7.4.

#### Filter 3

Tretji filter vsebuje le osnovni filter ‐posteriziranje‐, in sicer s parametrom  $ST = 2$ . Rezultat testne slike lahko vidimo na sliki 7.5.



(a) Original

(b) Drugi filter

**Slika 7.4:** Primerjava originalne slike s sliko, obdelano s filtrom št. 2

(a) Original

(b) Tretji filter

**Slika 7.5:** Primerjava originalne slike s sliko, obdelano s filtrom št. 3

#### Filter 4

Četrti filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = 100$ ,  $G = -100$  in  $B = -100$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 3$ . Rezultat testne slike lahko vidimo na sliki 7.6.



(a) Original

(b) Četrtri filter

**Slika 7.6:** Primerjava originalne slike s sliko, obdelano s filtrom št. 4

### Filter 5

Peti filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = -100$ ,  $G = -100$  in  $B = 100$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 3$ . Rezultat testne slike lahko vidimo na sliki 7.7.



(a) Original

(b) Peti filter

**Slika 7.7:** Primerjava originalne slike s sliko, obdelano s filtrom št. 5

### Filter 6

Šesti filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = -100$ ,  $G = 100$  in  $B = -100$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 3$ . Rezultat testne slike lahko vidimo na sliki 7.8.



**Slika 7.8:** Primerjava originalne slike, obdelane s filtrom št. 6

### Filter 7

Sedmi filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐posteriziranje‐, in sicer s parametrom  $ST = 5$ . Rezultat obdelamo še s filtrom ‐uravnovešanje barv v prostoru HSL‐ s parametri  $H = -41$ ,  $L = -15$  in  $S = 6$ . Rezultat testne slike lahko vidimo na sliki 7.9.

### Filter 8

Osmi filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐posteriziranje‐, in sicer s parametrom  $ST = 6$ . Rezultat obdelamo še s filtrom ‐uravnovešanje barv‐ s parametri  $R = -29$ ,  $G = 40$  in  $B = 100$ . Rezultat testne slike lahko vidimo na sliki 7.10.



(a) Original

(b) Sedmi filter

**Slika 7.9:** Primerjava originalne slike s sliko, obdelano s filtrom št. 7



(a) Original

(b) Osmi filter

**Slika 7.10:** Primerjava originalne slike s sliko, obdelano s filtrom št. 8

### Filter 9

Deveti filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom “uravnovešanje barv v prostoru HSL”, in sicer s parametri  $H = -41$ ,  $L = -20$  in  $S = 25$ . Rezultat obdelamo še s filtrom “posteriziranje” s parametrom  $ST = 4$ . Rezultat testne slike lahko vidimo na sliki 7.11.



(a) Original (b) Deveti filter

Slika 7.11: Primerjava originalne slike s sliko, obdelano s filtrom št. 9

## Filter 10

Deseti filter je sestavljen iz treh osnovnih filtrov. Najprej sliko obdelamo s filtrom "uravnovešanje barv", in sicer s parametri  $R = 100$ ,  $G = -100$  in  $B = -100$ . Rezultat obdelamo še s filtrom "posteriziranje" s parametrom  $ST = 3$  in nazadnje še s filtrom "uravnovešanje barv v prostoru HSL" s parametri  $H = -41$ ,  $L = -10$  in  $S = 20$ . Rezultat testne slike lahko vidimo na sliki 7.12.



(a) Original (b) Deseti filter

**Slika 7.12:** Primerjava originalne slike s sliko, obdelano s filtrom št. 10

### Filter 11

Enajsti filter je sestavljen iz treh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = 34$ ,  $G = -38$  in  $B = 24$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 4$  in nazadnje še s filtrom ‐uravnovešanje barv v prostoru HSL‐ s parametri  $H = -65$ ,  $L = 0$  in  $S = 0$ . Rezultat testne slike lahko vidimo na sliki 7.13.



**Slika 7.13:** Primerjava originalne slike s sliko, obdelano s filtrom št. 11

### Filter 12

Dvanajsti filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = 40$ ,  $G = -40$  in  $B = 28$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 4$  in nazadnje še enkrat s filtrom ‐uravnovešanje barv‐, vendar s parametri  $R = -100$ ,  $G = -42$  in  $B = 100$ . Rezultat testne slike lahko vidimo na sliki 7.14.

### Filter 13

Trinajsti filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = 45$ ,  $G = -45$  in



(a) Original

(b) Dvanajsti filter

**Slika 7.14:** Primerjava originalne slike s sliko, obdelano s filtrom št. 12

$B = 35$ . Rezultat obdelamo še s filtrom “posteriziranje” s parametrom  $ST = 4$ . Rezultat testne slike lahko vidimo na sliki 7.15.



(a) Original

(b) Trinajsti filter

**Slika 7.15:** Primerjava originalne slike s sliko, obdelano s filtrom št. 13

#### Filter 14

Štirinajsti filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom “uravnovešanje barv”, in sicer s parametri  $R = -45$ ,  $G = -55$

in  $B = 30$ . Rezultat obdelamo še s filtrom "posteriziranje" s parametrom  $ST = 4$ . Rezultat testne slike lahko vidimo na sliki 7.16.



Slika 7.16: Primerjava originalne slike s sliko, obdelano s filtrom št. 14

### Filter 15

Petnajsti filter vsebuje le osnovni filter "posteriziranje", in sicer s parametrom  $ST = 3$ . Rezultat testne slike lahko vidimo na sliki 7.17.



Slika 7.17: Primerjava originalne slike s sliko, obdelano s filtrom št. 15

### Filter 16

Šestnajsti filter je sestavljen iz dveh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = 30$ ,  $G = -40$  in  $B = 16$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 3$ . Rezultat testne slike lahko vidimo na sliki 7.18.



**Slika 7.18:** Primerjava originalne slike s sliko, obdelano s filtrom št. 16

### Filter 17

Sedemnajsti filter je sestavljen iz treh osnovnih filtrov. Najprej sliko obdelamo s filtrom ‐uravnovešanje barv‐, in sicer s parametri  $R = 20$ ,  $G = -30$  in  $B = 20$ . Rezultat obdelamo še s filtrom ‐posteriziranje‐ s parametrom  $ST = 4$  in nazadnje še s filtrom ‐uravnovešanje barv v prostoru HSL‐ s parametri  $H = -30$ ,  $L = -10$  in  $S = 12$ . Rezultat testne slike lahko vidimo na sliki 7.19.



**Slika 7.19:** Primerjava originalne slike s sliko, obdelano s filtrom št. 17



# Poglavlje 8

## Nove funkcionalne možnosti

Do sedaj smo originalno instalacijo ‐15 sekund slave‐ predelovali na način, da jo ohranjamo kot digitalno umetnost, se pravi brez vidnih sprememb. Vendar pa kakšna manjša po tako dolgem času lahko uporabniško izkušnjo le popestri.

Najpomembnejša posodobitev, ki bi to umetniško instalacijo naredila veliko bolj popularno, je integracija s socialnimi omrežji, kar je podrobneje opisano v 9. poglavju. Dandanes se za izmenjavo slikovnih informacij uporablja predvsem socialna omrežja, ki pred 10 leti sploh še niso obstajala.

### 8.1 Dve slike

Naslednja posodobitev, ki prvotno niti ni bila v načrtu, je zamenjava zaslona LCD. Ta posodobitev se je izvedla zaradi nezdružljivosti mobilnega telefona s starim zaslonom LCD. Zopet se je pojavil problem, omenjen v 4. poglavju; težko je nadgraditi le eno komponento, saj obstaja možnost, da ni združljiva z ostalo opremo. Ker stari zaslon nima priključka HDMI, smo poskusili s pretvornikom HDMI v DVI, a brez uspeha. Isti pretvornik smo poskusili uporabiti na novejšem zaslonu. Tu je bil prenos slike s telefona na zaslon uspešen.

Večina današnjih zaslonov LCD je formata 16:9 ali pa še več v širino, starega formata 4:3 skorajda ni več. Naš cilj pa je kvadratna slika, saj je

tudi sam Andy Warhol uporabljal tak format, zato smo najprej žeeli najti čim bolj kvadraten zaslon. Vendar pa se je kmalu rodila ideja o še širšem zaslonu, takem, pri katerem je širina malo več kot enkrat daljša od višine, kot je prikazano na sliki 8.1b. S tem smo omogočili možnost prikazovanja dveh slik vzporedno – kot da bi naenkrat imeli razstavljeni dve različici istega portreta. Na podoben način je svoje popart portrete razstavljal tudi Andy Warhol. Seveda ima vsaka slika drug obraz, če le-ta obstaja.

Zaradi prejšnjega razloga o zamenjavi zaslona je bilo potrebno zamenjati tudi okvir. Kot že pri prejšnji posodobitvi okvirja bo tudi tokrat ostal bogato okrašen, le na sredini bo združen, kot je prikazano na sliki 8.1b.



(a) Originalni okvir

(b) Novi razširjeni okvir

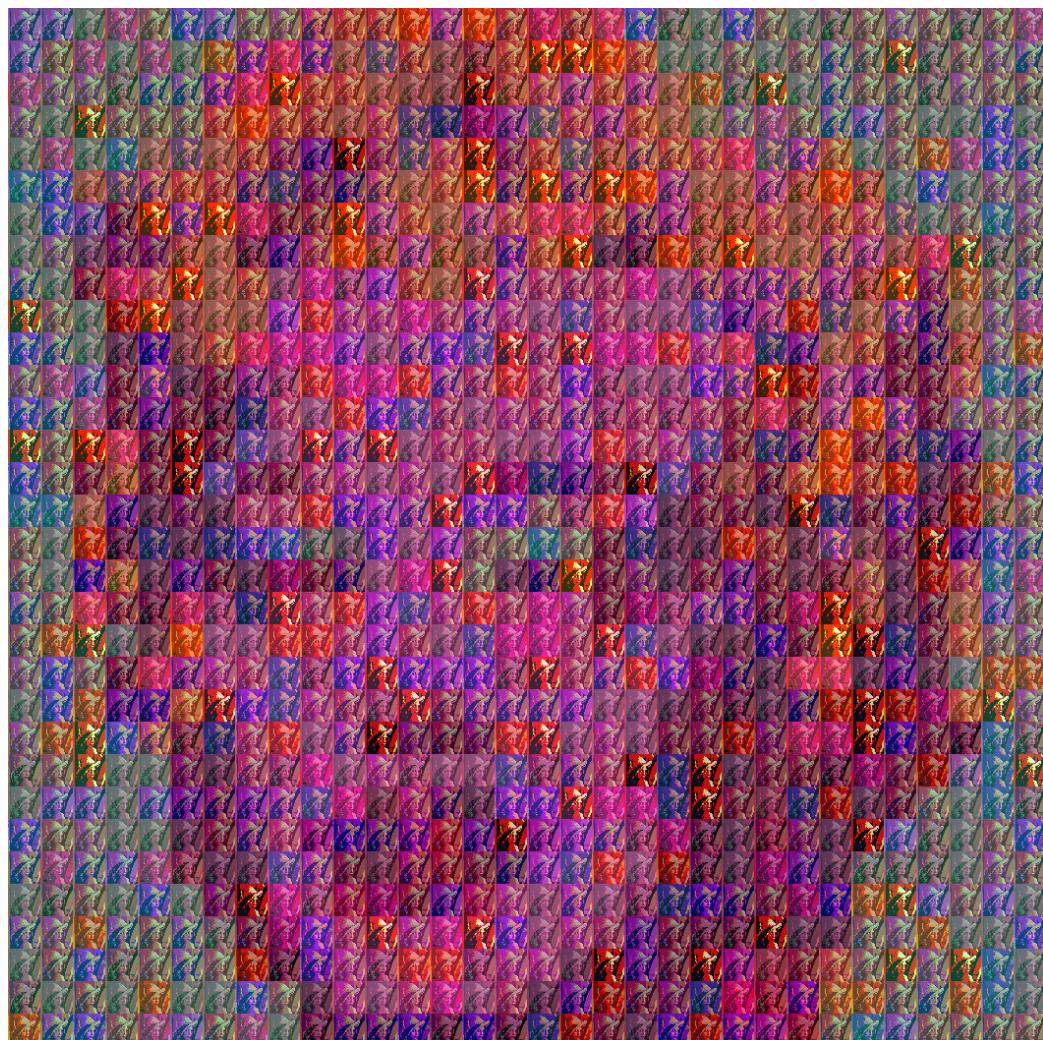
**Slika 8.1:** Prikaz razširitve okvirja zaradi menjave širšega zaslona LCD

## 8.2 Obraz kot mozaik iz majhnih obrazov

Prvotna instalacija ‐15 sekund slave‐ me je najbolj prepričala s sliko Marilyn Monroe 3.1, ki je sestavljena iz več obrazov [11], da izgleda kot mozaik [17]. Ker tukaj nimamo tako velike baze obrazov, smo vse skupaj malo poenostavili in uporabili kar iste obraze, vendar obdelane z drugimi filtri. Da pa je bila slika od daleč razločna, smo obrazu, ki je prikazan kot ‐piksel‐, prilili originalno barvo<sup>1</sup>. Ta slika (primer lahko vidimo na sliki 8.2) je končni prikaz naše nove

<sup>1</sup>Prejšnji končni rezultat instalacije obraza smo spremenili v sliko velikosti  $\text{širina} * \text{višina} = \text{novo število obrazov}$ .

instalacije.



**Slika 8.2:** Prikaz 1024 slik Lene, prikazanih na podoben način kot slika 3.1.



# Poglavlje 9

## Povezovanje s socialnimi omrežji

Ena izmed pomembnejših posodobitev je povezava s socialnimi omrežji, ki so vsak dan bolj popularna. Vsak dan se na različnih socialnih omrežjih doda na tisoče ‐selfijev‐<sup>1</sup>. Da pa ta ‐selfi‐ še bolj pade v oči, se jim lahko doda kakšen vnaprej določen filter. Ti so pri pametnih telefonih, ki so danes ena izmed najpogostejših naprav za fotografiranje ‐selfijev‐, na voljo samo en dotik stran. Obstajajo pa tudi socialna omrežja, ki ponujajo osnovne filtre že pri nalaganju slike.

Naša instalacija se lahko interpretira kot generator ‐selfijev‐. Oseba, ki stoji pred instalacijo, ve, da bo kmalu njenih petnajst sekund. Pred njo se bo pokazala popart slika, ki je njen umetniški ‐selfi‐, in če želi, je ta popart slika kmalu javno objavljena na enem izmed socialnih omrežij; lahko se na primer dotakne ikone in tako odloči, da se njen portret prenese na želeno socialno omrežje.

---

<sup>1</sup>Slika, ko oseba fotografira samo sebe pri različnih dejavnostih.

## 9.1 Možnosti

Najpogosteje so uporabljena naslednja socialna omrežja:

**Facebook** Trenutno najbolj razširjeno socialno omrežje, ki ga uporablja vse generacije.

**Twitter** Socialno omrežje, najbolj namenjeno kratkim sporočilom, imenovanim ‐tviti‐.

**Instagram** Socialno omrežje, predvsem namenjeno nalaganju slik. Omogoča tudi veliko število filtrov za obdelavo le-teh.

**Google+** Alternativa socialnemu omrežju Facebook podjetja Google.

**Pinterest** Socialno omrežje, predvsem namenjeno ustvarjalnostim, ki so prikazane v obliki slik.

**SnapChat** Aplikacija, ki omogoča pošiljanje slik, ki so vidne le nekaj sekund. Po pretečenem času, to je maksimalno deset sekund [23], slika ni več na voljo.

## 9.2 Implementacija s socialnim omrežjem Facebook

Skoraj vsa socialna omrežja imajo omogočene javne API-je in Facebook ni nobena izjema. Preko vmesnika API, imenovanega Graph, lahko programsko naredimo večino akcij, ki so na voljo uporabniku preko spletnega brskalnika.

Da pa je poskrbljeno za varnost in zasebnost, je za dovoljenje uporabe Graph API-ja potrebno narediti novo aplikacijo (*angl. app*). Tu določimo, katere pravice želimo naši aplikaciji dodeliti. Več pravic kot bomo zahtevali, manj zaupanja vredna bo naša aplikacija, če ni uporabniku očitno, zakaj aplikacija te pravice potrebuje. Zato je priporočljivo, da se zahtevajo le pravice, ki jih nujno potrebujemo za delovanje naše aplikacije. V našem

primeru potrebujemo pravico, da lahko pišemo in objavljamo slike na zid uporabnika.

Ko aplikacijo pripravimo, jo morajo uslužbenci pri podjetju Facebook potrditi. To je varnostni postopek, brez katerega naše aplikacije ne moremo javno uporabljati.

Ko je aplikacija narejena in potrjena, dobimo poseben ključ, ki je potreben za povezavo z Graph API-jem. Graph API lahko uporabljam na več načinov. Najbolj osnoven je preko protokola HTTP, a je hkrati najbolj zahteven za implementacijo. Drugi način je preko knjižnice, prilagojene za naprave Android. S to knjižnico lahko na enostaven način opravimo vse osnovne akcije, kot sta prijava v sistem in objavljanje slike.

### 9.3 Omejitve

Ena izmed največjih nevarnosti uporabe socialnih omrežij je nenadzorovan pošiljanje slik na socialno omrežje. Na slikah so lahko osebe, ki si ne želijo biti objavljene na socialnem omrežju, saj je javno. Še več, na slikah so lahko otroci, katerih objavljanje slik na socialnem omrežju je brez dovoljenja staršev prepovedano. Res je, da smo žeeli slike izbrisati petnajst sekund po objavi, vendar pa večina socialnih omrežij te slike še vedno hrani, čeprav jih ne vidimo več. In če se podamo še v najhujšo skrajnost, nekdo bi lahko napisal program, ki vse slike shranjuje na nek strežnik, in bi nato z njimi razpolagal po svoji volji.



# Poglavlje 10

## Sklepne ugotovitve

V sklopu magistrske naloge smo posodobili strojno in programsko opremo instalacije ‐15 sekund slave‐. Razvoj novih uporabnih funkcionalnosti nam je omogočil, da smo instalaciji dodali tudi nekaj novih možnosti, ki naredijo uporabniško izkušnjo še bolj zanimivo.

Digitalni fotoaparat in osebni računalnik smo zamenjali s pametnim telefonom, s čimer smo dosegli, da instalacija zavzame manj prostora, saj je uporabljenih manj komponent, kot jih je bilo prej, poleg tega pa je omogočeno tudi lažje vzdrževanje in odpravljanje morebitnih napak, saj se vse nahaja na eni napravi.

S posodobitvijo programske opreme smo omogočili, da instalacija deluje na eni napravi in dodali nove funkcionalnosti:

- povezovanje s socialnimi omrežji, kar je predvsem za mlajše uporabnike zaželena prednost;
- sestavljanje končnega obrazu iz več obrazov, ki so obdelani z različnimi filterji;
- posodobitev in optimizacija algoritmov za filtriranje slik;
- posodobitev in optimizacija algoritmov za zaznavanje obrazov.

V prihodnosti obstaja možnost, da se filtri uporabijo tudi v drugih aplikacijah, saj so filtri napisani kot knjižnica, uporaba te pa je enostavna.

Predstavljajmo si aplikacijo, kot je Instagram, ki ima tudi možnost popart filtrov, napisanih v tej magistrski nalogi.

Poleg tega bo potrebno redno vzdrževanje instalacije, tako na nivoju strojne opreme kot tudi na nivoju programske opreme, saj želimo ostati v koraku s časom.

Med izvedbo naloge smo se večkrat soočili z vprašanjem, do kakšne mere instalacijo še lahko spreminjam, da umetnina še vedno ostane ista in ne nekaj povsem novega. Na tem področju se je tudi pojavilo največ težav. Čeprav je na prvi pogled le enostavna ponovna implementacija že narejene instalacije, pa vendar pri umetniškem delu ni vse samo v programu. Zelo veliko pozornosti moramo namreč nameniti temu, da umetnina za opazovalca ostane taka, kot je bila.

# Literatura

- [1] F. Dietrich, “Visual intelligence: The first decade of computer art (1965–1975)”, *Leonardo*, zv. 19, št. 2, str. 159–169, 1986.
- [2] S. Juvan, “Predelava fotografij z barvnimi filtri v slogu pop-art”, Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2002.
- [3] J. Kovač, P. Peer in F. Solina, “Illumination independent color-based face detection”, v *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis (ISPA 2003)*, IEEE, zv. 1, str. 510–515, 2003.
- [4] W. Lieser, *The World of Digital Art*. h.f. Ullmann, 2010.
- [5] P. Miller, “Technology for art’s sake”, *IEEE Spectrum*, zv. 35, št. 7, str. 30–37, 1998.
- [6] P. Miller, “The engineer as catalyst: Billy Klüver on working with artists”, *IEEE Spectrum*, zv. 35, št. 7, str. 20–29, 1998.
- [7] P. J. Phillips, W. T. Scraggs, A. J. O’Toole, K. W. Bowyer, P. J. Flynn, C. L. Schott in M. Sharpe, “Frvt 2006 and ice 2006 large-scale experimental results”, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, zv. 32, št. 5, str. 831–846, 2010.
- [8] B. Serexhe, ur., *Preservation of Digital Art: Theory in Praxis*. Karlsruhe: AMBRA | V, ZKM | Center for Art in Media, 2013.

- [9] J. B. Simpson, *Simpson's contemporary quotations: The most notable quotes from 1950 to the present*. HarperCollins Publishers, 1997.
- [10] F. Solina, “15 seconds of fame”, *Leonardo*, zv. 37, št. 2, str. 105–110, 2004.
- [11] F. Solina, *15 sekund slave in virtualno smučanje / 15 Seconds of Fame and Virtual Skiing. Exhibition Catalogue*. Ljubljana: ArtNetLab, 2005.
- [12] F. Solina, B. Batagelj, S. Juvan in J. Kovačič, “Color-based face detection in the “15 seconds of fame” art installation”, v *Proceedings of Mirage 2003*, NRIA Rocquencourt, France, str. 38–47, 2003.
- [13] F. Solina, G. Majcen, N. Bovcon in B. Batagelj, “Preservation of a computer-based art installation”, *EuroMed 2014*, str. 643–650, 2014.
- [14] F. Solina, P. Peer, B. Batagelj in S. Juvan, “15 seconds of fame—an interactive, computer—vision based art installation”, v *Proc. 7th International Conference on Control, Automation, Robotics and Vision*, IEEE, zv. 1, str. 198–204, 2002.
- [15] A. Trifonova, L. Jaccheri in K. Bergaust, “Software engineering issues in interactive installation art”, *International Journal of Arts and Technology*, zv. 1, št. 1, str. 43–65, 2008.
- [16] P. Viola in M. J. Jones, “Robust real-time face detection”, *International journal of computer vision*, zv. 57, št. 2, str. 137–154, 2004.
- [17] S. Štulac, “Samodejna gradnja mozaika”, Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2008.
- [18] A. Warhol, “Marilyn Monroe”, *Collection of The Andy Warhol Museum, Pittsburgh*, 1967.
- [19] A. Warhol, “Andy Warhol, Exhibition catalogue”, *Moderna Museet, Stockholm*, 1968.

- [20] Wikipedia, *Android* — Wikipedia, The Free Encyclopedia, dostopno na: [https://en.wikipedia.org/wiki/Android\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Android_(operating_system)) [zadnji obisk 26. 04. 2015].
- [21] Wikipedia, *Andy Warhol* — Wikipedia, The Free Encyclopedia, dostopno na: [https://en.wikipedia.org/wiki/Andy\\_Warhol](https://en.wikipedia.org/wiki/Andy_Warhol) [zadnji obisk 26. 04. 2015].
- [22] Wikipedia, *GIMP* — Wikipedia, The Free Encyclopedia, dostopno na: <https://en.wikipedia.org/wiki/GIMP> [zadnji obisk 26. 04. 2015].
- [23] Wikipedia, *Snapchat* — Wikipedia, The Free Encyclopedia, dostopno na: <https://en.wikipedia.org/wiki/Snapchat> [zadnji obisk 26. 04. 2015].

