Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Bernarda Pust

Mobilna aplikacija za zajemanje slik oblačil in sestavljanje stilov oblačenja

diplomsko delo

univerzitetni študijski program
računalništvo in informatika

Mentor: doc. dr. Peter Peer
Asistent: as. Bojan Klemenc

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01918/2013
Datum: 04.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BERNARDA PUST**

Naslov: **MOBILNA APLIKACIJA ZA ZAJEMANJE SLIK OBLAČIL IN SESTAVLJANJE STILOV OBLAČENJA**

MOBILE APPLICATION FOR CLOTHING IMAGE ACQUISITION AND OUTFIT STYLE COMPOSITION

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Napišite mobilno aplikacijo za zajem slik oblek, ki bodo služile za sestavljanje stilov oblačenja. Omogočite ustrezno obdelavo slik, predvsem omogočite odstranitev ozadja in ustrezne barvne korekcije. Preglejte obstoječe algoritme za ločevanje ospredja od ozadja, izberite ustreznega in ga prilagodite za uporabo v aplikaciji. Odstranjevanje ozadja naj bo čim bolj avtomatizirano. Predstavite zgradbo aplikacije in rezultate ter predlagajte izboljšave in možnosti nadaljnjega razvoja.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko
UNIVERZA V LJUBLJANI
2

Dekan:
za
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Podpisana Bernarda Pust, z vpisno številko **63070177**, sem avtorica diplomskega dela z naslovom:

Mobilna aplikacija za zajemanje slik oblačil in sestavljanje stilov oblačenja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Petra Peera in asistenta Bojana Klemenca
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 2. julija 2013 Podpis avtorice:

Na tem mestu bi se rada zahvalila moji družini, za vso pomoč in podporo med časom študija. Zahvaljujem pa se tudi mentorju, doc. dr. Petru Peeru in asistentu Bojanu Klemencu za vse nasvete in pomoč pri izdelavi diplomske naloge.

Kazalo

Povzetek

V današnjem času vedno več ljudi uporablja pametne telefone in z njimi povezane aplikacije, ki omogočajo hiter dostop do želenih informacij ali določene funkcionalnosti. Na trgu se pojavljajo vedno nove aplikacije in vsaka poskuša uporabniku ponuditi nekaj boljšega, kar prej še ni obstajalo.

Diplomsko delo se osredotoča na prototip aplikacije za sestavljanje stilov oblačenja za ciljno platformo iOS, ki je trenutno ena izmed vodilnih glede na število uporabnikov. Aplikacija omogoča shranjevanje oblačil v virtualno omaro, tako da z vgrajenim fotoaparatom slikamo nek kos oblačila ali pa ga vzamemo iz knjižnice telefona. Kasneje je mogoče vsa ta oblačila uporabiti pri kreiranju novih stilov. S tem ima uporabnik možnost, da ugotovi kaj obleči, brez da bi moral odpreti vrata svoje omare. Uporabnik lahko sliki oblačila spreminja barvne lastnosti in ji nato odstrani neželeno ozadje. Najpomembnejši del aplikacije predstavlja odstranjevanje ozadja. Odstranjevanje je narejeno s pomočjo algoritma GrabCut, ki je implementiran v knjižnici OpenCV.

Odstranjevanje ozadja pri večini tipičnih primerov uporabe deluje zadovoljivo. Pri kontrastnem ozadju je rezultat odstranjevanja primeren že po prvem koraku. Kadar pa sta ozadje in oblačilo podobnih barvnih odtenkov, je ponavadi potrebna dodatna pomoč uporabnika.

Ključne besede:

segmentacija slike, odstranjevanje ozadja, GrabCut, obdelava slike, OpenCV, kreiranje stilov, iOS, XCode

Abstract

Nowadays, more and more people are using smart phones. Applications enable fast access to information and necessary functionality. There are new applications published every day and every one of them wants to offer user something, that hasn't existed before.

The diploma thesis describes development of an application for iOS platform, which is one of the leading platforms on the market at the moment. Application offers the user to create new fashion looks from clothes that were added to a virtual closet, either with use of device's integrated camera or from the device's library. User can create look, without even opening closet doors. There is an option of changing color properties and removing unwanted background around the object of interest. Alpha matting is accomplished with the GrabCut algorithm that is implemented in OpenCV library.

The background removal is successful in most cases. The result is usually satisfactory for typical use cases. When the background and foreground contain same colors, additional user input is required. The user has to mark additional area as background or foreground.

Key words:

image segmentation, background removal, GrabCut, image editing, OpenCV, creating looks, iOS, XCode

Poglavje 1

Uvod

Danes vedno več ljudi uporablja pametne telefone in z njimi povezane aplikacije. Na trgu se pojavlja vedno več aplikacij in vsaka izmed njih uporabniku ponuja nove ali pa samo izboljšane funkcionalnosti. Ljudje imajo v današnjem času vedno manj časa, zato si marsikdo želi skrajšati čas potreben za ugotovitev, kaj obleči tisti ali pa celo naslednji dan. Včasih je bilo potrebno za planiranje stila iz omare vzeti veliko kosov oblačil, kar je časovno potratno. Oblačila velikokrat tudi ostanejo izven omare, kar zahteva še dodaten čas, potreben za pospravljanje. Na Applovi spletni trgovini za aplikacije AppStore že obstaja nekaj aplikacij, ki omogočajo sestavo novega stila na telefonu. Primeri take aplikacije so *I Wear, Stylish Girl, Dress App in Style Book*. Večina izmed njih se ne osredotoča na samo obdelavo slike in odstranjevanje ozadja, temveč le na sestavljanje stilov oblačenja. Pri tistih, ki omogočajo odstranjevanje ozadja, v veliko primerih le-to ne deluje najbolje, predvsem ko sta ozadje in objekt zanimanja podobne barve. Prav zaradi tega se prototip razvite aplikacije osredotoča na ta del, saj je izgled kreiranega stila povsem drugačen, če oblačila ne vsebujejo tudi neželenega ozadja.

Aplikacija, ki je bila razvita v okviru te diplomske naloge, omogoča uporabniku sestavljanje stilov, brez da bi moral pri tem odpreti vrata omare in iz nje vzeti kakršenkoli kos. Ko so enkrat oblačila shranjena v virtualni

omari, jih lahko uporabimo v nešteto stilih. Opis že obstoječih aplikacij in programov se nahaja v poglavju 2.

Uporabnik pridobi sliko iz knjižnice telefona ali pa s pomočjo vgrajene kamere. Na sliki je možno izvesti uravnavanje sivine, svetlosti, kontrasta in popravka game. Za uravnavanje sivine je potrebno izbrati točko, ki je v resnici siva, na sliki pa je zaradi neprimerne osvetlitve obarvana drugače. Svetlost, kontrast in gama se uravnavajo ročno s pomočjo drsnika. Odstranitev ozadja v večini poteka s pomočjo algoritmov iz knjižnice OpenCV. Glavni algoritem za odstranjevanje ozadja je GrabCut, ki pri prvem koraku zahteva označitev objekta s pomočjo pravokotnika, kasneje pa je mogoče algoritmu podati še dodatne informacije. Opis algoritmov je bolj natančno opisan v poglavju 3, njihova implementacija pa v poglavju 4.

Oblačilom je mogoče dodati opis in jim določiti kategorijo. Ko se oblačila nahajajo v virtualni omari, jih je mogoče dodati kateremukoli poljubnemu stilu. Sliko oblačila je pri kreiranju stila možno premikati, obračati in ji spreminjati velikost. Stil je možno shraniti in jih kasneje pregledovati. Zaslonske slike izgleda aplikacije se nahajajo v poglavju 5.

Delovanje odstranjevanja ozadja je zelo odvisno od slike, predvsem od podlage in oblačila. Bolj kot sta si med seboj podobna, slabši bodo rezultati odstranjevanja. Če sta zelo kontrastna, potem je ozadje odstranjeno v večini primerov že po prvem koraku. Problem povzročajo tudi sence, saj večina slik ni narejena v popolnih pogojih in z dovolj osvetlitve. Delovanje in rezultati aplikacije so opisani v poglavju 6, nadaljnje izboljšave in nadgradnje pa v poglavju 7.

Poglavje 2

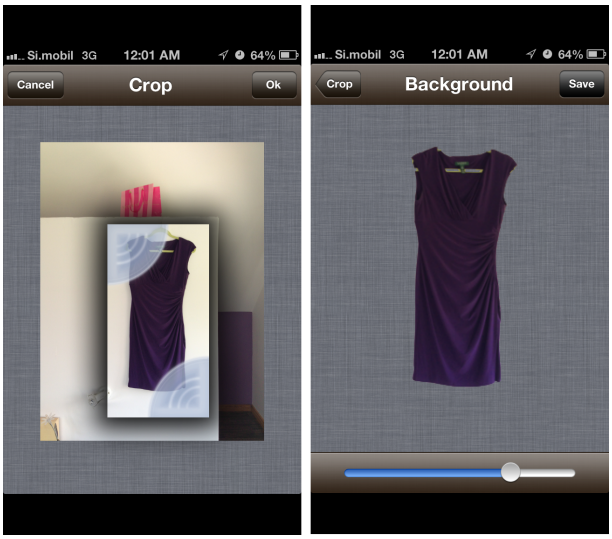
Obstoječe rešitve

Kljub temu, da obstaja že veliko podobnih aplikacij obstaja ravno toliko možnih izboljšav teh aplikacij. Obstajajo podobne aplikacije za mobilno platformo in pa tudi spletne strani, ki ponujajo sestavljanje stilov. V nadaljevanju je opisanih nekaj izmed teh primerov, nekaj prednosti in slabosti vsakega izmed njih.

I Wear[1] aplikacija ponuja možnost dodajanja novih oblačil, vendar nastane problem pri odstranitvi ozadja, ki ne deluje najbolje, saj se redkokdaj da odstraniti samo ozadje in ne tudi del oblačila. Tak primer prikazuje slika 2.1. Kadar pa sta oblačilo in ozadje kontrastna, odstranitev deluje v redu 2.2. Aplikacija je na spletni trgovini *App Store* sicer zastonj, vendar ima v tej različici omejeno možnost uporabe in za neomejeno uporabo zahteva več

[1]https://itunes.apple.com/us/app/i-wear.../id427485702?mt=8

Slika 2.1: Primer slabe odstranitve ozadja pri aplikaciji *I Wear*.

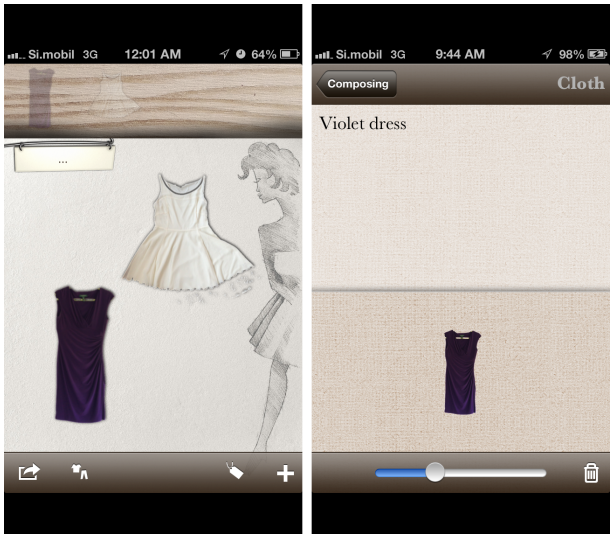


Slika 2.2: Primer dobre odstranitve ozadja pri aplikaciji *I Wear*.

nakupov znotraj aplikacije. Oblačila se lahko razvrstijo po kategorijah, ni pa možno določiti nobene dodatne lastnosti. Prav tako ni ponujenega iskanja po imenu oblačila, kar otežuje iskanje točno določenega oblačila, ko zbirka oblačil postane velika. Primer sestavljanja stila prikazuje slika 2.3. Trenutna različica aplikacije ni prilagojena velikosti zaslona pri *iPhone 5*.

Stylish Girl[2] je prav tako na voljo brezplačno na App Store. Pri dodajanju oblačila ne ponuja nobene obdelave slike in odstranjevanja ozadja in si tako shrani kar celotno sliko. Ponuja možnost, da se sliki določi veliko lastnosti. Poleg imena se lahko določi še tip oblačila, za kateri letni čas je primerno, stil, znamko, ceno, prodajalca in velikost. Način določanja lastnosti prikazuje slika 2.4. Prednost aplikacije je, da je možno izbrati neko oblačilo, za katerega so nato prikazani vsi stili, ki vsebujejo ta kos oblačila. Možno je tudi določiti, kdaj je bil stil nošen in ga dodati med priljubljene. Pri sestavljanju stila ni možna rotacija slike oblačila, ampak samo premikanje in povečevanje. Sestavi se lahko seznam, kaj vzeti s sabo na potovanje. Ponuja tudi dodajanje izdelkov iz njihove baze na seznam želja in pa shranjevanje že ustvarjenih

[2]https://itunes.apple.com/si/app/stylish-girl-your-fashion/id301669520?mt=8



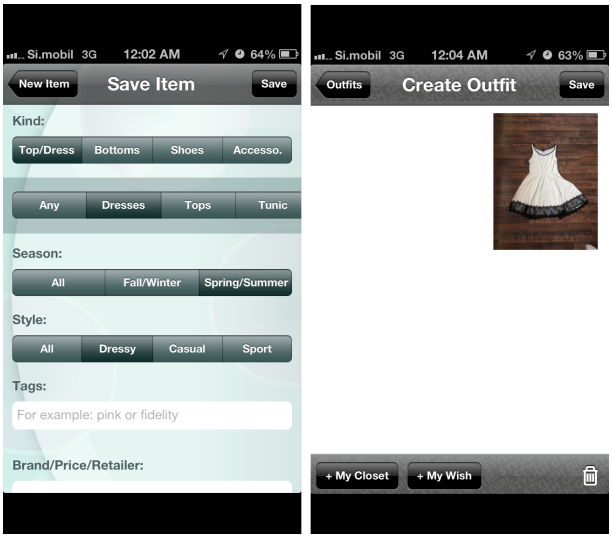
Slika 2.3: Uporaba aplikacije *I Wear*. Leva slika prikazuje primer sestavljanja stila, desna pa spreminjanje lastnosti oblačila. S pomočjo drsnika se spremeni velikost oblačila.

stilov, ki so jih pripravili drugi uporabniki. Dodajanje izdelka na seznam želja prikazuje slika 2.5.

DressApp[3] ne ponuja obreza slike niti odstranitve ozadja. Oblačilu je možno določiti znamko, ceno, letni čas, status in v katero kategorijo spada, kar prikazuje slika 2.6. Obstajajo samo štiri kategorije, dodatnih pa ni mogoče dodajati. Prednost aplikacije je v katalogu oblek, ki jih prikazuje slika 2.7, s katerega lahko naložimo določene kose v svojo omaro in jih nato uporabimo pri ustvarjanju novega stila.

Style Book[4] je plačljiva aplikacija na *App Store*, ki ima junija 2013 ceno 3.99 $. Sliki je mogoče odstraniti ozadje. Količino odstranjenega ozadja lahko prilagajamo s pomočjo drsnika, možni pa so tudi ročni popravki. Slika 2.9 prikazuje odstranjevanje ozadja. Testni primer na videu je narejen na zelo kontrastnem ozadju, kar otežuje ocenitev kvalitete odstranjevanja bolj kompleksnih in podobnih barv ozadja. Ponuja tudi nekaj navdihov za krei-

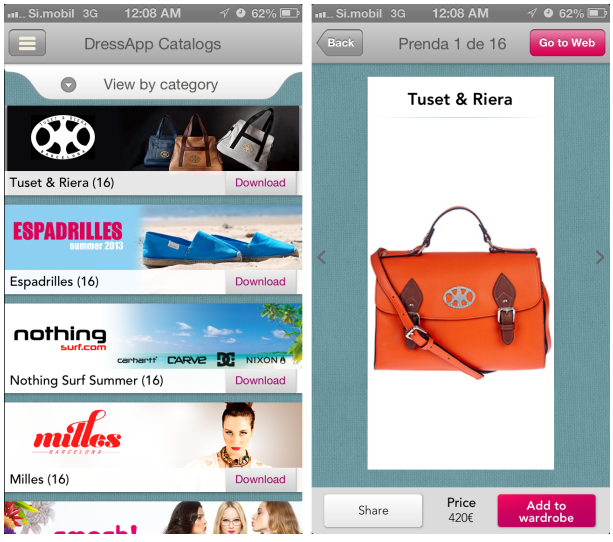
[3]https://itunes.apple.com/us/app/dressapp-your-fashion-pocket/id500213071?mt=8
[4]https://itunes.apple.com/us/app/stylebook/id335709058?mt=8



Slika 2.4: Uporaba aplikacije *Stylish girl*. Leva slika prikazuje zaslonsko sliko določanja lastnosti oblačila, desna pa sestavljanje stila s tem oblačilom, kjer se uporablja celotna slika skupaj z ozadjem.

Slika 2.5: Dodajanje čevljev iz njihove baze izdelkov na seznam želja pri aplikaciji *Stylish girl*.

Slika 2.6: Zaslonska slika določanja lastnosti oblačila pri aplikaciji *DressApp*.



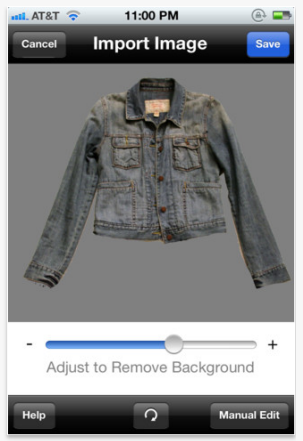
Slika 2.7: Katalog oblačil in dodatkov, ki jih lahko uporabnik doda v svojo virtualno omaro pri aplikaciji *DressApp*.

Slika 2.8: Leva slika prikazuje sestavljanje stila pri aplikaciji *Style Book*, desna pa prikaz koledarja nošenja stilov. Slika je pridobljena s strani Style Book v aplikaciji AppStore [14].

ranje stilov in omogoča ustvarjanje seznama pakiranja. Stil se lahko doda v koledar (prikazan na sliki 2.8), s pomočjo katerega je možno slediti, kdaj je bil določen stil oblečen. Nudi tudi statistiko stilov. Na uradni spletni strani aplikacije so naloženi videi, kjer poudarjajo tudi to, da je potrebno slikati oblačilo na enobarvnem brez odsevnem ozadju, ki je čim bolj v kontrastu z oblačilom. Za ozadje je priporočeno, da se na tla položi enobarvno tkanino. Drugo pravilo, ki ga poudari pa je, da so slike enakomerno osvetljene [21].

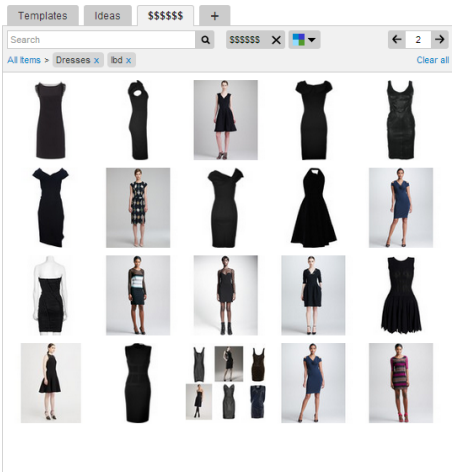
Spletna stran *Polyvore*[5] ponuja možnost sestavljanja stilov. Uporabnik ima najprej možnost izbire ozadja, kjer je ponujenih več predlog. Pri dodajanju oblačil na to ozadje, je največja pomanjkljivost to, da ni možno prosto premikati oblačil, vendar jih je mogoče dodati le v točno določena polja, kar prikazuje slika 2.10. Slika 2.11 prikazuje možnost uporabe oblačil iz baze, kar je velika prednost te strani. Uporabnik lahko doda tudi svojo sliko, vendar mora biti le-ta prej nekje objavljena, saj ne ponuja možnosti nalaganja direktno iz računalnika. Za dodajanje slike je potrebno uporabiti program Clipper[6], s pomočjo katerega se naloži sliko s spletne strani. Problem nastane, ker nekatere strani onemogočajo delovanje tega programa.

[5]http://www.polyvore.com
[6]http://www.polyvore.com/cgi/clipper



Slika 2.9: Zaslonska slika odstranjevanja ozadja s pomočjo drsnika pri aplikaciji *Style Book*. Možni so tudi ročni popravki. Slika je pridobljena s strani Style Book v aplikaciji AppStore [14].

Slika 2.10: Sestavljanje stila na spletni strani *Polyvore*.



Slika 2.11: Iskanje po virtualni bazi oblačil in dodatkov na spletni strani *Polyvore*.

Vsak od omenjenih programov ima določene slabosti in prednosti. Glavno slabost predstavlja odstranjevanje ozadja, saj nekatere aplikacije tega sploh ne ponujajo, druge pa imajo omejeno delovanje. Aplikacije prav tako nimajo nobenih barvnih nastavitev. Ker je ta del pri sestavljanju stilov vizualno zelo pomemben, se prototip aplikacije razvite v tem diplomskem delu osredotoča predvsem na obdelavo slike in odstranjevanje ozadja na sliki. Uporabljene metode so opisane v poglavju 3, implementacija pa v poglavju 4.

Poglavje 3

Uporabljene metode

Problem aplikacije med drugim predstavlja odstranjevanje ozadja, s pomočjo katerega se iz slike izloči objekt, ki nas zanima. Sliko je v nekaterih primerih zaradi neprimerne osvetlitve potrebno pred tem tudi barvno obdelati. Sliki je možno spremeniti osvetljenost, kontrast, vrednost game in ji uravnovesiti sivino. Za odstranjevanje ozadja je glavni algoritem GrabCut, opisanih pa je še nekaj ostalih podobnih algoritmov. Pri odstranjevanju se uporabi tudi Gaussovo zabrisanje in določitev prehodov. Opisi vseh teh algoritmov se nahajajo v naslednjih podpoglavjih.

3.1 Barvne transformacije

3.1.1 RGBA in prosojnost

RGBA predstavlja barvni prostor, ki je sestavljen iz rdeče barve (*angl. red*), zelene barve (*angl. green*), modre barve(*angl. blue*) ter kanala alfa (*angl. alpha*), ki hrani dodatno informacijo o stopnji prosojnosti. S pomočjo tega je možno kontrolirati, koliko je vidno skozi barvo. Vrednost 0, predstavlja popolnoma prosojno sliko [27].

3.1.2 Uravnovešenje sivine

Uravnovešenje sivine (*angl. Gray balance*) predstavlja popravljanje nerealnih barv. Sliko prilagodi tako, da siva barva v resnici predstavlja sivo tudi na sliki. Popačenje barv je lahko posledica nepopolne osvetlitve, kar povzroči da ima siva barva na sliki npr. več rumenih odtenkov [26].

Slika je prilagojena s pomočjo sprememb vrednosti *RGB*. Vrednosti se spremenijo tako, da objekti postanejo nevtralni. Postopek je lahko narejen avtomatsko ali pa z izbiro neke sivinske točke. Glede na izbrano točko se nato prilagodi celotna slika.

$$avg = \frac{R' + G' + B'}{3} \tag{3.1}$$

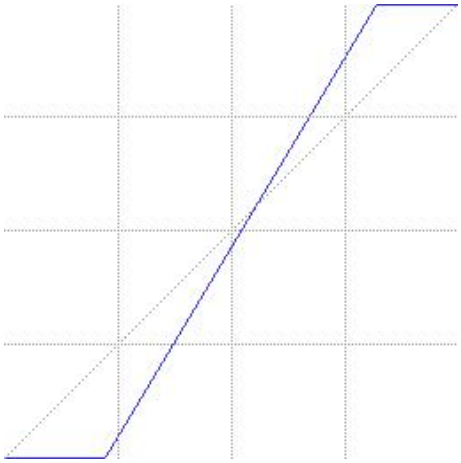
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} avg/R'_w & 0 & 0 \\ 0 & avg/G'_w & 0 \\ 0 & 0 & avg/B'_w \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{3.2}$$

Najprej je potrebno izračunati povprečno vrednost komponent izbrane točke s pomočjo formule (3.1). R',G' in B' predstavljajo začetne vrednosti komponent. Prilagajanje je nato narejeno s pomočjo formule (3.2) kjer R, G in B predstavljajo uravnotežene vrednosti slikovne pike. R'_w, G'_w in B'_w pa so vrednosti izbrane slikovne pike, ki naj bi predstavljala sivo področje [4].

3.1.3 Prilagajanje kontrasta

Prilagajanje kontrasta (*angl. contrast adjustment*) [9] predstavlja razliko med najtemnejšim in najsvetlejšim območjem slike, torej razliko v osvetljenosti. Slike z visokim kontrastom imajo široko distribucijo med belo in črno piko, slike z majhnim pa ozko. Svetel objekt na svetlem ozadju ima slab kontrast, temen objekt na svetlem ozadju pa dober kontrast.

S povečevanjem kontrasta se povečuje razlika med svetlimi in temnimi deli slike. Slike tako postanejo bolj živahne. Z nižanjem pa se med seboj



Slika 3.1: Primer grafa spremenjenih vrednosti kontrasta. Modra črta predstavlja za 40 povečano vrednost kontrasta. Os x predstavlja vrednost kontrasta vhodne slike, os y pa spremenjene izhodne.

približajo sence in najbolj osvetljeni deli slike. Prilagajanje ne deluje na celotno sliko enako, in je zelo odvisno od slike same. Slika 3.1 prikazuje povečanje kontrasta za vrednost 40.

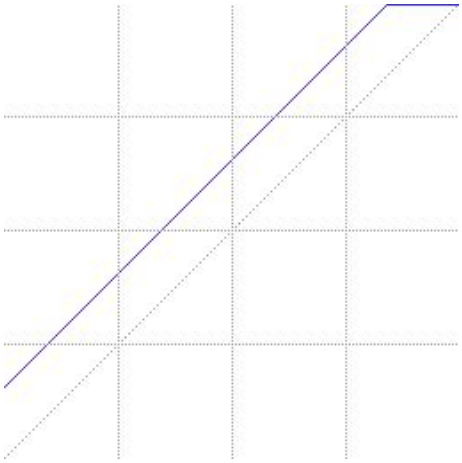
Za spremembo je najprej potreben izračun korekcijskega faktorja kontrasta s pomočjo formule (3.3), kjer C predstavlja želeno stopnjo kontrasta. V naslednjem koraku se prilagodi R, G in B glede na formulo (3.4). Stopnja kontrasta je v območju od -255 do 255. Negativne vrednosti zmanjšajo količino kontrasta, pozitivne vrednosti pa povečajo.

$$F = (259 \cdot (C + 255)) \,/\, (255 \cdot (259 - C)) \tag{3.3}$$

$$R' = F \cdot (R - 128) + 128 \tag{3.4}$$

3.1.4 Nadzor svetlosti

Osvetljenost (*angl. brightness*) [8] predstavlja osvetljenost celotne slike. Če je osvetljenost previsoka, so bele slikovne pike preveč nasičene in podrobnosti na tem območju se zmanjšajo. Če je osvetljenost prenizka, pride do nasičenosti



Slika 3.2: Primer grafa spremenjenih vrednosti svetlosti. Zgornja črta predstavlja osvetljenost povečano za vrednost 40. Os x predstavlja vrednost osvetljenosti vhodne slike, os y pa spremenjene izhodne.

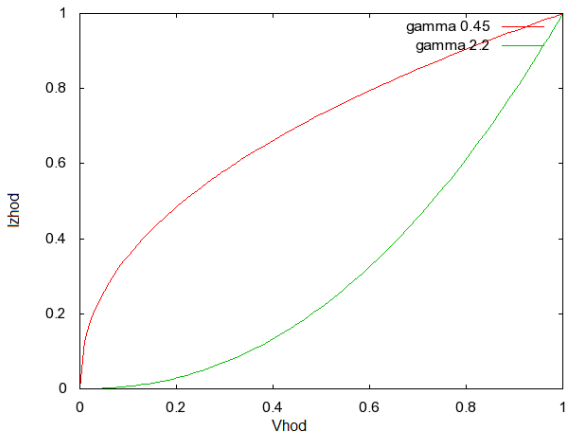
najtemnejših slikovnih pik. Graf na sliki 3.2 prikazuje spremembo vrednosti osvetljenosti, in sicer povečanje za vrednost 40.

V sivinski sliki bi lahko vsaka barvna pika na nek način predstavljala osvetljenost. Osvetljenost predstavlja območje, ki sprejme manj ali več svetlobe. Za razliko od korekcije game, osvetljenost osvetli celotno sliko enakomerno, tudi sence.

Svetlost spremenimo tako, da rdeči, zeleni in modri komponenti spremenimo vrednosti. Svetlost ima vrednosti med -255 in 255. Negativne vrednosti sliko potemnijo, pozitivne pa osvetlijo. Pri povečevanju osvetljenosti, se lahko izgubi nekaj kontrasta v najsvetlejših delih slike.

3.1.5 Korekcija game

Gama (*angl. gamma*) [10] je predstavljena z grško črko γ. Predstavlja razmerje med vhodnimi in izhodnimi vrednostmi. Izhod je sorazmeren z vhodom potenciranim z vrednostjo game. Pri velikosti game 1, je izhod linearna črta. Graf je prikazan na sliki 3.3.



Slika 3.3: Primer grafa spremenjenih vrednosti game [15].

Formula (3.5) prikazuje izračun novih vrednosti. Korekcijo game (*angl. gamma correction*) izračunamo tako, da vhod potenciramo z inverzom game, kar prikazuje formula (3.6).

$$I' = 255 \cdot (\frac{I}{255})^{\gamma} \tag{3.5}$$

$$I' = 255 \cdot (\frac{I}{255})^{1/\gamma} \tag{3.6}$$

Povečevanje game povzroči, da slika zgleda svetlejša, vendar je sprememba nelinearna. Osvetljenost se spremeni samo pri sencah in srednjih tonih. Tisti, ki pa so že svetli, ostanejo isti. Območje game je odvisno od primera uporabe, v tem delu je bilo uporabljeno od 0.25 do 7.99.

3.2 Določitev prehodov

Določitev prehodov (*angl. thresholding*) predstavlja najlažjo metodo segmentiranja. Ločevanje temelji na intenziteti slikovnih pik ozadja in objekta, ki nas zanima. Glede na problem se določi ustrezen prag (*angl. threshold*), nato se vsaka slikovna pika primerja s to vrednostjo. Vsaki slikovni piki

lahko določimo vrednost od 0, ki predstavlja črno, do 255, ki predstavlja belo barvo. S pomočjo tega ločimo regije v sliki [23].

Določitev prehodov vključuje tudi paket OpenCV, kjer obstaja 5 različnih tipov operacij. Za potrebe te diplomske naloge je uporabljena Binarna določitev pragov. Ta je predstavljena z operacijo (3.7). Če je intenziteta slikovne pike `src(x,y)` večja kot prag, potem nastavimo vrednost na MaxVal, drugače na 0 [23].

$$dst(x,y) = \begin{cases} maxVal & if src(x,y) > thresh \\ 0 & otherwise \end{cases} \tag{3.7}$$

3.3 Gaussovo zabrisanje

Gaussovo zabrisanje (*angl. Gaussian Blur*) sliko procesira in zamegli s pomočjo Gaussove porazdelitve. Vsaki slikovni piki (*angl. pixel*) določi vrednost, ki je enaka uteženi povprečni vrednosti sosednjih slikovnih pik in jo izračunamo s pomočjo Gaussove formule. Pri izračunu je uporabljena dvodimenzionalna Gaussova formula (3.8). Slika na ta način izgubi podrobnosti. S povečevanjem velikosti jedra slika postaja vedno bolj zamegljena [6].

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{3.8}$$

Uporaba metode, katere implementacija se nahaja v OpenCV:
`GaussianBlur (InputArray src, OutputArray dst, Size ksize, double SigmaX, double SigmaY, int borderType)` [6]

- `src` - vhodna slika, ki ima lahko kakršnokoli število kanalov,
- `dst` - izhodna slika, ki je enake velikosti in tipa kot src,
- `ksize` - velikost Gaussovega jedra; širina in višina sta lahko različni, vendar morata biti obe pozitivni in lihi števili,
- `SigmaX` - standardni odklon v smeri x,

- `SigmaY` - standardni odklon v smeri y.

3.4 Segmentacija slike

Pridobivanje objektov iz ospredja ima pomembno vlogo v veliko aplikacijah, kjer je možno urejati sliko ali video. Enega izmed glavnih delov prototipa aplikacije, opisanega v tem diplomskem delu, predstavlja prav odstranjevanje ozadja pri oblekah. Ozadje je potrebno pri sliki odstraniti, da lahko pri kreiranju stila uporabljamo slike, kjer ni nemotečega ozadja, ki drugače pokvari celoten izgled stila. Ozadje zavzame tudi precejšen del že tako omejenega prostora na zaslonu telefona. Pri urejanju slike je potrebno hitro in učinkovito pridobivanje ospredja oziroma objekta iz slike. Ospredje predstavlja območje na sliki, ki nas zanima. Ozadje slike pa predstavlja vse ostale slikovne pike, ki nas ne zanimajo in niso del objekta ter jih želimo odstraniti. Ločevanje objekta od ozadja izvedemo s pomočjo segmentacije. Človek lahko brez večjih težav loči objekt od ozadja, vendar za računalnik ta problem ni tako preprost. Ločevanje ospredja od ozadja zato temelji na različnih značilnostih slike, kot so npr. barva, tekstura, robovi. Najbolj pogosto uporabljena je detekcija robov, kjer opazujemo razliko med sosednjimi slikovnimi pikami. Velika razlika v barvi predstavlja rob, regijo dobimo kot rezultat zaprtega območja med zaznanimi robovi. Ta tehnika je uporabna za slike, kjer so meje jasno vidne [11]. V diplomskem delu nas zanimajo samo robovi med ospredjem in ozadjem.

Slika $I_z(z = (x, y))$ je zapisana kot kombinacija slike ospredja F_z in maskiranja ozadja B_z z uporabo α in je predstavljena z enačbo (3.9), kjer ima α_z vrednost med 0 in 1. Skrajna vrednost 1 označuje definitivno ospredje, 0 pa definitivno ozadje, vmesne vrednosti pa označujejo neznano slikovno piko z. Večina pristopov ločevanja ospredja in ozadja se zanaša na uporabnikovo začetno označitev ozadja ali ospredja [24].

Če lahko α zasede samo vrednosti 0 ali 1 potem pride do binarne segmentacije, kjer vsaka slikovna pika pripada samo ospredju ali samo ozadju.

Večina algoritmov pri segmentiranju poleg slike dobi še en vhodni podatek o treh regijah (*angl. trimap*), kamor so mapirane točke. Tri regije predstavljajo ozadje, ospredje in vmesno neznano regijo. Uporabnik lahko te regije določi ročno. Problem ločevanja slike je preveden na ocenjevanje ozadja, ospredja in α za vsako slikovno piko v neznanem območju glede na znane slikovne pike, ki pripadajo ozadju ali ospredju. Nekateri algoritmi zahtevajo zelo natančno označevanje, medtem ko je pri drugih potrebno samo nekaj oznak ospredja in ozadja. Bolj kot so tri regije natančno določene, bolj uspešno je ločevanje. Če uporabnik z rezultatom ni zadovljen, lahko tri regije popravi in ponovno zažene algoritem [24].

Sosednje slikovne pike s podobnimi barvami imajo običajno podobne vrednosti α. S pomočjo znanih slikovnih pik, ki pripadajo ospredju ali ozadju, se izračuna vrednosti α na neznanem območju [24].

$$I_z = \alpha_z F_z + (1 - \alpha_z) B_z \tag{3.9}$$

3.4.1 Načini segmentacije

Obstaja več načinov za segmentacijo slike, med drugim tudi metoda Čarobne paličice (*angl. Magic wand*) [18], Interaktivna segmentacija s pametnimi škarjami (*angl. Interactive segmentation with intelligent scissors*) [16], Metoda mehkih škarij (*Soft scissors*) [25], Ruzonova in Tomasijeva metoda (*angl. Ruzon and Tomasi's Method*) [19], Bayesovo ločevanje (*angl. Bayes matting*) [18], Metoda izbijanja (*angl. Knockout*) [**?**], Geodetsko maskiranje (*angl. Geodesic matting*) [1], Metoda minimalnega reza v grafu (*angl. Graph Cut*) [2] in GrabCut (*slo. Algoritem označi-reži*), ki je uporabljen v tem diplomskem delu [18]. GrabCut je predstavljen v poglavju 3.4.2.

Uporabnik mora določiti slikovne pike, kjer naj bi bila meja med ozadjem in ospredjem. Začetna razdelitev, ki jo naredi uporabnik, se lahko zelo razlikuje. Nekateri algoritmi potrebujejo zelo natančno označitev, kjer je potrebno označiti rob objekta, pri drugih pa je dovolj že označitev dela ospredja in ozadja ali pa samo obris objekta s pomočjo pravokotnika.

Čarobna paličica

Pri čarobni paličici (*angl. Magic wand*) uporabnik izbere točko, ki se uporabi za izračun regije podobnih slikovnih pik z določeno toleranco. Problem tega pristopa je izbrati pravilno toleranco, ki jo je včasih celo nemogoče določiti. Če se porazdelitev v barvnem prostoru med ospredjem in ozadjem prekriva, algoritem ne doseže želene segmentacije [18].

Interaktivna segmentacija s pametnimi škarjami

Interaktivna segmentacija (*angl. Interactive segmentation with intelligent scissors*) se uporabja pri ročni segmentaciji slike. Uporabnik mora na grobo označiti celoten objekt. Pametne škarje omogočajo hitro pridobivanje objektov s pomočjo označitve objekta. Ko se miška nahaja v bližini roba objekta, se obris dotakne meje in objame celoten objekt. Zaznavanje objekta deluje s pomočjo iskanja optimalne poti v uteženem grafu. Optimalno iskanje po grafu zmanjša občutljivost na šum. Občutljivost poveča tudi sprotno učenje, ki preverja podobnost tipu roba, kateremu trenutno sledi in ne išče samo najbolj poudarjenega v okolici. Glavni problem tega orodja nastane pri zelo veliki ali pri majhni teksturi, ko obstaja med dvema točkama več minimalnih poti, kar posledično pomeni, da mora uporabnik dodati veliko dodatnih točk [16].

Mehke škarje

Metoda mehkih škarij (*angl. Soft scissors*) je interaktivno orodje za segmentacijo slike. Sposobno je izločiti objekt s kompleksnim ospredjem kot je žival z dlako. Algoritem deluje sproti z uporabnikovim označevanjem robov ospredja [25]. Uporabnik na ta način sproti vidi, kako bo izgledalo ospredje slike.

Ruzonova in Tomasijeva metoda

Pri Ruzonovi in Tomasijevi metodi (*angl. Ruzon and Tomasi's Method*) je vsaka verjetnostna porazdelitev v barvnem prostoru modelirana kot Gaussova množica [19]. Za barvo med njima se izračuna vrednost α s pomočjo ocene podobnosti. Neznana regija je označena kot ozko območje, ki označuje rob ospredja. To območje je predstavljeno z verigo slikovnih pik. Pri slikah, kjer ima neznana regija veliko teksturo, pride do večjih napak. Vrednosti α so izračunane neodvisno od sosednjih slikovnih pik zaradi, česar se lahko pojavijo luknje v končni segmentaciji [19].

Bayesovo maskiranje

Pri Bayesovem maskiranju (*angl. Bayes matting*) uporabnik določi tri regije, ki jih sestavljajo ospredje, ozadje in neznano območje. Za neznano območje se izračuna vrednosti α. Algoritem vrne dobre rezultate, ko to območje ni preveliko. Za označitev območij je potrebna pomoč uporabnika [18]. Če je tekstura slike zelo različna in so tri regije slabo razdeljene, imajo tudi rezultati veliko šuma. Ker algoritem izhaja iz Ruzonove in Tomasijeve metode, tudi tukaj neznano območje predstavlja ozek pas okoli objekta, ki predstavlja ospredje. Za definicijo sosednjih slikovnih pik se širita znani območji ozadja in ospredja. Za določitev barvne porazdelitve se poleg znanih vzorcev ozadja in ospredja uporabi tudi sosednje vrednosti F_s, B_s in α_s tako, da vsaka slikovna pika v območju prispeva h Gaussovemu ospredju in ozadju. Maskiranje je rešeno z uporabo maksimalne posteriorne verjetnosti [24].

Metoda izbijanja

Pri Metodi izbijanja (*angl. Knockout*) je barva ospredja za neznano slikovno piko izračunana kot utežena vsota barv znanih sosednjih slikovnih pik. Uteži so odvisne od razdalje med pikama. Tudi barva ozadja je izračunana na enak način in nato prilagojena glede na pozicijo slikovne pike in barvo ospredja. Vrednosti α so najprej izračunane za vsak barvni kanal posebej, njihova

utežena skupna vrednost pa predstavlja α vrednost slikovne pike [24].

Geodetsko maskiranje

Geodetsko maskiranje (*angl. Geodesic matting*) meri uteženo geodetsko razdaljo pri naključnem sprehodu od izhodiščne točke do ospredja. Čas izračuna razdalje se linearno povečuje in za izračun ne potrebuje veliko pomnilnika, kar omogoča hitro pridobitev visoko kvalitetnih rezultatov na podlagi uporabnikove označitve le dela slike [1].

Metoda minimalnega reza v grafu

Metoda minimalnega reza v grafu (*angl. Graph Cut*) je interaktivna segmentacijska tehnika, ki omogoča doseg segmentacije, tudi kadar ozadje in ospredje nista dobro razdeljena. Cilj je razdeliti sliko na dva dela in sicer na objekt in na ozadje. Izvor in ponor grafa predstavljata vozlišči ospredja in ozadja. Uporabnik označi del slike kot definitivno ozadje in na podlagi tega algoritem loči ostali del slike. Če uporabnik spremeni ali poda dodatne informacije, se segmentacija hitro spremeni. Pomanjkljivost tega algoritma je, da deluje samo na sivinskih slikah, zato je v naslednjem poglavju 3.4.2 opisan algoritem GrabCut, ki predstavlja razširitev na barvne slike in je uporabljen pri aplikaciji opisani v tem diplomskem delu [2].

3.4.2 GrabCut

V tem diplomskem delu je za odstranitev ozadja uporabljen algoritem GrabCut, ki za tipične primere uporabe izdelane aplikacije vrača primerne rezultate in deluje sprejemljivo hitro. Drugi razlog za izbiro tega algoritma je tudi, da za delovanje ne potrebuje veliko pomoči uporabnika. Pred prvo segmentacijo mora uporabnik obrisati objekt le s pomočjo pravokotnika. Poleg tega je algoritem implementiran v knjižnici OpenCV, ki jo lahko neposredno uporabimo na izbrani platformi iOS.

GrabCut [18] (*slov. algoritem označi-reži*) predstavlja razširitev algoritma *Graph Cut*, ki rešuje problem največjega pretoka/ minimalnega reza. Algoritem določi minimalni strošek rezanja, ki bo popolnoma prekinil povezave med vozliščema ospredja in ozadja. Graf z dodatnima vozliščema ospredja in ozadja prikazuje slika 3.4. Strošek predstavlja vsota uteži, ki morajo biti odrezane. Medtem ko je Graph Cut namenjen samo sivinskim slikam, je uporaba GrabCut razširjena tudi na barvne slike. Algoritem se uporablja za interaktivno odstranitev ozadja objekta v kompleksnem okolju. Cilj algoritma je doseči čim večjo uspešnost odstranjevanja ozadja s čim manjšim sodelovanjem uporabnika [7].

Segmentacija z algoritmom Graph Cut

Graph Cut je predstavljen v raziskavi, ki sta jo objavila Boykov and Jolly [2] in predstavlja izhodno točko za GrabCut. Algoritem deluje samo na sivinskih slikah. Slika je predstavljena kot tabela vrednosti $\mathbf{z} = (z_1, ..., zn, ..., z_N)$, ki predstavljajo posamezne slikovne pike. Segmentacijo slike pa predstavlja tabela vrednosti $\underline{\alpha} = (\alpha_1, ..., \alpha_N)$, kjer posamezne vrednosti predstavljajo prosojnost vsake slikovne pike. Vrednosti $\underline{\alpha}$ so lahko med 0 in 1, vendar je pri začetni segmentaciji le-ta poenostavljena in zasede samo vrednosti 0 in 1, kjer 0 predstavlja oznako ozadja, 1 pa ospredja. Parameter $\underline{\theta}$ opisuje porazdelitev sivih tonov ospredja in ozadja ter je sestavljen iz histogramov sivinskih vrednosti (3.10). Histogrami so pridobljeni iz začetnih treh regij T_B (ozadja) in T_F (ospredja). Naloga segmentacije je ugotoviti neznane vrednosti $\underline{\alpha}$ iz podatkov o sliki z in modela $\underline{\theta}$ [18].

$$\underline{\theta} = \{h(z; \alpha), \alpha = 0, 1\} \tag{3.10}$$

Minimizirana Gibbsova energijska funkcija(3.11) je konstruirana tako, da njen minimum predstavlja dobro segmentacijo. Pri tem upošteva sivinska histograma ospredja in ozadja slike ter skladnost prosojnosti. Izraz U (3.12) ocenjuje prileganje prosojnosti $\underline{\alpha}$ podatkom $\mathbf{z}$, glede na histogram modela

$\underline{\theta}$. Gladkost V je zapisana kot (3.13), kjer C predstavlja množico parov sosednjih slikovnih pik in *dis()* Evklidsko razdaljo med pari. V praksi dobre rezultate dosežemo, če so za sosednje slikovne pike izbrane tiste, ki so povezane vertikalno, horizontalno ali diagonalno (8 smeri) [18].

$$E\left(\underline{\alpha}, \underline{\theta}, \mathbf{z}\right) = U\left(\underline{\alpha}, \underline{\theta}, \mathbf{z}\right) + V\left(\underline{\alpha}, \mathbf{z}\right) \tag{3.11}$$

$$U\left(\underline{\alpha}, \underline{\theta}, \mathbf{z}\right) = \sum_{n} -logh\left(z_n; \alpha_n\right) \tag{3.12}$$

$$V\left(\underline{\alpha}, \mathbf{z}\right) = \gamma \sum_{(m,n)\in C} dis(m,n)^{-1}\left[\alpha_n \neq \alpha_m\right] e^{-\beta\left(z_m - z_n\right)^2} \tag{3.13}$$

Boykov in Jolly predlagata, da se za vrednost β izbere vrednost večja od 0 in se izračuna po formuli(3.14). Matematično upanje vzorca slike označuje $\langle\rangle$. Takšna vrednost β zagotavlja, da eksponentni del v formuli (3.13) ustrezno zaznava visok in nizek kontrast. Konstanta γ ima vrednost 50 in je bila določena na podlagi primerjave s pravilno segmentacijo petnajskih slik [18].

$$\beta = \left(2\left\langle\left(z_m - z_n\right)^2\right\rangle\right)^{-1} \tag{3.14}$$

$$\underline{\alpha} = argmin_{\underline{\alpha}} E(\underline{\alpha}, \underline{\theta}) \tag{3.15}$$

Segmentacija je narejena na podlagi globalnega minimuma izraza (3.15). Minimizacija je narejena z običajnim iskanjem minimalnega reza v grafu pretokov in je podlaga za začetno ločevanje ospredja in ozadja pri GrabCut. Sivinski model histogramov, uporabljen pri Graph Cut, pri GrabCut zamenjamo z Mešanim Gaussovim modelom (*angl. Gaussian Mixture Model - GMM*), saj ustreznih histogramov ni praktično zgraditi. Algoritem minimalnega rezanja z enim korakom nadomestimo z iterativnim postopkom. Tretja sprememba pa je, da mora uporabnik označiti izmed treh regij samo T_B s pomočjo pravokotnika, s katerim označi objekt, ostali dve pa lahko ostaneta neoznačeni [18].

Segmentacija z algoritmom GrabCut

Slika je sestavljena iz n slikovnih pik (z_n) v barvnem prostoru RGB. Zaradi nepraktičnosti histogramov v barvnem prostoru se uporabi GMM. Algoritem potrebuje dva GMM, in sicer enega za ospredje in enega za ozadje. Vsak izmed njiju predstavlja Gaussov kovariančni model s K komponentami. Za sledljivost GMM se uporablja dodatni vektor $\mathbf{k} = \{k_1, ..., k_n, ..., k_N\}$, kjer je $k_n \in \{1, ...K\}$, in dodeli vsaki slikovni piki unikatno GMM komponento z ospredja ali ozadja glede na to ali ima α_n vrednost 0 ali 1 [18].

Dopolnjena energijska Gibbsova formula je predstavljena z enačbo (3.16). Izraz U (3.17) uporabi barvni model GMM, kjer se D izračuna s pomočjo formule (3.18). Funkcija p predstavlja Gaussovo verjetnostno porazdelitev, π pa utežene koeficiente. Parameteri modela so sedaj $\underline{\theta} = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k), \alpha = 0, 1, k = 1...K\}$, kjer π predstavlja uteži, μ sredino in Σ kovarianco 2K Gaussovih komponent za ospredje in ozadje. Gladkost se izračuna po formuli (3.19) [18].

$$E\left(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}\right) = U\left(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}\right) + V\left(\underline{\alpha}, \mathbf{z}\right) \tag{3.16}$$

$$U\left(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}\right) = \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n) \tag{3.17}$$

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -logp(z_n|\alpha_n, k_n, \underline{\theta}) - log\pi(\alpha_n, k_n) \tag{3.18}$$

$$V\left(\underline{\alpha}, \mathbf{z}\right) = \gamma \sum_{(m,n)\in C} \left[\alpha_n \neq \alpha_m\right] e^{-\beta\|z_m - z_n\|^2} \tag{3.19}$$

Namesto enega koraka minimiziranja energije je pri GrabCut delovanje iterativno. V prvem koraku se določi k_n vrednost vsem n slikovnim pikam. Drugi korak je implementiran kot množica ocenjevalnih postopkov Gaussovih

parametrov. Če za komponento k v modelu GMM vzamemo model ospredja, potem je podmnožica slikovnih pik definirana kot $F(k) = \{z_n : k_n = k \text{in } \alpha_n = 1\}$. Sredina $\mu(\alpha, k)$ in kovarianca $\Sigma(\alpha, k)$ sta ocenjeni kot sredina in kovarianca vrednosti slikovnih pik v množici *F(k)*. Utež $\pi(\alpha, k) = \frac{|F(k)|}{\sum_k |F(k)|}$, kjer $|F(k)|$ predstavlja velikost množice F(k). Zadnji korak predstavlja globalno optimizacijo z iskanjem minimalnega reza. Vsak izmed teh treh korakov iterativne minimizacije je lahko prikazan kot minimizacija celotne energije E. E se zmanjšuje monotono in konvergira vsaj k enemu lokalnemu minimumu. Ko ni več znatnega zniževanja energije, se iteriranje zaključi [18].

Iterativni minimizacijski algoritem dovoljuje, da uporabnik označi samo del slike kot ozadje T_B, T_F pa ostane prazna. Slikovne pike T_U imajo na začetku začasne oznake, s časom pa pripadejo ospredju ali pa ozadju. Oznake ozadja T_B pa ostanejo nespremenjene. V tej implementaciji je začetni T_B označen s pomočjo pravokotnika, s katerim uporabnik označi objekt in tako loči znano ozadje od neznanega ospredja. Območje, ki se nahaja izven njega pripada ozadju [18].

Ko sta objekt, ki nas zanima, in ozadje podobne barve in teksture je potrebno dodatna pomoč uporabnika. Uporabnik s pomočjo prsta na zaslonu označi del slike kot definitivno ozadje ali ospredje in s tem algoritmu poda nove informacije. Označiti je potrebno samo del napačno segmentiranega ozadja [18].

Postopek izvajanja algoritma GrabCut je naslednji [18]:

I. Nastavitev vrednosti

- Uporabnik označi objekt zanimanja s pomočjo pravokotnika. Območje izven pravokotnika označuje T_B. Ospredje T_F je nastavljeno na 0, neznano območje T_U pa kot komplement ozadja $\overline{T}_B$.
- Za vsak $n \in T_B$ vrednost α_n nastavi na 0, vsak $n \in T_F$ pa nastavi na 1.
- GMM ozadja in ospredja sta inicializirana glede na množici, kjer je $\alpha_n = 0$ in $\alpha_n = 1$.

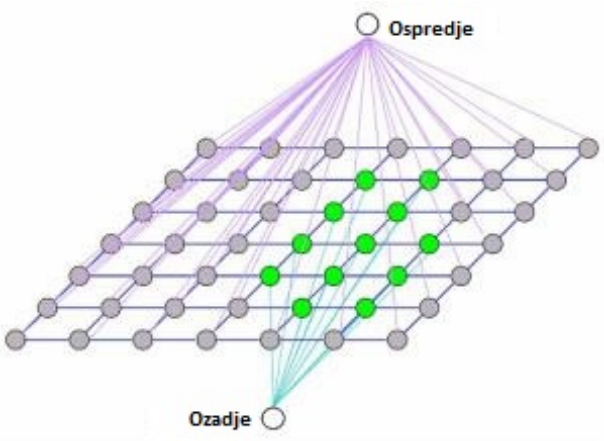
II. Iterativna minimizacija

1. Določi GMM komponente vsaki slikovni piki. Za vsak $n \in T_U$ je $k_n := argmin_{k_n} D_n(\alpha_n, k_n, \theta, z_n)$.
2. Nauči se GMM parametre iz podatkov $\mathbf{z}$: $\underline{\theta} := argmin_{\underline{\theta}} U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$.
3. Segmentiranje z uporabo minimalnega reza za rešitev: $min_{\{\alpha_n : n \in T_U\}}\ min_{\mathbf{k}} \mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$
4. Ponavljanje korakov od 1 do 3 do konvergiranja.
5. Uporabi dobljeno masko in določi mehke prehode med ospredjem in ozadjem.

III. Sodelovanje uporabnika

Uporabnik, s pomočjo prsta na zaslonu, označi del narobe označenih slikovnih pik kot ozadje, s čimer se nastavi $\alpha_n = 0$ ali kot ospredje, kjer je $\alpha_n = 1$.

- Temu sledi izvajanje tretje točke iterativne minimizacije.
- Lahko se ponovi izvajanje celotne iterativne minimizacije.



Slika 3.4: Graf predstavlja prikaz vozlišč slikovnih pik in dodatni vozlišči za ospredje in ozadje. [7]

Poglavje 4

Implementacija

Poglavje opisuje način imlementacije metod, ki so opisane v prejšnjem poglavju. Ker knjižnica OpenCV že vsebuje kar nekaj izmed teh metod, predstavlja le-ta pomemben del diplomske naloge. Poleg določitve prehodov in Gaussovega zabrisanja vsebuje tudi implementacijo najpomembnejšega algoritma GrabCut s pomočjo katerega deluje odstranjevanje ozadja. Nekaj več splošnih informacij o OpenCV je opisanih v poglavju 4.1. Ciljno platformo za implementacijo aplikacije predstavlja operacijski sistem iOS, ki deluje na pametnih telefonih. Za razvoj se uporablja razvojno okolje XCode, ki ga je mogoče namestiti na operacijskem sistemu OS X in je nameščeno na računalnikih podjetja Apple. Več podatkov o iOS in Xcode se nahaja v poglavju 4.2.

4.1 OpenCV

OpenCV[1] je odprtokodna knjižnica za računalniški vid (*angl. Open Source Computer Vision Library*). Razvilo ga je podjetje *Intel*. *OpenCV* je odprtokodna knjižnica z *licenco BSD*, ki omogoča spreminjanje in vključevanje v drugo kodo. Sestavljena je iz več sto algoritmov za računalniški vid. Trenutno je možno naložiti različice 2.x, ki so napisane v *programskem jeziku C++*,

[1] http://opencv.org/

medtem ko so pri različicah 1.x uporabljali *programski jezik C*. Med drugim ga je možno uporabiti na platformah Windows, iOS, Android in Linux [17].

Paket vključuje več knjižnic. Na voljo je več modulov: [17].

- *core* – med drugim vključuje večdimenzionalno tabelo Mat, ki jo uporabimo za matrike,
- *imgproc* – modul za procesiranje slik;
- *video* – uporabno za obdelavo videa;
- *calib3d* – osnovni geometrijski algoritmi;
- *features2d* – prepoznavanje značilk;
- *objdetect* – zaznavanje objektov;
- *highgui* – vmesnik za zajem videa, slike;
- *gpu* – pospešeni algoritmi iz drugih modulov;
- še nekatere podporne module.

V tem diplomskem delu je uporabljen modul *imgproc*, ki med drugim vsebuje tudi implementacijo algoritma GrabCut.

4.2 iOS in XCode

Mobilni operacijski sistem *iOS* je razvilo podjetje *Apple*. Na trgu se je prvič pojavilo leta 2007 z začetkom prodaje prve generacije telefona *iPhone*. Jeseni 2012 je bilo na njihovi spletni trgovini za aplikacije *App Store* objavljenih kar 700.000 različnih aplikacij. Do takrat je bilo opravljenih 30 milijard prenosov. iOS predstavlja 21% delež uporabnikov pametnih telefonov in 43.6% delež uporabnikov tablic na svetu [12]. Prav zaradi razširjenosti tega operacijskega sistema je bil le-ta izbran za platformo pri implementaciji aplikacije opisane v tej diplomski nalogi.

Vmesnik sestavljajo elementi kot so drsniki, stikala in gumbi. Omogoča zaznavanje več točkovnih dotikov. Operacijski sistem je narejen na osnovi operacijskega sistema *OS X*, ki se uporablja na računalnikih *Apple*. Trenutno aktualna različica je 6.1.3 in je naložena na *iPhone, iPad, iPod touch in Apple TV*. Obstaja pa že tudi beta različica iOS 7, ki pa je v času pisanja na voljo samo za razvijalce [12] .

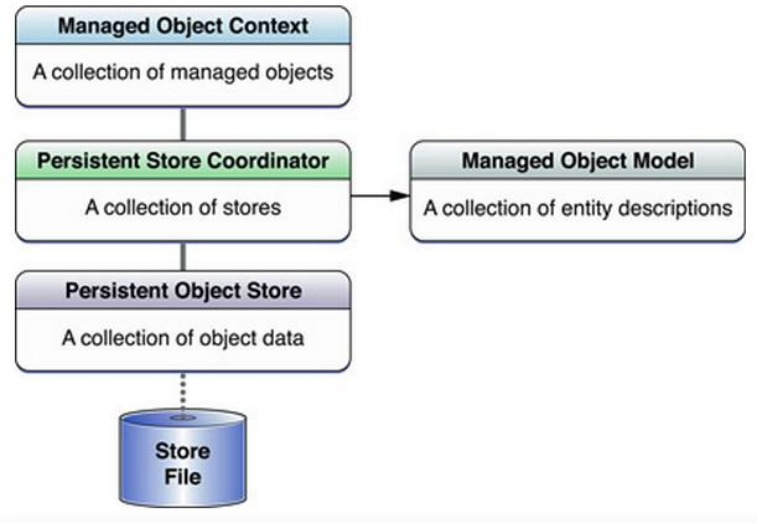
Za razvoj je potreben programski razvojni paket (*angl. software development kit*) in v *OS X* vgrajeno razvojno okolje *XCode*, ki vključuje urejevalnik kode in grafični urejevalnik uporabniškega vmesnika. Zagotovi orodja za obvladovanje celotnega razvojnega toka. Za razvoj se v večini uporablja objektno orientirani programski jezik `Objective-C` (*slo. Objektni-C*), možno pa je dodatno uporabiti tudi drug jezik [20].

Aplikacijo je pred objavo na trgovini *App Store* potrebno testirati tudi na dejanskih napravah, ne samo s pomočjo simulatorja. Za testiranje na napravi je potrebno najprej registrirati telefon in aplikacijo podpisati z razvijalskim certifikatom. Aplikacijo za objavo je potrebno registrirati preko *iTunes Connect*. Ko naložimo še binarne datoteke, Apple pregleda ustreznost aplikacije za objavo, in če ni nobenih pripomb, se aplikacija objavi [20].

Za hranjenje podatkov je uporabljeno ogrodje Core Data, ki je bolj podrobno opisano v poglavju 4.2.1. Grafični vmesnik aplikacije pa je narejen s pomočjo scenarija, ki ga predstavi poglavje 4.2.2.

4.2.1 Core Data

Core Data je ogrodje, ki ga ponuja Apple [13] in je namenjeno obvladovanju, kje in kako so podatki shranjeni. Uporablja se pri hranjenju objektnega modela (*angl. model object*) v datoteko in ponovnem dostopu do teh podatkov. Na voljo je od različice operacijskega sistema iOS 3.0 naprej. Omogoča avtomatsko podporo za razveljavitev (*angl. undo*) in ponovitev (*angl. redo*) in možnost hranjenja manjše podmnožice objektnega modela v pomnilniku. Model je narejen s pomočjo grafičnega vmesnika. Starejšo različico uporabniške datoteke je mogoče nadgraditi na novejšo.



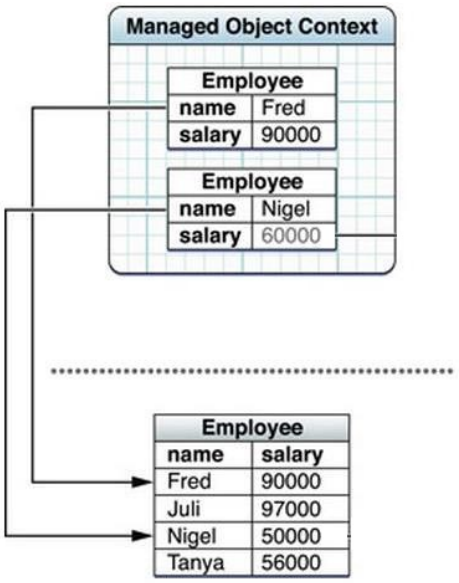
Slika 4.1: Sklad ogrodja Core Data [13].

Omogoča komunikacijo *SQLite* in Objective-C brez eksplicitnega ustvarjanja povezave ali urejanja sheme podatkovne baze. Zmanjša tudi čas, potreben za sestavo primernih *SQL* poizvedb in ročno obvladovanje le-teh ter izhodnih podatkov.

Shema je definirana v datoteki s končnico *.xcdatamodel*. Projektu je za uporabo potrebno dodati ogrodje Core Data. Entiteta je poimenovana kot upravljan objekt (*angl. managed object*) in predstavlja objekt, ki bo vseboval zbirko podatkov. Entiteta vsebuje atribute, ki pa lahko hranijo tudi odnos do drugih entitet s katerimi so povezani in s pomočjo katerih pridobimo ostale povezane podatke.

Sklad predstavlja objekte ogrodja Core Data, ki med sabo sodelujejo, za pridobivanje in shranjevanje podatkov v skladišče (*angl. persistent store*). Prikazan je na sliki 4.1. Skladišče predstavlja datoteka, kjer so shranjeni vsi podatki. Za hranjenje je med drugim možno uporabiti *SQLite*.

Upravljani objekt je podrazred razreda `NSManagedObject`, ki predstavlja generični razred in implementira vse osnovne funkcionalnosti, ki jih potrebuje objekt Core Data. Predstavlja zapis objekta v tabeli podatkovne baze,



Slika 4.2: Upravljana objekta v kontekstu in tabela skladišča [13].

torej podatke, s katerimi operiramo v aplikaciji. Upravljani objekt je vedno povezan s kontekstom. Kontekst upravljanega objekta (*angl. managed object context*) je primerek razreda `NSManagedObjectContext`, ki predstavlja prostor objekta. Odgovoren je za obvladovanje zbirke (*angl. collection*) upravljanih objektov, ki so med seboj povezani. Nov upravljani objekt se vstavi v kontekst, prav tako sem shranimo tudi podatke ki jih pridobimo iz podatkovne baze. Vse spremembe so shranjene v pomnilniku, dokler konteksta ne shranimo v skladišče.

Slika 4.2 prikazuje dva upravljana objekta, ki pripadata dvema zapisoma v podatkovni bazi. En podatek je bil spremenjen v pomnilniku, vendar sprememba še ni bila shranjena v bazo. V bazi sta še dva dodatna podatka, ki pa se ne nahajata v kontekstu.

Model upravljanega objekta (*angl. managed object model*) je primerek razreda `NSManagedObjectModel` in predstavlja shemo, ki opiše podatkovno bazo. Model je zbirka opisov entitet (primerek `NSEntityDescription`). Entitetni opis hrani podatke o imenu entitete, o lastnostih in o razredu, ki je

uporabljen za predstavitev entitete v aplikaciji. Vsak upravljani objekt ima referenco na entiteto, kateri pripada. *Core Data* uporabi model za povezavo med objekti v aplikaciji in zapisi v podatkovni bazi.

Koordinator skladišča (*angl. Persistent store coordinator*) ima centralno vlogo pri tem, kako *Core Data* obvladuje podatke. Z njim ni direktne interakcije. Večina aplikacij ima samo eno skladišče, vendar je pri zahtevnih aplikacijah mogoče imeti več skladišč. Naloga koordinatorja je obvladovanje teh skladišč in prikaz več skladišč navzven kot eno samo. V fazi pridobivanja podatkov le-te pridobimo iz vseh skladišč.

4.2.2 Scenarij

Scenarij (*angl. storyboard*) vizualno predstavlja uporabniški vmesnik aplikacije in je dostopen v okolju XCode. Prikaže zaporedje krmilnikov pogleda (*angl. UIViewController*) in povezave (*angl. segue*), ki predstavljajo njihovo zaporedje. S pomočjo scenarija se oblikuje uporabniški vmesnik, saj omogoča postavljanje gumbov, tabel, slikovnih in tekstovnih polj. Omogoča tudi povezavo teh elementov s samim krmilnikom pogleda.

Poglavje 5

Aplikacija “My Style“

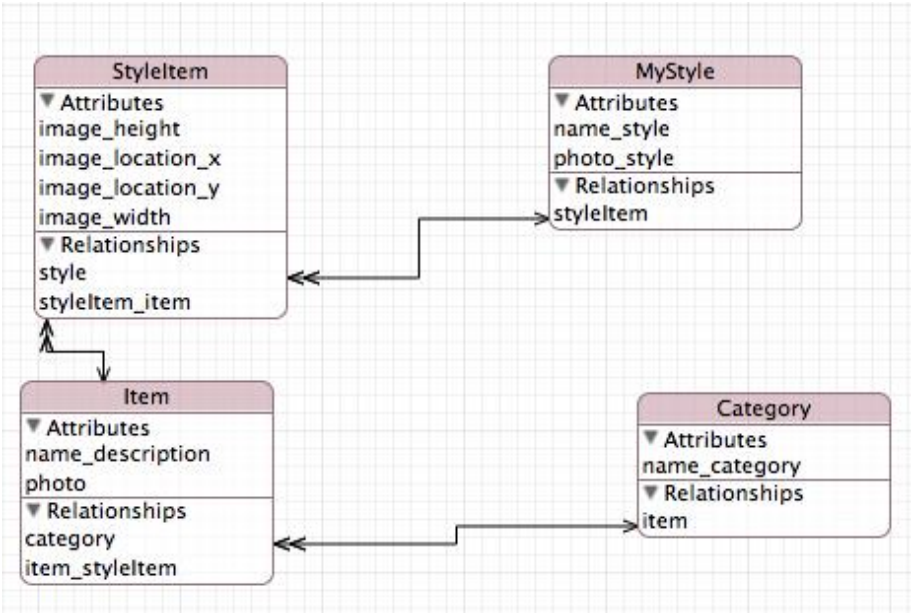
5.1 Implementacija aplikacije

Aplikacija je napisana v okolju *XCode* in je razvita za različico *iOS 6*, saj ob razvoju same aplikacije verzija iOS 7 še ni bila dostopna. Pri razvoju je veliko algoritmov za obdelavo slike uporabljenih iz *knjižnice OpenCV*, ki med drugim vsebuje tudi glavni algoritem za odstranitev ozadja slike `GrabCut`.

5.1.1 Podatkovni model

Podatkovni model prototipa je sestavljen iz štirih entitet, ki so prikazane na sliki 5.1. Glavni entiteti sta poimenovani `Item` in `MyStyle`. Entiteta `Item` hrani sliko oblačila in pa opis tega oblačila. Ima tudi dve razmerji z entitetama s katerima je povezana. Razmerje `category` predstavlja povezavo z entiteto `Category`, `item_styleItem` pa razmerje do entitete `StyleItem`. Entiteta `Category` ima le en atribut in sicer ime kategorije oblačil, prav tako pa ima shranjeno tudi razmerje item do entitete `Item`. `MyStyle` vsebuje 2 atributa, in sicer ime stila in pa sliko celotnega stila ter eno razmerje `styleItem`. `StyleItem` hrani podatke o tem kje se določena slika oblačila na ekranu nahaja in njena velikost ter povezavi do entitet `Item` in `MyStyle`. Povezave z entitetami omogočajo pridobivanje med seboj povezanih podatkov.

Pri nadgradnji aplikacije, kjer bi imel uporabnik možnost dodajanja svo-



Slika 5.1: Prikazuje podatkovni model hranjenja podatkov v aplikaciji.

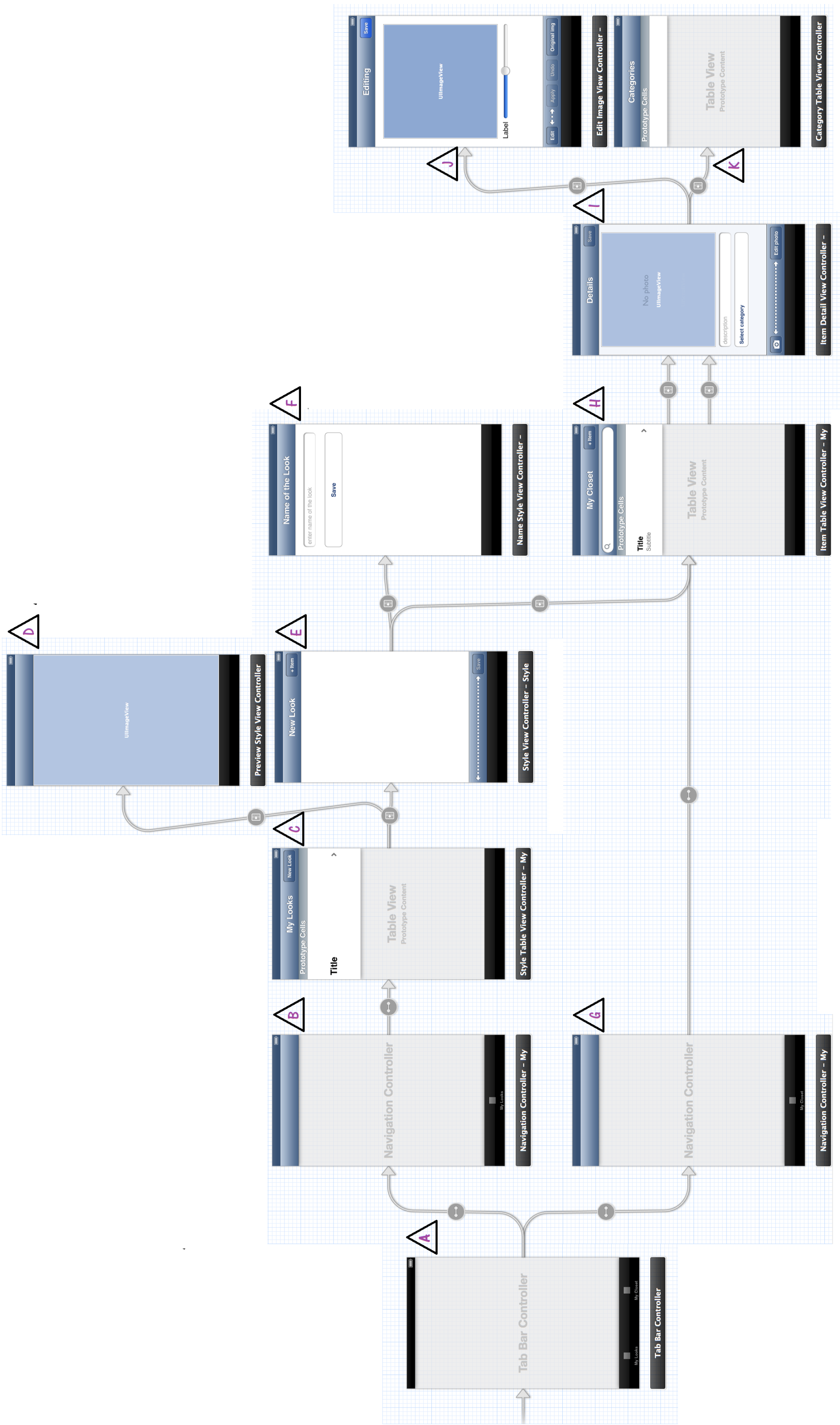
jih metapodatkov, bi bilo model potrebno ustrezno prilagoditi. Namesto entitete `Category` bi lahko dodali dve novi entiteti in sicer eno za poimenovanje nove kategorije, ter drugo za seznam imen, ki bi pripadal tej kategoriji. S tem bi uporabnik dobil možnost sestavljanja poljubnih filtrov iskanja.

5.1.2 Scenarij

Slika 5.2 prikazuje scenarij celotne aplikacije. Vstop v aplikacijo poteka preko `TabBarController`, znotraj katerega se nato pojavita dva krmilnika navigacije (*angl. Navigation Controller*), kjer en predstavlja kreiranje stilov, drugi pa prikaz in dodajanje novih oblačil v virtualno omaro. Bolj natančen opis posameznih delov se nahaja v naslednjih podpoglavjih.

Style Table View Controller

Slika krmilnika pogleda `Style Table View Controller` je prikazana na sliki 5.2 pod črko *C*. Glavni del predstavlja tabela, kjer so prikazani že ustvarjeni



Slika 5.2: Zasloni in oblika celotne aplikacije.

stili. Vsaka celica v tabeli ima na levi strani sliko stila, na desni pa njegovo poimenovanje. S klikom na stil se premaknemo na zaslon `Preview Style View Controller`. V zgornji vrstici se na desni strani nahaja tudi gumb, ki nam omogoča kreiranje novega stila in s tem povezavo na zaslon `Style View Controller`.

Style View Controller

`Style View Controller` je prikazan pod sliko 5.2 pod črko *E*. Sredinski del zaslona predstavlja prazen pogled, na katerega lahko dodajamo oblačila. V zgornjem desnem kotu se nahaja gumb, s pomočjo katerega se stilu doda novo oblačilo iz virtualne omare. Drugi gumb pa se nahaja v desnem spodnjem kotu in nam omogoča shranitev stila, ko smo enkrat z njim zadovoljni. S pomočjo tega je uporabnik preusmerjen na zaslon `Name Style View Controller`. Ko je dodano novo oblačilo, je sliko možno premikati, obračati, spreminjati njeno velikost in jo izbrisati.

Preview Style View Controller

`Preview Style View Controller` ima samo en element, in sicer slikovni gradnik (*angl. UIImageView*), ki omogoča prikaz ustvarjenega stila in ne omogoča nobene povezave na nadaljnje krmilnike pogleda, možna pa je vrnitev na prejšnjega. Prikazan je na sliki 5.2 pod črko *D*.

Name Style View Controller

`Name Style View Controller` zaslon je prikazan na sliki 5.2 pod črko *F*. Zaslon vsebuje tekstovno polje za vnos besedila oziroma poimenovanja stila. Pod njim se nahaja gumb, ki omogoča dokončno shranitev stila. Po kliku gumba je narejena preusmeritev na začetni pogled, kjer je stil sedaj dodan v tabelo obstoječih stilov.

Item Table View Controller

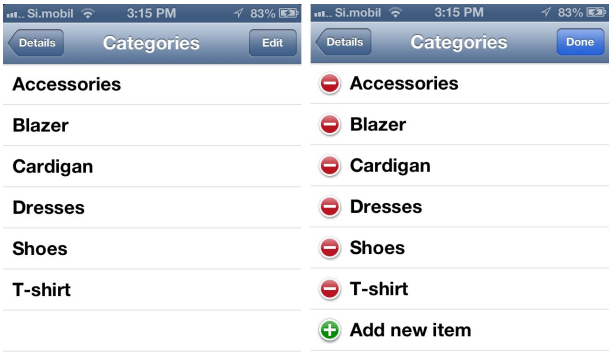
`Item Table View Controller` je prikazan na sliki 5.2 pod črko *H*. Glavni del okna predstavlja tabela. Vsaka celica ima na levi strani sliko oblačila, na desni pa naziv in kategorijo. S klikom na celico nas preusmeri na podrobnosti oblačila. Nad tabelo se nahaja polje za iskanje po tabeli oblačil. V zgornjem desnem kotu pa je gumb, ki omogoča dodajanje novega oblačila v virtualno omaro in s pomočjo katerega je narejena preusmeritev na isti pogled kot pri podrobnostih.

Item Detail View Controller.

`Item Detail View Controller` je prikazan na sliki 5.2 pod črko *I*. Osrednji del predstavlja slikovni gradnik, kjer je prikazana slika oblačila. Sliko dodamo s pomočjo gumba z ikono kamere, ki se nahaja v spodnjem levem kotu in nam ponudi možnost slikanja ali pa izbire slike iz knjižnice telefona. Pod sliko se nahaja tekstovno polje, v katerega vpišemo opis oblačila, pod njim pa je gumb za izbiro kategorije `Select category`, ki nas preusmeri na pogled, kjer so naštete kategorije. Sliko je možno tudi urejati in sicer s klikom na gumb `Edit Photo`, ki se nahaja v spodnjem desnem kotu. Ob kliku na gumb nas preusmeri na krmilnik `Edit Image View`. V zgornjem desnem kotu je gumb za shranitev oblačila, ki nas po shranitvi vrne na prejšnji zaslon.

Edit Image View Controller

`Edit Image View Controller` je prikazan na sliki 5.2 pod črko *J*. Največji del zaslona zavzema slikovni gradnik, kjer je prikazana slika ki se jo trenutno ureja. Pod njo se nahaja drsnik, s pomočjo katerega pri določenih metodah spreminjamo vrednosti. V spodnji vrstici se nahajajo štirje gumbi. Prvi gumb `Edit` ponuja izbiro metode, ki jo želimo izvesti na sliki. Ko smo z želenim učinkom zadovoljni je potrebno pritisniti gumb `Apply`. Če nam željen učinek ni všeč, lahko le-tega razveljavimo s pomočjo gumba `Undo`, vendar je to možno narediti le za zadnji korak. Zadnji gumb, `Original img`, omogoča



Slika 5.3: Na levi strani je prikazana zaslonska slika seznama obstoječih kategorij, na desni pa zaslonska slika po pritisku gumba `Edit`, kjer je vidno dodajanje novih in odstranjevanje obstoječih kategorij.

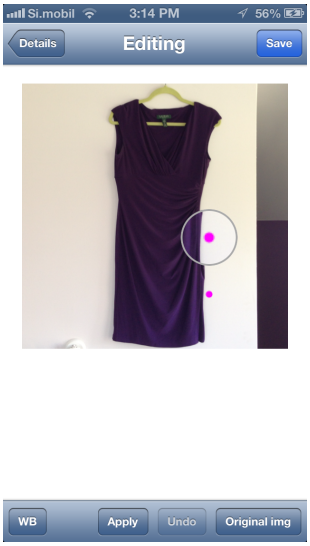
razveljavitev vseh do sedaj izvedenih metod. Urejanje je potrebno shraniti s pritiskom na gumb `Save`, ki se nahaja v zgornjem desnem kotu, in vrne uporabnika na prejšnji zaslon. Več o samem delovanju metod urejanja je opisanih v poglavju 5.2.

Category Table View Controller

`Category Table View Controller` je prikazan na sliki 5.2 pod črko *K*. Glavni del predstavlja tabela kategorij oblačil. Programsko se v zgornji desni kot doda gumb `Edit`, ki nam s pritiskom na znak plus v vrstici tabele omogoča dodati novo kategorijo. S pritiskom na znak minus pa je določeno možno kategorijo izbrisati. Delovanje dodajanja in brisanja je prikazano na sliki 5.3.

5.2 Jedro aplikacije

Jedro prototipne aplikacije predstavljata zajem in obdelava slike. Možnih je več načinov urejanja barve slike in odstranjevanje ozadja. Aplikacija ponuja tudi urejanje sivine, kontrasta, osvetlitve in game.

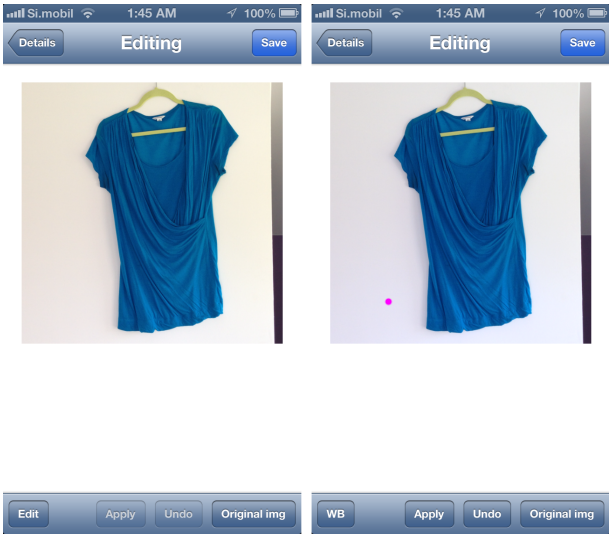


Slika 5.4: Zaslonska slika povečevalnega stekla pri uravnovešenju sivine.

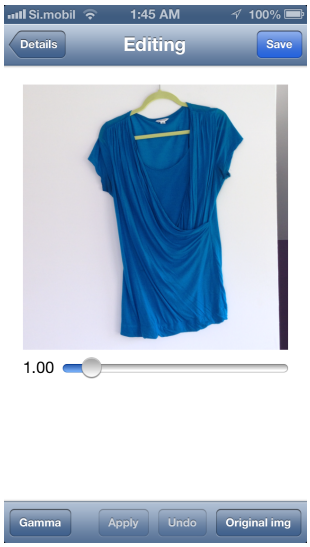
Uravnovešenje sivine deluje na podlagi izbrane točke na sliki, ki naj bi v resnici predstavljala sivo barvo. Sam opis metode se nahaja v poglavju 3.1.2. Točko se izbere s pritiskom na želeno mesto. V primeru, da izbrana točka ni na pravilnem mestu, jo je možno premakniti. Če je pritisk daljši od dveh sekund, se nad prstom na zaslonu pojavi povečevalno steklo, ki omogoča bolj natančno postavitev točke. Prikaz povečane slike je prikazan na sliki 5.4, samo delovanje uravnavanja pa na sliki 5.5.

Prilagajanje kontrasta je opisano v poglavju 3.1.3. Ob izbiri te metode se pod sliko prikaže drsnik in polje za izpis trenutne vrednosti kontrasta. Začetna vrednost je nastavljena na 0. Kontrast je možno zmanjšati do -255 in povečati do 255. S premikom drsnika v desno se kontrast povečuje, v levo pa zmanjšuje. Delovanje metode je prikazano na slikah 5.6 in 5.7.

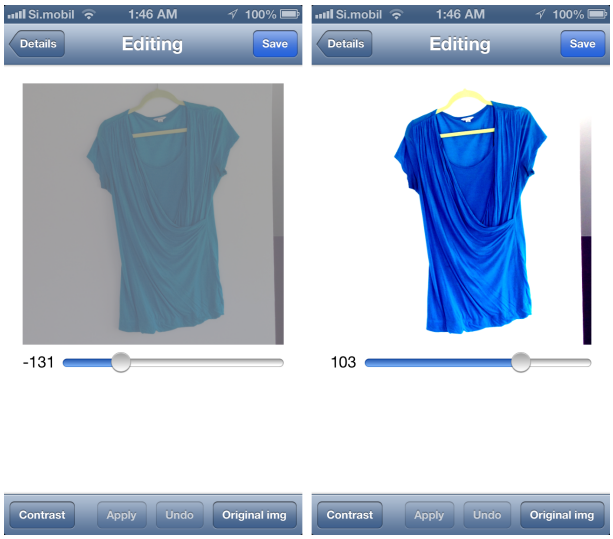
Korekcija game je opisana v poglavju 3.1.5. Prikaže se drsnik, ki ima začetno vrednost nastavljeno na 1. Minimalna vrednost je nastavljena na 0.25, maksimalna pa na 7.99. S pomikom drsnika v desno se gama povečuje in slika postaja svetlejša ter obratno v drugo smer. Prikaz delovanja je prikazan na sliki 5.8. Originalna slika je prikazana na sliki 5.6.



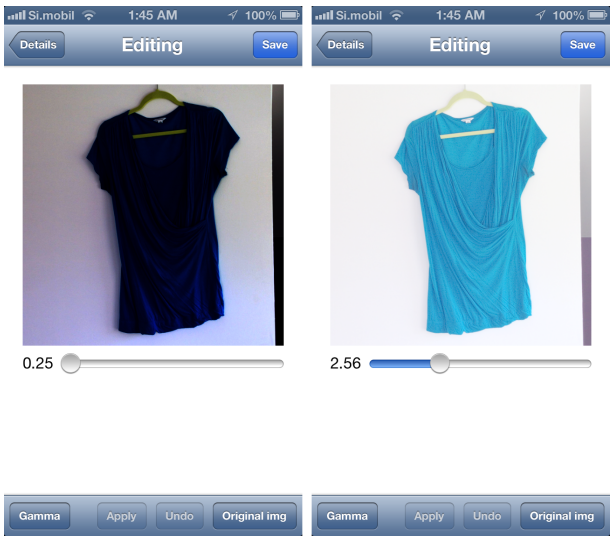
Slika 5.5: Zaslonska slika delovanja uravnovešenja sivine. Na levi se nahaja slika pred uporabo metode, na desni pa njen rezultat.



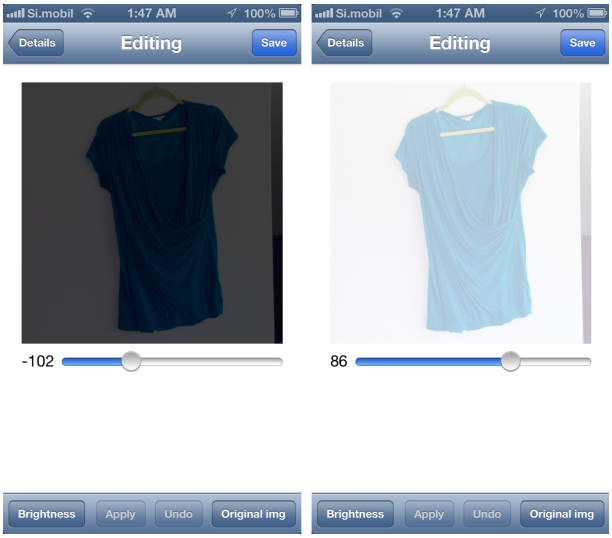
Slika 5.6: Zaslonska slika originalne slike oblačila. Vrednost kontrasta in osvetljenosti sta 0, gama pa ima vrednost 1.



Slika 5.7: Zaslonska slika rezultata prilagajanja kontrasta. Na levi strani je prikaz slike kjer je kontrast zmanjšan na vrednost -131, na desni pa je kontrast povečan na 103.



Slika 5.8: Zaslonska slika rezultata korekcije game. Na levi je prikazana znižana vrednost game na 0.25, na desni pa rezultat pri povišani vrednosti na 2.56.



Slika 5.9: Zaslonska slika rezultata nadzora svetlosti. Na levi je prikazan rezultat pri znižani svetlosti na vrednost -102, na desni pa pri zvišani na 86.

Nadzor svetlosti je opisan v poglavju 3.1.4. Na zaslonu se pojavi drsnik, ki ima začetno vrednost nastavljeno na 0. Minimalna vrednost, ki jo je možno izbrati je -255, maksimalna pa 255. S pomikom drsnika v desno se svetlost povečuje, v levo pa zmanjšuje. Delovanje je predstavljeno na sliki 5.9, originalna slika pa je prikazana pod 5.6.

Odstranjevanje ozadja je narejeno s pomočjo algoritma `GrabCut`. Pri sami implementaciji je uporabljen že napisan algoritem iz knjižnice *OpenCV*. Več o samem algoritmu je opisano v poglavju 3.4.2. Prvi korak predstavlja označitev objekta s pomočjo pravokotnika. Ob izbiri metode se na zaslonu pojavi pravokotnik, ki ga je mogoče premikati in mu spreminjati velikost, ter ga tako prilagoditi da objame objekt, ki nas zanima. Pri izbiri slike iz knjižnice ali ustvarjanju nove s pomočjo kamere, je potrebno oblačilo postaviti na sredino. Objekt se ne sme dotikati nobenega od robov. Pravokotnik ne sme pokrivati celotnega območja slike, saj potem pri algoritmu največji pretok/minimalni rez ponor ali izvor ni povezan z ostalimi vozlišči. Za pravilno delovanje mora obvezno obstajati področje zunaj pravokotnika, za katerega

Slika 5.10: Prva slika prikazuje pravilno začetno označitev objekta zanimanja. V primeru, da je rezultat začetne odstranitve ozadja nezadovoljiv, je potrebno dodatno označevanje. Na srednji sliki je z rdečo barvo prikazano dodatno označevanje ozadja, na zadnji pa z rumeno označevanje ospredja.

se privzame da je ozadje. Pravilna označitev je prikazana na sliki 5.10.

S pomočjo drsnika, ki se nahaja pod sliko je možno izbrati velikost slike, na kateri se bo algoritem izvedel. Če je drsnik pomaknjen na levo, se algoritem izvede zelo hitro, vendar je rezultat slabši predvsem pri robovih, ki postanejo bolj nazobčani. Kadar je drsnik pomaknjen desno, izvajanje postane počasnejše, vendar je rezultat boljši. Če je drsnik povsem na desni, se algoritem izvede nad originalno velikostjo slike. Ko je objekt obrisan in velikost izbrana je potrebno pritisniti gumb `Apply`, ki se nahaja v spodnji vrstici.

Če rezultat prvotne odstranitve ozadja ni zadovoljiv, je možno dodatno označiti dele slike in ponovno zagnati algoritem. Ponujena je možnost označitve ozadja in ospredja. Pri označevanju ospredja oziroma območja zanimanja se transparentno prikaze že izrezan del slike. Označevanje ospredja je prikazano na zadnji zaslonski sliki na sliki 5.10. Ni potrebno označiti celotnega dela, temveč samo del, saj algoritem s tem pridobi nove 100% pravilne

informacije, v katero območje spada. Popravljanje ozadja prikazuje srednja slika na sliki 5.10.

Poglavje 6

Rezultati

Aplikacija je sestavljena iz dveh delov. Prvi del predstavlja zavihek `My Looks`, kjer je možno pregledovati že obstoječe stile in kreirati nove. Ta del je bolj podrobno opisan v poglavju 6.1. Drugi del pa je `My Closet`, kjer je ponujen pregled celotne virtualne omare in pa dodajanje nove slike ter njeno urejanje. Delovanje tega dela aplikacije in rezultati urejanja slike so opisani v poglavju 6.2.

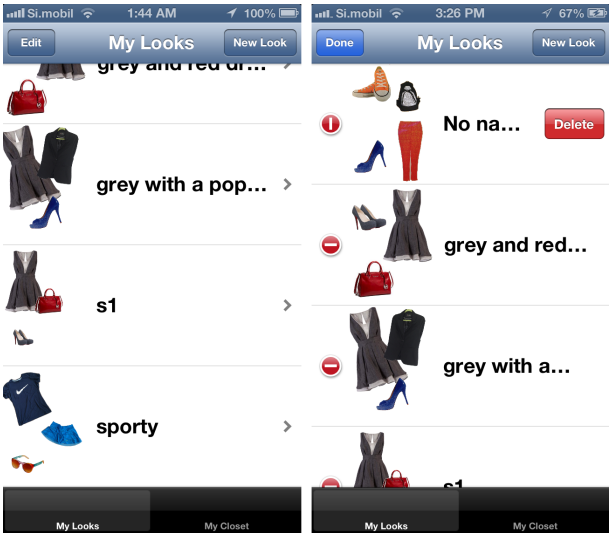
6.1 Rezultati dela aplikacije "My Looks"

Ob zagonu aplikacije se najprej odpre zavihek z že ustvarjenimi stili. Zaslonska slika le-tega je prikazana pod sliko 6.1. S klikom na eno izmed vrstic se nam prikaže že obstoječ stil. Obstoječ stil je v prototipni aplikaciji možno le pogledati in pa izbrisati, ni pa ga možno tudi urejati.

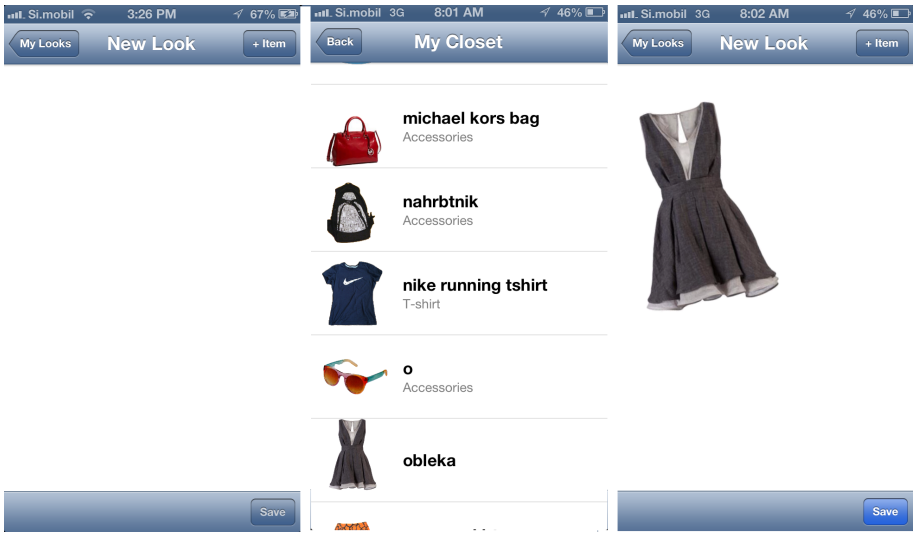
S klikom na gumb `New look` se odpre prazno okno prikazano na sliki 6.2. Na njega je možno dodajati objekte, ki so shranjeni v virtualni omari.

Gumb `+Item` omogoča dodajanje objekta iz virtualne omare. Po omari je omogočeno iskanje glede na naziv oblačila. Zaenkrat iskanje ni narejeno tudi po kategorijah. Samo iskanje omogoča hiter dostop do želenega oblačila. Delovanje iskanja je prikazano na sliki 6.3

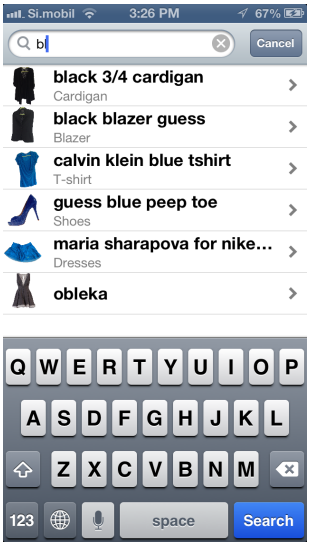
Ko je stilu dodan prvi objekt, se omogoči shranjevanje. Če se uporabnik



Slika 6.1: Zaslonski sliki prikaza seznama že ustvarjenih stilov. Desna slika prikazuje način brisanja stila.



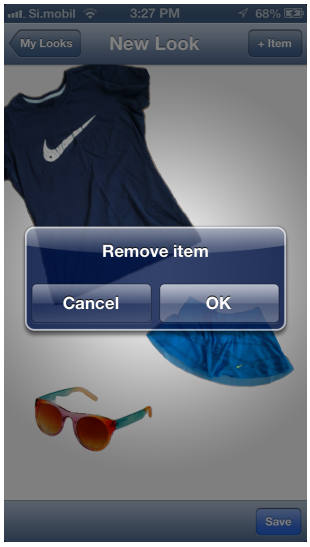
Slika 6.2: Prva slika prikazuje zaslonsko sliko, pred dodajanjem oblačila. Na drugi je prikazan seznam oblačil, ki jih lahko uporabnik doda. Tretja slika pa prikazuje ponovno zaslon iz prve slike, vendar se sedaj na njem že nahaja izbrano oblačilo.



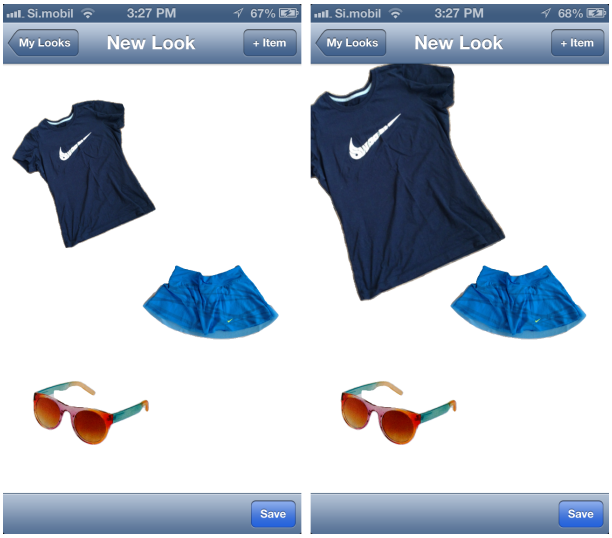
Slika 6.3: Zaslonska slika prikaza iskanja po virtualni omari z iskalnim nizom “bl”. Iskanje v prototipni verziji poteka samo po opisu oblačil, ne pa tudi po ostalih metapodatkih.

kasneje premisli in želi objekt izbrisati, je to možno narediti z dvakratnim klikom na objekt. Pred dokončnim izbrisom aplikacija še preveri, če uporabnik to res želi storiti. Pojavi se opozorilo, kjer odstranitev potrdi ali pa zavrne. Brisanje objekta je prikazano na sliki 6.4.

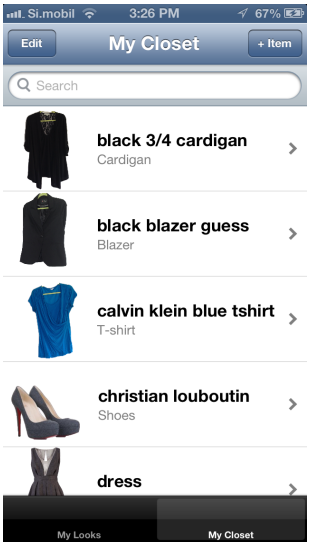
Dodano sliko oblačila je mogoče po zaslonu premikati, ji spreminjati velikost in jo zavrteti. Povečava slike je prikazana na sliki 6.5.



Slika 6.4: Zaslonska slika brisanja objekta. Aplikacija preveri če uporabnik res želi objekt odstraniti s pomočjo opozorila.



Slika 6.5: Prikaz rezultata spreminjanja velikosti oblačila, kjer je črna majica na levi strani manjša kot na desni sliki.



Slika 6.6: Zaslonska slika prikaza virtualne omare.

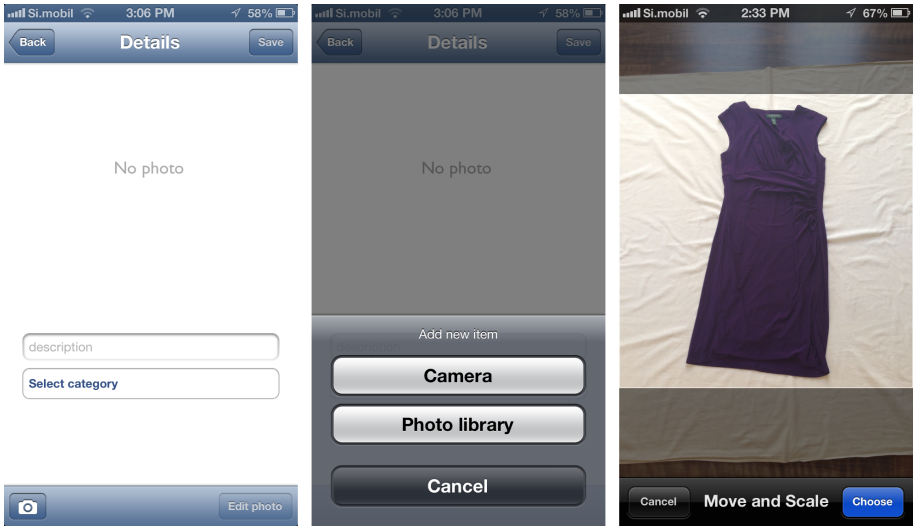
6.2 Rezultati dela aplikacije "My Closet"

Drugi zavihek aplikacije predstavlja `My Closet` oziroma virtualna omara. Na sliki 6.6 je prikazan seznam oblačil, ki so že bila dodana v virtualno omaro. Z izbiro oblačila se prikažejo podrobnosti oblačila v novem oknu. Informacije je možno urejati, prav tako pa je možna kasnejša dodatna obdelava slike.

S klikom na `+Item` se odpre enako okno, kot če je bilo izbrano oblačilo, le da je le-to v tem primeru prazno. Slika okna je prikazana pod sliko 6.7. Sliko oblačila je možno pridobiti iz knjižnice telefona ali pa jo narediti s pomočjo kamere na telefonu. Če želi uporabnik uporabiti sliko z medmrežja in ne svojih lastnih oblačil, je potrebno sliko najprej shraniti na telefon. Način izbire zajema slike je prikazan na sliki 6.7.

Ko je slika enkrat izbrana, jo je potrebno prilagoditi tako, da se slika nahaja znotraj kvadrata, ki ga prikazuje slika 6.7. Sliko je mogoče premikati in ji spreminjati velikost. Ko je uporabnik s postavitvijo zadovoljen, izbiro potrdi z gumbom `Choose`.

Sama odstranitev ozadja je v veliki meri odvisna od slike. Bolj kot je ozadje kontrastno v primerjavi s samim objektom zanimanja, boljši so rezul-



Slika 6.7: Prva slika prikazuje zaslonsko sliko dodajanja novega oblačila v virtualno omaro. Na drugi je prikazan način zajema slike oblačila. Zadnja slika pa prikazuje prilagoditev slike pri zajemu.

tati in hitrejše je delovanje algoritma. Zaradi počasnosti delovanja algoritma na telefonu, je bil dodan drsnik, s pomočjo katerega lahko uporabnik izbere velikost slike, na kateri bo izvajal algoritem. Velikost 640 predstavlja velikost originalne slike. Glede na izbiro velikosti je odvisno delovanje algoritma in pa zaobljenost robov objekta. Pri manjši velikosti slike so robovi bolj nazobčani, delovanje pa hitrejše, pri večji pa so robovi bolj gladki, delovanje pa se upočasni.

V primeru da je oblačilo, ki nas zanima, slikano na kontrastnem ozadju, je rezultat odstranjevanja v veliki večini primerov zadovoljiv že po prvi iteraciji in dodatno označevanje ospredja in ozadja ni potrebno. Primer odstranjevanja je prikazan na sliki 6.8. Oblačilo je slikano na kontrastnem ozadju, prav tako pa je bila primerna tudi označitev objekta s pravokotnikom. Ustrezen rezultat odstranjevanja je bil tako pridobljen že po prvi iteraciji algoritma `GrabCut`.

Kljub kontrastnemu ozadju, je včasih potrebno dodatno sodelovanje uporabnika, kadar so na sliki sence. Če se noben del sence ne pojavi zunaj

Slika 6.8: Originalna slika na levi strani in rezultat odstranitve ozadja na desni.

pravokotnika, potem te v večini primerov ostanejo del ospredja. Ta del je v ponavadi odstranljiv že v naslednji iteraciji, ko ta del uporabnik s pomočjo prsta na zaslonu označi kot definitivno ozadje. Tak primer je prikazan na sliki 6.9. Ker je svetloba na sliki padala iz desne strani, je na levi strani oblačila ostalo nekaj ozadja.

Kadar je objekt postavljen na ozadje, ki je podobne barve, algoritem deluje nekoliko slabše in je potreben dodaten vnos podatkov. Uporabnik s pomočjo prsta na zaslonu označi del slike, ki je trenutno v napačni kategoriji, torej je trenutno na primer v ozadju, moralo pa bi predstavljati ospredje ali obratno. Kljub vsem popravkom pa je v nekaterih primerih rezultat odstranjevanja manj uspešen. Primer takega odstranjevanja je prikazan na sliki 6.10. Oblačilo je bele barve in je postavljeno na beli steni. Po začetni iteraciji je bilo na sliki še nekaj neželenih delov, ki so označeni z rdečo barvo. S tem se algoritmu poda dodatne informacije, da je ta označen del definitivno ozadje. Vendar je na tretji sliki vidno, da je na desni strani slika malo preveč obrezana in je odstranjen tudi majhen del obleke.

Delovanje pa je v končni fazi odvisno tudi od uporabnikove izbire oblačila s pomočjo pravokotnika. Z njim mora čim bolj natančno obrisati objekt zanimanja, saj v primeru da označi premajhen del, v ozadje pade tudi del oblačila. Do velikih odstopanj pri rezultatih pride predvsem takrat, ko ozadje ni enobarvno in konstantno. Bolj kot je natančen v tej fazi obrisa, boljši so

Slika 6.9: Odstranitev ozadja, kjer ostanejo deli, ki so nekoliko senčni. Prva slika prikazuje označitev interesnega območja. Na drugi je prikazan rezultat prve iteracije odstranitve ozadja. Ker ozadje ni bilo popolnoma odstranjeno, je na tretji sliki uporabnik z rdečo barvo dodatno označil ozadje. Četrta slika pa prikazuje rezultat druge iteracije algoritma, ko je ozadje uspešno odstranjeno.

Slika 6.10: Prikaz slabe odstranitve ozadja, ko sta objekt in ozadje podobne barve.

Slika 6.11: Primer pravilne označitve objekta zanimanja na sliki.

Slika 6.12: Označitev ozadja, kjer je del zunaj pravokotnika povsem drugačen kot ozadje, ki se nahaja v neposredni bližini oblačila.

rezultati. Če ozadje ni konstantno in uporabnik označi objekt preveč na široko, algoritem v veliko primerih ne bo popolnoma obrezal ozadja in potrebni so kasnejši popravki. Primer pravilne označitve ozadja je prikazan na sliki 6.11, kjer je bilo ozadje odstranjeno v enem koraku in ni bilo potrebnega dodatnega sodelovanja uporabnika. Na sliki 6.12 je prikazan primer, ko je bil pravokotnik postavljen povsem pravilno, saj bi v primeru zmanjšanja pravokotnika odstranili tudi del oblačila. Problem tega primera je, da se ozadje, ki se nahaja zunaj pravokotnika povsem, razlikuje od ozadja v neposredni bližini oblačila. Zaradi tega algoritem ne dobi dovolj informacij in del v neposredni bližini kljub kontrastni barvi ostane. S pomočjo dodatne označitve ozadja uporabnik doseže željen rezultat.

Sliki 6.13 in 6.14 prikazujeta še dva primera odstranjevanja ozadja. Obe sliki sta narejeni na enakem ozadju. Pri sliki 6.13 je na robu obleke viden

Slika 6.13: Primer odstranitve ozadja svetlega objekta na temnem ozadju.

Slika 6.14: Primer odstranitve ozadja temnega objekta na temnem ozadju.

tanek temen rob, ki pa pri sliki 6.14 zaradi temne barve oblačila ni viden. Oba rezultata sta bila zadovoljiva že po prvi iteraciji in dodatno sodelovanje uporabnika ni bilo potrebno.

O vsakem oblačilu virtualna omara hrani še dva podatka. Prvi predstavlja opis oblačila. Pri iskanju se pregleduje te podatke. Drug podatek pa predstavlja kategorijo oblačila. Za shranitev vnosa je potrebno vpisati vse podatke. Slika 6.15 predstavlja pravilno izpolnjene podatke, ki so pripravljeni na shranitev v virtualno omaro.

Slika 6.15: Podatki pripravljeni za shranitev v virtualno omaro.

Poglavje 7

Zaključek

Diplomsko delo predstavlja opis prototipa aplikacije za shranjevanje oblačil v virtualno omaro in kreiranje stilov. Pri shranjevanju je možnih več načinov urejanj slike, kot so urejanje sivine, kontrasta, svetlosti in vrednosti game ter bistveni del, ki omogoča odstranjevanje ozadja.

Odstranjevanje ozadja je narejeno s pomočjo algoritma `GrabCut`. Delovanje je narejeno na način, da je potrebno čim manj sodelovanja uporabnika. Če je ozadje v kontrastu z oblačilom, v večini primerov zadostuje že začetna označitev objekta. Okoli objekta v prvem koraku narišemo pravokotnik, ki algoritmu posreduje informacijo, da je v njegovi okolici ozadje. V primeru, da prvotna odstranitev ni zadovoljiva, je možno ponoviti prvi korak ali pa dodatno ročno označiti del ozadja ali ospredja, ki je trenutno napačen. Iteracije se lahko ponavljajo, dokler uporabnik z rezultatom ni popolnoma zadovoljen.

Oblačilu se doda tudi opis, po katerem je potem možno iskati določena oblačila, kar pride zelo prav, ko enkrat virtualna omara vsebuje veliko oblačil. Zadnji del prototipa pa je sestavljen iz dejanskega sestavljanja in poimenovanja stilov, ki jih ustvari uporabnik z izbiro oblačil iz virtualne omare.

Ker je aplikacija samo prototip, je možnih še veliko izboljšav. Ena izmed njih bi bila že sama pohitritev algoritma `GrabCut`, ki sedaj ob 100% velikosti slike predstavlja za telefon kar precejšen zalogaj in potrebuje nekaj sekund. Pohitritev bi bilo možno narediti z dodatnim procesiranjem na grafični kar-

tici. Naslednja izboljšava bi bila bolj avtomatska obdelava barv v sliki, ki jih trenutno uporabnik popravlja ročno s pomočjo drsnika.

Obstaja pa tudi še veliko možnosti, ne samo za izboljšave, temveč tudi za razširitev aplikacije. Ko ima uporabnik enkrat shranjena oblačila v virtualni omari, bi lahko z ustvarjenimi stili naredil še statistiko nošenja oblačil. Shraniti bi bilo mogoče določen stil na določen dan, nato pa bi izpisalo, kdaj in kolikokrat je bil določen kos oblačila nošen. Lahko bi tudi predlagal, katerega oblačila že dolgo nismo nosili in pa, katera so naša najljubša oblačila. Možno bi bilo tudi dodati pošiljanje stilov po elektronski pošti ali pa objavljanje na trenutno priljubljenih socialnih omrežjih. Trenutno iskanje deluje samo na podlagi opisa oblačila. Dodati pa bi bilo možno tudi opcijo, da uporabnik pri iskanju nastavi določene filtre.

Še ena možna nadgradnja bi bila avtomatska klasifikacija oblačil, kjer bi aplikacija sama prepoznala nekatere meta podatke oblačila, kot so npr. v katero kategorijo spada oblačilo, kakšen je material oblačila ali prevladujočo barvo oblačila.

Literatura

[1] X. Bai, G. Sapiro, "A geodesic framework for fast interactive image and video segmentation and matting", *Proceedings of IEEE ICCV*, 2007.

[2] Yuri Y. Boykov, Marie-Pierre Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images", *International Conference on Computer Vision (ICCV), str. 105-112*, 2001.

[3] (2013) Brightness and Contrast Adjustments. Dostopno na: `http://www.dspguide.com/ch23/5.htm`.

[4] (2013) Color balance. Dostopno na: `http://en.wikipedia.org/wiki/Color_balance#cite_note-Viggiano2004-8`.

[5] (2013) Contrast Adjustment Explained. Dostopno na: `http://documentation.apple.com/en/color/usermanual/index.html#chapter=9%26section=3%26hash=apple_ref:doc:uid:Color-UserManual-90886PRI-1008090`.

[6] (2013) Gaussian Blur. Dostopno na: `http://docs.opencv.org/modules/imgproc/doc/filtering.html?highlight=gaussianblur#gaussianblur`.

[7] (2013) GrabCut. Dostopno na: `http://grabcut.weebly.com/background--algorithm.html`.

[8] (2013) Image Processing Algorithms: Brightness Adjustment. Dostopno na:
http://www.thecryptmag.com/Online/55/imgproc_4.html.

[9] (2013) Image Processing Algorithms: Contrast Adjustment. Dostopno na:
http://thecryptmag.com/Online/56/imgproc_5.html.

[10] (2013) Image Processing Algorithms: Gamma Correction. Dostopno na:
http://www.dfstudios.co.uk/articles/
image-processing-algorithms-part-6/.

[11] (2013) Implementing the GrabCut - Segmentation Technique as a Plugin for the GIMP. Dostopno na:
http://www.cs.ru.ac.za/research/g02m1682/.

[12] (2013) iOS. Dostopno na:
http://en.wikipedia.org/wiki/IOS.

[13] (2013) iOS Core Data. Dostopno na:
http://developer.apple.com/library/ios/#documentation/
DataManagement/Conceptual/iPhoneCoreData01.

[14] (2013) iTunes - Style Book App. Dostopno na:
https://itunes.apple.com/us/app/stylebook/id335709058?mt=8

[15] (2013) Measurements of video gamma, and the impact of gamma on light curves. Dostopno na:
http://www.astrogeeks.com/Bliss/OccultVideo/videogamma.html

[16] Eric N. Mortensen , William A. Barrett, "Interactive segmentation with intelligent scissors", *Graphical Models and Image Processing, str. 349-384*, 1998.

[17] (2013) OpenCV. Dostopno na:
`http://docs.opencv.org/modules/core/doc/intro.html`.

[18] C. Rother, V. Kolmogorov, A. Blake, "GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts", *Microsoft Research Cambridge, UK*, 2004.

[19] M. Ruzon, C. Tomasi, "Alpha estimation in natural images", *Proceedings of IEEE CVPR, str. 18–25*, 2000.

[20] (2013) Start Developing iOS Apps. Dostopno na:
`https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/GetToolsandInstall.html`.

[21] (2013) Style Book. Dostopno na:
`http://www.stylebookapp.com/`.

[22] Justin F. Talbot, Xiaoqian Xu, "Implementing GrabCut", *Brigham Young University*, 2006. Dostopno na:
`research.justintalbot.org/papers/Grabcut.pdf`.

[23] (2013) Threshold. Dostopno na:
`http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html`.

[24] Jue Wang, Michael F. Cohen, "Image and Video Matting: A Survey", *Foundations and Trends in Computer Graphics and Vision*, 2007.

[25] J. Wang, M. Agrawala, M. Cohen, "Soft scissors: an interactive tool for realtime high quality matting," *Proceedings of ACM SIGGRAPH*, 2007.

[26] (2013) White Balance. Dostopno na:
`http://www.cambridgeincolour.com/tutorials/white-balance.htm`.

[27] (2013) Working with RGBA colour. Dostopno na:
`http://24ways.org/2009/working-with-rgba-colour/`.

Slike