ŠOLSKI CENTER VELENJE

ELEKTRO IN RAČUNALNIŠKA ŠOLA

Trg mladosti 3, 3320 Velenje

MLADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

RAČUNALNIŠKI VID: PRETVORBA SKICE V DATOTEKO

Tematsko področje: tehniške vede

Avtorji:

Tomaž Čede, 4. letnik

Jon Rojnik Goršek, 4. letnik

Mentor:

Gregor Hrastnik, univ. dipl. inž. rač. in inf.

Velenje, 2021

Raziskovalna naloga je bila opravljena na Elektro in računalniški šoli Šolskega centra Velenje.

Mentor: Gregor Hrastnik, univ. dipl. inž. rač. in inf.

Datum predstavitve:

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD ŠC Velenje, šolsko leto 2020/2021

KG pretvarjanje / računalniški vid / skica / datoteka

AV Tomaž Čede / Jon Rojnik Goršek

SA HRASTNIK, Gregor

KZ 3320 Velenje, SLO, Trg mladosti 3

ZA Šolski center Velenje, Elektro in računalniška šola

LI 2021

IN RAČUNALNIŠKI VID: PRETVORBA SKICE V DATOTEKO

TD Raziskovalna naloga

OP VI, 38 str., 26 sl., 12 vir.

IJ SL

JI sl / en

AI Dandanes se umetnost vedno bolj prestavlja v digitalni format, precej umetnikov pa še vedno riše na roko. Prišla sva do ideje, da bi z mobilno aplikacijo pretvarjala skice, ki so narisane na papir, v digitalno obliko. Najina aplikacija omogoča pretvorbo skice, narisane na papir, v datoteko v obliko SVG. Aplikacija je narejena za Android s pomočjo razvojnega okolja Visual Studio z vtičnikom Xamarin, ki nam omogoča, da razvijamo mobilne aplikacije za Android in IOS. Pretvorbo v SVG sva naredila v s skripti programskem jeziku Python, ki se izvajajo na strežniku. V aplikaciji se lahko nastavljajo tudi parametri za pretvorbo. Mobilno aplikacijo sva testirala s pomočjo skic dveh umetnic in s pretvorjenimi slikami odgovorila na vprašanje, če je pretvorjena SVG datoteka primerljiva s skico, prerisano s pomočjo grafične tablice. Ugotovila sva, da aplikacija v trenutnem stanju še ni zmožna točno pretvoriti skice, a bi z optimiziranjem kode to v prihodnosti lahko dosegla.

KEY WORDS DOCUMENTATION

ND ŠCV, šolsko leto 2020/2021

CX converting / computer vision / sketch / file

AU Tomaž Čede / Jon Rojnik Goršek

AA HRASTNIK, Gregor

PP 3320 Velenje, SLO, Trg mladosti 3

PB Šolski center Velenje, Elektro in računalniška šola

PY 2021

TI COMPUTER VISION: CONVERTING A SKETCH TO A FILE

DT RESEARCH WORK

NO VI, 38 str., 26 sl., 12 vir.

LA SL

AL sl / en

AB Nowadays, art is increasingly being moved to digital format, and quite a few artists are still drawing by hand. We came up with the idea to use a mobile app to convert sketches drawn on paper into digital form. Our application allows you to convert a sketch drawn on paper into a SVG file. The app is made for Android using the Visual Studio development environment with the Xamarin plugin, which allows us to develop mobile apps for Android and iOS. We did the conversion to SVG with Python scripts running on our server. Conversion parameters can also be set in the application. We tested the mobile application with the help of sketches by two artists and used the converted images to answer the question of whether the converted SVG file is comparable to a sketch redrawn with the help of a graphics tablet. We found that the app in its current state is not yet able to accurately convert sketches, but by optimizing the code, it could achieve this in the future.

KAZALO VSEBINE

KAZALO SLIK

1. UVOD

Ljudje si dandanes ne predstavljamo sveta brez naprav. Dandanes vedno večji del našega življenja vključuje te naprave. To velja tudi za umetnike, ker tudi umetniki svoje slike in portrete objavljajo na različni straneh in forumih, kjer jih drugi umetniki lahko komentirajo. Preden sploh objavijo te slike, pa jih morajo nekako spraviti v digitalni format, s tem da jih slikajo ali pa jih narišejo v aplikacijo ter jih od tam delijo. Večina umetnikov že uporablja grafične tablice, s katerimi rišejo v svojo priljubljeno aplikacijo. Kaj pa, ko narišejo skico, ki jim je všeč, a je skica narisana na papir? To se je nama zgodilo že večkrat, zato sva si zadala, da rešiva ta problem. Po dolgem premisleku sva si zamislila aplikacijo, v katero bomo lahko naložili sliko skice in bo aplikacija vrnila našo skico v takšnem formatu, da lahko risanje nadaljujemo v priljubljeni mobilni ali namizni aplikaciji za risanje. Aplikacijo sva nato naredila v razvijalnem okolju Visual Studio z vtičnikom Xamarin ter ustvarila skripte v programskem jeziku Python, ki nama iz slike, naložene na spletni strežnik, ustvari SVG datoteko, katero lahko nato odpremo v poljubni aplikaciji ter nadaljujemo z risanjem.

1.1 Hipotezi

Zadala sva si dve hipotezi:

1. Mobilna aplikacija za pretvorbo skice v SVG datoteko izniči potrebo po prerisovanju skice v grafični program.

2. Pretvorjena SVG datoteka je primerljiva s skico, prerisano s pomočjo grafične tablice.

2. PREGLED STANJA TEHNIKE

Najprej sva raziskala, kako umetniki zdaj rešujejo ta problem. Odkrila sva, da trenutno umetniki svoje skice oz. risbe najprej skenirajo ter ta sken slike dajo v aplikacijo in rišejo čez njega. [1]

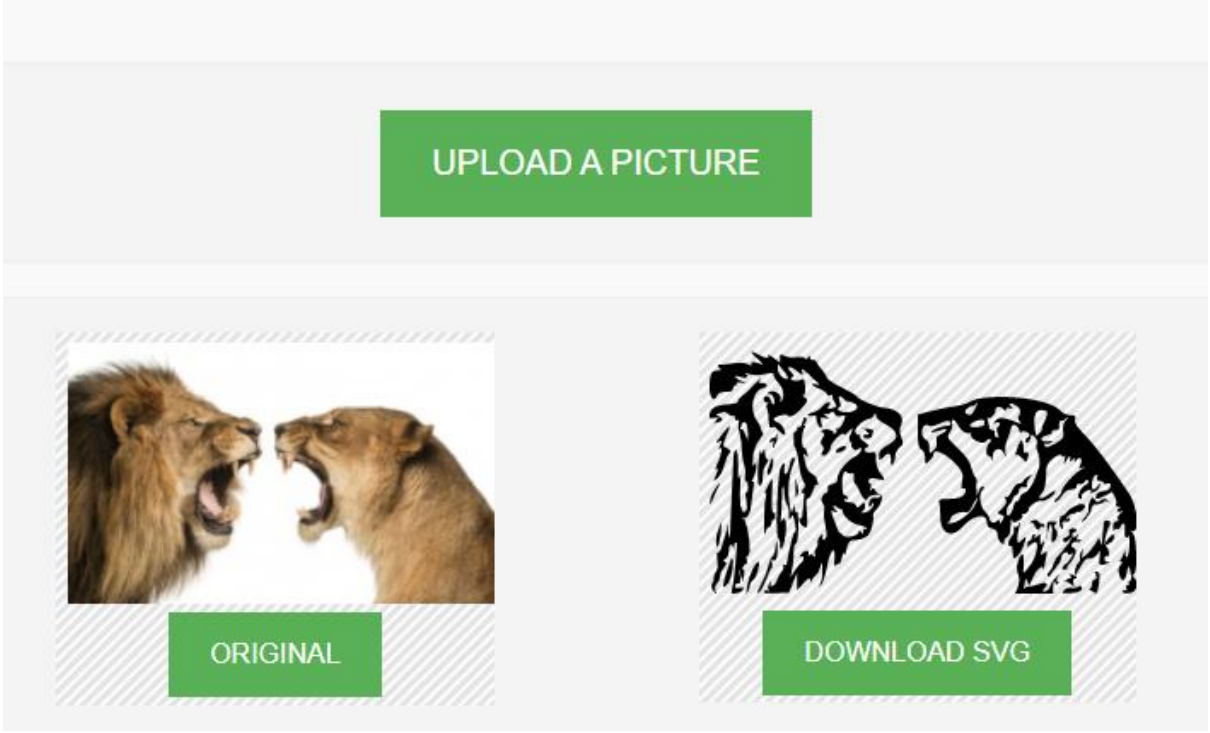
2.1 Podobne aplikacije

Ker je bila najina ideja, da narediva aplikacijo, sva nato pogledala, če obstajajo že kakšne podobne aplikacije oz. druge rešitve, ki vključujejo neko programsko kodo, ki bi odstranila potrebo po skeniranju slike.

Spletni pretvorniki

So računalniški programi, ki lahko spremenijo shranjevalno obliko datotek (npr.jpeg v SVG). Predvsem jih najdemo na spletnih straneh. Uporabljajo se za več različnih namenov npr. pretvarjanje datotek v format, ki ga lahko naša naprava podpira, shranjevanje videov iz socialnih omrežij (npr. youtube to mp4 - https://yt1s.com/youtube-to-mp4 , ki pretvori vneseno youtube povezavo v ustrezno mp4 datoteko), pretvorbo gif datotek v videe ….

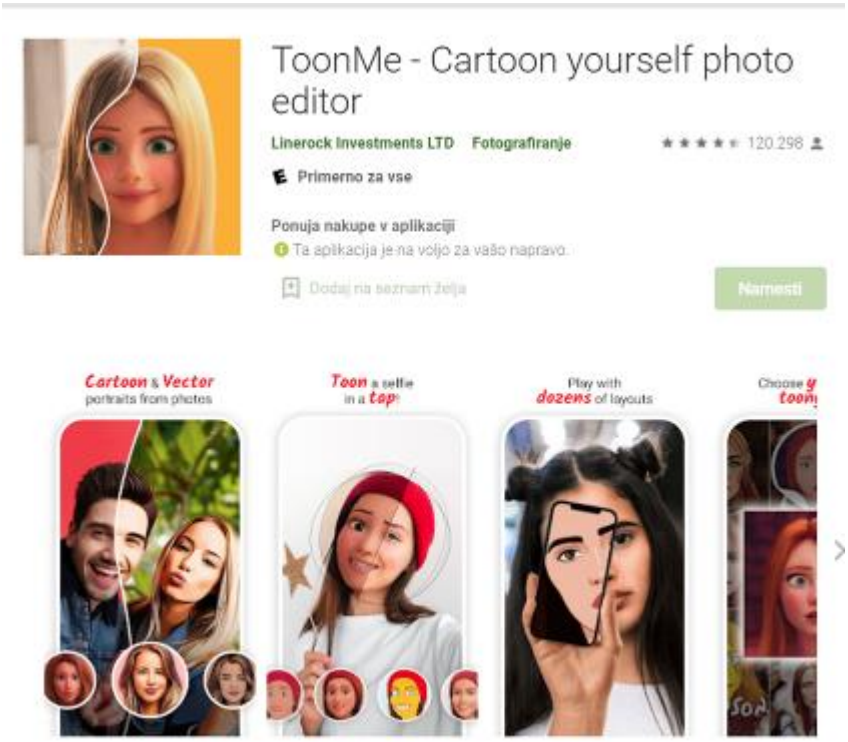
Obstajajo tudi online pretvorniki, ki pretvarjajo slike v SVG datoteke, podobno kot pri najini ideji, s tem da so te aplikacije spletne za razliko od najine, ki je pa mobilna aplikacija.



Slika 1 : Online pretvornik za SVG datoteke

Vir: pclSVG.com

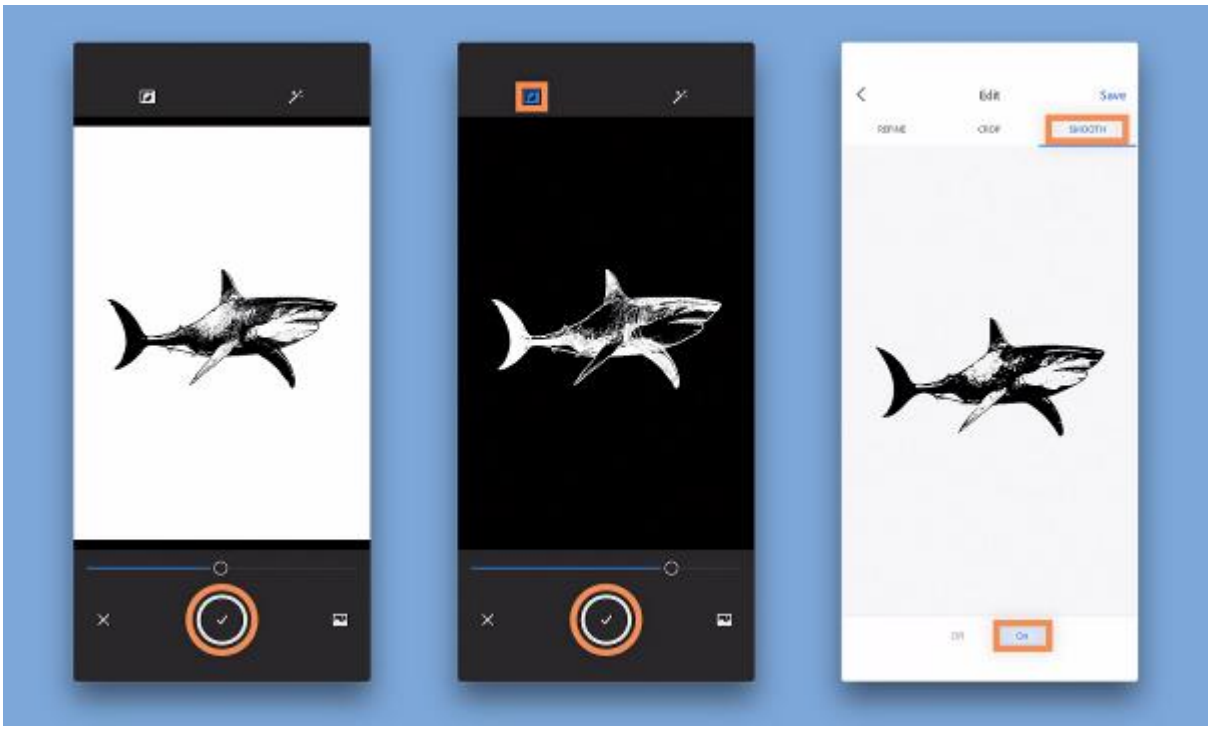
Medtem ko sva raziskovala to področje, sva si ogledala tudi, katere mobilne aplikacije že obstajajo. Na Android telefonih obstaja nekaj aplikacij, ki delujejo podobno kot online SVG pretvorniki, a nobena ni posebej namenjena pretvarjanju skic v digitalni format, kar poskušava doseči midva. Obstajajo aplikacije kot npr. ToonMe, ki omogočajo uporabnikom, da iz slike, posnete na svojem telefonu, naredijo digitalno risbo, a to naredijo samo z različnimi filtri za razliko od najine aplikacije, ki poleg filtrov v sliki najde črte ter jih nariše v novo SVG datoteko. [2]



Slika 2 : Aplikacija ToonMe

Vir: https://play.google.com/store/apps/details?id=com.vicman.toonmeapp&hl=sl&gl=US

Obstaja tudi aplikacija podjetja Adobe, imenovana Adobe Capture. Lahko jo uporabljamo kot pretvornik slik v vektorske formate (SVG,EPS ...) in si jih podobno kot v najini aplikaciji pošljemo na računalnik ali pa uporabimo v ostalih programih podjetja Adobe, ki jih imamo na telefonu. Za razliko od najine aplikacije pa lahko s Adobe Capture prepoznamo še različne vzorce, materiale in jih uporabimo v 3D - objektih, lahko prepoznamo tudi font pisave, poiščemo barve in ustvarimo svoje čopiče za uporabo v drugih ilustratorskih programih. [3]



Slika 3 : Aplikacija Adobe Capture

Vir: https://helpx.adobe.com/si/mobile-apps/how-to/capture-grab-colors-themes-shapes.html

2.2 Računalniški vid

Ker najina aplikacija za prepoznavanje črt na sliki uporablja algoritme računalniškega vida, sva raziskala tudi to področje.

Računalniški vid je področje računalniške znanosti, ki omogoča računalniku prepoznavanje in procesiranje objektov v slikah in videih na podoben način kot ljudje. To mu omogoča s pomočjo senzorjev, računalnikov in algoritmov računalniškega učenja.

Na nek način se računalniški vid vrti okoli prepoznavanja vzorcev. Zato je en način učenja računalnika o razumevanju vizualnih podatkov tak, da mu dodeljujemo po več tisoč označenih slik, katere kasneje izpostavimo različnim programom oz. algoritmom, ki omogočajo računalniku iskanje vzorcev v vseh elementih, ki se nanašajo na oznake. Npr. če bi računalniku podali tisoč slik psov, bi te slike analiziral in ugotovil vse barve, oblike, razmike med oblikami, itd. in iz tega ugotovil, kaj oznaka “pes” pomeni.

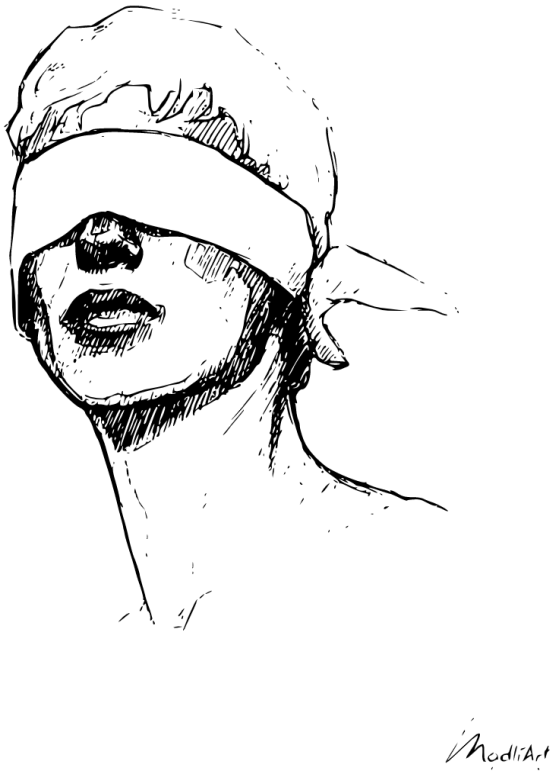
Primer programske opreme računalniškega vida je Tensor flow. To je Googlova odprto kodna knjižnica za številčne izračune in obsežno računalniško učenje, ki uporablja Python kot priročen front-end api za grajenje aplikacij. Ta ima trenutno narejeno največjo bazo prepoznavanja, namenjeno vsakdanjim uporabnikom. [4]

Dandanes je računalniški vid uporabljen v raznih področjih, kot je prepoznava okolja samovozečih avtomobilov, prepoznavanje obraza, avtomatizacija medicine, itd.

2.3 SVG datoteke

Naslednji korak je bil, da raziščeva tudi, kako delujejo SVG datoteke, saj sva se odločila za ta format datoteke.

SVG je “eXtensible Markup Language (XML) “ - vektorski grafični format za splet in druga okolja. Uporablja oznake podobne kot html, vendar je strožji.



Slika 4 : SVG datoteka

Foto: Jon Rojnik Goršek

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
 width="735.000000pt" height="902.000000pt" viewBox="0 0 735.000000 902.000000"
 preserveAspectRatio="xMidYMid meet">

<g transform="translate(0.000000,902.000000) scale(0.100000,-0.100000)"
fill="#000000" stroke="none">
<path d="M2940 7657 c-140 -73 -253 -135 -288 -159 -23 -16 -46 -27 -51 -24
-4 3 -50 0 -101 -6 -76 -9 -94 -8 -101 3 -7 11 -45 -22 -154 -134 -80 -82
-145 -143 -145 -136 0 7 8 21 18 32 9 11 12 17 5 14 -6 -4 -35 -42 -64 -85
-29 -42 -55 -79 -58 -82 -11 -9 -91 -175 -91 -188 0 -7 -4 -11 -9 -8 -5 3 -10
-46 -12 -112 -4 -117 -4 -118 -42 -174 -21 -32 -42 -60 -48 -63 -5 -4 -7 -15
-4 -24 3 -10 -4 -30 -17 -47 -12 -16 -18 -26 -13 -22 4 4 21 -31 36 -78 16
-46 36 -92 44 -101 9 -9 14 -24 12 -34 -3 -11 -1 -22 4 -25 5 -3 9 -16 9 -29
0 -13 5 -27 12 -31 8 -5 9 2 5 24 -5 25 -3 30 9 25 12 -4 14 -15 9 -47 -4 -22
-10 -45 -13 -51 -4 -5 -7 -14 -8 -20 -1 -5 -14 -54 -28 -109 -29 -111 -29
-109 -17 -102 9 6 7 -115 -3 -161 -4 -16 -8 -109 -11 -208 -2 -98 -9 -200 -16
-225 -8 -32 -9 -64 -2 -110 110 -65 2 82 c0 46 5 113 11 150 5 38 16 129 23
```

Slika 5 : Koda SVG datoteke

Foto Jon Rojnik Goršek

Koda v SVG datoteki je samo besedilo oz. tekst, ki opisuje sestavne dele slike, tj. črte, oblike, krivulje … Lahko ga uredimo v urejevalniku besedila in s tem spremenimo prikazano sliko. Vse vrste animacij in interakcij lahko dodamo preko CSS ali JavaScript. [5]

Zakaj bi uporabili SVG?

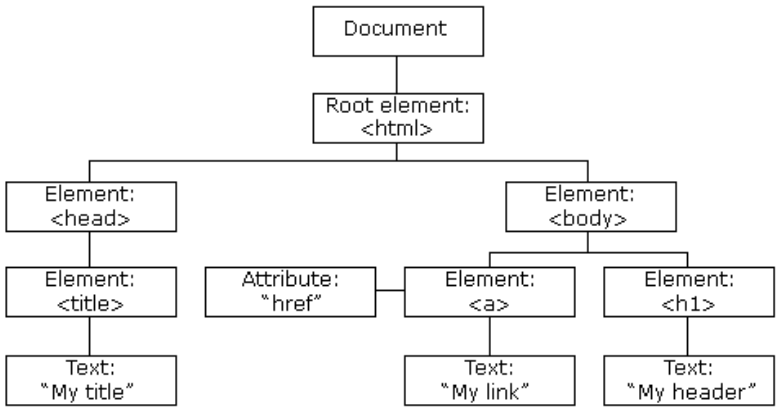
Ločljivost je neodvisna in odzivna, zaradi vektorske narave SVG datotek, je ločljivost neodvisna, to pomeni da lahko slika dobro izgleda na katerem koli zaslonu. Pri uporabi SVG datotek tako ni več potrebe po več različicah datotek (npr. ikon, slik) za različne velikosti zaslonov npr. pri spletnih straneh ali mobilnih aplikacijah. Elementi znotraj SVG so lahko tudi animirani, da ustvarimo zanimive interaktivne izkušnje.

Majhne velikosti datotek - V primerjavi s drugimi tipi datotek (png, jpeg ...) so SVG datoteke veliko manjše velikosti, še posebej če so pravilno optimizirane (za optimizacijo obstaja več rešitev, vse od različnih ukazov do ročnega odstranjevanja točk in skupin.

Je stilsko sposoben - z uporabo razredov (ang. class) in oznak (angl. id) lahko urejamo elemente znotraj SVG datotek.

Je interaktiven - z uporabo JavaScripta (objekten skriptni programski jezik, ki pomaga pri ustvarjanju interaktivnih spletnih strani), HTML in CSS lahko oblikujemo elemente znotraj SVG datotek.

Ima svoj vodljiv DOM (Document Object Model - z njim lahko JavaScript dostopa do vseh elementov HTML dokumenta in jih tudi spremeni), omogoča nam, da je lahko naša SVG datoteka katerekoli velikosti, v brskalniku pa jo prikažemo v drugačni, ne da bi spremenili lastnosti v SVG. [6]



Slika 6 : Primer DOM

Vir : https://www.w3schools.com/js/js_htmldom.asp

2.4 Python in računalniški vid

Za uporabo filtrov ter pretvarjanje slike v SVG format uporabljava Python ter nekatere Python knjižnice, zato sva raziskala tudi ta programski jezik.

Python je večnamenski programski jezik, ustvarjen 1990. Ko je bil ustvarjen, so predvsem gledali na to, da je koda pregledna, kar so dosegli z uporabo presledkov. Python je objektno orientiran programski jezik, kar pomeni da uporablja tako imenovane "objekte", to so spremenljivke posebnih podatkovnih tipov, ki jih lahko uporabnik sam sestavi in vsebujejo različne podatke ter lahko izvajajo tudi svoje funkcije, imenovane metode. Prva verzija Pythona je prišla ven leta 1991 in je bila naslednik programskega jezika ABC. Naslednja verzija, Python 2.0, je prišla ven leta 2000 z veliko novih sposobnosti kot npr. dodan je bil Garbage Collection sistem, kar omogoča Python, da samodejno ravna in ureja spomin na računalniku. Najnovejša verzija je Python 3, ki pa ni čisto združljiv s prejšnjimi verzijami, zato moramo staro kodo večinoma preurejati, da deluje na novejših verzijah.

Python sam nima veliko možnosti, a ko dodamo knjižnice (oz. module), postane veliko bolj funkcionalen in lahko z njim ustvarjamo bolj kompleksne programe, kot so A.I. (Artificial intelligence) ali uporabljamo računalniški vid (Computer Vision). Najbolj pogosto uporabljene knjižnice so:

Math, ki omogoča dodajanje različnih matematičnih operacij

Random. ki generira naključne vrednosti spremenljivkam

Datetime, ki omogoča dobiti točen datum ter čas v izbranem časovnem pasu

Midva sva Python uporabila, da sva naredila programe, ki na sliko dodajo filtre ter potem, ko so filtri prilagojeni ustvari potem tudi pretvori celo sliko v SVG datoteko, katero si potem lahko uporabnik shrani ter jo naprej uporablja v poljubnem programu. [7]

2.5 Prepoznavanje s pomočjo filtrov

Filtri so matematične funkcije, ki se izvajajo nad sliko. V računalništvu je slika 2-dimenzionalna tabela, ki vsebuje vrednosti barv, navadno RGB (odvisno od formata slike in modula za pretvorbo). Vse kar filtri počnejo, je spreminjanje vrednosti 2D-tabele po določenih navodilih.

V najini aplikaciji sva uporabila naslednje filtre:

Gaussov filter

Gaussov filter se uporablja za megljenje slik in odstranitev šumov in detajlov iz slik.

Filter se premika čez vsako točko slike (angl. pixel) posamično in jo izbere kot jedro, okoli nje izbere skupino točk, imenovano kernel (velikost in širina morata biti liho število, zato da je jedro vedno v sredini). Za izvedo filtra na trenutno izbrani točki, se izračuna uteženo povprečje RGB vrednosti vseh točk v kernelu. To pomeni, da se čez kernel točk postavi Gausov kernel ter se njune vrednosti pomnožijo.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Slika 7 : Primer Gausovega kernela

Vir: https://computergraphics.stackexchange.com/questions/39/how-is-gaussian-blur-implemented

Posebnost pri Gaussovem filtriranju je, da imajo točke bližje jedra večjo vrednost (težo), kot pa tiste, ki so od jedra bolj oddaljene. Na koncu se povprečna vrednost uteženih točk (pomnoženih s Gausovim kernelom) shrani v jedro. Torej večji kot nastavimo "radius" megljenja, bolj zamegljena bo končna slika. [8]

Gaussov filter uporablja Gaussovo funkcijo za izračun transformacije, ki je mora opraviti nad posamezno točko v sliki. [9]

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

Slika 8 : Gaussova funkcija

Vir : https://www.geeksforgeeks.org/apply-a-gauss-filter-to-an-image-with-python/

Slika 9 : slika po Gaussovem filtriranju

Foto: Jon Rojnik Goršek

Sivinski filter

Z uporabo sivinskega filtra (ang. grayscale) sva barvno sliko pretvorila v sivinsko, katere edine barve so različni odtenki sive. To nam koristi zato, ker je zaradi tega potrebno manj informacij za vsako točko slike. To pomeni, da moramo za vsako točko določiti samo eno vrednost od 0 (črno) pa do 255 (belo), namesto da določimo vrednosti rdeče, zelene in modre komponente (RGB), saj imajo pri sivinskih slikah te tri barve isto vrednost. [10]

Slika 10 : Primer slik po sivinskem filtru

Foto: Jon Rojnik Goršek

Threshold

Po uporabi sivinskega filtra uporabimo threshold oz. mejno vrednost barve. To je eden izmed najenostavnejših načinov segmentiranja slike in z uporabo le-tega lahko pretvorimo sivinske slike v binarne (vrednost samo 0 - črna ali 255 - bela).

Deluje tako, da najprej določimo vrednost barve, filter pa potem spremeni vse točke, ki imajo manjšo vrednost, v črne (RGB vrednost barve 0) in vse točke, ki imajo večjo vrednost v bele (RGB vrednost barve 255).

Slika 11 : Slika po obdelavi s filtrom threshold

Foto: Jon Rojnik Goršek

Iskanje robov

S filtrom iskanje robov (ang. Find edges) zaženemo jedro za zaznavanje robov. Ta pa vrne sliko z visokimi spremembami intenzivnosti, robovi so v odtenkih bele, ostali deli slike pa so pobarvani črno.

2.6 Vmesnik API

Vmesnik API (angl. Application Programming Interface) je aplikacijski programski vmesnik, ki omogoča sodelovanje dveh različnih aplikacij med seboj. Primer vmesnika API v vsakdanjem življenju lahko vidimo pri socialnih omrežjih pa tudi že pri sporočilih. Takoj ko se aplikacija poveže na internet in želimo preko nje pošiljati podatke, je velika možnost, da je v ozadju API, ki skrbi, da se na strani strežnika izvedejo vsi potrebni ukazi oz. naloge ter vrne oz. pošilja potrebne datoteke nazaj. Vmesnik API tudi vključuje sloj varnosti med napravo in strežnikom, saj naprava in strežnik nikoli nista neposredno povezana med seboj, ampak komunicirata med sabo z zelo malo podatki in bi bilo zelo težko poškodovati uporabnikovo napravo oz. strežnik. [11]

3. METODOLOGIJA

3.1 Izdelava aplikacije

Izdelala sva mobilno aplikacijo, s katero lahko skico narisano na papir, pretvorimo v SVG datoteko, ki jo lahko uporabnik odpre v poljubnem programu za risanje slik. Preden uporabnik ustvari svojo datoteko, lahko v aplikaciji tudi nastavi intenzivnost filtrov, dokler mu nastala skica ni všeč ter jo nato lahko ustvari in odpre v lokalni aplikaciji na mobilnem telefonu ali pa si jo pošlje na elektronsko pošto ter jo nato prenese na računalnik, kjer lahko naprej ureja narejeno sliko. Za dodajanje filtrov ter izdelovanje SVG datotek sva naredila programe v programskem jeziku Python, ti programi pa se izvajajo na strežniku, ki nama nato vrne narejeno datoteko.

Uporabila sva Python knjižnice Numpy, PIL, os, sys ter SVGwrite.

Knjižnico PIL sva uporabila za odpiranje slik ter za lažjo uporabo filtrov, saj omogoča, da na sliko damo filtre brez kompliciranih računov.

Knjižnico Numpy sva uporabila, da iz slike, ki sva jo že obdelala s filtri, narediva dvodimenzionalno polje, iz katerega črpava podatke za ustvarjanja SVG datoteke.

Knjižnico SVG write sva uporabila za risanje črt v novo datoteko.

Knjižnico os sva uporabila za brisanje slik iz serverja.

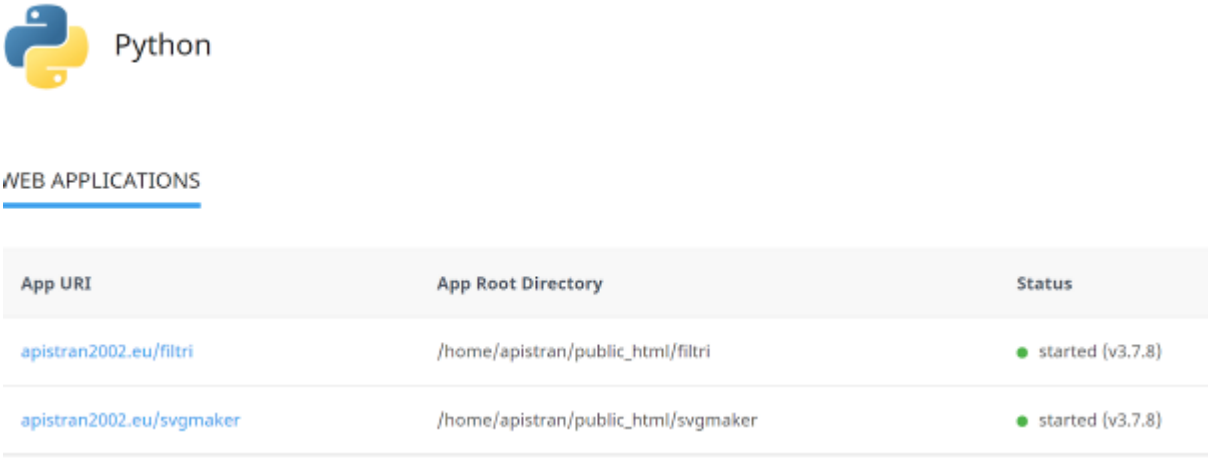
Knjižnico sys sva uporabila za pridobivanje podatkov iz najinega vmesnika API.

Aplikacijo sva začela naprej razvijati v programskem jeziku Python z modulom Kivy, ki omogoča, da razvijemo aplikacije za različne platforme, od mobilnih naprav do namiznih računalnikov. Slabost tega načina razvoja je, da bi potrebovala operacijski sistem Linux, da bi aplikacijo lahko naložila na Android telefon.

Tako sva se odločila, da bova aplikacijo razvila v orodju Xamarin, saj sva v tem okolju že delala, Python programe oz. skripte, ki sva jih napisala, pa bi izvajala preko spletnega strežnika in bi podatke pošiljala nazaj na telefon preko vmesnika API.

Xamarin je razširitev za popularno orodje za razvijanje programske opreme Microsoft Visual Studio, ki omogoča uporabnikom, da razvijajo aplikacije za mobilne operacijske sisteme Android, iOS in UWP ter omogoča, da za logiko aplikacije uporabljamo programski jezik C#. [12]

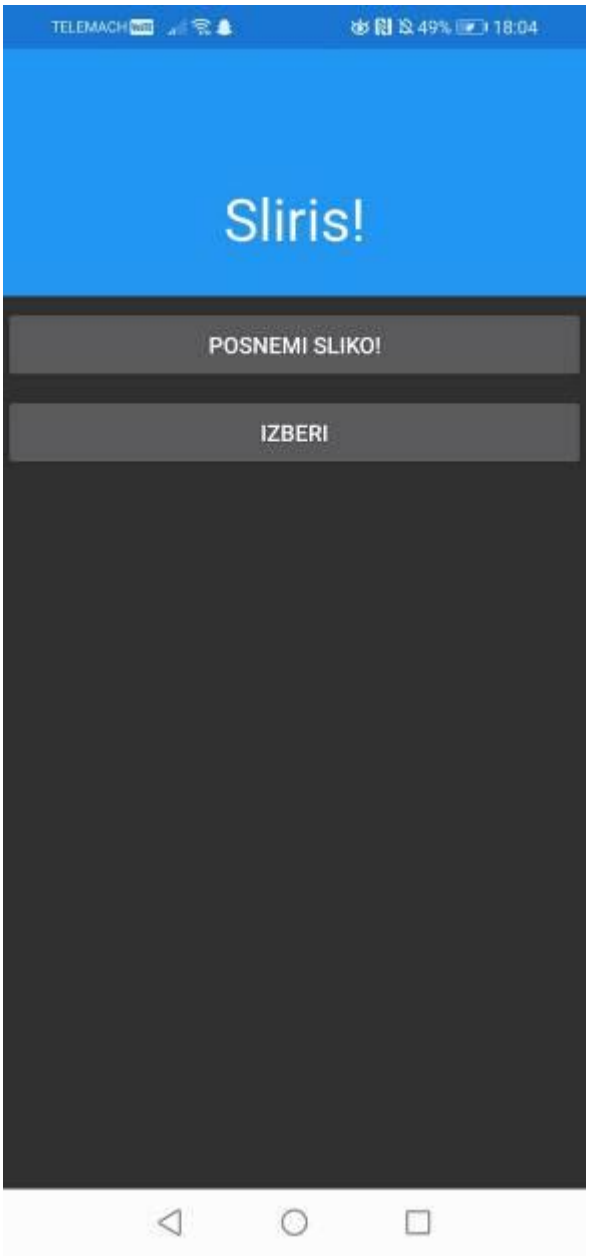
Vmesnik API sva uporabila za izboljšavo delovanja aplikacije in tudi za lažji dostop do podatkov kadarkoli. Deluje tako, da pošlje izbrano sliko in vrednosti filtrov na strežnik, ki ga imava najetega pri podjetju neoserv. Tam pa s temi podatki preuredi sliko in jo pošlje nazaj.



Slika 12 : Python aplikacije na strežniku

Foto Tomaž Čede

Najprej sva naredila osnovni uporabniški vmesnik mobilne aplikacije ter prestavila skripte na lokalni strežnik, da bi lahko tudi testirala delovanje. Nekaj težav sva imela z izvajanjem skriptov na strežniku, a sva to hitro rešila, tako da je aplikacija delovala. Nato sva prestavila vse datoteke iz lokalnega strežnika na spletnega. Na začetku aplikacija ni delovala, saj se skripti niso hoteli izvajati, zato sva morala na strežniku v nadzorni plošči cpanel uporabiti funkcijo “Setup Python App”, ki omogoča izvajanje Python skriptov na strežniku. Aplikacija zdaj omogoča, da izberemo sliko iz galerije (Slika 14) ali sliko sami posnamemo preko aplikacije (Slika 15), ji prilagodimo filtre ter ustvarimo SVG datoteko, ki si jo lahko pošljemo na elektronsko pošto ali pa jo odpremo v brskalniku (Slika 17).

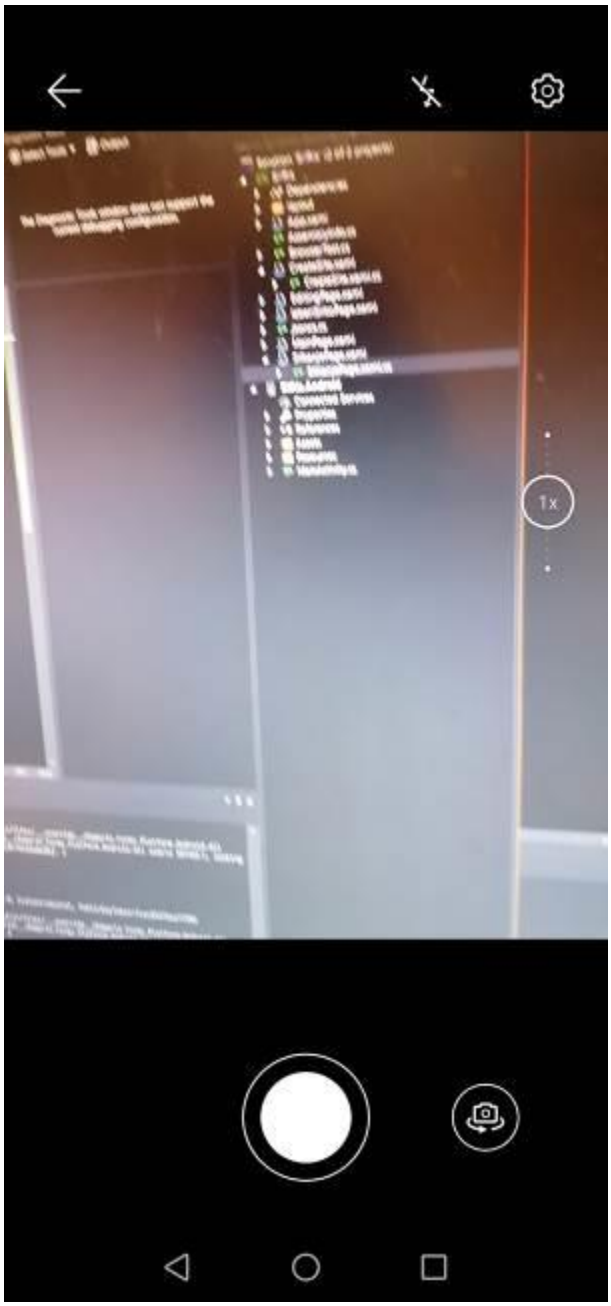


Slika 13 : Prva stran aplikacije

Foto: Tomaž Čede

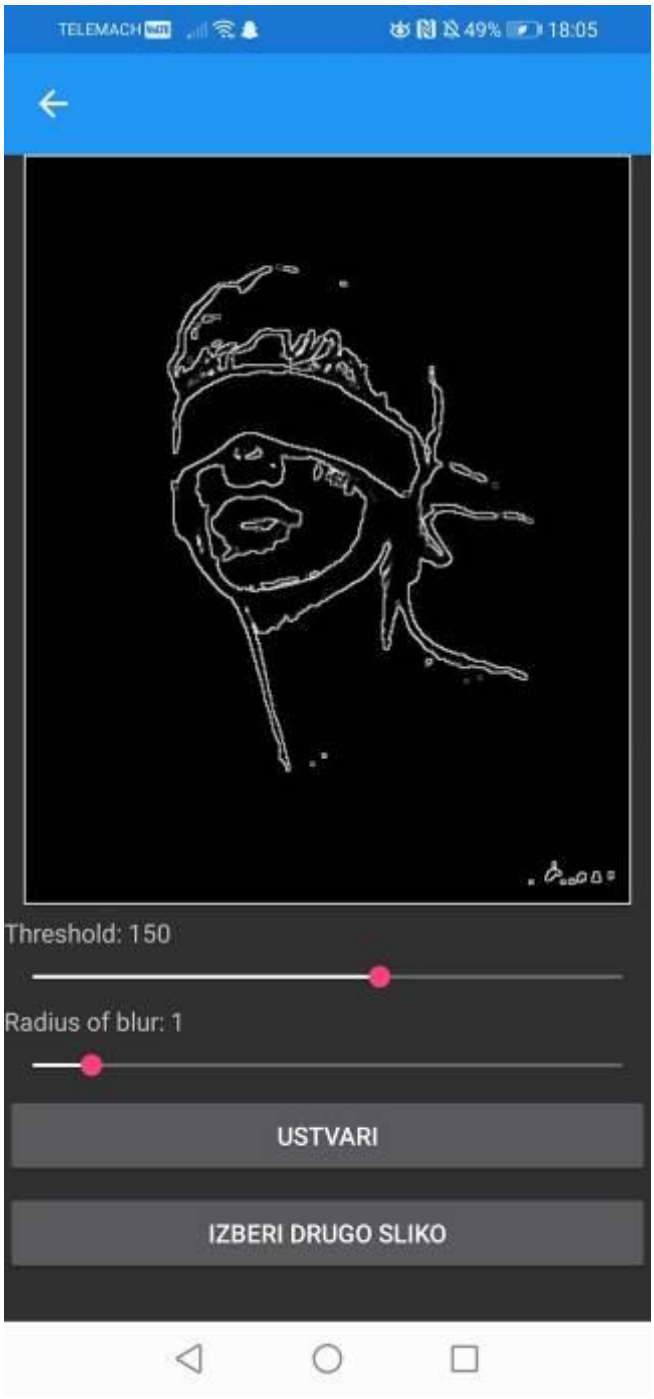
Slika 14 : Druga stran aplikacije (izbor slike)

Foto: Tomaž Čede



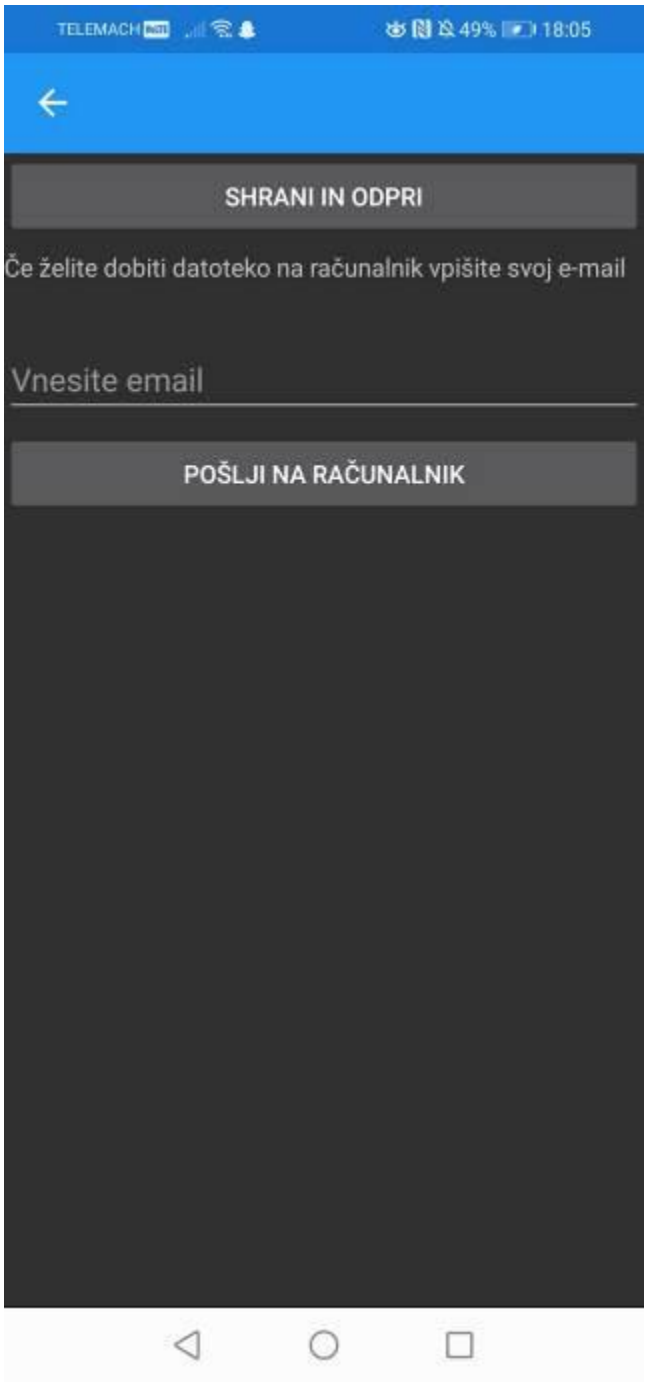
Slika 15 : Druga stran aplikacije (slikanje s kamero)

Foto: Tomaž Čede



Slika 16 : Tretja stran aplikacije (prilagajanje filtrov)

Foto: Tomaž Čede



Slika 17 : Četrta stran aplikacije (izdelava SVG datoteke)

Foto: Tomaž Čede

3.2 Testiranje pretvorbe skic s pomočjo aplikacije

Za skice, ki sva jih uporabila, da potrdiva oz. zavrneva hipoteze, sva prosila umetnici Zalo Klepac Keržan in Nežo Špelo Zupančič, ki sta nama poslali skice, narisane na papir ter prerisane v umetniški program. Skice, ki sva jih prejela, sva potem uporabila v aplikaciji in jim tudi prilagodila filtre ter na koncu ustvarila še SVG datoteke. Zala je za prerisovanje uporabila grafično tablico, priklopljeno na računalnik, Neža pa je uporabila tablični računalnik, na katerega riše s pisalom za grafične tablice.

Digitalno prerisano skico bova vizualno primerjala z nastalo SVG datoteko ter bova tako ugotovila, če je najin program zmožen ustvarjati skice, ki so primerljive s temi, ki so narisane na roko.

Skice:

Slika 18 : Skica 1 (na papirju)

Skica: Zala Klepac Keržan

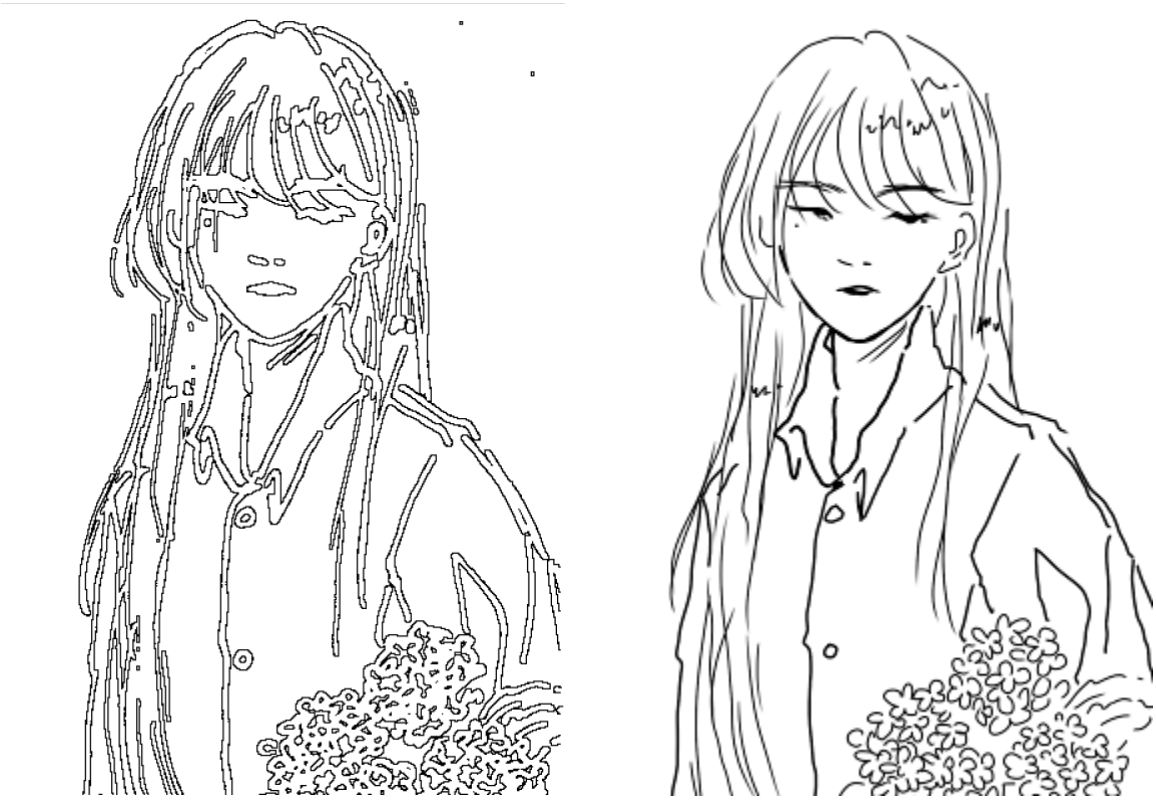
Slika 19 : Skica 2 (na papirju)

Skica: Zala Klepac Keržan

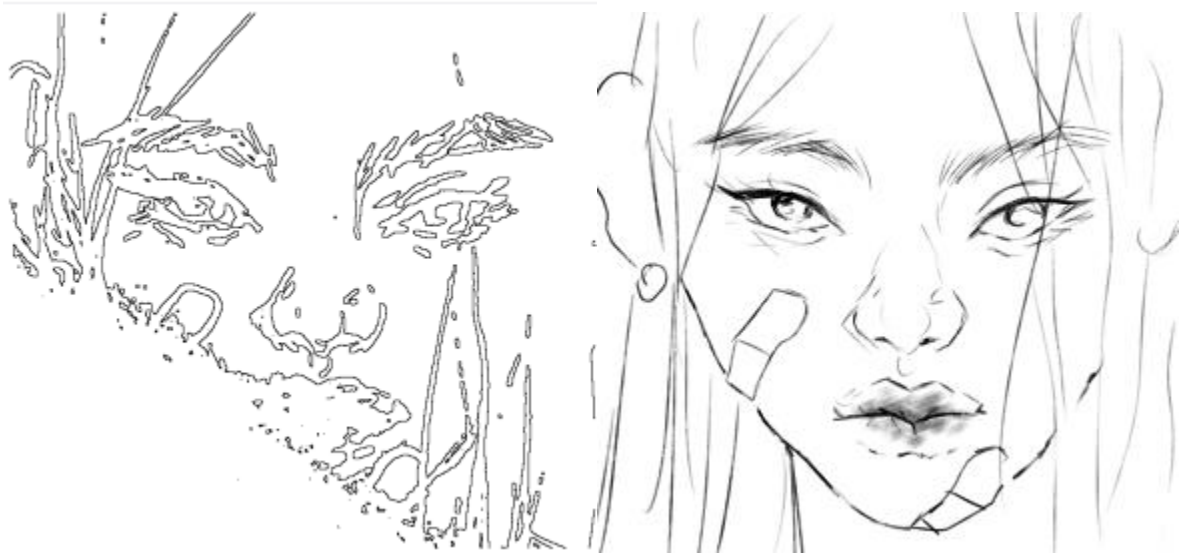
Slika 20 : Skica 3 (na papirju)

Skica: Neža Špela Zupančič

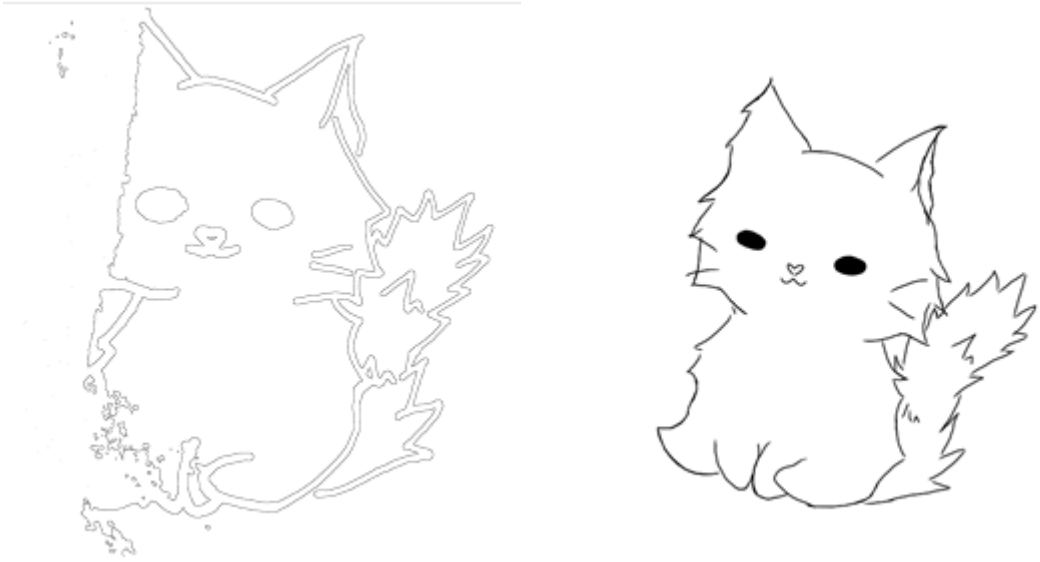
Pretvorjene v SVG datoteke:



Slika 21 : Prerisana skica 1 (levo SVG datoteka, desno prerisana na roko)



Slika 22 : Prerisana skica 2 (levo SVG datoteka, desno prerisana na roko)



Slika 23 : Prerisana skica 3 (levo SVG datoteka, desno prerisana na roko)

4. REZULTATI IN RAZPRAVA

4.1 Izdelana aplikacija

Z aplikacijo sva precej zadovoljna, saj dosega osnovni cilj, ki sva si ga zadala, da nama uspe spremeniti skico v digitalno obliko, sicer pa ima aplikacija še nekaj napak (angl. bug), ki jih še nisva razrešila. Največji napaki sta povezani z velikostjo slike in sencami na sliki. Če je slika večja od 1 MB, se aplikacija ustavi. Če so na sliki sence, to bistveno vpliva na končno SVG datoteko. Sicer sva dodala vse funkcionalnosti, ki sva si na začetku zadala, kar pa bi lahko nadgradila, je ustvarjanje SVG datoteke, da bi bila bolj podobna začetni sliki.

Prednosti najine aplikacije so, da je enostavna za uporabo, je hitrejša kot prerisovanje, saj SVG datoteko naredi v nekaj sekundah (odvisno sicer od velikosti slike) in naredi skico, kot je na papirju.

Slabosti aplikacije so kvaliteta nastale SVG datoteke, saj črte iz slike samo obrobi in jih ne preriše dobesedno, ter da ima aplikacija nekaj že prek omenjenih napak, ki bi jih morala bolj intenzivno raziskati, da bi jih popravila.

4.2 Rezultati testiranja

Z rezultati na dobljenih skicah nisva preveč zadovoljna, saj aplikacija še vedno ni v stanju, da bi skice dobesedno prerisala, ampak vse črte samo obrobi. Zaradi tega nastale SVG datoteke tudi ne izgledajo preveč dobro.

Skica 1

S pretvorbo prve skice sva najbolj zadovoljna, saj je tudi najbolj podobna originalni sliki. Še vedno pa se vidi v zgornjem desnem kotu nekaj pobarvanih pikslov, saj je tam bila senca, in čeprav sva lahko večino popravila s prilagoditvijo filtrov, je v SVG datoteki še vedno vidno, kje se je ta senca nahajala.

Slika 24 : Napake pri pretvorbi prve skice

Foto: Tomaž Čede

Skica 2

Pri pretvorbi druge skice je bilo kar nekaj napak. Sliko je pokrivala zelo rahla senca, a že to je bilo dovolj, da je spodnji levi del skice nastal razpršen in nedokončan. To napako sva poskušala rešiti s prilagoditvijo filtrov, a je to najboljši rezultat, ki sva ga lahko dobila.

Slika 25 : Napake pri pretvorbi druge skice

Foto: Tomaž Čede

Skica 3

Ta skica je imela podobne napake kot druga. Levi del slike je prekrivala senca, zato je del nastale digitalne skice ostal neprerisan. Prilagoditve filtrov tudi v tem primeru niso pomagale.

Slika 26 : Napake pri pretvorbi tretje skice

Foto: Tomaž Čede

Ugotovila sva, da je največja slabost pri najini aplikaciji, da ne pretvarja skice pravilno, če je na sliki prevelika senca. Tudi manjše sence že vplivajo na kvaliteto pretvorbe datoteke.

4.3 Hipoteze

Prvo hipotezo, ki pravi, da mobilna aplikacija za pretvorbo skice v SVG datoteko izniči potrebo po prerisovanju skice v grafični program, lahko delno potrdiva. Skica, ki nastane z uporabo najine aplikacije, lahko deluje kot dovolj dobra začetna točka risbe in kot solidna pomoč pri nadaljevanju risbe. Sama po sebi pa skica definitivno ni zadovoljiva. Če bi hoteli, da je pretvorjena skica tudi končni izdelek, bi jo bilo veliko bolje lastnoročno prerisati.

Drugo hipotezo, ki pravi, da je pretvorjena SVG datoteka primerljiva s skico, prerisano s pomočjo grafične tablice, lahko zavrneva, saj se končni rezultat tudi po popravkih aplikacije ne more primerjati z dejansko prerisano skico. V najini datoteki so vse črte samo obrobljene, kar jo že samo po sebi naredi drugačno, kot pa če bi jo sami prerisali, poleg tega pa se občasno zgodijo še napake (še posebej pri sencah), ki na koncu vseeno naredijo skico, prerisano s pomočjo grafične tablice, lepšo in boljšo.

5. ZAKLJUČEK

Z izdelovanjem aplikacije sva se več naučila o izdelovanju Python programov ter o računalniškem vidu. Prav tako sva se naučila več o delovanju vmesnikov API ter kako se izvajajo Python skripti na strežniku. Uporabila ter nadgradila sva tudi svoje znanje programskega jezika C# ter vtičnika Xamarin.

Delo s Pythonom nama na začetku ni bilo všeč, saj nisva vedela, kako deluje, a sva se sproti učila ter brala dokumentacijo in sva lahko na koncu izdelala skripte, ki se zdaj izvajajo na strežniku.

Aplikacijo bi lahko nadgradila s tem, da bi bolj optimizirala Python skripte ter nadgradila način, kako program preriše skico, da bi izgledala bolj čista ter bolj podobna skici, ki jo prerisuje. Lahko bi tudi nadgradila izgled aplikacije ter pospešila njeno delovanje s tem, da bi bolj optimizirala skripte na strežniku ter kodo same aplikacije.

Meniva, da bi lahko bila optimizirana verzija aplikacije v pomoč umetnikom ter jim olajšala delo in skrajšala čas, ki ga vložijo v to, da spravijo sliko v digitalni format.

6. POVZETEK

Dandanes se umetnost vedno bolj prestavlja v digitalni format, precej umetnikov pa še vedno riše na roko. Prišla sva do ideje, da bi z mobilno aplikacijo pretvarjala skice, ki so narisane na papir, v digitalno obliko. Najina aplikacija omogoča pretvorbo skice, narisane na papir, v datoteko v obliko SVG. Aplikacija je narejena za Android s pomočjo razvojnega okolja Visual Studio z vtičnikom Xamarin, ki nam omogoča, da razvijamo mobilne aplikacije za Android in IOS. Pretvorbo v SVG sva naredila s skripti v programskem jeziku Python, ki se izvajajo na strežniku. V aplikaciji se lahko nastavljajo tudi parametri za pretvorbo. Mobilno aplikacijo sva testirala s pomočjo skic dveh umetnic in s pretvorjenimi slikami odgovorila na vprašanje, če je pretvorjena SVG datoteka primerljiva s skico, prerisano s pomočjo grafične tablice. Ugotovila sva, da aplikacija v trenutnem stanju še ni zmožna točno pretvoriti skice, a bi z optimiziranjem kode to v prihodnosti lahko dosegla.

7. ZAHVALE

Zahvaljujeva se najinemu mentorju Gregorju Hrastniku za pomoč in podporo pri raziskovalni nalogi. Zahvaljujeva se tudi Neži Špeli Zupančič in Zali Klepac Keržan za skice, uporabljene pri testiranju in tudi dr. Nataši Meh Peer za lektoriranje.

Viri in literatura

[1] Digitizing art
https://www.shutterstock.com/blog/digitizing-art-guide (14.02.2021).
[2] Toon me app
https://play.google.com/store/apps/details?id=com.vicman.toonmeapp&hl=sl&gl=US (22.02.2021)
[3] Adobe Capture app
https://www.adobe.com/si/products/capture.html (22.02.2021)
[4] Tensorflow
https://en.wikipedia.org/wiki/TensorFlow (14.02.2021)
[5] Why SVG is better then PNG
https://www.growfox.co.uk/blog/5-reasons-you-should-be-using-svgs-over-pngs (14.02.2021)
[6] Why you should use SVG
https://www.creativebloq.com/news/6-reasons-why-you-should-be-using-svg (14.02.2021)
[7] Python
https://en.wikipedia.org/wiki/Python_(programming_language) (14.02.2021)
[8] Image bluring
https://datacarpentry.org/image-processing/06-blurring/ (14.02.2021)
[9] Gaussain blur
https://www.geeksforgeeks.org/apply-a-gauss-filter-to-an-image-with-python/ (14.02.2021)
[10] Grayscale
https://www.linkedin.com/learning/advanced-photoshop-color-correction/understanding-grayscale-values-and-color (14.02.2021)
[11] API
https://www.mulesoft.com/resources/api/what-is-an-api (14.02.2021)
[12] Xamarin
https://en.wikipedia.org/wiki/Xamarin (14.02.2021)