



УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR)РАД

Кандидат: Драган Марјановић

Број индекса: e13069

Тема рада: Евалуација метода за манипулацију
слике на мобилним уређајима

Ментор рада: проф. др. Миодраг Темеринац

Нови Сад, јул, 2013.



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Драган Марјановић		
Ментор, МН:	Др Миодраг Темеринац		
Наслов рада, НР:	Евалуација метода за манипулацију слике на мобилним уређајима		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2013		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страна/цитата/табела/слика/графика/прилога)	8/26/0/1/17/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:			
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У раду је дато једно решење корекције слике на мобилним уређајима		
Датум приhvатања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	Др Небојша Ћевалица, доцент	
	Члан:	Др Иштван Пап, доцент	Потпис ментора
	Члан, ментор:	Др Миодраг Темеринац, ред. професор	



KEY WORDS DOCUMENTATION

Accession number, ANO:		
Identification number, INO:		
Document type, DT:	Monographic publication	
Type of record, TR:	Textual printed material	
Contents code, CC:	Bachelor Thesis	
Author, AU:	Dragan Marjanovic	
Mentor, MN:	Miodrag Temerinac, PhD	
Title, TI:	Evaluation of methods for image manipulation on mobile devices	
Language of text, LT:	Serbian	
Language of abstract, LA:	Serbian	
Country of publication, CP:	Republic of Serbia	
Locality of publication, LP:	Vojvodina	
Publication year, PY:	2013	
Publisher, PB:	Author's reprint	
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendices)	8/26/0/1/17/0/0	
Scientific field, SF:	Electrical Engineering	
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW:	Computer engineering, image processing	
UC		
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N:		
Abstract, AB:	In this paper is given one solution for image correction on mobile devices.	
Accepted by the Scientific Board on, ASB:		
Defended on, DE:		
Defended Board, DB:	President:	Nebojsa Pjevalica, PhD
	Member:	Istvan Pap, PhD
	Member, Mentor:	Prof. Miodrag Temerinac, PhD
	Menthor's sign	

SADRŽAJ

1.	Uvod	1
2.	Teorijske osnove.....	3
2.1	Geometrijske transformacije	3
2.1.1	Projektivna geometrija.....	3
2.1.2	Afine transformacije	4
2.1.3	3D Rotacija.....	6
2.2	Gausova metoda.....	6
2.3	Interpolacija	7
2.4	Android	9
2.5	Android Native Development Kit (NDK).....	10
3.	Evaluacija algoritma	11
3.1	Algoritam zasnovan na 3D rotaciji i projektivnoj transformaciji	11
3.1.1	Proširenje algoritma sa inverznom projektivnom transformacijom	12
3.2	Algoritam zasnovan na afinim transformacijama	13
4.	Koncept rešenja	14
4.1	Aplikativni deo (SDL dijagram)	16
4.2	Algoritamski deo (C programski jezik)	17
5.	Programsko rešenje.....	18
5.1	Aplikativni deo.....	18
5.1.1	ScannerMenu	18
5.1.2	CustomTouchListener	19
5.1.3	ScannerProcessing	19
5.2	Algoritamski deo (C kod)	20
5.2.1	Init.....	21
5.2.2	ImgCorrection.....	21
5.2.3	SolveSystem	21
5.2.4	Interpolation	22

5.2.5 AddAlpha	22
5.2.6 Application.mk fajl.....	22
6. Rezultati.....	23
7. Zaključak	25
8. Literatura	26

SPISAK SLIKA

Slika 1 Originalna i ispravljena slika.....	1
Slika 2 Prikaz označavanja ivica dokumenta.....	2
Slika 3 Matrica afinih transformacija	4
Slika 4 Proširena matrica afinih transformacija.....	5
Slika 5 Rotacija u 3D prostoru oko x ose	6
Slika 6 Rotacija u 3D prostoru oko y ose	6
Slika 7 Problem pri rotiranju slike za ugao od 45°	8
Slika 8 Proračunavanje vrednosti piksela bilinearnom interpolacijom	8
Slika 9 Arhitektura Android operativnog sistema	9
Slika 10 Grafički prikaz Java aplikacije koja koristi JNI	10
Slika 11 Rezultati korekcije slike gore opisanim algoritmom	12
Slika 12 Sprega Jave sa C stranom preko JNI-a	15
Slika 13 Grafički prikaz aplikativnog dela (SDL dijagram).....	16
Slika 14 Prikaz algoritma realizovanog u C kodu	17
Slika 15 Izgled glavnog menija	18
Slika 16 Izlazne slike iz CamScanner-a.....	24
Slika 17 Izlazne slike realizovane aplikacije	24

SPISAK TABELA

Tabela 1 Vremena potrebna za korekciju slika različitih rezolucija..... 23

SKRAĆENICE

SDK – *Android Software Development Kit*, Android okruženje za razvoj programske podrške.

NDK – *Android Native Code Development Kit*, radno okruženje koje omogućava Java da poziva C kod.

JNI – *Java Native Interface*, specifikacija koja omogućava pozivanje C koda iz Java okruženja.

API – *Application Programable Interface*, Aplikaciona programska sprega

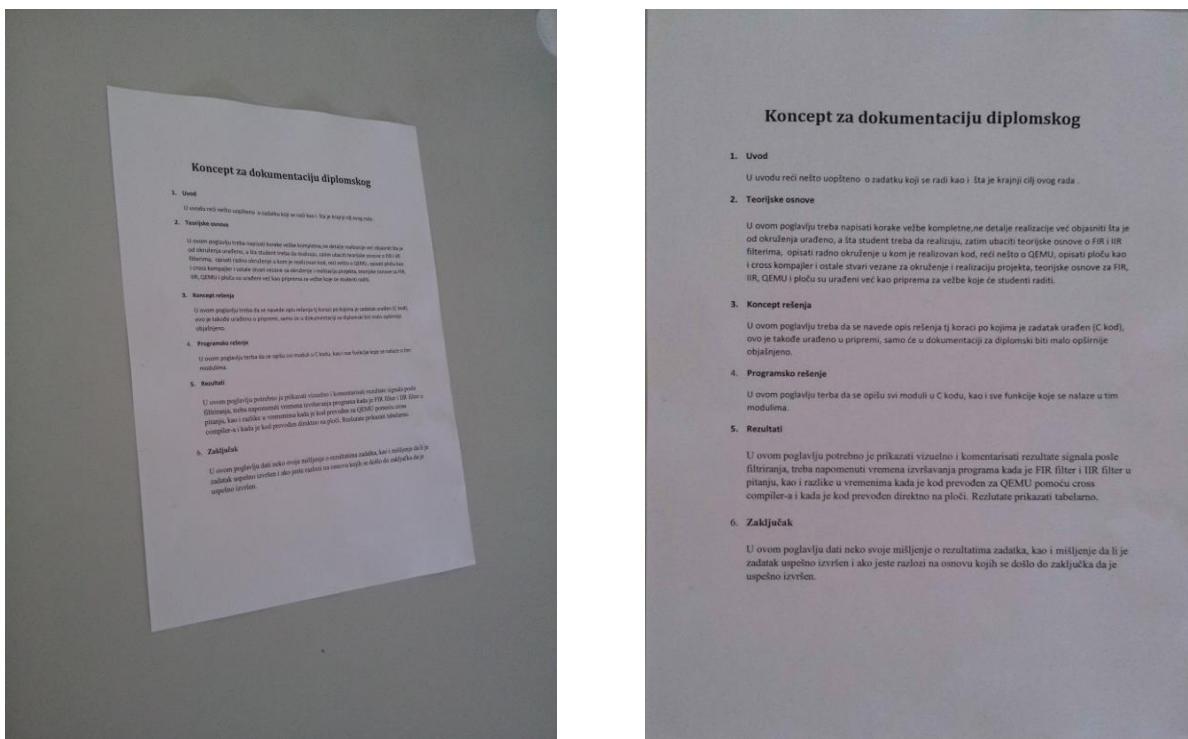
SDL – *Specification and Description Language*, formalni jezik za specifikaciju i opis sistema.

SIMD – *Single Instruction Multiple Data*, jedna instrukcija više podataka.

MPx – Megapiksel, milion piksela.

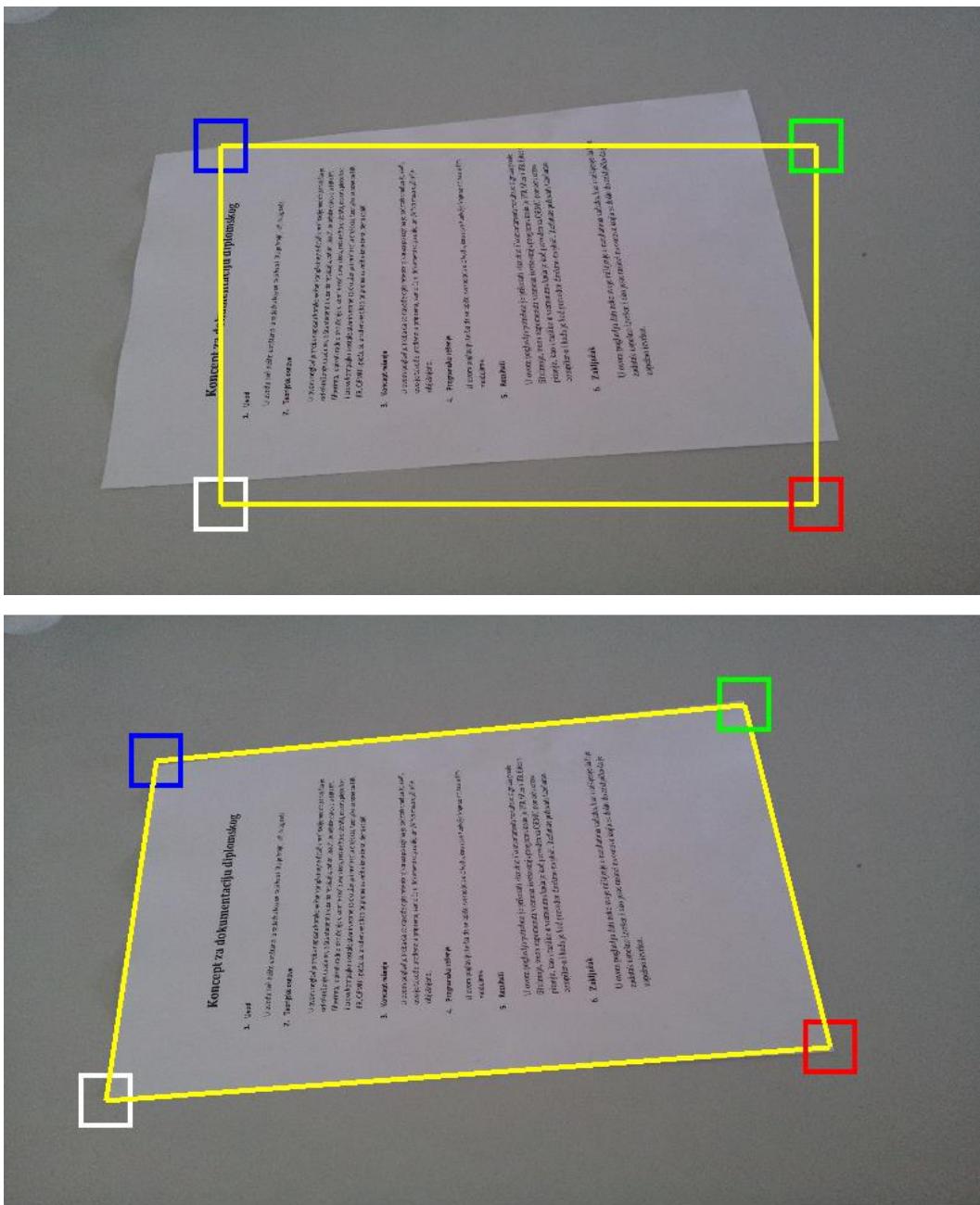
1. Uvod

Kada uslikamo dokument, tablu ili neki sličan predmeta pravougaonog oblika na slici će delovati kao nepravilan četvorougao (ivice bliže posmatraču će delovati duže u odnosu na ivice koje se nalaze dalje od posmatrača). Cilj ovog zadatka je razviti aplikaciju primenjivu na mobilnom uređaju, koja vrši ispravljanje uslikanog dokumenta tako da posmatrač dobije utisak kako je dokument skeniran.



Slika 1 Originalna i ispravljena slika

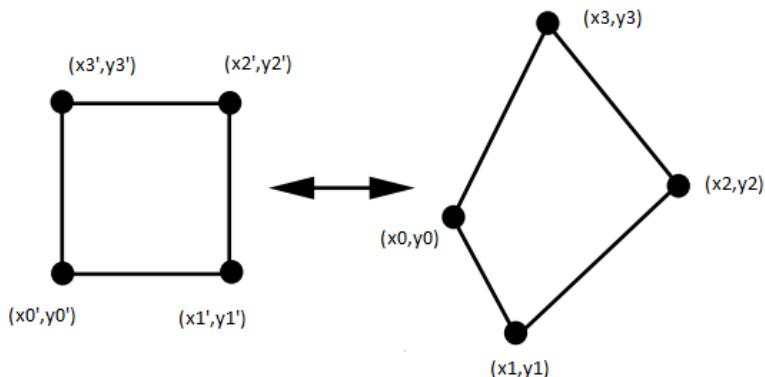
Iz galerije je potrebno odabrati sliku nad kojom se vrši korekcija i odrediti ivice predmeta koji se ispravlja. Pritiskom na odgovarajući taster vrši se korekcija. Nakon toga potrebno je obezbediti mogućnost čuvanja korigovane slike.



Slika 2 Prikaz označavanja ivica dokumenta

2. Teorijske osnove

Glavna problematika ovog rada je kako nepravilan četvorougao preslikati u pravougaonik, odnosno kako pronaći sve geometrijske trasformacije koje su za to neophodne.



2.1 Geometrijske transformacije

2.1.1 Projektivna geometrija

Projektivna geometrija [1] je nastala iz potrebe da se trodimenzionalni prostor realno prikaže na dvodimenzionalnom mediju. Kad se kreira slika nekog pravougaonog predmeta ivica koja je dalje od posmatrača (kamere) delovaće manje u odnosu na njoj paralelna ivicu koja se nalazi bliže posmatraču. Usled ovoga predmet koji je u realnom prostoru pravougaonog oblika na slici će izgledati kao nepravilan četvorougao.

Gore opisana transformacija se može opisati sledećim promenjivima :

a_{x,y,z} - 3D koordinate tačke koja se projektuje

c_{x,y,z} - 3D koordinate tačke koja predstavlja kameru

θ_{x,y,z} - orijentacija kamere

e_{x,y,z} - relativna pozicija posmatrača u odnosu na ekran

b_{x,y} - 2D koordinate koje predstavljaju projekciju tačke a

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{bmatrix} \begin{bmatrix} \cos \theta_y & 0 & \sin \theta_y \\ 0 & 1 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y \end{bmatrix} \begin{bmatrix} \cos \theta_z & -\sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix} - \begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ \mathbf{c}_z \end{bmatrix} \right)$$

$$\begin{aligned} \mathbf{b}_x &= \frac{\mathbf{e}_z}{\mathbf{d}_z} \mathbf{d}_x - \mathbf{e}_x \\ \mathbf{b}_y &= \frac{\mathbf{e}_z}{\mathbf{d}_z} \mathbf{d}_y - \mathbf{e}_y \end{aligned}$$

2.1.2 Afine transformacije

Afine transformacije [2] omogućavaju da više linearnih transformacija kao što su rotacija, skaliranje, smicanje i translacija predstavimo samo jednom matricom čiji koeficijenti opisuju date transformacije. Afine transformacije čuvaju kolinearnost, paralelnost, odnos duži na pravoj, kao i odnos površina.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slika 3 Matrica afinih transformacija

Iz gore navednog može se zaključiti kako same afine transformacije nisu dovoljne kako bi se dokument koji je nepravilan četvorougao transformisao u pravougaonik, jer one omogućavaju transformacije paralelograma. Ovaj problem se rešava proširivanjem matrice afinih transformacija sa koeficijentima koji opisuju projektivne transformacije.

Kada se matrica afinih transformacija proširi dodatnim koeficijentima [3] koji opisuju projektivne transformacije moguće je opisati transformaciju koja preslikava temena nepravilnog četvorougla (x, y) u temena pravougaonika (x', y').

$$\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slika 4 Proširena matrica afinih transformacija

$$sx' = ax + by + c$$

$$sy' = dx + ey + f$$

$$s = gx + hy + 1$$

$$x' = \frac{ax + by + c}{gx + hy + 1}$$

$$y' = \frac{dx + ey + f}{gx + hy + 1}$$

2.1.3 3D Rotacija

Matrice koje opisuju rotacije u prostoru oko x i oko y ose prikazane se na slikama ispod.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Slika 5 Rotacija u 3D prostoru oko x ose

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Slika 6 Rotacija u 3D prostoru oko y ose

2.2 Gausova metoda

Gausova metoda [4] predstavlja jednu od metoda za rešavanje sistema linearnih jednačina.

Pretpostavimo da imamo sistem od N linearnih jednačina

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2N}x_N &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \dots + a_{3N}x_N &= b_3 \\ a_{N1}x_1 + a_{N2}x_2 + a_{N3}x_3 + \dots + a_{NN}x_N &= b_N \end{aligned}$$

gde su x_i nepoznate, a_{ij} koeficijenti uz nepoznate, a b_i slobodni članovi.

Gausov metod se zasniva na redukciji sistema na trougaonu formu.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1N}x_N &= b_1 \\ a_{22}x_2 + a_{23}x_3 + \dots + a_{2N}x_N &= b_2 \\ a_{33}x_3 + \dots + a_{3N}x_N &= b_3 \\ \dots + a_{NN}x_N &= b_N \end{aligned}$$

Najvažnija osobina trougaone forme jeste da su svi koeficijenti $a_{ij} = 0$ za svako $i > j$. Sada možemo iz poslednje jednačine pronaci $x_N = b_N / a_{NN}$. Uvrštavanjem x_N u sledeću jednačinu možemo dobiti x_{N-1} i tako dalje dok ne pronađemo x_1 .

Sada je potrebno pronaći način kako da se sistem od N linearnih jednačina svede na trougaonu formu. Ukoliko želimo da u drugoj jednačini koeficijent uz x_1 svedemo na 0, potrebno je da prvu jednačinu pomnožimo sa M i da je zatim dodamo drugoj jednačini.

$$(a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1N}x_N = b_1) * M + \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2N}x_N = b_2$$

Kako bi koeficijent uz x_1 u drugoj jednačini sveli na 0 potrebno je da je $M = -a_{21} / a_{11}$.

Nakon sabiranja prve i druge jednačine dobijamo novu jednačinu koja zamenjuje originalnu jednačinu. Na isti način moguće je transformisati i treću jednačinu samo što je ovde $M = -a_{31}/a_{11}$. Ovaj postupak se primenjuje nad svim jednačinama sistema i dobija se novi sistem jednačina.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1N}x_N &= b_1 \\ a_{22}x_2 + a_{23}x_3 + \dots + a_{2N}x_N &= b_2 \\ a_{32}x_2 + a_{33}x_3 + \dots + a_{3N}x_N &= b_3 \\ a_{N2}x_2 + a_{N3}x_3 + \dots + a_{NN}x_N &= b_N \end{aligned}$$

Nakon ovoga potrebno je koeficijente uz x_2 u trećoj, četvrtoj, N-toj jednačini svesti na 0. Za to je potrebno drugu jednačinu pomnožiti sa $M = -a_{j2} / a_{22}$, a zatim dodati j-toj jednačini.

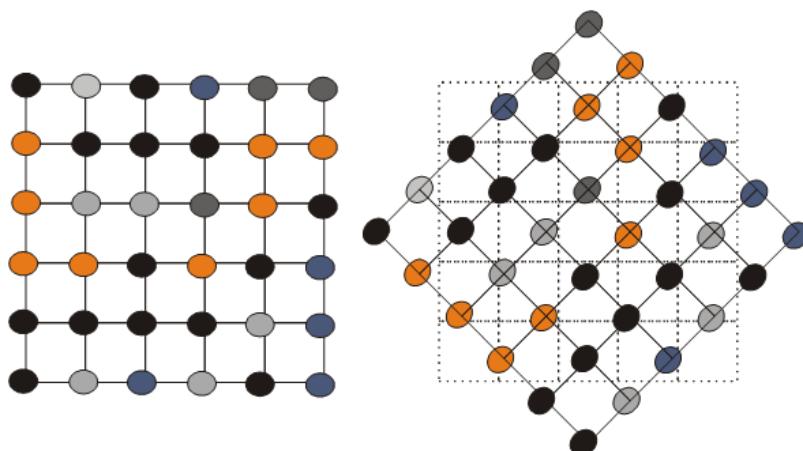
Posle ove faze sve jednačine sem prve 2 su ostale bez člana x_2 . Postupak se ponavlja sve dok se sistem ne svede na trougaonu formu.

2.3 Interpolacija

Prilikom primene većine geometrijskih transformacija nad slikom, pikseli originalne slike se ne preslikavaju u čvorove rešetke izlazne slike. Da bi se odredile vrednosti u čvorovima rešetke u odredišnoj slici koristi se interpolacija.

**Originalna slika u čvorovima
rešetke su pikseli.**

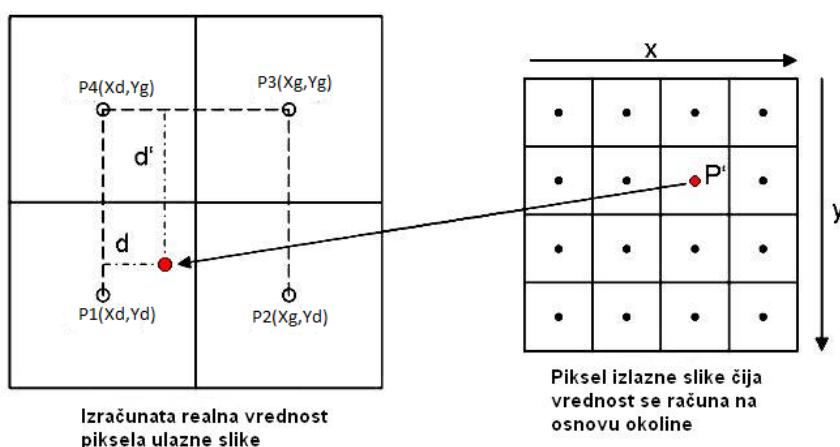
**Slika rotirana za 45 stepeni.
Pikseli iz prethodne slike se
više ne nalaze u čvorovima
rešetke.**



Slika 7 Problem priotiranju slike za ugao od 45°

Interpolacija [5] je realizovana u 2 koraka. Prvi korak čini određivanje koordinata piksela u originalnoj slici korišćenjem inverzne funkcije geometrijskih transformacija koje se sprovode nad originalnom slikom.

Izračunate koordinate ovih piksela u originalnoj slici su uglavnom realne vrednosti, pa se u drugom koraku koristi bilinearna interpolacija. Zaokruživanjima na manju i veću celobrojnu vrednost dobija se okolina na osnovu koje se proračunava vrednost za piksel u izlaznoj slici.



Slika 8 Proračunavanje vrednosti piksela bilinearnom interpolacijom

$$D = P1; \quad C = P2 - P1; \quad B = P4 - P1; \quad A = P3 - P4 + P1 - P2;$$

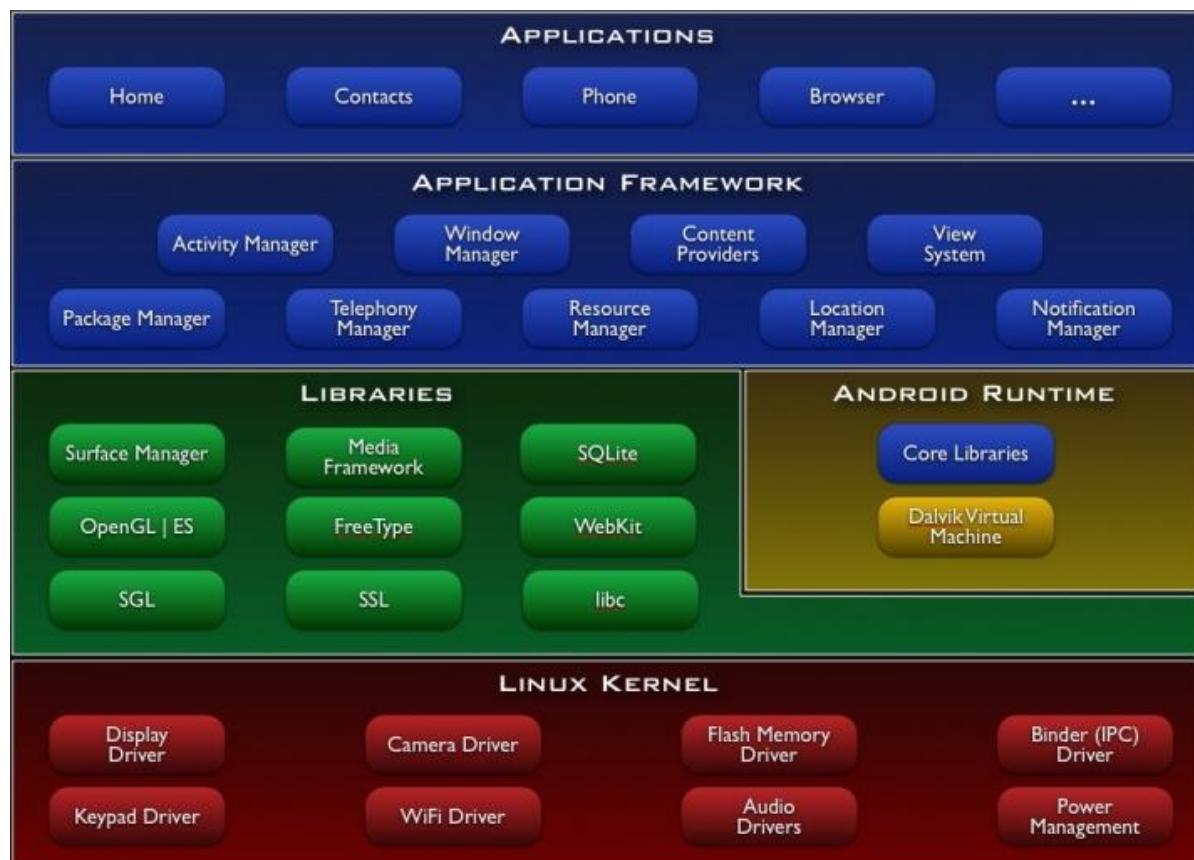
$$P' = A * (x - xd) * (y - yd) + B * (x - xd) + C * (y - yd) + D;$$

2.4 Android

Android [6] je operativni sistem zasnovan na Linux jezgru i trenutno se koristi na većini moblinih uređaja (tableta, pametnih telefona, čitača elektronskih knjiga i mnogih drugih).

Iako su C i C++ programski jezici korišćeni za radno okružje (framework), većina aplikacija pisana je u Java programskom jeziku koristeći Android Software Development Kit (SDK). Postoji mogućnost pisanja aplikacija i u C/C++ programskom jeziku, ali tada se koristi Android Native Code Development Kit (NDK). Ovim postupkom omogućava se bolje raspolažanje resursima kao i mnogo bolje performanse aplikacija.

Arhitektura Androida prikazana je na slici i stoji se iz više nivoa koji naležu jedan na drugi.



Slika 9 Arhitektura Android operativnog sistema

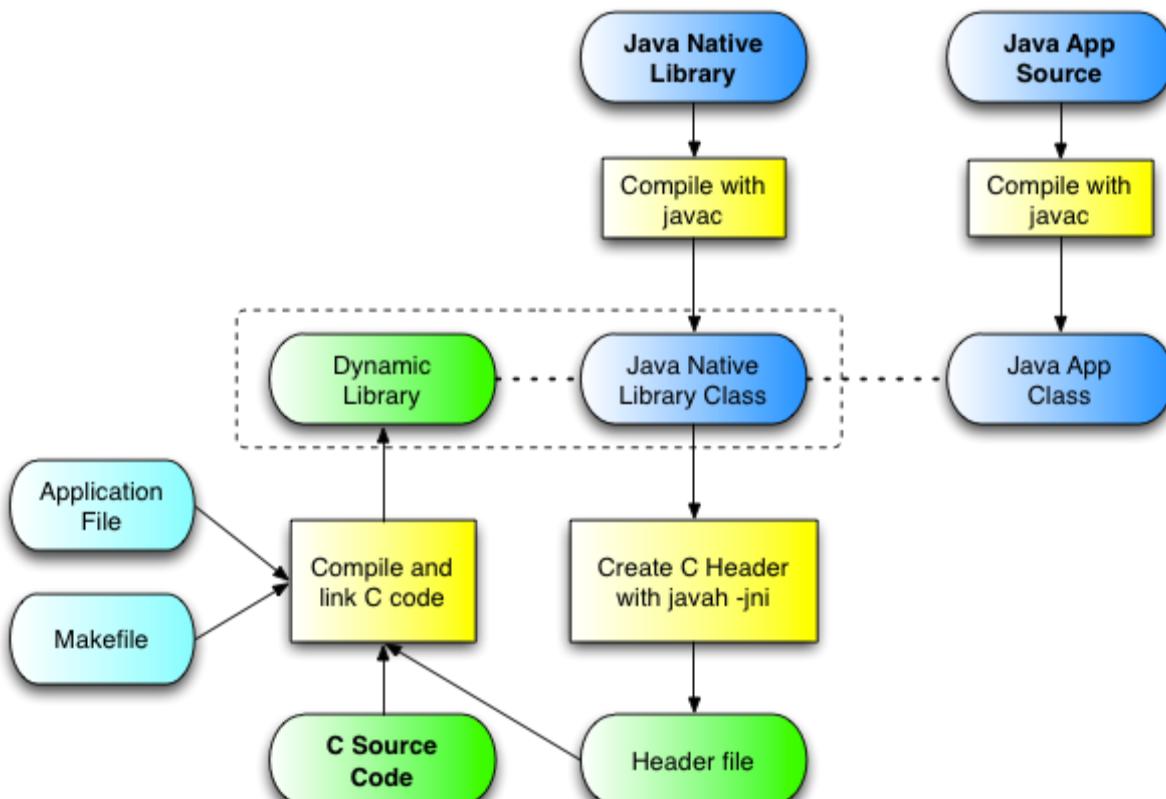
Na dnu arhitekture nalazi se jezgro Linux-a koje deluje kao sloj apstrakcije između hardvera i ostatka steka. Sadrži sve neophodne drafvere među kojima su najvažniji za međuprocesorsku komunikaciju i upravljanje napajanjem.

U narednom nivou su biblioteke pisane u C/C++ programskom jeziku. Zatim sledi Andorid Runtime koji sačinjavaju 2 važne komponente : Core libraries – biblioteke jezgra, a druga je Dalvik Virtual Machine koja pokreće aplikacije kao zasebne procese, odnosno kao instance virtualne mašine. Nakon ovog sloja dolazi aplikacijsko okruženje koje dozvoljava upotrebu svih API-ja.

Na vrhu se nalaze same aplikacije koje su vidljive krajnjem korisniku.

2.5 Android Native Development Kit (NDK)

NDK [7] je radno okruženje (framework) koje omogućava da Java poziva kod pisan u C ili C++ programskom jeziku. Java kod prevodi se u bajt-kod, dok se drugi programski jezici kao što su C i C++ prevode u mašinski kod koji se direktno izvršava u procesoru. Ovaj kod se lakše može prilagoditi ciljanoj platformi, pa samim tim se postižu mnogo bolje performanse.



Slika 10 Grafički prikaz Java aplikacije koja koristi JNI

3. Evaluacija algoritma

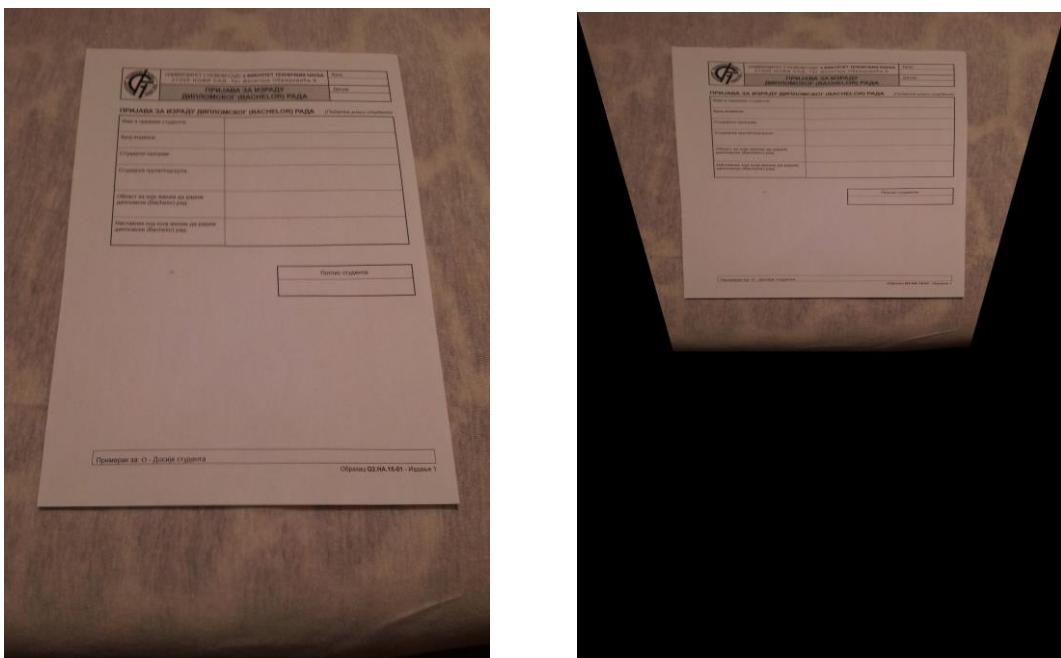
3.1 Algoritam zasnovan na 3D rotaciji i projektivnoj transformaciji

Ovaj algoritam se zasniva na rotaciji svakog piksela oko x ili y ose. Svaki piksel slike smešten je u matricu i njegova pozicija se može predstaviti preko vrste (koordinata y) i kolone (koordinata x).

Ukoliko korisnik želi da rotira sliku za 10° po horizontalnoj osi, svaki piksel date slike će biti rotiran oko x ose za taj ugao. Ovom 3D rotacijom pikselima se dodaje prostorna koordinata z koja je inicijalno bila 0, pa se mogla ignorisati. Sada je potrebno preko projektivne transformacije iz 3D koordinata piksela izračunati 2D koordinate kako bi se prikazali na ekranu.

Interpolacija je realizovana u 2 koraka, u prvom se inverznom funkcijom od gore navedenih transformacija svakom pikselu izlazne slike nalazi njemu odgovarajući u ulaznoj, a zatim se u drugom koraku vrši bilinearna interpolacija.

Ovaj algoritam je uspešno ispravljao dokumente u ravan posmatrača, ali nije zadržavao proporciju dokumenta, jer se iz same slike dokumenta, koja predstavlja projekciju originala ne mogu odrediti njegove stvarne dimenzije, na njih utiče i komponenta z, koja je u slikama uvek 0.



Slika 11 Rezultati korekcije slike gore opisanim algoritmom

3.1.1 Proširenje algoritma sa inverznom projektivnom transformacijom

Algoritam od gore proširuje se podatkom pod kojim uglom je dokument uslikan i ukoliko su sve tačke dokumenta u istoj ravni moguće je znati za koje uglove po horizontalnoj i vertikalnoj osi je potrebno rotirati dokument kako bi se postavio u ravan paralelnu kameri (posmatraču).

Kada nam je poznat taj ugao, znamo da ukoliko se dokument rotira za taj ugao dovešće se u ravan $z = z_c$ (udaljenost kamere od dokumenta). Sada dobijamo još jednu vezu između stvarnih i projektovanih koordinata dokumenta i moguće je inverznom projektivnom transformacijom izračunati 3D koordinate dokumenta. 3D rotacijom za date uglove dokument se ispravlja u ravan posmatrača. Takođe i ovaj algoritam koristi interpolaciju kao i prethodni.

Glavna mana ovog algoritma je velika kompleksnost, što je i sprečilo njegovu dalju realizaciju.

3.2 Algoritam zasnovan na afnim transformacijama

Ovaj algoritam se pokazao kao najefikasniji i ujedno jednostavan za implementaciju. Za njegovu realizaciju korišćene su affine transformacije u kombinaciji sa projektivnom geometrijom.

$$x' = \frac{ax + by + c}{gx + hy + 1} \quad y' = \frac{dx + ey + f}{gx + hy + 1}$$

Tačke (x,y) predstavljaju koordinate temena nepravilnog četvorougla, dok (x',y') predstavljaju temena pravougaonika u koji će se četvorougao preslikati. Kako znamo temena i nepravilnog četvorougla (zadaje ih korisnik) i pravougaonika (određen je širinom i visinom slike) potrebno je postaviti sistem od 8 linearnih jednačina sa 8 nepoznatih. Po rešavanju sistema Gausovom metodom, dobijamo koeficijente a,b,c,d,e,f,g i h koji opisuju transformaciju koju je potrebno realizovati.

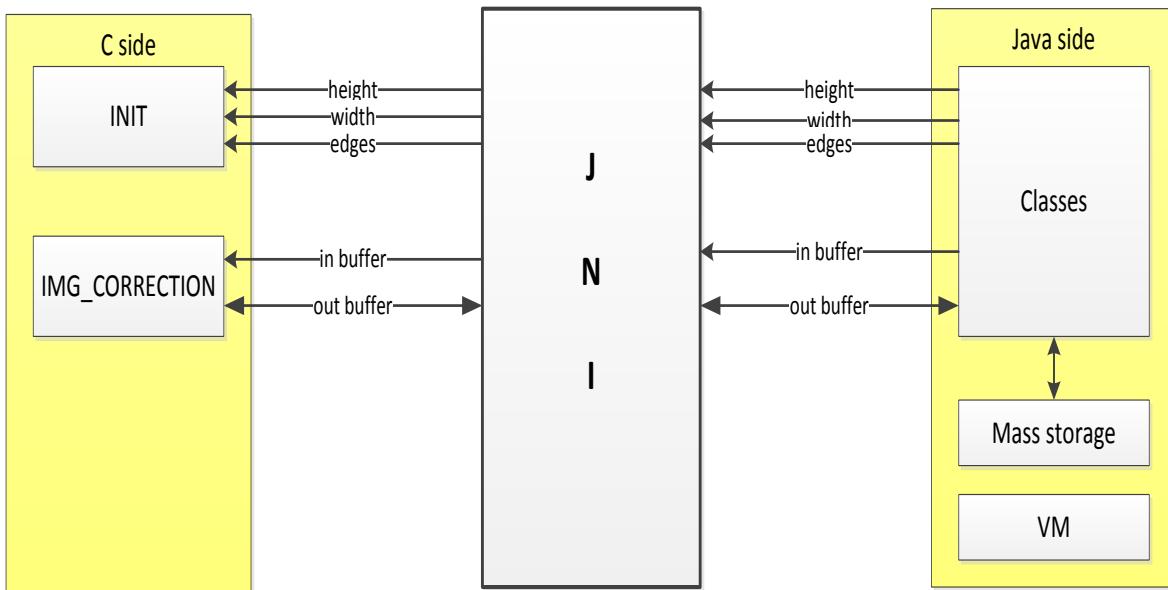
Sada je poznata transformacija koja preslikava nepravilan četvorougao u pravougaonik. Kako je za realizaciju interpolacije potrebna inverzna funkcija geometrijskih transformacija neophodno je definisati transformaciju pravougaonika u nepravilan četvorougao.

U prvom koraku interpolacije proračunavaju se odgovarajuće koordinate piksela ulazne slike korišćenjem transformacije koja preslikava pravougaonik u nepravilan četvorougao. U drugom koraku se biliearnom interpolacijom proračunavaju vrednosti svih piksela izlazne slike.

4. Koncept rešenja

Rešenje je podeljeno na 2 celine, algoritamski deo (C programski jezik) i aplikativni deo (realizovan u Javi), gde se nalazi sprega sa korisnikom.

U aplikativnom delu korisnik odabira sliku iz galerije nad kojom želi da vrši korekciju i ta slika se smešta u ulazni bafer koji će kao parametar biti prosleđen funkciji za korekciju slike koja se nalazi u algoritamskom delu. Korisnik ručno zadaje ivice predmeta koji želi da ispravi i poziva se prvo funkcija koja je zadužena za inicijalizaciju promenjivih u algoritamskom delu. Njeni argumenti su dimenzije slike (određuju pravougaonik, ispravljen predmet) i koordinate temena originalnog predmeta. Nakon inicijalizacije promenjivih poziva se funkcija koja vrši korekciju slike. Njoj se od parametara prosleđuju ulazni bafer koji sadrži originalnu sliku i izlazni bafer u koji će da bude smeštena izlazna slika.



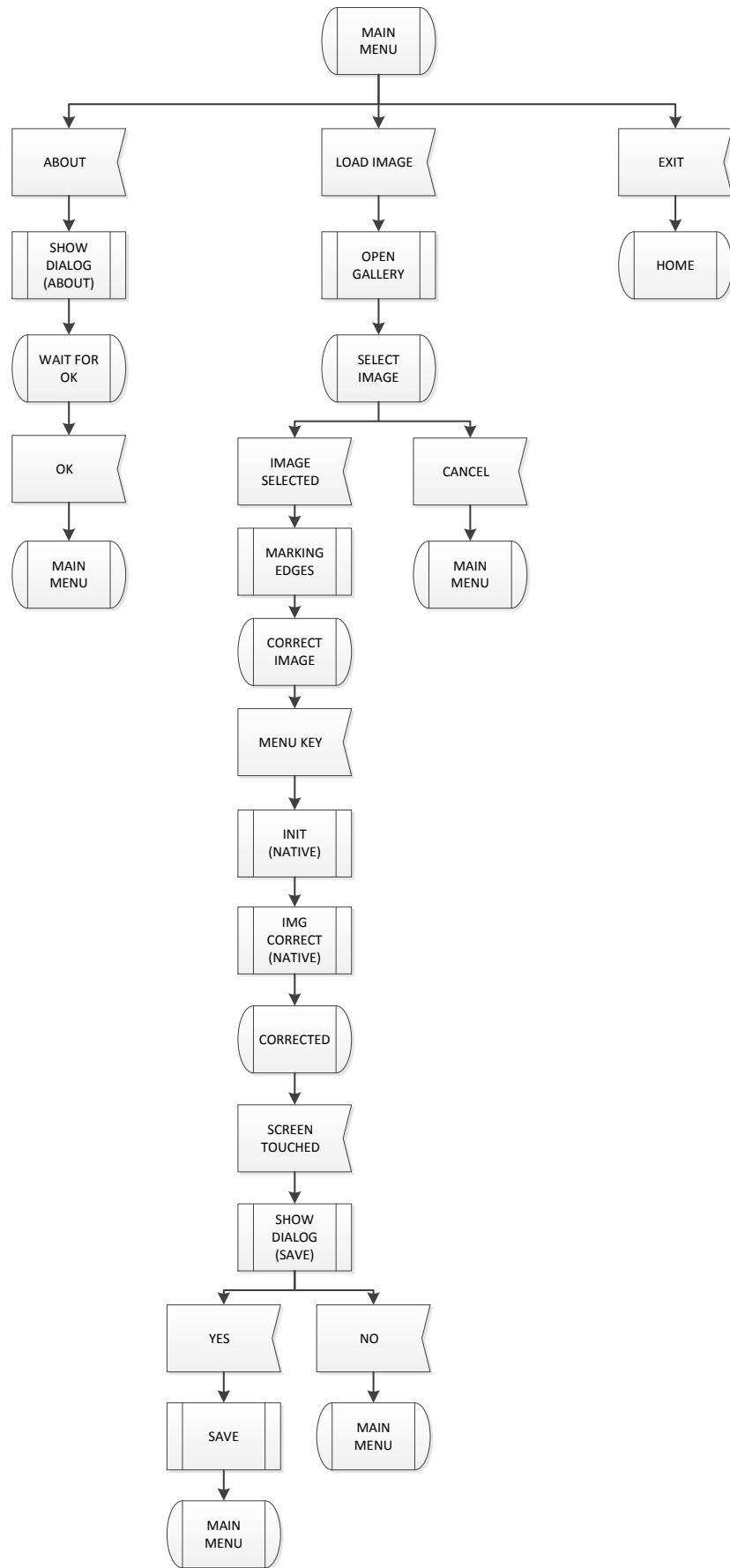
Slika12 Sprega Java sa C stranom preko JNI-a

Po završetku ove funkcije izlazna slika se u aplikativnom delu prikazuje korisniku. Po dodiru na ekran ispisuje se dijalog koji korisniku daje opciju da sačuva ispravljenu sliku ili da je odbaci. Kada korisnik odabere neku od opcija može da izabere novu sliku nad kojom želi da vrši korekciju.

Algoritamski deo rešava sistem od 8 jednačina sa 8 nepoznatih na osnovu prosleđenih temena predmeta od strane aplikativnog dela. Rešavanjem sistema dobijaju se koeficijenti koji opisuju transformaciju koju je potrebno realizovati nad svakim pikselom ulazne slike kako bi se ispravila. Sledeći korak je interpolacija koja u 2 koraka daje izlaznu sliku.

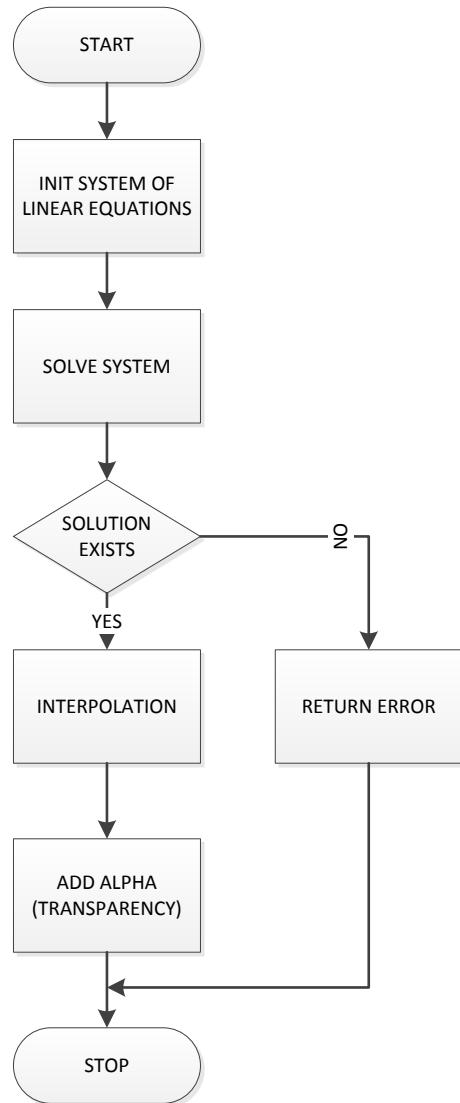
Na kraju se u izlazni bafer dodaje providnost (alfa) čime se završava algoritamski deo.

4.1 Aplikativni deo (SDL dijagram)



Slika 13 Grafički prikaz aplikativnog dela (SDL dijagram)

4.2 Algoritamski deo (C programske jezik)



Slika 14 Prikaz algoritma realizovanog u C kodu

5. Programsко rešenje

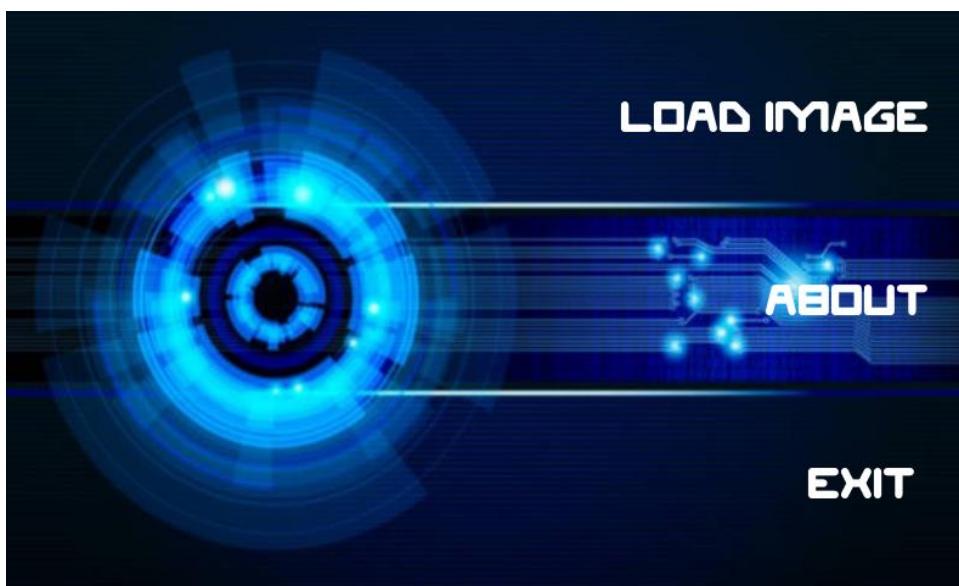
5.1 Aplikativni deo

Aplikativni deo se sastoji iz 3 klase :

- ScannerMenu
- CustomTouchListener
- ScannerProcessing

5.1.1 ScannerMenu

ScannerMenu je klasa koja se prva pokreće i u njoj se nalazi glavni meni koji se prikazuje korisniku pri pokretanju aplikacije.



Slika 15 Izgled glavnog menija

Interfejs se sastoji od tekstulanih polja (engl. TextView) na koja se postavljaju osluškivači (engl. listener) kako bi se stvorio željeni odziv na komandu korisnika.

Metode klase ScannerMenu :

- onCreate – metoda koja se prva poziva i zadužena je za postavljanje rasporeda (engl. layout) kreiranog u main.xml fajlu, kreiranje fonta i postavljanje osluškivača na tekstualna polja.
- onClick – metoda koja se poziva na klik (dodir na ekran) korisnika. U zavisnosti od opcije koju je korisnik izabrao vrši se odgovarajući odziv. Ako je pritisnuto tekstualno polje LOAD IMAGE pokreće se klasa ScannerProccesing, na polje ABOUT poziva se metoda aboutDialog() i na kraju na polje EXIT aplikacija se zatvara.
- aboutDialog – ova metoda ispisuje dijalog u kome je dat opis kako treba da izgleda interakcija korisnika sa aplikacijom.

5.1.2 CustomTouchListener

Klasa koja implementira samo jednu metodu onTouch čiji je zadatak da na dodir menja boju fonta tekstualnih polja.

5.1.3 ScannerProcessing

Klasa ScannerProcessing sadrži klasu za iscrtavanje po ekranu myView. U ovoj klasi se nalazi metoda onDraw koja iscrtava prvo sliku koja je učitana iz galerije, a zatim i novu sliku koja se dobila ispravljanjem ulazne. Uz ulaznu sliku ova metoda iscrtava 4 linije i kvadrata koja je neophodno (od strane korisnika) pozicionirati na krajeve predmeta koji se ispravlja. Metoda onDraw poziva se pri svakoj promeni pozicije bilo kog od pravougaonika ili posle ispravljanja slike kada se prikazuje samo nova slika.

Metode klase ScannerProcessing:

- onCreate – metoda koja se prva poziva za inicijalizaciju pojedinih promenjivih, kao i za pozivanje galerije slika kako bi korisnik odabrao željeni fajl.
- onActivityResult– preuzima putanju fajla kojeg je korisnik odabrao u galeriji i poziva metodu onDraw koja ga iscrtava na ekran.
- onTouchEvent – metoda koja se poziva pri svakom dodiru ekrana i ukoliko je slika učitana proverava se da li je korisnik dodirnuo neki od kvadrata kako bi im se promenila pozicija. Ukoliko je slika obrađena i može biti sačuvana, na dodir ekrana poziva se metoda saveDialog koja korisniku daje opciju da sačuva izlaznu sliku.
- onKeyDown– ukoliko je pritisnut meni taster koordinate na kojima se nalaze kvadrati uzimaju se kao koordinate ivica predmeta i poziva se NATIVE funkcija koja vrši korekciju slike.
- saveDialog– prikazuje dijalog koji korisniku daje mogućnost čuvanja slike, zatim nakon odabira završava se ovaj *Activity* i prikazuje se glavni meni.
- onDestroy– oslobađanje zauzetih resursa i završetak *Activity*-a

5.2 Algoritamski deo (C kod)

U algoritamskom delu je realizovano preslikavanje nepravilnog četvorougla u pravougaonik preko sledećih funkcija :

- NATIVE funkcija Init
- NATIVE funkcija ImgCorrection
- SolveSystem
- Interpolation
- AddAlpha

5.2.1 Init

```
Init(JNIEnv *env, jobject thiz, jintArray jedges, jint jwidth, jint jheight)
```

Zadatak ove funkcije je da na osnovu parametara prosleđenih iz Java inicijalizuje promenjive kako bi se kasnije u daljoj obradi koristile.

Prva 2 parametra ove funkcije su prisutna u svim NATIVE funkcijama. Prvi parametar predstavlja pokazivač na JNIEnv strukturu koja predstavlja vezu NATIVE koda sa Java virtualnom mašinom. Drugi parametar je referenca na samu klasu koja sadrži NATIVE metode.

Treći parametar je niz celobrojnih vrednosti koji predstavljaju (x,y) koordinate ivica nepravilnog četvorougla. Ovaj nepravilan četvorougao će se preslikati u pravougaonik širine jwidth i visine jheight.

5.2.2 ImgCorrection

```
ImgCorrection(JNIEnv *env, jobject thiz, jbyteArray inBuff, jbyteArray outBuff)
```

ImgCorrection je NATIVE funkcija koja kao parametre dobija ulazni i izlazni bafer. Ova funkcija poziva sve ostale funkcije koje su neophodne kako bi se dobila izlazna slika (nešto tipa main funkcije).

5.2.3 SolveSystem

```
void SolveSystem()
```

Nakon što se u funkciji Init postave koordinate temena predmeta koji se ispravlja i koordinate pravougaonika u koji se ispravlja poziva se funkcija SolveSystem koja rešava sistem od 8 linearnih jednačina sa 8 nepoznatih Gausovom metodom.

U ovoj funkciji nalaze se još i pozivi nekoliko funkcija :

- AllocMatrix – alokacija matrica za rešenja $x[N]$, koeficijente $a[N][N]$ i slobodne članove $b[N]$, N – red sistema.
- ReadMatrix – na osnovu prosleđenih ivica ulaznog i izlaznog četvorougla postavlja se sistem linearnih jednačina.
- Diagonal – ova funkcija proverava da li ima 0 na glavnoj dijagonali i zamena redova u tom slučaju (kako bi se izbeglo deljenje sa 0).
- FreeMatrix – oslobođanje zauzetih matrica.

5.2.4 Interpolation

```
void Interpolation()
```

Funkcija koja se sastoji iz dva dela. Prvi deo za svaki piksel izlazne slike proračunava odgovarajući piksela ulazne slike koristeći transformaciju iz pravougaonika u nepravilan četvorougao.

Koordinate ovog piksela ulazne slike su uglavnom realne vrednosti pa se vrši zaokruživanje na veću i manju celobrojnu vrednost kako bi se dobila okolina tog piksela iz koje će se proračunati izlazni piksel (bilinearna interpolacija-drugi korak).

5.2.5 AddAlpha

Postavljanje prozirnosti (alfa) na 255 (neprozirno) u izlaznom baferu.

5.2.6 Application.mk fajl

U ovom fajlu nalazi se komanda `APP_ABI := armeabi armeabi-v7a`, koja govori kompjleru da se radi o ARM arhitekturi. Ovom komandom kompjler se navodi da koristi ARM Thumb[®]-2 set instrukcija, kao i naprednu SIMD (engl. Single Instruction Multiple Data) arhitekturu. Proširena napredna SIMD arhitektura, njena implementacija i prateća programska podrška se često nazivaju NEON tehnologija.

6. Rezultati

Rešenje je testirano na Android platformi na nekoliko uređaja, SAMSUNG Galaxy S2, HTC One X i LG P990, koji rade na ARM Cortex-A9 procesoru.

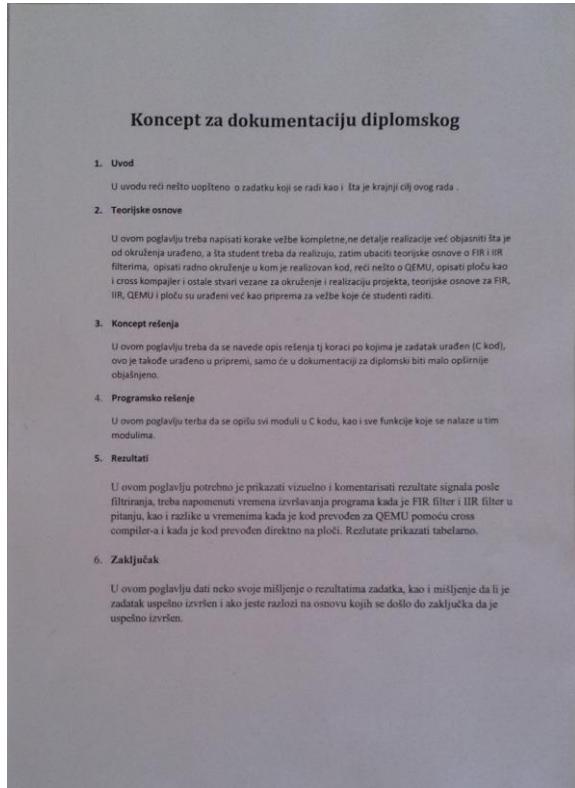
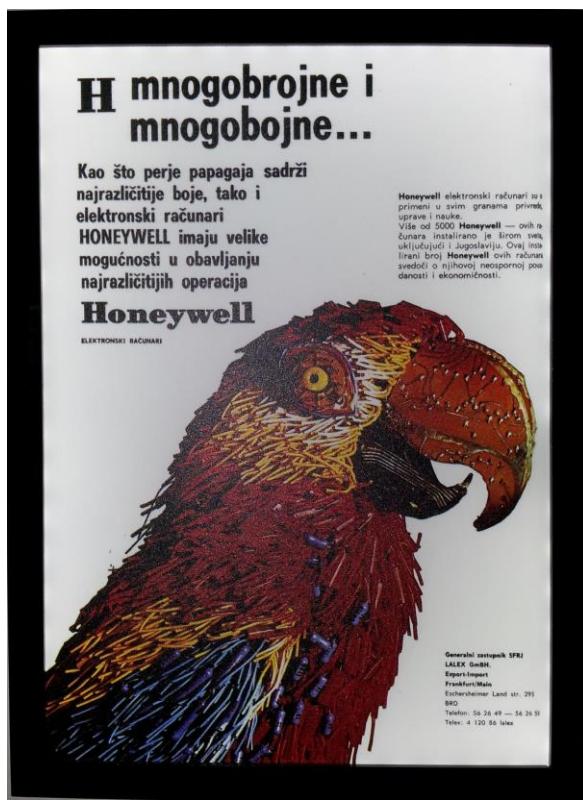
Za različite rezolucije slike dobijaju se različita vremena za koja se izvrši korekcija.

Rezolucija slike [MPx]	Vreme za koje se izvrši korekcija slike[ms]
8	2 110
6.5	1 799
3.2	945
2.4	782
0.4	135

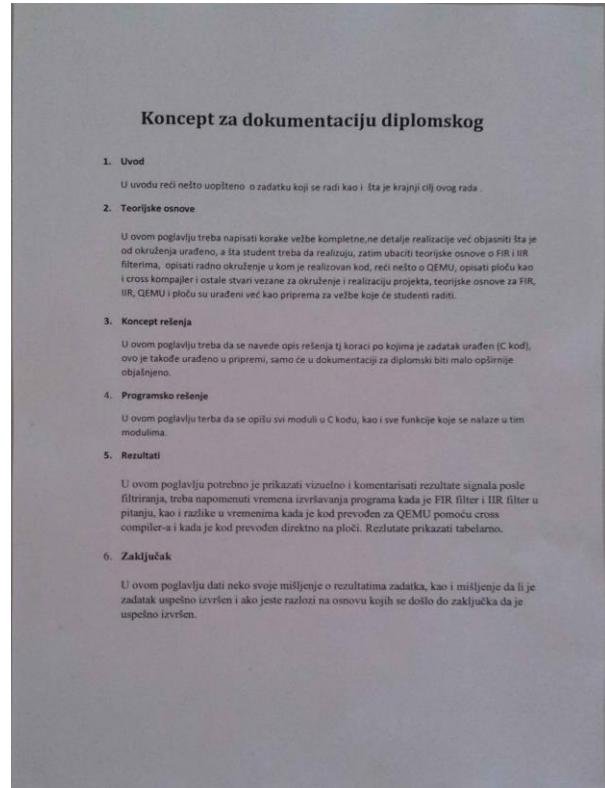
Tabela 1 Vremena potrebna za korekciju slika različitih rezolucija

Poređenja radi, slična aplikacija CameraScanner (može se pronaći na Android Marketu) izvršava korekciju slike od 8 MPx za 3 800 – 4 800 ms. Vremena su data u rasponu, jer ova aplikacija radi samo sa pikselima koji prestavljaju predmet, pa za različite slike se dobijaju različita vremena.

Takođe, dati su izlazi iz obe aplikacije radi poređenje kvaliteta izlazne slike.



Slika 16 Izlazne slike iz CamScanner-a



Slika 17 Izlazne slike realizovane aplikacije

7. Zaključak

U ovom radu predstavljeno je jedno rešenje algoritma i aplikacije za korekcije uslikanih dokumenata.

Subjektivnim poređenjem (performanse i kvalitet slike) sa aplikacijama slične funkcionalnosti, kao na primer CamScanner, može se zaključiti da ova aplikacija ravnopravno parira jednom komercijalnom rešenju.

Aplikacija je ograničena na ispravljanje predmeta čije sve tačke se nalaze u istoj ravni, odnosno nemoguće je potpuno ispraviti izgužvane ili iscepane dokumente. Ukoliko su ivice dokumenta neravne, ostaće takve i nakon ispravljanja.

Zanimljivo bi bilo realizovati ovaj algoritam korišćenjem neke druge interpolacije, na primer bikubične i tako dobijene rezultate (kvalitet slike i performanse) porediti sa ovim rešenjem.

Rešenje je moguće dodatno poboljšati ubacivanjem detekcije ivica, kako korisnik ne bi morao manuelno da označava ivice dokumenta. Takođe, vrlo praktično bi bilo kada bi se pored opcije da se učita slika, dodala i opcija da se uslika dokument koji želi da se koriguje.

8. Literatura

- [1] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/3D_projection
- [2] ArcGIS Resources,
http://resources.esri.com/help/9.3/arcgisdesktop/com/gp_toolref/coverage_tools/how_transform_coverage_works.htm
- [3] Geometric image transformation, North Carolina Central University
- [4] Gauss's method to solve system of linear equation,
http://www.alexeypetrov.narod.ru/Eng/C/gauss_about.html
- [5] Miodrag Temerinac, Željko Lukač : OAiS DSP2 skripta, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka
- [6] Wikipedia, the free encyclopedia,
[http://en.wikipedia.org/wiki/Android_\(operating_system\)](http://en.wikipedia.org/wiki/Android_(operating_system))
- [7] JNITutorial, Marko Kovačević, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka