Matematički fakultet

Master rad

Geometrijske konstrukcije na uređajima osetljivim na dodir

Autor:
Milica Selaković

Mentor:
Prof. dr Predrag Janičić

Članovi komisije:
Prof. dr Predrag Janičić
Matematički fakultet, Univerzitet u Beogradu
Prof. dr Srđan Vukmirović
Matematički fakultet, Univerzitet u Beogradu
Prof. dr Filip Marić
Matematički fakultet, Univerzitet u Beogradu

Avgust, 2018.

Abstract

This thesis gives a description of different methods for recognizing geometric objects on raster pictures, as well as a short overview of their results. Description of point movements in different tools for dynamic geometry is also given. The thesis introduces an approach for moving of constructed points which uses tools that automatically solve construction problems.

The proposed approaches are implemented within the tool Touch&Drag. The tool enables the hand drawing of points, lines, circles, and polygons as well as their recognition in real time. In addition, the tool also recognizes the relationships between those objects. A distinguishing feature of the tool Touch&Drag is that it enables moving of constructed points in addition to free points.

Sažetak

U tezi je dat opis metoda za prepoznavanje geometrijskih figura na rasterskim slikama, kao i kratak prikaz rezultata ovih metoda. Opisano je i pomeranje tačaka u alatima za dinamičku geometiju. Predloženo je rešenje za pomeranje konstruisanih tačka koje koristi alate za automatsko rešavanje konstruktivnih problema.

Opisana rešenja implementirana su u okviru alata Touch&Drag. U alatu je omogućano crtanje rukom tačaka, pravih, krugova i poligona i njihovo prepoznavanje u realnom vremenu. Takođe, omogućeno je prepoznavanje međusobnih odnosa tih figura. Za razliku od ostalih alata za dinamičku geometriju, u alatu Touch&Drag moguće je pomerati ne samo slobodne, već i konstruisane tačke.

Sadržaj

1 Uvod **4**

2 Dinamička geometrija **7**
2.1 Geometric Supposer 8
2.2 GCLC 8
2.3 GeoGebra 9
2.4 Sketchometry 10

3 Prepoznavanje figura na rasterskim slikama **11**
3.1 Hafova transformacija 11
3.2 Diskretna krivina 13
3.3 Metod najmanjih kvadrata 14
3.3.1 Srednjekvadratna aproksimacija u Hilbertovom prostoru . 15
3.4 Poređenje tri algoritma na primerima 18

4 Pomeranje tačaka u alatima za dinamičku geometriju **21**
4.1 Lokacijski problemi konstrukcije trougla i ArgoTriCS 21
4.2 Pomeranje konstruisanih tačaka 23

5 Alat Touch&Drag **26**
5.1 Biblioteka GeometryCalculations 29
5.1.1 Implementacija algoritma za prepoznavanje figura 33
5.2 Pomeranje tačaka 36
5.3 Pregled dodatnih funkcionalnosti alata 38

6 Poređenje sa srodnim sistemima **40**
6.1 Prepoznavanje geometrijskih figura na skicama 40
6.2 Pomeranje tačaka 41

7 Zaključci i dalji rad **42**

A **46**

B **48**

Predgovor

U ovoj tezi biće opisan alat za dinamičku geometriju Touch&Drag koji sam razvijala protekle dve godine. Značajan deo rezultata ove teze je sadržan u radu Milica Selaković, Vesna Marinković, Predrag Janičić:
New Dynamics in Dynamic Geometry: Dragging Constructed Points,
koji je na recenziranju u jednom međunarodnom časopisu.

Alat Touch&Drag dostupan je na Google Play prodavnici[1] i besplatan je za korišćenje. U alatu je omogućano crtanje tačaka, pravih, krugova i poligona i njihovo prepoznavanje u realnom vremenu. Takođe, omogućeno je prepoznavanje međusobnih odnosa tih figura. Za razliku od ostalih alata za dinamičku geometriju u alatu Touch&Drag moguće je pomerati ne samo slobodne, već i konstruisane tačke. U glavi 5 je dat detaljan prikaz alata, implementacije i integracije sa sistemom ArgoTriCS.

Prepoznavanje geometrijskih figura na rasterskim slikama opisano je u glavi 3. U poglavlju 3.4 dat je kratak prikaz rezultata opisanih metoda. U okviru glave 4 opisano je pomeranje tačaka u alatima za dinamičku geometriju i sistem ArgoTriCS, koji se koristi kao sisem za rešavanje konstruktivnih problema.

Prvenstveno, izuzetnu i najveću zahvalnost, dugujem mentoru, prof. dr Predragu Janičiću, na pruženoj prilici, strpljenju, poverenju i razumevanju, na svestranoj pomoći i podršci u svim fazama izrade ovog master rada. Želela bih da se zahvalim i članovima komisije prof. dr Filipu Mariću i prof. dr Srđanu Vukmiroviću za komentare i sugestije tokom izrade rada. Veliko hvala i dr Vesni Marinković, koja mi je nesebično pomagala u tome da moj rad bude unikatan. Želela bih da se zahvalim i kolegi Aleksandru Veljkoviću, koji je osmislio izgled aplikacije. Hvala i dr Tijani Šukilović za pomoć tokom izrade aplikacije.

Na kraju, veliku zahvalnost dugujem i mojoj porodici za svu podršku i pomoć tokom studija.

[1] `https://play.google.com/store/apps/details?id=touch.drag.milica.master`

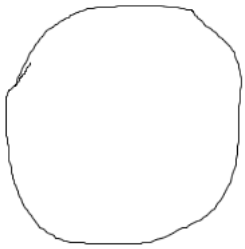
Glava 1

Uvod

Konstrukcije lenjirom i šestarom nezaobilazan su deo matematičkog obrazovanja više od dve i po hiljade godina. Ilustracije geometrijskih konstrukcija obično čine (neformalan) deo rešenja, ali često ih nije lako kreirati ,,slobodnom rukom". Tokom poslednjih decenija razvijeni su različiti programi za dinamičku geometriju, svi sa ciljem da se omogući lakše ilustrovanje konstrukcija.

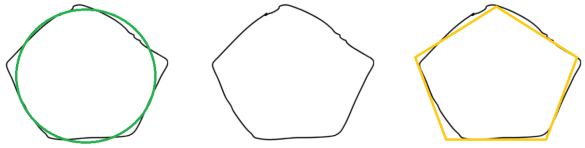
Uprkos postojanju velikog broja alata za dinamičku geometriju, predavači ih u maloj meri koriste. Razlozi ovome su nedovoljna obučenost predavača i tehnička opremljenost. Takođe, korišćenje alata u toku predavanja oduzima vreme, pa i predavači koji nisu vični crtanju radije skiciraju sliku na tabli nego što koriste neki od postojećih alata.

Uređaji osetljivi na dodir omogućavaju ilustrovanje geometrijskih konstrukcija, kao i interakciju sličnu onoj kada se koriste papir i olovka, ali imaju i brojne prednosti koje se mogu iskoristiti. Analizom skica unetih u uređaj moguće je olakšati ilustraciju konstrukcija i dati rešavanju ove grupe matematičkih problema potpuno novu dimenziju.



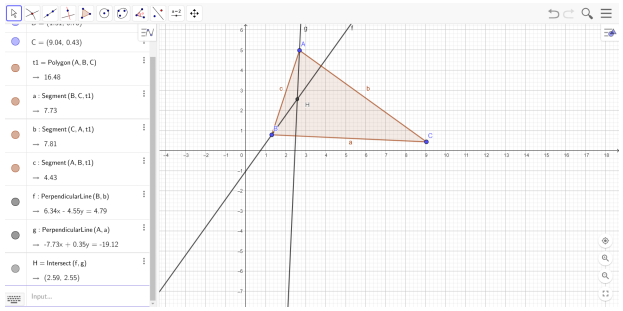
Slika 1.1: Skica kruga

Kako pogoditi šta se krije iza skice? Problem iako trivijalan čoveku, računaru uopšte nije lak. Računar može da prepozna figure na slici, koristeći, na primer, Hafovu transformaciju, o kojoj će biti više reči u daljem tekstu. Međutim, krug na slici 1.1 neće prepoznati. Za većinu algoritama za prepoznavanje figura na rasterskim slikama se očekuje da su figure na slikama neke iz malog skupa figura, na primer, prave, krugovi, poligoni. Jedan pristup rešavanju ovog problema jeste dozvoliti grešku prilikom provere odstupanja. Postavlja se pitanje koliku grešku i sa čime je treba uporediti? Ako postavimo grešku tako da na slici 1.1 bude prepoznat krug, onda će i figura sa skice sa slike 1.2 biti prepoznata kao krug.



Slika 1.2: Prepoznavanje skice na slici u sredini. Da li prepoznati figuru kao krug(levo) ili kao petougao (desno)?

U ovom radu biće opisana implementacija algoritma za prepoznavanje geometrijskih figura na osnovu skica. Ovaj algoritam je implementiran u Android aplikaciji Touch&Drag koja je glavna tema ovog rada i o kojoj će biti reči dalje u radu.



Slika 1.3: Prikaz konstrukcije ortocentra u alatu GeoGebra[4]

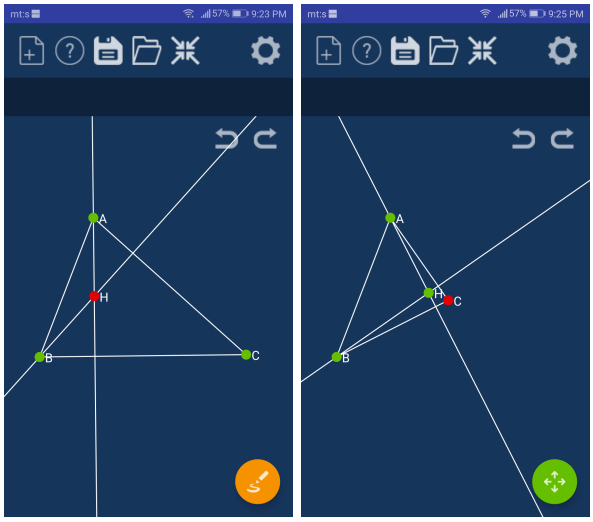
U većini alata je omogućeno pomeranje slobodnih tačaka. Slobodne figure su one čija konstrukcija ne zavisi od prethodno konstruisanih figura. Ova osobina, iako veoma prosta, veoma je efektna jer pomaže razvijanju intuicije i boljem razumevanju problema. Razmotrimo sledeći jednostavan scenario.

Primer 1 *Korisnik crta trougao ABC i potom konstruiše visine iz temena A i B i njihovu presečnu tačku H, ortocentar trougla. Nakon toga, pomerajući temena istražuje šta se dešava sa pravama koje je konstruisao i otrocentrom.*

Ovu osobinu softverskog alata nije teško implementirati. Dovoljno je pamtiti spisak konstruktivnih koraka koji se izvode i pri promeni pozicije tačaka samo

ponovo izvršiti konstruktivne korake. Ako bi, pak, korisnik poželeo da pomera konstruisanu tačku H, alat mu to ne bi dozvolio.

U ovom radu biće prikazan novi pristup za pomeranje tačaka koji omogućava i pomeranje konstuisanih tačaka. Ova funkcionalnost ne postoji ni u jednom alatu do sada. U alatu Touch&Drag je implementirano pomeranje konstruisanih tačaka na osnovu ideja iz ovog rada.



Slika 1.4: Prikaz pomeranja konstruisane tačke H u alatu Touch&Drag

Glava 2

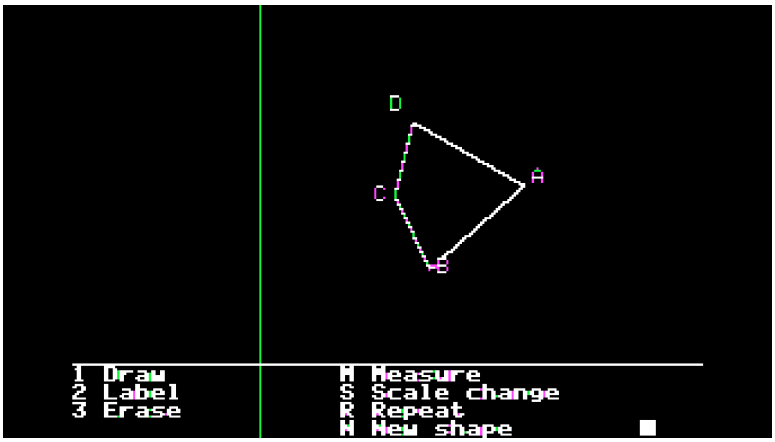
Dinamička geometrija

Prvi alat za dinamičku geometriju, Geometric Supposer[2] nastao je početkom 80tih godina XX veka. Zatim su nastali i alati Cabri, Geometer's Sketchpad i mnogi drugi. Ovi alati su tokom godina pronašli put do učionica širom sveta i postali nezamenljivi deo matematičkog obrazovanja. Najpopularniji geometrijski alat GeoGebra ima više od deset miliona korisnika. Ovi alati su se tokom godina razvijali i dobijali razne dodatne funkcije uključujući i automatsko dokazivanje geometrijskih teorema.

U alatima za dinamičku geometriju, geometrijske figure obično su predstavljene kroz Dekartov ravanski model euklidske ravni. Površ (platno), na kojoj se nalazi crtež, odgovara pravougaoniku u Dekartovoj ravni. Cela konstrukcija se pamti u obliku spiska konstruktivnih koraka lenjirom i šestarom. Pored osnovnih konstruktivnih koraka lenjirom i šestarom, većina alata daje opciju i naprednijih konstrukcija, kao što je konstrukcija središta duži, upravne prave, paralelne prave, itd.

Pomeranje slobodnih figura je obično podržano u ovim alatima. Pri promeni neke figure dovoljno je samo izvršiti ponovo konstruktivne korake i slika će se izmeniti. Pomeranje slobodnih figura pomaže u istraživanju veza među figurama. Šta će se desiti sa konstruisanim orotocentrom trougla ako se pomeri jedno teme? Ili pak, kada dva kruga više neće imati presek?

Kortenkamp u svojoj doktorskoj disertaciji[1] uvodi determinizam, konzervativizam i neprekidnost kao osobine alata za dinamičku geometriju. Alat je deterministički ukoliko u alatu za isti početni skup figura, izvodeći iste konstruktivne korake, dobijamo iste figure. Većina alata za dinamičku geomeriju je deterministička. Konzervativizam je direkna posledica determinizma, i u determinističkim alatima razlika između ove dve osobine se teško primećuje. Konzervativizam se ogleda u tome da kada pomerimo figure i potom poništimo promene, vraćajući se unazad, dobijemo iste figure kao i pre promena. Ova osobina je prisutna u svim determinističim alatima, ali se može očekivati i kod nedeterminističkih alata. Treće, i kako Kortenkamp navodi najizazovnije, svojstvo alata za dinamičku geometriju je neprekidnost. Neprekidan alat jeste onaj u kome ne dolazi do velikih ,,skokova" izazvanih malim promenama parametara.



Slika 2.1: Geomeric Supposer: konstrukcija četvorougla

U nastavku će biti ukratko opisani neki od alata za dinamičku geometriju.

2.1 Geometric Supposer

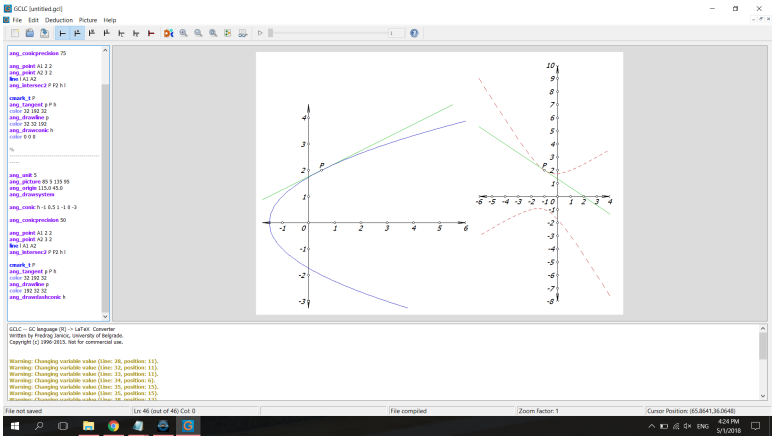
Geometric Supposer [2] je alat za dinamičku geometriju koji je napravio najveći pomak u matematičkom obrazovanju i omogućio razvoj ostalih alata. Pisanjem manjih programa korisnik crta figure koje je moguće konstruisati korišćenjem lenjira i šestara. Moguće je nacrtati četvorouglove, duži, paralelne prave, normale, simetrale uglova, opisane i upisane krugove. Takođe, korisnik može da meri rastojanje, uglove i površine, kao i da računa aritmetičke kombinacije navedenih mera. Ovaj alat ima i osobinu da zapamti konstrukciju i iskoristi je iznova.

2.2 GCLC

GCLC [3] (od "Geometry Constructions → LaTeXConverter") je alat za vizuelizaciju i pravljenje ilustracija. Glavne svrhe su mu:

1. pravljenje matematičkih ilustracija visokog kvaliteta;
2. upotreba u nastavi matematike (posebno geometrije);
3. kao alatka u geometrijskom rezonovanju.

Osnovna ideja koja stoji iza sistema GCLC je da geometrijske konstrukcije nisu slike, nego formalne procedure. Zbog toga, u GCLC-u, pravljenje matematičkih ilustracija zasnovano je na njihovom opisivanju, a ne crtanju. U opisivanju ilustracija može se koristiti veliki broj osnovnih i složenih konstrukcija i izometrijskih transformacija, krive, površi, simbolički izrazi, moduli za crtanje stabla i grafova, kontrola toka, korisnički definisane funkcije, itd.



Slika 2.2: GCLC - prikaz krivih drugog reda

GCLC ima ugrađena i tri automatska dokazivača (jedan zasnovan na metodi površina, drugi na Vuovoj metodi i treći na Grebnerovim bazama) koji mogu da dokažu veliki broj kompleksnih geometrijskih teorema.

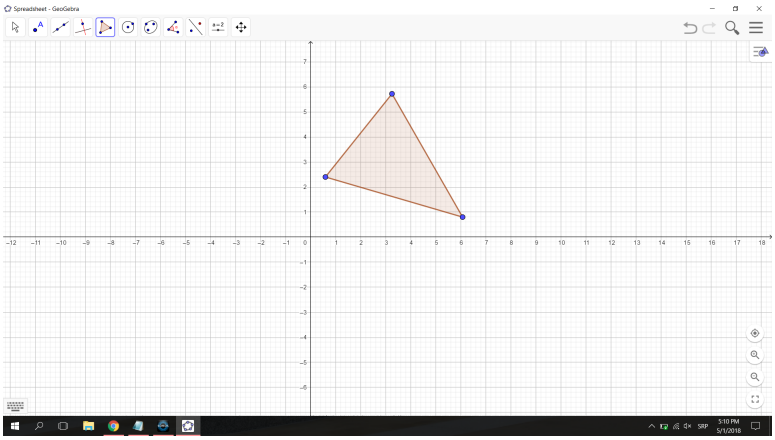
2.3 GeoGebra

GeoGebra je interaktivni matematički sistem razvijan sa namenom za lakše savladavanje nauke (posebno matematike) od nivoa osnovne škole pa do univerzitetskog nivoa. Alat ima podršku za veliki broj osnovih i složenih konstrukcija, kao što su: tačke, vektori, duži, konstrukcija prave kroz tačku, crtanje grafika zadate funkcije i mnoge druge. Sve nastale figure (tačke, prave, poligoni, konike, itd.) mogu se menjati dinamički. GeoGebra je i deterministički i neprekidni alat. Korisnik može uneti i modifikovati elemente putem panela za konstrukciju i panela za unos.

GeoGebra nije samo alat za dinamičku geometriju. Alat ima podršku za algebru i matematičku analizu, pa je tako u alatu moguće izračunati integral ili izvod zadate funkcije.

Glavne osobine ovoga alata su:

- Interaktivno okruženje (2D i 3D);
- Ugrađeni tabelarni prikaz;
- Podrška za simbolička i algebarska izračunavanja;
- Podrška za matematičku analizu i statistiku;
- Mogućnost pisanja skripti;
- Veliki broj materijala za učenje na *GeoGebra Materials*.

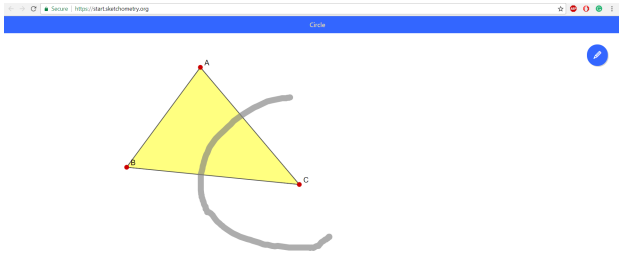


Slika 2.3: Prikaz trougla u alatu GeoGebra

2.4 Sketchometry

Sketchometry je matematički alat namenjen za ilustrovanje geometrijskih konstrukcija u euklidskoj ravni i crtanje grafika funkcija. Korisnik slobodnom rukom skicira svoje konstrukcije, a alat ih potom pretvara u geometrijske konstrukcije. Nastale figure se potom mogu pomerati i modifikovati. Ovaj alat je namenjen za jednostavan rad u nastavi i prevashodno je namenjen uređajima osetljivim na dodir, iako postoji i verzija za internet pregledač. Na ovaj način mogu se konstruisati tačke, krugovi, prave, poligoni, paralelne prave, presečne tačke, itd.

Ideja da se skice prenose u geometrijske konstrukcije daje nove mogućnosti u nastavi geometrije. Crtanjem slobodnom rukom ubrzava se proces crtanja, a dobija se ispravna geometrijska konstrukcija na kojoj se sve jasno vidi.



Slika 2.4: Sketchometry

Glava 3

Prepoznavanje figura na rasterskim slikama

Problem prepoznavanja geometrijskih oblika kao što su prave, krugovi, kružni isečci, elipse, itd. na rasterizovanim slikama je centralni problem u prepoznavanju obrazaca, računarskom vidu, robotici i mnogim drugim oblastima. Zbog značaja ove oblasti razvijeni su raznovrsni algoritmi za prepoznavanje figura, a među najznačajnijim su Hafova transformacija, Radonova transformacija, analiza krivine i srednjekvadratna aproksimacija funkcija.

Mašinsko učenje je jedan od mogućih pristupa ovom problemu. U radu [5] je prikazano poređenje rezultata prepoznavanja figura na slikama primenom algoritama mašinskog učenja. Međutim, iako su rezultati koji daju ovi aloritmi veoma dobri, u ovom master radu bavićemo se drugim metodama.

3.1 Hafova transformacija

Hafova transformacija je metod za prepoznavanje krivih koja koristi dualnost između tačaka na krivoj i parametara te krive. Klasična Hafova transformacija identifikovala je prave na slici, ali kasnije je proširena na identifikaciju proizvoljnih krivih, najčešće krugova ili elipsi. Hafovu transformaciju, na koju danas referišemo, osmislili su Ričard Djuda i Piter Hart 1972. [6] i nazvali je generalizovana Hafova transformacija po srodnom patentu iz 1962. Pola Hafa [7]. U nastavku ćemo opisati kako funkcioniše klasična Hafova transformacija prepoznavanja pravih na slici. Razmatraćemo rasterizovanu sliku sa belom pozadinom i iscrtanom crnom pravom.

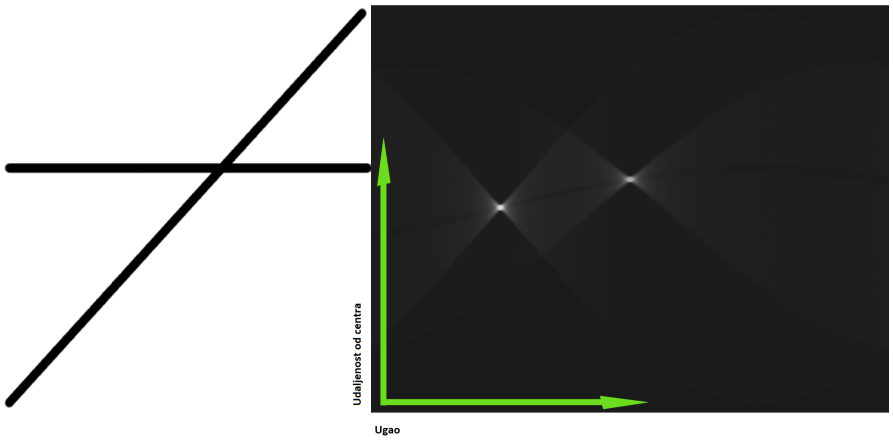
Za svaki crni piksel na slici postoji beskonačno mnogo pravih koje ga sadrže, jedna za svaki mogući ugao. Svaka od ovih pravih, osim vertikalne, se može prikazati u eksplicitnoj formi:

$$y = ax + b,$$

gde su (x, y) koordinate piksela kroz koji prava prolazi. Ako ovu jednačinu razmatramo drugačije, tako da su x i y konstante, a a i b koordinate, jednačina se može zapisati u obliku:

$$b = -xa + y,$$

što predstavlja pravu u prostoru parametara a i b. Dakle, svaka tačka odgovara pravoj u parametarskom domenu. I obratno, svaka tačka u parametarskom domenu odgovara jednoj pravoj u prostoru slike. Primetimo da tačke preseka pravih u parametarskom domenu odgovaraju kolinearnim pikselima (tačkama) u prostoru slike.



Slika 3.1: Na slici desno je prikazan Hafov prostor parametara slike levo

Korišćenje prethodnog eksplicitnog oblika reprezentacije prave dovodi do problema kada se na slici nalaze vertikalne prave (tj. prave oblika $y = c$). Rešenje navedenog problema dali su Ričard Djuda i Piter Hart. Oni su predložili korišćenje Heseove normalne forme umesto gore navedene:

$$\rho = x \sin\theta + y \cos\theta. \tag{3.1}$$

U ovom slučaju se svaka prava identifikuje sa parom (ρ, θ) u parametarskom prostoru. Ovim unapređenjem nestaje problem vertikalnih pravih.

Hafovom transformacijom konvertujemo tačke u prave u parametarskom prostoru. Ako postoje kolinearne tačke na slici, onda će se sve odgovarajuće prave seći u jednoj tački. Određivanje da li ima preseka u prostoru parametara izvodi se procedurom ,,glasanja".

Kako procedura zapravo funkcioniše? Prvo kreiramo dvodimenzionalni niz (akumulator) i postavimo svako polje inicijalno na 0. Neka vrste označavaju ρ, a kolone θ. Veličina niza zavisi od tačnosti koja nam je potrebna. Na primer, ako za ugao hoćemo tačnost 1 biće nam potrebno 180 kolona. Najveća vrednost za ρ je dijagonala slike, pa ako hoćemo tačnost 1 piksela, broj vrsta će biti dužina

dijagonale. Prolazimo sada kroz svaki piksel. Kada naiđemo na beli piksel njega preskačemo, jer on ne utiče na glasanje.

Razmotrimo situaciju kada naiđemo na crni piksel. Poznate su nam koordinate (x, y) piksela. U jednačinu 3.1 uvrstimo vrednosti $\theta = 0, 1, 2, ..., 180$ i izračunamo odgovarajuće ρ. Za svaki dobijeni par (ρ, θ) uvećamo vrednost na poziciji (ρ, θ) u akumulatoru za 1. Po završenom obilasku svih piksela pronađemo poziciju (ρ, θ) maksimalne vrednosti u akumulatoru. Prava koja se nalazi na slici ima jednačinu $\rho = x \sin\theta + y \cos\theta$.

Kako je prostor parametara diskretan, javljaju se sledeći nedostaci:

- Detektovanje debljine linije („bin-splitting" problem);
- Detektovanje duži;
- Problem ocene parametara;
- Brzina i potrebna memorija.

Usavršavanjem ovih nedostataka nastale su brza Hafova transformacija [8], modifikacija povezanosti i debljine linije [9], robusna Hafova transformacija [10], generalizacija Hafove transformacije za proizvoljne oblike [11].

3.2 Diskretna krivina

Kako je svaka ravanska kriva jedinstveno određena svojom krivinom do na translaciju, to se prepoznavanje krive na slici može svesti na problem određivanja njene krivine[12]. Uvodeći pojam diskretne krivine i korišćenjem mere sličnosti među objektima predložen je novi metod detekcije i klasifikacije krivih na rasterizovanim slikama.

Definicija 1 *Regularna (parametrizovana) ravanska kriva je preslikavanje* $\alpha : (a, b) \to \mathbb{R}^2$*, klase* C^k*, za neko* $k \geq 1$*, tako da je* $\frac{d\alpha}{dt} \neq 0$*, za sve* $t \in (a, b)$*.*

Razmatrajmo regularnu ravansku krivu $\alpha : (a, b) \to \mathbb{R}^2$, parametrizovanu dužinom luka krive s, tj. rastojanjem duž krive merene od neke fiksne tačke c:

$$s(t) = \int_c^t \alpha'(u)du, \quad c \leq t \leq b.$$

Tada je $\alpha'(s)$ njen tangentni vektor $T(s)$ i $\alpha''(s)$ je kolinearan jediničnom vektoru normale $N(s)$. Tada imamo

$$\alpha''(s) = k(s)N(s). \tag{3.2}$$

Definicija 2 *Funkciju* $k(s)$ *nazivamo krivina krive* $\alpha(s)$ *u tački* s*.*

Definicija 3 *Diskretna ravanska kriva je preslikavanje* $\alpha : A \to \mathbb{R}^2$*, gde je* A *diskretan podskup* (a, b)*.*

Neka je $P = \alpha(t), t \in (a, b)$, tačka na diskretnoj krivoj i neka su $Q_i = \alpha(s_i), i = 1, \ldots, n$ tačke na krivoj u okolini tačke P. Ako P nije na kraju krive, pretpostavljamo da su tačke Q_i ravnomerno raspoređene. Odnosno, pretpostavljamo da je broj tačaka sa jedne stane tačke P približno jednak broju tačaka sa druge strane tačke P.

Kriva α može biti razvijena u Tejlorov red u okolini tačke P

$$\alpha(s) = \alpha(t) + (s-t)\alpha'(t) + \frac{(s-t)^2}{2}\alpha''(t) + o((s-t)^2). \tag{3.3}$$

Koristeći 3.2 i 3.3 za svako Q_i dobijamo

$$\overrightarrow{PQ_i} = \alpha(s_i) - \alpha(t) \approx (s_i - t)T(t) + \frac{(s_i - t)^2}{2}k(P)N(t).$$

Kako su Q_i balansirani, parametri $(s_i - t)$ se poništavaju i sumirajući po svim i dobijamo

$$\sum_{i=1}^{n} \overrightarrow{PQ_i} \approx \frac{k(P)N(t)}{2} \sum_{i=1}^{n} (s_i - t)^2 = \frac{k(P)N(t)}{2} \sum_{i=1}^{n} ||\overrightarrow{PQ_i}||^2. \tag{3.4}$$

Iz jednačine 3.4 dobijamo aproksimaciju krivine u tački P krive α:

$$k(P) \approx \frac{2||\sum_{i=1}^{n} \overrightarrow{PQ_i}||}{\sum_{i=1}^{n} ||\overrightarrow{PQ_i}||^2}. \tag{3.5}$$

Potom se prepoznavanje vrši na sledeći način:

- Duž krive, za svaku tačku P_i izračunamo k_i, diskretnu krivinu u zadatoj tački.
- Kao što je rečeno na početku, svaka kriva je jedinstveno određena svojom krivinom. Ostaje da na osnovu diskretnog skupa vrednosti funkcije krivine $\{k_i | 1 \leq i \leq n\}$ proverimo koja je to funkcija. U radu [12] je predložena sledeća mera udaljenosti dva skupa

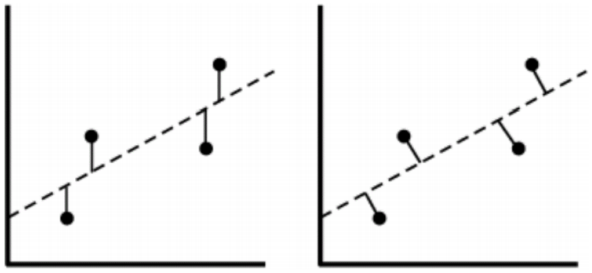
$$D(P, Q) = \frac{1}{2}\left(\frac{1}{N_P}\sum_{p \in P} \min_{q \in Q} d(p, q) + \frac{1}{N_Q}\sum_{q \in Q} \min_{p \in P} d(p, q)\right). \tag{3.6}$$

  Gde je d L_2 metrika. Ovom merom se proverava udaljenost dva skupa tako što se pronađe prosek rastojanja tačke i njoj najbliže iz drugog skupa.

3.3 Metod najmanjih kvadrata

Metod najmanjih kvadrata ima širu primenu nego prethodna dva metoda. Ovaj metod se koristi u regresionoj analizi za rešavanje preodređenih sistema. Metod traži najbolje rešenje tako da je suma kvadrata reziduala (razlika izračunate vrednosti i originalne vrednosti) najmanja, odakle je i ime metoda.

Primer 2 *Za zadati skup tačaka pronaći pravu koja najbolje srednjekvadratno aproksimira zadati skup tačaka* $\{(x_i, y_i | i = 1, \ldots, n\}$.



Slika 3.2: Metod najmanjih kvadrata minimizuje sumu kvadrata rastojanja na levoj, a ne na desnoj slici.

3.3.1 Srednjekvadratna aproksimacija u Hilbertovom prostoru

Definicija 4 *Vektorski prostor koji je kompletan (svaki Košijev niz elemenata vektorskog prostora konvergira elementu tog prostora) u odnosu na metriku indukovanu skalarnim proizvodom:*

$$d(x, y) = ||x - y|| = \sqrt{(x - y)^2}$$

se naziva Hilbertovim prostorom.

Definicija 5 *Za sistem vektora* $\{e_i | i \in \mathbb{N}\}$*, kažemo da je ortonormiran ukoliko važi*

$$e_i \cdot e_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$$

Sledeće teoreme navodimo bez dokaza, dokazi se mogu naći u [13].

Teorema 1 *Za ortonormiran sistem* $\{e_i | i \in \mathbb{N}\}$ *u Hilbertovom prostoru* $\mathcal{H}$*, sledeća tvrđena su ekvivalentna:*

- *Za svako* $x \in \mathcal{H}$ *i svako* $\epsilon > 0$*, postoje skalari* $\lambda_1, \lambda_2, \ldots, \lambda_n$ *takvi da važi* $||x - \sum_{i=1}^{n} \lambda_i e_i|| < \epsilon$*;*
- *Za svako* $x \in \mathcal{H}$ *važi* $\sum_{i=1}^{\infty} (x \cdot e_i) e_i = x$*, pri čemu se podrazumeva konvergencija u smislu metrike prostora* $\mathcal{H}$*;*

- *Za svako* $x \in \mathcal{H}$ *važi* $\sum_{i=1}^{\infty}(x \cdot e_i)^2 = ||x||^2$*;*
- *Ako je vektor* $x \in \mathcal{H}$ *takav da je* $x \cdot e_i = 0$ *za svako* $i \in \mathbb{N}$*, onda važi* $x = 0$*.*

Teorema 2 *Neka je* f *element Hilbertovog prostora* $\mathcal{H}$ *i neka je* $\mathcal{H}'$ *njegov potprostor čiju bazu čine elementi* $\{g_1, g_2, \ldots, g_n\}$*. Postoji element najbolje aproksimacije* $g^* = \sum_{i=1}^{n} c_i^* g_i \in \mathcal{H}'$*, takav da važi*

$$\left|\left| f - g^* \right|\right| = \inf_{c_1,\ldots,c_n} \left|\left| f - \sum_{i=1}^{n} c_i g_i \right|\right|.$$

Dodatno, važi da je $(f - g^*) \cdot x = 0$ *za sve* $x \in \mathcal{H}'$ *akko je* g^* *element najbolje aproksimacije za* f *iz* $\mathcal{H}'$*.*

Na osnovu ove teoreme sledi da se koeficijenti najbolje aproksimacije mogu odrediti iz sistema:

$$\sum_{i=1}^{n} c_i(g_i \cdot g_j) = f \cdot g_j, \quad j = 1, \ldots, n. \tag{3.7}$$

Ako se za Hilbertov prostor $\mathcal{H}$ uzme prostor funkcija $\mathcal{L}_2[a, b]$, odnosno prostor funkcija integrabilnih sa kvadratom na intervalu $[a, b]$, u kome je norma definisana integralom

$$||f||^2 = \int_a^b f^2(x)dx \quad f \in \mathcal{L}_2[a, b]$$

onda se element najbolje aproksimacije naziva elementom *najbolje srednjekvadratne aproksimacije*.

Za funkcije zadate na diskretnom skupu tačaka, metoda koja odgovara srednjekvadratnoj aproksimaciji je ***metod najmanjih kvadrata***. Integral se zamenjuje sumom, te su skalarni proizvod i odgovarajuća norma dati jednakostima:

$$f \cdot g = \sum_{i=1}^{m} f(x_i)g(x_i)$$

$$||f||^2 = f \cdot f = \sum_{i=1}^{m} f^2(x_i)$$

Jednačine 3.7 za ovaj konkretan izbor skalarnog proizvoda dobijaju sledeći oblik:

$$\sum_{i=1}^{n} c_i \sum_{k=1}^{m} g_i(x_k)g_j(x_k) = \sum_{k=1}^{m} f(x_k)g_j(x_k), \quad j = 1, \ldots, n.$$

Razmenom redosleda sumiranja, dobija se:

$$\sum_{k=1}^{m} g_i(x_k)(\sum_{i=1}^{n} c_i g_j(x_k)) = \sum_{k=1}^{m} f(x_k)g_j(x_k), \quad j = 1, \ldots, n.$$

ili zapisano u matričnoj notacji

$$\begin{bmatrix} g_1(x_1) & \dots & g_1(x_m) \\ \vdots & \ddots & \vdots \\ g_n(x_1) & \dots & g_n(x_m) \end{bmatrix} \begin{bmatrix} g_1(x_1) & \dots & g_n(x_1) \\ \vdots & \ddots & \vdots \\ g_1(x_m) & \dots & g_n(x_m) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} g_1(x_1) & \dots & g_1(x_m) \\ \vdots & \ddots & \vdots \\ g_n(x_1) & \dots & g_n(x_m) \end{bmatrix} \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix}. \tag{3.8}$$

Uvodeći sledeće oznake u 3.8

$$A = \begin{bmatrix} g_1(x_1) & \dots & g_n(x_1) \\ \vdots & \ddots & \vdots \\ g_1(x_m) & \dots & g_n(x_m) \end{bmatrix} \quad x = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} \quad b = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix}$$

dobijamo se jednačine normale:

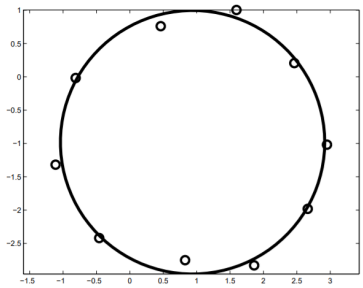
$$A^T A x = A^T b,$$

odnosno

$$x = (A^T A)^{-1} A^T b, \tag{3.9}$$

koje predstavljaju rešenje optimizacionog problema

$$\min_x ||Ax - b||^2.$$



Slika 3.3: Srednjekvadratna aproksimacija tačaka krugom

Primer 3 *Za zadati skup tačaka* $\{(x_i, y_i) | i = 1, \dots, n\}$ *pronaći krug koji ga najbolje srednjekvadratno aproksimira.*

Razmatrajmo jednačinu kruga sa centrom u tački (a, b) *i poluprečnikom* r*:*

$$(x - a)^2 + (y - b)^2 = r^2.$$

Ukoliko želimo da krug sadrži sve navedene tačke (x_i, y_i) dobijamo preodređeni sistem jednačina, koji nije linearan po nepoznatim veličinama a,b i r. Međutim, prezapisivanjem jednačina sistema u obliku

$$x^2 - 2ax + a^2 + y^2 - 2by + b^2 = r^2,$$

odnosno

$$2ax + 2by + r^2 - a^2 - b^2 = x^2 + y^2,$$

i uvođenjem smene $c = r^2 - a^2 - b^2$, dobijamo preodređeni linearni sistem po nepoznatim a, b i c,

$$2x_i a + 2y_i b + c = x_i^2 + y_i^2, \quad i = 1, \ldots, n.$$

Uvodeći oznake

$$A = \begin{bmatrix} 2x_1 & 2y_1 & 1 \\ \vdots & \vdots & \vdots \\ 2x_n & 2y_n & 1 \end{bmatrix} \quad x = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad b = \begin{bmatrix} x_1^2 + y_1^2 \\ \vdots \\ x_n^2 + y_n^2 \end{bmatrix}$$

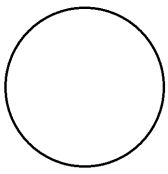
dobijamo

$$Ax = b.$$

Rešenje ovog problema dato je formulom 3.9.

3.4 Poređenje tri algoritma na primerima

U ovom odeljku razmotrićemo tri primera na kojima ćemo uporediti rezultat prepoznavanja, grešku i vreme izvršavanja tri ranije opisana algoritma. U prvom primeru ulaz će biti krug, u drugom poligon koji ima 180 temena i koji liči na krug, a na kraju razmotrićemo primer u kome je ulaz šestougao. Metod najmanjih kvadrata i algoritam zasnovan na diskretnoj krivini su implementirani u MATLAB-u[14], dok su za Hafovu transformaciju korišćene MATLAB funkcije *houghlines* za detekciju duži na rasterskoj slici i *imfindcircles* za detekciju krugova.



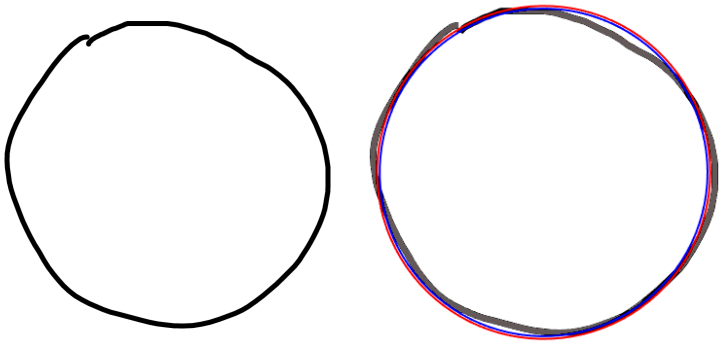
Slika 3.4: Ulaz u prvom primeru

Rezultati rada algoritama u primeru kada je ulaz krug, prikazan na slici 3.4, prikazani su u sledećoj tabeli[1]:

[1]Dobijene vrednosti su izmerene na računaru sa procesorom Intel i5-52000U i 12 GB RAM memorije.

	prepoznata figura	greška	vreme izvršavanja (u sekundama)
Hafova transformacija	krug	nije poznato	2,015
Diskretna krivina	krug	0	0,000409
Metod najmanjih kvadrata	krug	0	0,000124

Dužina izvršavanja algoritama se veoma razlikuje, Hafova transformacija je znatno sporija od druga dva algoritma. Jedan od razloga je taj što je ulaz za Hafovu transformaciju rasterska slika, dok su za druga dva vektori tačaka sa kruga.



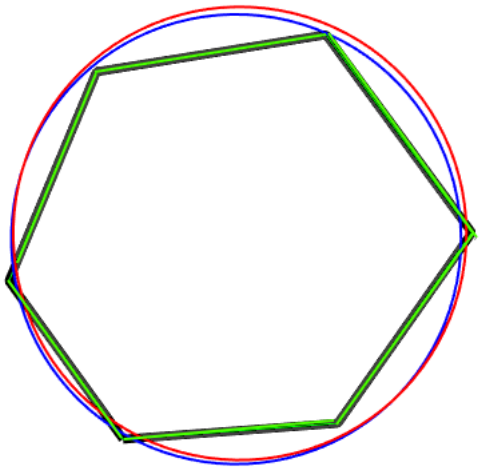
Slika 3.5: Ulaz (levo) i rezultati prepoznavanja (desno) u drugom primeru. Plavom linijom je prikazan krug prepoznat metodom najmanjih kvadrata, dok je crvenom prikazan krug prepoznat algoritmom zasnovanom na diskretnoj krivini.

Na slici 3.5 je prikazan poligon sa 180 temena koji veoma liči na krug. Rezultati rada algoritama u ovom slučaju su prikazani u sledećoj tabeli:

	prepoznata figura	greška	vreme izvršavanja (u sekundama)
Hafova transformacija	ništa nije prepoznato	nije poznato	2,50724
Diskretna krivina	krug	$3{,}9{\cdot}10^{-7}$	0,00124
Metod najmanjih kvadrata	krug	7,9354	0,00011

Hafova transformacija ne daje dobre rezultate kada je na slici figura koja liči na krug. Greška algoritma zasnovanog na diskretnoj krivini računata je formulom 3.6, dok je greška za metod najmanjih kvadrata računata kao prosek rastojanja tačaka od aproksimiranog kruga i izražena je u pikselima.

U poslednjem primeru razmatraćemo šest izabranih tačaka iz prethodnog primera kako bismo uporedili razliku u greškama između metoda najmanjih



Slika 3.6: Ulaz sa rezultatima prepoznavanja u trećem primeru. Zelenom linijom je prikazan rezultat prepoznavanja Hafove transformacije, crvenom rezultat prepoznavanja algoritma zasnovanog na diskretnoj krivini i plavom rezultat prepoznavanja metode najmanjih kvadrata.

kvadrata i algoritma zasnovanog na diskretnoj krivini. Rezultati rada algoritama u slučaju kada je ulaz šestougao dati su u sledećoj tabeli:

	prepoznata figura	greška	vreme izvršavanja (u sekundama)
Hafova transformacija	šest duži	nije poznato	1,1002
Diskretna krivina	krug	$1{,}6114 \cdot 10^{-5}$	0,0000156
Metod najmanjih kvadrata	krug	7,9317	0,0000938

Mala promena greške u metodu najmanjih kvadrata je očekivana jer su temena odabranog šestougla na krugu (slika 3.6). Promena greške kod algoritma zasnovanog na diskretnoj krivini je takođe mala. Ovo dovodi do toga da je teško postaviti granicu između figura koje bi trebalo da budu prepoznate kao krug i onih koje koje bi trebalo da budu prepoznate kao poligon.

Glava 4

Pomeranje tačaka u alatima za dinamičku geometriju

U većini geometrijskih alata korisnik koristeći neke slobodne tačke konstruiše druge tačke i figure. Potom korisnik pomerajući slobodne tačke istražuje promene koje se dešavaju nad konstruisanim figurama. Ovaj scenario, iako jednostavan, veoma je efektan, posebno u matematičkom obrazovanju. Pomerajući polazne tačke i posmatrajući promene đaci razvijaju intuiciju i dolaze do zaključaka o odnosima koji važe među geometrijskim figurama.

Osim opisanog standardnog scenarija u alatu Touch&Drag moguće je pomerati i konstruisane tačke. Ovo, za razliku od standardnog scenarija, nije lako izvesti. Da bi se pomerila konstruisana tačka potrebno je rešiti (uglavnom kompleksan) konstruktivan problem. Razmotrimo sledeći jednostavan primer.

Primer 4 *Korisnik bira tri slobodne tačke A, B i C i onda, koristeći normale i preseke, konstruiše ortocentar H trougla ABC. Ako bi korisnik pomerio, na primer tačku A, konstrukcija tačke H bi se ponovo izvela sa novim tačkama A, B i C i dobili bi novu tačku H. Ako bi, pak, korisnik želeo da pomeri tačku H, zadržavajući pozicije B i C i dozvoljavajući A da se pomeri, bilo bi potrebno rešiti sledeći konstruktivni problem: za zadate tačke B, C i H konstruisati tačku A takvu da je H ortocentar trougla ABC.*

4.1 Lokacijski problemi konstrukcije trougla i ArgoTriCS

Lokacijski problemi konstrukcije trougla su konstruktivni problemi čiji je cilj, korišćenjem lenjira i šestara, konstruisati trougao na osnovu zadatih pozicija tri karakteristične tačke. Viljem Vernik [20] je 1982. godine predstavio listu lokacijskih problema konstrukcije trougla gde su karakteristične tačke iz sledećeg skupa tačaka:

- A, B, C, O: temena i centar opisanog kruga;
- M_a, M_b, M_c, G: središta stranica trougla i težište;
- H_a, H_b, H_c, H: podnožja visina i ortocentar;
- T_a, T_b, T_c, I: podnožja bisektrisa unutrašnjih uglova u trouglu i centar upisanog kruga.

Harold Koneli je razmatrao proširen skup tačaka [21], koji uključuje četiri dodatne tačke:

- E_a, E_b, E_c: tri Ojlerove tačke, koje predstavljaju središta duži određenih temenima A, B i C i ortocentrom H;
- N: centar Ojlerovog kruga trougla ABC (to je krug koji sadrži devet značajnih tačaka trougla: tri središta stranica, tri podnožja visina i tri Ojlerove tačke).

Nadalje sve ove tačke zvaćemo Konelijeve tačke. Nema svaki problem, određen Konelijevim tačkama, jedinstveno rešenje. Neke od trojki tačaka su redundatne (jedna se može konstruisati na osnovu druge dve), dok za neke postojanje rešenja zavisi od pozicija tačaka. Postoje i trojke tačaka na osnovu kojih se ne može rekonstruisati trougao, jer on ne postoji ili se ne može konstruisati korišćenjem lenjira i šestara.

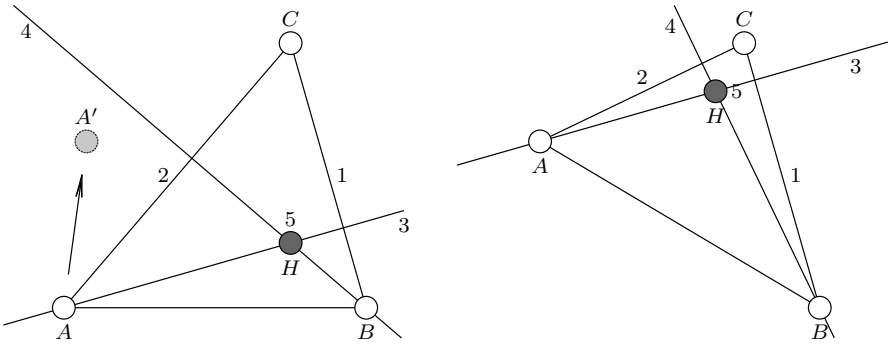
ArgoTriCS [17, 18, 19] je alat kojim se, na osnovu nekog raspoloživog geometrijskog znanja rešava lokacijski problem konstrukcije trougla. Na osnovu raspoloživog znanja, alat zaključuje da li je problem redundatan, rešiv/nerešiv ili postojanje rešenja zavisi od pozicije tačaka. Nama su interesnatni samo oni koji su rešivi za zadate pozicije karakterističnih tačaka. ArgoTriCS za svaku trojku Konelijevih tačaka koja određuje rešiv problem, generiše spisak osnovnih konstruktivnih koraka (dodatak A) i uslove pod kojima rešenje postoji.

Primer 5 *Za problem: za zadate tačke A, B i H konstruisati trougao ABC tako da je tačka H ortocentar traženog trougla, ArgoTriCS automatski generiše sledeće rešenje:*

1. *Konstruisati pravu h_a kroz tačke A i H (pravilo W02);*
2. *Konstruisati pravu h_b kroz tačke B i H (pravilo W02);*
3. *Konstruisati pravu b tako da sadrži tačku A i upravna je na pravu h_b (pravilo W10);*
4. *Konstruisati pravu a tako da sadrži tačku B i upravna je na pravu h_a (pravilo W10);*
5. *Konstruisati presečnu tačku C pravih a i b (pravilo W03).*

   Uslovi određenosti: prave a i b nisu iste; tačke B i H nisu iste; tačke A i H nisu iste.

   Uslovi nedegenerisanosti: prave a i b nisu paralelne.



Slika 4.1: Pomeranje slobodne tačke A i rekonstrukcija konstruisane tačke H

4.2 Pomeranje konstruisanih tačaka

Razmotrimo jednostavan primer pomeranja slobodne tačke.

Primer 6 *Pretpostavimo da korisnik bira slobodne tačke A, B i C (temena trougla) i da potom izvodi sledeću konstrukciju:*

1. *Konstruisati pravu a kroz tačke B i C;*
2. *Konstruisati pravu b kroz tačke A i C;*
3. *Konstruisati pravu h_a tako da sadrži tačku A i upravna je na pravu a;*
4. *Konstruisati pravu h_b tako da sadrži tačku B i upravna je na pravu b;*
5. *Konstruisati presečnu tačku H pravih h_a i h_b.*

Ako se tačka A pomeri, ostavljajući tačke B i C na istim pozicijama, cela konstrukcija se ponovo izvodi i dobija se nova pozicija tačke H (slika 4.1).

Primetimo da je pomeranje slobodne tačke dosta lako za implemetaciju. Ako bi, pak, želeli da pomerimo konstruisanu tačku, problem bi bio potpuno drugačiji. Pomeranjem konstruisane tačke možemo istražiti šta se dešava sa slobodim tačkama ako se konstruisane nalaze na zadatim pozicijama. Ili, specijalno, šta se dešava sa temenima trougla ako se konstruisane tačke nalaze na zadatim pozicijama. Razmatraćemo poslednje pitanje sa posebnom pažnjom na tri bitna problema.

Koje su tačke fiksne? U standarnom scenariju pomeranja tačaka podrazumevano je da su sve slobodne tačke fiksne, osim one koja se pomera. Međutim, u slučaju pomeranja konstruisanih tačaka tu pretpostavku ne možemo da napravimo, moramo odabrati koje su tačke fiksne. Naime, pozicije svih tačaka (slobodnih i konstruisanih) su ograničene konstrukcijom, pa samo neke mogu biti označene kao fiksne. Razmatraćemo scenario u kome se pomera jedna tačka trougla, dok su neke druge dve izabrane tačke fiksne.

Koja je karakteristična tačka odabrana konstruisana tačka? Ne možemo pomerati konstruisanu tačku bez nekog prethodnog znanja o njoj. Na primer, ako pomeramo ortocentar nekog trougla, mi možemo iskoristiti podatak da je konstruisana tačka koja se pomera ortocentar trougla. U standardnom scenariju ovo pitanje takođe ima trivijalan odgovor - nemamo nikakvo znanje o tačkama koje pomeramo, ali nam to znanje i ne treba. Odgovor na pitanje možemo dobiti na više načina, najjednostavniji bi bio upoređivanje koordinata tačke sa svim karakterističnim tačkama u trouglu. Bolji pristup bi bio analiza konstrukcije karakteristične tačke.

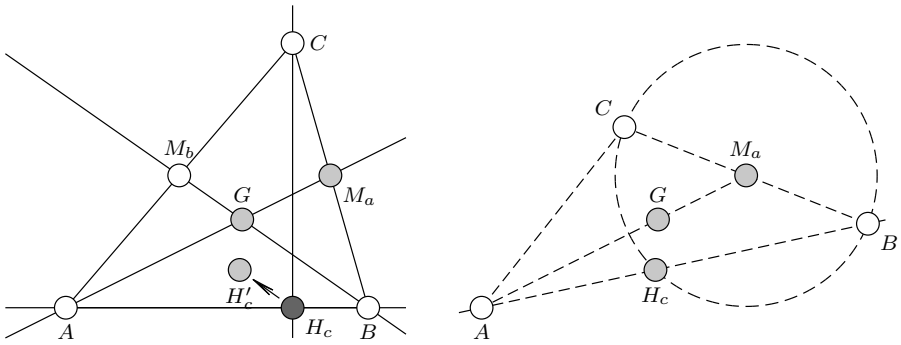
Kako rekonstruisati trougao? Iako imamo podatke o odabranim fiksim tačkama i karakterističnoj tački koju pomeramo, i dalje je teško da rekonstruišemo trougao. Ako uspemo da rekonstruišemo temena trougla, ostale tačke je lako rekonstruisati izvodeći konstruktivne korake. Ovaj problem je zapravo lokacijski problem konstrukcije trougla. Rešavanje ovog problema može se delegirati nekom rešavaču kao što je ArgoTriCS ili se može pamtiti unapred izračunata lista svih mogućih konstrukcija.

Primer 7 *Pretpostavimo da korisnik bira slobodne tačke A, B i C (temena trougla) i da potom izvodi sledeću konstrukciju:*

1. *Konstruisati tačku M_a koja je središte stranice BC;*
2. *Konstruisati tačku M_b koja je središte stranice AC;*
3. *Konstruisati pravu m_a kroz tačke A i M_a;*
4. *Konstruisati pravu m_b kroz tačke B i M_b;*
5. *Konstruisati presečnu tačku G pravih m_a i m_b;*
6. *Konstruisati pravu c kroz tačke B i C;*
7. *Konstruisati pravu h_c tako da sadrži tačku C i upravna je na pravu c;*
8. *Konstruisati presečnu tačku H_c pravih c i h_c.*

Razmotrimo situaciju kada korisnik pomera tačku H_c, dok su tačke M_a i G fiksirane. Iz konstrukcije zaključujemo da je tačka H_c podnožje visine iz temena C, tačka M_a središte stranice BC i tačka G težište trougla ABC. Da bismo rekonstruisali trougao ABC kada se pomeri tačka H_c potrebno je da rešimo sledeći konstrukcijski problem: za zadate pozicije tačaka H_c, M_a i G konstruisati trougao ABC, tako da je H_c podnožje visine iz tačke C, M_a središte stranice BC i G težište trougla ABC. Za ovaj konstrukcijski problem ArgoTriCS daje sledeće rešenje:

1. *Konstruisati tačku A tako da važi $\overrightarrow{AG} : \overrightarrow{AM_a} = 2 : 3$ (pravilo W01);*
2. *Konstruisati pravu c kroz tačke A i H_c (pravilo W02);*



Slika 4.2: Konstrukcija tačaka M_a, G i H_c trougla ABC (levo) i rekonstrukcija trougla ABC (desno)

3. *Konstruisati krug $k(M_a, B)$ čiji je centar tačka M_a i koji sadrži tačku H_c (pravilo W06);*

4. *Konstruisati presečnu tačku B kruga $k(M_a, B)$ i prave c, koja je različita od tačke H_c (pravilo W08);*

5. *Konstruisati tačku C tako da važi $\overrightarrow{BM_a} : \overrightarrow{BC} = 1 : 2$ (pravilo W01).*

Izvodeći ove konstruktivne korake za novu poziciju tacke H, dobijamo nove pozicije temena trougla ABC (slika 4.2 desno).

Glava 5

Alat Touch&Drag

Alat Touch&Drag je razvijan za Android uređaje[1]. Ovaj alat, koristeći prednosti uređaja osetljivih na dodir, pretvara skice u geometrijske konstrukcije. U alatu je omogućeno pomeranje, ne samo slobodnih, već i konstruisanih tačaka.

Slika 5.1: Konstruisanje trougla u alatu Touch&Drag

Glavni deo korisničkog interfejsa jeste platno po kome korisnik crta. Otisak koji korisnik ostavlja dok crta po platnu se prepoznaje u hodu i iscrtava ispreki-

[1] Aplikacija se može skinuti sa Google Play prodavnice i besplatna je za korišćenje.

danom linijom, dok su ostali, ranije prepoznati objekti, iscrtani punom linijom. Po prestanku crtanja isprekidana linija zamenjuje se punom linijom.

Postoje tri režima rada:

1. *Osnovni režim za crtanje.* U ovom režimu mogu se crtati novi objekti, ali se ne mogu menjati postojeći.

2. *Režim za pomeranje.* Ovaj režim omogućava pomeranje slobodnih tačaka na platnu, kao i uvećavanje i smanjivanje prikaza.

3. *Režim za odabir.* U ovom režimu je omogućeno menjanje tipa tačaka.

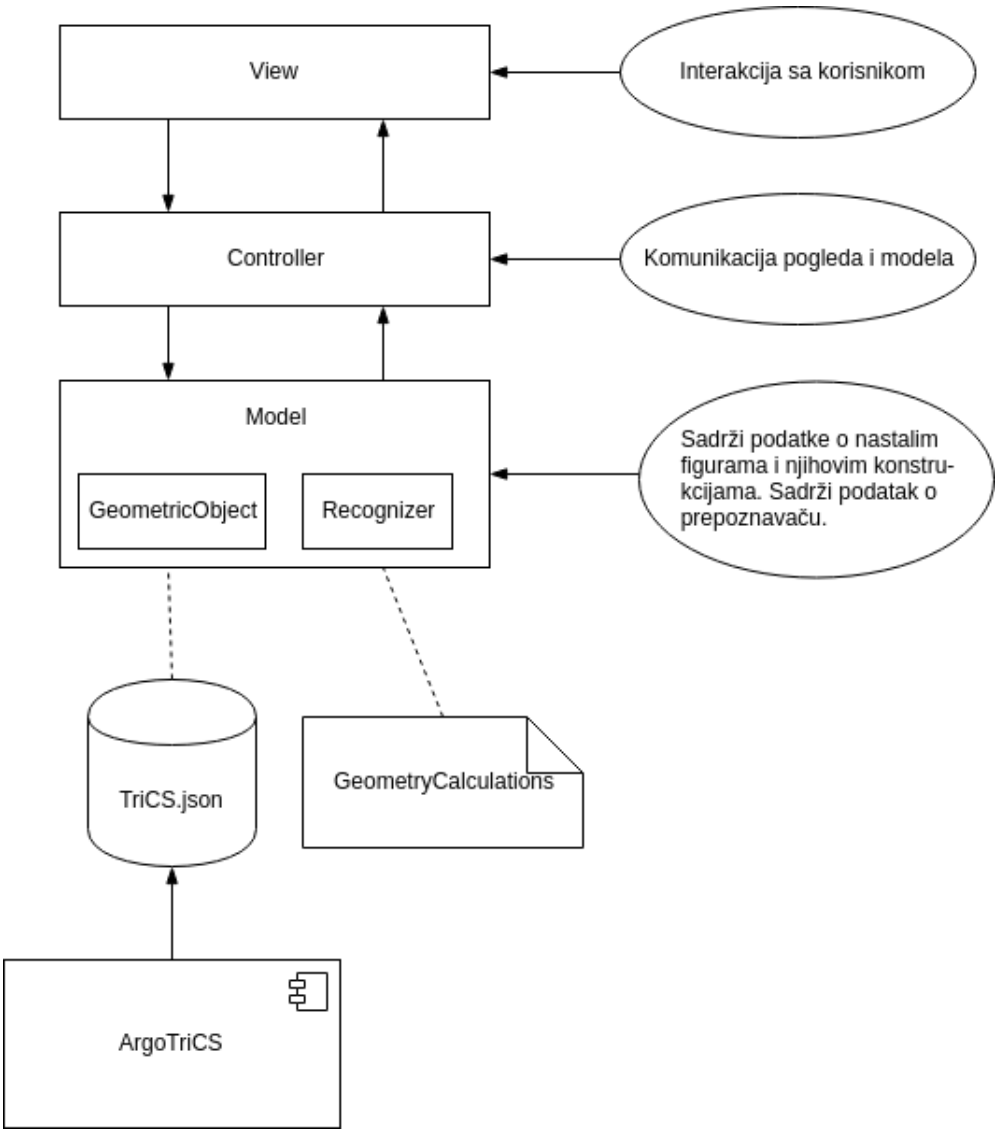
Alat Touch&Drag implementiran je u programskom jeziku Java koristeći biblioteke AndroidSDK[16] i GeometryCalculations. Na slici 5.2 je prikazana arhitektura sistema, osnovne uloge svakog sloja, kao i spoljne komponente koje se koriste.

AndroidSDK je skup biblioteka i alata koje omogućavaju razvijanje, testiranje i debagovanje aplikacija za android uređaje. AndroidSDK sadrži emulator, dokumetaciju, primere i uputstva. Razvoj android aplikacija koristeći AndroidSDK podržan je na većini Linux, Mac OS i Windows platforma.

Korisnik interaguje sa pogledom koji o akcijama korisnika obaveštava kontroler (veza između pogleda i modela). Kontroler obaveštava model o promenama na pogledu i koje akcije bi trebalo izvesti i šalje signale pogledu u slučaju kada se model promeni da je potrebno da se ažurira.

Model platna na kojem korisnik crta pamti sve informacije o nacrtanim geometrijskim figurama i opisu njihovih konstrukcija. Model komunicira sa spoljašnjim komponentama. On sadrži rešenja koja obezbeđuje rešavač ArgoTriCS. Rešenja, koja su iz ArgoTriCS-a eksportovana u JSON datoteku, se učitavaju u mapu koja je sadržana u modelu. U slučaju zamene ili dodavanja rešenja drugih rešavača potrebno ih je eksportovati u odgovarajuću datoteku koja bi se učitala. U ovom slučaju je odabrana JSON datoteka zbog manjeg broja konstrukcija, u slučaju većeg broja konstrukcija JSON bi mogao biti zamenjen bazom podataka.

Razmotrimo komunikaciju između komponenti u osnovnom režimu rada. Korisnik crta na platnu, pogled obaveštava kontrolera da se promene dešavaju. Kontroler reaguje na promene i obaveštava model da je crtanje u toku. Kontroler šalje modelu vektor tačaka koje predstavljaju trag koji je korisnik ostavio na platnu. Kada dobije taj vektor tačaka, model ih prosleđuje prepoznavaču koji na osnovu vektora tačaka prepoznaje odgovarajuću geometrijsku figuru. Kada je geometrijska figura prepoznata ona se dodaje u vektor svih nacrtanih geometrijskih figura i opis njene konstrukcije se pamti. Potom model obaveštava kontroler da je došlo do promene i da je potrebno ponovo iscrtati pogled. Kontroler obaveštava pogled da se ponovo iscrta i pogled se ponovo iscrtava.



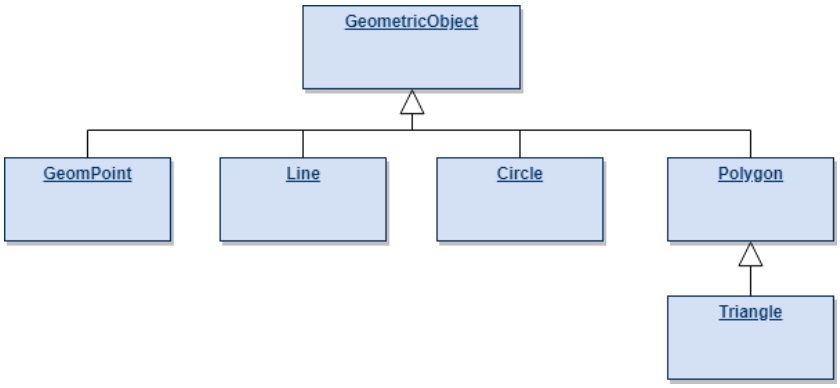
Slika 5.2: Arhitektura alata Touch&Drag, osnovne uloge svakog sloja, kao i spoljne komponente koje se koriste

5.1 Biblioteka GeometryCalculations

Tokom razvijanja alata Touch&Drag razvijena je android biblioteka GeometryCalculations. Ova biblioteka je napisana u programskom jeziku Java koristeći biblioteku OpenCV.

OpenCV (Open Source Computer Vision Library [15]) je biblioteka funkcija za analizu, obradu i modifikaciju slika. Napisana je i optimizovana u C/C++, te može koristiti prednosti višejezgarnog izračunavanja. Biblioteka ima C++, C, Python i Java interfejs i podržana je na Linux, Windows, Mac OS, iOS i Android operativnim sistemima. Dizajnirana je sa fokusom na aplikacije za efikasan rad u realnom vremenu. U ovom radu korišćene su funkcije za Hafovu transformaciju i metodu najmanjih kvadrata.

U okviru biblioteke GeometryCalculations nalaze se implementacije algoritma za prepoznavanje figura, klase kojima su predstavljene figure, implementacija osnovnih geometrijskih konstrukcija, kao i druge pomoćne klase. Spisak osnovnih geometrijskih konstrukcija može se naći u dodatku A.

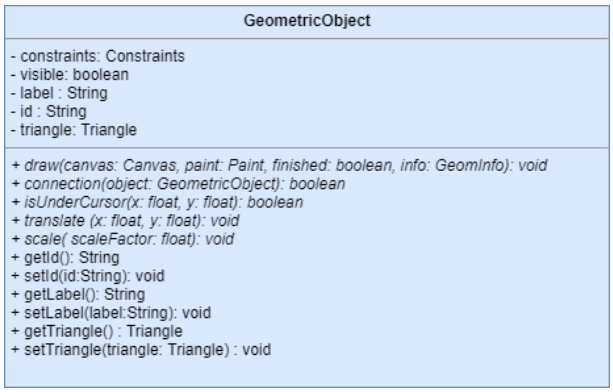


Slika 5.3: Hijearhija klasa kojima su predstavljene figure

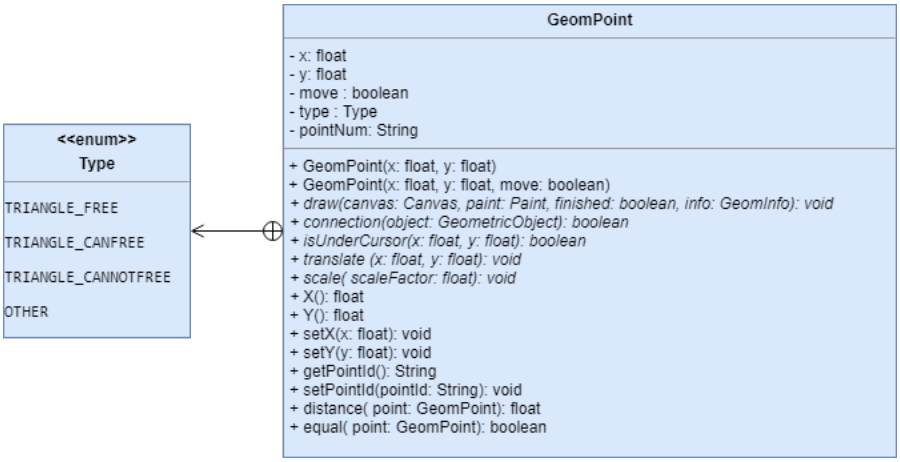
Geometrijska figura predstavljena je klasom `GeometricObject`. UML prikaz ove klase nalazi se na slici 5.4. Klasa čuva opšte podatke o figuri kao što su identifikator, oznaka, vidljivost, itd. Klasa `GeometricObject` sadrži apstraktne metode za crtanje (*draw*), proveru da li je figura povezana sa drugom figurom (*connect*), translaciju figure (*translate*), skaliranje figure (*scale*), proveru da li se figura nalazi na zadatoj lokaciji (*isUnderCursor*), kao i pomoćne metode.

Klasu `GeometricObject` direktno nasleđuju sledeće klase:

• Klasa `GeomPoint`, kojom je predstavljena tačka u ravni. UML prikaz ove klase može se videti na slici 5.5. Klasa, pored implementacija nasleđenih apstraktnih metoda klase `GeometricObject`, sadrži i metode za računanje udaljenosti od zadate tačke (*distance*) i proveru poklapanja sa zadatom tačkom (*equal*). Metod *equal* vraća `True` ukoliko je zadata tačka na rastojanju manjem od *minimalDistance*(5.1).



Slika 5.4: Klasni dijagram klase GeometricObject

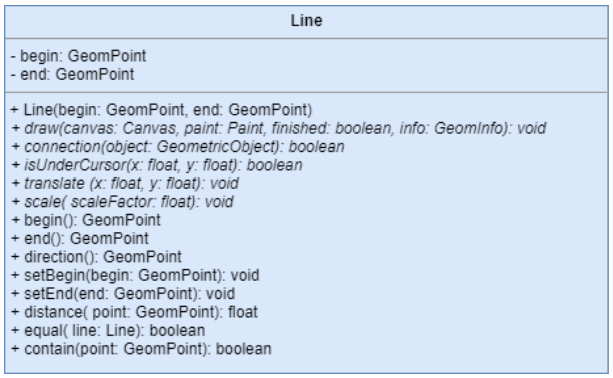


Slika 5.5: Klasni dijagram klase GeomPoint

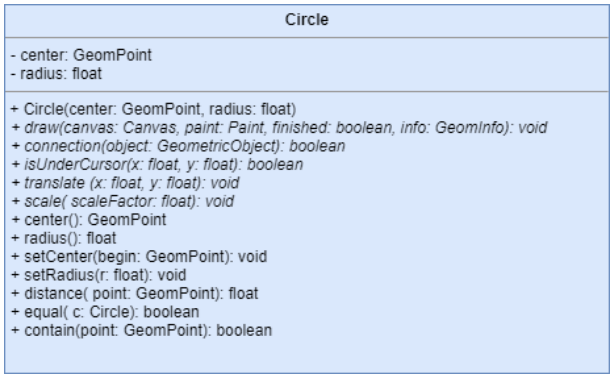
• Klasa `Line`, kojom je predstavljena prava u ravni. UML prikaz klase `Line` može se videti na slici 5.6. Pored nasleđenih metoda, u ovoj klasi su implementirani metodi za računanje udaljenosti od zadate tačke (*distance*), proveru da li tačka pripada pravoj (*contain*) i proveru poklapanja sa zadatom pravom (*equal*).

• Klasa `Circle`, kojom je predstavljen krug u ravni.UML dijagram ove klase prikazan je na slici 5.7. Ova klasa pored nasleđenih metoda implementira i metode za računanje udaljenosti od zadate tačke (*distance*), proveru da li tačka pripada krugu (*contain*) i proveru poklapanja sa zadatim krugom (*equal*).

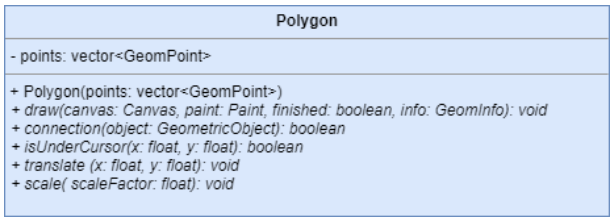
• Klasa `Polygon` kojom je predstavljen poligon u ravni. UML dijagram



Slika 5.6: Klasni dijagram klase Line



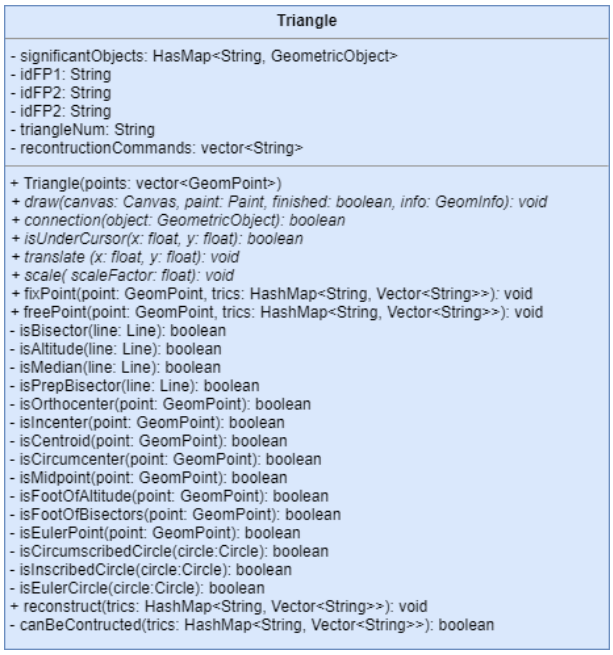
Slika 5.7: Klasni dijagram klase Circle



Slika 5.8: Klasni dijagram klase Polygon

može se videti na slici 5.8.

• Klasu `Polygon` nasleđuje klasa `Triangle`. Klasom `Triangle` je predstavljen trougao u ravni (slika 5.9). Svaki trougao je jedinstveno određen sa tri tačke (aktivna trojka). Na osnovu ove tri tačke se trougao rekonstruiše kori-



Slika 5.9: Klasni dijagram klase Triangle

steći konstruktivne korake koji su zapamćeni u vektoru *recontructionCommands*. Konstruktivni koraci koji su zapamćeni u vektoru su rešenje koje ArgoTriCS daje za problem: *za zadate tačke iz aktivne trojke konstruisati trougao ABC*. U slučaju da su te tri tačke temena trougla, vektor *recontructionCommands* je prazan. U režimu za odabir moguće je promeniti tip jedne od tačaka koje određuju trougao i time je izbaciti iz aktivne trojke, tada trougao i dalje postoji i interno je određen svojim temenima, ali takav trougao korisnik ne može da modifikuje. Metodima *fixPoint* i *freePoint* menja se aktivna trojka. Pored ovih metoda u klasi `Triangle` su implementirani i pomoćni privatni metodi kojima se proverava veza trougla sa zadatom figurom. U alatu su podržane sledeće provere veze trougla sa drugim figurama:

- da li je prava simetrala ugla (*isBisectorLine*);
- da li je prava visina trougla (*isAltitude*);
- da li je prava tržišna linija (*isMedian*);
- da li je prava simetrala stranice (*isPrepBisector*);
- da li je tačka ortocentar trougla (*isOthocenter*);
- da li je tačka centar upisanog kruga trougla (*isIncenter*);
- da li je tačka centar opisanog kruga trougla (*isCircumcenter*);
- da li je tačka težište trougla (*isCentroid*);

– da li je tačka središte stranice trougla (*isMidpoint*);
– da li je tačka podnožje visine trougla (*isFootOfAltitude*);
– da li je tačka presek simetrale ugla sa stranicom (*isFootOfBisectors*);
– da li je tačka Ojlerova tačka (*isEulerPoint*);
– da li je krug opisani krug trougla (*isCircumscribedCircle*);
– da li je krug upisani krug trougla (*isInscribedCircle*);
– da li je krug Ojlerov krug (*isEulerCircle*).

Konstante, koje se koriste u proveri odnosa između geometrijskih figura i tokom prepoznavanja figura, definisane su u klasi `Constraints`. Konstante direktno zavise od karakteristika ekrana uređaja na kom se aplikacija izvršava, zbog čega se instanca klase kreira pri pokretanju android aplikacije. Svakoj novoj instanci klase `GeometricObject` instanca klase `Constraints` se prosleđuje kroz konstruktor. Definisane su konstante *minimalDistance*, *minNumOfPoints*, *maximalNumberOfPoints*, *minimalRatio*, *maximalRatio*, *maximalRadius* i *EPS*. Do vrednosti ovih konstanti se došlo eksperimentalno. Zbog raznovrsnosti uređaja i očekivanja korisnika vrednosti ovih konstanti se mogu menjati.

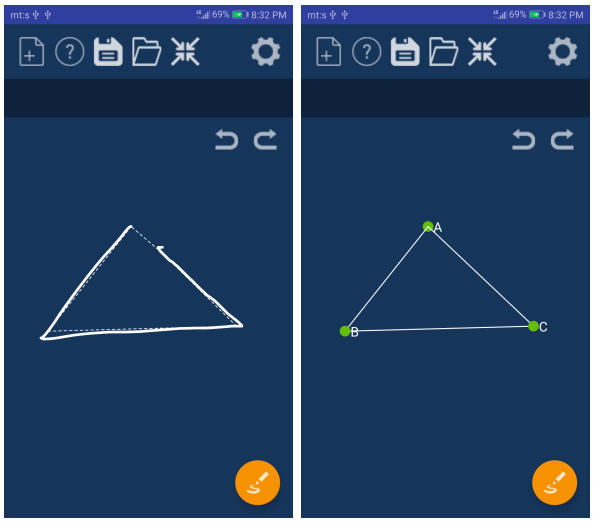
U okviru biblioteke nalaze se i klase `ConnectionCalculation` i `GeometricConstructions`. U klasi `ConnectionCalculation` su implementirane metode koje proveravaju da li su dve figure povezane. Ove metode se pozivaju u okviru implementacija metoda *connect*. U klasi `GeometricConstructions` su implementirane osnovne geometrijske konstrukcije.

5.1.1 Implementacija algoritma za prepoznavanje figura

Implementacija algoritma za prepoznavanje figura nalazi se u okviru klase `GeometricObjectRecognizer`. Metod *getGeometricObject* kao argument prima diskretizovan vektor tačaka otiska na ekranu i kao rezultat vraća prepoznatu figuru.

Prvobitna implementacija algoritma prepoznavanja figura koristila je Hafovu transformaciju. Korišćene su funkcije *HoughCircles* i *HoughLines* iz biblioteke OpenCV. Hafova transformacija nije davala dobre rezultate jer otisak koji ostaje na ekranu dok korisnik crta nije pravilna figura. Mana ovog pristupa je još i to što funkcije *HoughCircle* i *HoughLines* primaju kao argumente rastersku sliku, pa dolazi do znatnog usporenja rada aplikacije, što je nedopustivo za aplikaciju koja treba da radi u realnom vremenu.

Sledeća implementacija koristila je algoritam zasnovan na diskretnoj krivini opisan u odeljku 3.2. Ova implementacija je davala bolje rezultate u odnosu na prethodnu. Ukoliko je otisak sličan pravoj ili poligonu metod *getGeometricObject*, implementacija zasnovana na diskretnoj krivini, vraća pravu, odnosno poligon. Međutim ako je otisak sličan krugu metod ne vraća krug. Naime, zbog malog broja tačaka koje se zapamte, u proseku oko 20 tačaka, dolazi do problema da se figure koje bi idejno trebale biti prepoznate kao krugovi budu prepoznate kao poligoni. Ako bi se konstante podesile tako da prepoznavanje krugova bude bolje, onda prepoznavanje poligona postaje lošije.



Slika 5.10: Skiciranje trougla (levo) i prikaz prepoznatog trougla na osnovu skice (desno)

Najbolje rezultate dala je implementacija koja koristi metod najmanjih kvadrata opisan u odeljku 3.3. Prepoznavanje se vrši na sledeći način:

1. Ukoliko je broj tačaka manji od *minNumOfPoints* metod *getGeometricObject* vraća `Nothing` kao naznaku da ni jedna figura nije prepoznata. Ova provera je uvedena kako bi se rešio problem nagomilavanja tačaka.

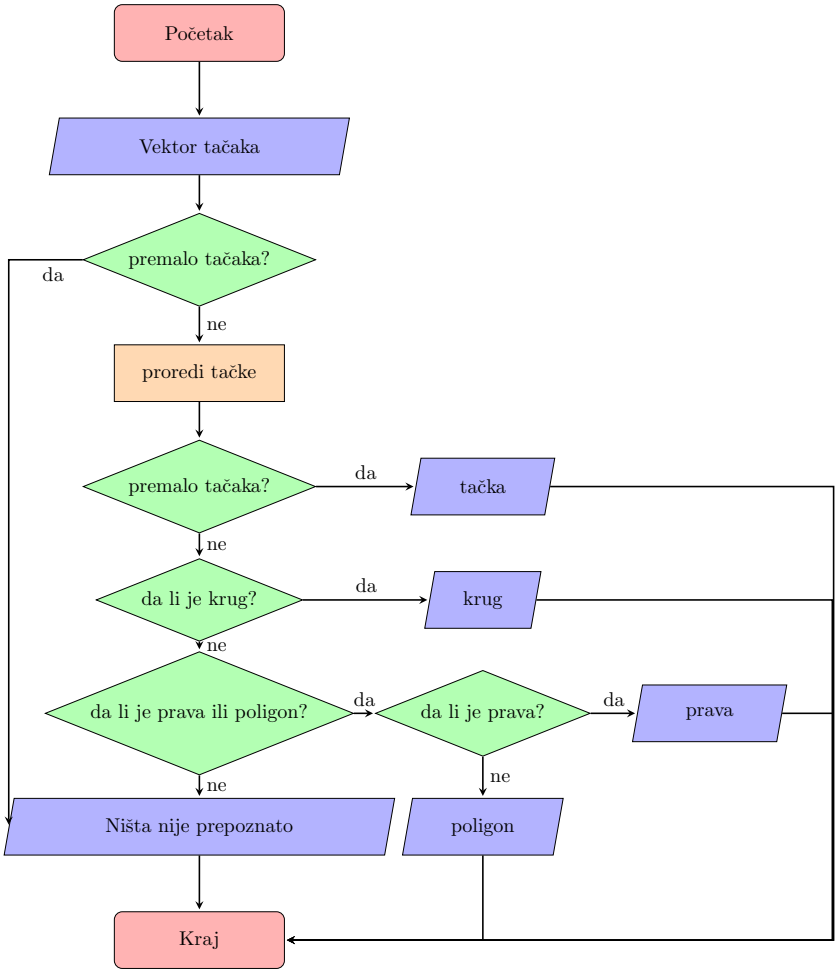
2. Zbog numeričkih grešaka prilikom deljenja sa malim brojevima skup tačaka se proređuje tako da rastojanje između svake dve uzastopne tačka bude bar *minimalDistance*.

3. Ukoliko je broj tačaka posle proređivanja manji od *minNumOfPoints* metod vraća tačku.

4. Metodom najmanjih kvadrata, na način opisan u primeru 3, dati vektor tačaka se aproksimira krugom. Korišćena je implementacija najmanjih kvadrata iz biblioteke OpenCV. Odluka da li je dobijeni krug zaista dobra aproksimacija datog vektora tačaka vrši se proverom odnosa udaljenosti tačke od centra dobijenog kruga i poluprečnika kruga. Ukoliko je ovaj odnos u intervalu $(minimalRatio, maximalRatio)$ i poluprečnik manji od *maximalRadius* metod *getGeometricObject* vraća aproksimirani krug. Provera poluprečnika vrši se jer metod najmanjih kvadrata vektor tačaka, koje najpre pripadaju pravoj, aproksimira krugom velikog poluprečnika.

5. Ukoliko aproksimirani krug nije dovoljno dobra aproksimacija vektora tačaka onda se metodom *isPolyLine* proverava da li se dati skup tačaka

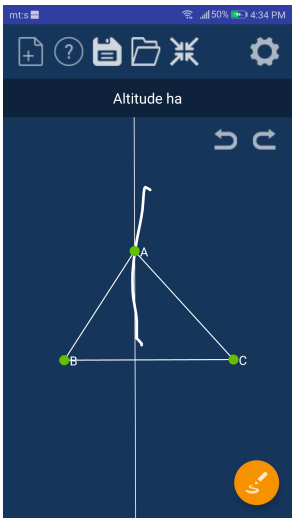
može aproksimirati pravom ili poligonom. Proverom ugla između svake dve uzastopne tačke odlučuje se da li tačke leže na pravoj, pripadaju poligonu ili nijedno od navedenog. Kôd ovog metoda je naveden u dodatku B.



Slika 5.11: Dijagram toka prepoznavanja figura

6. Metod *getGeometricObject* vraća `Nothing`, kao naznaku da ništa nije prepoznato.

Prepoznavanje figure se vrši u hodu, dok korisnik crta. Figura koja je prepoznata na osnovu dotadašnje skice se iscrtava isprekidano, kako bi korisnik uvek imao informaciju šta je prepoznato na osnovu skice i mogao da je menja ne bi li dobio željeni rezultat. Po završetku prepoznavanja figure linearanom pretragom po ranije nacrtanim figurama proveravaju se odnosi prepoznate figure sa već



Slika 5.12: Konstrukcija visine trougla

postojećim. Ukoliko postoji približna veza sa nekom već postojećom figurom, prepoznata figura se menja tako da data veza zaista važi. Na primer, ukoliko je prepoznata prava približno jednaka[2] visini trougla, prava se menja tako da ona postane visina trougla. Po završenom prepoznavanju ova veza se pamti u obe geometrijske figure. Ovakvo rešenje olakšava odgovaranje na pitanje *koja je karakteristična tačka odabrana konstruisana tačka*. Kada se kontruiše karakteristična tačka, sistem prepozna koja je to karakteristična tačka i tu informaciju upamti. Tada konstruisana tačka ima informaciju koja je karakteristična tačka i kom trouglu pripada. Takođe, i trougao, kome je konstrisana tačka karakteristična tačka, pamti informaciju o karakterističnoj tački. Moguća situacija je da novo prepoznata figura nije samo povezana sa jednom, već sa više figura. U ovom slučaju alat prepoznaje samo prvo pronađenu vezu. Veze između figura se proveravaju takođe u hodu i informacija o pronađenoj vezi se ispisuje (slika 5.12).

5.2 Pomeranje tačaka

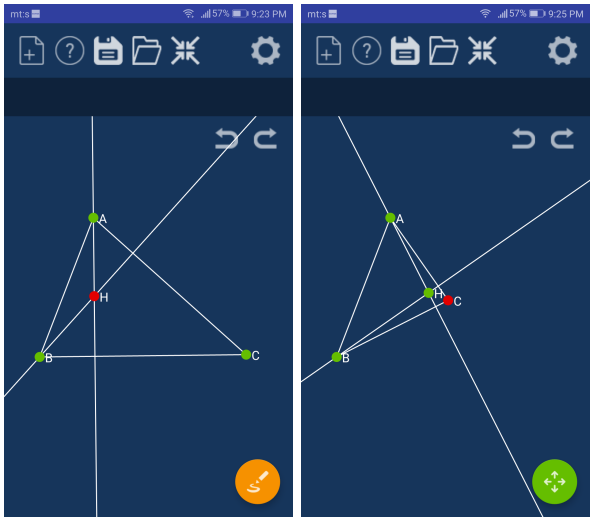
Kao i u većini alata za dinamičku geometriju, u alatu Touch&Drag je omogućeno pomeranje figura. Tačnije omogućeno je pomeranje tačaka, a kako su sve ostale figure zavisne od tačaka, one se pomeraju (menjaju) pomeranjem tačaka od kojih zavise. Za razliku od ostalih alata, u ovom alatu je moguće pomerati i konstruisane karakteristične tačke trougla.

Za pomeranje konstruisanih tačaka koristi se ranije izračunata lista kon-

[2]Prava je približno jednaka visini trougla ukoliko je udaljenost odgovarajućeg temena i podnožja visine iz tog temena od prave manja od *minimalDistance*.

strukcija u alatu ArgoTriCS (pamte se konstrukcije samo za rešive probleme). Ova lista konstrukcija se pamti u *JSON* formatu. Pri pokretanju aplikacije konstrukcije se učitavaju u mapu *trics* čiji je ključ konstruktivni problem (niska oblika $X\ Y\ Z$) i vrednost rešenje konstruktivnog problema *za zadate tačke* $X\ Y\ Z$ *konstruisati trougao* ABC. Kako alat ArgoTriCS daje rešenja samo za tačke iz Konelijevog korpusa, dozvoljeno je pomeranje samo Konelijevih tačaka.

Da bi pomerao neku konstruisanu karakterističnu tačku trougla, korisnik mora da promeni aktivnu trojku, kojom je određen trougao. Kao što je već pomenuto, ovo se može uraditi u režimu za odabir. Pre nego što odabere novu tačku koju želi da pomera, korisnik mora da ukloni jednu od prethodnih tačaka iz trojke. Kada ostanu dve tačke X i Y u trojci, korisniku se nudi opcija da doda treću tačku Z takvu da ArgoTriCS daje jedinistveno rešenje za problem "$X\ Y\ Z$"(odnosno postoji ključ "$X\ Y\ Z$" u mapi *trics*). Pri promeni trojke tačaka koje određuju trougao, ažurira se vektor *recontructionCommands* i postavlja se na *trics*[XYZ].



Slika 5.13: Pre pomeranja tačke H (levo) i posle (desno)

Pri pomeranju karakteristične tačke šalje se signal trouglu, čija je to karakteristična tačka, da je potrebno da se rekonstruiše. Prvo se izvode konstruktivni koraci iz vektora *recontructionCommands* i time se dobijaju nove pozicije tema trougla, a potom se izvode konstruktivni koraci kojima su opisane sve figure na slici i slika se ažurira. Postoje situacije kada na osnovu pozicija tačaka trougao ne može biti rekonstruisan (degenerisani slučajevi). Tada trougao nestaje sa svim svojim karakterističnim figurama (interno se temena postavljaju na NULL, a potom pri izvođenju konstruktivnih koraka sve figure koji zavise od njih postaju NULL) osim tačaka iz aktivne trojke. Kada tačke opet budu na pozicijama tako da određuju trougao, trougao će se pojaviti na slici sa svim svojim ranije konstruisanim karakterističnim figurama.

Primer 8 *Razmotrimo situaciju kada korisnik konstruiše trougao ABC i njegov ortocentar H i želi da pomeri tačku H.*

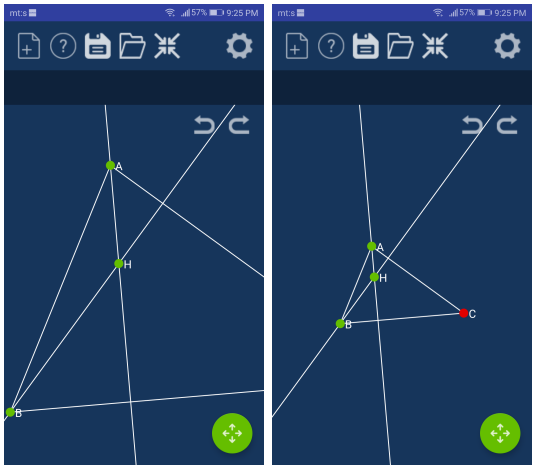
U osnovnom režimu, korisnik skicira trougao i alat prepoznaje trougao. Potom skicirajući prave koje su ,,blizu" visina h_a i h_b, korisnik konstruiše visine. Ortocentar H se jednostavno konstruiše kao presek pravih h_a i h_b dodirom na mesto preka pravih.

Kada je izvršio opisanu konstrukciju korisniku je dozvoljeno da u režimu za pomeranje pomera samo temena trougla ABC. Da bi pomerao tačku H potrebno je da je odabere u režimu za odabir. Pri ulasku u režim za odabir korisnik ima samo mogućnost da ukloni neko od temena trougla, jer su sve tri tačke odabrane. Po uklanjanju tačke C (ista situacija bi bila da je uklonjeno neko drugo teme) omogućen je odabir tačke H. Korisnik bira tačku H i prelazi u režim za pomeranje. Sada korisnik može da pomera tačke A, B i H. Pomeranjem tačke H trougao se automatski rekonstruiše i nova slika se prikazuje.

5.3 Pregled dodatnih funkcionalnosti alata

Pored opisanih funkcionalnosti omogućene su i sledeće:

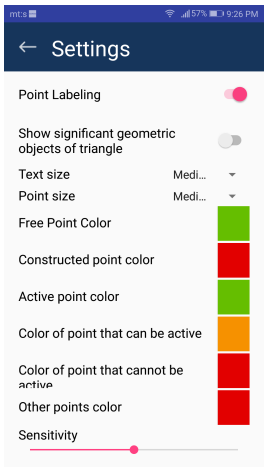
• Čuvanje/Učitavanje. Moguće je sačuvati konstrukciju i potom je kasnije ponovo učitati. Konstrukcije se čuvaju u lokalnom formatu u memoriji aplikacije na uređaju. Za sada nije podržano učitavanje i eksportovanje drugih formata, kao ni razmena sačuvanih konstrukcija između aplikacija na različitim uređajima.



Slika 5.14: Tačka C van ekrana (levo), trougao centriran (desno)

• Centriranje slike. Usled pomeranja tačaka dešava se da neke figre ,,pobegnu" iz okvira ekrana. Pritiskom na dugme slika se translira u centar i skalira tako da sve figure budu u okviru ekrana.

- Undo/Redo.



Slika 5.15: Podešavanja u alatu Touch&Drag

- Podešavanja. Moguće je promeniti boju i veličinu prikaza tačaka, kao i prikaz oznaka tačaka. Za pomoć pri konstrukciji karakterističnih tačaka, pravih i krugova trougla, može se aktivirati pretprikaz karakterističnih figura. Takođe, može se podesiti i osetljivost.

Glava 6

Poređenje sa srodnim sistemima

6.1 Prepoznavanje geometrijskih figura na skicama

Ideja o prepoznavanju skica nije nova. Skice i dijagrami su način komunikacije koji je prethodio pojavi pisma pre više od 30 vekova. Suterland je 1963. godine predstavio Sketchpad [23], prvi interaktivni sistem koji je koristio svetlosnu olovku za crtanje dijagrama direktno na ekranu. Glavni nedostaci ovog alata su neprepoznavanje figura na osnovu skica, ograničeni resursi i visoka cena računara.

U radu [25] je predložen algoritam prepoznavanja oblika na osnovu skica. Preciznost prepoznavanja ovog algoritma je preko 90%. Pretprocesiranje tačaka se sastoji od aproksimacije tačaka poligonskom linijom, filtriranja i računanja konveksnog omotača. Rezultat pretprocesiranja je konveksan poligon. Broj temena poligona se potom smanjuje, uklanjaju se temena kod kojih je ugao blizak opruženom uglu. Skica se klasifikuje kao poligon ili elipsa. Za prepoznavanje poligona se koristi metod najmanjih kvadrata. Ose i žiže elipse se računaju na osnovu skupa proređenih tačaka i potom se minimizuje razlika između skice i prepoznate elipse.

Opis sistema za prepoznavanje oblika na osnovu skica koji koristi osnovna geometrijska znanja i ne zavisi od konkretne primene dat je u radu [26]. Proces prepoznavanja se sastoji od dve faze. U prvoj fazi se skica aproksimira jednom primitivnom geometrijskom figurom (prava, luk, krug, zavojnica) ili kombinacijom primitivnih figura. U rekurzivnom algoritmu se skica deli na osnovu pravca i krivine nacrtale linije (skice). Zatim se delovi aproksimiraju primitivnim figurama. U postprocesiranju se analiziranjem veza između prepoznatih figura uklanjaju suvišne primitivne figure. Za osnovne figure ovaj prepoznavač je prepoznao ispravnu figuru u 98% slučaja.

Istraživanje u oblasti prepoznavanja oblika na skicama je u ranim fazama bilo fokusirano na konkretan domen, kao što je prepoznavanje osnovnih oblika u UML dijagramima, mašinstvu, web dizajnu, itd. Kako bi se olakšao razvoj novih prepoznavača nastao je LADDER[27], jezik za opisivanje crteža, prikaza i modifikacije skiciranih dijagrama. Međutim, da bi se napisao prepoznavač za novi domen potrebno je napisati gramatiku kojom se opisuje novi domen. U LADDER-u je između ostalih napisan i PaleoSketch[28]. PaleoSketch prepoznaje osnovne geometrijske oblike kao što su prave, poligoni, krugovi, elipse, lukovi, itd. Prepoznavači napisani u LADDER-u su prepoznavali najviše dvadesetak različitih oblika. U 2010., razvijen je sistem koji prepoznaje stotinak različitih oblika. Međutim, i ovaj sistem je zavisio od prethodno određenog skupa oblika koji se mogu prepoznati. Veći napredak u prepoznavanju oblika su napravili naučnici iz Microsoft Research-a[29]. Oni su razvili sistem za prepoznavanje oblika koji ne zavisi od domena i nije ograničen prethodno određenim skupom oblika koji se mogu prepoznati. Najnoviji metod za prepoznavanje oblika na skicama patentiran je u januaru 2018.

U okviru alata GeoGebra 4.2 dodata je mogućnost crtanja figura slobodnom rukom. Ukoliko je figura nacrtana slobodnom rukom slična nekoj geometrijskoj figuri (tačka, prava, krug, elipsa, poligon, itd.) onda će ona biti prepoznata i iscrtana. Alat ne daje informacije u toku crtanja o tome da li se i koja figura može prepoznati.

Crtanje u alatu Sketchometry se izvodi slobodnom rukom. U toku crtanja se vrši prepoznavanje i ispisuje se koja bi figura bila prepoznata ako bi crtanje prestalo u tom trenutku. U toku prepoznavanja proveravaju se odnosi sa postojećim figurama (paralelnost, normalnost, pripadnost, itd.). Na primer, ako je prava koja se crta skoro paralelna sa nekom već nacrtanom, ona će se prepoznati kao prava paralelna sa tom pravom.

6.2 Pomeranje tačaka

Od samog početka u alatima za dinamičku geometriju moguće je modifikovati ulazne parametre i posmatrati posledice. Po promeni ulaznih parametara u alatima se izvodi konstrukcija ostalih figura i korisniku se prikazuje izmenjena slika. Ni u jednom alatu do sada ne postoji mogućnost pomeranja konstruisanih tačaka[1]. Alat Touch&Drag je prvi alat za dinamičku geometriju u kome postoji ova funkcionalnost.

[1]U jednoj verziji alata Geoplan-Geospace (iz 2005.) postojala je slična funkcionalnost zasnovana na algebarskom pristupu. Nažalost, više detalja o toj funkcionalnosti nije nikada objavljeno.

Glava 7

Zaključci i dalji rad

U okviru ovog rada su prikazana moguća rešenja za prepoznavanje geometrijskih figura koje se skiciraju na uređajima osetljivim na dodir. Algoritam za prepoznavanje geometrijskih figura, opisan u tekstu, implementiran je u aplikaciji Touch&Drag. Pokazalo se da poznati algoritmi nisu davali najbolje rezultate, kao i da prepoznavanje nije jednostavno. U okviru aplikacije implementirani su i algoritmi za prepoznavanje odnosa između prepoznatih figura.

U daljem radu na aplikaciji Touch&Drag potrebno je unaprediti prepoznavanje tako da prepoznaje više figura. Potrebno je popraviti iskustveno izabrane vrednosti za dopustiva odstupanja. Ovo se može unaprediti, na primer, korišćenjem algoritama mašinskog učenja. Pri prepoznavanju veza između prepoznatih figura trenutni pristup pronalazi prvu vezu kojoj je greška dopustiva i potom prestaje pretragu. To se može unaprediti tako što će se umesto prve tražiti sve veze, kojima su greške dopustive i birati ona koja ima najmanju grešku.

U radu je takođe opisana nova funkcionalnost: pomeranje konstruisanih karakterističnih tačaka trougla. Pomerajući konstruisanu tačku, korisnik istražuje uticaj na ostale postojeće figure. Ova funkcionalnost je implementirana u alatu Touch&Drag. U radu nisu pokrivene sve moguće karakteristične tačke, već samo Konelijeve tačake. U daljem radu na aplikaciji moguće je proširiti skup karakterističnih tačaka trougla. Takođe, skup figura može se proširiti sa tačaka na dodatne geometrijske figure. Ovo otvara nove konstrukcijske probleme koje je potrebno (automatski) rešiti. Predloženo rešenje koristi rešavač AgroTriCS, korišćenje drugog rešavača ili kombinovanje rezultata više njih može povećati skup figura. U alatu Touch&Drag može se sačuvati konstrukcija samo u lokalnom formatu na uređaju na kome je aplikacija instalirana. Dodavanjem opcija čuvanja i čitanja drugih formata povećala bi se povezanost sa drugim alatima i uvećao broj gotovih konstrukcija koje je moguće istraživati u novom svetlu.

Bibliografija

[1] U. Kortenkamp *Foundations of dynamic geometry* Ph.D. thesis, ETH Zurich (1999.). URL doi.org/10.3929/ethz-a-003876663

[2] M. Yerushalmy, R. A. Houde *The Geometric Supposer: Promoting Thinking and Learning* The Mathematics Teacher 79 (6), 418–422, 1986.

[3] P. Janičić, P. Quaresma *System description: GCLCprover + GeoThms* International Joint Conference on Automated Reasoning (IJCAR-2006), Vol. 4130 of Lecture Notes in Artificial Intelligence, Springer-Verlag, pp. 145–150, 2006.

[4] M. Hohenwarter, K. Fuchs *Combination of dynamic geometry, algebra and calculus in the software system GeoGebra* Computer Algebra Systems and Dynamic Geometry Systems in Mathematics Teaching Conference 2004, Pecs, Hungary, pp. 128–133, 2004.

[5] Lisha Zhang, Zhengxing Sun *An experimental comparison of machine learning for adaptive sketch recognition* Applied Mathematics and Computation 185, 1138–1148, 2007.

[6] Richard O. Duda, Peter E. Hart *Use of the Hough transformation to detect lines and curves in pictures* Communications of the ACM 15 (1), 180-188, 1972.

[7] Paul V. C. Hough *Method and means for recognizing complex patterns* US Patent 3,069,654, 1962.

[8] Hungwen Li, Mark A. Lavin, Ronald J. Le Master *Fast Hough transform: A hierarchical approach* Computer Vision, Graphics, and Image Processing 36 (2–3), 139-161, 1986.

[9] M.C.K. Yang, Jong-Sen Lee, Cheng-Chang Lien, Chung-Lin Huang *Hough transform modified by line connectivity and line thickness* `http://ieeexplore.ieee.org/abstract/document/608293/`

[10] M.Atiquzzaman, M.W.Akhtar *A Robust Hough Transform Technique for Complete Line Segment Description* Real-Time Imaging 1 (6), 419-426, 1995.

[11] Dana H. Ballard *Generalizing the Hough transform to detect arbitrary shapes* Pattern Recognition 13, 2, 1981.

[12] Tijana Z. Šukilović *Curvature based shape detection* Computational Geometry 48 (3), 180-188, 2014.

[13] Desanka P. Radunović *Numeričke metode* Akademska misao, Beograd, 2003.

[14] MATLAB, User's Guide, The MathWorks, Inc

[15] `https://opencv.org/`

[16] `https://en.wikipedia.org/wiki/Android_software_development`

[17] V. Marinković, P. Janičić *Towards Understanding Triangle Construction Problems* Intelligent Computer Mathematics - CICM 2012, Vol. 7362 of Lecture Notes in Computer Science, Springer, pp. 126-141, 2012.

[18] V. Marinković, P. Janičić, P. Schreck *Solving Geometric Construction Problems Supported by Theorem Proving* Proceedings of the 10th International Workshop on Automated Deduction in Geometry (ADG 2014), CISUC Technical Report TR 2014/01, University of Coimbra, pp. 121-146, 2014.

[19] V. Marinković *ArgoTriCS - Automated Triangle Construction Solver* Journal of Experimental & Theoretical Artificial Intelligence 29 (2), 247-271, 2017.

[20] W. Wernick *Triangle constructions vith three located points* Mathematics Magazine 55(4), 227–230, 1982.

[21] H. Connelly *An extension of triangle constructions from located points* Forum Geometricorum 9, 109–112, 2009.

[22] Chen Hua *MVC Design Pattern* Comput Knowledge Technol, 2007.

[23] I.E. Sutherland *Sketchpad: A Man-Machine Graphical Communications System* Technical Report 296, MIT Lincoln Laboratories, 1963.

[24] Charles C. Tappert, Ching Y. Suen, Toru Wakahara *The state of the art in on-line handwriting recognition* Transactions on pattern analysis and machine intelligence, 12 (8), 1990.

[25] Jin Xiangyu, Liu Wenyin, Sun Jianyong, and Zhengxing Sun *On-line graphics recognition* Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on, pages 256–264, 2002.

[26] Bo Yu and Shijie Cai *A domain-independent system for sketch recognition* Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia, GRAPHITE '03, pages 141–146, New York, NY, USA, 2003.

[27] Tracy Hammond, Randall Davis *LADDER, a sketching language for user interface developers* Computers&Graphics, Elsevier, pages 518-532, 2005.

[28] B. Paulson and T. Hammond *Paleosketch: accurate primitive sketch recognition and beautification* IUI, 2008.

[29] Zhenbang Sun, Changhu Wang, Liqing Zhang, Lei Zhang *Query-Adaptive Shape Topic Mining for Hand-Drawn Sketch Recognition* Proceedings of the 20th ACM international conference on Multimedia, pages 519-528

Dodatak A

Spisak osnovnih konstruktivnih koraka:

W01 Ako su date tačke X, Z i W, i racionalan broj r moguće je konstruisati tačku Y tako da važi: $\frac{XY}{ZW} = r$;

W02 Ako su date dve tačke X i Y moguće je konstruisati pravu XY; uslov određenosti je da su tačke X i Y različite;

W03 Ako su date dve prave, moguće je konstruisati njihovu zajedničku tačku; uslov nedegenerisanosti je da prave nisu paralelne, a uslov određenosti da nisu jednake;

W04 Ako su dati prava i krug, moguće je konstruisati njihove zajedničke tačke; uslov nedegenerisanosti je da se prava i krug seku;

W05 Ako su dati prava i krug i jedna njihova zajednička tačka, moguće je konstruisati njihovu drugu zajedničku tačku; uslov nedegenerisanosti je da se prava i krug seku;

W06 Ako su date dve različite tačke X i Y moguće je konstruisati krug $k(X, Y)$ sa centrom u tački X koji sadrži tačku Y ; uslov nedegenerisa- nosti je da su tačke X i Y različite;

W07 Ako su data dva kruga, moguće je konstruisati njihove dve zajedničke tačke; uslov nedegenerisanosti je da se krugovi seku, a uslov određenosti da su krugovi različiti;

W08 Ako su data dva kruga i jedna njihova zajednička tačka, moguće je konstruisati njihovu drugu zajedničku tačku; uslov nedegenerisanosti je da se krugovi seku, a uslov određenosti da su krugovi različiti;

W09 Ako su date tačke X i Y moguće je konstruisati krug sa prečnikom XY ; uslov nedegenerisanosti je da su tačke različite;

W10 Ako su dati tačka X i prava p moguće je konstruisati pravu q koja sadrži tačku X i upravna je na pravu p;

W11 Ako su dati prava p i tačka X koja ne pripada pravoj p moguće je konstruisati krug k sa centrom X koji dodiruje pravu p; uslov nedegenerisanosti je da tačka X ne pripada pravoj p;

W12 Ako su dati krug k i tačka X van kruga k, moguće je konstruisati dve tangente iz tačke X na krug k; uslov nedegenerisanosti je da je tačka X van

kruga k;

W13 Ako su dati krug k, tačka X van kruga k i jedna tangenta iz tačke X na krug k, moguće je konstruisati drugu tangentu iz tačke X na krug k; uslov nedegenerisanosti je da je tačka X van kruga k;

W14 Ako su date tačke X i Y moguće je konstruisati medijatrisu duži XY; uslov određenosti je da su tačke X i Y različite;

W15 Ako su dati tačka X, prava p i racionalan broj r, moguće je konstruisati pravu koja je slika prave p pri homotetiji u odnosu na tačku X sa koeficijentom r;

W16 Ako su dati tačka X i prava p moguće je konstruisati pravu koja sadrži tačku X i paralelna je pravoj p;

W17 Ako su date tačke X i Y, koeficijenti A, B, CiD i ugao α moguće je konstruisati pravu q tako da je ugao $\sphericalangle(XY, q) = A\frac{\alpha}{2^B} + C\frac{\pi}{2^D}$;

W18 Ako su date tačke X i Y, koeficijenti A, B, CiD i ugao α moguće je konstruisati pravu q tako da je ugao $\sphericalangle(q, XY) = A\frac{\alpha}{2^B} + C\frac{\pi}{2^D}$;

W19 Ako su date tačke X, Y i Z moguće je konstruisati tačku W koja je harmonijski spregnuta sa ostalim; uslov nedegenerisanosti je da su tačke X i Y različite, tačke Y i Z različite i da tačka Y nije središte duži XZ;

W20 Ako su date tačke X i Y i ugao α moguće je konstruisati skup tačaka iz kojih se duž XY vidi pod uglom $A\frac{\alpha}{2^B} + C\frac{\pi}{2^D}$;

W22 Ako su dati tačka X i krug k_1 moguće je konstruisati krug k_2 sa centrom u tački X koji iznutra dodiruje krug k_1; uslov nedegenerisanosti je da je tačka X unutar kruga k_1 i da tačka X nije centar kruga k_1;

point Za zadata dva broja x i y moguće je konstruisati tačku X sa koordinatama (x, y);

line Za zadate tačke X i Y moguće je konstruisati pravu sa prolazi kroz tačke X i Y;

circle Za zadatu tačku X i realan broj r moguće je konstruisati krug $k(X, r)$ sa centrom u tački X i poluprečnika r;

polygon Za zadate tačke $X_1, X_2, \ldots X_n$ moguće je konstruisati poligon čija su temena zadate tačke;

triangle Za zadate tačke A, B i C moguće je konstruisati trougao sa temenima u tačkama A, B i C.

Dodatak B

```
GeometricObject isPolyLine(Vector<GeomPoint> points){
  int lessThenPI = 0;
  int n = points.size();
  Vector<GeomPoint> breakPoints = new Vector<>();

  breakPoints.add(points.firstElement());

  for (int i = 1; i < n - 1; i++) {
      GeomPoint begin = points.get(i - 1);
      GeomPoint end = points.get(i + 1);
      Point P = points.get(i);
      double angle = ConnectionCalculations.angle(begin, P, end);

      if (angle < constants.getMaxAngle()
        && angle > constants.getMinAngle()) {
          lessThenPI++;
          if (lessThenPI > constants.getMaximalNumberOfPoints()) {
            // ovo onda nije ni pologon ni prava
            // maximalNumberOfPoints tacaka je
            // na malom rastojanju
            // a ugao izmedju njih nije blizu opruzenog
            // zakljucujemo da nije prava ili pologon
              return null;
          }
              breakPoints.add(P);
      } else {
        lessThenPI = 0;
      }
  }

  breakPoints.add(points.lastElement());
  if (breakPoints.size() > 2) {
      int size = breakPoints.size();
      GeomPoint first = breakPoints.firstElement();
      GeomPoint last = breakPoints.lastElement();
      GeomPoint secondLast = breakPoints.elementAt(size - 2);

      double angle = ConnectionCalculations.angle(secondLast, last,
     first);
      if (first.distance(last) < constants.getMinimalDistance()
        || Math.abs(angle) > constants.getMaxAngle()) {
        // poslednja tacka je na maloj udaljenosti od prve
```

```
            // ili je ugao blizu opruzenog
            // izbacujemo poslednju tacku
              breakPoints.removeElementAt(size - 1);
        }

        if (size == 3) {
            GeometricObject triangle = new Triangle(breakPoints);
            triangle.setConstants(constants);
            return triangle;
        }

        GeometricObject polygon = new Polygon(breakPoints);
        polygon.setConstants(constants);
        return polygon;
    }
    // ugao izmedju svake dve uzastopne tacke je blizu opruzenog
    // prepoznajemo pravu
    GeometricObject line = new Line(breakPoints.firstElement(),
        breakPoints.lastElement());
    line.setConstants(constants);
    return line;
}
```

isPolyLine

Biografija kandidata

Milica G. Selaković rođena je u Užicu 17. septembra 1993. godine. Osnovnu školu ,,Slobodan Sekulić" i gimnaziju ,,Užička gimnazija" završila je u Užicu sa odličnim uspehom. Tokom osnovne i srednje škole bila je dobitnik velikog broja nagrada na takmičenjima iz matematike, fizike i hemije.

Studije na smeru Računarsvo i informatika na Matematičkom fakultetu upisala je školske 2012/13. godine, a diplomirala je u junu 2016. godine sa prosečnom ocenom 9,91. Tokom osnovnih studija bila je dobitnik nagrade Matematičkog fakulteta za izuzetan uspeh tokom studija, kao i stipendista fonda za mlade talente. Master studije na smeru Računarstvo i informatika upisala je školske 2016/17. godine. Tokom master studija radila je tromesečnu praksu u kompaniji Facebook.

Od oktobra 2016. do septembra 2018. bila je zaposlena na Matematičkom fakultetu kao saradnik u nastavi. Tokom rada na fakultetu držala je vežbe iz pet predmeta.