SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1552

Pronalaženje, prepoznavanje i praćenje razdjelne linije iz perspektive vozača

Petar Palašek

Zagreb, lipanj 2010.

Zahvaljujem doc. dr. sc. Siniši Šegviću na mentorstvu i pomoći pruženoj pri pisanju završnog rada.

SADRŽAJ

1. Uvod

Računalni vid se u prometu primjenjuje za rješavanje mnogih problema, poput automatiziranog pronalaženja i karakteriziranja prometne signalizacije, najčešće u svrhu poboljšanja sigurnosti sudionika u prometu. Razvijeni su mnogi sustavi za pomoć vozačima pri vožnji koji detektiraju trenutnu poziciju vozila na prometnom traku, te koristeći model ceste mogu upozoriti vozača o npr. neočekivanom skretanju vozila s traka. Neki od postupaka za pronalaženje i modeliranje ceste i prometnog traka opisani su u [3], [5] i [9].

Jedan od problema koji se može svrstati u ranije spomenute je i pronalaženje i prepoznavanje razdjelne linije na cesti, što je tema ovog završnog rada. Znanje o poziciji i vrsti razdjelne linije moglo bi se također iskoristiti za poboljšanje sigurnosti u prometu, a moglo bi pomoći i kod detekcije prometnog traka i kod kartiranja tlocrta prometnica [7].

Prije nego što krenemo s detektiranjem razdjelne linije trebamo sliku koju dobivamo s kamere obraditi tako da bude prikladnija za provođenje algoritama razvijenih u sklopu ovog završnog rada.

Najprije ćemo iz ulazne slike snimljene iz perspektive vozača pomoću inverzne perspektivne transformacije dobiti sliku kakvu bi imali da cestu snimamo iz ptičje perspektive (tj, kakvu bismo dobili da smo koristili kameru koja je okomita na cestu). Time ćemo postići paralelnost i konstantnu udaljenost linija na cesti i tako olakšati daljnju analizu slike. Inverzna perspektivna transformacija objašnjena je u [13].

Nakon toga ćemo binarizirati odziv upravljivog filtra temeljenog na drugoj derivaciji Gaussove funkcije na dobivenoj inverzno perspektivnoj slici, te time boju svih piksela koji leže na linijama postaviti na bijelu, dok će ostali ostati crni. Korištenje upravljivih filtara za detektiranje linija objašnjeno je u [10] i [1], a binarizacija odziva opisana je u [10].

Osnovna ideja predloženog pristupa detekciji razdjelne linije je sljedeća: najprije je potrebno odabrati skup točaka koje se nalaze na razdjelnoj liniji na ulaznoj slici i pomoću njih aproksimirati liniju parabolom. Model parabole za aproksimaciju linije

odabran je umjesto modela pravca kako bi aproksimacija bila što točnija i kako bi se kasnije olakšalo prepoznavanje vrste linije. Budući da razdjelna linija može biti i dvostruka, moguće je da ćemo dobiti i dvije aproksimirajuće parabole, svaku za jednu liniju. Koristeći tu, odnosno, te parabole, možemo jednostavnim postupkom odrediti o kojoj se točno vrsti linije radi. Također, koristeći aproksimacijske parabole možemo pretpostaviti u kojem će se smjeru kretati linija u budućnost i tako riješiti i problem praćenja linije.

U prva dva poglavlja opisane su ideje iza postupaka za pronalaženje, prepoznavanje i praćenje razdjelne linije, zajedno s detaljnim objašnjenjem upotrebljenih matematičkih postupaka. Slijedi poglavlje u kojem su opisani detalji iz programske implementacije, poglavlje u kojem se analiziraju eksperimentalni rezultati i zaključak.

2. Pronalaženje razdjelne linije

2.1. Pronalaženje točaka koje leže na razdjelnoj liniji

Kao što je objašnjeno u uvodu, slika koju dobivamo na ulazu je binarizirana verzija inverzne perspektivne slike s kamere postavljene na automobilu. Na toj su slici bijeli pikseli oni za koje smo nakon obrade upravljivim filtrom zaključili da leže na liniji na cesti, dok su ostali pikseli crni. Za daljnju analizu potrebno je odabrati skup točaka koji ćemo koristiti za aproksimiranje linije parabolom.

Budući da nas u okviru ovog rada zanima samo razdjelna, tj. središnja linija, nije potrebno promatrati cijelu ulaznu sliku, već možemo promatrati samo dio slike koji ćemo nazvati područje interesa (eng. *Region Of Interest, ROI*). Horizontalni pomak simetrale područja interesa određen je parametrom x_s, dok širinu zadajemo parametrom d_{ROI}.

Na početku, područje interesa omeđeno je pravcima

$$y_1 = x_s - d_{ROI} \tag{2.1}$$

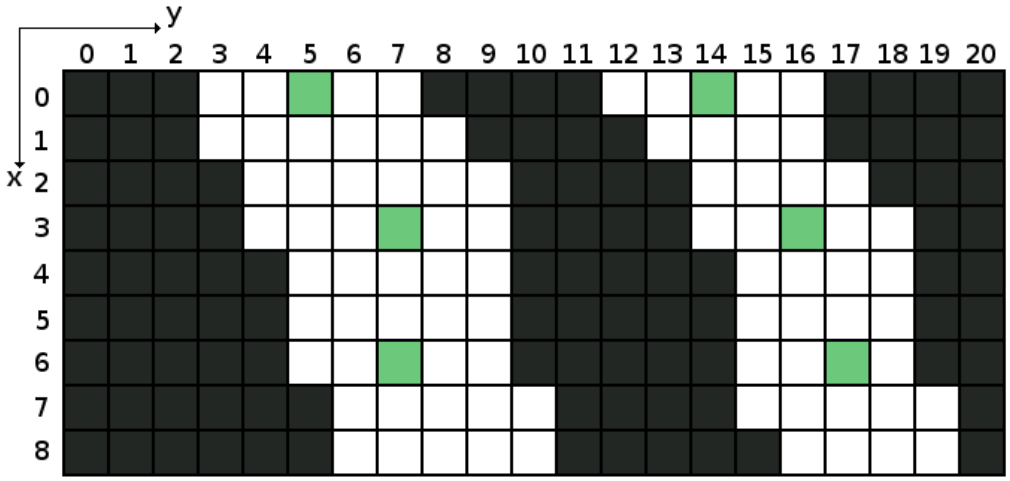
i

$$y_2 = x_s + d_{ROI}, \tag{2.2}$$

dok se kasnije određuje na način opisan u poglavlju 3.2.

Na slici 2.1 prikazani su uvećani pikseli dijela hipotetskog područja interesa na kojem se nalaze dvije linije. Analiziramo prijelaze iz crne u bijelu i iz bijele u crnu boju za svaki n-ti red u području interesa. Koordinate svake točke za koju se odredi da se nalazi na sredini bijelog područja u trenutnom redu dodajemo u skup odabranih točaka. U ovom primjeru, točke za skup smo birali iz svakog 3. reda, a odabrane točke označene su zelenom bojom.

Detaljnije objašnjenje postupka kojim biramo točke nalazi se u poglavlju o implementaciji (4.2.1.).



Slika 2.1: Dio hipotetskog područja interesa i odabrani skup točaka

2.2. Aproksimiranje parabole kroz zadani skup točaka

Želimo zadani skup točaka aproksimirati parabolom i želimo da ta aproksimirana parabola bude što je bolja moguća. Možemo reći da parabola dobro aproksimira skup točaka ako prolazi dovoljno blizu svih točaka iz skupa. Kako bi ta parabola najbolje aproksimirala skup, moramo postići da udaljenosti svih točaka od nje budu minimalne. To postižemo metodom najmanjih kvadrata.

2.2.1. Metoda najmanjih kvadrata

Metodom najmanjih kvadrata minimiziramo sumu kvadrata udaljenosti točaka iz skupa koji aproksimiramo do aproksimirajuće parabole. Kako bismo pojednostavili račun, ne tražimo pravu udaljenost od točke do parabole, već vertikalnu udaljenost; za parabolu zadanu jednadžbom $y = a_2x^2 + a_1x + a_0$, vertikalna udaljenost točke $T(x_t, y_t)$ od te parabole iznosi

$$d_v = a_2x_t^2 + a_1x_t + a_0 - y_t. \tag{2.3}$$

Naš je cilj pronaći za koje je parametre a_2, a_1, a_0 suma kvadrata vertikalnih udaljenosti svih točaka od parabole minimalna, odnosno, tražimo vektor $\vec{a}$ za koji vrijedi:

$$\vec{a} = \underset{\vec{a}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(a_2x_i^2 + a_1x_i + a_0 - y_i\right)^2. \tag{2.4}$$

U matričnom obliku to možemo zapisati kao:

$$\vec{a} = \underset{\vec{a}}{\operatorname{argmin}} \left(\begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right), \tag{2.5}$$

odnosno

$$\vec{a} = \underset{\vec{a}}{\operatorname{argmin}} \left(\mathbf{X}\vec{a} - \vec{y}\right). \tag{2.6}$$

Vektor $\vec{a}$ dobit ćemo iz jednadžbe

$$\vec{y} = \mathbf{X}\vec{a}, \tag{2.7}$$

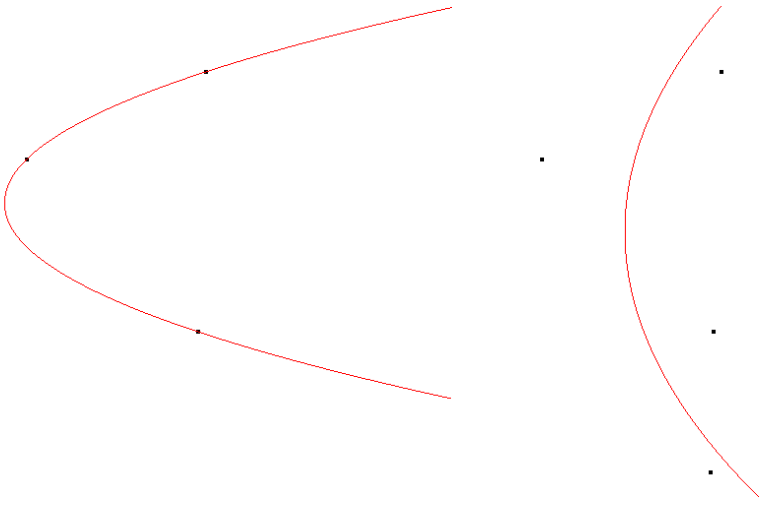
generaliziranim inverzom matrice $\mathbf{X}$:

$$\begin{aligned} \vec{y} &= \mathbf{X}\vec{a} \quad / \cdot \mathbf{X}^T \\ \mathbf{X}^T\vec{y} &= \mathbf{X}^T\mathbf{X}\vec{a} \quad / \cdot (\mathbf{X}^T\mathbf{X})^{-1} \\ \vec{a} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\vec{y} \end{aligned} \tag{2.8}$$

Metoda najmanjih kvadrata daje dobre rezultate kad u skupu točaka ne postoje tzv. *outlieri*, točke koje ne pripadaju populaciji koju želimo modelirati.

Uzmimo za primjer skup od tri točke i aproksimirajmo ga parabolom koristeći gore opisanu metodu. Dobit ćemo parabolu koja prolazi kroz sve tri točke, baš kako smo i očekivali. No, ako u taj isti skup dodamo još jednu točku koja ne leži na ranije dobivenoj paraboli (odudara od ostatka skupa, *outlier*), dobit ćemo potpuno drugačiju parabolu od ranije dobivene, za koju je moguće da neće prolaziti kroz nijednu od točaka iz skupa. Opisani primjer prikazan je na slici 2.2.

Na konačne vrijednosti parametara parabole utječu sve točke iz skupa, ali udaljenije točke (*outlieri*) utječu više od ostalih.



(a) Tri zadane točke (b) Tri zadane točke i jedan *outlier*

Slika 2.2: Aproksimiranje parabole kroz zadani skup točaka metodom najmanjih kvadrata

Problem *outliera* rješavamo tzv. RANSAC [4] algoritmom.

2.2.2. Algoritam RANSAC

Kako bismo osigurali da skup točaka koji želimo aproksimirati parabolom ne sadrži *outliere*, odnosno, da se sastoji samo od *inliera*, koristimo algoritam RANSAC (Random Sample Consensus).

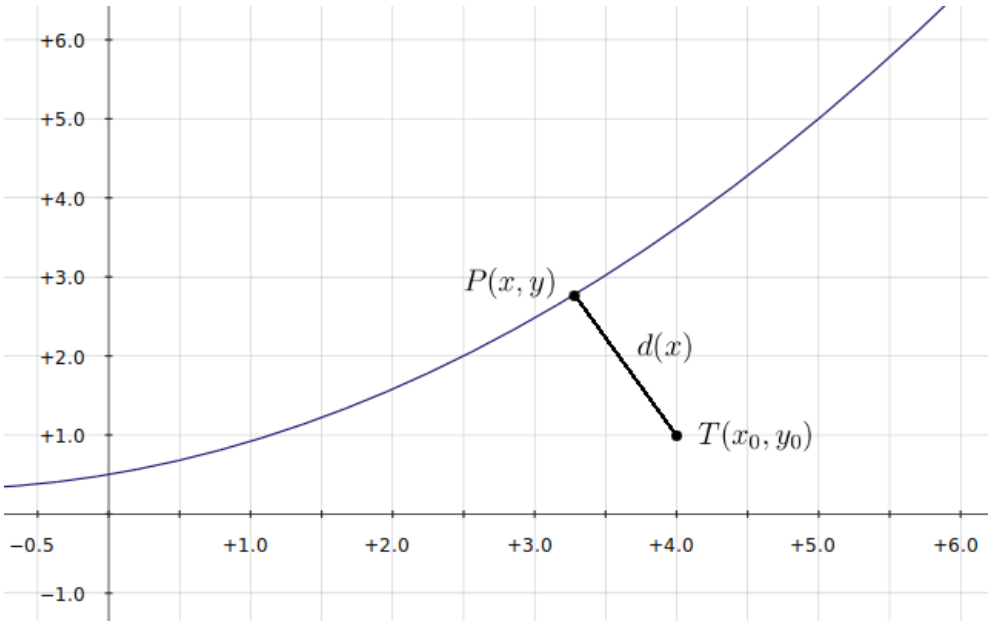
Općenito, RANSAC generira n modela koristeći nasumično odabrane podskupove ulaznog skupa, uspoređuje ih i vraća najbolji.

U ovom slučaju, algoritmom RANSAC u svakoj od n iteracija nasumično odabiremo po tri točke iz skupa i kroz njih metodom najmanjih kvadrata aproksimiramo parabolu (generiramo model parabole). Ako je zadnji generirani model bolji od svih prethodno generiranih modela, trenutni model pamtimo kao najbolji i nastavljamo algoritam.

Kriteriji po kojima odlučujemo je li neki model bolji od drugog su broj točaka koje su od modela udaljene za manje od d i prosječna vrijednost tih udaljenosti. Točke koje su udaljene manje od d od najbolje rangiranog modela su *inlieri* skupa. Ako dva modela imaju isti broj *inliera*, bolji je onaj kojem je prosječna vrijednost udaljenosti manja.

Udaljenost točke od parabole

Udaljenost točke $T(x_0, y_0)$ od parabole $y = a_2x^2 + a_1x + a_0$ računamo tako da na paraboli odaberemo točku $P(x, y)$ i minimiziramo duljinu dužine $\overline{TP}$.



Slika 2.3: Prikaz parabole, odabrane točke T i njihove udaljenosti $\overline{TP}$

Duljina dužine $\overline{TP}$ ovisi o x koordinati točke P i iznosi:

$$\begin{aligned} d(x) &= \sqrt{(x_0 - x)^2 + (y_0 - y)^2} = \\ &= \sqrt{(x_0 - x)^2 + (y_0 - a_2x^2 - a_1x - a_0)^2} = |y_0 - a_0 = q| = \\ &= \sqrt{(x_0 - x)^2 + (q - a_2x^2 - a_1x)^2} = \\ &= \sqrt{a_2^2x^4 + 2a_2a_1x^3 + x^2(1 + a_1^2 - 2a_2q) - 2x(x_0 + a_1q) + x_0^2 + q^2} \end{aligned} \tag{2.9}$$

Kako bismo izračunali minimalnu vrijednost duljine dužine $\overline{TP}$, odnosno, dobili udaljenost točke T od parabole, gornji izraz za $d(x)$ trebamo derivirati po x i izjednačiti s nulom. Kako bismo olakšali račun, derivirat ćemo samo izraz ispod korijena, jer će korijen iz minimalne vrijednosti rezultirati minimalnom vrijednošću:

$$\frac{d}{dx}d(x)^2 = 4a_2^2x^3 + 6 * a_2a_1x^2 + 2x(1 + b^2 - 2a_2q) - 2(x_0 + a_1q) = 0 \tag{2.10}$$

Dobili smo polinom trećeg stupnja čije nultočke predstavljaju potencijalne x koordinate točke P na paraboli. Udaljenost točke T od parabole pronaći ćemo tako da uzmemo najmanju vrijednost izraza $d(x)$ koju dobijemo kad u njega uvrstimo svaku od maksimalno tri nultočke gornjeg polinoma.

Nultočke ćemo izračunati koristeći Cardanovu formulu [2][6][8][11], koja je detaljno objašnjena u sljedećem poglavlju.

Konačno, nakon što smo RANSAC-om pronašli najbolji model i uspjeli odrediti najvjerojatniji skup *inliera*, metodom najmanjih kvadrata možemo odrediti parametre parabole za koju ćemo reći da najbolje aproksimira početni skup točaka.

2.2.3. Cardanova formula

Dana je jednadžba trećeg stupnja oblika

$$a_3x^3 + a_2x^2 + a_1x + a_0 = 0. \tag{2.11}$$

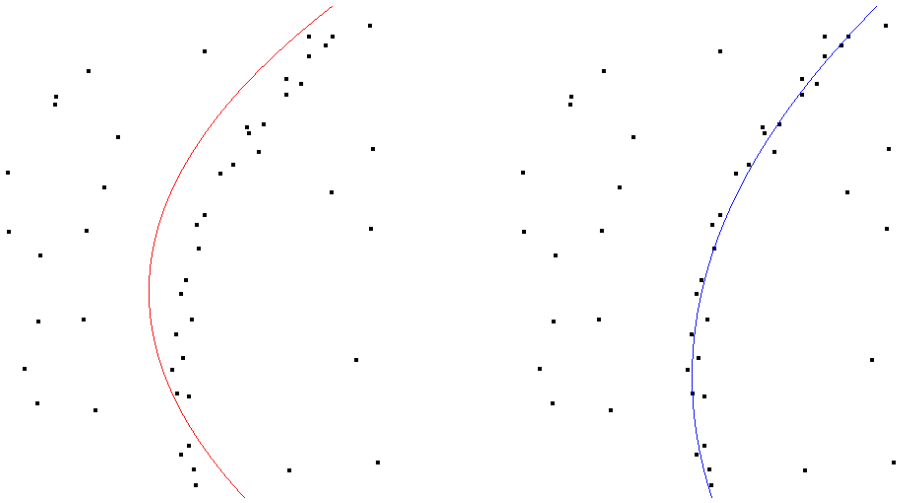
Dijeljenjem jednadžbe s a_3 dobivamo normiranu jednadžbu

$$x^3 + ax^2 + bx + c = 0, \tag{2.12}$$

gdje je $a = \frac{a_2}{a_3}$, $b = \frac{a_1}{a_3}$ i $c = \frac{a_0}{a_3}$. Uvođenjem supstitucije $x = y - \frac{a}{3}$ rješavamo se kvadratnog člana i dobivamo tzv. kanonski oblik jednadžbe trećeg stupnja:

$$y^3 + py + q = 0, \tag{2.13}$$

gdje je $p = b - \frac{a^2}{3}$, $q = \frac{2a^3}{27} - \frac{ab}{3} + c$.



(a) Metoda najmanjih kvadrata (b) Algoritam RANSAC

Slika 2.4: Usporedba aproksimiranja parabole kroz 50 ručno odabranih točaka korištenjem metode najmanjih kvadrata i algoritma RANSAC

Pretpostavimo da je rješenje ove jednadžbe oblika $y = u + v$. Iz toga slijedi da mora vrijediti

$$(u+v)^3 + p(u+v) + q = 0, \tag{2.14}$$

odnosno,

$$(3uv + p)(u+v) + (u^3 + v^3 + q) = 0. \tag{2.15}$$

Ako odaberemo da je rješenje $y = u + v$ takvo da zadovoljava uvjet

$$3uv + p = 0, \tag{2.16}$$

iz jednadžbe 2.15 dobivamo:

$$u^3 + v^3 = -q. \tag{2.17}$$

Time smo rješavanje početne jednadžbe sveli na rješavanje sustava dviju jednadžbi:

$$\begin{cases} u^3 + v^3 = -q \\ uv = -\frac{p}{3}. \end{cases} \tag{2.18}$$

Umjesto gornjeg sustava promatrat ćemo sustav

$$\begin{cases} u^3 + v^3 = -q \\ u^3 v^3 = -(\frac{p}{3})^3, \end{cases} \tag{2.19}$$

koji nije ekvivalentan sustavu 2.18, ali svako rješenje sustava 2.18 zadovoljava i sustav 2.19. To znači da ćemo prvi sustav riješiti tako da od rješenja drugog sustava odaberemo ona koja zadovoljavaju drugu jednadžbu prvog sustava.

Prema Viètеovim formulama, za nultočke kvadratne jednadžbe $y = ax^2 + bx + c$ vrijede relacije $x_1 + x_2 = -\frac{b}{a}$ i $x_1 x_2 = \frac{c}{a}$. Iz toga zaključujemo da su u^3 i v^3 iz gornjeg sustava jednadžbi nultočke kvadratne jednadžbe

$$t^2 + qt - \left(\frac{p}{3}\right)^3 = 0. \tag{2.20}$$

Slijedi da je

$$u = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}, \tag{2.21}$$

$$v = \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}. \tag{2.22}$$

Rješenja u i v koja zadovoljavaju uvjet $uv = -\frac{p}{3}$ iz 2.18 tvore rješenja početne jednadžbe 2.13 i dana su izrazom

$$y = u + v, \tag{2.23}$$

odnosno

$$y = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}. \tag{2.24}$$

Gornja formula 2.24 naziva se Cardanova formula (ime je dobila prema talijanskom matematičaru Gerolamu Cardanu koji ju je 1545. godine objavio u svom djelu *Artis Magnae*)[6].

Kako bismo olakšali računanje, najprije ćemo izračunati u po formuli 2.21, a v tada možemo dobiti iz ranije zadanog uvjeta 2.16. Nakon toga možemo sva tri rješenja početne jednadžbe u kanonskom obliku (2.13) zapisati eksplicitno:

$$y_1 = u + v \tag{2.25}$$

$$y_2 = u * \varepsilon + v * \varepsilon^2 \tag{2.26}$$

$$y_3 = u * \varepsilon^2 + v * \varepsilon \tag{2.27}$$

gdje ε predstavlja treći korijen iz 1,

$$\varepsilon = \frac{-1}{2} + \frac{\sqrt{3}}{2}\mathrm{i}. \tag{2.28}$$

Konačno, da bi dobili rješenja početne jednadžbe, svakom od gornjih rješenja moramo oduzeti $\frac{a}{3}$ zbog supstitucije s početka:

$$x_i = y_i - \frac{a}{3}, i \in \{1, 2, 3\}. \tag{2.29}$$

Diskriminanta

U okviru ovog rada ograničit ćemo se samo na traženje realnih nultočaka, budući da nas kompleksna rješenja ne zanimaju.

Diskriminanta jednadžbe trećeg stupnja zadane u kanonskom obliku iznosi

$$D = \left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3. \tag{2.30}$$

Radi lakšeg računa uvest ćemo supstitucije $Q = \frac{q}{2}$ i $P = \frac{p}{3}$, pa diskriminantu možemo pisati kao

$$D = Q^2 + P^3. \tag{2.31}$$

Ovisno o vrijednosti diskriminante razlikujemo 3 različita slučaja:

1. $D > 0$, postoji 1 realno i 2 konjugirano kompleksna rješenja,
2. $D = 0$, sva 3 rješenja su realna, barem 1 je dvostruko,
3. $D < 0$, sva 3 rješenja su realna i različita, tzv. *casus irreductibilis*.

1. slučaj, $D > 0$

U ovom slučaju nema problema i jedino realno rješenje dobivamo korištenjem formule 2.25.

2. slučaj, $D = 0$

Ovdje možemo razlikovati dva podslučaja, kad je $Q = 0$ i kad je $Q \neq 0$.

Kod prvog podslučaja postoji samo jedno rješenje i ono je trostruko:

$$y_1 = y_2 = y_3 = 0. \tag{2.32}$$

U drugom podslučaju imamo jedno jednostruko i jedno dvostruko rješenje koje dobivamo iz:

$$u = \sqrt[3]{-Q} \tag{2.33}$$

$$y_1 = 2u \tag{2.34}$$

$$y_2 = y_3 = -u \tag{2.35}$$

Zašto je to tako lako možemo zaključiti iz 2.25, 2.26 i 2.27.

3. slučaj, $D < 0$, casus irreductibilis

Ovaj slučaj, poznat po nazivu *casus irreductibilis* (nesvodljivi slučaj), je najzanimljiviji jer se kod njega javlja problem vađenja korijena iz negativnog broja. Baš je zbog tog slučaja 1545. godine Cardano definirao i uveo skup kompleksnih brojeva.

Znamo da svaki kompleksni broj $z = x+\mathrm{i}y$ možemo prikazati u trigonometrijskom obliku

$$z = r(\cos\varphi + \mathrm{i}\sin\varphi). \tag{2.36}$$

Također, znamo da n-ti korijen iz kompleksnog broja možemo dobiti formulom

$$\sqrt[n]{z} = \sqrt[n]{r}\left(\cos\left(\frac{\varphi+2k\pi}{n}\right) + \mathrm{i}\sin\left(\frac{\varphi+2k\pi}{n}\right)\right), \qquad k \in \{0,...,n-1\} \tag{2.37}$$

Kako vrijedi da je $D < 0$, iz 2.21 možemo dobiti da je

$$u^3 = z = -Q + \mathrm{i}\sqrt{-D}. \tag{2.38}$$

Modul r i argument φ kompleksnog broja z možemo izračunati izrazima

$$|z| = r = \sqrt{Q^2 + \sqrt{-D}^2} = \sqrt{Q^2 - (P^3 + Q^2)} = \sqrt{-P^3} \tag{2.39}$$

$$arg(z) = \varphi = \arccos\left(\frac{-Q}{\sqrt{P^3}}\right). \tag{2.40}$$

Modul r' i argument φ' 3. korijena iz gornjeg kompleksnog broja z, odnosno modul i argument broja u, iznosit će

$$|u| = |\sqrt[3]{z}| = \sqrt[3]{r} = r' = \sqrt{-P} \tag{2.41}$$

$$arg(u) = \varphi' = \frac{\varphi}{3}. \tag{2.42}$$

Istim postupkom možemo iz 2.22 dobiti:

$$v^3 = w = -Q - \mathrm{i}\sqrt{-D}, \tag{2.43}$$

te iz toga izračunati modul i argument 3. korijena iz kompleksnog broja w, odnosno modul i argument broja v. Možemo primjetiti da su moduli korijena u i v jednaki, ali im se argumenti razlikuju:

$$|v| = |\sqrt[3]{w}| = \sqrt[3]{r} = r' = \sqrt{-P} \tag{2.44}$$

$$arg(v) = \frac{2\pi - \varphi}{3} = \frac{2\pi}{3} - \varphi'. \tag{2.45}$$

Sada sve korijene iz 2.38 možemo zapisati kao:

$$u_1 = r' \left(\cos\left(\varphi'\right) + \sin\left(\varphi'\right)\right), \tag{2.46}$$

$$u_2 = r' \left(\cos\left(\frac{2\pi}{3} + \varphi'\right) + \sin\left(\frac{2\pi}{3} + \varphi'\right)\right), \tag{2.47}$$

$$u_3 = r' \left(\cos\left(\frac{4\pi}{3} + \varphi'\right) + \sin\left(\frac{4\pi}{3} + \varphi'\right)\right), \tag{2.48}$$

te sve korijene iz 2.43 kao:

$$v_1 = r' \left(\cos\left(\frac{2\pi}{3} - \varphi'\right) + \sin\left(\frac{2\pi}{3} - \varphi'\right)\right), \tag{2.49}$$

$$v_2 = r' \left(\cos\left(\frac{4\pi}{3} - \varphi'\right) + \sin\left(\frac{4\pi}{3} - \varphi'\right)\right), \tag{2.50}$$

$$v_3 = r' \left(\cos\left(2\pi - \varphi'\right) + \sin\left(2\pi - \varphi'\right)\right). \tag{2.51}$$

Od svih parova gore navedenih korijena samo 3 para zadovoljavaju uvjet 2.16: (u_1, v_3), (u_2, v_2) i (u_3, v_1). Uvrštavanjem tih parova u formulu 2.23 dobivamo tri realna rješenja jednadžbe 2.13:

$$y_1 = 2r' \cos\varphi', \tag{2.52}$$

$$y_2 = 2r' \cos\left(\frac{2\pi}{3} - \varphi'\right), \tag{2.53}$$

$$y_3 = 2r' \cos\left(\frac{2\pi}{3} + \varphi'\right). \tag{2.54}$$

Time smo obradili sva tri slučaja i objasnili kako dobiti sva realna rješenja jednadžbe 2.13.

Na kraju ne smijemo zaboraviti od svih dobivenih rješenja oduzeti $\frac{a}{3}$, zbog supstitucije s početka:

$$x_i = y_i - \frac{a}{3}, i \in \{1, 2, 3\}. \tag{2.55}$$

3. Prepoznavanje i praćenje detektirane linije

3.1. Prepoznavanje detektirane linije

Nakon što smo estimirali koeficijente parabole koja aproksimira razdjelnu liniju na cesti, iskoristit ćemo jednostavni algoritam za odlučivanje radi li se o punoj ili crtkanoj liniji.

Reći ćemo da je linija puna ako nigdje nije prekinuta, tj. ako su svi elementi binarizirane slike koji leže ispod aproksimirajuće parabole bijeli. Vidimo da odluku o vrsti detektirane linije možemo donijeti prebrojavanjem crnih elemenata binarizirane slike ispod parabole; ako je broj crnih elemenata ispod parabole veći od nekog zadanog praga, zaključit ćemo da je detektirana linija crtkana, odnosno, ako je broj crnih elemenata manji od praga, detektirana linija je puna.

3.2. Praćenje detektirane linije

Kao što je već objašnjeno u poglavlju 2.1., skup točaka koje koristimo za generiranje parabole koja aproksimira razdjelnu liniju tražimo u području nazvanom područje interesa. Kako linije na slici pribavljenoj iz kamere mijenjaju svoje pozicije kroz vrijeme, ovisno o trenutnoj lokaciji automobila na cesti, može se dogoditi da razdjelna linija izađe iz područja interesa, te u tom slučaju liniju nećemo moći detektirati. Da bismo to izbjegli, područje interesa mora se također mijenjati kroz vrijeme i to na način koji će osigurati da se razdjelna linija u svakom trenutku nalazi unutar područja interesa.

Praćenje detektirane linije realizirano je pomicanjem granica područja interesa ovisno o lokaciji parabole koja aproksimira detektiranu liniju. Ako je na trenutnom

okviru (trenutnom *frameu*) aproksimirajuća parabola

$$y = a_2x^2 + a_1x + a_0, \tag{3.1}$$

područje interesa na sljedećem okviru (sljedećem *frameu*) bit će ograničeno parabolama

$$y_1 = a_2x^2 + a_1x + a_0 - d_{ROI} \tag{3.2}$$

i

$$y_2 = a_2x^2 + a_1x + a_0 + d_{ROI}, \tag{3.3}$$

gdje je d_{ROI} zadani parametar širine područja interesa.

Vidimo da su pravci iz poglavlja 2.1. koji na početku omeđuju područje interesa zapravo parabole s koeficijentima $a_2 = 0$, $a_1 = 0$ i $a_0 = x_s \pm d_{ROI}$.

4. Detalji programske implementacije

Programska implementacija izvedena je u programskom jeziku C++, a može se isprobati u sklopu CVSH (*Computer Vision SHell*) ljuske s navedenim argumentima:

`-sf="ulaznaDatoteka.wmv" -a=ppalasek -c="90 22 2 -10 (-533,0) (605,0) (309,296) 7 20 fixed:ipv-panasonic-720"`.

Obrada slike pokreće se pozivom metode `process`:

```
void alg_ppalasek::process(
  const img_vectorAbstract& imgs,
  const win_event_vectorAbstract& events,
  int msDayUtc, int frame)
```

Metodu `process` možemo opisati sljedećim pseudokodom:

```
void process(...)
  if (prviPozivFunkcije):
    prosla parabola = pravac y=xs; //y=0*x^2+0*x+xs
  obaviPretprocesiranje(ulaznaSlika,slika);
  //tocke koje leze na nekoj crti u podrucju interesa:
  sveTocke=pronadjiTocke(slika,koefProsleParabole,droi);
  //tocke kroz koje smo aproksimirali najbolju parabolu:
  tockeNaGlavnoj=ransac(sveTocke,300,koef);
  crtajParaboluIPrepoznajVrstuCrte(koef);
  if (sveTocke.size()-tockeNaGlavnoj.size()>5):
    ostaleTocke=sveTocke-tockeNaGlavnoj;
    if (pronadjiParalelnu(koef,ostaleTocke,koef2)):
      crtajParaboluIPrepoznajVrstuCrte(koef2);
  koefProsleParabole=koef;
```

4.1. Pretprocesiranje

Kao što je objašnjeno u uvodu, prije nego što krenemo s detekcijom razdjelne linije, na ulaznoj ćemo slici napraviti inverznu perspektivnu transformaciju, koristeći upravljive filtre detektirati sve linije i na kraju dobivenu sliku binarizirati. Kako su ovi postupci vremenski dosta zahtjevni, ulaznu sliku ćemo skalirati nakon inverzne perspektivne transformacije.

Detaljnija objašnjenja algoritama korištenih kod pretprocesiranja mogu se pronaći u [13], [10] i [1].

Inverzna perspektivna transformacija

Inverznu perspektivnu sliku dobivamo pozivom metode `ext_ipt::process` kojoj predajemo ulaznu sliku `imgSrc` i koja rezultat transformacije sprema u `imgIPT`.

```
void ext_ipt::process(
  const img_wrap& imgSrc, img_wrap& imgIPT)
```

Na dobivenoj slici trebamo maknuti trokute koji nam stvaraju problem kod obrade upravljivim filtrom[10], što radimo metodom `ext_iptTriangles`:

```
void ext_iptTriangles(
  const img_wrap& imgIPT,
  const img_wrap& imgSteerResponse,
  const img_wrap& imgSteerOrientation,
  const int margin)
```

Primjer slike prije i nakon inverzne perspektivne transformacije prikazan je na slikama 4.1(a) i 4.1(b).

Za ubrzavanje inverzne perspektivne transformacije korišteno je rješenje iz [12], pa se u ext_ipp.cpp umjesto starog poziva `ipp::homography` koristi novi poziv `ext_pbosilj::homography8bpp`.

```
void ext_pbosilj::homography8bpp(
  const img_wrap& src,
  const boost::numeric::ublas::matrix<double>& Hi,
  img_wrap& dst)
```

Odziv upravljivog filtra

Računanje odziva upravljivog filtra na slici pokrećemo pozivom metode `process` iz razreda `ext_steer`, kojoj predajemo ulaznu sliku `imgSrc` i parametar debljine filtra `sigma`.

```
int ext_steer::process(
  const img_wrap& imgSrc,
  double sigma)
```

Nakon toga odziv možemo dobiti pozivom:

```
img_wrap& ext_steer::imgSteerResponse()
```

Primjer odziva upravljivog filtra prikazan je na slici 4.1(c).

Binarizacija

Odziv upravljivog filtra iz gornjeg odjeljka binariziramo metodom `ipp::binarize`.

```
int ipp::binarize(
  const img_wrap& imgSrc,
  double threshold, int margin,
  img_wrap& imgBinary)
```

Binarizirana slika će nakon poziva biti spremljena u varijabli `imgBinary`.

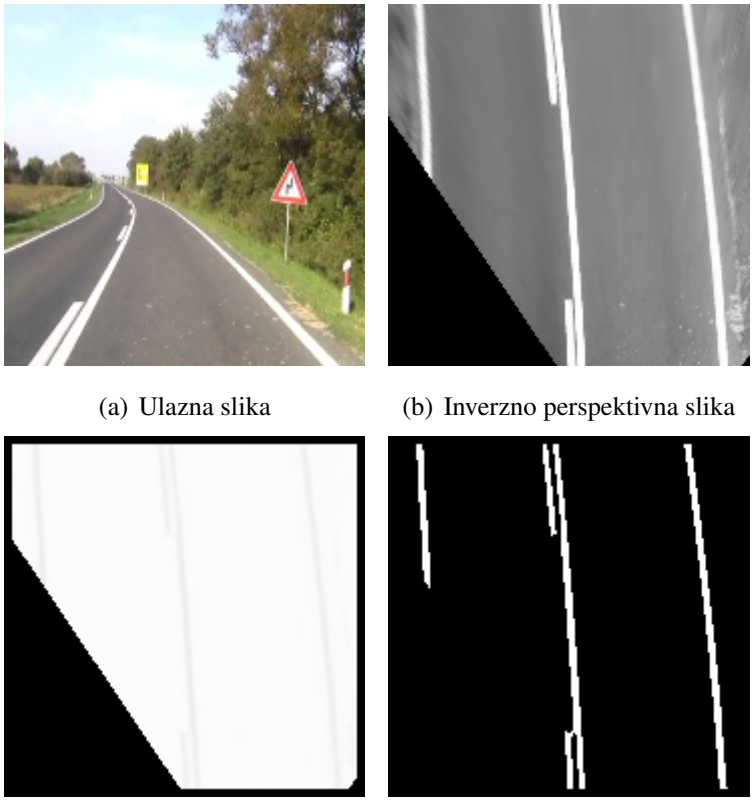
Primjer rezultata binarizacije prikazan je na slici 4.1(d).

4.2. Pronalaženje razdjelne linije

4.2.1. Pronalaženje točaka koje leže na razdjelnoj liniji

Točke koje leže na razdjelnoj liniji dobit ćemo metodom `pronadjiTocke` iz razreda `ext_ppalasek` kojoj predajemo sliku `slika` dobivenu ranije opisanim pretprocesiranjem ulazne slike i tri double vrijednosti `a`, `b` i `c` koje predstavljaju koeficijente parabole pronađene u prošlom *frameu*. Metoda vraća vektor parova koji predstavljaju x i y koordinate pojedine točke.

```
std::vector<std::pair<int, int> >
ext_ppalasek::pronadjiTocke(
  img_wrap &slika, double a, double b, double c)
```



(a) Ulazna slika (b) Inverzno perspektivna slika

(c) Odziv upravljivog filtra (d) Binarizirana slika

Slika 4.1: Primjer pretprocesiranja ulazne slike s prikazanim međurezultatima

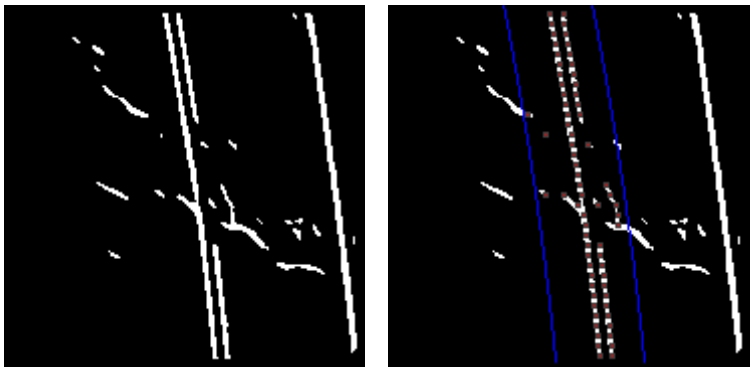
Kao što je objašnjeno u poglavljima 2.1. i 3.2., točke koje leže na razdjelnoj liniji tražimo u području interesa koje je omeđeno parabolama zadanim izrazima 3.2 i 3.3. Rezultat biranja skupa točaka na hipotetskom primjeru prikazan je na slici 2.1, a postupak kojim smo do njega došli može se opisati sljedećim pseudokodom:

```
vektorTocaka pronadjiTocke(...)
  c-=droi; //za lijevu granicu ROIa
  for (int x=0; x<visina; x+=3):
    y=x*x*a+x*b+c; boja=slika(x,y); prviY=y;
    for (int d=0; d<=2*droi; d+=1):
      if (slika(x,y+d)!=boja ili d==2*droi):
        if (boja==crna): prviY=y+d; //dosli smo iz crne
        else: //dosli smo iz bijele
          drugiY=y+d;
          vektorTocaka.add(x,prviY+(drugiY-prviY)/2);
        boja=slika(x,y+d); //nova boja
```

Treba obratiti pažnju na to da se koordinatni sustav u kojem crtamo parabole razlikuje od standardnog koordinatnog sustava u kojem su prikazani pikseli slike. Parabole koje tražimo na cesti vertikalne su orijentacije, pa je radi lakšeg računa koordinatni sustav u kojem ih prikazujemo zarotiran $90°$ u desno u odnosu na standardni koordinatni sustav kojem je ishodište u donjem lijevom uglu.

Radi jednostavnosti, u gore opisanom pseudokodu sav račun obavlja se u rotiranom koordinatnom sustavu s ishodištem u gorenjem lijevom uglu. U konkretnoj izvedbi potrebno je pretvarati koordinate iz jednog u drugi sustav. Također, korak kojim se krećemo po vertikalnoj osi u konkretnoj izvedbi nije 3, već 5 piksela.

Primjer pronalaženja točaka prikazan je na slici 4.2, na kojoj možemo vidjeti i smetnje uzrokovane sjenama na cesti. Na slici su plavim parabolama označene granice područja interesa.



(a) Binarizirana slika (b) Odabrane točke

Slika 4.2: Primjer pronalaženja točaka na razdjelnoj liniji uz smetnje uzrokovane sjenama

4.2.2. Metoda najmanjih kvadrata

Koeficijente parabole dobivene aproksimacijom odabranog skupa točaka metodom najmanjih kvadrata dobivamo pozivom metode `racunajKoefParabole` iz razreda `ext_ppalasek` kojoj predajemo vektor točaka i pokazivač na niz *doubleova* u koji će se spremiti izračunati koeficijenti.

```
void ext_ppalasek::racunajKoefParabole(
  std::vector<std::pair<int, int> >& tocke,
  double *koef)
```

Koeficijente parabole računamo prema formuli 2.8, koristeći sljedeće metode za rad s matricama iz razreda `ext_matrice`:

```
double ext_matrice::det3x3(    //determinanta matrice 3x3
  double *mat)
void ext_matrice::inv3x3(      //inverz matrice 3x3
  double *mat,
  double *rez)
void ext_matrice::transponiraj(//transponiranje matrice
  double *mat,                  //nxm
  int n, int m,
  double *rez)
void ext_matrice::mnozi(       //mnozenje matrica r1xs1 i
  double *a, double *b,        //s1xs2
  int r1, int s1, int s2,
  double *c)
```

4.2.3. Odabir točaka za estimiranje linije metodom RANSAC

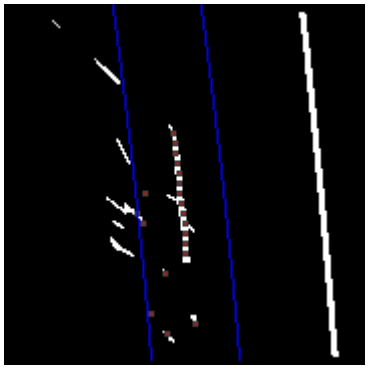
Točke koje ne pripadaju razdjelnoj liniji (*outlieri*, vanpopulacijski podaci) filtriramo postupkom RANSAC. *Inliere* zadanog skupa točaka i koeficijente parabole koja ih aproksimira dobivamo pozivom metode `ransac` iz razreda `ext_ppalasek`. Metodi `ransac` predajemo skup točaka, broj iteracija i pokazivač na niz *doubleova* u koje će se spremiti izračunati koeficijenti.

```
std::vector<int> ext_ppalasek::ransac(
  std::vector<std::pair<int, int> >& tocke,
  int brIteracija,
  double *koef)
```

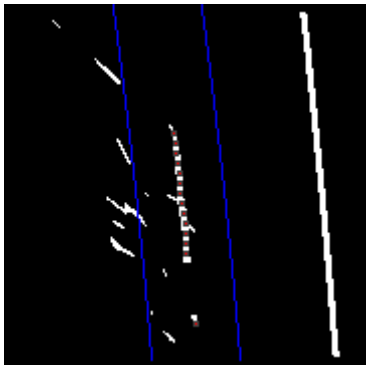
Metoda `ransac` opisana je sljedećim pseudokodom:

```
vektorTocaka ransac(...)
  for (int i=0; i<brIteracija; i+=1):
    randTocke=dajTriRandomTocke(skupTocaka);
    model=dajModelKrozTocke(randTocke);
    if (model bolji od bestModel):
      bestModel=model;
  koef=bestModel.getKoefParabole();
  return bestModel.getTockeNaParaboli();
```

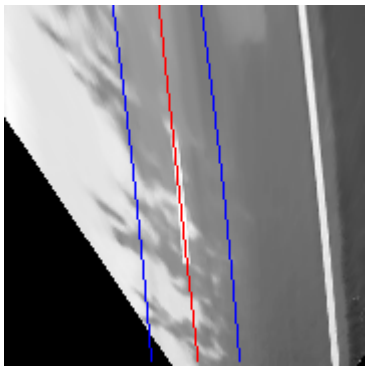
Primjer biranja *inliera* RANSAC-om i njihovog aproksimiranja parabolom prikazan je na slici 4.3.



(a) Skup točaka



(b) *Inlieri* odabrani RANSAC-om



(c) *Inlieri* aproksimirani parabolom

Slika 4.3: Primjer biranja *inliera* iz skupa točaka i njihova aproksimacija parabolom

Dva modela parabola uspoređuju se prema kriterijima opisanim u 2.2.2.. Udaljenost točke od parabole dobiva se postupkom opisanim u istom poglavlju, pozivom metode `udaljenost`. Metoda `udaljenost` prima parametre `a`, `b` i `c` koji predstavljaju koeficijente parabole i `x0` i `y0` koji predstavljaju koordinate točke.

```
double udaljenost(
  double a, double b, double c,
  double x0, double y0)
```

Metoda `udaljenost` koristi metodu `cardano` koja vraća broj rješenja polinoma 3. stupnja oblika $a_3x^3+a_2x^2+a_1x+a_0=0$ i sprema ih u vektor *doubleova* `solution`.

```
int cardano(
  double a3, double a2, double a1, double a0,
  std::vector<double>& solution)
```

4.2.4. Traženje paralelne linije

Budući da razdjelna linija može biti i dvostruka, trebamo provjeriti postoji li linija paralelna prvoj prepoznatoj. Kao što je opisano u pseudokodu s početka ovog poglavlja, paralelna linija se traži ukoliko je, nakon pronalaženja glavne linije, u skupu svih točaka ostalo više od 5 *outliera*.

Traženje paralelne linije pokrećemo pozivom metode `pronadjiParalelnu` kojoj predajemo koeficijente `a` i `b` prve prepoznate parabole $y = ax^2+bx+c$, skup točaka `ostaleTocke` i pokazivač na varijablu `c` u koju će se spremiti izračunati koeficijent c paralelne parabole.

```
bool ext_ppalasek::pronadjiParalelnu(
  double a, double b,
  std::vector<std::pair<int, int> >& ostaleTocke,
  double *c)
```

Metoda `pronadjiParalelnu` vraća `true` ukoliko je paralelna linija pronađena, odnosno `false` ako nije.

Postupak pronalaženja paralelne linije opisujemo sljedećim pseudokodom:

```
bool pronadjiParalelnu(...)
  for (tocka t : tocke):
    c=t.y-a*t.x*t.x-b*t.x; //c=y-a*x^2-b*x
    model=modelSKoeficijentima(a,b,c);
    if (model bolji od bestModel):
      bestModel=model;
  koef=bestModel.getKoefParabole();
  if (bestModel.getBrTocakaNaParaboli()>5):
    return true;
  return false;
```

Primjer traženja paralelne linije prikazan je na slici 4.4.

4.3. Prepoznavanje detektirane linije

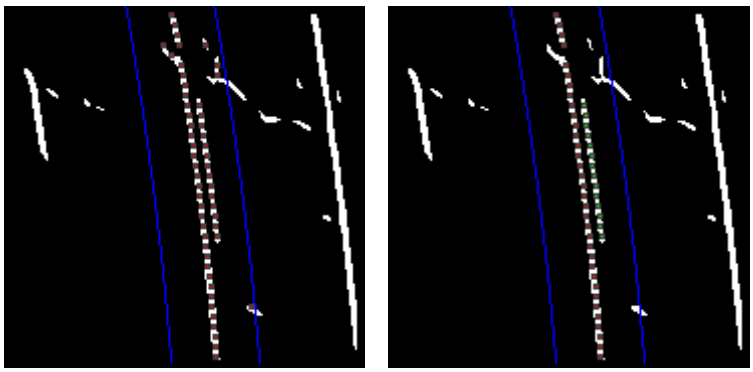
Vrstu detektirane linije određujemo postupkom opisanim u poglavlju 3.1., pozivom metode `prepoznajIOznaciLiniju` iz razreda `ext_ppalasek`. Metodi predajemo koeficijente `a`, `b` i `c` estimirane parabole, binariziranu sliku `slika`, odredište

`annRezultat` na kojem želimo označiti liniju i vrijednost `bojaTocke` koja označava boju kojom ćemo označiti točke na liniji.

```
int ext_ppalasek::prepoznajIOznaciLiniju(
  double a, double b, double c,
  const img_wrap& slika,
  win_ann& annRezultat,
  int bojaTocke);
```

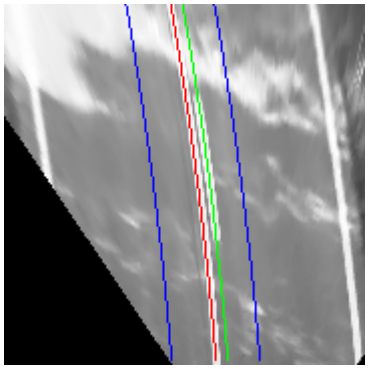
Slijedi pseudokod metode `prepoznajIOznaciLiniju`:

```
vrstaLinije prepoznajIOznaciLiniju(...)
  for (int x=5; x<visina; x+=5):
    int y=(int)(x*x*a+x*b+c);
    if (slika(x,y)==crna):
      brCrnih+=1;
      oznaciMaluTocku(annRezultat,x,y,bojaTocke);
    else:
      oznaciVelikuTocku(annRezultat,x,y,bojaTocke);
  if (brCrnih>3):
    return crtkana;
  return puna;
```



(a) Skup točaka

(b) *Inlieri* glavne (crvene točke) i paralelne linije (zelene točke)



(c) Aproksimirajuće parabole

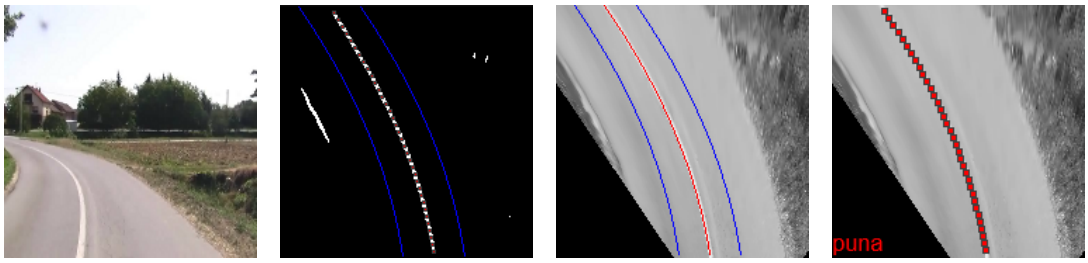
Slika 4.4: Primjer biranja *inliera* iz skupa točaka i njihova aproksimacija parabolom

5. Eksperimentalni rezultati

Ostvareno programsko rješenje isprobano je na nekoliko snimaka ceste pribavljenih kamerom postavljenom na automobilu. Rezultati pronalaženja, prepoznavanja i praćenja razdjelne linije prikazani su u nastavku.

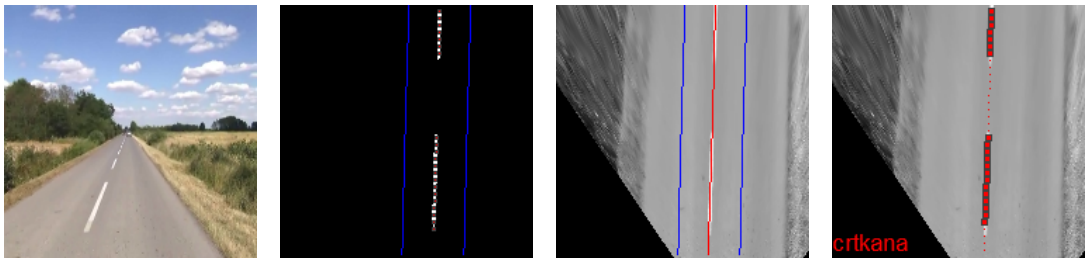
Uspješno pronalaženje i prepoznavanje razdjelne linije

Na slikama 5.1, 5.2 i 5.3 prikazani su redom primjeri uspješnog pronalaženja i prepoznavanja pune, crtkane i dvostruke razdjelne linije. Uspješno pronalaženje razdjelne linije rezultiralo je i uspješnim praćenjem u sljedećim *frameovima*.



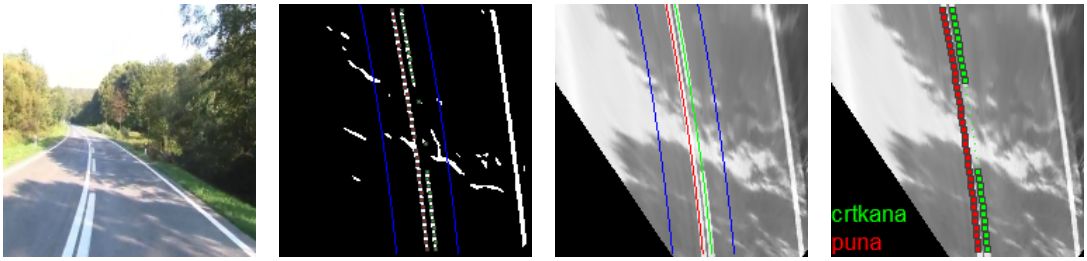
(a) (b) (c) (d)

Slika 5.1: Primjer uspješnog prepoznavanja pune razdjelne linije u zavoju: ulazna slika (a), skup točaka dobivenih RANSAC-om (b), parabola estimirana metodom najmanjih kvadrata (c), rezultat obrade (d)



(a) (b) (c) (d)

Slika 5.2: Primjer uspješnog prepoznavanja crtkane razdjelne linije



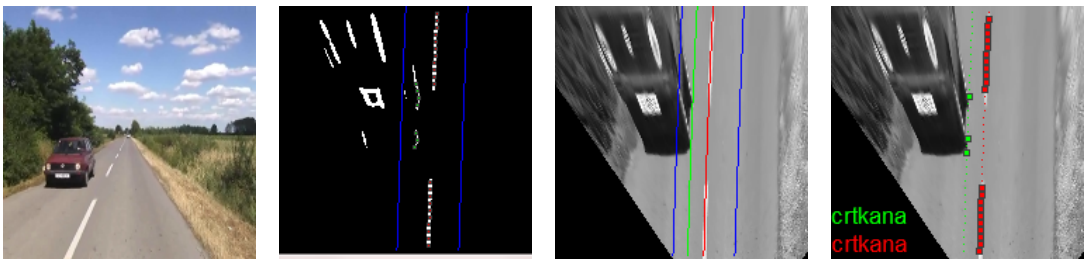
(a) (b) (c) (d)

Slika 5.3: Primjer uspješnog prepoznavanja dvostruke razdjelne linije uz sjene prisutne na cesti

Problemi kod pronalaženja razdjelne linije

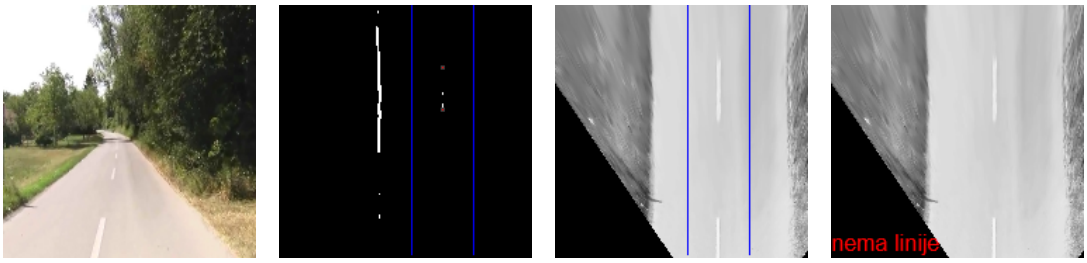
Slika 5.4 prikazuje primjer detektiranja linije koja ne postoji, tzv. *false positive* detekcija. Problem je u tome što je upravljivi filtar rub prolazećeg automobila prepoznao kao liniju na cesti, pa ispada da se radi o dvostrukoj razdjelnoj liniji.

Slika 5.5 prikazuje primjer na kojoj linija na cesti postoji, ali nije detektirana, tzv. *false negative* detekcija. Do problema dolazi zbog slabe vidljivosti linije na cesti i loše odabranog praga binarizacije. Rješenje ovog problema predloženo je u zaključku.



(a) (b) (c) (d)

Slika 5.4: Primjer detektiranja nepostojeće paralelne linije



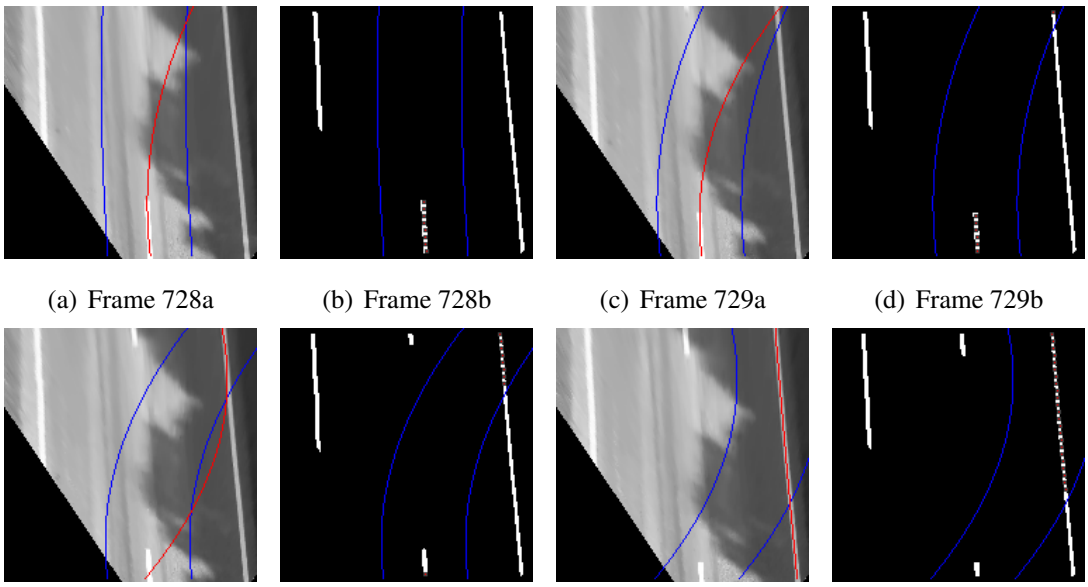
(a) (b) (c) (d)

Slika 5.5: Primjer neuspješnog detektiranja razdjelne linije

Problem skretanja područja interesa

Problem koji se javlja u nekim situacijama prilikom praćenja detektirane linije prikazan je na slici 5.6. Usred manjka točaka na liniji u području interesa, parabola aproksimirana kroz pronađene *inliere* naginje u stranu, što rezultira preseljenjem područja interesa sa središnje, tj. razdjelne linije, na lijevu ili desnu liniju koja označava rub ceste. Nakon što se područje interesa preselilo na rubnu liniju, ona će biti praćena do kraja videa.

Rješenje za ovaj problem također je predloženo u zaključku.



(a) Frame 728a (b) Frame 728b (c) Frame 729a (d) Frame 729b

(e) Frame 730a (f) Frame 730b (g) Frame 731a (h) Frame 731b

Slika 5.6: Skretanje područja interesa

6. Zaključak

Programska implementacija rješenja problema pronalaženja, prepoznavanja i praćenja razdjelne linije na cesti napisana u okviru ovog završnog rada daje dobre rezultate ukoliko su uvjeti na cesti dovoljno dobri, u smislu da je razdjelna linija jasno vidljiva, te ukoliko su parametri obrade dobro postavljeni.

Posebnu pažnju treba obratiti pri odabiru praga binarizacije, koji uvelike utječe na rezultate detekcije i prepoznavanja linije na cesti. Kako linije nisu jednake kvalitete na svim dijelovima ceste koje promatramo i kako cesta nije uvijek pod jednakim osvjetljenjem, zaključujemo da jedna vrijednost praga binarizacije ne može davati jednako dobre rezultate na svim područjima ceste. Rješenje koje bi taj problem trebalo riješiti jest uvođenje adaptivnog praga, koji bi ovisio o trenutnom okviru koji se obrađuje. Time bi se riješio i problem prikazan na slici 5.5, gdje linija nije detektirana zbog loše odabranog praga binarizacije. Jedna od ideja za ostvarenje adaptivnog praga je promatranje odnosa broja odabranih *inliera* i ukupnog broja točaka pronađenih u području interesa ovisno o odabranom pragu. Binarizacija bi se tada izvela s nizom različitih vrijednosti praga, a odabrao bi se onaj prag koji bi rezultirao najboljim odnosom broja *inliera* i ukupnog broja točaka.

Drugi problem koji se javio tijekom eksperimentiranja na stvarnim sljedovima slika jest skretanje područja interesa s razdjelne na rubnu liniju. Razlog zbog kojeg do toga dolazi je neprikladnost modela parabole kao modela za aproksimaciju linije u nekim situacijama, kao npr. u situaciji prikazanoj na slici 5.6. Kako bi se riješio ovaj problem, trebalo bi u svakom okviru, uz model parabole, kroz skup odabranih točaka aproksimirati i model pravca. Nakon usporedbe aproksimirajućeg pravca i parabole, za odabir granica područja interesa u sljedećem okviru koristio bi se onaj model koji je u trenutnom okviru bolje aproksimirao odabrani skup točaka. Kako bismo dobili još bolje rezultate, kod generiranja modela trebali bismo uzeti u obzir i smjer upravljivog filtra. Također bi se moglo razmisliti o korištenju isječka kružnice kao modela umjesto parabole.

Uvođenjem parametra za širinu linije mogao bi se eventualno izbjeći problem

prikazan na slici 5.4. Također, uvođenjem parametra za širinu ceste, postupak opisan u ovom završnom radu mogao bi se proširiti i na praćenje rubova ceste, te možda iskoristiti i za detekciju prometnog traka. Raspoznavanje detektirane linije moglo bi se poboljšati ocjenjivanjem vidljivosti i kontrastnosti linije.

LITERATURA

[1] Majić A. Pronalaženje prometnog traka korištenjem upravljivih filtara. Završni rad, Fakultet elektrotehnike i računarstva, Zagreb, 2009. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/majic09.pdf.

[2] Divjak B. Cardanova formula, veljača 2006. URL http://www.foi.hr/CMS_library/studiji/dodiplomski/IS/kolegiji/mmzi/zadaci/cardano.pdf. 26.05.2010.

[3] McCall J. C. i Triverdi M. M. Video based lane estimation and tracking for driver assistance: survey, system and evaluation. 2006. URL http://escholarship.org/uc/item/1bg5f8qd.

[4] Sunglok C., Taemin K., i Wonpil Y. Performance evaluation of RANSAC family. 2009. URL http://www.bmva.org/bmvc/2009/Papers/Paper355/Paper355.pdf.

[5] Jung C. R. i Kelber C. R. Lane following and lane departure using a linear-parabolic model. 2005. URL http://linkinghub.elsevier.com/retrieve/pii/S0262885605001265.

[6] Gusić I. Zašto su uvedeni kompleksni brojevi, veljača 2004. URL http://e.math.hr/old/povmat/pov1.html. 26.05.2010.

[7] Sikirić I., Brkić K., i Šegvić S. Recovering a comprehensive road appearance mosaic from video. MIPRO, Opatija, 2010. URL http://www.zemris.fer.hr/~ssegvic/pubs/sikiric10mipro.pdf.

[8] Pahikkala J. Cardano's formulae, veljača 2008. URL http://planetmath.org/encyclopedia/CardanosFormulae.html. 26.05.2010.

[9] ZuWhan K. Robust lane detection and tracking in challenging scenarios. 2008. URL http://escholarship.org/uc/item/50n0c8cg.

[10] Gulić M. Pronalaženje prometnog traka u odzivu upravljivog filtra. Završni rad, Fakultet elektrotehnike i računarstva, Zagreb, 2009. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/gulic09zavrad.pdf.

[11] Westwood M. Cardano's formulae, ožujak 2009. URL http://www.proofwiki.org/wiki/Cardano's_Formula. 27.05.2010.

[12] Bosilj P. Optimizacija implementacije projekcijskog ravninskog preslikavanja. Završni rad, Fakultet elektrotehnike i računarstva, Zagreb, 2010. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/bosilj10zavrad.pdf.

[13] Šegvić S., Brkić K., Kalafatić Z., Stanistavljević V., Budimir D., i Dadić I. Towards automatic assessment and mapping of traffic infrastructure by adding vision capabilities to a geoinformation inventory. Proceedings of MIPRO, Opatija, 2009. URL http://www.zemris.fer.hr/~ssegvic/pubs/mipro09.pdf.

Pronalaženje, prepoznavanje i praćenje razdjelne linije iz perspektive vozača

Sažetak

U ovom radu bavimo se pronalaženjem, prepoznavanjem i praćenjem razdjelne linije na cesti u slikama pribavljenih iz vozila u pokretu. Pretprocesiranje ulazne slike sastoji se od inverzne perspektivne transformacije i binarizacije odziva upravljivog filtra temeljenog na drugoj derivaciji Gaussove funkcije. Nakon toga, algoritmom RANSAC odabiremo skup točaka koji zatim aproksimiramo parabolom pomoću metode najmanjih kvadrata. Estimiranu parabolu koristimo za prepoznavanje vrste razdjelne linije, te za njezino praćenje. Prikazani su i komentirani rezultati testiranja ostvarenog programskog rješenja.

Ključne riječi: računalni vid, razdjelna linija, RANSAC, parabola, inverzna perspektivna transformacija, upravljivi filtar, Cardanova formula

Detecting, recognizing and tracking the central road surface line in images acquired from the driver's perspective

Abstract

In this work we consider detecting, recognizing and tracking of the central road surface line in images acquired from a moving vehicle. The preprocessing consists of inverse perspective transformation and binarization of the response of a filter based on the second derivative of a Gaussian. Using the RANSAC algorithm we choose a set of points which is then approximated using the least square method. The estimated parabola is later used for recognizing and tracking of the central road surface line. The results of implementation testing are shown and discussed.

Keywords: computer vision, central road surface line, RANSAC, parabola, inverse perspective transformation, steerable filter, Cardano's formula