

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Magistro tezės

Automobilio tipo atpažinimas vaizdų sekoje

Atliko: II mag. kurso, Kompiuterinio modeliavimo grupės studentas

Tomas Anbinderis (parašas)

Darbo vadovas:

doc. Algimantas Juozapavičius (parašas)

Vilnius
2005

Turinys

1	Įvadas.....	6
2	Sekimo bei atpažinimo metodų apžvalga	7
2.1	Sekimas.....	7
2.1.1	Judesio išskyrimas	7
2.1.2	Prognozavimas	8
2.2	Atpažinimas	10
2.2.1	Sekamo objekto kontūrai	10
2.2.1.1	Fiksuoti filtrai	10
2.2.1.2	Kraštų aptikimas	12
2.2.1.3	Tiesių aptikimas.....	13
2.2.1.4	Aktyvūs kontūrai (Snakes)	14
2.2.2	Vaizdų kombinavimas	16
3	Automobilio sekimas vaizdų sekoje	20
4	Automobilio tipo atpažinimas	22
4.1	Sekamo objekto aprašymas	24
4.2	3-D modelis	24
4.2.1	Vidiniai parametrai	24
4.2.2	Išoriniai parametrai.....	26
4.2.3	Pilnas 3-D modelis	27
4.3	Atitinkamų segmentų parinkimas (angl. segment matching)	28
4.3.1	Atstumas tarp vidurio taškų.....	28
4.3.2	Centruotas atstumas tarp vidurio taškų.....	29
4.4	Parametrų priderinimas.....	29
4.5	Vaizdų kombinacijos metodas.....	30
5	Testavimų rezultatai ir išvados	32
6	Literatūros sąrašas	46
7	Priedai	48
7.1	Priedas A	48

Naudojami sutrumpinimai

KF – Kalmano filtras

EKF – Išplėstas Kalmano filtras (angl. Extended Kalman Filter)

HT – Hofo transformacija (angl. Hough Transform)

SHT – Standartinė Hofo transformacija

PPHT – Progresyvioji tikimybinė Hofo transformacija

BMA – Blokų Tapatinimo Algoritmas (angl. Block Matching Algorithm)

pav. – paveiksliukas

Paveiksliukų sąrašas

pav. 2.1	11
pav. 2.2	11
pav. 2.3	12
pav. 2.4	13
pav. 2.5	18
pav. 2.6	19
pav. 3.1	20
pav. 3.2	21
pav. 4.1	23
pav. 4.2	24
pav. 4.3	25
pav. 4.4	26
pav. 4.5	27
pav. 4.6	28
pav. 4.7	30
pav. 5.1	32
pav. 5.2	33
pav. 5.3	33
pav. 5.4	34
pav. 5.5	34
pav. 5.6	35
pav. 5.7	35
pav. 5.8	36
pav. 5.9	36

pav. 5.10	37
pav. 5.11	37
pav. 5.12	38
pav. 5.13	38
pav. 5.14	39
pav. 5.15	39
pav. 5.16	40
pav. 5.17	40
pav. 5.18	41
pav. 5.19	41
pav. 5.20	42
pav. 5.21	42
pav. 5.22	43
pav. 5.23	43
pav. 5.24	44
pav. 5.25	45
pav. 5.26	45

Lentelių sąrašas

lentelė 5.1	32
lentelė 5.2	36
lentelė 5.3	38
lentelė 5.4	44

Santrauka

Šiame darbe yra analizuojami įvairūs automobilio sekimo bei automobilio tipo atpažinimo metodai. Yra tiriama ryšiai tarp šių metodų bei skirtingos metodų modifikacijos. Automobilio sekimas, kuris yra pagrįstas judesio išskyrimo metodais, naudoja prognozavimo mechanizmus, leidžiančius prognozuoti judančio objekto koordinatas sekančiame vaizdų sekos kadre. Prognozė remiasi judančio objekto padėties bei dydžio pasikeitimais kadre. Automobilio tipo (klasės) atpažinimas vykdomas lyginant sekamo automobilio kontūrus ir karkasinį modelį [16]. Sekamo automobilio kontūrai yra gaunami aptinkant kraštus Canny metodo pagalba bei naudojant metodą Hofo transformacijos pagrindu. Karkasinis modelis yra nusakomas 12 formos bei 5 padėties parametrais. Šiame darbe yra analizuojami metodai, leidžiantys įvykdyti šį palyginimą. Be to, automobilio poklasė yra atpažįstama naudojant S.Ullmano [17] pasiūlytą vaizdų kombinavimo metodą. Testavimo rezultatai bei išvados yra pateikiamos darbo pabaigoje.

Summary

This paper describes different vehicle tracking and classification methods. Different links (relations) between these methods and their modifications are studied. In vehicle tracking process, which is based on motion segmentation algorithms, estimators are used. Estimators can predict moving objects coordinates in the next frame. Estimators predictions are based on information about moving objects position and size change in current frame. Vehicle class recognition is performed by comparing tracked vehicle contours and flexible wireframe model [16]. Tracking vehicle contours are extracted using method based on Hough transform and Canny edge detector. Wireframe model is defined by 12 shape and 5 pose parameters. This paper describes different methods, which can perform this comparison. Vehicle subclass is recognized using methods which are based on view combination method [17]. Testing results are given at the end of this paper.

1 Įvadas

Šio darbo tikslas yra sukurti metodus, kurie galėtų automatiškai sekti judančius objektus (automobilius) vaizdų sekoje ir atpažinti jų tipą. Objektų sekimas bei klasifikavimas yra plačiai tiriamos problemos. Tai tema, kuri turi didelę panaudojimo sritį, nuo eismo stebėjimo iki medicininio vaizdavimo. Šiame darbe sprendžiamos eismo stebėjimo iš statinės kameros problemos.

Darbo eigoje buvo analizuojami pagrindiniai kompiuterinės regos bei kompiuterinės geometrijos algoritmai. Didžiausias dėmesys yra sukoncentruotas ties metodais, kurie sprendžia automobilio *sekimo* bei jo tipo *atpažinimo* (klasifikavimo) problemas. Sekimo metodų tikslas yra surasti bei sekti mus dominančias vaizdo sritis, kurias reikės atpažinti. Teoriškai galima būtų atpažinti (klasifikuoti) kiekvieną galimą vaizdo regioną, tačiau atpažinimo (klasifikavimo) metodai bendrai yra gana brangūs, skaičiavimo prasme, todėl ši užduotis būtų neįvykdoma (vėl gi skaičiavimo prasme). Klasifikavimo metodų tikslas yra nuspręsti, kokiam tipui priklauso automobilis (pvz.: pikapas, sedanas ar sunkvežimis). Šiame darbe daugiau yra sprendžiama objekto atpažinimo (identifikacijos), negu klasifikacijos problema.

Objektų sekimas yra užduotis, kurios sprendimas yra gana aiškus, todėl darbo dėmesys sukoncentruotas ties objektų tipo atpažinimo problema. Biologinėms sistemoms objektų atpažinimas yra natūralus procesas, tačiau dirbtinėse skaičiavimo sistemose, objektų atpažinimas gali būti labai sunkus uždavinys dėl vienai klasei priklausančių objektų vaizdinių skirtumų. Be to, tas pats objektas, pateiktas atpažinimui skirtingomis vaizdavimo sąlygomis, duoda didesnius skirtumus, negu du vienai klasei priklausančios objektai, kurie yra pateikti vienodomis vaizdavimo sąlygomis [17], todėl formuojant šabloninį modelį atpažinimui, reikia didelį dėmesį kreipti į nagrinėjamo objekto vaizdavimo sąlygų pakeitimą. Dauguma šiame darbe apžvelgtų objektų atpažinimo metodų vykdo objekto atpažinimą pagal jo kontūrus. Metodai, kurie nebuvo analizuojami, vykdo atpažinimą, pvz., pagal tekstūras.

Darbe yra aprašomas automobilio tipo atpažinimo metodas, kuris remiasi D.Koller [16] pasiūlyto karkasinio automobilio modelio bei sekamo automobilio kontūrų palyginimu. Karkasinis modelis yra apibrėžiamas automobilio vidiniais bei išoriniais parametrais.

Be to, darbe yra siūlomas metodas, kuris leidžia gauti automobilio, naujomis vaizdavimo sąlygomis, vaizdą, turint tik šio automobilio 2-D kontūrus. Tai įvykdoma surandant optimalų 3-D karkasinį modelį (4.2, 4.3, 4.4 skyriai), bei taikant S.Ullmano [17] pasiūlytą vaizdų kombinavimo metodą (2.2.2, 4.5 skyriai) šiam sudarytam optimaliam 3-D modeliui ir turimam 2-D kontūrai.

2 Sekimo bei atpažinimo metodų apžvalga

2.1 Sekimas

Judančio objekto *sekimo* procesas gali būti paskirstytas į du atskirus procesus. Tai *judesio išskyrimas* (*segmentavimas*) bei objekto padėties sekančiame kadre *prognozavimas*.

2.1.1 Judesio išskyrimas

Turbūt pats charakteringiausias automobilio parametras vaizde yra tai, kad automobiliai dažniausiai juda. Be to, jų judesys yra gana nusakomas. Automobiliai dažniausiai juda lygiagrečiai kažkokios jų pagrindinės ašies, kuri gali kisti, jei automobilis vykdo posūkio manevrą. Aišku, ši prielaida gali būti padaryta tik, jei automobilis juda, tačiau dažnai vaizduose automobilis būna vienintelis judantis objektas vaizde. Taip pat dažnai mus domina tik judantys automobiliai.

Yra keletas sunkumų įvertinant judesį. Yra operuojama tik vaizdais iš vienos kameros, todėl vienintelė turima informacija yra pasaulio projekcija. Perspektyva, pateikiama šios projekcijos, duoda skirtingą judesį skirtinguose vaizdo dalyse. Tai vadinamasis *paralakso* (angl. *parallax*) efektas: objektai, kurie yra arčiau kameros, duoda didesnę transliaciją (postūmį) vaizde, negu objektai, kurie yra toliau nuo tos pačios kameros, net jei realiame pasaulyje šių objektų transliacijos parametrai buvo vienodi.

Kai vaizde nėra kitų judančių objektų, judesio duodamos informacijos pagalba galima aptikti vaizde mus dominantį objektą – automobilį. Vienas pagrindinių judesio išskyrimo algoritmų yra *fono atimtis* (angl. *background subtraction*). Šis metodas yra gana efektyvus judančių objektų vaizde išskyrimui. Fono atimties metodas susiveda į turimo vaizdo atimtį iš fono vaizdo, kuris turi būti gautas iš anksto. Po atimties rezultato vaizde liks tik judantys objektai:

$$| frame_i - background_i | > Th ,$$

kur $frame_i$ yra nagrinėjamas naujas vaizdas, $background_i$ – fono vaizdas, Th – iš anksto pasirinktas slenkstis. Šis metodas ypač gerai tinka aplikacijų, kuriose fonas nesikeičia, realizavimui. Tai galėtų būti video-konferencijos arba video stebėjimas (pvz., eismo sekimo aplikacijose). Šio metodo pagrindinis privalumas yra tai, kad jis yra pigus skaičiavimo prasme. Tačiau fono atimties metodas turi kelis akivaizdžius trūkumus. Metodas gražina klaidingus rezultatus, jei, pvz.: judančio objekto taškų reikšmės sutaps su fono taškų reikšmėmis kažkokoje vaizdo srityje arba įvyks apšvietimo pasikeitimai (dėl paros laiko pasikeitimo arba dėl

nenumatomų priežasčių, kaip, pvz., debesys). Akivaizdus žingsnis, kurį reikia įvykdyti, norint pagerinti fono atimties metodą, yra *fono vaizdo atnaujinimo (prisitaikymo)* mechanizmo (angl. adaptive background) realizavime. Tikslas yra gauti kuo geresnį fono vaizdą. Daugelis metodų yra pasiūlyta tam, kad atnaujinti fono vaizdą. Darbuose [1][2] yra siūlomi efektyvūs metodai, sprendžiantys šias problemas. Viena iš paprasčiausių idėjų būtų prilyginti fono vaizdą prieš taiėjusiam kadrai:

$$|frame_i - frame_{i-1}| > Th,$$

tačiau šis metodas veikia tik tam tikromis sąlygomis, kurios priklauso nuo objekto judėjimo greičio, bei kadrų per sekundę skaičiaus (kadrų dažnio). Be to, šis metodas yra labai jautrus slenksčio Th reikšmės pakeitimui. Fonui gauti galima naudoti kelių praeitų kadrų vidurkį arba „bėgantį“ (*pasvertą*) vidurkį:

$$background_i = \alpha * frame_i + (1 - \alpha) * background_i,$$

kur α yra mokymosi tempas (greitis) (pvz., $\alpha = 0.05$).

Kitas būdas sekti judantį objektą yra apskaičiuoti *optinį srautą* vaizde. Nors metodai, apskaičiuojantys optinį srautą, yra gana brangūs skaičiavimo prasme (lyginant su, pvz., fono atimties metodu), tačiau šie metodai grąžina ne tik judesio taškus (regionus), bet ir judesio vektorius. Tai padidina sugebėjimą aptikti judantį automobilį, net jei jis yra dalinai uždengtas ar jei stebimas eismas yra perpildytas.

Darbe [10] yra aprašomas plačiai naudojamas blokų tapatinimo algoritmas (angl. Block Matching Algorithm) (BMA). BMA padalina nagrinėjamą vaizdą į nedidelius, fiksuoto dydžio blokus ir lygina juos su blokais ankstesniajame kadre. Suformuojamas optinis srautas. Automobiliai yra aptinkami grupuojant blokus, turinčius panašius judesio vektorius. Beje, šis metodas yra naudojamas judesio įvertinimui MPEG vaizdų sekos suspaudimo standarte.

Plačiai žinomas *Horn-Schnuck* [18] algoritmas irgi naudojamas apskaičiuoti optiniam srautui. Šis iteracinis metodas priklauso algoritmų klasei, kuri yra žinoma kaip relaksacijos algoritmų klasė (angl. relaxation algorithm class). Po pirmos iteracijos srauto vektoriai rodo tik kraštų judesį. Kai iteracija tęsiasi, vyksta vektorių išlyginimas. Rezultate yra gaunama kažkas panašaus į vektorių lauką. Aišku, laukas turi kur kas daugiau triukšmo ir keli vektoriai būna visiškai neteisingi, bet idėja tikrai veikia, nors ir tam tikru lygiu.

2.1.2 Prognozavimas

Aišku, optinio srauto apskaičiavimo metodai padidina sugebėjimą vykdyti sėkmingą objekto sekimą, tačiau reikalingas specialus įrankis, įgalinantis vykdyti objekto sekimą. Jei objektas, pvz., ilgesniam laikui yra uždengiamas kažkoku kitu objektu, judesio išskyrimo algoritmai negalės aptikti šio objekto. Šiai problemai spręsti yra naudojami „įvertintojai“. Po to,

kai judesys vaizde buvo atpažintas, galima efektyviai sekti judantį objektą, prognozuojant judančio objekto koordinatas sekančiame kadre, remiantis duomenimis iš praeitų kadrų. Šiai prognozei (angl. estimation) įvykdyti dažnai naudojamas *Kalmano filtras*. Šio filtro bei jo praplėtimo detalus aprašymas yra darbe [4]. Šio filtro algoritmas prognozuoja sekančią sistemos būseną. Kalmano filtras sprendžia bendrą problemą prognozuojant sistemos būseną x diskrečiajame laiko procese, kuris yra apibrėžiamas tiesine stochastine (atsitiktine, tikimybine) lygtimi:

$$x_{k+1} = Ax_k + w_k,$$

matavimas z apibrėžiamas:

$$z_k = Hx_k + v_k,$$

kur atsitiktiniai dydžiai w_k ir v_k yra atitinkamai proceso ir matavimo triukšmas. Jie yra nepriklausomi vienas nuo kito (nekoreliuoti), be to, triukšmas yra baltas su normaliu tikimybinio pasiskirstymu (angl. normal probability distribution). Po to, kai filtras prognozuoja sekančią sistemos būseną, filtras yra koreguojamas matavimais z . Paprasčiausiu atveju vektorius, nusakantis matavimą z , atitinka sistemos būsenos vektorių. Pradinė sistemos būsena gali būti parinkta atsitiktinai.

Jei lygtis, kuri apibrėžia sistemos būsenos pakeitimą yra netiesinė, naudojamas *Kalmano filtro praplėtimas* (angl. extended Kalman filter) (EKF) [3][4].

Daugelis darbų aprašo įvairius Kalmano filtro arba Kalmano filtro išplėtimo panaudojimus, prognozuojant judančio objekto koordinatas sekančiame kadre. Darbuose [11], [2] judančio objekto koordinatės sekančiame kadre apskaičiuojamos remiantis judančio objekto greičiu, objekto koordinatėmis vaizde, bei kadrų skaičiumi per sekundę. Darbe [2] yra aprašomas Kalmano filtro panaudojimas, kur siūlomas uždengimo (angl. occlusion) problemos sprendimas. Vektorius nusakantis sistemos būseną yra apibrėžiamas taip:

$$x(k) = (\Delta x_center(k), \Delta y_center(k), \Delta xsize(k), \Delta ysize(k))^T,$$

kur $\Delta x_center(k)$, $\Delta y_center(k)$ yra sekamo objekto centro x bei y koordinatinių pasikeitimai, o $\Delta xsize(k)$, $\Delta ysize(k)$ yra objekto dydžio vaizde pasikeitimai.

Kalmano filtro išplėtimo papildymas yra siūlomas [5]. Darbe yra pristatomas dinaminis automobilio modelis, kur yra įvertinami tokie parametrai, kaip, pvz., atstumas tarp priekinių ir galinių ratų. Kalmano filtro išplėtimas yra daug kur naudojamas sekant objektus. Tačiau todėl, kad nėra tiesioginio ryšio tarp naudojamo dinaminio modelio ir vairuotojo, kuris kontroliuoja automobilio judesį, veiksmų, įprastas EKF neduoda gerų rezultatų, jei atliekamas sudėtingas manevras. Siūlomas EKF papildymas sprendžia šias problemas.

Kalmano filtrą galima naudoti ne tik prognozuojant objekto koordinatas sekančiame kadre, bet ir jo kontūrus. Darbe [6] yra siūloma išskaidyti prognozavimo procesą į du filtras:

vieną judesio prognozei, o kitą automobilio kontūrų formos prognozei. Kontūrų aproksimacijai yra naudojami kubiniai splainai (vadinami gyvatėmis) (angl. *snakes*). Siūloma šiam splainui naudoti 12 kontrolinių taškų. Būtent iš šitų 12 x ir y koordinatinių porų ir sudaromas būsenos vektorius. Tai paprasčiausias Kalmano filtro panaudojimo atvejis, kai matavimo vektorius atitinka būsenos vektorius. Šis galingas Kalmano filtro realizavimas leidžia sėkmingai spręsti uždengimo problemą.

Tarp kitų plačiai naudojamų „įvertintojų“ verta paminėti *ConDensation* algoritimą[7][8].

Darbe [9] objektų sekimui yra naudojamas *Kanade-Lucas-Tomasi* metodas. Darbe yra aprašomi vaizdų registravimo mechanizmai, leidžiantys spręsti judančios kameros problemą.

2.2 Atpažinimas

Po to, kai judantis objektas buvo susektas, reikia jį *atpažinti*. Po sekimo žingsnio vienintelė turima informacija yra judančio objekto koordinatės vaizde bei judėjimo kryptis. Objekto atpažinimui reikalingi šio objekto nusakantys parametrai. Paprasčiausias būdas klasifikuoti objektus (automobilius) – pagal objekto dydį. Darbe [11] siūlomas metodas paskirstantis judančius automobilius į klases, remiantis *objekto dydžio vaizde* matavimais. Kiti paprasčiausi parametrai, kurie aprašo judantį automobilį, pvz., *greitis* [2], irgi galėtų būti naudojami automobilių klasifikavimui (pvz., sunkvežimiai turėtų judėti lėčiau, negu lengvasis automobilis), tačiau šių parametrų nepakaks sprendžiant atpažinimo problemą. Be abejo, reikia parametrų, kurie galėtų tiksliau (sudėtingiau) aprašyti objektus, kuriuos reikia atpažinti.

Darbe [12] yra aprašomas klasifikacijos metodas, kuris remiasi *objekto fragmentine reprezentacija* (angl. *fragment-based*). Čia vienai klasei priklausančios objektai yra formuojami iš vaizdo fragmentų, kurie yra naudojami kaip objekto sudedamosios dalys. Tokiu būdu gali būti suformuota didelė aibė klasės objektų. Klasifikacijos tikslui, fragmentai irgi yra suskirstyti į klases, kur kiekviena klasė yra alternatyvių fragmentų rinkinys, tokių, kaip, pavyzdžiui, skirtinga šukuosena arba akys. Darbe [13] irgi yra naudojamas fragmentinės vaizdų reprezentacijos metodas. Čia yra naudojama vaizdinė struktūra (angl. *pictorial structure*). Ši struktūra susideda iš fragmentų aibės bei ryšių tarp šių fragmentų porų. Darbe [14] siūlomas šios vaizdinės struktūros išplėtimas. Tačiau automobilio tipo atpažinimo užduočiai spęsti šis metodas netinka.

2.2.1 Sekamo objekto kontūrai

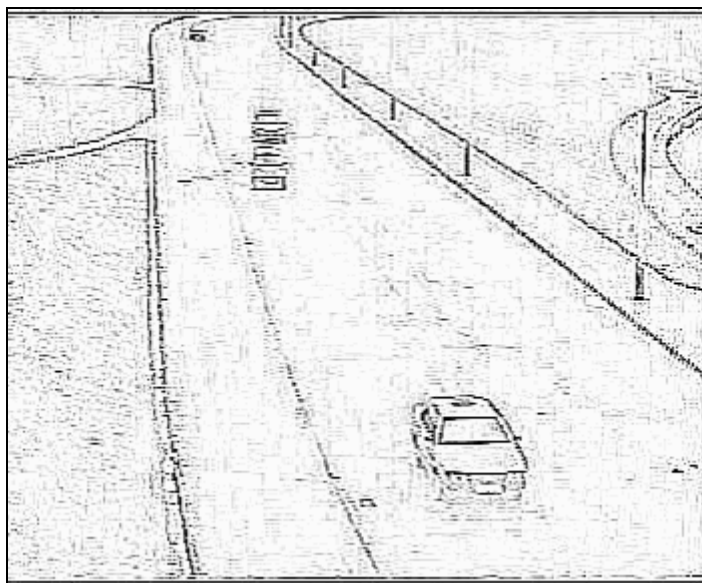
2.2.1.1 Fiksuoti filtrai

Objektų kontūrams aprašyti galima naudoti fiksuotus filtrus (angl. *fixed filters*): *Sobel* bei *Laplace*. Rezultatai gauti po filtrų panaudojimo turi daugiau triukšmo, negu, pvz.: po žemiau aprašyto *Canny* metodo kraštams aptikti panaudojimo. 2.1 pav., 2.2 pav. bei 2.3 pav. yra

pavaizduoti pradinis vaizdas, gautas po *Laplace* filtro panaudojimo, bei gautas po *Sobel* filtro panaudojimo atitinkamai.



pav. 2.1



pav. 2.2



pav. 2.3

2.2.1.2 Kraštų aptikimas

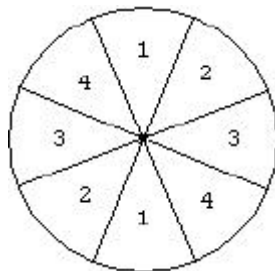
Efektyvus, tačiau nesprendžiantis atpažinimo skirtingomis vaizdavimo sąlygomis problemos metodas yra siūlomas [15]. Siūlomas automobilio tipo atpažinimo metodas remiasi objekto aptiktais kraštais. Atpažinimas vykdomas pagal iš anksto paruoštus vaizdus. Siūlomas metodas yra gana efektyvus, tačiau, pakitus kameros parametrams, reikia keisti ir visą turimų šabloninių modelių duomenų bazę.

Tam, kad efektyviai aprašyti objektą, galima naudoti metodą, siūlomą [3]. Čia yra naudojamas *Canny* metodas kraštų aptikimui, ir po to aptiktiems kraštams yra pritaikomas metodas *Hof* transformacijos pagrindu tam, kad aptikti linijų segmentus.

Kraštai – tai ribos, atskiriančios sritis (regionus), turinčias skirtingas ryškumo arba spalvos reikšmes. J.Canny [19] yra pasiūlęs efektyvų metodą kraštams aptikti. Šis metodas operuoja su pilko atspalvio (gradientiniais) vaizdais ir rezultate grąžina vaizdą, kur visi nelygūs nuliui taškai žymi aptiktus kraštus. Toliau yra aprašomi 4 metodo žingsniai:

1. *Vaizdo išlyginimas*. Vaizdas yra išlyginamas Gaussian funkcija su iš anksto pasirinktu pločio parametru.
2. *Diferenciacija*. Išlygintas vaizdas, gautas po pirmo žingsnio, yra diferencijuojamas atsižvelgiant į kryptis x ir y . Iš apskaičiuotų gradientų reikšmių x ir y gali būti apskaičiuoti gradiento dydis ir kampas, naudojant įžambinės ir arktangento funkcijas.
3. *Nemaksimumo nuslopinimas*. Po to, kai gradiento reikšmė buvo apskaičiuota kiekviename vaizdo taške, kraštai gali būti taškuose, kurių gradiento reikšmė yra lokalus maksimumas. Tai yra įvykdoma slopinant nemaksimumus, t.y. taškus,

kurių gradiento reikšmės nėra lokalūs maksimumai. Tačiau šiuo atveju nemaksimumai, kurie yra statmeni krašto kryptims, skirtingai nuo triukšmo kraštų kryptimi, turi būti nuslopinti, nes yra laukiama, kad kraštas tęsis palei pratęstą kontūrą. Algoritmas pradeda mažindamas gradiento kampą iki vieno iš keturių sektorių atvaizduotų 2.4 pav. Operuojant su 3x3 taško kaimynais, kiekviename taške centrinis kaimynystės elementas yra lyginamas su jo dviem kaimynais palei gradiento kryptį, atitinkančią sektoriaus reikšmę. Jei centrinė reikšmė yra nemaksimumas, t.y. nedidesnė, negu kaimynų reikšmės, ji yra slopinama.



pav. 2.4

4. Canny metodas naudoja taip vadinamą „histerezinį“ (angl. hysteresis) slenkstį. Tai reiškia, jog naudojamas slenkstis turi dvi reikšmes, kurios atitinka viršutinę bei apatinę ribas. Naudojant šį slenkstį, kai yra analizuojamas linijos segmentas, yra išvengiama taip vadinamo blykštelėjimo (angl. streaking) efekto. Analizuojant linijos segmentą, jei reikšmė yra didesnė, negu viršutinė riba, ji yra tuoj pat priimama, jei mažesnė, negu apatinė riba - tuoj pat atmetama. Taškai, kurie yra tarp dviejų ribų, yra priimami tik, jei jie sujungti su taškais, kurie turi didelį atsaką (angl. strong response).

Detalesnis algoritmo aprašymas yra siūlomas [19].

2.2.1.3 Tiesių aptikimas

Hofo transformacija (angl. Hough transform) (HT) – tai labai populiarus metodas, kurio pagalba galima išgauti geometrinius primityvus iš rastrinių vaizdų. Paprasčiausios algoritmo realizacijos aptinka tiesias linijas, tačiau algoritmas nesunkiai modifikuojamas sunkesnių detalių radimui. Yra kelios pagrindinės HT algoritmų klasės, kurių panaudojimas priklauso nuo turimo vaizdo.

Standartinė Hofo transformacija (angl. Standart Hough Transform) (SHT) algoritmas, kaip ir visi HT algoritmai operuoja su diskrečiąja aibe (pavienių) primityvių parametrų. Jei norima aptikti tieses, tai parametrai yra ρ ir θ tokie, kad tiesės lygtis yra:

$$\rho = x \cos(\theta) + y \sin(\theta)$$

Čia ρ yra atstumas iki tiesės nuo nulinio taško, o θ yra kampas tarp x ašies bei vektoriaus, kuris yra statmenas tiesei ir prasideda nuliniame taške. Kiekvienas vaizdo taškas gali priklausyti kelioms tiesėms, kurios yra aprašytos minėtais parametrais. Kitaip tariant, tam, kad atskirti triukšmą nuo „gerų“ duomenų, yra apibrėžiamas kaupiklis (angl. „*accumulator*“) masyvas $A(\rho, \theta)$, kuriame visų elementų pradinės reikšmės yra lygios nuliui. Kiekvienam taškui, kurio reikšmė pradiname vaizde yra nelygi nuliui, visų *accumulator* masyvo elementų, atitinkančių tieses, kurios kerta nagrinėjamą tašką, reikšmės yra padidinamos vienetu. Toliau pasirenkama kažkokia slenksčio reikšmė Th tam, kad panaikinti triukšmą (t.y. paliekamos tik tos parametru poros (ρ, θ) , kurioms $A(\rho, \theta)$ yra didesnis negu pasirinktas slenkstis Th). Visos likusios poros charakterizuoja aptiktas linijas.

Daugiamatė Hofo transformacija (angl. Multidimensional Hough Transform) (MHT) yra SHT algoritmo modifikacija. Šis algoritmas, panaudodamas vaizdo žemesnę rezoliuciją, apskaičiuoja parametru poras, taikydamas SHT algoritmą. Po to algoritmas parametru erdvėje aptinka parametru sritis, kurios turėtų atitikti tieses pradiname vaizde. MHT algoritmas turėtų būti naudojamas, kai pradiniai vaizdai turi kelias tieses ir mažai triukšmo.

Progresyvioji tikimybinė Hofo transformacija (angl. Progressive Probabilistic Hough Transform) (PPHT). Šio metodo idėja yra tame, kad yra operuojama su atsitiktinai parinktais taškais. Kiekvieną kartą, kai pasikeičia bet kurio *accumulator* masyvo elemento reikšmė, didžiausia reikšmė yra lyginama su pasirinktu slenksčiu. Jei reikšmė yra didesnė, negu pasirinktas slenkstis, tai visi taškai, kurie priklauso „koridoriui“, apibrėžtam šia didžiausia reikšme, yra šalinami. Jei taškų skaičius viršija iš anksto pasirinktą reikšmę, t.y. tiesios linijos ilgį, tai aptikta detalė laikoma tiesiąja, kitaip skaitoma, kad tai triukšmas. Po to procesas pasikartoja iš pradžių, kol vaizde nebelieka taškų. Algoritmas pagerina rezultatą po kiekvieno žingsnio, todėl gali būti sustabdytas bet kuriuo metu. Algoritmo detalesnis aprašymas yra siūlomas [20], kur teigiama, jog PPHT algoritmas nesunkiai modifikuojamas visais atvejais, kai įmanoma modifikuoti SHT algoritmą. PPHT algoritmo trūkumas yra tas, kad, skirtingai nuo SHT, šis algoritmas klaidingai aptinka kai kurias vaizdo detales, pvz., tiesių susikirtimus. Daugiau informacijos galima rasti [21].

2.2.1.4 Aktyvūs kontūrai (Snakes)

Kaip jau minėta, kontūrų aproksimacijai galima naudoti taip vadinamus aktyvius kontūrus arba gyvates (angl. *snakes*). „*Gyvatė*“ [22] yra apibrėžiama kaip parametrinė energiją-minimizuojanti uždaroji kreivė, kurią veikia išorinės jėgos. Gyvatės kreivė atitinka energijos funkciją:

$$E = E_{int} + E_{ext},$$

kur E_{int} yra vidinė gyvatės energija, kurią formuoja gyvatės konfigūracija, E_{ext} yra išorinė energija, kurią valdo išorinės jėgos. Gyvatės tikslas yra surasti tokią padėtį, kuri galėtų minimizuoti energiją.

Tegul $p_1, \dots, p_n$ yra diskretus gyvatės atvaizdavimas, t.y. taškų seka vaizdo plokštumoje.

Kaip *vidinės energijos* funkcija dažnai yra naudojama kontūrų tolydumo energija (angl. continuity energy) bei kontūrų kreivumo energija (angl. curvature energy), t.y.:

$$E = E_{cont} + E_{curv},$$

kur:

- E_{cont} yra *tolydumo energija*. Energija yra $E_{cont} = |d - \|p_i - p_{i-1}\||$, kur d yra vidutinis atstumas tarp visų porų $(p_i - p_{i-1})$. Minimizuojant E_{cont} visuose gyvatės taškuose $p_1, \dots, p_n$, gyvatės taškai yra veikiami taip, kad sumažinti skirtumą tarp šių taškų porų atstumų.
- E_{curv} yra *kontūrų kreivumo energija*. Kuo kontūras yra lygesnis, tuo mažesnė yra jo kreivumo energija. $E_{curv} = \|p_{i-1} - 2p_i + p_{i+1}\|^2$.

Išorinė energija [22] yra apibrėžiama, kaip:

$$E = E_{img} + E_{con},$$

kur E_{img} – vaizdo energija ir E_{con} – papildomų apribojimų (ryšių) energija.

Yra siūlomi du vaizdo energijos apibrėžimo variantai:

1. $E_{img} = -I$, kur I yra vaizdo intensyvumas. Šiuo atveju gyvatė yra pritraukiama prie šviesių vaizdo linijų.
2. $E_{con} = -\|grad(I)\|$. Šiuo atveju gyvatė yra pritraukiama prie vaizdo kraštų.

Papildomų apribojimų (ryšių) energijos funkcijos E_{con} variantas yra siūlomas [22].

Apibendrinant aukščiau išdėstytą informaciją, bendra energija kiekviename gyvatės taške gali būti užrašyta kaip:

$$E_i = \alpha_i E_{cont,i} + \beta_i E_{curv,i} + \gamma_i E_{img,i},$$

kur α, β, γ yra svoriai kiekvienos rūšies energijai. Tada pilna gyvatės energija yra visų taškų energijų E_i suma.

α, β, γ prasmės yra tokios:

- α – veikia kontūro tolydumą, t.y. didelis α padaro gyvatės taškus lygiau išdėstytus.
- β – veikia gyvatės kampus, t.y. didelis β padaro kampus tarp gyvatės kraštų bukesniais.
- γ – nusako, kiek gyvatė bus jautresnė vaizdo energijai, negu tolydumo ar kreivumo energijoms.

Siūlomas toks darbo su gyvate algoritmas:

1. kuriama gyvatė su pradine konfigūracija

2. kiekviename taške yra apibrėžiamos α, β, γ reikšmės.
3. toliau vykdomas gyvatės energijos minimizavimas
4. įvertinama gyvatės padėtis. Jei yra būtina, koreguojamos α, β, γ reikšmės, bei, galbūt vaizdo duomenys, ir pakartojamas 3-čias žingsnis.

Yra trys gerai žinomi algoritmai gyvatės energijai minimizuoti. Darbe [22] minimizacijos procesas remiasi pakeitimų skaičiavimu. Darbe [23] yra naudojamas dinaminis programavimas. Brangus skaičiavimo prasme algoritmas yra siūlomas [24]. Pastarasis algoritmas duoda gana gerus rezultatus. Šio algoritmo schema kiekvienam taškui pateikiama žemiau:

1. Naudojant lygtį $E_i = \alpha_i E_{cont,i} + \beta_i E_{curv,i} + \gamma_i E_{img,i}$ yra apskaičiuojamos energijoms visiems taškams taško „kaimynystėje“. Prieš apskaičiuojant E_i , kiekviena energijos rūšis $E_{cont,i}$, E_{curv} , E_{img} turi būti normalizuota naudojant formulę: $E_{normalizuota} = (E_{img} - min) / (max - min)$, kur max ir min yra maksimali bei minimali reikšmė taško kaimynystėje.
2. Pasirinkti padėtį su mažiausia energija ir į šią padėtį gyvatę.
3. Kartoti visus žingsnius, kol nebus pasiektas konvergavimas.

Konvergavimo kriterijai:

- Pasiektas maksimalus iteracijų skaičius
- Taškų, kurių padėtis praeitojoje iteracijoje buvo pakeista, skaičius yra mažesnis, negu iš anksto pasirinktas slenkstis.

2.2.2 Vaizdų kombinavimas

Svarbi atpažinimo problema yra objektų atpažinimas skirtingomis vaizdavimo sąlygomis. Darbe [17] yra siūlomas metodas, leidžiantis gauti modelio vaizdą naujomis vaizdavimo sąlygomis tiesiogiai iš 2-D vaizdų. Siūlomi vaizdų kombinavimo (angl. view morphing) metodai leidžia gauti vaizdus iš 2-jų ar 3-jų objektų skirtingomis vaizdavimo sąlygomis. Dažnai objektų atpažinimo teorijose yra naudojamas objektų 3-D struktūrinis aprašymas, t.y. sistema kažkokiu būdu saugo ir operuoja objektų 3-D modeliais. Kai ši sistema susiduria su nauju objekto 2-D vaizdu sistema, sprendžia, ar tai yra vaizdas vieno iš saugomų 3-D modelių. Tačiau įmanoma atpažinti objektą nenaudojant 3-D struktūrinės informacijos apie objektą. Vietoj to yra naudojami 2-D objekto vaizdai tiesiogiai, tuo pačiu yra išvengiama būtinybės atstatyti objekto 3-D struktūrą iš objekto 2-D vaizdo.

S.Ullmano darbe [17] 3-D objektas yra išreiškiamas per 2-D objekto vaizdų tiesinę kombinaciją. Jei $M = M_1, \dots, M_k$ yra aibė vaizdų, išreiškiančių duotą objektą, o P yra 2-D vaizdas to objekto, kurio tipą reikia atpažinti, tada P laikomas M aibės egzemplioriumi, jei:

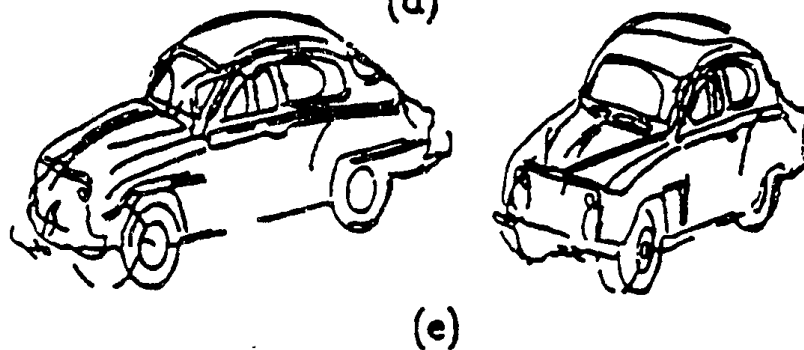
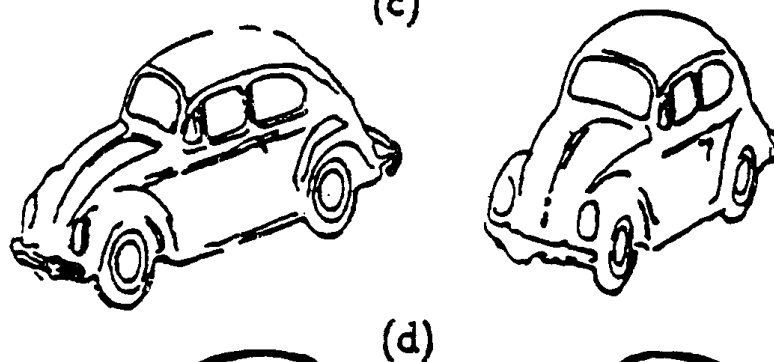
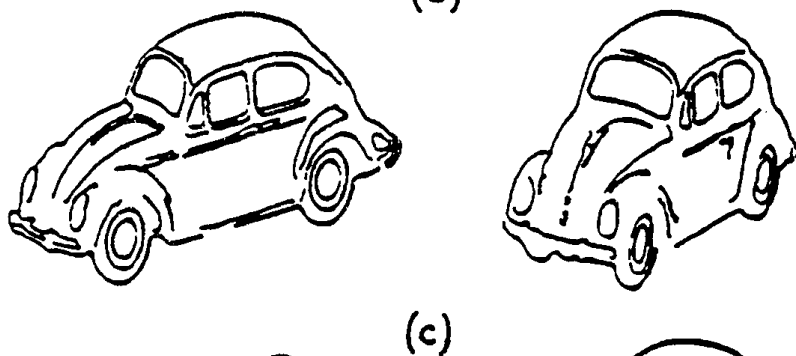
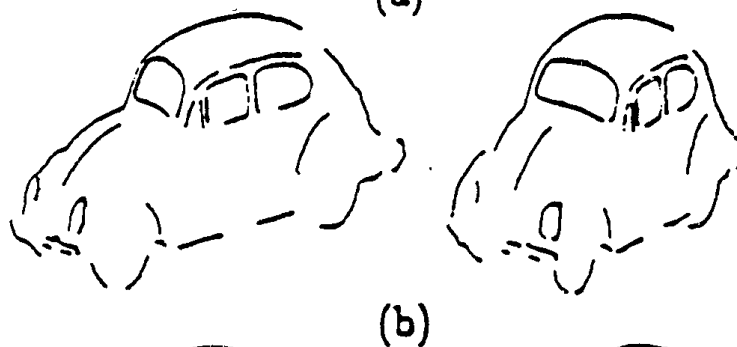
$$P = \sum_{i=1}^k c_i M_i$$

kažkokiom konstantom c_i .

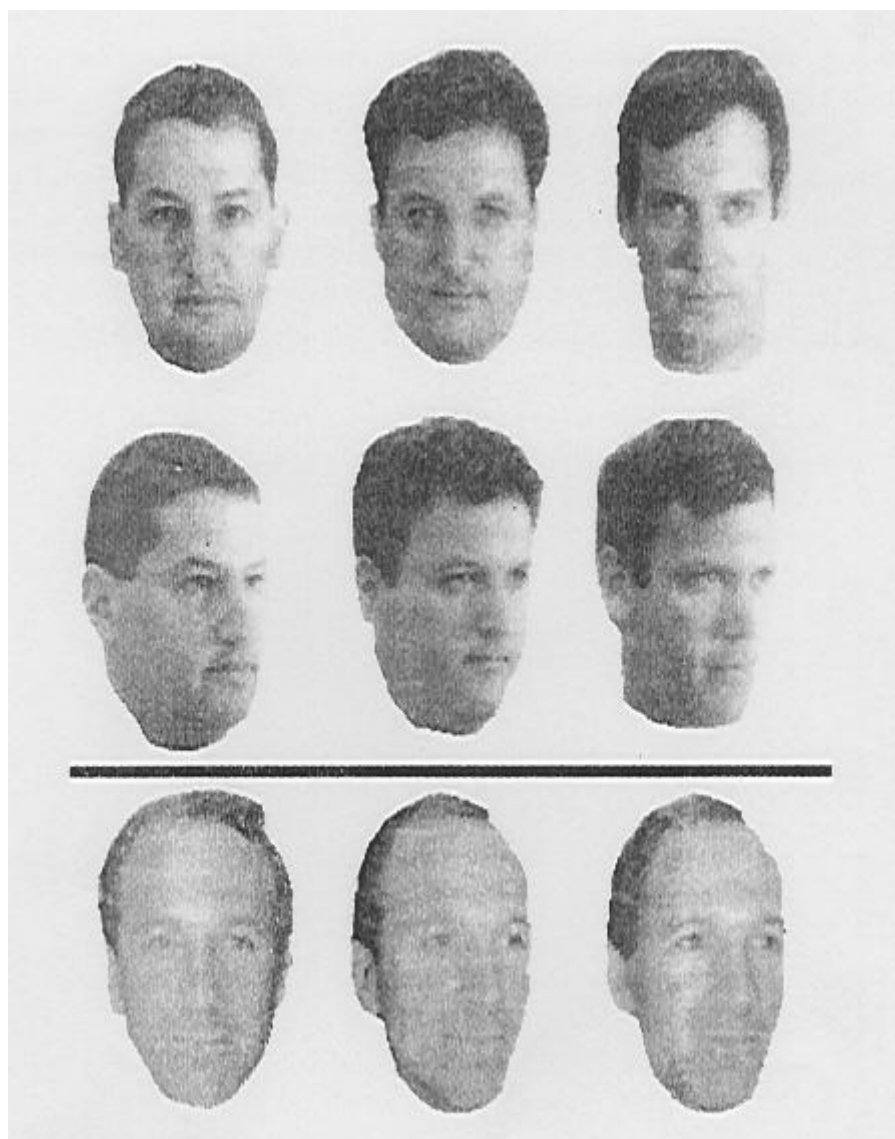
Tiesinė vaizdų kombinacija turi sekančią prasmę. Tarkim (x_i, y_i) , (x'_i, y'_i) , $(x^{''}_i, y^{''}_i)$ yra atitinkančių taškų (angl. corresponding points) koordinatės, t.y. koordinatės taškų vaizde, kurie yra kilę iš tų pačių taškų objekte, 3-juose skirtinguose vaizduose. Tegul X_1, X_2, X_3 yra x -koordinatinių vektoriai taškų 3-juose vaizduose. Tarkime, dabar mes susiduriame su nauju vaizdu, ir X' yra x -koordinatės tašku vektorius šiame vaizde. Jei X' kilęs iš to paties objekto taško, tai X' gali būti išreikštas kaip X_1, X_2, X_3 tiesinė kombinacija, t.y. $X' = a_1 X_1 + a_2 X_2 + a_3 X_3$ kažkokiom konstantom a_1, a_2, a_3 . Analogiškai y -koordinatėms, $Y' = b_1 Y_1 + b_2 Y_2 + b_3 Y_3$ kažkokiom konstantom b_1, b_2, b_3 . Taigi, bendras koordinatinių skaičius yra 6. Galime laikyti, jog kiekvienas iš 3-jų taškų $x_i, x'_i, x^{''}_i$ turi jam priklausančią svorį. Tada tiesinė taškų kombinacija yra jų svorių centras. Darbe [17] teigiama, jog 3 vaizdų gali būti pakankamai operuojant su objektais, turinčiais aštrius kontūrus (angl. sharp bounding contours), objektams, turintiems lygius kontūrus siūloma naudoti 5 vaizdus (čia yra kalbama tik apie vienos iš dviejų objekto padėčių (iš priekio ir iš galo) realizavimą).

Tiesioginis būdas panaudoti vaizdų tiesinio kombinavimo metodą yra atstatyti kombinacijos koeficientus, tada panaudojant šiuos koeficientus suformuoti objekto, naujomis sąlygomis, vaizdą ir palyginti jį su vaizdu, kurį reikia atpažinti.

2.5 pav. yra aprašomas vaizdų kombinavimo metodo panaudojimas, formuojant objekto naujomis vaizdavimo sąlygomis vaizdą. Dalyje (a) yra pateikiami 3 to paties VW tipo automobilio vaizdai ($\pm 30^\circ$ rotacijos aplink vertikalią ašį). Dalyje (b) pateikiami 2 nauji vaizdai, kurie yra gaunami vaizdų kombinacijos metodu. Dalyje (c) pateikiami realūs nauji automobilio vaizdai, kuriuos reikia atpažinti. Dalyje (d) pateikiami realių ir gautų kombinavimo metodų vaizdų palyginimo rezultatai. Rezultatai yra geri visame $\pm 30^\circ$ diapazone. Dalyje (e) pateikiami modelio (gauto vaizdų kombinavimo metodu) bei panašaus tipo (Saab) automobilio palyginimo rezultatai.



pav. 2.5



pav. 2.6

Vaizdų kombinavimo metodas leidžia gauti objekto naujomis vaizdavimo sąlygomis vaizdą, turint tik vieną objekto vaizdą vienomis vaizdavimo sąlygomis. Tai atliekama remiantis klasės generalizacija. 2.6 pav. virš horizontalios linijos yra pateikiami trijų skirtingų veidų vaizdai (kiekvienas dvejomis vaizdavimo sąlygomis – iš priekio ir iš šono). Po horizontalia linija yra pateikiama: (iš kairės) realus naujas veido vaizdas iš priekio, (iš dešinės) sugeneruotas veido vaizdas iš šono (šis vaizdas gaunamas taikant vaizdų kombinavimo metodą vaizdams virš horizontalios linijos bei naujam veido vaizdai iš priekio), bei (viduryje) palyginimui yra pateikiamas realus vaizdas iš šono.

3 Automobilio sekimas vaizdų sekoje

Kaip jau buvo minėta, automobilio sekimo procesas susideda iš dviejų atskirų procesų: judesio išskyrimo ir prognozės.

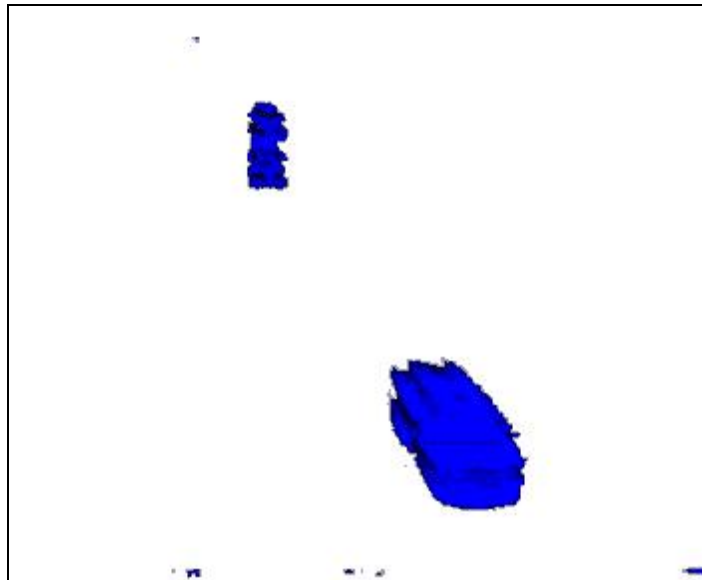
Siūlomo sekimo algoritmo žingsnius galima būtų apibrėžti taip:

1. Išskirti judesio taškus vaizde.
2. Apjungti judesio taškus į judančius objektus (automobilius).
3. Pažymėti judančius objektus (Kalmano filtro pagalba).

Automobilių atpažinimo sistema - tai eismo stebėjimo sistema, t.y. operuojama su statine kamera, todėl fono vaizdai gali būti nesunkiai gauti. Šiame darbe aprašytoje sistemoje judesiui išskirti yra naudojamas fono atimties metodas. Šio metodo panaudojimo pagrindinė priežastis yra jo vykdymo greitumas. 3.1 pav. yra pavaizduotas nagrinėjamas kadras iš vaizdų sekos, o 3.2 pav. judesio algoritmo metodo rezultatas.



pav. 3.1



pav. 3.2

Po to, kai judesys buvo išskirtas, reikia apjungti judančius taškus į objektus (t.y. automobilius). Po to reikia įvykdyti žymėjimo žingsnį, t.y. nustatyti ar šis automobilis yra naujas objektas sekimui, ar jis jau buvo sekamas ir praeitame kadre. Kitaip sakant, žymėjimo žingsnio tikslas yra nustatyti, kokiam objektui (naujam ar jau sekamam) priklauso kiekvienas vaizdo taškas. Tam, kad nusakyti, ar judantis automobilis jau buvo sekamas praeitajame kadre, yra naudojamas Kalmano filtras. Kalmano filtras prognozuoja judančio objekto koordinatės sekančiame kadre, todėl gaunant naują kadrą yra iš karto aišku, kur reikėtų šio objekto ieškoti. Jei objektas nebuvo surastas (naudojamas kažkoks slenkstis), tai laikoma, kad objektas pradingo iš kameros akiračio. Tada objektui yra didinamas skaitliukas, kuris skaičiuoja, kiek kadrų mes jau nematėm objekto. Tačiau objekto koordinatės yra prognozuojamos iki to momento, kol skaitliukas nepasieks kažkokios maksimalios reikšmės (pvz. 20 kadrų). Kalmano filtro sistemos būsenos vektorius yra apibrėžiamas taip:

$$x(k) = (\Delta x_center(k), \Delta y_center(k), \Delta xsize(k), \Delta ysize(k))^T$$

Po kiekvieno filtro žingsnio tikslūs matavimai yra gaunami iš vaizdo. Jei objektas nebuvo susektas, jo tiksliais matavimais laikoma Kalmano filtro prognozė. Detalesnis Kalmano filtro aprašymas yra pateiktas 2.1.2 skyriuje.

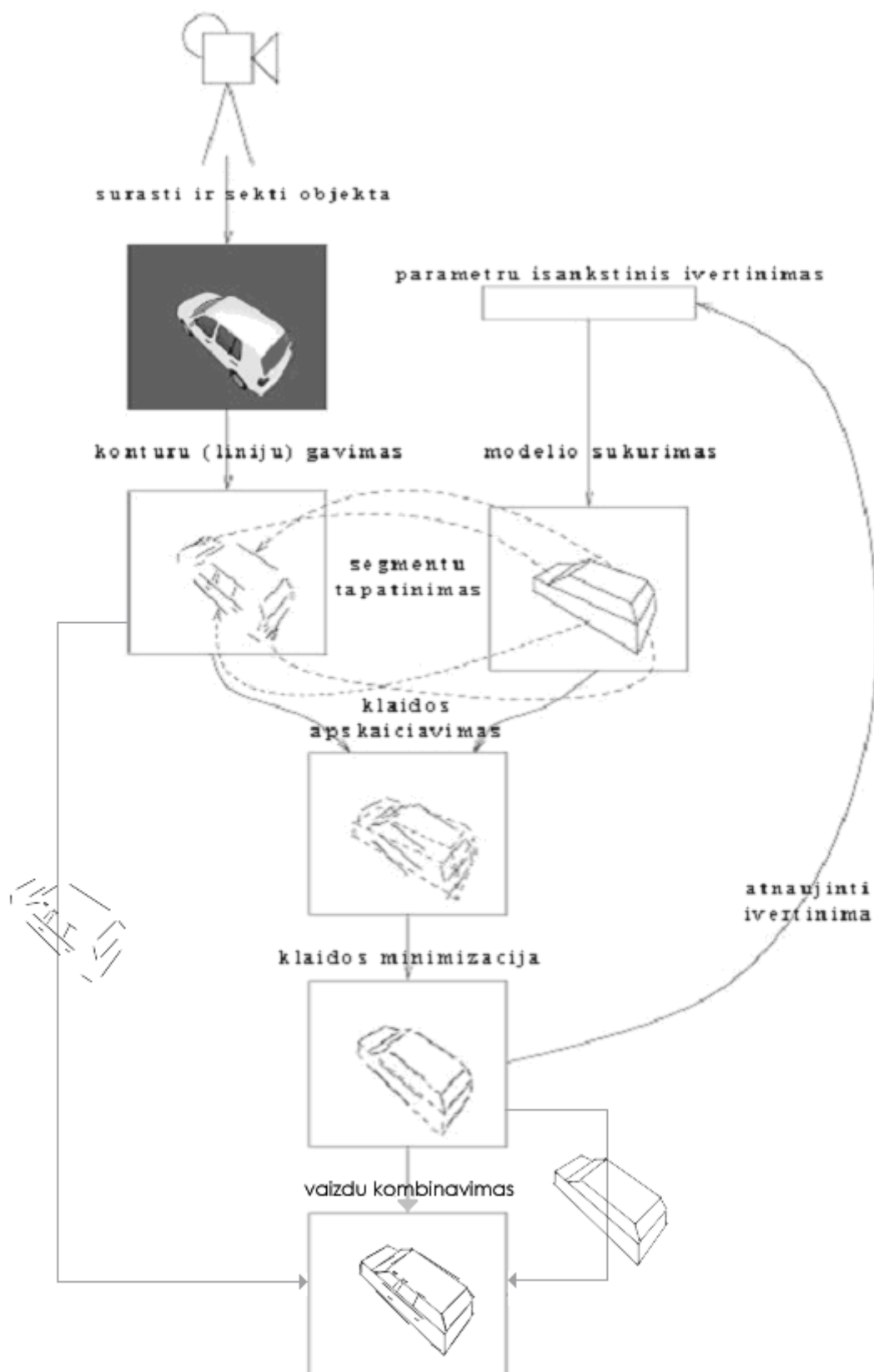
4 Automobilio tipo atpažinimas

Šiame skyriuje sprendžiama pagrindinė šio darbo problema – objekto (t.y. automobilio) atpažinimas.

Siūlomo automobilio tipo atpažinimo algoritmo žingsniai:

1. Aptikti objekto kontūrus (kraštus) vaizde (naudojant Canny metodą).
2. Hofo transformacijos pagalba aptikti tiesių linijų segmentus (toliau - sekamo objekto linijų segmentai).
3. Sukurti pradinį 3-D modelį remiantis pradiniais duomenimis apie objekto translacią bei rotaciją (čia jau turi būti apibrėžti kameros kalibravimo parametrai).
4. Išrinkti atitinkančius vienas kitam segmentus tarp sekamo objekto linijų segmentų (2-o žingsnio veikimo rezultatas) bei sukurto 3-D modelio 2-D projekcijos vaizde linijų segmentų (3-čio žingsnio veikimo rezultatas).
5. Modifikuoti pradinį modelį tokiu būdu, kad minimizuoti klaidą, apibrėžiančią skirtumus tarp 4-ame žingsnyje parinktų atitinkamų segmentų.
6. Jei reikia, tai grįžtama prie 3-čio žingsnio, sukuriant kitos klasės automobilio modelį.
7. Nustatoma pagrindinė automobilio klasė remiantis nustatytais optimalaus (t.y. minimizuojančio klaidą) 3-D modelio parametrais.
8. Vaizdų kombinacijos metodo pagalba yra sukuriamas pilnas automobilio modelis, kuris formuojamas iš vaizdo, gauto po 2-o žingsnio (objekto kontūrų segmentai) bei optimalaus modelio (po 7-o žingsnio).
9. Lyginant pilną automobilio modelį su duomenų bazėje turimais šios klasės automobilių pilnais modeliais, nustatoma automobilio poklasė.

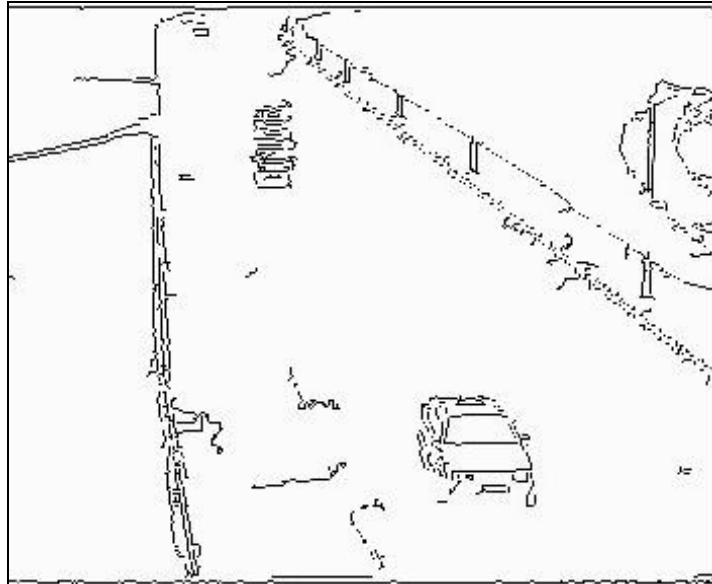
Siūlomas automobilio tipo atpažinimo algoritmas yra *4.1 pav.*:



pav. 4.1

4.1 Sekamo objekto aprašymas

Kaip jau minėta, sekimo žingsnis grąžina judančio objekto koordinates vaizde. Tai reiškia, jog tiksliai yra žinoma, kurią vaizdo sritį reikia atpažinti. Canny metodas, aprašytas 2.2.1.2 skyriuje, leidžia aptikti judančio objekto kraštus, t.y. ribas, atskiriančias nagrinėjamos vaizdo dalies sritis, turinčias skirtingas ryškumo spalvas. Analizuojant vaizdą, kuris yra 3.1 pav., Canny metodo vykdymo rezultate gaunamas vaizdas 4.2 pav.:



pav. 4.2

Tam, kad aptikti tiesių linijų segmentus, šiam vaizdai yra naudojamas metodas Hough transformacijos pagrindu.

4.2 3-D modelis

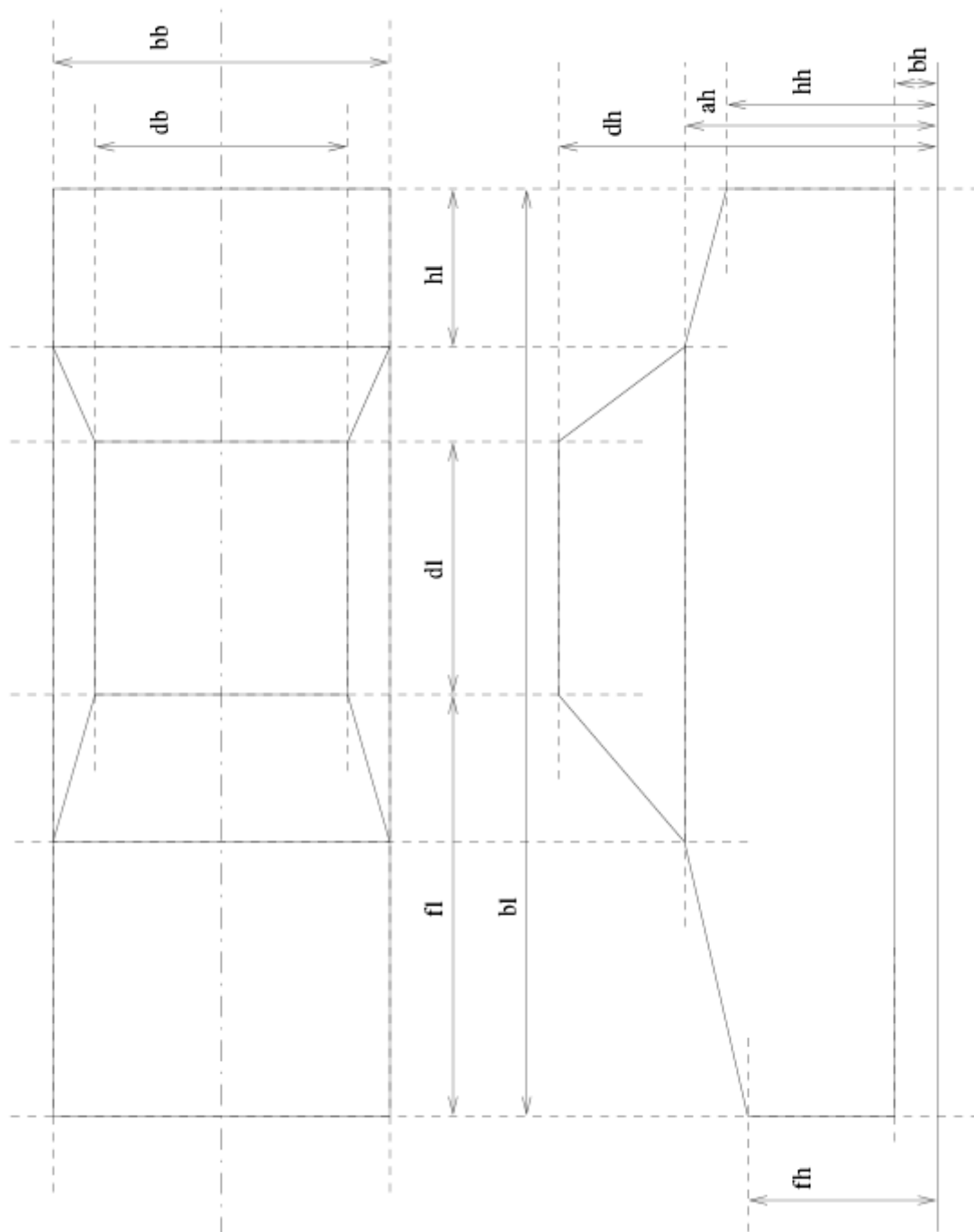
Šis skyrius aprašo 3-D modelį, kurį reikia sukurti siūlomo algoritmo 3-ame žingsnyje. Tam, kad sėkmingai sukurti pradinį modelį tolimesniam palyginimui, reikia aprašyti kintantį 3-D modelį, kurio padėti bei formą galima būtų aprašyti baigtine parametru aibe.

Darbe [16] yra siūlomas 17-os parametru automobilio aprašymo būdas. Automobilio 3-D modelį sudaro 12 vidinių parametru, kurie apibrėžia modelio formą, bei 5 išoriniai parametrai, kurie apibrėžia kuriamo modelio rotacijos bei transliacijos (postūmio) reikšmes.

4.2.1 Vidiniai parametrai

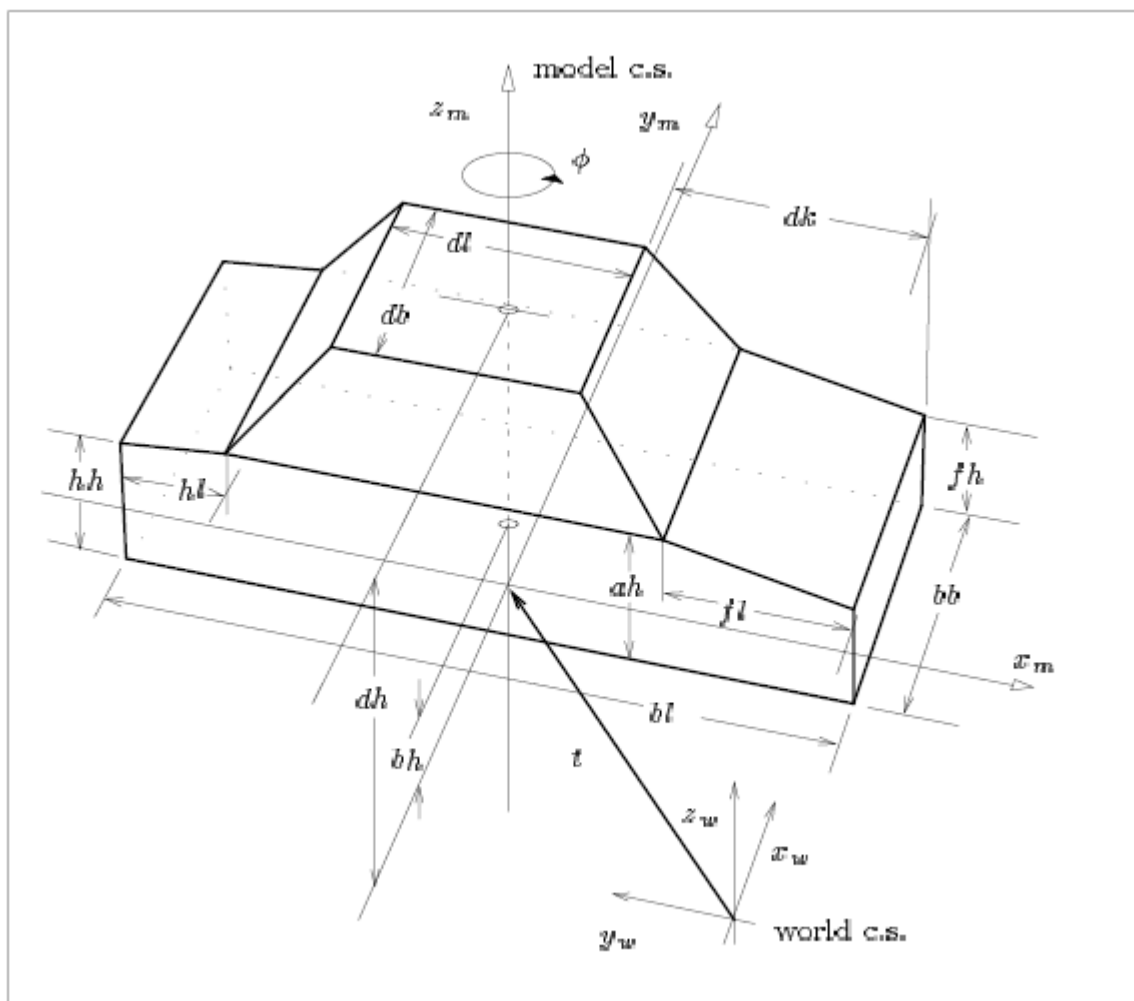
Vidinių automobilio modelį sudaro 12 ilgio parametru. Įvairių parametru įtaką prototipo formai atspindi 4.3 pav. bei 4.4 pav. Šių parametru rinkinį patogiau užrašyti kaip vektorių (formos parametru vektorių):

$$pf = (bl, bb, bh, dl, db, dh, dk, fl, fh, hl, hh, ah)^T$$



pav. 4.3

4.2.2 Išoriniai parametrai



pav. 4.4

Išorinį prototipo modelį sudaro 2 rotacijos bei 3 translacijos parametrai. Bendrai taikant 3-D rotaciją yra 2 skirtingi susitarimai: objekto rotaciją, kai ašys yra fiksuotos, arba kai rotacija yra taikoma ašims. Šiame darbe rotacija yra apibrėžiama kaip kameros rotacija, t.y. rotacija yra taikoma ašims. Modelio rotacijos bei translacijos reikšmės lieka tos pačios. Gerai žinomas kameros modelis yra PTZ (Pan-Tilt-Zoom). Fiksuojame y-ašį, t.y. lieka tik 2 rotacijos parametrai iš 3-jų. Rotacija aplink visas ašis yra apibrėžiama per rotacijos matricas (yra svarbu, kokia tvarka vykdyti rotacijas):

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix}$$

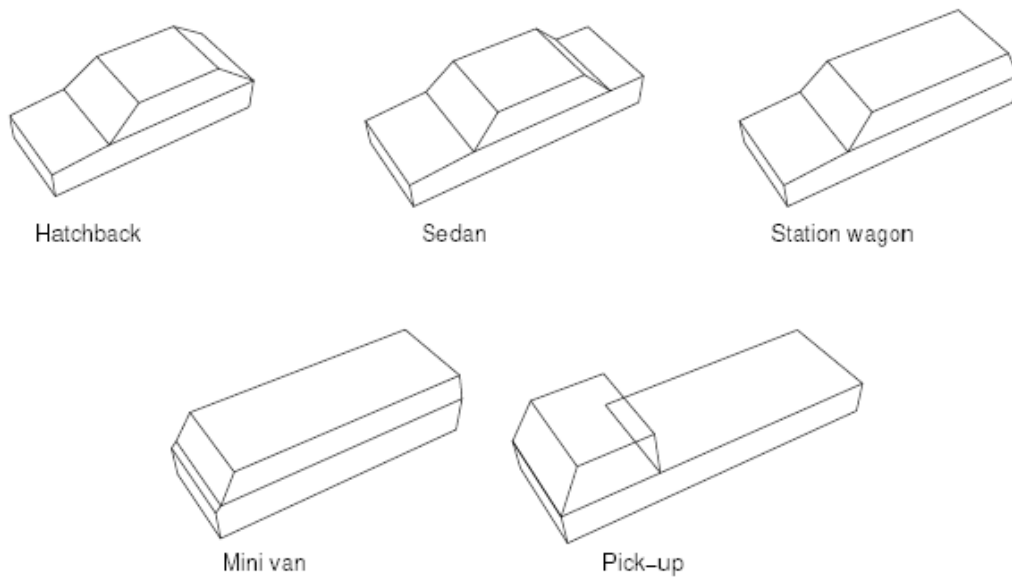
$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.2.3 Pilnas 3-D modelis

Pilnas prototipo modelis yra sudarytas iš 17 parametrų. Turint 17 modelio parametrų belieka suprojektuoti modelį pagal 12 parametrų objekto erdvėje, po to gauti kameros erdvės (angl. camera space) koordinatas įvykdant rotacijas bei translacijas pagal likusius 5 parametrus. Projektuojant 3-D modelį į 2-D vaizdą reikia atkreipti dėmesį į tai, kad kai kurios objekto dalys turi būti „nematomos“, t.y. kaž kurios objekto dalys yra arčiau mūsų, todėl uždengia objekto sritis, kurios yra toliau. Šiam uždaviniui spręsti dažniausiai yra naudojamas *Z-buffering* algoritmas.

Penkios pagrindinės automobilių klasės, jų parametrų reikšmės bei laukiamos parametrų reikšmių variacijos (pasikeitimų ribos vienoje klasėje) yra atvaizduotos 4.5 pav. bei 4.6 pav.:



pav. 4.5

s	Hatchback		Sedan		Stationwagon		Mini-van		Pick-up	
	$\mu_{\tilde{s}}$	$\sigma_{\tilde{s}}$	$\mu_{\tilde{s}}$	$\sigma_{\tilde{s}}$	$\mu_{\tilde{s}}$	$\sigma_{\tilde{s}}$	$\mu_{\tilde{s}}$	$\sigma_{\tilde{s}}$	$\mu_{\tilde{s}}$	$\sigma_{\tilde{s}}$
bl	3.80	0.80	4.40	1.00	4.40	1.00	4.60	1.00	5.50	1.20
bb	1.50	0.10	1.60	0.10	1.60	0.10	1.60	0.10	1.80	0.20
bh	0.30	0.10	0.30	0.10	0.30	0.10	0.30	0.10	0.60	0.20
dl	1.50	0.30	1.60	0.30	2.60	0.30	4.10	0.50	1.20	0.30
db	1.30	0.10	1.40	0.10	1.40	0.10	1.50	0.10	1.60	0.10
dh	1.40	0.10	1.40	0.10	1.40	0.10	1.50	0.10	1.90	0.20
dk	1.60	0.30	1.60	0.30	1.60	0.30	0.40	0.10	0.40	0.10
fl	1.00	0.30	1.10	0.30	1.10	0.30	0.10	0.10	0.00	0.10
fh	0.50	0.10	0.50	0.10	0.50	0.10	0.60	0.10	0.60	0.10
hl	0.00	0.20	0.70	0.10	0.00	0.10	0.00	0.10	3.90	1.00
hh	0.60	0.10	0.55	0.10	0.60	0.10	0.70	0.10	0.60	0.10
ah	0.60	0.10	0.60	0.10	0.60	0.10	0.70	0.10	0.60	0.10

pav. 4.6

4.3 Atitinkamų segmentų parinkimas (angl. segment matching)

Tam, kad būtų įmanoma priderinti pradinio modelio segmentus prie sekamo automobilio segmentų (kas yra reikalaujamas 5-ame siūlomo algoritmo žingsnyje), reikia parinkti vienas kitą atitinkančių segmentų poras. Sekančiame atpažinimo algoritmo žingsnyje, kai bus vykdomas parametrų priderinimas, bus analizuojami tik tie segmentai, kuriems buvo parinkta pora.

4.3.1 Atstumas tarp vidurio taškų

Skaiciuojame kiekvieno segmento vidurio taškus:

$$\mathbf{m}_{ix} = \frac{\mathbf{x}_{is} + \mathbf{x}_{ie}}{2}$$

$$\mathbf{m}_{jy} = \frac{\mathbf{y}_{js} + \mathbf{y}_{je}}{2}$$

Dabar skaiciuojame atstumus tarp visų įmanomų segmentų porų vidurio taškų:

$$d_{ij} = \|\mathbf{m}_{ix} - \mathbf{m}_{jy}\|_2$$

Tie segmentai laikomi *atitinkančias* vienas kitą, atstumas tarp kurių yra mažiausias.

$$d_{ij} \leq d_{ik}, \text{ visiems } k$$

Šis metodas yra gana paprastas ir grąžins gerus rezultatus tik, jei skirtumai tarp lyginamų segmentų bus maži.

4.3.2 Centruotas atstumas tarp vidurio taškų

Yra būdas pagerinti 4.3.1 skyriuje paminėtą metodą. Metodą galima pagerinti panaikinant vidurkius iš lyginamų segmentų aibių:

$$\mathbf{m}_X = \frac{1}{n} \sum_{i=1}^n \mathbf{m}_{ix}$$

$$\mathbf{m}_Y = \frac{1}{m} \sum_{j=1}^m \mathbf{m}_{jy}$$

$$d_{ij} = \| (\mathbf{m}_{ix} - \mathbf{m}_X) - (\mathbf{m}_{jy} - \mathbf{m}_Y) \|_2$$

4.4 Parametrų priderinimas

Parametrų priderinimas gali būti apibrėžtas kaip tokių parametrų apskaičiavimas, kurie minimizuoja apibrėžtą klaidą. Mūsų atveju parametrai, kurie yra derinami, tai sekamo automobilio rotacijos, translacijos bei formos parametrai. Klaidos apibrėžimas remiasi skirtumų tarp sekamo automobilio linijų segmentais ir segmentais iš šabloninio 3-D modelio.

Tam, kad minimizuoti klaidą tarp sekamo automobilio linijų segmentų (toliau sekamų segmentų) ir šabloninio 3-D modelio linijų segmentų (toliau šablono segmentų) reikia apibrėžti šią klaidą. Darbe [16] klaidos apibrėžimui naudojami segmentų vidurio taškai, kryptis ir ilgiai.

Darbe [5] yra siūlomas klaidos apibrėžimas, kuris susideda iš 3-jų komponentių. Dvi klaidos komponentės yra apibrėžiamos kaip statmeni atstumai tarp šablono segmento pradžios (pabaigos) taškų ir tiesės, apibrėžtos sekamo segmento. Trečia klaidos komponentė yra apibrėžiama kaip atstumas tarp derinamų segmentų vidurio taškų. Aukščiau išdėstytų klaidų apibrėžimas yra pavaizduotas 4.7 pav. Taškai s ir e yra šablono segmento pradžios ir pabaigos taškai atitinkamai. Taškas m yra šio segmento vidurio taškas ir yra apibrėžiamas kaip:

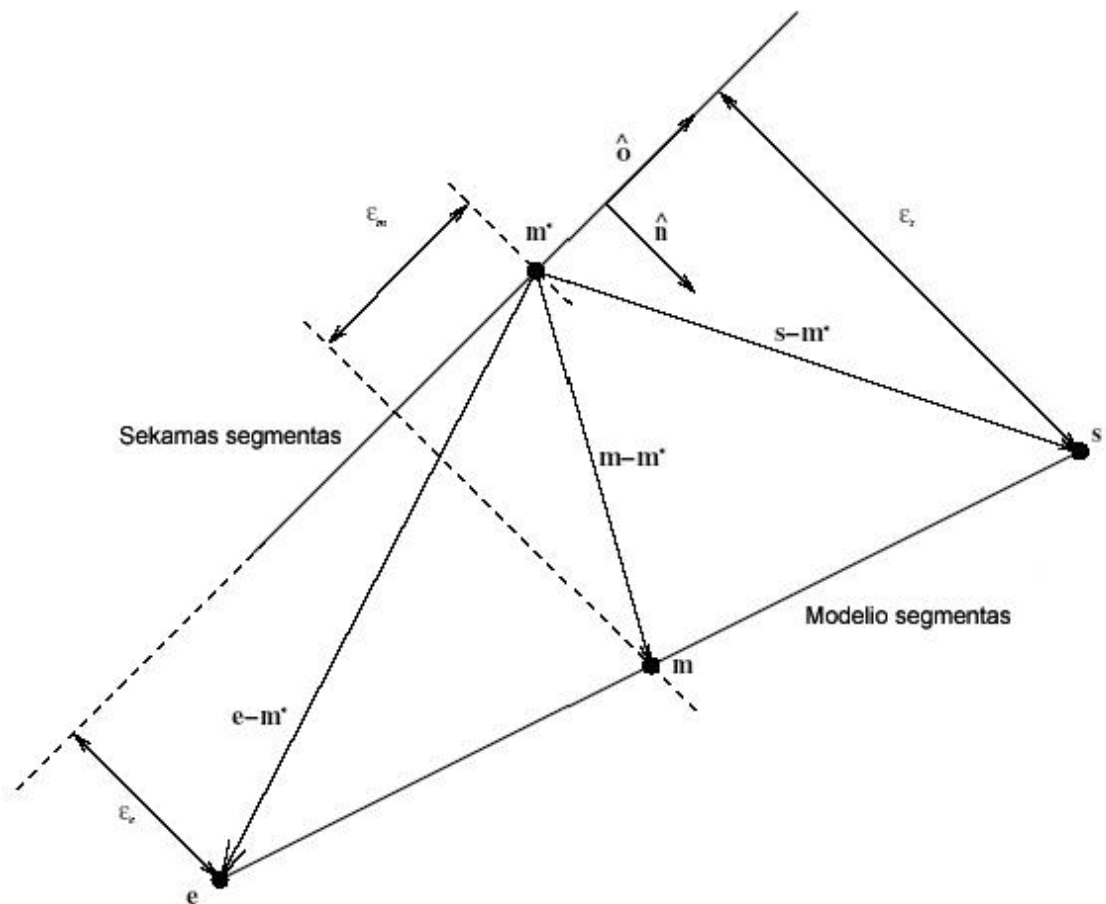
$$m = \frac{1}{2}(s + e),$$

Taškas m^* yra sekamo segmento vidurio taškas ir yra apibrėžiamas analogiškai taškui m .

Trys klaidos komponentės gali būti apjungtos į klaidos vektorių ε , t.y.:

$$\varepsilon = \begin{pmatrix} \varepsilon_s \\ \varepsilon_e \\ \varepsilon_m \end{pmatrix}$$

kur ε_s ir ε_e yra statmeni atstumai nuo segmento pradžios (ir atitinkamai pabaigos) taškų. ε_m yra atstumas tarp derinamų segmentų vidurio taškų.



pav. 4.7

Klaida ε yra funkcija, kuri priklauso nuo šabloninio modelio segmento pradžios ir pabaigos taškų, t.y.:

$$\varepsilon = \varepsilon(s, e).$$

Geras metodas šiai klaidai minimizuoti yra siūlomas [5]. Aprašomas metodas remiasi Niutono iteracijų metodu (reikalauja Jakobiano (angl. Jacobian) apskaičiavimo), kuris irgi yra išdėstytas [5]. Be to, [5] yra siūlomas ir kitas Levenberg-Marquardt iteracijų metodas.

Minimizuojant klaidą dažnai reikia įvesti papildomus sprendimo apribojimus. Darbe [5] siūloma fiksuoti linijas, t.y. keisti tik linijos ilgio (bet ne rotacijos ar transliacijos) parametrus.

Darbe [16] klaidos minimizacijai yra siūloma naudoti išplėstą Kalmano filtrą (EKF).

Šio žingsnio pagalba galime nustatyti automobilio klasę bei sužinoti optimalaus 3-D modelio formos bei padėties parametrus.

4.5 Vaizdų kombinacijos metodas

Metodas, aprašytas [17], leidžia gauti objektą naujomis vaizdavimo sąlygomis (angl. view conditions) turint kelis objekto skirtingomis vaizdavimo sąlygomis vaizdus. Po parametrų

priderinimo (5-as žingsnis) algoritmo žingsnio mes turime optimalaus modelio parametrus. Tai reiškia, jog įmanoma gauti šio optimalaus modelio 2-D vaizdus bet kuriomis vaizdavimo sąlygomis. Be to, turime ir pradinius sekamo objekto 2-D linijų segmentus, kuriuos gavome po Hofo (2-as atpažinimo algoritmo žingsnis) transformacijos panaudojimo. Panaudojant [17] vaizdų kombinavimo metodą šiems vaizdams yra gaunamas pilnas automobilio modelis, kuris leidžia nustatyti automobilio poklasę. Poklasių pradiniai vaizdai turi būti įvesti tiesiogiai, t.y. naudojamas tiesioginio apmokymo (angl. supervised learning) principas. Poklasės nustatymui vaizdai yra lyginami tiesiogiai, t.y. lygiais yra laikomi tie kontūrai, kurie turi daugiausiai vienodų taškų.

5 Testavimų rezultatai ir išvados

Darbe aprašyta metodų sistema susideda iš kelių metodų. Kiekvieno metodo veikimas priklauso ne tik nuo *jam* parinktų parametrų (pvz., slenksčio), bet ir nuo parametrų, nustatytų kitiems metodams. Darbo eigoje buvo išbandytos skirtingos parametrų kombinacijos. Sekimo žingsnio rezultatai yra gana geri, tačiau atpažinimo metodas veikia žymiai prasčiau. Pagrindinė problema vykdant atpažinimą yra ta, kad yra labai nedaug tikslios informacijos apie objektą (automobilį). Tam, kad gauti geresnį objekto aprašymą galima būtų operuoti su didesnės rezoliucijos pradiniais vaizdais, tačiau kompiuterinės regos algoritmai yra gana brangūs skaičiavimo prasme, todėl didelės rezoliucijos vaizdų sekos apdirbimas realiaame laike yra gana komplikauta užduotis.

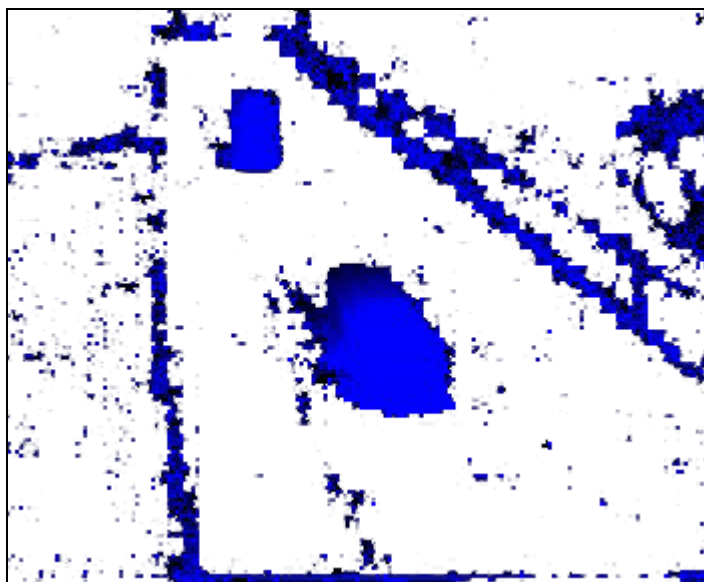
5.2-5.6 pav. yra pavaizduoti judesio išskyrimo algoritmo veikimo rezultatai panaudojant skirtingas judesio slenksčio reikšmes (taikant vaizdai 5.1 pav.). 5.1 lentelėje yra užrašytos slenksčių reikšmės kiekvienai iliustracijai:

pav.	slenkstis
5.2	10
5.3	20
5.4	30
5.5	40
5.6	50
5.7	60

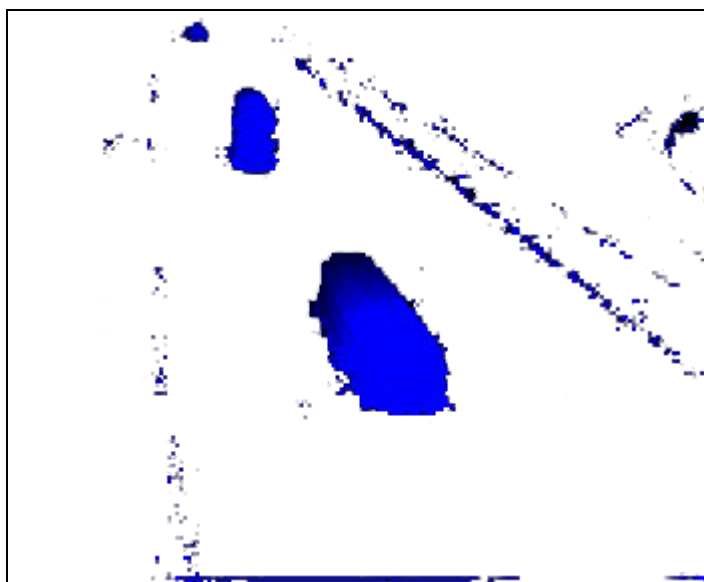
lentelė 5.1



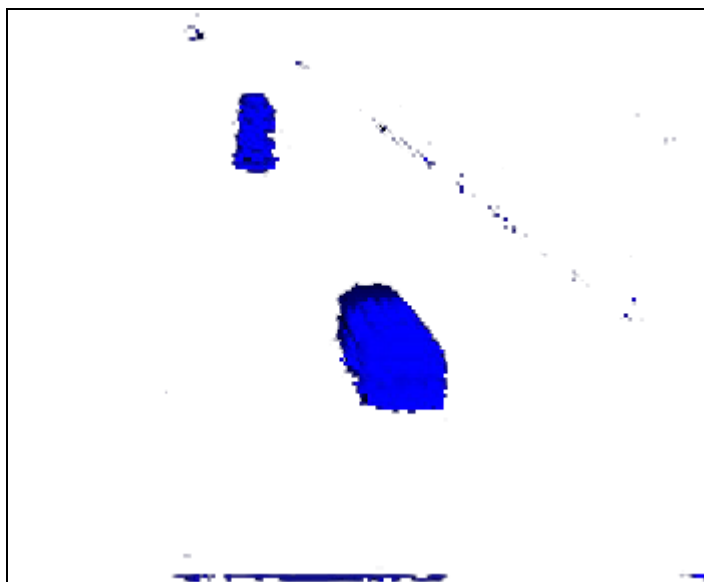
pav. 5.1



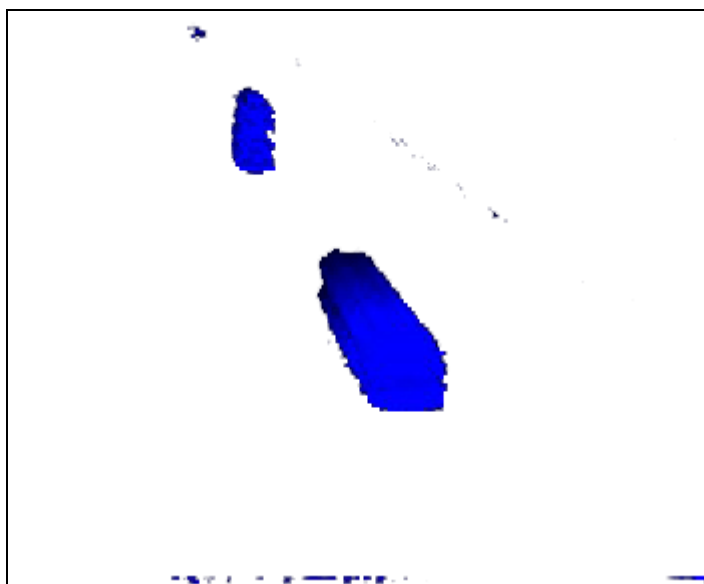
pav. 5.2



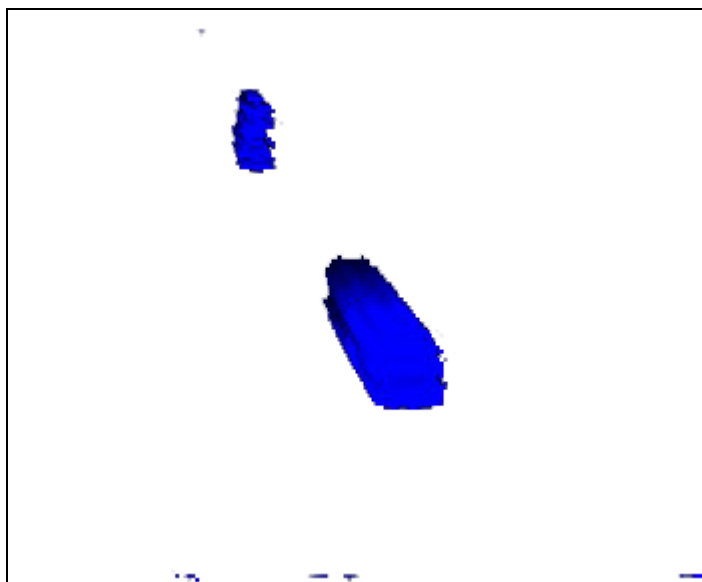
pav. 5.3



pav. 5.4



pav. 5.5



pav. 5.6



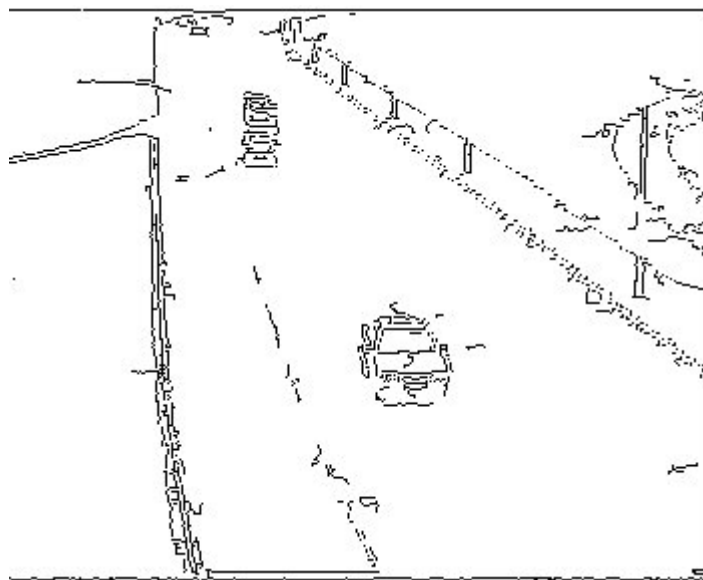
pav. 5.7

Kaip matyti, kai slenksčio reikšmė yra mažesnė už 30, tai vaizde lieka daug triukšmo, kai reikšmė pakyla virš 50 - rezultatai nebesikeičia. Taigi, šiam metodui vertėtų naudoti slenksčio reikšmę nuo 30 iki 50.

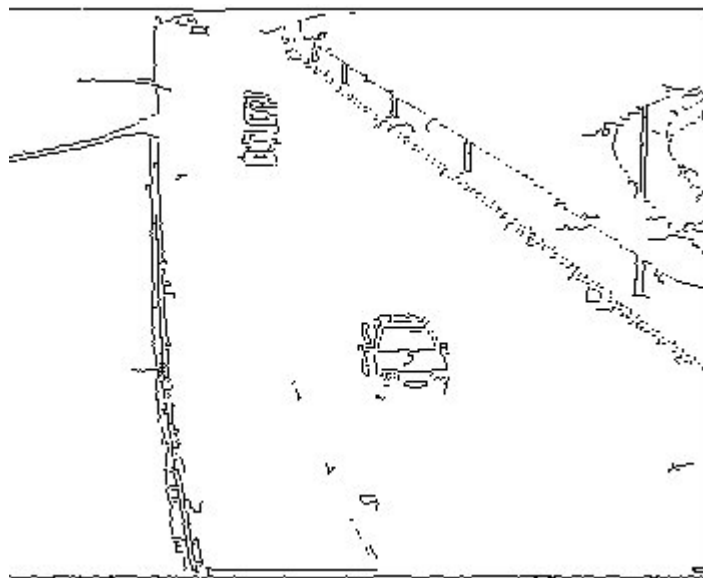
Kraštams aptikti darbe yra naudojamas Canny metodas. 5.8-5.11 pav. yra pavaizduoti Canny kraštų išskyrimo algoritmo veikimo rezultatai panaudojant skirtingas slenksčių bei apertūros reikšmes (taikant vaizdai 5.1 pav.). 5.2 lentelėje yra užrašytos parametrų reikšmės kiekvienai iliustracijai:

pav.	apatinis slenkstis	viršutinis slenkstis	naudojamo Sobel filto apertūros dydis
5.8	0	200	3
5.9	0	250	3
5.10	50	250	3
5.11	50	250	5

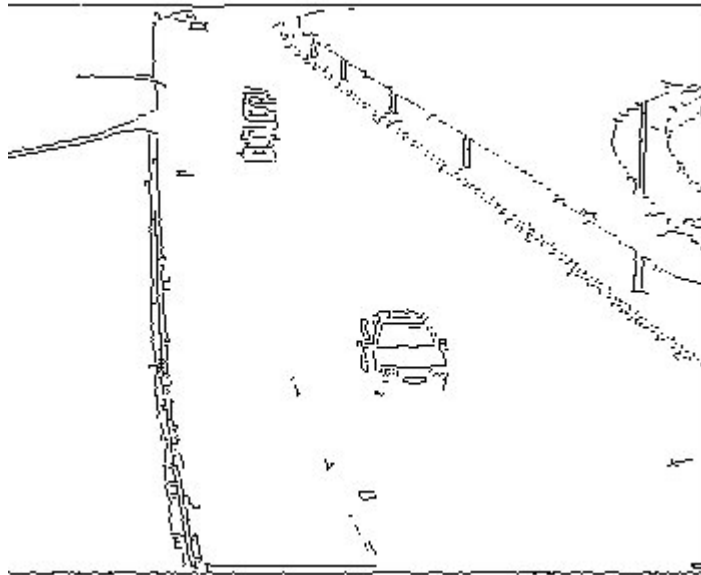
lentelė 5.2



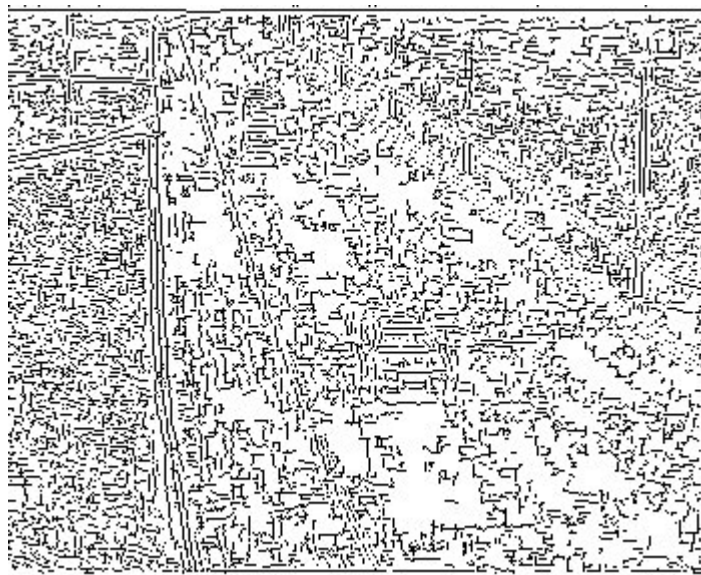
pav. 5.8



pav. 5.9



pav. 5.10



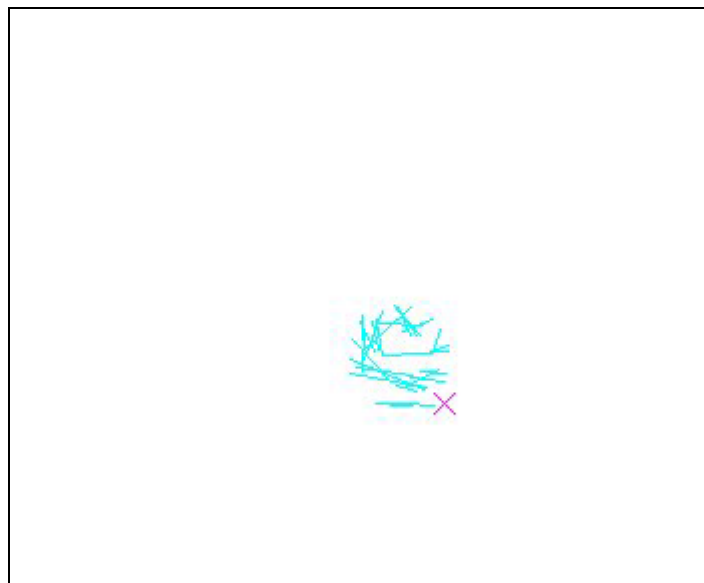
pav. 5.11

Viršutinio ir apatinio slenksčio pasirinkimas priklauso nuo sekamo objekto dydžio, vaizdo kokybės bei rezoliucijos. Canny algoritme naudojamo Sobel filtro apertūros dydžio reikšmė turėtų būti lygi 3.

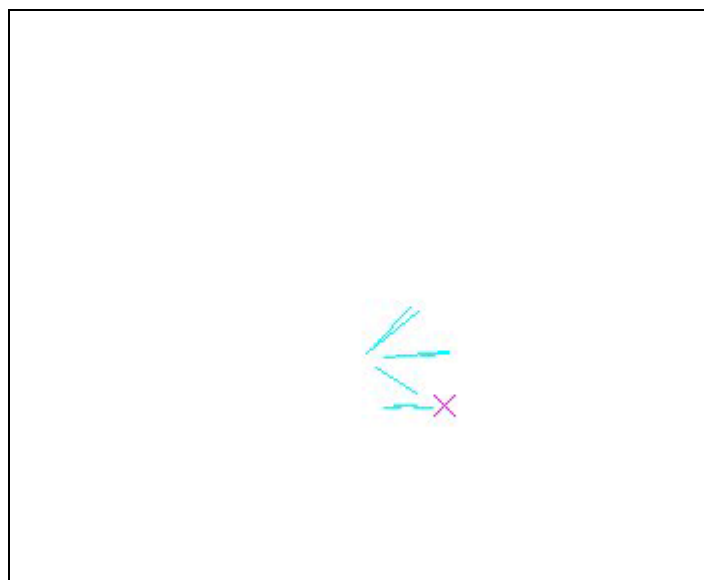
Linijų segmentams aptikti yra naudojamas PPHT metodas. 5.12-5.16 pav. yra pavaizduoti PPHT tiesių segmentų aptikimo algoritmo veikimo rezultatai panaudojant skirtingas parametrų reikšmes (taikant vaizdai 5.1 pav.). 5.3 lentelėje yra užrašytos parametrų reikšmės kiekvienai iliustracijai:

pav.	minimalus segmento ilgis	maksimalus tarpas
5.12	5	5
5.13	10	5
5.14	10	10
5.15	15	5
5.16	15	10

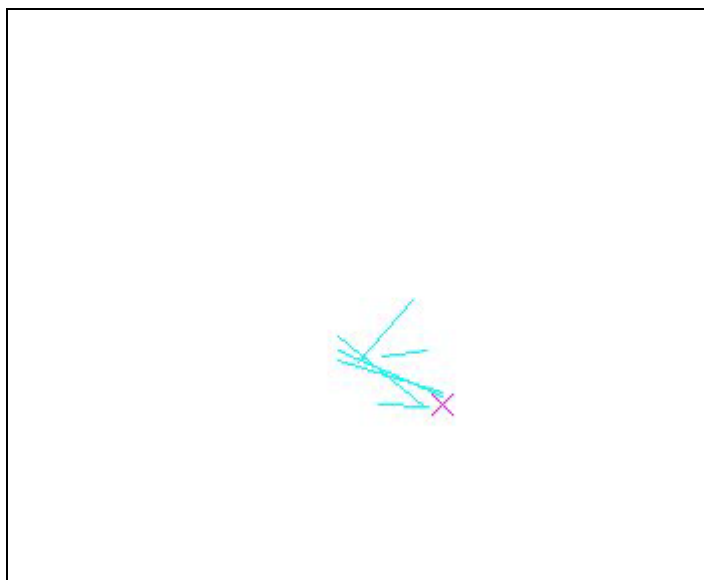
lentelė 5.3



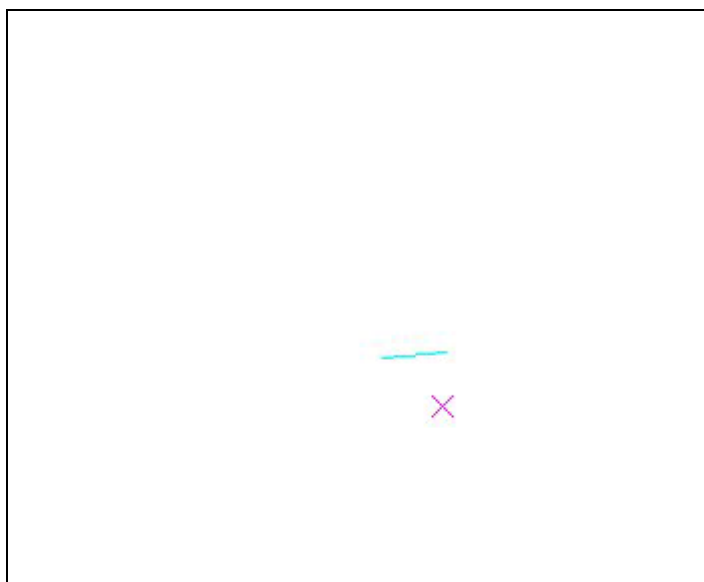
pav. 5.12



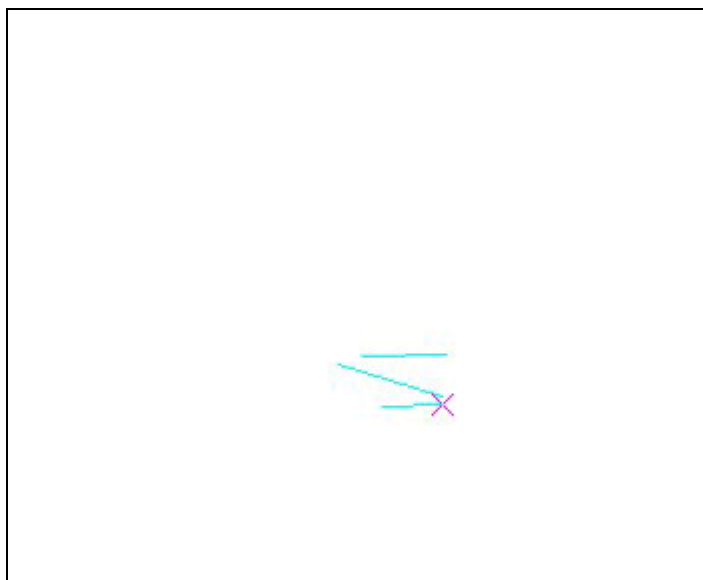
pav. 5.13



pav. 5.14

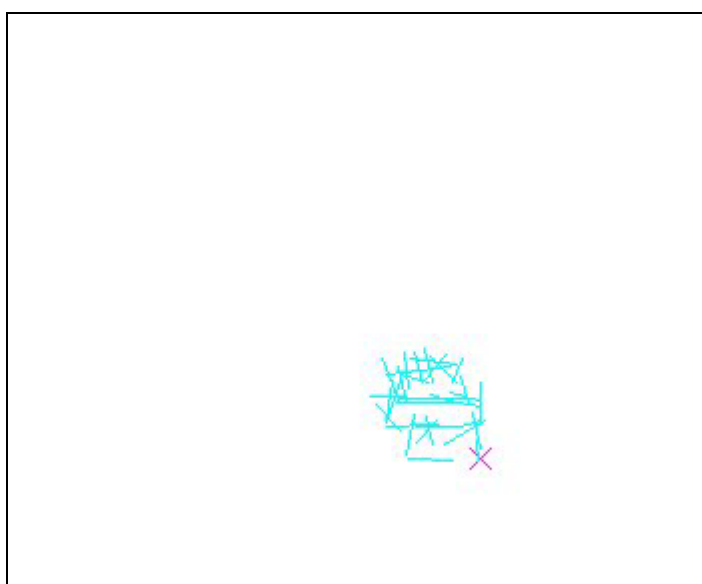


pav. 5.15

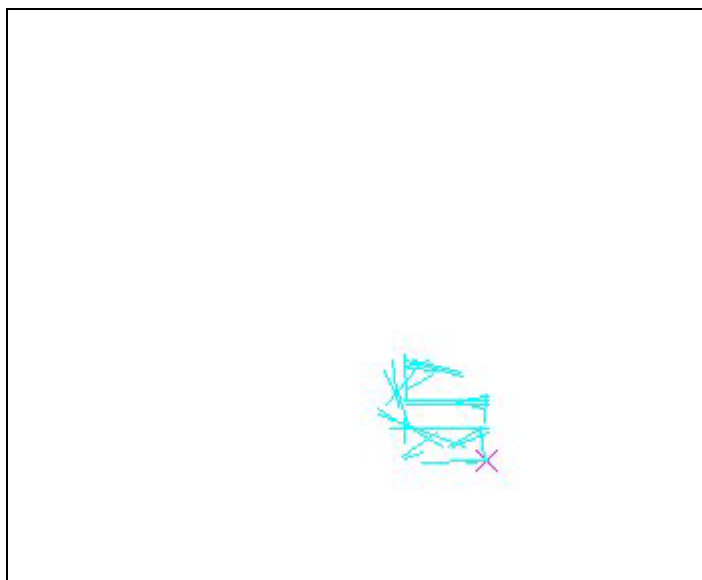


pav. 5.16

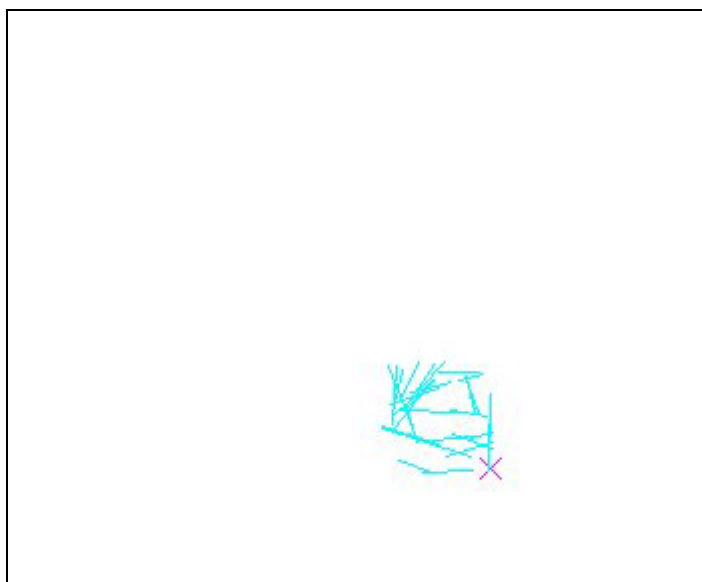
Darbe naudojamai vaizdo medžiagai geriausi rezultatai buvo gaunami naudojant parametrus 5 ir 5. Žemiau (5.17-5.22 *pav.*) yra pavaizduoti PPHT algoritmo panaudojimo, keliems vienas po kito einantiems vaizdo sekos kadrams, rezultatai:



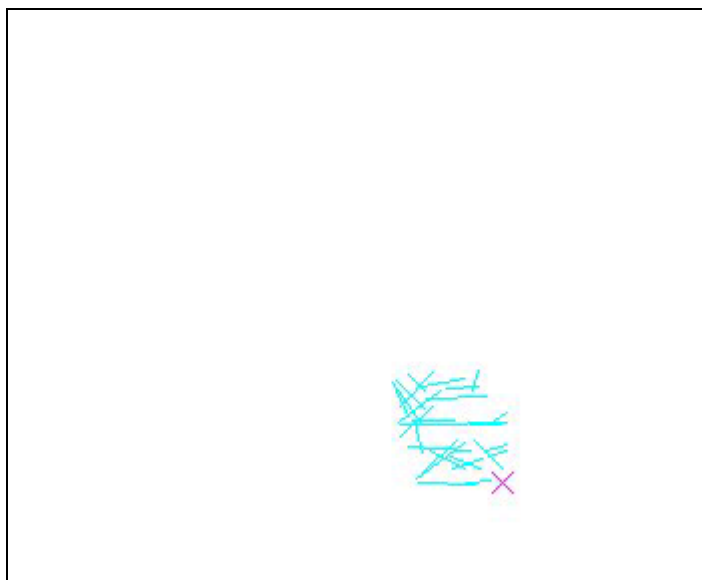
pav. 5.17



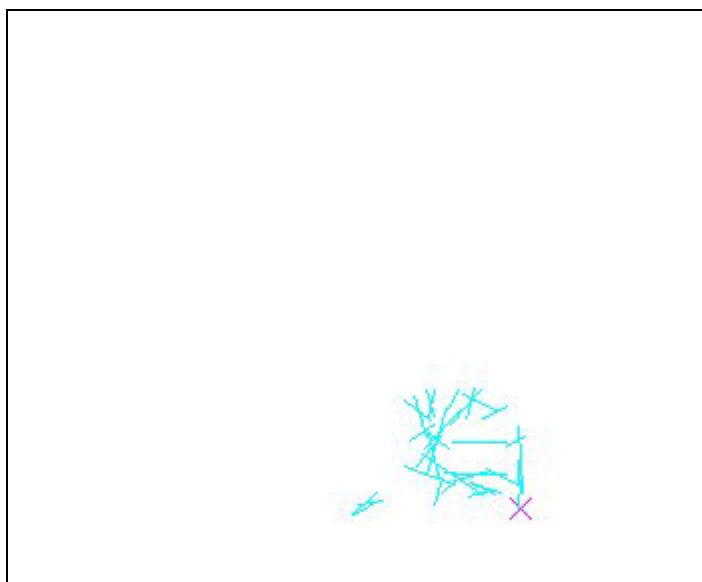
pav. 5.18



pav. 5.19



пав. 5.20

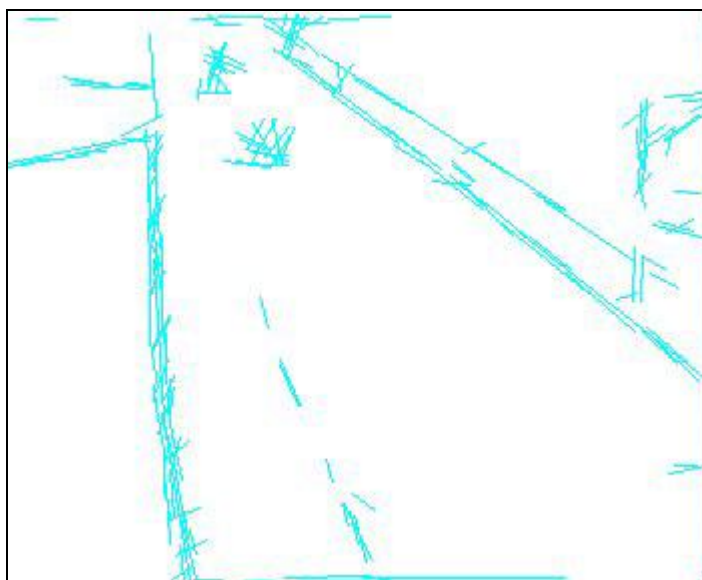


пав. 5.21



pav. 5.22

PPHT taikymo rezultatai visam kadrai yra pavaizduoti 5.23 pav.:



pav. 5.23

Iš atvaizduotų rezultatų matosi, jog norint turėti pakankamai gerą sekamo automobilio formos aprašymą, sekamas automobilio vaizdas kadre neturėtų būti mažesnis už ~ 50 taškų.

Darbe aprašytas algoritmas nors ir galėtų būti taikomas automobilio tipo atpažinimui, tačiau tam, kad grąžinami rezultatai būtų kuo tikslesni, nagrinėjama vaizdų seka turėtų būti kuo geresnės kokybės. Nuo vaizdų kokybės (rezoliucijos bei objektų matomumo) priklauso ir daugelio algoritmų parametrų pasirinkimas.

Siūlomo metodo grąžinami rezultatai labai priklauso nuo eismo tankumo. 5.4 lentelėje yra pateikiami metodo taikymo rezultatai skirtingo eismo tankumo vaizdų sekoms (5.24-5.26 pav.):

vaizdų sekos failo pavadinimas	judesio slenkstis	Canny parametrai (apatinis sl./viršutinis sl./Sobel apert. dydis)	Hough parametrai (minimalus segmento ilgis, maksimalus tarpas)	Sekimo tikslumas (%)	Atpažinimo tikslumas (%)
street1.avi (eismas ne tankus – žr. 5.24 pav.)	50	50/250/3	5/5	87,5	62,5
street1.avi (eismas ne tankus – žr. 5.24 pav.)	90	50/250/3	5/5	100	62,5
street2.avi (eismas tankus – žr. 5.25 pav.)	50	50/250/3	5/5	~65	~35,5
street3.avi (eismas tankus – žr. 5.26 pav.)	50	50/250/3	5/5	~60,5	~27

lentelė 5.4



pav. 5.24



pav. 5.25



pav. 5.26

6 Literatūros sąrašas

- [1] „*A robust video object segmentation scheme with prestored background information*” (Jinhui Pan, Chia-Wen Lin, Chuang Gu, and Ming-Ting Sun)
- [2] „*Traffic parameter Extraction using Video-based Vehicle Tracking*” (Young-Kee Jung, Yo-Sung Ho)
- [3] „*Vehicle Recognition and Tracking from Road Image Sequences*” (Gian Luca Foresti, Member, IEEE, Vittorio Murino, Member, IEEE, and Carlo Regazzoni, Member, IEEE)
- [4] „*An Introduction to the Kalman Filter*” (Gary Bishop, Greg Welch)
- [5] „*Visual Vehicle Tracking Using An Improved EKF**” (Jianguang Lou, Hao Yang, Weiming Hu, Tieniu Tan)
- [6] „*Robust Multiple Car Tracking with Occlusion Reasoning*” (Dieter Koller, Joseph Weber, and Jitendra Malik)
- [7] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/ISARD1/condensation.html
- [8] <http://www.dcs.gla.ac.uk/~jhw/cond.html>
- [9] „*Vehicle detection and classification in video sequences*” (Andreas Böckert)
- [10] „*Vehicle Detection and Tracking Using the Block Matching Algorithm*” (Luigi Di Stefano, Enrico Viarani)
- [11] „*Algorithms for Vehicle Classification*” (Surendra Gupte, Nikos Papanikolopoulos)
- [12] „*A Fragment-Based Approach to Object Representation and Classification*” (Shimon Ullman, Erez Sali, Michel Vidal-Naquet)
- [13] „*Pictorial Structures for Object Recognition*” (Pedro F. Felzenszwalb, Daniel P. Huttenlocher)
- [14] „*Extending Pictorial Structures for Object Recognition*” (M. Pawan Kumar P.H.S. Torr A. Zisserman)
- [15] „*Vehicle Classification System with Local-Feature Based Algorithm Using CG Model Images*” (Tatsuya Yoshida, Shirmila Mohottala, Masataka Kagesawa, Katsushi Ikeuchi)
- [16] „*Moving object recognition and classification based on recursive shape parameter estimation*” (Dieter Koller)
- [17] „*Three-dimensional object recognition based on the combination of views*” (Shimon Ullman)
- [18] „*Determining Optical Flow*” (Berthold K.P. Horn and Brian G. Schunck)
- [19] „*A Computational Approach to Edge Detection*” (J.Canny)
- [20] „*Progressive Probabilistic Hough Transform*” (J.Matas, C.Galambos, J.Kittler)

- [21] „*Introductory Techniques for 3-D Computer Vision*“ (Emanuele Trucco, Alessandro Verri)
- [22] „*Snakes: Active Contour Models*“ (M. Kass, A. Witkin, and D. Terzopoulos)
- [23] „*Feature Extraction from Faces Using Deformable Templates in CVPR*“ (A.Y.Yuille, D.S.Cohen, and P.W.Hallinan)
- [24] „*A Fast Algorithm for Active Contours and Curvature Estimation*“ (D. J. Williams and M. Shah)

7 Priedai

7.1 Priedas A

Darbe nagrinėjama sistema yra realizuota Visual C++ kalba. Dėl didelių programos kodų apimties, šiame priede yra pateikiami tik sistemos pagrindinių klasių antraščių (angl. headers) failai.

```
////////////////////////////////////
////////////////////////////////////---MyVision.h---////////////////////////////////////
////////////////////////////////////

#include "cv.h"
#include "highgui.h"
#include "MyTime.h"          // Added by ClassView
#include "StructureModel.h"  // Added by ClassView
//#include "vtcsView.h"
#include "MyStructures.h"
////////////////////////////////////

class CMyVision
{
public:
    CvPoint getCirclePoint(int xcenter, int ycenter, float angleDegrees, int dist);
    void getSnakeContours(IplImage * src, IplImage *dst);
    void getHoughLineSegments(IplImage *src, IplImage *dst, int th);
    void getHoughLines(IplImage * src, int max_lines);
    IplImage* morphViews(IplImage* img1, IplImage* img2, float alpha);
    void findAndMatchContours(IplImage * img1, IplImage * img2);
    // CStructureModel model;
    void getMovingObjectEdge(IplImage *src, IplImage *dst, int nMovingObjectIndex, IplImage
*motionImg);
    bool doClassify(IplImage* img, IplImage* dst1, IplImage *motionImg);
    void showCentroid(IplImage *img, int k, IplImage *originalImg);
    void showMovingObjects(IplImage *img, IplImage *originalImg, IplImage *motionImg);
    void setCameraParams(double Tx, double Ty, double Tz, IplImage *img);
    void updateOptions(int classifierID, CString fileName, BOOL bOutputAVI, BOOL bShowMotion, BOOL
bShowEdge, BOOL bShowCalc, BOOL bTrain, BOOL bStop, BOOL bClassify, BOOL bInterpolate);
    void attach(CView * pView, CListCtrl * pListOutput, CString * pEditOutput);
    void startClassify();
    void my_cvSetPixel(IplImage* img, int x, int y, int red, int green, int blue);
    CMyVision();
    CMyVision(CView * pView, CListCtrl * pListOutput, CString * pEditOutput);
    int start();
```

```

virtual ~CMyVision();
struct movingObject
{
    CvPoint vertex1;
    CvPoint vertex2;
    bool bFound;
    int nIdle;
    int nAge;
    int nID;
    bool bTrained;
    bool bSeen;
    CString timeFirstSeen;
    CString timeLastSeen;
    double distanceFirstSeen;
    double distanceLastSeen;
    int frameFirstSeen;
    int frameLastSeen;
    int nVotes;
    //centroid
    CvPoint centroid;
    bool bCentroidClassified;
    // 2005.02.25
    IplImage* contourImage;
    CStructureModel model;
    double averageSpeed;
    CvPoint point2DFirstSeen;
    CvKalman* KalmanTracker;
    CvPoint lastPoint;
    CvPoint lastSize;
} movingObjects[MAX_MOVING_OBJECTS];
struct optionsStruct
{
    CString fileName;
    BOOL bOutputAVI;
    BOOL bShowMotion;
    BOOL bShowEdge;
    BOOL bShowCalc;
    BOOL bTrain;
    BOOL bStop;
    BOOL bClassify;
    BOOL bInterpolate;
    int classifierID;
}

```

```

    } options;

    struct pointStruct
    {
        int x;
        int y;
    };

    CArray <pointStruct,pointStruct> modelsArray[MAX_MODELS];
    int modelsTypesIDArray[MAX_MODELS];
    pointStruct minXY[MAX_MODELS];
    pointStruct maxXY[MAX_MODELS];
    struct cameraStruct
    {
        double Tx3D_m;
        double Ty3D_m;
        double Tz3D_m;
        double metersPerPix;
        double scaleRatio;
        double Cx2D_pix;
        double Cy2D_pix;
        double focalLength;
    } camera;
    bool bCameraParamsNotSet;
    double cameraT[3];
private:
    IplImage * histogram(IplImage* src);
    void updateResEdit(CString messageBuf);
    CvCapture* capture;
    int frameIndex;
    void updateResList(int nID, int arrayIndex, double distanceFromCamera, double speed, double
speedByFrames);
    CMyTime m_myTime;
    bool loadModels(CString fileName, int modelTypeID);
    CView * pMainView;
    CListCtrl * pMainListOutput;
    CString * pMainEditOutput;
    int nTotalMovingObjects;
    int nTotalIDs;
    int nTotalStoredModels;
    void update_mhi( IplImage * img, IplImage * dst, int diff_threshold, IplImage * dst1);
    void releaseMovingObject(int index);
protected:
    void findRegion(int x, int y, byte currentRegion, IplImage *image, IplImage *dstImage);

```

```

void segmentEdge(IplImage * image, IplImage * dstImage);
void interpolate(IplImage * image);
};

/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////---StructureModel.h---/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////

#include "cv.h"
#include "highgui.h"
#include "StructureModelOpenGL.h" // Added by ClassView
#include "MyStructures.h"
#include "StructureModelView.h" // Added by ClassView
class CStructureModel// : public CMyVision
{
private:
public:
    void calibrate();
    void setModelValues(double *valuesArray);
    void drawModelManual(int alphaDegrees, int betaDegrees, int gammaDegrees, double offsetX, double
offsetY, double offsetZ);
    void drawModelOpenGL(int alphaDegrees, int betaDegrees, int gammaDegrees, double offsetX, double
offsetY, double offsetZ);
    structPoint3D points[VEHICLE_POINTS+1];
    CStructureModelOpenGL modelOpenGL;
    IplImage * tmpImage;
    IplImage * imageGray;
    double zBuffer[SIZE*SIZE];
    int polygonIndexBuffer[SIZE*SIZE];
    double getZ(structPlane plane, double x, double y);
    structPolygon polygonsArray[VEHICLE_POLYGONS+1];
    void hideLines();
    structPlane getPlane(structPoint3D p1, structPoint3D p2, structPoint3D p3);//, structPoint3D p4)
    structPoint3D performAllRotations(structPoint3D p);
    structPoint3D performXRotation(structPoint3D point);
    structPoint3D performYRotation(structPoint3D point);
    structPoint3D performZRotation(structPoint3D point);
    void drawAxes(IplImage * image);
    IplImage * image;
    byte connections[VEHICLE_POINTS+1][VEHICLE_POINTS+1];
    double size;
    void drawModel(int alphaDegrees, int betaDegrees, int gammaDegrees, double offsetX, double offsetY,
double offsetZ);

```

```

    structPoint3D getImagePlanePoint(int pointNum);
    structPoint3D getCameraSpacePoint(int pointNum);
    void start(int nModel);
    double focalLength;
    double xCenter;
    double yCenter;
    structPoint3D getObjectSpacePoint(int pointNum);
    double values[12 + 6 + 1];
    double variances[12 + 1];
    CStructureModel();
    virtual ~CStructureModel();
private:
    int red[VEHICLE_POINTS+1];
    int green[VEHICLE_POINTS+1];
    int blue[VEHICLE_POINTS+1];
};

/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////---MyStructures.h---/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////

struct structPoint3D
{
    double x;
    double y;
    double z;
};

struct structPolygon
{
    int pointNum1;
    int pointNum2;
    int pointNum3;
    int pointNum4;
    int pointNum5;
    int pointNum6;
};

struct structPlane
{
    double a;
    double b;
    double c;
    double d;
};

```

```

#define MAX_MOVING_OBJECTS 100
#define MAX_IDLE_TO_TRACK 30
#define MIN_AGE_TO_SHOW 10
#define MIN_AGE_TO_LEARN 75
#define TRACK_THRESHOLD 10 // maybe should depend on RATIO (used in camera model) ?????
#define MIN_MOVING_OBJECT_SIZE 10//30
#define EDGE_THRESHOLD 50
#define MOTION_THRESHOLD 50
#define MIN_DISTANCE_TO_LEARN 35
#define MOTION_SMALL_COMPONENT_THRESHOLD 30
#define MAX_MODELS 100
#define MIN_DISTANCE_TO_CLASSIFY 70
#define MIN_DISTANCE_TO_CALC_SPEED 100
#define SOME_BIG_VALUE 9999999
#define Y_MARGIN_TOP 0//10//50 // in pixels
#define Y_MARGIN_BOTTOM 0//20//60 // in pixels
// #define X_MARGIN 30 // in pixels // now is not used
#define ONSW 100//128//32 // OBJECTS_NORMALIZED_SIZE_WIDTH
#define ONSH 100//128//32 // OBJECTS_NORMALIZED_SIZE_HEIGHT
// STRUCTURE MODEL
// values array indexes
// translation
#define TX 13
#define TY 14
#define TZ 15
// rotation
#define ALPHA 16
#define BETA 17
#define GAMMA 18
#define VEHICLE_POINTS 16
#define VEHICLE_POLYGONS 10+2
#define SIZE 300
#define FL 20
#define MAX_MIN -1000

class CMyStructures
{
public:
    CMyStructures();
    virtual ~CMyStructures();
};

```