VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Akselerometro panaudojimas kuriant intelektualizuotas transportines sistemas

Atliko: 2 kurso, 2 grupės studentė
Ana Ščerba
2 kurso, 2 grupės studentas
Rolandas Tamašauskas

Darbo vadovas:
Dr. lekt. Alminas Čivilis

Vilnius
2011

Turinys

Anotacija

Šio darbo tikslas – sukurti efektyvius įvykių atpažinimo filtrus duomenims, gautiems pasitelkiant pagreičių matavimo prietaisus mobiliosiose technologijose. Atlikus sukurtų filtrų nuodugnius tyrimus įvertinti, kurie iš filtrų įvykius atpažįsta efektyviausiai. Neuroninių tinklų pagalba išskirti duobę ir greičio ribojimo kalnelį iš bendro įvykių srauto. Sukurtą prototipą realizuoti praktiškai ir išsamiai atvaizduoti gaunamus rezultatus. Taip pat susipažinti su kitų autorių sukurtais atpažinimo modeliais bei sistemomis, palyginti jų sukurtų algoritmų bei šio darbo atpažinimo rezultatus, praktiškai realizuoti išnagrinėtą modelį. Užfiksuotoms eismo įvykių koordinatėms pritaikyti tinkamiausią duomenų grupavimo algoritmą ir išsamiai atvaizduoti gautus rezultatus.

Summary

Accelerometer data analysis and usage for detecting traffic alteration

The aim of this work was to analyze the data which was gotten from accelerometer mounted in mobile device during the test drives through the city together with GPS (Global Positioning System) coordinates, to detect and report the surface conditions of roads as well as to find the way, how it could be represented in the map. The research was started by analyzing oscillation data from accelerometer. We had to keep in mind that there can be road bumps, pit holes, speed bumps and other road anomalies, car can accelerate quickly and break sharply or even crash into something, what would cause a sudden stop. In order to recognize events, different detection filters were applied on data. In addition to this, neuron network was used to recognize pit holes and speed bumps from all event flow. The results of event detection algorithms were compared with other scientist's works. In order to represent results clearly, database was created holding coordinates of the road events and other information like time, etc. The results were represented using an application programming interface made-up by Google, which was really suitable solution in our case. The whole system was programmed using Java servlets, which allowed to gather data from database using SQL (Structured Query Language) queries. While trying to represent accelerometer data, we faced difficulties in representing these road events on the map, as GPS each time returned answer with small variation of coordinates. In this case, we had to find a way, how to represent data clearly without overloading the map with information. After analyzing algorithms for objects clustering, the decision was made to use K-means algorithm for this objective as the most efficient and precise. In our research we

proved that accelerometer could really help to monitor traffic alteration by evaluating our algorithms on data and show it can successfully detect these traffic anomalies as well as reduce suspicious detections with the help of clustering mechanism.

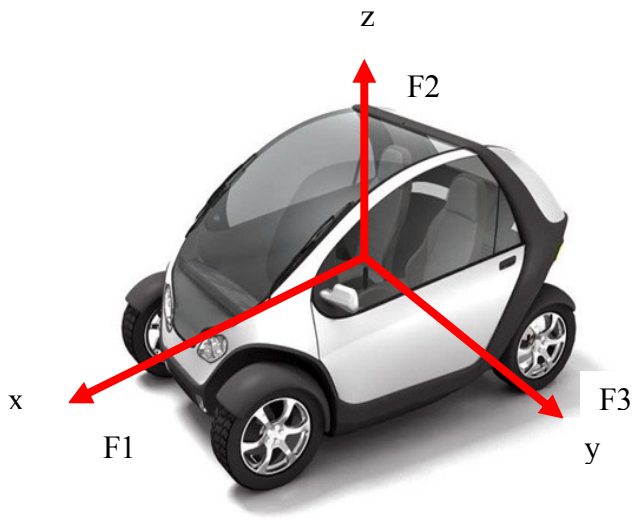
Įvadas

Kelių eismo autoįvykiai yra laikomi viena iš didžiausių municipalinių problemų visame pasaulyje. Eismas yra sąveika tarp žmonių elgesio, mašinų ir kelių infrastruktūros. Kasdien susiduriame su nesaugia aplinka mūsų keliuose. Siekiant apsaugoti žmonių sveikatą ir gyvybę keliuose yra diegiama daug įvairių priemonių į automobilius, tokių kaip: oro pagalvės, navigacinė sistema, saugos diržai ir begalė kitų. Pasinaudojus papildomais prietaisais, pavyzdžiui, akselerometru, galima nustatyti transporto priemonės būseną, būtent todėl šio prietaiso pagalba naudojasi mašinų gamintojai, kad, pavyzdžiui, nustatytų, kokia jėga turi atsitrenkti mašina, kad išsiskleistų oro pagalvė. Šio darbo tikslas yra išanalizuoti akselerometro, įmontuoto mobiliajame įrenginyje, parodymus ir juose atpažinti duobes ir greičio ribojimo kalnelius, užregistruotus transporto priemonei judant keliu, bei atvaizduoti kelyje gautas įvykio koordinates. Sukūrus prototipą įvykių atpažinimui ir jį įgyvendinus būtų galimybė lengvai ir greitai analizuoti kelių būklę, bei atpažinti pavojingus kelių ruožus, taip išvengiant nelaimių. Siekiant išsamiai įsigilinti į problemą buvo nagrinėti kitų autorių panašūs darbai. Daugiausia dėmesio buvo skirta duomenų analizavimui, nes įvairios nelaimės keliuose pateikia skirtingus rezultatus. Akselerometro duomenys yra fiksuojami x, y, z ašių kryptimis. Kiekviena ašis atspindi skirtingus įvykius. Jiems atpažinti buvo analizuojami įvairūs filtrai: z-piko, maksimumo ir vidurkio santykio, maksimalių reikšmių. Atlikus šių filtrų nuodugnius tyrimus įvertinti, kurie iš jų įvykius atpažįsta efektyviausiai. Siekiant išskirti koks tai buvo įvykis, duobė ar greičio ribojimo kalnelis, iš bendro įvykių srauto, buvo pasinaudota apmokytais neuroniniais tinklais. Taip pat buvo palyginti kitų autorių darbai šia tema ir nustatytas geriausiai įvykių radimui tinkantis prototipas. Įvykiai buvo patalpinti į sukurtą duomenų bazę. Pasinaudojus duomenų bazėje esančiais duomenimis bei Google API akselerometro parodymai buvo atvaizduoti žemėlapyje. To pasekoje, buvo sukurtas žemėlapis, kuriame yra atvaizduoti pavojingi kelių ruožai realiu laiku. Taip pat buvo pritaikytas grupavimo algoritmas, siekiant, kad besikartojantys pranešimai apie tą patį įvykį nebūtų atvaizduoti po kelis kartus toje pačioje vietoje. Toks žemėlapis padės padidinti vairuotojų atidumą važiuojant pavojingu kelio ruožu, bei sumažinti nelaimingų atsitikimų skaičių.

1. Problema

1.1. Problemos formulavimas

Kelių ir eismo būklės stebėjimas susilaukia vis daugiau dėmesio. Eismo situacijų bei kelių būklės pokyčiams nustatyti reikalingas prietaisas, fiksuojantis transporto priemonės būseną arba ją veikiančias jėgas tam tikru laiko momentu. Buvo nuspręsta fiksuoti jėgas, kurios veikia transporto priemonę, jai atsidūrus ekstremalioje eismo situacijoje. Darbe buvo panaudotas akselerometras, nes būtent šis prietaisas labiausiai tinka tokio pobūdžio matavimams, t.y. matuoti išorines jėgas, veikiančias objektą įvairiomis kryptimis. (1 pav.)



1 pav. Kryptys, kuriomis matuojamos jėgos.

1.2. Susiję darbai

Šių dienų pasaulyje yra kuriamos transporto sistemos skirtos įvairiems tikslams: maršruto planavimui, eismo srautų valdymui, skubios pagalbos greitam atvykimui ir t.t. Mokslo bendruomenė atlieka daug tyrimų, susijusių su eismo pokyčių stebėsena. Dauguma šių darbų grindžiami iš GPS įrenginių gaunama informacija, tačiau yra atliekami ir kitokie tyrimai, paremti akselerometro parodymais. Akselerometrai yra plačiai naudojami atliekant tyrimus ir nustatant, pavyzdžiui, kada turėtų išsiskleisti automobilio oro pagalvė avarijos metu. Vis dėl to, akselerometras gali būti

naudojamas ne tik tokiems tikslams. Jį galima naudoti ir kelio duobių bei transporto priemonės stabdymo atpažinimui.

Straipsnyje "Nericell: Kelių ir eismo sąlygų stebėsena naudojant Mobile Smartphones" (Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones) autoriai tiria kelių būsenų bei eismo sąlygų nustatymo problemą. Jie teigia, kad sprendžiant problemą reikia įdėti daug pastangų, bet tuo tarpu ji yra aktuali ir sprendimas atneš naujų patogumų eiliniam vartotojui. Savo darbo tikslui atlikti jie pasirinko Nericell - sistema, kuri atlieka eismo sąlygų stebėseną pasitelkiant mobiliuosius telefonus. Nericell pasinaudoja mobiliaisiais telefonais atliekant pokyčių sekimą bei perduodant duomenis į serverį tolimesnei analizei. Autoriai teigia, kad mobilieji telefonai yra pajėgus atlikti Nericell užsibrėžtus tikslus, nes juose įmontuota įvairių dalykų, tokių kaip mikrofonas, vaizdo kamera, GPS, akselerometras. Kiekvienas iš paminėtų dalykų gali būti naudojamas eismo pokyčiams nustatyti. Savo darbe autoriai didelį dėmesį skyrė duomenų koregavimui, pasinaudojant ašių akselerometro pakreipimui išilgai kanoninių ašių, bei duobių ir stabdymo žymių radimui metodų kūrimui. Taip pat jie pristatė metodą, kaip atskirti garso signalą nuo kitų garsų, įrašytų telefono atmintyje.

Atlikdami savo tyrimą, autoriai rinko duomenis važinėdami automobiliu tuo pačiu maršrutu ilgą laiką. Norint gauti kuo įvairesnių rezultatų buvo nuspręsta transporto priemonę vairuoti skirtingiems asmenims.

Autorių teigimu didžiausia problema analizuojant duomenis yra jų netikslumas. Kadangi autoriai duomenis gauna pasinaudojant mobiliaisiais telefonais, tai įmontuotas akselerometras gali būti bet kokioje padėtyje ir duomenys gali būti gauti su didele paklaida. Šiai problemai spręsti autoriai pasitelkė algoritmą, kuris gali jau gautus duomenys pakeisti tikslesniais. Jų pristatomas algoritmas buvo paremtas ašių krypčių pasukimu tokiu būdu, kad ašys sutaptu su transporto priemonės ašių sistema.

Turint tikslius arba jau pakoreguotus duomenis autoriai straipsnyje apibrėžia stabdymo bei duobių radimo algoritmus. Stabdymą galima aptikti pagal a_x akselerometro ašį. Norint aptikti stabdymą autoriai apskaičiuoja duomenų vidurkį, gautų a_x ašimi, per kitimo laiką. Jei vidurkis yra mažesnis už apskaičiuota ribinę reikšmę T (kuri stabdymo atveju lygi 0.11g), tai tada galima teigti, kad įvyko stabdymas. Duobių radimas yra sudėtingesnis, nes jis trunka tik apie sekundę. Autoriai remiasi panašiu būdu kaip ir stabdymo atvejo radimui. Jie apsibrėžia iš anksto ribinę reikšmę T, kuri priklauso nuo transporto priemonės greičio. Jei greitis viršija 25 km/h ir a_z ašies reikšmės yra didesnės už ribinę reikšmę T, tai yra registruojama duobė. Tuo tarpu, jei greitis yra mažesnis nei 25

km/h, tai ieškoma tokių a_z reikšmių, kurios būtų mažesnės už ribinę reikšmę T, bet palaikytų kitimą bent 20 ms.

Kitame straipsnyje [EGH08] autoriai tiria kelių būklę, tam pasitelkiant mobiliuosius sensorius. Pristatoma visa sistema ir jai pritaikyti algoritmai, skirti stebėti šiai svarbiai civilinei infrastruktūrai naudojant automobilius, aprūpintus šiais mobiliais sensoriais. Automobilyje įmontuoto akselerometro, bei gps imtuvo duomenys kartu su greičio parodymais yra perduodami bevieliu tinklu wifi arba GSM tinklo pagalba į pagrindinį serverį tolimesniam apdorojimui. Serverio duomenų bazėje yra laikomi visi duomenys, gaunami iš skirtingų automobilių tokiu formatu <laikas, vieta, greitis, kryptis, akseleracija>. Norint kad ši mobili duobių atikimo sistema veiktu sėkmingai straipsnio autoriams reikėjo išspręsti keletą problemų. Visų pirma yra labai daug įvykių, įtakojančių akselerometro parodymus, nesusijusių su duobėmis, tai staigus stabdymas, automobilio durų trankymas, staugus pasukimas ir t.t. Taip pat keliuose pilna įvairių asfalto sujungimų, geležinkelio pervažų, kurias atpažinti nuo realių duobių yra dar sudėtingiau. Todėl visiems šiems įvykiams išskirti reikia detalaus eksperimentinio tyrimo. Tačiau svarbiausia problema, su kuria susidūrė straipsnio autoriai buvo ta, kad automobiliams, važiuojantiems per ta pačia duobę, duomenys gali smarkiai skirtis priklausomai nuo judėjimo greičio, kaip automobilis priartėjo prie kliūties ir kokiose vietose pritvirtinti pagreičio sensoriai.

Šioms problemoms spręsti, autoriai visų pirma, eksperimentiniu būdu rinko duomenys, važiuodami per skirtingo tipo kelio kliūtis ir taip bandydami išskirti atskirus kliūčių šablonus. Sensorius buvo mokomas atpažinti įvykį pagal X ir Z akselerometro ašių parodymus, esamą automobilio greitį, bei šiuos duomenis skirstyti į grupes: lygus kelias, perėja arba kelio sujungimas, pervaža, duobė, šulinio dangtis, staigus sustojimas, posūkis. Galiausiai visiems duomenims, gautiems iš visų automobilių pritaikytas grupavimo algoritmas, pranešant tik apie įvykius keliuose, pasikartojusius daugiau nei keletą kartų. Įvykiai sugrupuojami į vieną, jeigu atstumas tarp jų mažesnis nei iš anksto pasirinktas. Leistina paklaida šiuo atveju buvo 3,3m.

Šio straipsnio autoriai teigia, kad jų pritaikytas algoritmas puikiai tinka kelių būklės stebėjimui. Pritaikius įvykių atpažinimo filtrus teisingų įvykių atpažinimas sudarė 99%.

Straipsnio [RMD01] autoriai supažindina su kitokia sistemos panaudojimo idėja. T.y. sistema naudojama tiltų išsikraipymams, bei vibracijoms aptikti ir jas stebėti taip užkertant kelią skaudžioms pasekmėms.

Integruojant GPS kartu su triašiu akselerometru viena iš akselerometro ašių turėtų sutapti su tilto pagrindine (išilgąja ašimi). Tuomet visi matavimų duomenys transformuojami į vienodą koordinačių

sistemą. Transformuotos koordinatės suprojektuojamos į tilto ašių sistemą pasinaudojant posūkio matrica.

Šie apskaičiuoti parametrai vėliau naudojami akimirksnio pagreičiams aptikti trimatėje tilto ašių sistemoje.

Straipsnio autoriai atliko eksperimentą, matuodami 60 metrų ilgio pėsčiųjų tilto svyravimus. Duomenys renkami keturiais GPS imtuvais, bei akselerometrais pritvirtintais skirtingose tilto pusėse, bei apdorojami programine įranga, sukurta C++.

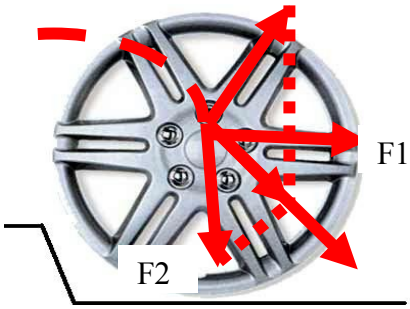
Šiame darbe pristatytas būdas koordinačių transformacijai, bei duomenų apdorojimui įgyvendinti. Tokiu būdu galima stebėti tiltų poslinkius, bei vibracijas.

1.3. Teorinis pagrindimas

Akselerometras suteikia galimybę matuoti jėgas keliomis kryptimis, pavyzdžiui x, y ir z ašimis. Transporto priemonei judant tam tikra kryptimi, akselerometras fiksuos švelnius virpesius, kylančius dėl automobilio vibravimo, nedidelių kelio nelygumų, bet automobiliui patiriant didesnes apkrovas, pavyzdžiui, įvažiuojant dideliu greičiu į posūkį, staigiai stabdant ar įsibėgėjant, akselerometro parodymai svyruos žymiai aukštesnėje skalėje dėl staiga kintančio pagreičio. Šiuos fizikinius procesus būtent ir apibrėžia Niutono mechanikos dėsniai. Jie apibendrina inercijos principą, kurio esmė yra tokia: bet kuris judantis kūnas stengiasi išlaikyti savo judėjimą, o bet kuris parimęs kūnas – rimties būseną. Kitais žodžiais tariant, kūnas išlaiko rimties būseną arba juda pastoviu greičiu tol, kol jį paveikia kuri nors jėga. Taip pat, reikėtų turėti omenyje tai, kad kuo stipresnė jėga veikia kūną, tuo labiau kinta kūno greitis. Gali kisti ne tik greičio modulis, bet ir kryptis; greičio pokytis per laiko vienetą ir bus pagreitis. Taigi juo stipresnė veikianti jėga, tuo didesnis kūno pagreitis. Be to, prisiminkime, kad antrasis Niutono mechanikos judėjimo dėsnis tvirtina, kad pagreitis, kuriuo juda kūnas, yra tiesiog proporcingas kūną veikiančiai jėgai ir atvirkščiai proporcingas to kūno masei, $F = ma$. Remdamiesi šiais dėsniais galime suprasti, kaip kūnai veikia vienas kitą. Jei automobilis stovi ant kelio, tai kelias veikia jį priešingos sunkiui (t. y. automobilio svoriui) krypties ir tokio pat dydžio jėga. Šis dėsnis apibendrina: sąveikaujant kūnams, kiekvieną veikiančią jėgą atitinka to paties didumo, bet priešingos krypties atoveikio (reakcijos) jėga. Toliau yra pateikta keletas teorinių pavyzdžių, kurie turėtų išsamiau viską paaiškinti.

Tarkime automobiliui važiuojant lygiu keliu, jis dideliu greičiu įvažiuoja i duobę (2 pav.). Tokiu atveju, automobilį veikianti jėga F1 pakis nežymiai. Tuo tarpu jėga F2, veikiama sunkio

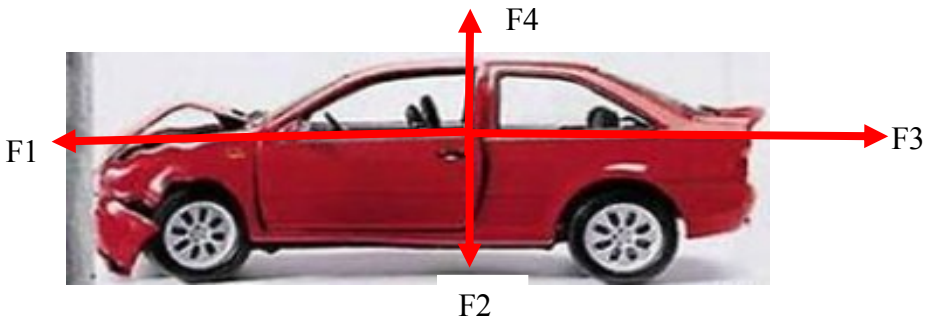
jėgos, staigiai padidės, įtakodama akselerometro parodymus z ašimi. Taigi, pagreitis matuojantis z ašimi sieks 0,5 g ir daugiau priklausomai nuo duobės tipo ir gylio. Remdamiesi tokiais prietaiso parodymais, galėsime daryti prielaidą, kad automobilis susidūrė su kelio nelygumais. Taip pat, reiktų atsižvelgti į tai, kad ratas krisdamas į duobę, jos dugną pasiekia ne iš karto, o veikiamas inercijos, priešinasi greičio kitimui judėdamas lanko trajektorija.



2 pav. Jėgos, veikiančios mašiną įvažiuojant į duobę

Šiuo atveju, kol automobilis važiuoja beveik tiesiu keliu pagreitis kinta nežymiai, bet kai tik automobilis įvažiuoja į duobę pagreitis staigiai padidėja. Tai paaiškinama tuo, kad kinta krentančio į duobę automobilio veikiančios jėgos. Padidėja traukimo jėga, nes atsiranda sunkio ir atramos reakcijos jėgų sudaromoji. Automobiliui išvažiuojant iš duobės atsiranda priešingos nei traukimo jėgos krypties sunkio ir atramos reakcijos jėgų sudaromoji. Todėl grafike pagreitis sumažėja, o po to stabilizuojasi [SW01].

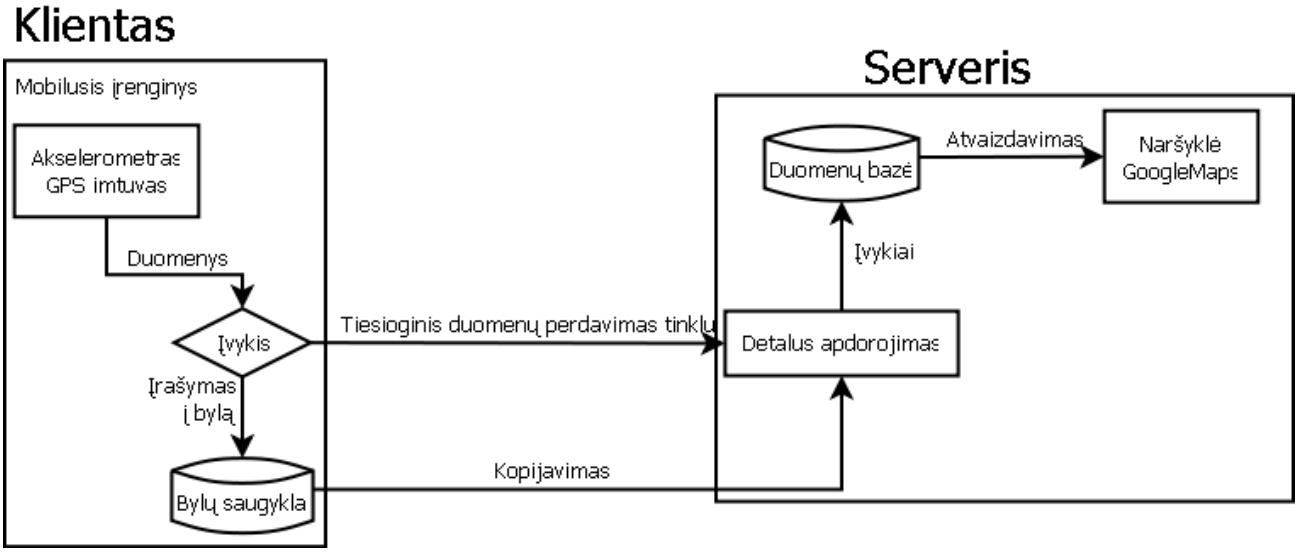
Galima sumodeliuoti kitokią situaciją. Tarkime, transporto priemonė judėdama lygiu keliu netikėtai susiduria su priešpriešine kliūtimi ir įvyksta staigus sustojimas (3 pav.). Iš Niutono mechanikos dėsnių yra žinoma, kad kiekvienas judantis kūnas, šiuo atveju tai automobilis, stengiasi išlaikyti savo judėjimą tol, kol jį paveikia kuri nors jėga. Todėl transporto priemonėje akselerometro parodymai x ašimi, įtakoti milžiniškos inercijos, žymiai išaugs. Tai leis manyti, jog įvyko rimtas auto įvykis, kurio metu gal būt net neišvengta aukų.



3 pav. Jėgos, veikiančios mašiną susidūrus su kliūtimi

Remiantis pateiktu teoriniu pavyzdžiu, akselerometras bus panaudotas pateiktiems įvykiams atpažinti. Prietaisas įmontuotas į automobilį ant tvirto, neamortizuojančio paviršiaus. Važiuojant automobilis įveikia įvairias kliūtis, dėl to prietaiso parodymai bus skirtingi. Duomenys yra fiksuojami bei analizuojami.

Diagramoje (4 pav.) yra pateikti kliento bei serverio pusėje atliekami veiksmai. Kaip jau buvo minėta, duomenys yra renkami mobiliajame įrenginyje įmontuoto akselerometro pagalba. Mobiliajame įrenginyje sukurta aplikacija atlieka pirminę duomenų analizę ieškodama bendrame duomenų sraute išsišokėlių. Aptikus numanomą įvykį, jo laikas, tuo metu veikusios jėgos ir GPS koordinatės, pasirinktinai, yra įrašomi į failą, kurį vėliau reikės importuoti arba perduoti GSM tinklu tiesiai į serverį tolimesniam įvykio atpažinimui ir užfiksavimui duomenų bazėje. Serverio pusėje duomenys yra analizuojami, t.y. apmokytais neuroniais tinklais įvykiai yra atpažįstami. Pasikartojančių įvykių koordinačių sutraukimui yra panaudotas K-means algoritmas. Sugrupuotus įvykius galima peržvelgti naršyklės pagalba GoogleMaps žemėlapyje.

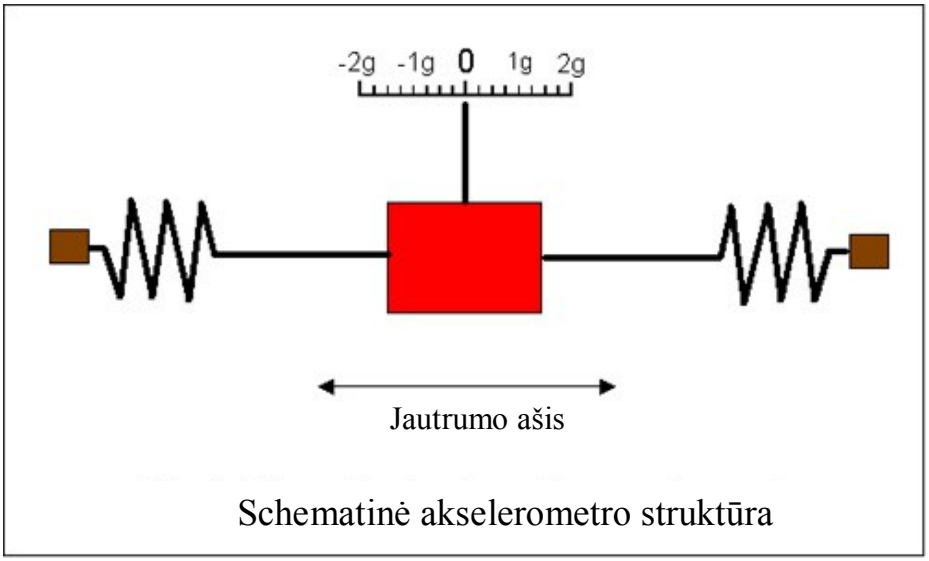


4 pav. Veiksmų diagrama

1.4. Akselerometras

Akselerometras – prietaisas, skirtas pagreičiui matuoti (5 pav.). Kai fizikinis kūnas įgyja pagreitį tam tikra ašimi, jis tampa priklausomas nuo jėgos, lygios $F = ma$. Taigi, akselerometrai

yra sukonstruoti tokiu principu, kad būtų matuojama žinomos masės kūno jį veikianti jėga tam tikra ašimi.



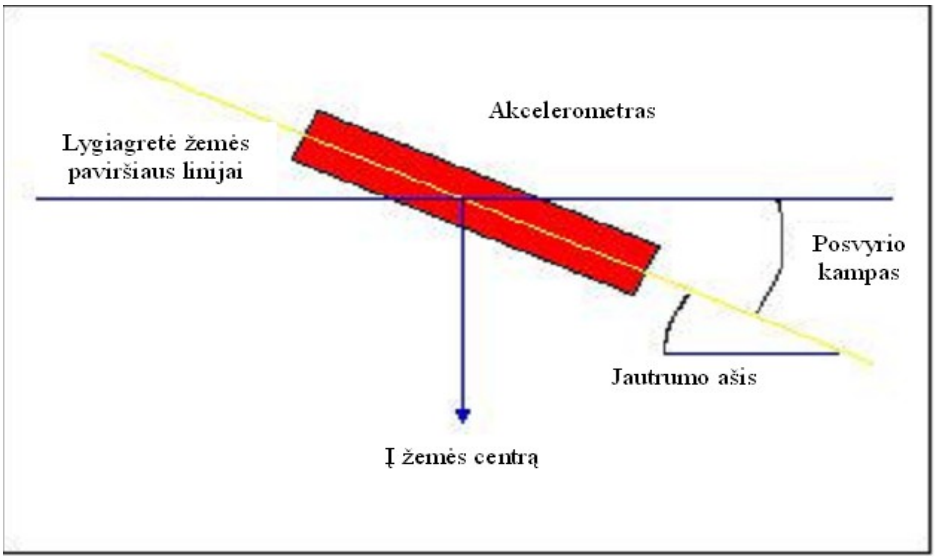
5 pav. Schematinė akselerometro struktūra

1.4.1. Akselerometro veikimas ir taikymas praktikoje

Šiuo metu pasaulyje yra išrasta įvairaus veikimo principo akselerometrų. Vieni iš šių prietaisų naudoja pjezoelektrinį efektą. Jie savyje turi mikroskopines kristalų struktūras, kurios įsitempia veikiamos pagreičio ir tai įtakoja įtampos generavimą. Kitas būdas pagreičiui apskaičiuoti yra pasikeitimų stebėjimas talpinėje varžoje. Jei yra dvi mikrostruktūros arti viena kitos, tuomet jos turi tam tikrą varžą tarp savęs. Jei pagreičio sukurta jėga pajudina vieną iš struktūrų, tuomet talpinė varža pasikeis. Pridėkite keletą schemų konvertuoti iš varžos į įtampą ir tokiu būdu jūs gausite akselerometrą. Šio prietaiso veikimo principų yra dar daugiau, kaip antai naudojimasis karšto oro burbulais, šviesa ir tt [Insu05].

Ankstyvaisiais 1990 metais buvo pristatyta nauja įrenginių MEMS (Microelectromechanical system) karta. Juose į vientisą silicio struktūrą jau buvo įmontuotas akselerometras [Gol06].

Taip pat, derėtų paminėti, kad akselerometro pagalba galime rasti ne tik pagreitį. Kadangi žemės traukos jėga veikia visus kūnus, akselerometras gali būti panaudotas posvyriui rasti (6 pav.).



6 pav. Akselerometro panaudojimas posvyrio kampo radimui

Kai jautrumo ašis rodo tiesiai į žemės centrą, akselerometras rodys 1g (darant prielaidą, kad jokio papildomo pagreičio šia kryptimi nėra). Kai prietaiso jautrumo ašis tampa lygiagreti žemės paviršiaus linijai, pagreitis tampa lygus 0. Vadovaujantis šia informacija, galime išvesti formulę, kurios pagalba galėsime apskaičiuoti tikslų posvyrio kampą:

$$Posvyrio\_kampas = \arcsin\left(\frac{ismatuotas\_pagreitis}{g}\right)$$

1.4.2. Akselerometras mūsų darbe

Duomenų rinkimui buvo naudojamas mobilus telefonas Samsung S5230(7 pav.) su jame įmontuotu triašiu akselerometru BOSCH SMB380. Akselerometras mobiliajame įrenginyje yra įmontuotas taip, kad akselerometro x ašis sutampa su su telefono pločiu, y ašis – telefono ilgiu bei z ašis – su telefono aukščiu. Akselerometro parodymams gauti buvo sukurta aplikacija JAVA ME programavimo kalba kuri su akselerometru bendrauja JSR-256 JAVA paketo pagalba. Duomenų rinkimo metu yra gaunama nuo 6 iki 8 pagreičio kitimo duomenų per sekunde trimis

7 pav. Įrenginys

ašimis. Eksperimentinių tyrimų metu buvo įsitikinta, kad tokio dažnio visiškai pakanka įvykių atpažinimui.

Siekiant įsitikinti fiksuojamų duomenų korektiškumu ir pakankamu prietaiso jautrumu staigiems gravitacijos pokyčiams SMB380 akselerometro parodymai buvo lyginami su dviašio analoginio akselerometro ADXL322 parodymais, kurio duomenys svyruodami 1 volto diapazone išeina per stereo 3,5mm kištuką į kompiuterio garso kortą. "Left" interpretuojama kaip X ašis, "Right" – Y. Pilnavertišką analizę analoginio įrenginio pagalba ribojo ašies trūkumas. Taip pat duomenys iš mobiliojo telefono buvo lyginami su kito skaitmeninio triašio akselerometro MMA7456L parodymais, kuris kompiuteriui duomenis perduoda „Bluetooth" technologijos dėka. Pastarasis prietaisas turėjo šiek tiek didesnį pagreičių fiksavimo dažnį, bet nebuvo mobilus įrenginys, o tai ribojo funkcionalumo ir praktinio panaudojimo galimybes.

1.5. Globalaus pozicionavimo sistema

GPS – tai globali pozicijos nustatymo sistema (**G**lobal **P**ositioning **S**ystem). Tai yra didelio tikslumo palydovinė radionavigacinė sistema, suteikianti informaciją apie objektų padėtį erdvėje (3D), jų judėjimo greitį, kryptį ir įveiktą atstumą, atstumus iki pasirinktų taškų, apie tikslų vietos laiką duotu momentu, geografinius duotos vietovės saulėtekio/saulėlydžio laikus, mėnulio fazes. Sistema susideda iš 24 dirbtinių žemės palydovų ir eilės antžeminių kontrolės bei valdymo stočių. Paprastai yra naudojami tik 21 palydovas, o likę 3 yra kaip rezerviniai. Kiekvienas iš šių palydovų griežtai nustatytais dažniais nuolatos siunčia tam tikrus signalus apie savo buvimo vietą erdvėje, tikslų pasaulinį laiką ir savo identifikacinius kodus.

1.5.1. Globalaus pozicionavimo sistemos prietaisas mūsų darbe

Sukurta aplikacija JAVA ME programavimo kalba, kuri paleista mobiliajame įrenginyje, turinčiame GPS imtuvą ir palaikančiame JAVA ME platformą geba gauti esamas vietos koordinates. Sėkmingam aplikacijos veikimui svarbią įtaką turi esamu momentu matomų palydovų skaičius, kurių turi būti mažiausiai trys. Norint užtikrinti stabilų ryšį su palydovais, vertėtų vengti tunelių ir šalia esančių aukštų pastatų.

2. Analizė

2.1. Akselerometro parodymų analizė

Pasinaudojant aukščiau išvardinta įranga buvo atliekamas tyrimas. Važiuojant automobiliu akselerometro, kuris integruotas mobiliajame telefone, pagalba buvo fiksuojami parodymai. Duomenys priklauso nuo to, kokiu keliu važiavo automobilis. Parodymai yra skirtingi, nes automobilio pravažiuotas kelias buvo įvairus: vietomis buvo duobėtas, kitur transporto priemonė susidūrė su kitomis kliūtimis, dėl kurių vairuotojas privalėjo stabdyti. Verta paminėti, kad duomenys gali būti skirtingi ne vien dėl kelio ruožų netolygumo, bet ir dėl skirtingų akselerometrų tipų, naudojamų praktikoje, nes šių prietaisų ne vien parodymų skalės intervalai gali skirtis, bet ir akselerometro jautrumo slenksčiai. Taip pat parodymus veikė skirtingų mašinų pakabos. Gauti duomenys buvo išanalizuoti su tikslu apibrėžti įvairius kelio įvykius. Buvo nagrinėjama, kokio didumo jėga veikė akselerometrą kokia ašimi. Akselerometro parodymai buvo registruojami trimis ašimis: x, y bei z. Yra pastebėta, kad fiksuojant nelygumus kelyje akselerometro duomenis nuo rimties būsenos skyrėsi 1 g vienetu. Taip pat duobėtas kelias pasižymėdavo akselerometro parodymų pokyčiais Z ašimi. Tuo tarpu įsibėgėjus ir staigiai stabdant pagreičio pokyčiai labiausiai buvo juntami Y ašimi.

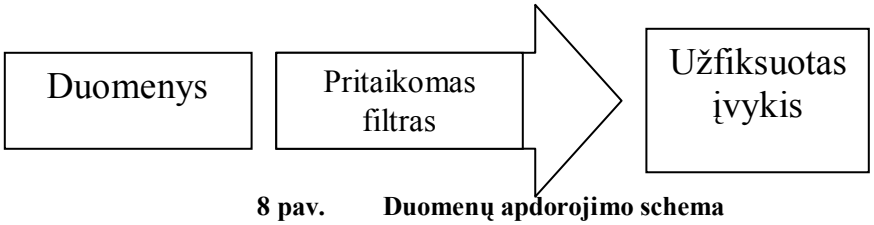
2.2. Įvykio atpažinimas

Kelias labai retai būna visiškai lygus. Važiuojant keliu beveik visą laiką yra jaučiami tam tikri nelygumai. Akselerometras – tai ypač jautrus prietaisas, kuris reaguoja į bet kokias pasikeitusias aplinkybes. Duomenų rinkimo metu buvo pastebėta, kad duomenys svyruodavo nuo 0,5 g vienetų iki 1,3 g vienetų. Jei svyravimai būtų pasiekę -8 g vienetų ar daugiau, tai žmogus būtų praradęs sąmonę ar būtų taip susižalojęs, kad ji ištiktų mirtis [VOS04].

Siekiant tobulinti įvykių atpažinimo algoritmą buvo nutarta papildomai remtis Jakob Eriksson bei kitų autorių straipsnyje[EGH08] aprašytu algoritmu, bei lyginti rezultatus ir efektyvumą. Šio straipsnio autoriai naudojosi kliūčių šablonais, kuriuos jie kūrė eksperimentiniu būdu rinkdami duomenis. Autorių [EGH08] straipsnyje sensorius buvo mokomas atpažinti įvykį pagal X ir Z akselerometro ašių parodymus, esamą automobilio greitį, bei šiuos duomenis skirstyti į grupes:

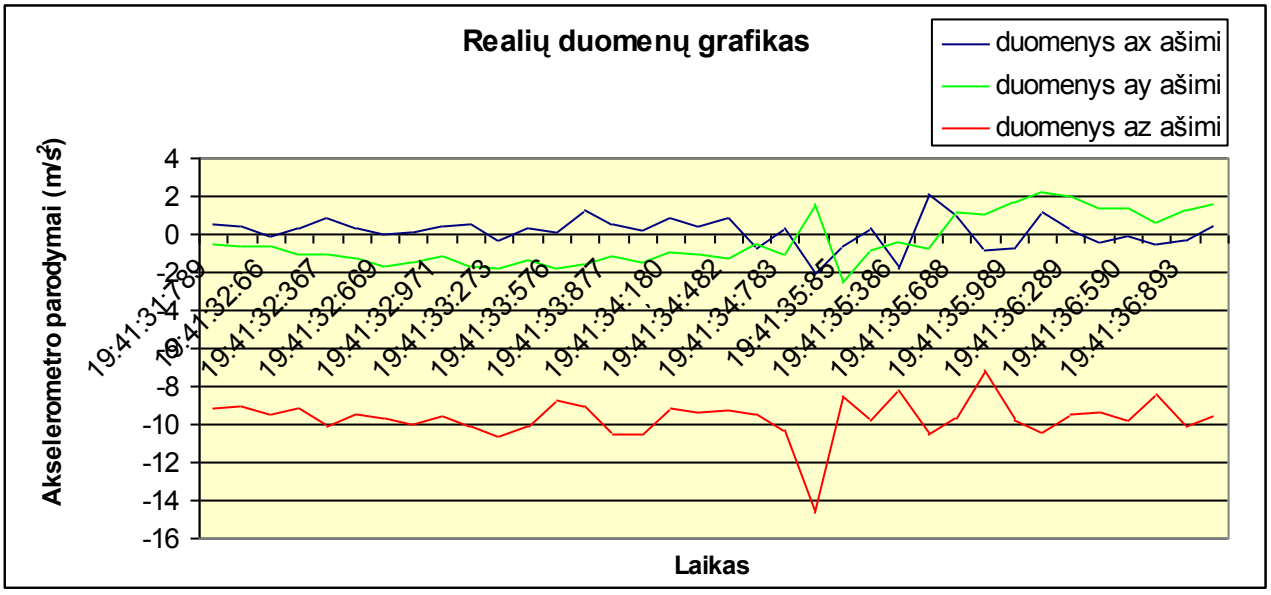
lygus kelias, perėja arba kelio sujungimas, pervaža, duobė, šulinio dangtis, staigus sustojimas, posūkis.

Išanalizavus kitų mokslininkų darbus nuspręsta gautus duomenis apdoroti pasinaudojant įvairiais filtrais (8 pav.) ir tokiu būdu ieškoti įvykio. Žemiau yra pateikti jų veikimo rezultatai.



8 pav. Duomenų apdorojimo schema

Kaip jau buvo minėta, akselerometro duomenys buvo gauti pasinaudojus mobiliajame telefone įmontuotu akselerometru automobiliui važiuojant kelio danga. Grafike (9 pav.) yra pateiktas realių duomenų grafikas automobiliui važiuojant per duobę. Duomenys yra atvaizduoti 3 ašimis, iš kurių kiekviena fiksuoja skirtingas automobilį veikiančias jėgas. Mobiliajame telefone akselerometro mikroschema yra įmontuota taip, kad akselerometro y ašis sutampa su telefono ilgiu, x ašis – su telefono pločiu bei z ašis – su telefono aukščiu. Automobilyje telefonas buvo įrengtas taip, kad akselerometro x ašis – a_x – yra statmena automobilio judėjimo krypčiai, akselerometro y ašis – a_y – sutampa su automobilio judėjimo kryptimi bei akselerometro z ašis – a_z – yra vertikali. Duomenys a_x ašimi parodo transporto priemonės stabdymą/lėtėjimą, parodymai a_y ašimi fiksuoja posūkius/svyravimus į šonus, o duomenys a_z ašimi atspindi kelio nelygumus.



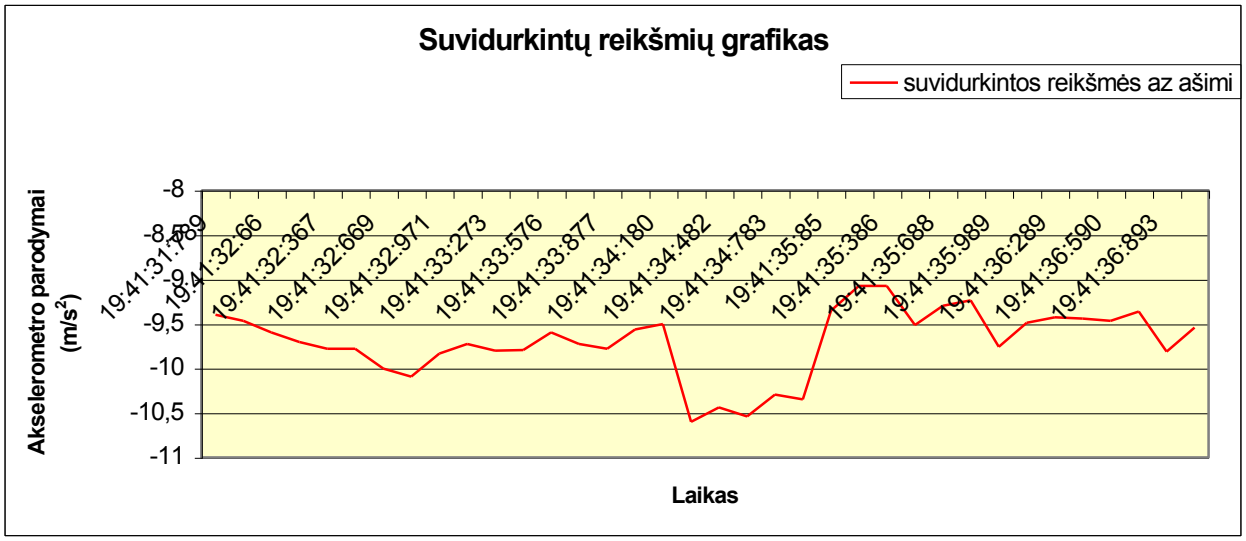
9 pav. Realių duomenų grafikas

Remiantis tuo, kad akselerometras yra įmontuotas mobiliajame telefone ir jo duomenys yra gaunami pasinaudojant JAVA ME sukurta programa, tai dėl mobiliojo telefono resursų trūkumo buvo nuspręsta jame atlikti tik pirminę duomenų analizę. Tuo tarpu išsamesnę akselerometro parodymų analizę atlikti daugiau techninių resursų turinčiame įrenginyje, tokių kaip serveris ar GRID aplinka.

Mobiliajame telefone pirminė duomenų analizė buvo atliekama tam, kad nereikšmingi duomenys neapkrautų serverio. Pradinei analizei buvo renkamas metodas, kuris išryškina nebūdingus akselerometro duomenis važiuojant keliu. Tuo tikslu buvo bandomi algoritmai ir ieškomas toks, kuris efektyviausiai išryškina galimai įvykusį įvykį. Kadangi kelio nelygumai yra fiksuojami akselerometro a_z ašimi, tai tolesnei pirminei analizei buvo nagrinėjami būtent šios ašies duomenys.

Buvo taikomi įvairūs duomenų filtrai norint rasti geriausią sprendimą įvykio fiksavimui.

- Realių duomenų grafikui pritaikomas high-pass filtras – šio filtro reikšmė ta, kad jis vidurkindamas duomenis, panaikina triukšmą, tai yra nesvarbius akselerometro parodymus, kuomet automobilis staigiai stabdo ar atlieka posūkį.



10 pav. Suvidurkintų reikšmių grafikas

Šiame tyrime (10 pav.) akselerometro pagalba gauti duomenys buvo apdorojami taip pat naudojant duomenų vidurkį. Kas 5 duomenų reikšmes buvo skaičiuojamas duomenų vidurkis, t.y.

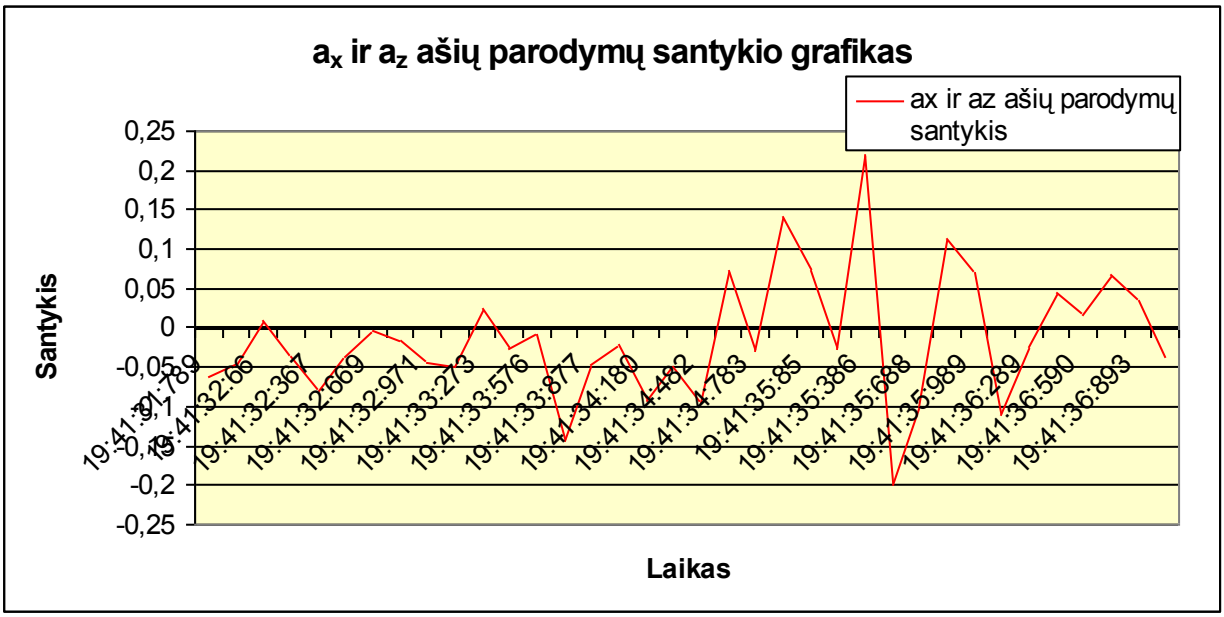
$$\text{vidurkis}_i = \frac{\sum_{i}^{i+} x_i}{5}, \text{ kur i = 1, 2, ...n-5; n - duomenų kiekis, } x_i \text{ - i-toji reikšmė z ašimi (1)}$$

Tuo tikslu buvo norėta surasti staigius duomenų šoktelėjimus. Šio algoritmo rezultatai buvo artimi High-pass filtro rezultatams autorių [EGH08] darbe.

- X ir Z ašies santykio filtras – šiame filtre atmetami visi duomenys, kai parodymai X ašimi yra daugiau nei n kartų mažesni nei Z ašimi. Šio filtro reikšmės yra apskaičiuojamos, kaip santykio formulė:

$$\text{santykis}_j = \frac{a_{x_j}}{a_{z_j}}, \quad (2)$$

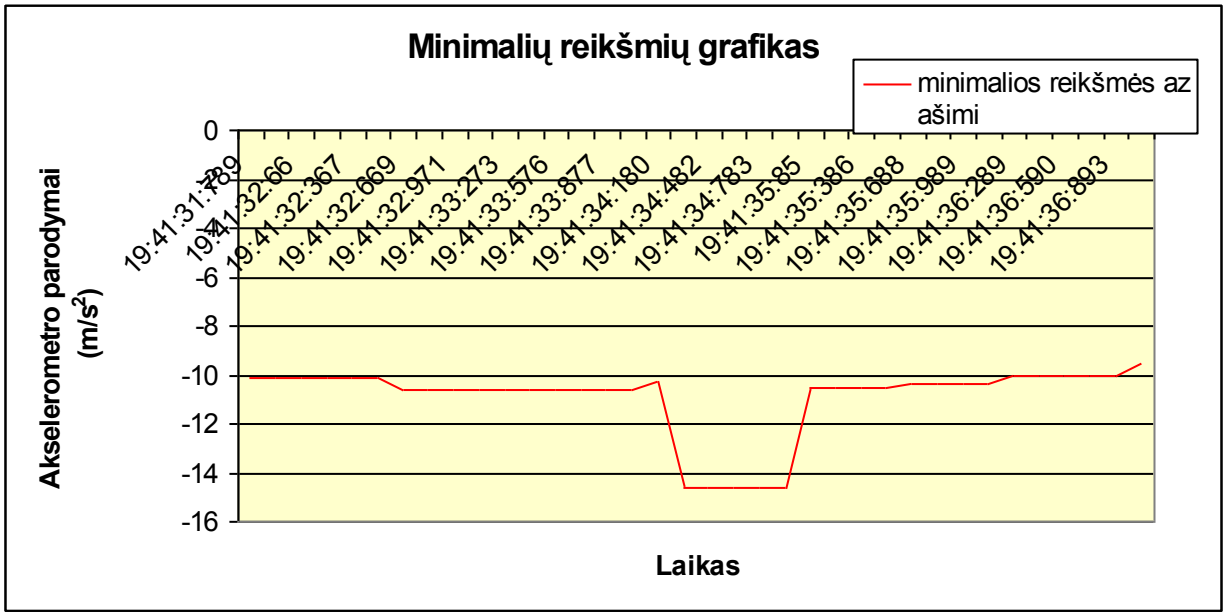
kur a_{x_j} – j-toji reikšmė x ašimi, o a_{z_j} – j-toji reikšmė z ašimi; j = 1..n, kur n – duomenų kiekis.



11 pav. x ir z ašių santykio grafikas

Mūsų darbe taip pat bandyta panaudoti X ir Z ašies santykio filtrą (11 pav.), tačiau jo darbo rezultatai nepateisino lūkesčių, filtras nesugebėjo išryškinti įvykio iš viso duomenų srauto. Tai aiškiai matoma aukščiau pateiktame grafike, duomenys tapo iškraipyti. Nepatenkinami šio filtro rezultatai gali būti susiję su skirtingų modelių akselerometrų naudojimu bei jų tvirtinimu skirtinguose automobilio taškuose mūsų darbe ir [EGH08] straipsnyje.

Dar vienas filtras, kuris buvo pritaikytas norint rasti staigesnius duomenų pakitimus, – minimalių reikšmių filtras (12 pav.).



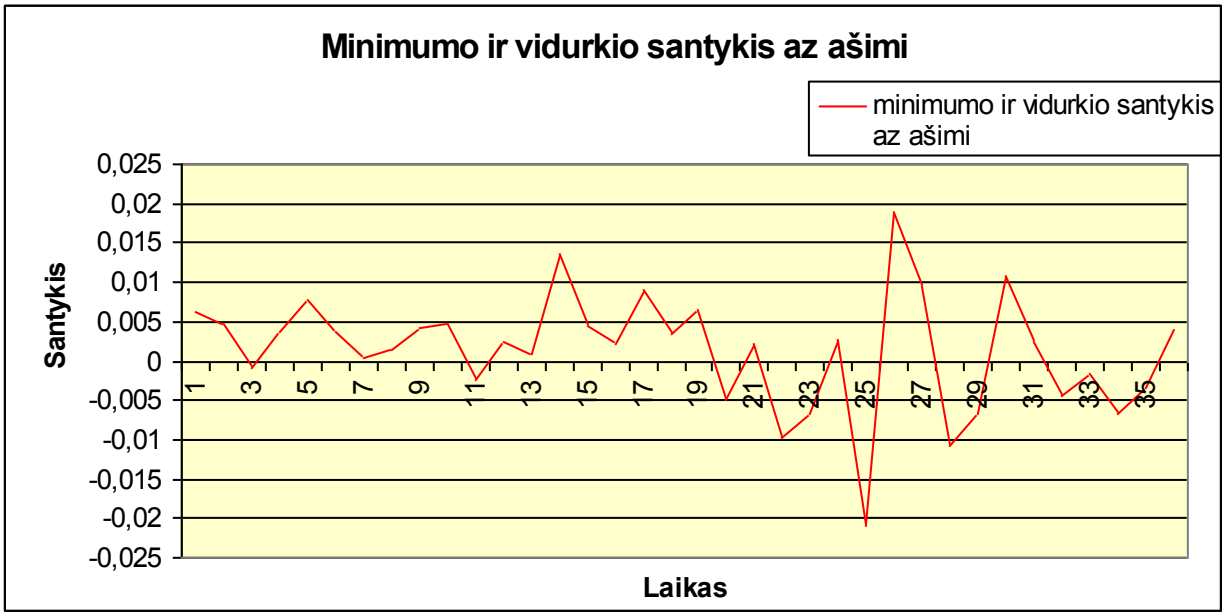
12 pav. Minimalių reikšmių grafikas

Šis algoritmas susijęs su duomenų minimumo išrinkimu iš nurodyto reikšmių intervalo. Realių duomenų grafike kas 5 reikšmes buvo skaičiuojamas minimumas ir slenkančiu būdu atvaizduojamas grafike:

$$\text{minimumas}_i = \min(z_i, z_{i+}, z_{i+}, z_{i+}, z_{i+}), \quad (3)$$

kur i = 1..n-5, z_i - reikšmė z ašimi, n – duomenų kiekis. Galima pastebėti, kur įvyko staigus reikšmių pokytis, bet šiame algoritme, kaip ir prieš tai aprašytuose, duomenų reikšmės bus iškreiptos ir neatitiks realių duomenų reikšmių. Ryšium su tuo, galimai įvykusio įvykio vietos koordinatės taip pat gali būti netikslios.

Mūsų darbe taip pat buvo pritaikytas savitas filtras, kurio [EGH08] straipsnio autoriai nenaudojo. Tai ašies vidurkio ir maksimumo santykio filtras (13 pav.).



13 pav. Minimumo ir vidurkio santykio grafikas

Šiuo filtru norėta sumažinti triukšmą, kitaip tariant sumažinti duomenų perteklių, kad išryškėtų tik duomenys su įvykiais. Galima pastebėti, kad šiame algoritme jau ryškiau yra matomi įvykiai vien dėl to, kad sumažintas triukšmas. Šis metodas pasižymėjo tuo, kad duomenyse kas 5 reikšmes slenkančiu būdu buvo ieškoma duomenų vidurkio bei minimumo santykio:

$$\text{vidurkis}_i = \frac{\sum_{i}^{i+} x_i}{5}, \quad (4)$$

$$\text{minimumas}_i = \min(z_i, z_{i+}, z_{i+}, z_{i+}, z_{i+}), \quad (5)$$

kur i = 1, 2, ...n-5; n - duomenų kiekis, z_i - i-toji reikšmė z ašimi.

$$\text{santykis}_i = \frac{vidurkis}{\min imumas}, \text{ kur i = 1, 2, ...n; n - duomenų kiekis. (6)}$$

- Kitas filtras yra Z - piko filtras – šis filtras atmeta visus duomenis, kurių maksimalus parodymas Z ašimi neviršija iš anksto nurodyto slenksčio. Šio algoritmo įgyvendinimas labai paprastas, kadangi pakanka rasti maksimalią reikšmę ir įsitikinti, kad ji viršija iš anksto nustatytą slenkstį.

Darbe taip pat nuspręsta apdoroti duomenis filtru panašiu į Z-piko filtrą. Vadinamieji išsišokėliai buvo ieškomi pasinaudojus ketvirčio intervalo (ang. interquartile range) metodu, kuris pasižymi tuo, kad duomenys yra skirstomi į kvartilius, o išsišokėliai – tai duomenys, kurie yra žemiau apatinio rėžio ir aukščiau viršutinio rėžio. Visą duomenų aibę reikia išrikiuoti didėjančia tvarka. Kvartiliai yra randami pagal formulę (7, 8), kur randama reikšmė yra vieta išrikiuotoje sekoje.

$$Q1 = (n+1) * 0.25 \quad (7)$$

$$Q3 = (n+1) * 0.75 \quad (8)$$

, kur *n* yra duomenų kiekis.

Vėliau randamas ketvirčio intervalas: IQR=Q1-Q3. Viršutinis ir apatinis rėžiai randami atitinkamai Q1 - 1.5(IQR) ir Q3 + 1.5(IQR).

Ištyrus įvairių kelių ruožų atkarpas pagal ketvirčio intervalo metodą buvo rasti 2 slenksčiai: apatinis bei viršutinis. Kadangi automobilis skirtinguose kelių atkarpose įveikė įvairias kliūtis, tai apatinis slenkstis svyravo intervale –11,503 – -12,954 m/s^2, o viršutinis – -7,606 – -6,718 m/s^2. Apskaičiavus gautų duomenų vidurkį apatinis slenkstis yra laikomas –12,5 m/s^2, o viršutinis – -7 m/s^2.

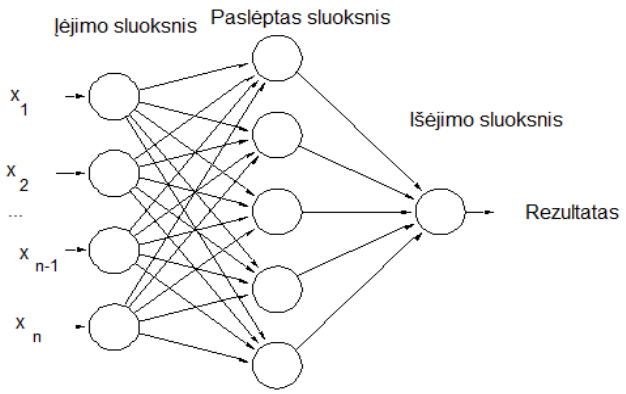
Darbe yra išskirti du analizės etapai: pirminė ir išsamesnė analizė, kuri reikalauja daugiau resursų nei mobilusis telefonas gali suteikti. Kadangi pirminės analizės tikslas yra efektyviai rasti galimus įvykius bei neapkrauti mobiliojo telefono sistemos, buvo nuspręsta naudoti ketvirčio intervalo metodą. Kiti, aukščiau paminėti, filtrai reikalauja daugiau skaičiavimų bei iteracijų skaičiaus, kas sulėtina akselerometro fiksuojamų duomenų gavimą ir gali sąlygoti galimo įvykusio įvykio praradimą.

Kaip jau buvo minėta, kvartilių reikšmės buvo rastos išanalizavus skirtingų kelių bei kliūčių važiuojančio automobilio akselerometro parodymus, o mobiliojo telefono programoje JAVA ME pritaikyti tik rastų slenksčių reikšmės.

Pasinaudojus programa mobiliajame telefone akselerometro reikšmės, kurios buvo mažesnės nei –12,5 m/s^2 ir didesnės už -7 m/s^2, buvo laikomos kaip kelio pakitimai. Kadangi mobiliajame telefone yra atliekama tik pirminė analizė, tai tolesnei analizei atlikti reikalinga duomenų atkarpa, kurioje galimai atsitiko įvykis. Dėl to buvo nuspręsta siųsti 15 duomenų prieš ir po galimo įvykio.

2.3. Detali įvykio analizė

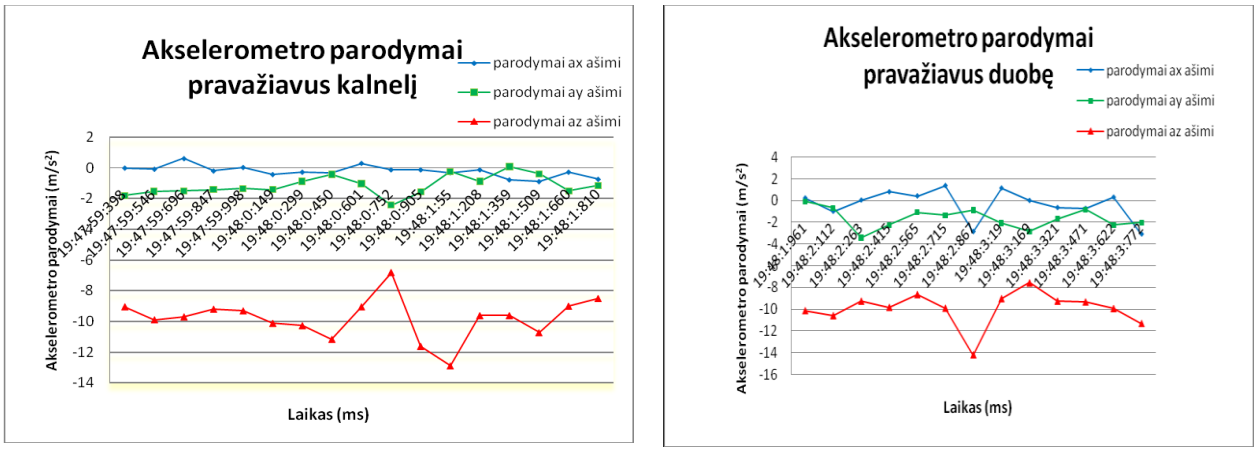
Mobiliajam įrenginiui atpažinus įvykį, jis yra, pasirinktinai, įrašomas į bylą arba siunčiamas tinklu į serverį. Serveris gavęs duomenis arba bylą vykdo tolimesnę duomenų analizę, kurios tikslas išskirti duobes ir greičio ribojimo kalnelius. Įvykio atpažinimas įgyvendintas pasinaudojant apmokytais neuroniniais tinklais. Dirbtiniai neuroniniai tinklai yra skaičiavimų tipas, kuris yra paremtas smegenyse atliekamais skaičiavimais. Neuroniniai tinklai yra naudojami ivairiose srityse kaip bankininkytė, finansai, elektronika, draudimas. Neuroninių tinklų tikslas yra iš įvestų duomenų gauti prasmingą rezultatą. Neuroninių tinklų veikimo schema yra pavaizduota 14 pav.



14 pav. Neuronų tinklas

Neuroniniai tinklai susideda iš 3 skirtingų neurono sluoksnių: įėjimo sluoksnis – pradiniai duomenys; išėjimo sluoksnis - paskutiniame sluoksnyje esantys neuronai ir gaunamas rezultatas; visi kiti sluoksniai vadinami paslėptais. Kiekviena įvesties reikšmė yra sujungta su kiekvienu neuronu per svorio matricą W. Naudojant neuroninį tinklą privalu jį apmokyti. Apmokymas gali vykti dviems būdais: su mokytoju ar be mokytojo [Hea09].

Šiame darbe neuroninių tinklų apmokymas buvo pasirinktas su mokytoju, tai yra buvo duoti pradiniai duomenys(vektoriai) bei pradinių duomenų gaunami rezultatai. Kadangi buvo pastebėti skirtingi akselerometro duomenų pokyčiai važiuojant per duobę ar kalneli, tai buvo norima atskirti kalnelius nuo duobių. (15 pav.)



15 pav. Akselerometro parodymai

Tam tikslui buvo analizuojami duomenys gauti iš mobiliojo telefono. Buvo pastebėta, kad duobę nuo kalnelio galima atskirti nagrinėjant šešis akselerometro parodymų taškus, todėl neuroninių tinklų apmokymui buvo imamas šešių vienetų vektorius. Taip pat buvo žinomi pradinių duomenų

rezultatai. Tinklai realizuoti pasinaudojant Matlab programa. Sėkmingam tinklo apmokymui reikėjo panaudoti 4 paslėptus sluoksnius.

Tinklo apmokymui buvo naudojami iš anksto pasirinkti eksperimentiniai, duobių ir greičio ribojimo kalnelių duomenys. Tiriant apmokyto tinklo efektyvumą buvo pastebėta, kad važiuojant kelio nelygumais (giliai provėžuotas kelias, nelygūs, užlopyti asfalto ruožai) yra retkarčiais identifikuojamos duobės. Norint išvengti klaidinančių rezultatų buvo pritaikytas filtras, kuris tikrina akselerometro parodymų nuokrypio dydį nuo rimties būsenos. Jei nuokrypis yra didelis ir jis tęsiasi kurį laiką, tai užfiksuota atkarpa nėra traktuojama kaip duobė. Taigi, apmokytam neuroniniam tinklui pateikus duomenis, gaunamas atsakymas, koks tai įvykis, duobė ar greičio ribojimo kalnelis. Gautas rezultatas kartu su įvykio koordinatėmis ir data įrašomas į duomenų bazę, kuri naudojama įvykių atvaizdavimui žemėlapyje.

2.4. Duomenų sintezė

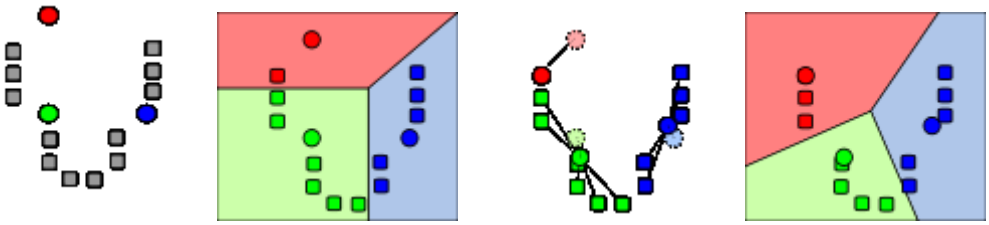
2.4.1. K – means grupavimas

Tai yra vienas iš populiarių dalijimo metodų. Jis yra taupus kompiuterio resursų atžvilgiu, nes skaičiavimo mazgų kiekis nepriklauso nuo duomenų grupių skaičiaus ir puikiai tinka labai dideliems duomenų kiekiams grupuoti. Algoritmo veikimas pasižymi tuo, kad jis grupuoja n objektų išsiskiriančių tam tikru požymiu į k particijų, k < n . Jo tikslas rasti duomenų grupės centrą ir sumažinti vidines grupės variacijas:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

Kur yra k grupių S_i, i = 1, 2, ..., k, ir μ_i yra centroidė visų taškų $x_i \in S_i$.

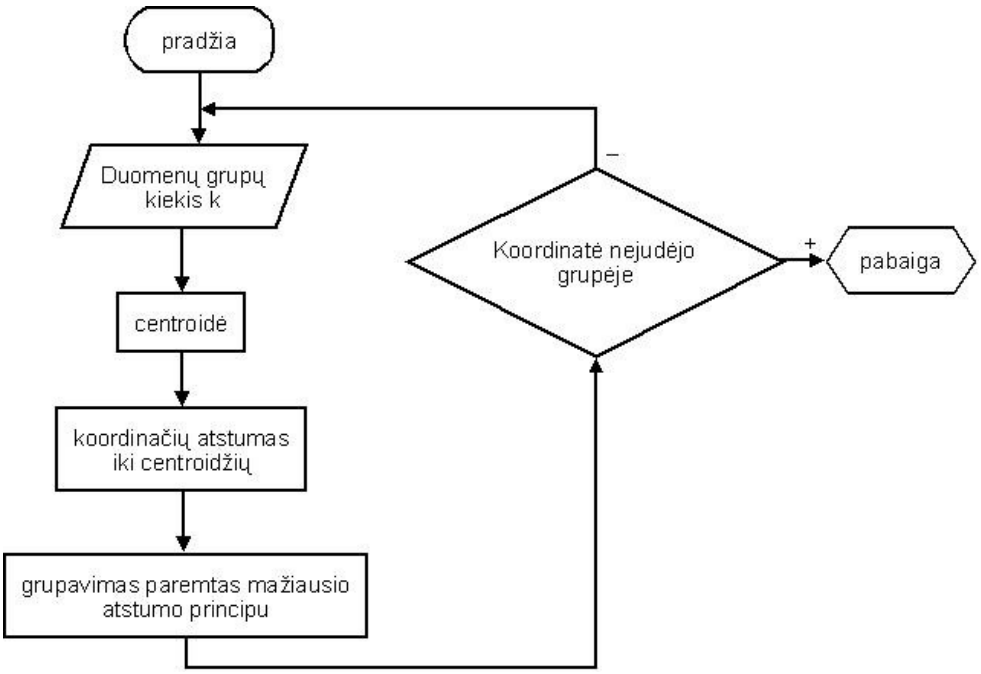
Šis grupavimo algoritmas pradeda veikimą pasirenkant k pradinių centroidžių, pagal kurias duomenys ir bus grupuojami. Pradines centroides galima rinktis taikant įvairius duomenų išrinkimo algoritmus, bet paprasčiausias būdas yra centroides pasirinkti atsitiktinė tvarka, įsitikinant, kad jos bus toli viena nuo kitos, 16 pav. [Moo06]



16 pav. **17 pav.** **18 pav.** **19 pav.**

Grupavimo etapai

Kiekvienas likęs laisvas vektorius (mūsų atveju koordinatė) yra priskiriamas artimiausiai, jau anksčiau pasirinktai centroidei, kaip pavaizduota 17 pav.. Priskyrimas centroidei paremtas mažiausio atstumo principu. Kai visos koordinatės tampa priskirtos centroidėms, yra ieškoma nauja centroidė kiekvienai duomenų grupei. Visas duomenų rinkinys tokiu būdu yra priskiriamas naujoms centroidėms kaip matome 18 pav. Algoritmas yra kartojamas tol, kol duomenų grupių centroidės nustoja keistis tarp iteracijų ir koordinatės nebėra priskiriamos naujoms centroidėms. Visą algoritmą galima pavaizduoti paprasta schema (20 pav.):



20 pav. Grupavimo algoritmo schema

2.4.2. Fuzzy c – means grupavimas

Fuzzy c-means (FCM) yra grupavimo algoritmas, kuris leidžia vienai reikšmei priklausyti dviems ar daugiau grupėms. Šis metoda yra dažnai naudojamas atpažinimui. Algoritmas remiasi funkcijos minimizacija:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\left\|x_i - c_j\right\|^2 \quad , \quad 1 \le m < \infty$$

kur *m* yra bet koks realus skaičius didesnis nei 1, *ir* u_{ij} yra narių x_i grupėje *j* laipsnis, x_i yra *i-ojo* d-matmenų matavimo duomenys, c_j yra grupės d-dimensijos centroidė, ir ||*|| yra bet kokios normos išreiškiančios panašumus tarp išmatuotų duomenų ir centro.

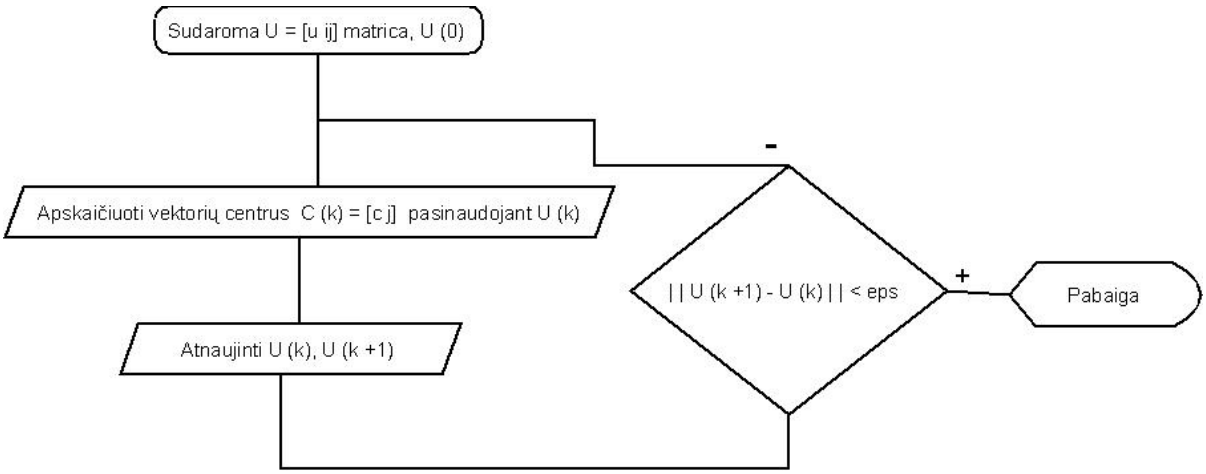
Fuzzy skilimas atliekamas per optimizavimo funkciją, nurodytą anksčiau, su narių u_{ij} ir grupių centroidžių c_j, atnaujinimais tokiais būdais:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\left\|x_i - c_j\right\|}{\left\|x_i - c_k\right\|}\right)^{\frac{2}{m-1}}} \quad , \quad c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

Ši iteracija pasibaigs, kai $\max_{ij}\left\{\left|u_{ij}^{(k+1)} - u_{ij}^{(k)}\right|\right\} < \varepsilon$, kur ε yra nutraukimo kriterijus tarp 0 ir 1, o *k* yra iteracijų skaičius. Ši procedūra konverguoja į lokalųjį minimumą arba J_m balno tašką. [Mat08].

Algoritmas veikimo principas nurodytas 21 pav. :

- Atsitiktine tvarka pasirenkami matricos elementai iš intervalo (0;1)
- Pasinaudojant matricos U elementais apskaičiuojami centroidžių taškai
- Apskaičiuojami nauji matricos U elementai
- Pagal sąlyga ||U(k+1) – U(k)|| < ε sprendžiama ar ieškomi nauji centroidžių taškai.



21 pav. Fuzzy – c veikimo schema

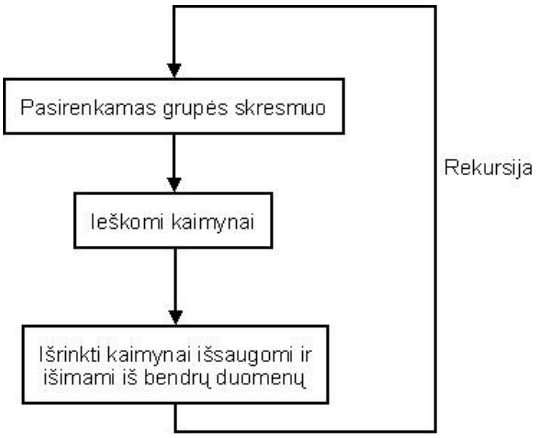
2.4.3. QT (quality threshold) grupavimas

QT grupavimo algoritmas sukurtas 1999 metais plačiausiai yra naudojamas genų grupavimo srityje, bet jį taip pat sėkmingai galima naudoti ir kitokio pobūdžio duomenims grupuoti.

Grupavimo algoritmo veikimo principas (22 pav.):

- Pasirenkamas norimas ieškomos grupės skersmuo.
- Kuriama grupė prijungiant patį artimiausią narį, kitą artimiausią narį ir tt. kol pasiekiamas iš anksto pasirinktas skersmens ilgis.
- Sukurta grupė išsaugoma ir išimama iš bendrų duomenų.
- Atliekama rekursija su likusiais duomenimis kol nebelieka taškų arba lieka tik pavieniai.

Algoritmas patogus tuo, kad nereikalauja iš anksto nusakyti kuriamų grupių skaičiaus bet reikalauja didesnių kompiuterio resursų (lyginant su K – means) atliekant skaičiavimus su didelėmis duomenų grupėmis. [Agi05]



22 pav. Veikimo schema

2.5. Įvykių grupavimo rezultatai

Pasikartojančių įvykių koordinačių grupavimas buvo atliekamas K – means, Qt, ir Fuzzy c means algoritmais, siekiant išsiaiškinti, kurio algoritmo pagalba galima pasiekti geriausių rezultatų, tai yra, kad sugrupuotų įvykių koordinatės būtų atvaizduotos kuo arčiau tikrojo įvykio vietos. Įvykių grupavimą straipsnio [EGH08] autoriai atliko naudodamiesi būtent K-means grupavimo algoritmu.

Pateikiame duomenis (23 pav. Užfiksuoti įvykiai), surinktus viename iš Vilniaus mikrorajonų:

23 pav. Užfiksuoti įvykiai

Žemėlapio apačioje atvaizduota daug pasikartojančių pranešimų apie tą patį įvyki. Dėl GPS prietaiso paklaidos, pranešimų koordinatės nežymiai skiriasi, kas sąlygoja daugkartinį duobės atvaizdavimą. Kitame žemėlapyje (24 pav. K – means) pateikiame K – means algoritmo pagalba sugrupuotus duomenis:

24 pav. K – means

Informacija žemėlapyje tapo aiškesnė, neperkrauta nereikalingais duomenimis.

Atliekamas Qt algoritmo tyrimas su tokiais pat duomenimis, rezultatai pateikiami žemėlapyje (25 pav. Qt):

25 pav. Qt

Atliekant tyrimą pasinaudojant Fuzzy c means algoritmu su tais pačiais duomenimis buvo gauti tokie rezultatai (26 pav.):

26 pav. Fuzzy c means

Fuzzy c means algoritme centroidžių reikšmės priklauso nuo pirminės matricos U elementų parinkimo, kurie yra parenkami atsitiktine tvarka, todėl kiekvieną kartą paleidžiant programą gali įvykti skirtingas iteracijų skaičius.

2.6. Apibendrinimas

Atlikus algoritmų daugkartinius testavimus, su retai ir tankiai išsidėsčiusiomis koordinatėmis, geriausius rezultatus parodė k – means grupavimo algoritmas. Duomenis grupuojant šiuo algoritmu paklaidos žemėlapyje buvo pačios mažiausios, randamos koordinatės buvo arčiausiai įvykusio įvykio. Kiti testuoti algoritmai sugrupuotus įvykius atvaizduodavo su didesne paklaida. Be to buvo pastebėta, kad grupuojant duomenis su skirtingais algoritmais gauti skirtingi grupių centrų taškai bei jų skaičius.

Ši duomenų grupavimo analizė atskleidė, kad, iš tirtų algoritmų, geriausiai duomenis grupuoja K – means grupavimo algoritmas, todėl straipsnio [EGH08] autoriai neapsiriko pasinaudodami K-means algoritmu, kadangi jis yra santykinai taupus kompiuterio resursų atžvilgiu, nes skaičiavimo mazgų kiekis nepriklauso nuo duomenų grupių skaičiaus ir puikiai tinka labai dideliems duomenų kiekiams grupuoti.

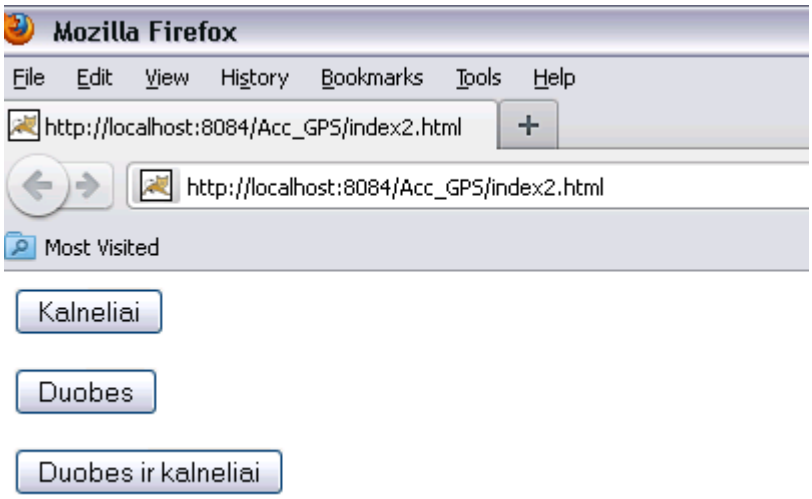
3. Realizacija

3.1. Google Maps API pritaikymas įvykių atvaizdavimui

Kelyje užfiksuotiems įvykiams, kaip duobės ar kalneliai, atvaizduoti yra panaudotas labai patogus ir paprastas būdas, tai yra Google kompanijos teikiama paslauga Google Maps API. Google Maps paslauga - tai nemokamas žemėlapių kūrimas, generavimas ir pristatymas interneto vartotojui ar kitokio pobūdžio paslaugoms, kurioms būtini internetiniai žemėlapiai. Su šia paslauga gaunama daugybė papildomų funkcijų, pavyzdžiui: gatvių žemėlapiai, maršrutų planuotojas, bei įvairių vietovių, esančių mieste, kaip restoranai, kinoteatrai, radimas.

Kaip ir daugelis Google web aplikacijų, Google maps plačiai naudoja JavaScript programavimo kalbą, kuri labai tinka šiuo atveju, nes kontroliuoja aplikacijų programą. Pasirinktos vietovės yra atvaizduojamos dinamiškai pažymint jas raudonu smeigtuku (sudaryto iš keleto pusiau permatomų bitų masyvų formato vaizdinių) ant žemėlapio vaizdo. Tokia žemėlapio technologija leidžia didesnį vartotojų interaktyvumą, sukuriant asinchronines tinklo užklausas Javascript ir XmlHttpRequest pagalba. Žemėlapis iš tikrųjų XmlHttpRequest naudoja taupiai, teikdamas pirmenybę IFrame technologijai, kuri pasižymi tuo, kad leidžia vieną Html dokumentą įmontuoti į kitą Html dokumentą. Taip pat yra naudojamas JSON(JavaScript Object Notation) - tekstinis duomenų apsikeitimo būdas. Šios technologijos žymiai pagerina žemėlapių savybes, kokybės bei žemėlapių atnaujinimo naršyklėje atžvilgiu. Kadangi beveik visas Google Maps parašytas kodas yra JavaScript ir XML, patyrę vartotojai sukūrė įrankius, kurie išplėtė ir pritaikė saviems poreikiams Google Maps galimybes.

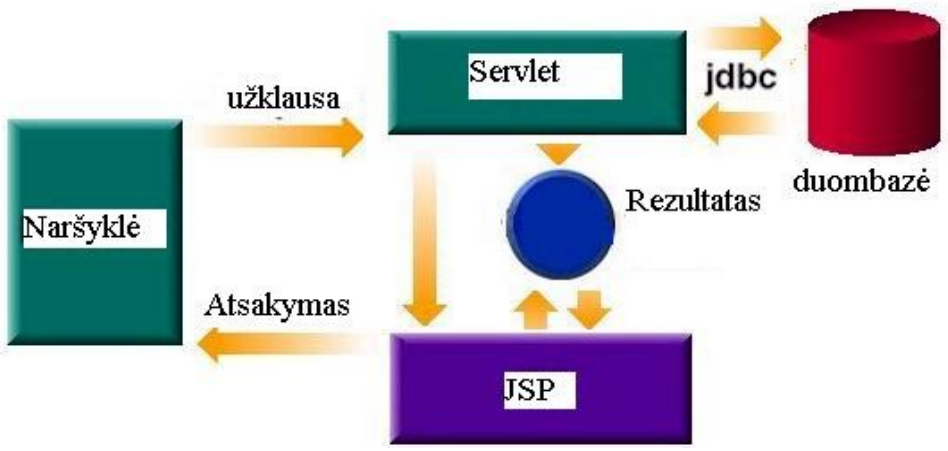
Visos šios aprašytos technologijos yra panaudotos mūsų darbe, kuriant tinklapį eismo situacijoms atvaizduoti. Iš pradžių eismo situacijų stebėtojui yra siūloma pasirinkti, ką jis pageidauja išvysti žemėlapyje, duobes ar kalnelius, kaip pavaizduota 27 pav. Pakeist paveiksleli i duobes ir kalnelius



27 pav. Pradinis programos puslapis

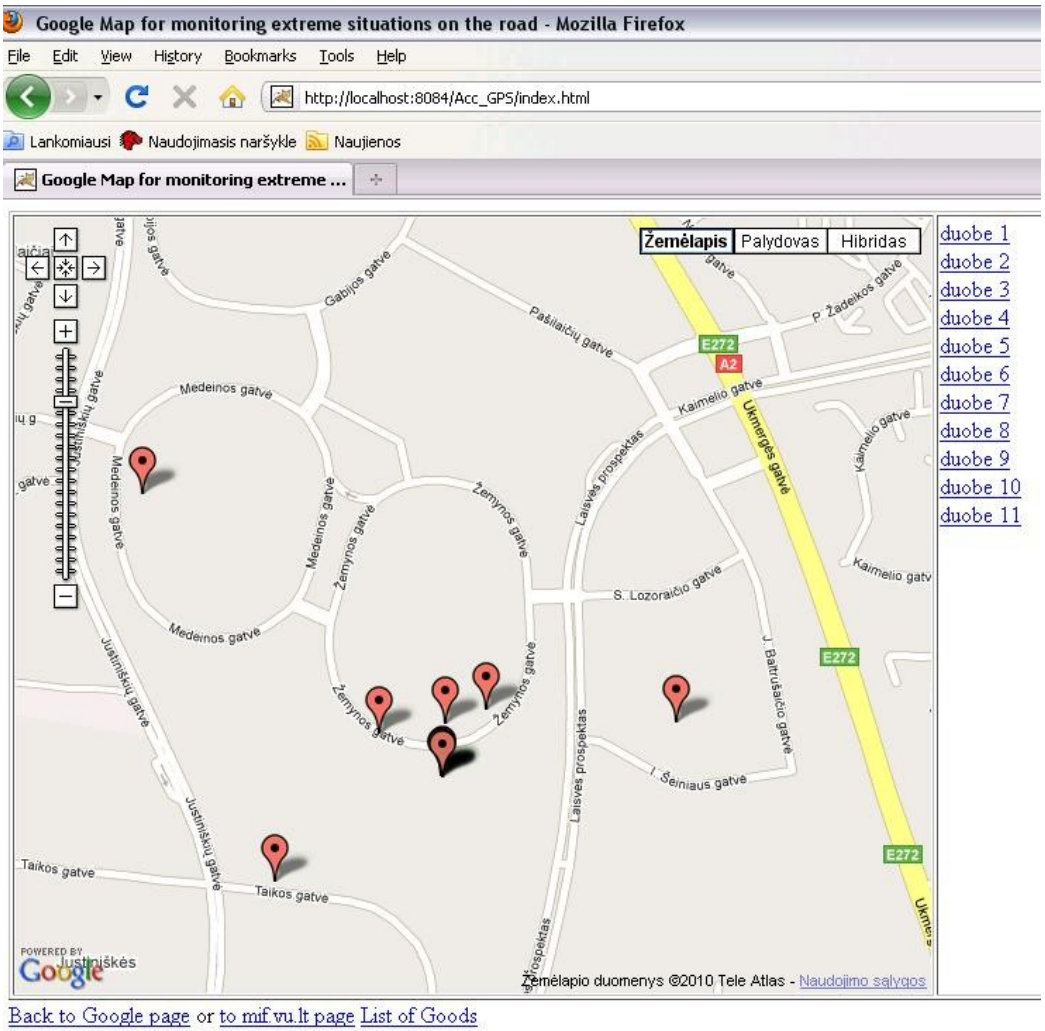
Programa buvo sukurta Java servlets technologija, kuri skirta dinaminių puslapių generavimui. Pagrindinė sąvoka šioje technologijoje – servletas. Tai serveryje esantis komponentas, gaunantis iš kliento (pvz., naršyklės) užklausą (angl. request) bei generuojantis atsaką (angl. response).

Taigi jeigu bus pasirinktas mygtukas „Duobės“, iš naršyklės bus siunčiama užklausa į mūsų sukurtą java klasę prašant gauti rezultatus su duobių koordinatėmis. Ši klasė siųs užklausą duomenų bazei ir pagal užfiksuoto įvykio tipą iš duomenų bazės bus suformuotas atsakymas, susidedantis iš koordinatinių taškų. Detali veiksmų schema pateikiama 28 pav.



28 pav. Programos veiksmų schema

Atsakyme gautos koordinatės bus galutinis rezultatas, kurį Google Maps API atvaizduoja žemėlapyje. Konkretus vaizdas matomas 29 pav.



29 pav. Atvaizduoti duomenys žemėlapyje

Navigacija žemėlapyje yra labai paprasta ir intuityvi. Net nepatyręs kompiuterių vartotojas su sistema apsipranta per kelias minutes ir sugeba naudotis ja be didesnių problemų. Navigacija po žemėlapį atliekama tampant patį žemėlapį pelės pagalba arba naudojantis judėjimo rodyklėmis išdėstytomis kairiajame viršutiniame žemėlapio kampe. Vaizdo priartinimas ir atitolinimas atliekamas zoom įrankio pagalba kairiajame žemėlapio krašte. Yra galimybė pasirinkti žemėlapio vaizdą, kaip standartini žemėlapį, palydovinę nuotrauką arba hibridą, palydovinės nuotraukos ir žemėlapio vaizdą.

3.2. Duomenų bazė

Šiame darbe yra naudojama viena iš populiariausių reliacinių duomenų bazių valdymo sistemų – MySQL. Atlikus tyrimą rezultatai įrašomi į šią duomenų bazę. Šiuo atveju svarbiausia informacija yra automobilio koordinatės bei įvykio tipas. Be to, yra svarbu įvesti laiką, kad atsirastų galimybė atnaujinti duomenis. Duomenų bazė yra sudaryta iš 4 stulpelių, kur CoordX – automobilio platumos koordinatė, CoordY – automobilio ilgumos koordinatė, Įvykio_tipas – užfiksuotas automobilio įvykis, Data – užfiksuoto įvykio data. Lentelės pavidalas yra toks:

CoordX	CoordY	Įvykis	Data

Išvados

Akselerometras ir GPS sistema turi gan paplitusį naudojimą. Pasinaudojant apdorotais šių prietaisų parodymais gali būti pagerintas eismo dalyvių saugumas keliuose. Šiame darbe akselerometro pagalba buvo surinkti pagreičio kitimo duomenys ir sukurti duobių bei greičio ribojimo kalnelių atpažinimo algoritmai. Remiantis tyrimais buvo nustatyta, kad pagreitis svyruojantis 0,17 g vieneto intervale yra dažnai pasitaikantys kelio nelygumai, bet jei g viršijo 0,3 g yra tikimybė, kad buvo kelio kliūtis. Įvykių atpažinimui buvo sukurtos ir pritaikytos įvairios duomenų apdorojimo algoritmų sekos ir modeliai, išnagrinėtos kitų mokslininkų sistemos, metodikos, ir lyginti rezultatai. Kai kurie iš literatūroje išnagrinėtų modelių nedavė teigiamų rezultatų, tai galima paaiškinti skirtingų akselerometru naudojimu. Šiame darbe pirminėje analizėje labiausiai pasiteisino išsišokėlių algoritmas, detalioje analizėje, serverio pusėje, įvykių klasifikavimo atpažinimui puikius rezultatus pateikė apmokytas neuroninis tinklas, kuris apmokytas kalnelius skirti nuo duobių. Siekiant informatyviai pateikti atpažintų įvykių keliuose rezultatus buvo pristatytas žemėlapis, kuriame atvaizduoti atpažinti įvykiai, gauti iš važiuojančio automobilio akselerometro pagalba. Toks būdas gali rasti platų paplitimą. Kelių tiesimo ir priežiūros darbuotojai galės greičiau nustatyti pačias pavojingiausias duobes. Pristatytame žemėlapyje duobių bei greičio ribojimo kalnelių koordinatės yra sutraukiamos pagal k vidurkių algoritmą ir pateikiamas pranešimų apie įvykį skaičius, taip yra palengvinama važiuojamosios dalies būklės analizė. Tokiu būdu municipalinės institucijos taupys laiką, reikalingą kelių analizei, o vairuotojai išvengs nemalonių įvykių. Šiame darbe surinkti, išanalizuoti ir atvaizduoti duomenys tikrai gali pagerinti kelių eismo važiavimo sąlygas bei vairuotojų atidumą, taip sukeliant mažiau eismo įvykių.

Ateityje planuojama tobulinti atpažinimo algoritmus ir sistemą, gebančią klasifikuoti ne tik į duobių ir greičio ribojimo kalnelių, bet ir šulinių, geležinkelio pervažų bei kitas grupes. Eksperimentiniu būdu ištirti, kaip skirtingos automobilių pakabos konstrukcijos įtakoja akselerometro parodymus bei įvykių atpažinimo ir klasifikavimo sistemos kokybę.

Literatūros sąrašas:

[Agi05] Agilent Technologies, QT (Quality Threshold)Clustering http://www.chem.agilent.com/cag/bsp/products/gsgx/Downloads/pdf/qt_clustering.pdf

[ARS06] Khaled Alsabti, Sanjay Ranka, Vineet Singh. An Efficient K-Means Clustering Algorithm. URL: http://www.cs.utexas.edu/~kuipers/readings/Alsabti-hpdm-98.pdf

[BD00] Howard Demuth, Mark Beale Neural Network Toolbox User guide, Version 4, 2000, URL: http://faculty.ksu.edu.sa/67980/Documents/ebook-Artificial_Neural_Network_(Matlab_Toolbox)_%5Bwww.ed2ksearch.tk%5D.pdf

[EGH08] - Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, Hari Balakrishnan. The Pothole patrol: Using a Mobile Sensor Network for Road Surface Monitoring, 2008 URL: http://db.csail.mit.edu/pubs/mobisys08.pdf

[Hea09] Jeff Heaton Introduction to Neural Network with JAVA, 2009, ISBN/ASIN**:** 1604390085

[Insu05] Texas Instruments. Accelerometers and how they work URL: www2.usfirst.org/2005comp/Manuals/Acceler1.pdf

[Mat08] Matteo Matteucci, A tutorial on Clustering Algorithms, 2008 http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/cmeans.html

[Moo06] Andrew Moore. K-means and Hierarchical Clustering. URL: http://www.autonlab.org/tutorials/kmeans.html

[MPR08] Prashanth Mohan, Venkata N. Padmanabhan, Ramachandran Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones, 2008

[RMD01] - G.W. Roberts, X. Meng, A. H. Dodson. The Usage of Kinematic GPS and Triaxial Accelerometers to Monitor the Deflections of Large Bridges , 2001 URL: http://www.fig.net/com6_orange/pdf/Session%20VIII_Paper%202.pdf

[SW01] Peter Shih and Harvey Weinberg. A useful role for the ADXL202 dual-axis accelerometer in speedometer-independent car-navigation system, Analog Dialogue, Volume 35, Nr1, 2001 URL: http://www.analog.com/library/analogDialogue/archives/35-04/ADXL202/

[Tri08] Trimble. All about GPS. URL: http://www.trimble.com/gps/index.shtml

[Vos04] Martin Voshell High Acceleration and the Human Body, 2004 http://csel.eng.ohio-state.edu/voshell/gforce.pdf

Priedai

Išeities kodai:

Serveris duomenų gavimui ir apdorojimui:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package server;

/**
 *
 * @author Rolandas
 */
import java.io.*;
import java.net.*;
import java.util.ArrayList;
import java.sql.*;
import java.text.SimpleDateFormat;

public class Provider {

    ServerSocket providerSocket;
    Socket connection = null;
    BufferedReader inn;
    String message;
    ArrayList date = new ArrayList();
    ArrayList lat = new ArrayList();
    ArrayList lon = new ArrayList();
    ArrayList ivykis = new ArrayList();
    String host = "localhost";
    String database = "akcelerometras";

    Provider() {
    }

    void run() {
        try {
            analyze();
            providerSocket = new ServerSocket(4444);
            System.out.println("Waiting for connection");
            connection = providerSocket.accept();
            System.out.println("Connection received from " + connection.getInetAddress().getHostName());
            inn = new BufferedReader(new InputStreamReader(connection.getInputStream()));
            FileWriter fstream = new FileWriter("files_from_phone/" + Math.round(Math.random() * 100) + ".txt");
            BufferedWriter out = new BufferedWriter(fstream);
            String inputLine;
            while ((inputLine = inn.readLine()) != null) {
//                System.out.println(inputLine);
                if (!inputLine.equals("disconnect")) {
                    out.write(inputLine + "\n");
                }
                if (inputLine.startsWith("disconnect")) {
                    out.close();
                    break;
                }
```

```
        }
        analyze();
    } catch (Exception ioException) {
        System.out.println(ioException.toString());
        ioException.printStackTrace();
    } finally {
        try {
            inn.close();
            providerSocket.close();
        } catch (IOException ioException) {
            System.out.println(ioException.toString());
            ioException.printStackTrace();
        }
    }
}

public static void main(String args[]) {
    Provider server = new Provider();
    while (true) {
        server.run();
    }
}

private void analyze() {
    try {
        File f = new File("./files_from_phone");
        File[] files = f.listFiles();

        for (int i = 0; i <= files.length; i++) {
            FileInputStream fstream = new FileInputStream(files[i]);
            DataInputStream in = new DataInputStream(fstream);
            BufferedReader br = new BufferedReader(new InputStreamReader(in));
            String strLine;
            int j = 0;
            int k = 0;
            FileWriter fstreamWrite = new FileWriter("out.txt");
            BufferedWriter output = new BufferedWriter(fstreamWrite);
            String[] temp = null;
            ArrayList<Double> duomenys = new ArrayList();

            while ((strLine = br.readLine()) != null) {

                j++;
                if (j == 16 + (k * 30)) {

                    temp = strLine.split(";");
                    duomenys.add(Double.parseDouble(temp[3].toString()));
                    strLine = br.readLine();
                    j++;
                    date.add(temp[0]);
                    lat.add(temp[4]);
                    lon.add(temp[5]);
                    for (int p = 1; p < 14; p++) {
                        temp = strLine.split(";");
                        duomenys.add(Double.parseDouble(temp[3].toString()));
                        strLine = br.readLine();
                        j++;

                    }
```

```
        if (tikrinimasKalnelio(duomenys)) {

            for (int p = 0; p < 6; p++) {
                output.write(duomenys.get(p) + " ");
            }
            output.write("\n");
            k++;
            duomenys.clear();
        } else {
            date.remove(date.size() - 1);
            lat.remove(lat.size() - 1);
            lon.remove(lon.size() - 1);
            duomenys.clear();
            k++;
        }
    }
}
output.close();

in.close();
// files[i].delete();

Runtime rt = Runtime.getRuntime();
Process pr = rt.exec("matlab -nosplash -nodesktop -nodisplay -r neuroniniaiTinklai.m");

new Thread().sleep(60000);
FileInputStream file = new FileInputStream("res.txt");
DataInputStream is = new DataInputStream(file);
BufferedReader buf = new BufferedReader(new InputStreamReader(is));

while ((strLine = buf.readLine()) != null) {
    String[] t = strLine.split(";");
    double t1 = Double.parseDouble(t[0]);
    double t2 = Double.parseDouble(t[1]);
    System.out.println(t1 + " " + t2);

    if ((t1 < t2) && (t2 > 0.8)) {
        ivykis.add("kalnelis");
    } else if ((t1 > t2) && (t1 > 0.8)) {
        ivykis.add("duobe");
    } else {
        ivykis.add("nera");
    }
}



String url = "jdbc:mysql://" + host + "/" + database;

try {
    Class.forName("com.mysql.jdbc.Driver").newInstance();
} catch (java.lang.ClassNotFoundException e) {
    System.err.print("ClassNotFoundException: ");
    System.err.println(e.getMessage());
}

try {
    //   con = DriverManager.getConnection(url, "demo", "demo");
```

```
                Connection con = DriverManager.getConnection(url, "root",
"akselerometras");//DriverManager.getConnection(url,"demo","demo");
                Statement stmt = con.createStatement();
                FileWriter file3 = new FileWriter("ivykiai.txt");
                BufferedWriter outt = new BufferedWriter(file3);
                for (int n = 0; n < ivykis.size(); n++) {
                   System.out.println(n);
                   if (!(ivykis.get(n).equals("nera"))) {
                     //  String updateString = "INSERT INTO duomenys VALUES ('0.0','0.0','" + ivykis.get(n) + "','" +
date.get(n) + "')";
                      String updateString = "INSERT INTO duomenys VALUES ('" + lat.get(n) + "','" + lon.get(n) + "','" +
ivykis.get(n) + "','" + date.get(n) + "')";
                      int rs2 = stmt.executeUpdate(updateString);

                   }
                   outt.write(ivykis.get(n) + " " + date.get(n) + "\n");
                }
                outt.close();

             } catch (SQLException ex) {
                System.err.println("SQLException: " + ex.getMessage());
             }
          }
       } catch (Exception e) {
       }
    }

    private boolean tikrinimasKalnelio(ArrayList<Double> kalnelis) {
       if (kalnelis.get(1) > -7) {
          for (int i = 2; i < 7; i++) {
             if (kalnelis.get(i) < -11) {
                return true;
             }
          }
          return false;
       } else {
          int kiekis = 0;
          for (int i = 4; i < kalnelis.size(); i++) {
             if ((kalnelis.get(i) > -8.4) || (kalnelis.get(i) < -11.3)) {
                kiekis++;
             }
          }
          if (kiekis >= 4) {
             return false;
          } else {
             return true;
          }
       }

    }
}
```

Serverinė dalis, skirta duomenų atvaizdavimui žemėlapyje:

```
/*
 * Duobes.java
 *
 * Created on Šeštadienis, 2011, Gegužės 3, 11.27
 */
```

```
package AccGPS;

import java.io.*;
import java.net.*;
import java.util.Random;

import javax.servlet.*;
import javax.servlet.http.*;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.math.BigDecimal;
import java.sql.*;
/**
 *
 * @author Rolandas
 * @version
 */
public class Duobes extends HttpServlet {
    String host     = "localhost";
    String database = "akcelerometras";
    String takelis = "C:/Documents and Settings/Ana/Desktop/Magistrinis/Acc+GPS/build/web/example.txt";
    private static int i = 0;
    private static String[] AutoNr = new String[1000];
    private static double[] CoordX = new double[1000];
    private static double[] CoordY = new double[1000];
    private static double[] Xacc = new double[1000];
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        System.out.println("hahaha");
        response.setContentType("text/html;charset=UTF-8");
        final PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Database system information page</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Database system information page</h1>");
        out.println("Nelygumų kelyje duomenys apdoroti! ");
        out.println(" <form action=index.html method=POST> ");
        out.println(" <tr> <input type=submit value=Žemėlapis> </tr> ");
        out.println(" </form> ");
        out.println("</body>");
        out.println("</html>");
        out.close();




        String url = "jdbc:mysql://" + host + "/" + database;
        Connection con;
        String createString, createString2;
        createString = "SELECT * FROM duomenys where Ivykis='duobe'";
      //  createString2 = "DELETE FROM duomenys where ZACC < 3 and ZACC >0.6 and
DATE_SUB(CURDATE(),INTERVAL 80 DAY) >= time";
        Statement stmt, stmt2;
```

```
        try {Class.forName ( "com.mysql.jdbc.Driver" );
            con = DriverManager.getConnection(url, "root", "akselerometras");

            stmt2 = con.createStatement();
          //  int rs2 =  stmt2.executeUpdate(createString2);
            stmt = con.createStatement();
            ResultSet rs =  stmt.executeQuery(createString);
            i = 0;
            FileOutputStream fout2 = new FileOutputStream(takelis, true);
            PrintStream out2 = new PrintStream(fout2);
            while (rs.next()) {
               CoordX[i] = rounder(rs.getDouble("COORDX"));
               CoordY[i] = rounder(rs.getDouble("COORDY"));
               System.out.println(CoordX[i]+" "+CoordY[i]+" ");
               i++;
               out.println(CoordX[i]+ " , " + CoordY[i] );
            }
            out2.close();
                  fout2.close();
            stmt.close();
            con.close();

        } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());

           } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }


        encapsulate(CoordX, CoordY, i);

        //     writeToFile();
    }

       public static double max(double[] t, int z) {
        double sum =0.0;
        for (int i=0; i<z; i++) {
            sum = sum +t[i];
        }
    return sum;
}

    public static double rounder(double a) {
        BigDecimal bd = new BigDecimal(a);
        BigDecimal bd_round = bd.setScale( 5, BigDecimal.ROUND_HALF_UP );
        return Math.abs(bd_round.doubleValue());
    }



    // K means

       private void encapsulate(double[] CoordX, double[] CoordY, int length) {
```

```
        int eventAmount = 0;
        String takelis = "C:/Documents and Settings/Ana/Desktop/Magistrinis/Acc+GPS/build/web/example.txt";
        File f = new File(takelis);
        f.delete();

        for (int i = 0; i < length; i++){
            eventAmount = 1;
            double oneX = CoordX[i];
            double oneY = CoordY[i];
            if(oneX != -1){
                for(int a = i; a < length; a++){
                    if(Math.sqrt(Math.pow((CoordX[a+1] - oneX),2) + Math.pow((CoordY[a+1] - oneY),2)) < 0.001){
                        CoordX[a+1] = -1; //0.001
                        CoordY[a+1] = -1;
                        eventAmount++;
                    }
                }

                try{

                    FileOutputStream fout = new FileOutputStream("C:/Documents and
Settings/Ana/Desktop/Magistrinis/Acc+GPS/build/web/example.txt", true);
                    PrintStream out = new PrintStream(fout);
                    System.out.println(" 2222 "+CoordX[i]+" "+CoordY[i]+" ");
                    out.println(oneX+"|"+oneY+"|pranesimu apie duobe: "+eventAmount+"<br>|duobe "+(i+1));
                   // out.newLine();
                    out.close();
                    fout.close();
                }catch (Exception e){
                    System.err.println("Error: " + e.getMessage());
                }

            }
        }

    }
     /*

    private void encapsulateQT(double[] CoordX, double[] CoordY, int length) {
    Random generator = new Random();
    int eventAmount = 0 , first = 0;
    double cluster_diameter = 0.001;


     File f = new File("C:/Documents and Settings/Ana/Desktop/Informatika/VI
semestras/kursinis/Acc+GPS2/Acc+GPS/build/web/example.txt");
        f.delete();
        for (int i = 0; i < length; i++){
            eventAmount = 1;
            first = generator.nextInt(length);
            System.out.println("QT first: "+first +" "+length);
            double oneX = CoordX[first];
            double oneY = CoordY[first];
            if(oneX != -1){
                for(int a = i; a < length; a++){                                    //0.001
                    if(Math.sqrt(Math.pow((CoordX[a+1] - oneX),2) + Math.pow((CoordY[a+1] - oneY),2)) <
cluster_diameter){
                        CoordX[a+1] = -1;
                        CoordY[a+1] = -1;
```

```
                    eventAmount++;
                }
            }

            try{
                FileWriter fstream = new FileWriter("C:/Documents and Settings/Ana/Desktop/Informatika/VI
semestras/kursinis/Acc+GPS2/Acc+GPS/build/web/example.txt", true);
                BufferedWriter out = new BufferedWriter(fstream);
                out.write(oneX+"|"+oneY+"|pranesimu apie duobe: "+eventAmount+"<br>|duobe ");
                out.newLine();
                out.close();
            }catch (Exception e){
                System.err.println("Error: " + e.getMessage());
            }

        }
    }


}/*

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/** Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

/** Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

/** Returns a short description of the servlet.
 */
public String getServletInfo() {
    return "Short description";
}
// </editor-fold>
//Fuzzy c means
/*
    private static void encapsulateFuzzyCMeans(double[] CoordX, double[] CoordY, int length) throws
FileNotFoundException{
    int m = 2;
     double[][] u = new double[1000][1000];
     double[] koord = new double[1000];
     double[][] c = new double[1000][1000];
     double[][] d = new double[1000][1000];
     double[][] u_update = new double[1000][1000];
     double[] li= new double[1000];
     double[] sum= new double[1000];
     Random gen = new Random();
```

```
        double denominator=0;
        int num ;
        int eventAmount;
        int n = 0;


        File f = new File("C:/Documents and Settings/Ana/Desktop/Informatika/VI
semestras/kursinis/Acc+GPS2/Acc+GPS/build/web/example.txt");
        f.delete();

        num=kiekis(CoordX, CoordY, length);
        //Kuriama matrica U(0)
        for (int kl = 0; kl < 1; kl++){
          for (int i = 0; i < length; i++){
             u[kl][i] = gen.nextDouble();
             sum[i]=u[kl][i];
             li[i]=1-u[kl][i];
          }
        }
        for (int kl = 1; kl < num-1; kl++){
          for (int i = 0; i < length; i++){
             u[kl][i] = gen.nextDouble()*li[i];
             li[i]=1-u[kl][i];
             sum[i]=sum[i]+u[kl][i];
          }
        }
        for (int kl = num-1; kl < num; kl++){
          for (int i = 0; i < length; i++){
             u[kl][i] = 1-sum[i];
          }
        }


        boolean event =true;

        while(event){
          for (int kl = 0; kl < num; kl++){
             c[kl][1]= 0;
             c[kl][2]= 0;
             denominator=0;
             for (int i = 0; i < length; i++){
                c[kl][1]=  c[kl][1] + Math.pow(u[kl][i],m)*CoordX[i];
                c[kl][2]= c[kl][2] + Math.pow(u[kl][i],m)*CoordY[i];
                denominator +=  Math.pow(u[kl][i],m);
             }
            if (denominator!=0){
                c[kl][1]=  c[kl][1]/denominator;
                c[kl][2]=  c[kl][2]/denominator;
             }
          }

          for(int kl =0; kl<num; kl++){
             for (int i = 0; i < length; i++){
               d[kl][i] = (Math.pow(Math.abs(CoordX[i]-c[kl][1]),2) + Math.pow(Math.abs(CoordY[i]-c[kl][2]),2));
               if(d[kl][i]==0) d[kl][i]=1;
             }
          }
          for (int j = 0; j < length; j++)
          {
```

```
            for(int kl = 0; kl<num; kl++){
               double denominator2 = 0.0;
               for (int ii = 0; ii < num; ii++)
               {
                  if(d[kl][ii]!=0)
                     denominator2 = denominator2 +Math.pow(d[kl][j]/d[ii][j],(2.0/(m-1.0)));
               }
                if(denominator2!=0){
                  u_update[kl][j] = 1.0/denominator2;
                  u[kl][j]=Math.abs(u[kl][j]-u_update[kl][j]);
                }
                else event=false;
            }
         }

         n=0;
         if(max(u, num, length) < 0.01){
           n++;
           event=false;
         }


          for(int kl=0; kl<num; kl++){
            for (int i=0; i<length; i++){
               u[kl][i] = u_update[kl][i];
            }
         }
      }

      for(int kl=0; kl<num; kl++){
         try{
            FileOutputStream fout = new FileOutputStream("C:/Documents and Settings/Ana/Desktop/Informatika/VI
semestras/kursinis/Acc+GPS2/Acc+GPS/build/web/example.txt", true);
            PrintStream out = new PrintStream(fout);
            out.println(c[kl][1]+"|"+c[kl][2]+"|pranesimu apie duobe: "+n+" <br>|duobe ");
            out.close();
            fout.close();
         }
         catch (Exception e){
               System.err.println("Error: " + e.getMessage());
         }

      }

   }



      public static int kiekis(double[] CoordX1, double[] CoordY1, int length) {
       int kiekis = 0;
       double[] neigiamas = new double[1000];
       for (int i = 0; i < length; i++){
          double oneX = CoordX1[i];
          double oneY = CoordY1[i];
          if(neigiamas[i] != -1.0){
             kiekis++;
             for(int a = i; a < length; a++){
                if(Math.sqrt(Math.pow((CoordX1[a+1] - oneX),2) + Math.pow((CoordY1[a+1] - oneY),2)) < 0.001){
                   neigiamas[a+1] = -1;
```

```
                }
            }
        }
    }
    return kiekis;
  }

  public static double max(double[][] t, int z, int length) {
    double sum =0.0;
    for (int k=0; k<z; k++) {
      for (int i=0; i<length; i++) {
        sum = sum +t[k][i];
      }
    }
  return sum;
}


  */

}

/*
 * Kalnelis.java
 *
 * Created on Šeštadienis, 2011, Gegužės 3, 11.27
 */

package AccGPS;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.math.BigDecimal;
import java.sql.*;


public class Kalnelis extends HttpServlet {


  String host     = "localhost";
  String database = "akcelerometras";
  private static int i = 0;
  private static double[] CoordX = new double[1000];
  private static double[] CoordY = new double[1000];

  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    final PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Database system information page</title>");
    out.println("</head>");
```

```
        out.println("<body>");
        out.println("<h1>Database system information page</h1>");
        out.println("Greičio ribojimo kalnelių duomenys apdoroti! ");
        out.println(" <form action=index.html method=POST> ");
        out.println(" <tr> <input type=submit value=Žemėlapis> </tr> ");
        //    out.println("<a href=index.html target=_blank>index.html</a>");
        out.println(" </form> ");
        out.println("</body>");
        out.println("</html>");
        out.close();

        String url = "jdbc:mysql://" + host + "/" + database;
        Connection con;
        String createString, createString2;
        createString = "SELECT * FROM duomenys where Ivykis='kalnelis'";
       // createString2 = "DELETE FROM duomenys where XACC < -3.0 and DATE_SUB(CURDATE(),INTERVAL 30
DAY) >= time";
        Statement stmt, stmt2;

        try {
            Class.forName ( "com.mysql.jdbc.Driver" );
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            //   con = DriverManager.getConnection(url, "demo", "demo");
            con = DriverManager.getConnection(url, "root",
"akselerometras");//DriverManager.getConnection(url,"demo","demo");
            stmt2 = con.createStatement();
         //   int rs2 =  stmt2.executeUpdate(createString2);
            stmt = con.createStatement();
            ResultSet rs =  stmt.executeQuery(createString);
            i = 0;
            while (rs.next()) {
               CoordX[i] = rounder(rs.getDouble("COORDX"));
               CoordY[i] = rounder(rs.getDouble("COORDY"));
               System.out.println(CoordX[i]+" "+CoordY[i]+" ");
               i++;
            }
            stmt.close();
            con.close();

        } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }
        writeToFile();
    }

    public static double rounder(double a) {
        BigDecimal bd = new BigDecimal(a);
        BigDecimal bd_round = bd.setScale( 5, BigDecimal.ROUND_HALF_UP );
        return Math.abs(bd_round.doubleValue());
    }

    private static void writeToFile() {
        try{
            String takelis = "C:/Documents and Settings/Ana/Desktop/Magistrinis/Acc+GPS/build/web/example.txt";
```

```
            FileWriter fstream = new FileWriter(takelis);
            BufferedWriter out = new BufferedWriter(fstream);
            for(int a = 0; a<i;a++){
               out.write(CoordX[a]+"|"+CoordY[a]+"| <br>kalnelis|kalnelis "+(a+1));
               out.newLine();
            }
            out.close();
        }catch (Exception e){
            System.err.println("Error: " + e.getMessage());
        }
    }



    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /** Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
    /** Returns a short description of the servlet.
     */
    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}

/*
 * AllData.java
 *
 */

package AccGPS;


import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;

import java.math.BigDecimal;
import java.sql.*;


public class AllData extends HttpServlet {
```

```
    String host     = "localhost";
    String database = "akcelerometras";
    private static int i = 0, j = 0;
    private static double[] CoordX = new double[1000];
    private static double[] CoordY = new double[1000];
    private static double[] CoordX2 = new double[1000];
    private static double[] CoordY2 = new double[1000];


    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        final PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Database system information page</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Database system information page</h1>");
        out.println("Autoįvykių duomenys apdoroti! ");
        out.println(" <form action=index.html method=POST> ");
        out.println(" <tr> <input type=submit value=Žemėlapis> </tr> ");
        //    out.println("<a href=index.html target=_blank>index.html</a>");
        out.println(" </form> ");
        out.println("</body>");
        out.println("</html>");
        out.close();

        String url = "jdbc:mysql://" + host + "/" + database;
        Connection con;
        String createString, createString2;
        createString = "SELECT * FROM duomenys";
        Statement stmt;

        try {
            Class.forName ( "com.mysql.jdbc.Driver" );
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            //   con = DriverManager.getConnection(url, "demo", "demo");
            con = DriverManager.getConnection(url, "root",
"akselerometras");//DriverManager.getConnection(url,"demo","demo");
            stmt = con.createStatement();
            ResultSet rs =  stmt.executeQuery(createString);
            i = 0;
            while (rs.next()) {
                if (rs.getString("Ivykis").equals("duobe"))
                {
                    System.out.println("duobe"+rs.getString("Ivykis"));
                    CoordX[i] = rounder(rs.getDouble("COORDX"));
                    CoordY[i] = rounder(rs.getDouble("COORDY"));
                    i++;
                }
                else {
                    System.out.println("kalne");
                    CoordX2[j] = rounder(rs.getDouble("COORDX"));
```

```
                CoordY2[j] = rounder(rs.getDouble("COORDY"));
                System.out.println(CoordX2[j]+" "+CoordY2[j]);
                j++;
            }
        }
        stmt.close();
        con.close();

    }catch(SQLException ex) {
        System.err.println("SQLException: " + ex.getMessage());
    }
    encapsulate(CoordX, CoordY, j, i);
}

public static double rounder(double a) {
    BigDecimal bd = new BigDecimal(a);
    BigDecimal bd_round = bd.setScale( 5, BigDecimal.ROUND_HALF_UP );
    return Math.abs(bd_round.doubleValue());
}

    private void encapsulate(double[] CoordX3, double[] CoordY3, int length, int len){
    int eventAmount = 0;
    String takelis = "C:/Documents and Settings/Ana/Desktop/Magistrinis/Acc+GPS/build/web/example.txt";
    File f = new File(takelis);
    f.delete();
    try{
        FileOutputStream fout = new FileOutputStream(takelis, true);
        PrintStream out = new PrintStream(fout);
        for (int i = 0; i < length; i++){
            eventAmount = 1;
            double oneX = CoordX3[i];
            double oneY = CoordY3[i];
            if(oneX != -1){
                for(int a = i; a < length; a++){
                    if(Math.sqrt(Math.pow((CoordX3[a+1] - oneX),2) + Math.pow((CoordY3[a+1] - oneY),2)) < 0.001){
                        CoordX3[a+1] = -1; //0.001
                        CoordY3[a+1] = -1;
                        eventAmount++;
                    }
                }
                System.out.println(CoordX[i]+" "+CoordY[i]);
                out.println(oneX+"|"+oneY+"|pranesimu apie duobe: "+eventAmount+"<br>|duobe "+(i+1));
            }
        }
        for(int a = 0; a<len;a++){
            out.println(CoordX2[a]+"|"+CoordY2[a]+"|<br>kalnelis|kalnelis "+(a+1));
        }
        out.close();
        fout.close();
        }catch (Exception e){
            System.err.println("Error: " + e.getMessage());
        }
 }

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/** Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Returns a short description of the servlet.
     */
    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}
```

Mobiliajam įrenginiui skirta aplikacija:

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Graphics;
import javax.microedition.location.LocationListener;
import javax.microedition.sensor.Data;
import javax.microedition.sensor.DataListener;
import javax.microedition.sensor.SensorConnection;
import javax.microedition.sensor.SensorInfo;
import javax.microedition.sensor.SensorManager;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;
import java.util.Calendar;
import java.util.Date;
import java.util.Vector;
import javax.microedition.io.Connector;
import javax.microedition.io.OutputConnection;
import javax.microedition.io.SocketConnection;
import javax.microedition.location.Coordinates;
import javax.microedition.location.Criteria;
import javax.microedition.location.Location;
import javax.microedition.location.LocationProvider;

public class AkselerometroSensorius extends Canvas implements CommandListener, DataListener, LocationListener,
Runnable {

    private SensoriausTestas midlet;
    private int width, height;
    private int d, r;
    private int x_int, y_int, z_int;
    private double x, y, z;
```

```
protected int xcoord, ycoord;
private Command startOnline;
private Command startOffline;
private Command sendData;
private Command disconnect;
private Command exit1;
private static SensorConnection sensor = null;
private SensorInfo infos[];
private static boolean isStopped = false;
private static boolean type_int = false;
private static boolean sensor_found = false;
private static final int BUFFER_SIZE = 1;
private Thread thread = null;
final public int SIZE = 10000;
SocketConnection client;
OutputStream os;
OutputStream out;
PrintStream output;
OutputConnection connection;
boolean offline = false;
Vector listX = new Vector();
Vector listY = new Vector();
Vector listZ = new Vector();
Vector listAllAxes = new Vector();
boolean pit = false;
int counter = 0;
double lat, lon;

public AkselerometroSensorius(SensoriausTestas midlet) {
    this.setFullScreenMode(true);
    this.midlet = midlet;
    x = 0;
    y = 0;
    z = 0;
    r = 5;
    d = 2 * r;
    width = getWidth();
    height = getHeight();
    xcoord = width / 2 - r;
    ycoord = height / 2 - r;
    exit1 = new Command("Exit1", Command.SCREEN, 1);
    startOnline = new Command("Start online", Command.SCREEN, 1);
    startOffline = new Command("Start offline", Command.SCREEN, 2);
    sendData = new Command("Send/save data", Command.SCREEN, 3);
    disconnect = new Command("Disconnect and Exit", Command.SCREEN, 4);
    this.addCommand(startOffline);
    this.addCommand(sendData);
    this.addCommand(disconnect);
    this.addCommand(exit1);
    this.addCommand(startOnline);
    this.setCommandListener(this);

}

private synchronized void initSensor() {
    sensor = openSensor();
    if (sensor == null) {
        return;
    }
```

```
    try {
      sensor.setDataListener(this, BUFFER_SIZE);
      while (!isStopped) {
        try {
          wait();
        } catch (InterruptedException ie) {
        }
      }
      sensor.removeDataListener();
    } catch (IllegalMonitorStateException imse) {
      imse.printStackTrace();
    } catch (IllegalArgumentException iae) {
      iae.printStackTrace();
    }
    try {
      sensor.close();
    } catch (IOException ioe) {
      ioe.printStackTrace();
    }
    if (isStopped) {
      sensor = null;
    }
  }

  private SensorConnection openSensor() {
    infos = SensorManager.findSensors("acceleration", null);
    if (infos.length == 0) {
      return null;
    }
    int datatypes[] = new int[infos.length];
    int i = 0;
    String sensor_url = "";
    if (!type_int) {
      System.out.println("Searching TYPE_DOUBLE sensor...");
      while (!sensor_found) {
        datatypes[i] = infos[i].getChannelInfos()[0].getDataType();
        if (datatypes[i] == 1) {
          sensor_url = infos[i].getUrl();
          System.out.println("Sensor: " + sensor_url + ": TYPE_DOUBLE found.");
          sensor_found = true;
        } else {
          i++;
        }
      }
    } else if (type_int) {
      System.out.println("Searching TYPE_INT sensor...");
      while (!sensor_found) {
        datatypes[i] = infos[i].getChannelInfos()[0].getDataType();
        if (datatypes[i] == 2) {
          sensor_url = infos[i].getUrl();
          System.out.println("Sensor: " + sensor_url + ": TYPE_INT found.");
          sensor_found = true;
        } else {
          i++;
        }
      }
    }
    try {
      return (SensorConnection) Connector.open(sensor_url);
```

```
            } catch (IOException ioe) {
                ioe.printStackTrace();
                return null;
            }
        }

        public void paint(Graphics g) {
            int size = 10;
            width = getWidth();
            height = getHeight();
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, width, height);
//          g.setColor(255, 0, 0);
//          g.drawRect(0, 0, width - 1, height - 1);
//          g.setColor(0, 0, 0);
//          g.fillArc(xcoord, ycoord, d, d, 0, 360);
//          g.setColor(0, 255, 0);
//          g.drawLine(0, height / 2, width, height / 2);
//          g.drawLine(width / 2, 0, width / 2, height);
            g.setColor(0, 0, 0);
            if (!type_int) {
                x_int = (int) (x * 10);
                y_int = (int) (y * 10);
                z_int = (int) (z * 10);

                g.drawString("x = " + x, 0, 20, Graphics.TOP | Graphics.LEFT);
                g.drawString("y = " + y, 0, 40, Graphics.TOP | Graphics.LEFT);
                g.drawString("z = " + z, 0, 60, Graphics.TOP | Graphics.LEFT);
                if (pit) {
                    g.drawString("Kazkas atsitiko?! :o", 0, 100, Graphics.TOP | Graphics.LEFT);
                }

            } else if (type_int) {
//              g.drawString("INT x = " + x_int, 0, 20, Graphics.TOP | Graphics.LEFT);
//              g.drawString("INT y = " + y_int, 0, 40, Graphics.TOP | Graphics.LEFT);
//              g.drawString("INT z = " + z_int, 0, 60, Graphics.TOP | Graphics.LEFT);
            }
            /*
            if (z_int < 0) {
            g.setColor(255, 0, 0);  // Red
            int diameter = z_int;
            g.drawArc(width / 2 - diameter / 2, height / 2 - diameter / 2, diameter, diameter, 0, 360);
            } else if (z_int > 0) {
            g.setColor(0, 0, 255);  // Blue
            int diameter = z_int;
            g.drawArc(width / 2 - diameter / 2, height / 2 - diameter / 2, diameter, diameter, 0, 360);
            }
            if (x_int < 0) {
            for (int i = x_int; i < 0; i = i + 10) {
            g.setColor(255, Math.abs(i), 0); // Red <-> Yellow
            g.fillArc(xcoord + i, ycoord, size, size, 0, 360);
            }
            } else if (x_int > 0) {
            for (int i = 0; i < x_int; i = i + 10) {
            g.setColor(255, i, 0); // Red <-> Yellow
            g.fillArc(xcoord + i, ycoord, size, size, 0, 360);
            }
            }
            if (y_int < 0) {
```

```
        for (int i = y_int; i < 0; i = i + 10) {
        g.setColor(255, Math.abs(i), 0); // Red <-> Yellow
        g.fillArc(xcoord, ycoord - i, size, size, 0, 360);
        }
        } else if (y_int > 0) {
        for (int i = 0; i < y_int; i = i + 10) {
        g.setColor(255, i, 0); // Red <-> Yellow
        g.fillArc(xcoord, ycoord - i, size, size, 0, 360);
        }
        }
         */
    }

    protected void sizeChanged(int w, int h) {
        xcoord = w / 2 - r;
        ycoord = h / 2 - r;
    }

    protected void keyPressed(int keyCode) {
    }

    protected void keyReleased(int keyCode) {
    }

    protected void keyRepeated(int keyCode) {
    }

    protected void pointerDragged(int x, int y) {
    }

    protected void pointerPressed(int x, int y) {
    }

    protected void pointerReleased(int x, int y) {
    }

    public void commandAction(Command c, Displayable d) {

        if (c == startOnline) {
            try {
                client = (SocketConnection) Connector.open("socket://88.223.35.49:" + 4444);
                os = client.openOutputStream();
                Criteria cr = new Criteria();
                cr.setHorizontalAccuracy(500);

                LocationProvider lp = LocationProvider.getInstance(cr);
                lp.setLocationListener((LocationListener) this, 1, 1, 1);
            } catch (Exception e) {
                System.out.println(e.toString());
            }
            offline = false;
            //          this.removeCommand(startOnline);
            //          this.removeCommand(startDoubleCommandOffline);
            start();
        }
        if (c == startOffline) {
            try {
                //  connection = (OutputConnection) Connector.open("file:///root1/" + getDate("") + ".txt",
Connector.READ_WRITE);
```

```
            connection = (OutputConnection) Connector.open("file:///Music/" + getDate("") + ".txt",
Connector.READ_WRITE);
            javax.microedition.io.file.FileConnection fc = (javax.microedition.io.file.FileConnection) connection;
            if (!fc.exists()) {
               fc.create();
            }
            out = connection.openOutputStream();
            output = new PrintStream(out);
            output.println("Data; x; y; z; latitude, longtitude");
            Criteria cr = new Criteria();
            cr.setHorizontalAccuracy(500);
            LocationProvider lp = LocationProvider.getInstance(cr);
            lp.setLocationListener((LocationListener) this, 1, 1, 1);
         } catch (Exception e) {
         }
         offline = true;
         //       this.removeCommand(startOnline);
         //       this.removeCommand(startDoubleCommandOffline);
         start();
      }
      if (c == sendData) {
         sendData();
      }
      if (c == disconnect) {
         try {
            if (offline) {
               out.close();
               connection.close();
            } else {
               os.write("disconnect\n".getBytes());
               Thread.sleep(1000);
            }
         } catch (Exception e) {
         }
         stop();
         midlet.notifyDestroyed();
      }
   }

   public void dataReceived(SensorConnection sensor, Data[] data, boolean isDataLost) {
      String endLine = "";
      if (!offline) {
         endLine = "\n";
      }
      if (!type_int) {
         double[] directions = getDirections(data);
         x = directions[0];
         y = directions[1];
         z = directions[2];

         listX.addElement(getDate(":") + ";" + Double.toString(x));
         listY.addElement(Double.toString(y));
         listZ.addElement(Double.toString(z));
         //      output.println(Double.parseDouble(listZ.elementAt(listZ.size()-5).toString()));
         if ((Double.parseDouble(listZ.elementAt(listZ.size() - 15).toString()) < -12.5) ||
               (Double.parseDouble(listZ.elementAt(listZ.size() - 15).toString()) > -7.0)) {
            pit = true;
            counter = 0;
```

```
            directions = getDirections(data);


            for (int i = listZ.size() - 30; i < listZ.size(); i++) {
               listAllAxes.addElement(listX.elementAt(i) + ";" + listY.elementAt(i) + ";" + listZ.elementAt(i) + ";" + lat +
";" + lon + endLine);
            }
            listX.removeAllElements();
            listY.removeAllElements();
            listZ.removeAllElements();
            refreshList();

         }
      }

      if (listX.size() > 10000) {
         listX.removeAllElements();
         listY.removeAllElements();
         listZ.removeAllElements();
         refreshList();
      }
      counter++;
      if (counter > 10) {
         counter = 0;
         pit = false;
      }
      repaint();
   }

   private static double[] getDirections(Data[] data) {
      double[][] doubleValues = new double[3][BUFFER_SIZE];
      double[] directions = new double[3];
      for (int i = 0; i < 3; i++) {
         doubleValues[i] = data[i].getDoubleValues();
         double temp = 0;
         for (int j = 0; j < BUFFER_SIZE; j++) {
            temp = temp + doubleValues[i][j];
         }
         directions[i] = temp / BUFFER_SIZE;
      }
      return directions;
   }

   private synchronized void setStopped(boolean stopped) {
      isStopped = stopped;
      notify();
      if (thread != null) {
         thread = null;
      }
   }

   synchronized void start() {

      setStopped(false);
      if (thread == null) {
         thread = new Thread(this);
      }
      thread.start();
```

```
    }

    synchronized void stop() {
        setStopped(true);
        thread = null;
    }

    public void run() {
        initSensor();
    }

    public void sendData() {
        if (offline) {
            for (int i = 0; i < listAllAxes.size(); i++) {
                output.println(listAllAxes.elementAt(i).toString());
            }
        } else {
            try {
                os.write(new String("Array " + listAllAxes.size() + "\n").getBytes());
            } catch (IOException ex) {
            }
            for (int i = 0; i < listAllAxes.size(); i++) {
                try {
                    os.write(listAllAxes.elementAt(i).toString().getBytes());
                } catch (Exception e) {
                }
            }
            listAllAxes.removeAllElements();
        }
    }

    public void refreshList() {
        for (int i = 0; i < 30; i++) {
            listX.addElement(getDate(":") + ";" + Double.toString(0.0));
            listY.addElement(Double.toString(0.0));
            listZ.addElement(Double.toString(-9.8));
        }
    }

    public String getDate(String delimeter) {
        Date dd = new Date();
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(dd);
        String strdate = calendar.get(Calendar.HOUR_OF_DAY) + delimeter + calendar.get(Calendar.MINUTE) +
delimeter + calendar.get(Calendar.SECOND) + delimeter + calendar.get(Calendar.MILLISECOND);
        return strdate;
    }

    public void locationUpdated(LocationProvider provider, Location location) {

        GPSInfo gpsInfo = new GPSInfo(location);
        Thread gpsInfoThread = new Thread(gpsInfo);
        gpsInfoThread.start();
    }

    public void providerStateChanged(LocationProvider provider, int newState) {
    }

    class GPSInfo implements Runnable {
```

```
        Location location;

        GPSInfo(Location location) {
            this.location = location;
        }

        public void run() {

            try {
                Coordinates c = location.getQualifiedCoordinates();
                if (c != null) {
                    lat = c.getLatitude();
                    lon = c.getLongitude();
                } else {
                    System.out.println("\nnull coordinate");
                }

            } catch (Exception e) {
                System.out.println("\nexception");

            }
        }
    }
}
```