UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Nagelj

Spletni sistem za usmerjanje po cestnem omrežju z video posnetki visoke ločljivosti

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Uroš Lotrič

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01714/2010
Datum: 05.10.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MIHA NAGELJ**

Naslov: **SPLETNI SISTEM ZA USMERJANJE PO CESTNEM OMREŽJU Z VIDEO POSNETKI VISOKE LOČLJIVOSTI**

ONLINE ROAD NETWORK ROUTING SYSTEM WITH HIGH DEFINITION VIDEO

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V nalogi izdelajte programsko premo s spletnim vmesnikom za usmerjanje po slovenskem cestnem omrežju. Ta naj uporabniku poišče najugodnejšo pot med izbranima krajema iz zanjo iz geokodiranih posnetkov cest sestavi video posnetek v visoki ločljivosti.

Mentor:
prof. dr. Uroš Lotrič

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Original izdane teme

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Miha Nagelj,

z vpisno številko 63050062,

sem avtor diplomskega dela z naslovom:

Spletni sistem za usmerjanje po cestnem omrežju z video posnetki visoke ločljivosti

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  prof. dr. Uroša Lotriča

- so elektronska oblika diplomskega dela, naslov, povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 14. 04. 2011 Podpis avtorja:

Zahvala

Iskreno se zahvaljujem mentorju prof. dr. Urošu Lotriču, za pomoč, strokovne nasvete in usmerjanje pri izdelavi diplomskega dela.

Zahvaljujem se podjetju LUZ, d.d., kjer sem imel v času študija priložnost sodelovati na številnih zanimivih projektih. V vseh letih sodelovanja sem pridobil ogromno izkušenj, ki so pripomogle pri izdelavi diplomskega dela.

Za pripravljenost za sodelovanje pri izdelavi diplomskega dela, se zahvaljujem podjetju DFG consulting, d.o.o., v katerem so razvili Mobilni kartirni sistem.

Zahvaljujem se tudi družini in Nataši za podporo, spodbudo in navdih v času študija.

Kazalo

Povzetek

Nove tehnologije omogočajo zajem in pregledovanje digitalnih video posnetkov cest skupaj z geolokacijskimi podatki. Uporabljajo jih različne službe; predvsem za pregledovanje stanja cestišča, signalizacije in drugih objektov cestne infrastrukture. Vse to lahko izvajajo brez obiska terena. Posnetke cestnih odsekov pa lahko uporabimo tudi za navigacijo po cestnem omrežju.

V okviru diplomskega dela sem najprej razvil programsko opremo, s katero izvorne video posnetke cest lahko pretvorimo v obliko, primerno za objavo na spletu. Iz dobljenih posnetkov in podatkov o cestnem omrežju nato spletna storitev generira potrebne podatkovne strukture in poišče najkrajšo pot med vozlišči omrežja.

Vse dele rešitve povezuje bogata internetna aplikacija, ki vsebuje interaktivni zemljevid, modul za iskanje najkrajše poti med vnešenimi točkami in pregledovalnik posnetkov cestnih odsekov. Uporabnik si lahko s spletnim brskalnikom načrtuje pot z več postanki, pri tem pa mu je poleg klasičnega izrisa poti na zemljevidu, na voljo tudi realen pogled na cestišče s pomočjo posnetkov visoke ločljivosti. Poleg samega ogleda okolice, lahko uporabnik dinamično meri prečno dolžino elementov cestišča ali naravnih ovir, ki se nahajajo na načrtovani poti. Aplikacija zagotavlja vse podatke, ki jih uporabnik potrebuje pri načrtovanju poti in je namenjena predvsem načrtovanju izrednih prevozov.

Ključne besede:

Usmerjanje, geografski informacijski sistem, spletno predvajanje video posnetkov.

Abstract

New technology enables the capture of road digital video recordings and associated spatial data. Together, they are used by various departments in particular to review the state of the road, traffic signs and other road infrastructure facilities. All this can be performed without having to visit the field. Recordings of road sections can also be used for purposes of navigation over the road network.

In my thesis, I first developed software to convert source video recordings of roads to the form suitable for web publishing. Resulting video clips and spatial data of the road network are then used by web service which generates data structures necessary for finding the shortest path between nodes in the network. This information is used by the routing module, which implements an effective and fast algorithm to find the shortest path on demand.

All parts of solution are brought together by the rich internet application. It contains an interactive map, module for the shortest path calculation and high definition video player of road sections. The user can plan a route with multiple stops using interactive map in web browser and in addition to conventional display of route on the map, the user is provided with a realistic view of the road. User can observe the surroundings of the route and can also dynamically measure the transverse length of the objects or natural barriers on the road. Application provides all the information needed for route planning and is intended primarily for planning of heavy transport.

Keywords:

Routing, Geographic Information System, Video streaming.

1 Uvod

Geografski informacijski sistem (GIS) je računalniško voden informacijski sistem, ki se osredotoča na prostorske podatke. Področij znotraj GIS je zelo veliko in zajemajo tako prikazovanje, urejanje, analiziranje, zajemanje in upravljanje prostorskih podatkov kot tudi povezovanje prostorskih podatkov z neprostorskimi. Programska oprema, ki sem jo razvil v okviru diplomskega dela, predstavlja razširitev klasičnih operacij, ki nam jih nudi GIS.

V današnjem času je zaradi široke dostopnosti velik del razvoja osredotočen na spletne GIS. Hiter razvoj tehnologij svetovnega spleta nam omogoča, da uporabnikom lahko ponujamo vedno več funkcionalnosti, ki so bile včasih možne le v namiznih okoljih [8].

Uporabniki lahko s pomočjo spletnih aplikacij prostorske podatke prikazujejo, izpisujejo, urejajo, filtrirajo, vnašajo, po njih iščejo in delajo analize. Vse te operacije lahko izvajajo tudi na podatkih o cestni infrastrukturi. Cestna infrastruktura zajema ceste, prometne znake, talne oznake, ograje, semaforje, portale in druge objekte, ki se nahajajo v bližini ceste. Zajem podatkov o teh objektih, se že nekaj let opravlja s pomočjo geokodiranih video posnetkov cestnih odsekov.

Razpoložljivost podatkov v obliki video posnetkov, skupaj z razcvetom multimedijskih vsebin na svetovnem spletu, je pripeljala do razmišljanj o širitvi uporabe posnetkov tudi na uporabnike spletnih GIS. Objava geokodiranih video posnetkov cest na splet pa zahteva določeno število faz, o katerih bom govoril v naslednjih poglavjih.

V drugem poglavju sem opisal sistem in postopek za snemanje cest. Obrazložil sem podatkovni model, ki se uporablja za povezavo video posnetkov s prostorom. Na kratko so obrazloženi tudi standardni formati in kodeki za zapis video posnetkov.

Tretje poglavje je namenjeno opisu najbolj tipičnih protokolov za prenos multimedijskih vsebin. Opisal sem najbolj razširjene formate za spletni prikaz video posnetkov ter

delovanje strežnika IIS z razširitvijo Media Services, ki te posnetke streže lahkim odjemalcem.

Posnetki, zajeti s sistemom za snemanje cest, niso neposredno primerni za objavo na multimedijskih strežnikih. V četrtem poglavju je opisan postopek in aplikacija za pripravo video posnetkov ter spremljevalnih podatkov v obliko primerno za enostavno uporabo in povezavo z prostorskimi elementi (linije cestnih odsekov). Potrebno jih je tudi kodirati (pretvoriti) v format, ki omogoča učinkovit prenos prek omrežja ter upošteva različne zmogljivosti povezav.

Cilj diplomskega dela je izdelava sistema za video navigacijo, kar pomeni, da mora biti zmožen reševanja problema najkrajše (oz. najcenejše) poti. V petem poglavju sem definiral podatkovne strukture, namenjene predstavitvi cestnega omrežja kot graf vozlišč in povezav med njimi. Predstavil sem algoritme za iskanje najkrajše poti ter argumentiral izbiro algoritma A*.

Šesto poglavje diplomskega dela povzema razvoj spletnega predvajalnika geokodiranih video posnetkov. Predstavil sem aplikacijsko ogrodje Silverlight, ki je namenjeno razvoju naprednih internetnih aplikacij. Razdelal sem tako strežniški del aplikacije, ki služi kot spletna storitev (ang. web service) za lokacijske podatke, kot tudi uporabniški vmesnik aplikacije.

1.1 Uporabljena orodja in knjižnice

Vse module rešitve sem implementiral na Microsoftovi platformi .NET. Platforma zagotavlja izvajalno okolje na lahkih odjemalcih, namiznem okolju in spletnih strežnikih. Programska koda je napisana v programskem jeziku C# in razvita z orodjem Visual Studio 2010.

Celotna rešitev je zasnovana kot nadgradnja sistema Nukleus, ki ga razvijamo v podjetju LUZ, d.d. Sistem je zasnovan kot modularna platforma za razvoj naprednih in raznolikih GIS-rešitev. Ena njegovih najpomembnejših značilnosti je abstrakcija podatkovnega nivoja. Zagotavlja nam namreč univerzalen vmesnik za dostop do geografskih podatkov ne glede na to, kakšna podatkovna baza se uporablja v ozadju. Geografski podatki so tako lahko shranjeni v datotekah ESRI-shape, različnih relacijskih bazah ali v poljubni kombinaciji. Zato se v diplomskem delu ne spuščam v podrobnosti podatkovnega nivoja za prostorske podatke.

2 Sistem za snemanje cest

2.1 Mobilni kartirni sistem

Sistem za snemanje cest predstavlja veliko več kot le sam zajem posnetkov cest, zato se imenuje Mobilni kartirni sistem [2].

Mobilni kartirni sistem predstavlja informacijsko tehnologijo, ki je doživela svoj razcvet z napredkom v kinematični tehnologiji določanja položaja (ang. Global Positioning System, GPS), modernih komunikacijskih in prostorsko-informacijskih tehnologijah (GIS).

Z integracijo naštetih tehnologij v enoten sistem, lahko z mobilnim kartirnim sistemom pridobivamo podatke v realnem času, kartiramo objekte ter jih prostorsko vizualiziramo. Takšni sistemi se uporabljajo na različnih področjih za pridobivanje metričnih podatkov o prometni infrastrukturi (ceste, železnice, kolesarske poti...), v inteligentnih transportnih sistemih in kot pomoč pri prostorskem načrtovanju.

Sestavni del mobilnega kartirnega sistema je programski paket, ki omogoča učinkovit in hiter zajem ter obdelavo podatkov. V mobilni kartirni sistem so integrirani navigacijski senzorji (GPS, inercialna merilna enota IMU, odometer) in senzorji za kartiranje (video kamere).

Rezultat video in GPS/IMU snemanja prometne infrastrukture je video posnetek umeščen v državni koordinatni sistem. Na osnovi le-tega, s fotogrametričnim zajemom pridobimo metrične in opisne podatke, ki so osnova za vzpostavitev različnih katastrov prometne opreme.

2.2 Format in kodek video zapisa

Način zapisa video posnetkov na podatkovni medij je določen z vsebnikom (ang. container) in kodekom.

2.2.1 Format video datoteke

Format video posnetkov opisuje obliko in način zapisa podatkov, ki jih hranimo oz. prenašamo med napravami. Definira ga vsebnik, ki določa pravila za zapis kodiranih podatkov slike, zvoka in meta podatkov v datoteko na disku. Vsebnik omogoča, da programi na standarden način prepoznajo tipe podatkov v datoteki. Napredni formati vsebnikov poleg zapisa videa in zvoka podpirajo tudi zapis podnapisov, poglavij in drugih meta-podatkov. Vsebniki vsebujejo tudi podatke o časovni sinhronizaciji vseh naštetih podatkovnih virov.

Danes je v uporabi kar nekaj različnih formatov vsebnikov. Tabela 1 prikazuje nekaj največkrat uporabljenih formatov skupaj s primerjavo zmogljivosti.

	AVI	MP4	OGG	MKV	ASF
Lastnik	Microsoft	MPEG	Xiph.org	Matroska.org	Microsoft
Podpora za B-sličice	s prilagoditvami	Da	Da	Da	Da
VBR avdio	Da	Da	Da	Da	Da
Prilagodljivo št. sličic/s	Ne	Da	Ne	Da	Ne
Poglavja	Ne	Ne	Da	Da	Ne
Podnapisi	Ne	Da	Da	Da	Ne
Pretočno nalaganje	Ne	Da	Da	Da	Da

Tabela 1. Najpogosteje uporabljeni formati vsebnikov [4].

Mobilni kartirni sistem uporablja vsebnik AVI. Iz tabele je sicer razvidno, da po možnostih zaostaja za ostalimi (novejšimi) formati. Razlog je v starosti formata AVI, saj je bil zasnovan že leta 1992. Starost pa v tem primeru prinese tudi dobro lastnost, saj je format močno razširjen in zato tudi podprt v večini programov, ki se ukvarjajo z video posnetki.

2.2.2 Kodek

Kodek (ang. codec) je program namenjen kodiranju in/ali dekodiranju (v večini primerov tudi kompresiji in/ali dekompresiji) digitalnega toka podatkov. Kodiranje potrebujemo, ko želimo tok podatkov ali signal, pretvoriti v obliko primerno za prenos ali shranjevanje. Nasprotno pri dekodiranju podatke pretvorimo v obliko, ki je primerna za prikaz na zaslonu ali urejanje.

Najpomembnejša naloga kodekov je kompresija. Video posnetki zaradi svoje narave (zaporedje podobnih sličic) v osnovni obliki vsebujejo veliko podvojenih podatkov. Kodeki uporabljajo algoritme, ki te podobnosti zaznajo in jih zapišejo z manj podatki. Večina kodekov uporablja izgubno kompresijo, kar pomeni, da se del informacij med kodiranjem izgubi. Obstajajo tudi brezizgubni kodeki, vendar v večini primerov povečanje kvalitete na račun brezizgubne kompresije ne odtehta povečanja količine podatkov.

Algoritmi, ki jih za kompresijo uporabljajo kodeki, kot vhod sprejmejo različne parametre kot so:

- podatkovna širina izhodnih podatkov (ang. bitrate),
- število sličic na sekundo (ang. framerate),
- razmik med ključnimi sličicami (ang. keyframe),
- razmerje med ostrino in gladkostjo posnetka.

Ti parametri določajo kvaliteto izhodnega video posnetka in hkrati tudi omejujejo količino podatkov, ki so potrebni za zapis posnetka.

2.3 Zajem video posnetkov

Zajem oziroma snemanje video posnetkov poteka s snemalnim avtomobilom, ki se vozi po cestah določenega območja. Na njegovi strehi sta nameščeni dve kameri, usmerjeni naprej, ki snemata cestišče in objekte ob njem. Razlog za uporabo dveh kamer je v možnosti izdelave stereo posnetka. Ta nam omogoči določitev globine oz. oddaljenosti objektov od kamer in posledično tudi geokodiranje (določitev lokacije v prostoru) objektov. Za spletni predvajalnik stereo posnetek ni potreben, zato sem pri implementaciji rešitve uporabil samo posnetke ene kamere.

Kameri snemata v visoki ločljivosti 1920x1080 pik. Medijski tok pošiljata direktno na zunanji trdi disk, ki skrbi za segmentacijo videa v odlomke (ang. clip) ter ustrezno

strukturo in poimenovanje datotek. Segmentacija je potrebna zaradi omejitev velikosti datotek datotečnega sistema na disku ter lažje obvladljivosti datotek.

Vsak odlomek posnetka je predstavljen z eno datoteko AVI, ki vsebuje okoli 11 minut snemanja (do razlik prihaja, ker je posamezen odlomek omejen z velikostjo podatkov ne pa s časom). Video je znotraj posameznega odlomka zakodiran s kodekom Divx. Namen teh izvornih video posnetkov je predvsem prikazati čim boljšo kvaliteto slike, zato je uporabljena kompresija nizka. Medijski tok ima tako razmeroma visoko podatkovno širino, ki znaša 25 Mbit/s.

3 Objava in prenos multimedijskih vsebin

Poskusi prikazovanja multimedijskih vsebin na računalniških zaslonih segajo že v same začetke računalništva v sredini dvajsetega stoletja. Kljub temu pa je širša uporaba morala počakati nekaj desetletij, predvsem zaradi visoke cene in omejenih sposobnosti računalniške opreme.

Ob koncu 80. prejšnjega stoletja in skozi 90. so osebni računalniki postali dovolj zmogljivi, da so lahko prikazovali multimedijsko vsebino. Ta je postala izredno popularna in se hitro razširila med uporabniki, ki so bili navdušeni nad tem, da si lahko kar na računalniku ogledajo najnovejši film.

Izkušnja uporabnikov pa samo z možnostjo ogleda ni bila popolna. Izmenjava multimedijskih datotek je bila zaradi precejšnje količine podatkov možna samo prek zunanjih medijev, kar je bilo precej nepraktično in počasno. S prihodom interneta se je izmenjava datotek preselila na medmrežje.

Ideja ogleda multimedijskih vsebin na daljavo je v času začetkov interneta zvenela zelo futuristično. Internet je takrat predstavljal neko široko zbirko bolj ali manj statičnega besedila, tu in tam opremljenega s slikami. Pasovne širine povezav uporabnikov so bile nizke in razmeroma drage. Seveda pa se je internet od njegovih začetkov zelo hitro razvijal in s tem omogočil prenose veliko večje količine podatkov v kratkem času.

Danes je ogled video vsebin prek spleta postal povsem samoumeven in ga dnevno uporablja na milijone uporabnikov. Vse to so omogočile napredne tehnologije, ki so se razvile skupaj z rastjo spleta, vendar pa se najprej posvetimo dvema osnovnima principoma za prenos multimedijskih vsebin.

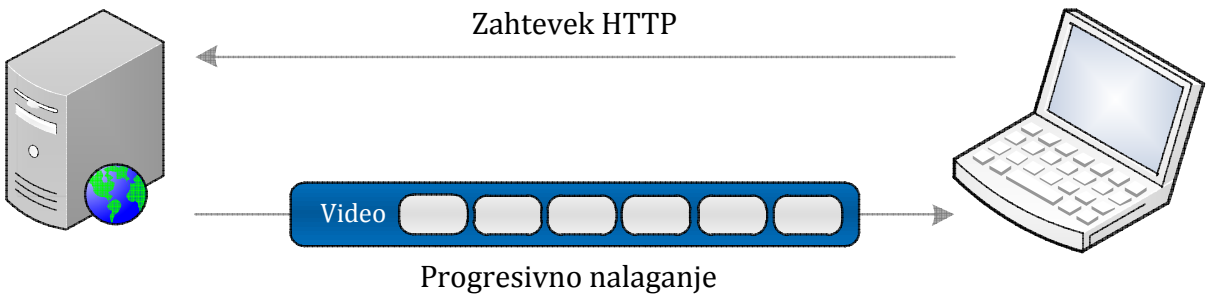
3.1 Progresivno nalaganje

Progresivno nalaganje (ang. progressive download) se je razvilo iz klasičnega protokola za nalaganje datotek iz spleta (ang. Hyper Text Transfer Protocol, HTTP). Izraz »progresivno« pomeni, da se predvajanje datoteke prične med prenosom datoteke, že preden je le-ta v celoti prenesena na uporabnikov računalnik.

Predvajalnik je zmožen takšnega načina nalaganja zaradi zaporedne organizacije datoteke in meta podatkov, ki se nahajajo v glavi datoteke. Naložijo se že na začetku prenosa in iz njih predvajalnik razbere tip datoteke in način organizacije podatkov. Za tem se začne prenos izseka video podatkov, ki ga imenujemo medpomnilnik (ang. buffer). Predvajanje posnetka se lahko prične že ko je prenesen samo ta izsek, dolg tipično 1 do 10s. Preostanek podatkov se naloži med predvajanjem in uporabnik tako lahko z ogledom prične veliko prej, kot če bi moral čakati, da se datoteka prenese v celoti.

Danes večinoma vsi odjemalci med progresivnim nalaganjem omogočajo tudi skoke na poljuben čas posnetka, tudi če podatki za zahtevan čas še niso bili preneseni. To omogoča protokol HTTP 1.1 pri katerem so možni zahtevki za prenos podatkov od nekega bajta naprej (ang. byte range). Kljub tej sposobnosti, pa se prenos ob zaustavitvi predvajanja ne ustavi in tako se prenese več podatkov, kot jih odjemalec dejansko potrebuje.

Iz omrežnega stališča je prenos video datoteke identičen prenosu datoteke katerega koli drugega tipa po protokolu HTTP. Na strani strežnika HTTP v tem primeru ne potrebujemo nobene dodatne programske opreme. Na sliki 1 je shematski prikaz poteka prenosa v primeru progresivnega nalaganja video posnetkov.

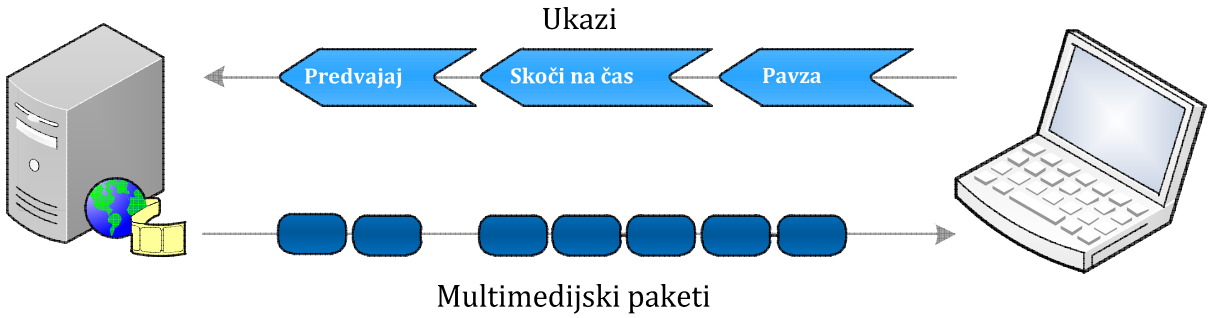


Slika 1. Progresivno nalaganje multimedijskih datotek.

3.2 Pretočni prenos multimedijskih datotek

Drugi osnovni princip nalaganja multimedijskih datotek se imenuje pretočno nalaganje (ang. streaming). Za razliko od progresivnega nalaganja se pri pretočnem ponavadi uporablja transportni protokol, ki je namenjen samo za multimedijske datoteke. Za uporabo pretočnega nalaganja zato potrebujemo posebno programsko opremo tako na strežniški, kot odjemalčevi strani.

Dober primer protokola za pretočno nalaganje multimedijskih datotek je protokol RTSP (ang. Real-Time Streaming Protocol). Protokol je izdelalo podjetje Microsoft in se uporablja v seriji produktov Windows Media Services. Definiran je kot protokol s stanjem (ang. stateful), kar pomeni da se od trenutka, ko se odjemalec poveže na pretočni strežnik, do takrat, ko se povezava konča, pretočni strežnik zaveda v kakšnem stanju je odjemalec. Poleg tega da odjemalec sprejema multimedijske podatke, mora tako tudi strežniku pošiljati ukaze kot so PLAY (predvajaj), PAUSE (pavza) ali TEARDOWN (ruši povezavo). Tak način komunikacije med pretočnim strežnikom in odjemalcem prikazuje slika 2.



Slika 2. Pretočni prenos multimedijskih datotek.

Po vzpostavitvi seje med odjemalcem in strežnikom, strežnik začne pošiljati multimedijske podatke kot enakomeren tok majhnih paketov. Tipična velikost paketa je 1452 bajtov, kar pomeni, da pri video posnetku z bitno hitrostjo (ang. bitrate) 1 Mbit/s, vsak paket nosi približno 11 milisekund video posnetka. Ti paketi se ponavadi zaradi večje hitrosti prenašajo po protokolu UDP (ang. User Datagram Protocol), ki ne vsebuje potrjevanja in ponavljanja.

Prednost pretočnega prenosa pred klasičnim nalaganjem je predvsem v tem, da je prenos sinhroniziran s predvajanjem multimedijske datoteke. To pomeni, da če ena sekunda video posnetka vsebuje en megabit podatkov, se bo v času predvajanja ene sekunde tega posnetka preneslo tudi približno toliko podatkov. Pri ogledu samo dela zelo dolgega posnetka, ta lastnost prinese precejšen prihranek pri porabi omrežnih

virov. S takšnim načinom prenosa se je izboljšala tudi uporabniška izkušnja pri skokih, saj so ti veliko hitrejši. Poleg dodatne učinkovitosti, se je s pretočnimi prenosi pojavila tudi možnost mrežnega predvajanja multimedijskih tokov, ki nastajajo iz izvorov v realnem času.

Zaradi namenskega protokola se pri pretočnih prenosih pojavijo tudi določene slabosti. Zaradi beleženja stanja je onemogočeno predpomnenje paketov v distribucijski mreži, kar zelo zmanjša skalabilnost. Poleg tega požarni zidovi privzeto večinoma blokirajo promet, ki poteka po nestandardnih protokolih in vratih, kar pomeni, da številni uporabniki ne morejo dostopati do multimedijske vsebine.

3.3 Adaptivno pretočno nalaganje

Adaptivno pretočno nalaganje je med tremi obravnavanimi najnaprednejši način za distribucijo multimedijskih datotek prek internetnega omrežja. Danes obstaja veliko programskih paketov, ki implementirajo adaptivno pretočno nalaganje. Na Microsoftovi platformi se tehnologija imenuje Smooth Streaming. Predstavlja hibrid osnovnih dveh konceptov, saj deluje kot pretočno nalaganje, v osnovi pa temelji na konceptu progresivnega nalaganja. Za isti posnetek omogoča tudi strežbo datotek z različno bitno hitrostjo, glede na zmogljivost odjemalca in pasovno širino omrežne povezave. Tehnologijo adaptivnega pretočnega nalaganja sestavljajo tri glavne komponente: format multimedijskih datotek, namenski strežnik in predvajalnik.

3.3.1 Format datoteke Smooth Streaming

Smooth Streaming format za vsak posnetek potrebuje več datotek na disku:

- strežniška datoteka manifest s končnico ism,
- odjemalčeva datoteka manifest s končnico ismc,
- multimedijske datoteke s končnico ismv, ena za vsako bitno hitrost.

Strežniška datoteka manifest

Strežniška datoteka manifest (datoteka ism) služi kot kazalo ostalih datotek in vsebuje lokacijo odjemalčeve datoteke manifest ter spisek bitnih hitrosti, ki so za posnetek na voljo. Sestavljena je iz teksta XML (ang. eXtensible Markup Language). Primer vsebine datoteke ism za posnetek z imenom 0221B prikazuje slika 3.

```
<?xml version="1.0" encoding="utf-16"?>
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <meta name="clientManifestRelativePath" content="0221B.ismc" />
  </head>
  <body>
    <switch>
      <video src="0221B_2000.ismv" systemBitrate="2000000">
        <param name="trackID" value="1" valuetype="data" />
        <param name="trackName" value="video" valuetype="data" />
        <param name="timeScale" value="10000000" valuetype="data" />
      </video>
      <video src="0221B_1190.ismv" systemBitrate="1190000">
        <param name="trackID" value="1" valuetype="data" />
        <param name="trackName" value="video" valuetype="data" />
        <param name="timeScale" value="10000000" valuetype="data" />
      </video>
      <video src="0221B_800.ismv" systemBitrate="800000">
        <param name="trackID" value="1" valuetype="data" />
        <param name="trackName" value="video" valuetype="data" />
        <param name="timeScale" value="10000000" valuetype="data" />
      </video>
      <video src="0221B_440.ismv" systemBitrate="440000">
        <param name="trackID" value="1" valuetype="data" />
        <param name="trackName" value="video" valuetype="data" />
        <param name="timeScale" value="10000000" valuetype="data" />
      </video>
    </switch>
  </body>
</smil>
```

Slika 3. Primer vsebine datoteke ism.

Odjemalčeva datoteka manifest

Kot pove že ime, je odjemalčeva datoteka manifest (datoteka ismc), namenjena samo odjemalcu oz. predvajalniku multimedijskih datotek. Za razliko od datoteke ism, v datoteki ismc ni imen datotek, ampak so posamezne bitne hitrosti opisane vsebinsko. Slika 4 prikazuje primer datoteke ismc.

```
<?xml version="1.0" encoding="utf-16"?>
<SmoothStreamingMedia MajorVersion="2" MinorVersion="1" Duration="8890000000">
  <StreamIndex Type="video" Name="video" Chunks="224" QualityLevels="4"
               MaxWidth="1280" MaxHeight="720" DisplayWidth="1280" DisplayHeight="720"
               Url="QualityLevels({bitrate})/Fragments(video={start time})">
<QualityLevel Index="0" Bitrate="2000000" FourCC="WVC1" MaxWidth="1280" MaxHeight="720"
/>
<QualityLevel Index="1" Bitrate="1190000" FourCC="WVC1" MaxWidth="960"  MaxHeight="540"
/>
<QualityLevel Index="2" Bitrate="800000"  FourCC="WVC1" MaxWidth="640"  MaxHeight="360"
/>
<QualityLevel Index="3" Bitrate="440000"  FourCC="WVC1" MaxWidth="428"  MaxHeight="240"
/>
    <c d="40000000">
      <f i="0" s="761" q="534" />
      <f i="1" s="481" q="337" />
      <f i="2" s="330" q="232" />
      <f i="3" s="218" q="153" />
    </c>
    <c d="40000000">
      <f i="0" s="886" q="555" />
      <f i="1" s="503" q="315" />
      <f i="2" s="368" q="231" />
      <f i="3" s="219" q="137" />
    </c>
    ...
    ...
    ...
  </StreamIndex>
</SmoothStreamingMedia>
```

Slika 4. Primer vsebine datoteke ismc.

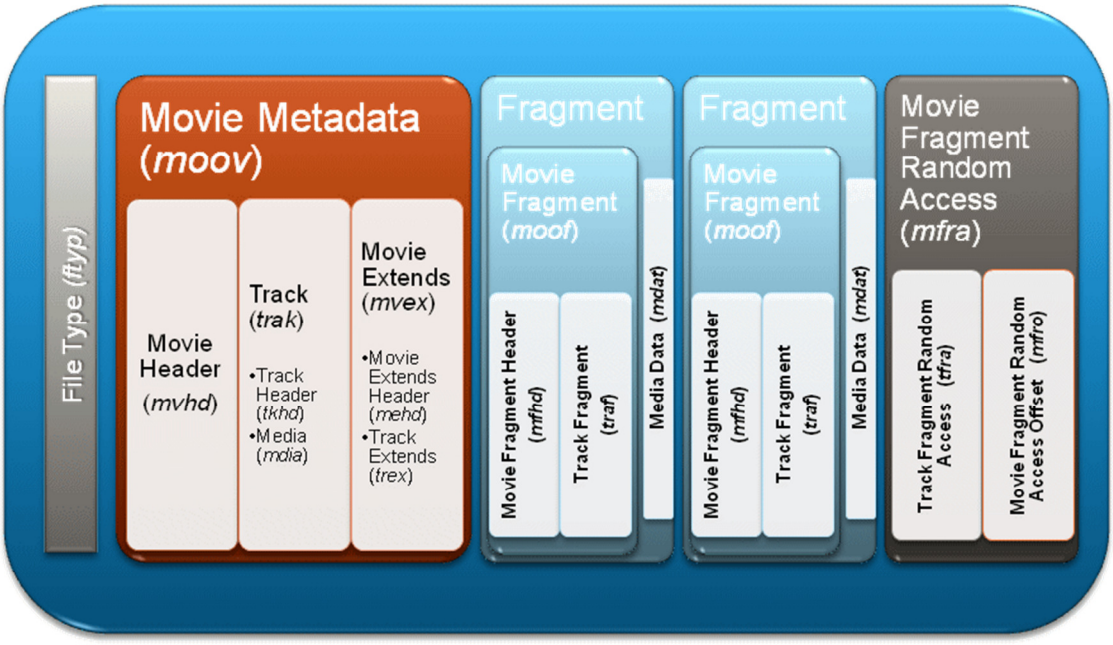
Datoteka v začetku vsebuje metapodatke, ki se nanašajo na posnetek, kot so dolžina posnetka, število fragmentov (ang. chunks), število bitnih hitrosti (ali kvalitet) in ločljivost posnetka. V nadaljevanju se nahajajo metapodatki za vsakega izmed nivojev kvalitet, ki so za posnetek na voljo.

Multimedijske datoteke

Definicijo formata Smooth Streaming multimedijske datoteke lahko razdelimo na dva nivoja. Prvi nivo je organizacija datoteke, ki se nahaja na strežniku, in vsebuje podatke celotnega video posnetka.

Klasične multimedijske datoteke so sestavljene iz dveh delov. Na začetku datoteke se nahajajo metapodatki, iz katerih lahko predvajalnik ugotovi informacije o multimedijskih podatkih, ki bodo sledili v drugem delu datoteke.

Pri adaptivnem pretočnem nalaganju se ta koncept nadgradi z razdelitvijo multimedijske datoteke na veliko majhnih fragmentov. Vsak fragment je razdeljen na metapodatke in multimedijske podatke ter s tem predstavlja zaključeno celoto, ki vsebuje vse potrebne podatke za predvajanje majhnega dela posnetka. Strukturo multimedijske datoteke Smooth Streaming prikazuje slika 5.



Slika 5. Struktura multimedijske datoteke Smooth Streaming [7].

Struktura sledi standardu »ISO/IEC 14496-12 ISO Base Media File«, ki je bolje poznan pod imenom »Fragmented MP4«. Predstavljamo si jo lahko kot vsebnik (ang. container) manjših zaključenih celot (fragmentov). Začne se z meta podatki, ki opisujejo datoteko, kot so število fragmentov, velikost fragmentov in informacije o kodekih (oznaka na sliki »moov«). Glavni del datoteke tako količinsko kot vsebinsko predstavljajo fragmenti. Na sliki sta prikazana samo dva, ampak v splošnem datoteka vsebuje veliko število fragmentov, saj privzeto en fragment vsebuje 2 sekundi posnetka. Datoteka se zaključi z indeksom fragmentov (oznaka »mfra«), ki služi za hitro lociranje pozicij fragmentov znotraj datoteke.

Odjemalec se celotne strukture datoteke ne zaveda in od strežnika zahteva samo posamezne fragmente. Drugi nivo formata tako predstavlja organizacijo posameznega fragmenta, ki si ga lahko predstavljamo kot kratek, zaključen del celotnega posnetka. Iz slike 5 lahko razberemo, da je posamezen fragment sestavljen iz metapodatkov (oznaka moof) in multimedijskih podatkov (oznaka mdat). Količinsko večino predstavljajo multimedijski podatki.

3.3.2 IIS Media Services

Druga komponenta tehnologije Smooth Streaming je strežnik multimedijskih datotek. Na voljo nam je kot brezplačna razširitev klasičnega spletnega strežnika na operacijskih sistemih windows, IIS7 (ang. Internet Information Services).

Strežnik odgovarja na zahteve odjemalca, ki med predvajanjem pošilja zahtevke za prenose fragmentov. Format zahtevkov je oblike REST (ang. Representational State Transfer), kar pomeni, da gre za enostavne GET HTTP zahtevke, brez beleženja stanja na strežniku (ang. stateless).

```
http://hostname/0221B.ism/QualityLevels(2000000)/Fragments(video=40000000)
```

Slika 6. Format zahtevka za prenos fragmenta.

Na zgornji sliki je prikazan primer zahtevka odjemalca za prenos enega fragmenta video posnetka. Parametri zahtevka so podčrtani in v naslednjem vrstnem redu:

- 0221B , ime posnetka,
- 2000000, zahtevana bitna hitrost v bitih na sekundo,
- 40000000, časovni odmik začetka fragmenta od začetka video posnetka (ena enota predstavlja 100ns).

Strežnik ob prejemu takega zahtevka izvede naslednje korake:

1. branje datoteke 0221B.ism iz katere strežnik razbere, da se video z bitno hitrostjo 2000000 bit/s nahaja v datoteki 0221B_2000.ismv;
2. branje indeksa fragmentov iz konca datoteke 0221B_2000.ismv (oznaka »mfra« na sliki 5) iz katerega strežnik razbere lokacijo zahtevanega fragmenta;
3. najdeni fragment iz datoteke ismv strežnik v nespremenjeni obliki zapiše v odziv (ang. response).

Postopek je zaradi učinkovite strukture datotek razmeroma enostaven in ne zahteva veliko procesnih virov. Protokol brez stanja nam omogoča tudi visoko skalabilnost, saj lahko uporabimo koncept predpomnenja. Odziv strežnika je za nek zahtevek vedno isti, zato lahko uporabljamo klasično infrastrukturo za predpomnenje običajnih zahtevkov HTTP. Strežniki za predpomnenje so danes zelo razširjeni in zato tudi cenovno veliko ugodnejši kot postavitev več multimedijskih strežnikov.

3.3.3 Odjemalec Smooth Streaming

Tretja komponenta tehnologije Smooth Streaming so programske knjižnice, ki nam pomagajo pri implementaciji odjemalca za predvajanje multimedijskih posnetkov iz strežnika IIS Media Services. Odjemalec ima v primerjavi s strežnikom večjo vlogo. Poleg samega predvajanja, mora skrbeti tudi za nalaganje multimedijskih podatkov in inteligentno izbiranje optimalne bitne hitrosti.

Potek ob začetku nalaganja posnetka lahko ponazorimo z dvema korakoma.

1. Na začetku se prenese odjemalčeva datoteka manifest (ismc) iz katere odjemalec razbere, s katerim kodekom je posnetek kodiran, katere bitne hitrosti so na voljo, ter druge parametre, ki jih potrebuje pri predvajanju in nalaganju.
2. Prvi fragment se naloži iz posnetka z najmanjšo bitno hitrostjo, zato da se doseže čim krajši čas nalaganja do prvega prikaza.

Fragmente odjemalec nalaga z zahtevki, ki so opisani v točki 3.3.2. Medtem vseskozi izvaja inteligentni algoritem, ki iz parametrov, kot so hitrost prenosa podatkov, velikost video okna in procesna zmogljivost računalnika, določa najprimernejšo bitno hitrost posnetka. Ti parametri se skozi čas spreminjajo, zato se med predvajanjem spreminja tudi izbira bitne hitrosti. Preklopi kvalitet potekajo med predvajanjem povsem brez zakasnitev in tako zagotavljajo bogato uporabniško izkušnjo.

4 Priprava video posnetkov

Video posnetki, ki jih pridobimo iz kamer, seveda v osnovi niso namenjeni hitri in široki distribuciji po internetnem omrežju, zato jih je za tak način uporabe potrebno pretvoriti v ustrezno obliko. Pred samo pretvorbo pa moramo seveda video posnetke snemalnih dni ustrezno urediti in razdeliti na smiselne enote. Analiza posnetkov, ki zvezni posnetek snemalnega dne razdeli na smiselne odseke, je opravljena takoj po končanem snemanju. Postopek bom opisal v naslednjem podpoglavju.

4.1 Analiza posnetega materiala

Snemanje posnetkov cest poteka po načrtu, ki je smiselno sestavljen tako, da se s stališča stroškov snemanja (čas, gorivo) čimbolj optimalno prevozi celotno omrežje cest. Absolutne optimalnosti se v večini primerov ne da doseči, kar pomeni, da se nekatere odseke cest prevozi (in s tem tudi posname) večkrat. Poleg tega se med snemalnim dnevom pojavijo zastoji, daljše ustavljanje zaradi semaforjev, ustavitev zaradi čiščenja umazanije z leče kamere ali operater preprosto potrebuje odmor med vožnjo. Kamera seveda med vsemi temi dogodki snema brez prestanka.

Končni snemalni posnetek dneva je tako zaporedje premešanih posnetkov odsekov cest, med katerimi se nahajajo tudi za samo bazo posnetkov cest neuporabni posnetki omenjenih izrednih dogodkov. Iz tega neurejenega razmeroma dolgega posnetka moramo izvleči smiselno urejene uporabne dele.

Za kakršen koli razrez moramo najprej povezati video posnetke z lokacijo v prostoru. Ta povezava je izvedena prek realnega časa. Tako kamera kot snemalnik GPS

trajektorije, namreč vseskozi beležita točen čas, zato jih prek tega podatka lahko natančno sinhroniziramo.

Razdelitev cestnega omrežja na odseke določa banka cestnih podatkov (BCP). V bazi BCP je vsaka os odseka predstavljena s prostorsko linijo. S prostorskimi relacijami lahko te linije povežemo s trajektorijami GPS in s tem prek realnega časa tudi z video posnetki.

Posnetke cestnega omrežja je najbolj smiselno organizirati po cestnih odsekih, tako kot je organizirana tudi baza BCP. Vzdolž enega cestnega odseka trajektorija ne poteka v enem kosu, ampak je ta lahko razdeljena na več delov. Trajektorijo (in s tem tudi pripadajoči video posnetek) moramo torej razrezati na točkah, kjer se začne prekrivati z odsekom in na točkah, kjer z opazovanega odseka odvije. Vsak zvezni del trajektorije med dvema takima točkama imenujemo kader.

Poleg klasičnih kartezičnih koordinat X in Y, se pri obravnavi cestnih odsekov uporablja tudi drugačen način za opis točke na liniji in sicer glede na stacionažo. Stacionaža neke točke na liniji predstavlja razdaljo vzdolž linije, od njenega začetka do te opazovane točke in se v BCP vodi v metrih.

Vsak kader je definiran z začetno in končno točko v realnem času snemanja. Prek tega časa lahko najdemo tako pozicijo v video posnetku kot tudi lokacijo v prostoru. Kader je seveda tudi direktno povezan s cestnim odsekom, zato ga lahko enolično opišemo z začetno in končno stacionažo. Prostorsko zaporedno sledeče kadre združimo v posnetek odseka v smeri ali v nasprotni smeri stacionaže.

4.2 Geolokacijske datoteke

Glavni rezultat analize posnetkov so torej točke, v katerih trajektorije ter s tem tudi posnetke, razdelimo na uporabne kadre. Za lažjo in bolj pregledno organiziranost posnetkov je potrebno v postopku analize dobljene točke, vključno z osmi odsekov in trajektorijami, zapakirati v trinivojsko strukturo datotek, ki omogoča učinkovito uporabo v aplikacijah za pregled posnetkov.

4.2.1 Datoteke VAL

Datoteke VAL se nahajajo na najnižjem nivoju in jih je od treh tipov največ. Vsaka datoteka opisuje en kader posnetka. Organizirana je kot tekstovna datoteka, v kateri

vsaka vrstica vsebuje z vejico ločene vrednosti (ang. Comma Separated Values, CSV). Vsak zapis je sestavljen iz podatkov o času, prostorski lokaciji, odseku in stacionaži. Frekvenca zapisov v datoteki je en zapis na sekundo. Slika 7 prikazuje primer prvih 20 zapisov datoteke VAL za odsek z oznako 0213. Prva vrstica je namenjena samo opisu posameznih polj in ni del datoteke VAL.

Zap. st	Oznaka ceste	Odsek	Smer	Datum	Ura	Posnetek	Sličica	X	Y	Z	Stacionaža
1,	8,	0213,	1,	20100520,	17:04:48.000,	12108,	39816,	458606.00,	106355.01,	316.46,	0.00
2,	8,	0213,	1,	20100520,	17:04:49.000,	12108,	39841,	458613.99,	106347.60,	316.44,	8.76
3,	8,	0213,	1,	20100520,	17:04:50.000,	12108,	39866,	458622.02,	106340.24,	316.47,	18.95
4,	8,	0213,	1,	20100520,	17:04:51.000,	12108,	39891,	458630.02,	106332.84,	316.46,	29.84
5,	8,	0213,	1,	20100520,	17:04:52.000,	12108,	39916,	458638.02,	106325.37,	316.43,	40.79
6,	8,	0213,	1,	20100520,	17:04:53.000,	12108,	39941,	458646.10,	106317.94,	316.39,	51.76
7,	8,	0213,	1,	20100520,	17:04:54.000,	12108,	39966,	458654.23,	106310.51,	316.35,	62.78
8,	8,	0213,	1,	20100520,	17:04:55.000,	12108,	39991,	458662.35,	106303.02,	316.32,	73.82
9,	8,	0213,	1,	20100520,	17:04:56.000,	12108,	40016,	458670.43,	106295.49,	316.27,	84.94
10,	8,	0213,	1,	20100520,	17:04:57.000,	12108,	40041,	458678.47,	106287.91,	316.23,	95.99
11,	8,	0213,	1,	20100520,	17:04:58.000,	12108,	40066,	458686.53,	106280.39,	316.21,	107.01
12,	8,	0213,	1,	20100520,	17:04:59.000,	12108,	40091,	458694.53,	106272.83,	316.15,	118.01
13,	8,	0213,	1,	20100520,	17:05:00.000,	12108,	40116,	458702.46,	106265.22,	316.12,	129.00
14,	8,	0213,	1,	20100520,	17:05:01.000,	12108,	40141,	458710.31,	106257.55,	316.07,	139.97
15,	8,	0213,	1,	20100520,	17:05:02.000,	12108,	40166,	458718.17,	106249.89,	316.02,	150.95
16,	8,	0213,	1,	20100520,	17:05:03.000,	12108,	40191,	458726.02,	106242.24,	315.97,	161.91
17,	8,	0213,	1,	20100520,	17:05:04.000,	12108,	40216,	458733.90,	106234.59,	315.91,	172.89
18,	8,	0213,	1,	20100520,	17:05:05.000,	12108,	40241,	458741.76,	106226.92,	315.88,	183.92
19,	8,	0213,	1,	20100520,	17:05:06.000,	12108,	40266,	458749.60,	106219.22,	315.83,	194.91
20,	8,	0213,	1,	20100520,	17:05:07.000,	12108,	40291,	458757.38,	106211.49,	315.82,	205.87
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.

Slika 7. Primer datoteke VAL.

4.2.2 Datoteke MVL

Vsaka datoteka MVL združuje več datotek VAL, ki opisujejo kadre istega odseka. Predstavlja vrstni red predvajanja kadrov za video posnetek odseka, v smeri naprej in nazaj. V datoteki MVL vsak zapis predstavlja en kader in vsebuje naslednje podatke o kadru:

- zaporedna številka v seznamu predvajanja,
- oznaka ceste,
- oznaka odseka,
- smer snemanja kadra,
- čas začetka kadra,
- čas konca kadra
- številka posnetka,
- začetna sličica,
- končna sličica,
- prostorski obseg kadra (minX, maxX, minY, maxY),
- stacionaža začetka,
- stacionaža konca,
- tip stacionaže.

4.2.3 Datoteka PAV

Datoteka PAV predstavlja naslednjo stopnjo generalizacije geolokacijskih podatkov. Predstavlja spisek vseh odsekov z video posnetkom, ter vsebuje tudi povzetke podatkov datotek MVL:

- zaporedna številka odseka,
- oznaka ceste,
- oznaka odseka,
- datum začetka snemanja odseka,
- datum konca snemanja odseka,
- prostorski obseg odseka (minX, maxX, minY, maxY),
- minimalna stacionaža odseka,
- maksimalna stacionaža odseka.

4.2.4 Datoteka Clips.dbf

Datoteke opisane v prejšnjih treh točkah so rezultat analize trajektorij GPS skupaj z osmi odsekov. Zaradi razreza posnetkov video kamer na odlomke, za uspešno povezavo potrebujemo tudi podatke o tem, kako so posnetki razrezani. Ti podatki so v datoteki Clips.dbf, ki vsebuje tabelo podatkov o odlomkih.

Filename	Cam	Run	Clip	FrmCnt	SFrm	EFrm	Date	STime	ETime
201003261110000	1	0	0	16521	0	16520	20100326	08:52:00.800	09:03:01.640
201003261110001	1	0	1	16524	16521	33044	20100326	09:03:01.640	09:14:02.600
201003261110002	1	0	2	16525	33045	49569	20100326	09:14:02.600	09:25:03.600
201003261110003	1	0	3	16524	49570	66093	20100326	09:25:03.600	09:36:04.560
201003261110004	1	0	4	16524	66094	82617	20100326	09:36:04.560	09:47:05.520
201003261110005	1	0	5	8668	82618	91285	20100326	09:47:05.520	09:52:52.240
201003261110100	1	1	0	16525	0	16524	20100326	09:55:27.560	10:06:28.560
201003261110101	1	1	1	16524	16525	33048	20100326	10:06:28.560	10:17:29.520
201003261110102	1	1	2	16524	33049	49572	20100326	10:17:29.520	10:28:30.480
201003261110103	1	1	3	16524	49573	66096	20100326	10:28:30.480	10:39:31.440
...	...	...	...	...	...	...	...	...	...

Tabela 2. Struktura tabele v datoteki Clips.dbf.

Pomen stolpcev v enakem vrstnem redu kot se pojavijo v tabeli:

- ime datoteke posnetka,
- številka kamere,
- številka posnetka v dnevu,
- številka odlomka,

- število sličic v odlomku,
- začetna sličica odlomka v posnetku,
- končna sličica odlomka v posnetku,
- datum snemanja,
- začetni čas odlomka,
- končni čas odlomka.

Datoteka vsebuje tudi druge metapodatke o odlomku, ki za povezovanje niso nujno potrebni in sem jih zaradi večje preglednosti izpustil.

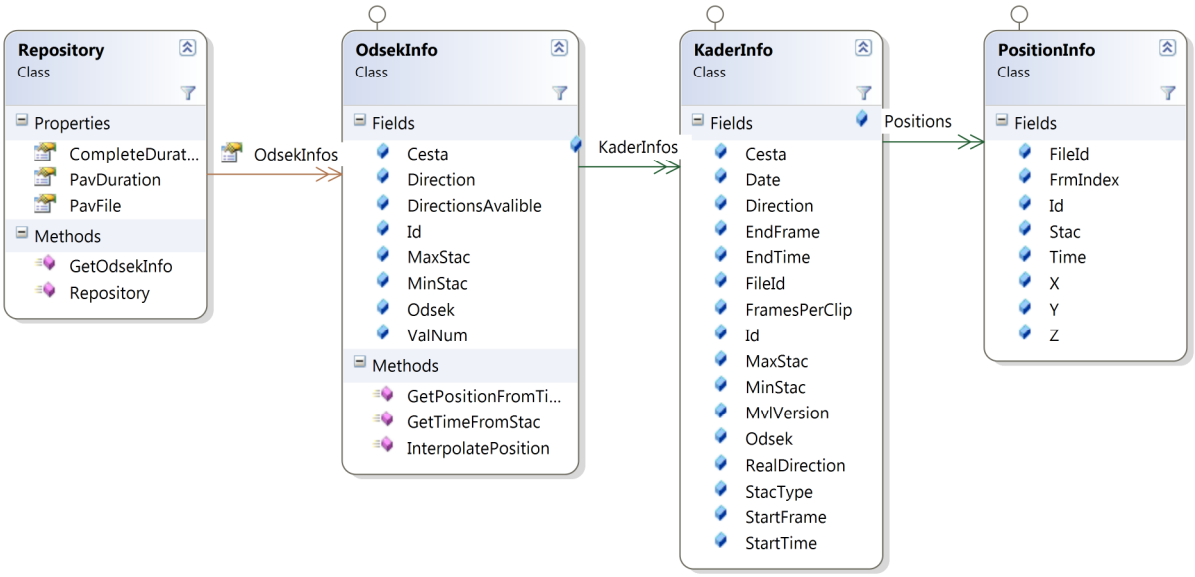
4.3 Aplikacija VideoEncoder

VideoEncoder je aplikacija, namenjena pretvorbi posnetkov, pridobljenih neposredno iz video kamer v format in obliko, primerno za učinkovito distribucijo prek spleta. Pretvorba je v tem primeru sestavljena iz dveh korakov. Najprej je potrebno iz zveznih posnetkov snemalnih dni izvleči uporabne kadre cestnih odsekov ter jih združiti v posnetek odseka, za vsako smer. Drugi korak je kodiranje teh izvlečenih in združenih posnetkov, v obliko primerno za objavo in distribucijo po spletu.

Implementacijo aplikacije sem začel z definicijo modela objektov, opisanih v prejšnjem podpoglavju.

4.3.1 Objektni model geolokacijskih datotek

V podpoglavju 4.2 sem opisal datoteke, ki predstavljajo podatke o odsekih, kadrih in posameznih zajetih lokacijah. Za uporabo v aplikaciji moramo za te objekte pripraviti model. Ta model mora vsebovati vse lastnosti objektov, ki jih pri implementaciji potrebujemo. Slika 8 prikazuje diagram razredov, ki jih lahko definiramo na podlagi vsebine geolokacijskih datotek.



Slika 8. Diagram razredov, ki nastopajo v geolokacijskih datotekah.

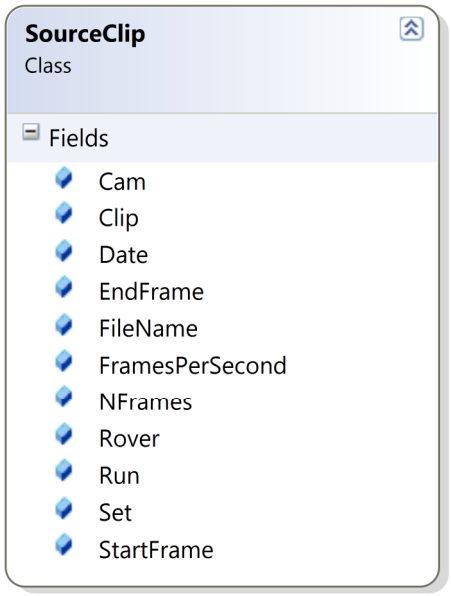
Prvi razred na levi predstavlja repozitorij odsekov. Repozitorij je objekt, ki predstavlja zbirko podatkov o odsekih, ki so bili posneti v okviru nekega letnega snemanja. Repozitorij lahko vsebinsko povežemo z datoteko PAV.

Vsak odsek, ki ga repozitorij vsebuje, sem predstavil z razredom OdsekInfo. Ta vsebuje vse lastnosti, ki jih preberemo iz ene vrstice datoteke PAV (definirana v točki 4.2.3). Poleg tega vsebuje tudi zbirko podatkov o kadrih, ki sestavljajo posnetek odseka.

Naslednji razred KaderInfo smiselno opisuje posamezni kader odseka. Vsebuje lastnosti, ki jih preberemo iz ene vrstice datoteke MVL (definirana v točki 4.2.2). Kot lahko ugotovimo že iz strukture geolokacijskih datotek, ta razred vsebuje tudi zbirko podatkov o zajetih pozicijah.

Vsaka izmed teh pozicij je predstavljena z razredom PositionInfo. Iz diagrama na sliki 8 vidimo, da vsebuje lastnosti, ki so zapisane v vsaki vrstici datoteke VAL (točka 4.2.1).

Če sledimo naslovom v podpoglavju 4.2, lahko ugotovimo, da na diagramu manjka še predstavitev odlomkov, ki jih definira datoteka Clips.dbf. Odlomki so z ostalimi geolokacijskimi objekti povezani le posredno, zato sem model odlomka predstavil na svoji sliki.

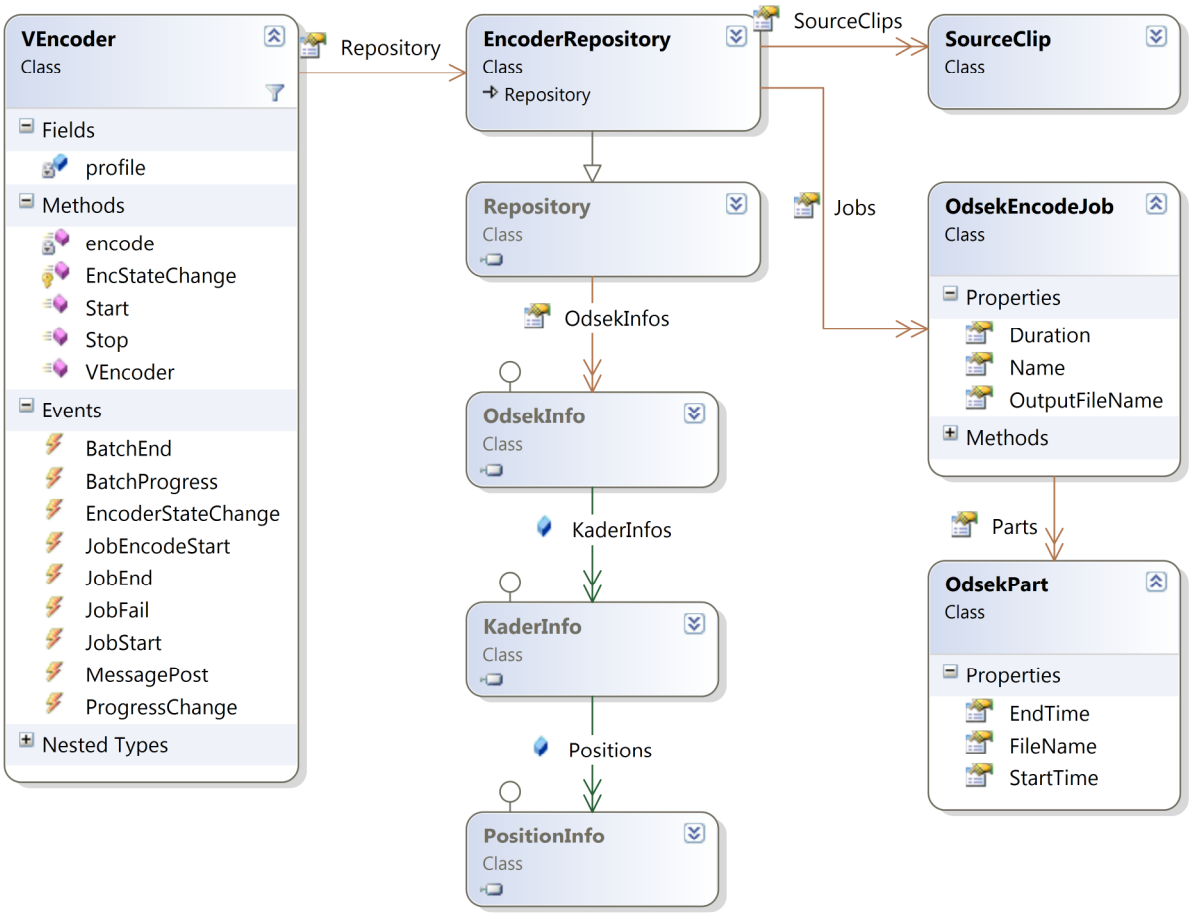


Slika 9. Razred SourceClip, ki predstavlja odlomek posnetka.

Model iz slike 8 je splošen model razredov, ki izhajajo iz geolokacijskih datotek. Uporaben je pri vseh aplikacijah, ki kakorkoli uporabljajo repozitorij podatkov o posnetkih odsekov. Za osnovo sem ga uporabil tako pri aplikaciji VideoEncoder kot pri predvajalniku video posnetkov.

4.3.2 Objektni model aplikacije VideoEncoder

Osnovni model geolokacijskih datotek, sem pri aplikaciji VideoEncoder razširil z razredi, ki jih potrebujemo za razrez/združevanje in kodiranje izvornih video posnetkov. Nastali razredni diagram prikazuje slika 10.



Slika 10. Razredni diagram glavnih razredov v aplikaciji VideoEncoder.

Iz diagrama je razvidno, da je osnovni repozitorij (razred Repository) razširjen z razredom EncoderRepository. Dopolni ga z povezavo na izvorne odlomke in z metodami, ki omogočajo razrez ter združevanje delov izvornih odlomkov v posnetke odsekov. Vse podatke, ki jih za tak razrez potrebujemo pri posnetku enega odseka, sem združil v razred OdsekEncodeJob. Na levi strani diagrama je prikazan razred VEncoder. Namenjen je predvsem izvajanju samega kodiranja zbirke opravil (ang. job), ki jih vsebuje referenciran objekt tipa EncoderRepository.

4.3.3 Kodiranje posnetkov

Razredi definirani v prejšnji točki nam zagotovijo vse potrebne podatke za razrez in združevanje delov video posnetkov. Za dokončno pretvorbo pa moramo izvesti še kodiranje. Za izbran format Smooth Streaming je najbolj primeren programski paket Microsoft Expression Encoder. Za razvoj posebnih aplikacij imamo na voljo razvojni paket Expression Encoder SDK.

Uporaba je zelo priročna, saj nam razvojni paket ponuja enostaven vmesnik za kodiranje. Definirati moramo profil za kodiranje, ki vsebuje vse parametre, kot so format, bitna hitrost in število sličic na sekundo. Poleg profila za začetek kodiranja potrebujemo le še poti do izvornih posnetkov ter začetne in končne čase, kjer se ti posnetki razrežejo.

4.4 Parametri pretvorbe

Pred izvedbo pretvorbe je najprej potrebno definirati njene parametre.

Najprej je potrebno izbrati kodek pretvorjenih posnetkov. Smooth Streaming nam nudi izbiro med kodekom VC1 in H.264. Zaradi manjše procesne zahtevnosti pri dekodiranju, sem za kodek izbral VC1.

Zelo pomembna je tudi izbira bitnih hitrosti. Izbira je odvisna od pasovne širine povezave s katero bodo uporabniki dostopali do aplikacije. V našem primeru je aplikacija namenjena uporabi prek interneta, zato sem zgornjo mejo bitne hitrosti postavil na 2 Mbit/s. Podobno moramo definirati tudi spodnjo mejo bitne hitrosti z najmanjšo pasovno širino povezave, ki jo bomo zahtevali od uporabnikov aplikacije. Predvideval sem, da danes velika večina uporabnikov do spleta dostopa s povezavo pasovne širine vsaj 500 kbit/s. Spodnjo mejo bitne hitrosti sem tako postavil na 440 kbit/s, saj povezave ne smemo popolnoma zapolniti samo z multimedijskimi prenosi, ker druga komunikacija v tem primeru postane neodzivna.

Določiti moramo tudi vmesne bitne hitrosti, saj je za zvezne preklope med spodnjo in zgornjo mejo prevelik razpon. S testiranjem sem ugotovil, da zadostujeta dve vmesni bitni hitrosti.

Z bitnimi hitrostmi so povezane tudi ločljivosti posnetka. Za maksimalno ločljivost sem izbral standardno resolucijo visoke ločljivosti (ang. High Definition, HD) 1280x720. Ostale ločljivosti sem smiselno določil z zmanjševanjem, premo sorazmerno z bitno hitrostjo. Končne vrednosti bitnih hitrosti in ločljivosti so prikazane v tabeli 3.

Bitna hitrost	Širina	Višina
2000 kbit/s	1280	720
1190 kbit/s	960	540
800 kbit/s	640	360
440 kbit/s	428	240

Tabela 3. Bitne hitrosti in ločljivosti.

4.5 Izvedba pretvorbe

V aplikaciji za navigacijo potrebujemo posnetke cest celotne Slovenije. Količino podatkov posnetih odsekov prikazuje tabela 4

Število odsekov	744
Skupna dolžina v razdalji	10.667 km
Skupen čas uporabnih posnetkov	307 ur
Skupna velikost izvornih posnetkov	4107 GB

Tabela 4. Podatki o posnetem materialu.

Kot vidimo je količina podatkov zelo velika. Za pretvorbo tako velike količine podatkov potrebujemo močno strojno opremo. Tabela 5 prikazuje podatke strežnika, ki sem ga uporabil za pretvorbo.

Tip procesorja	Intel Xeon CPU E5430
Delovna frekvenca jeder	2.66 GHz
Število jeder	8
Količina pomnilnika	16 GB
Operacijski sistem	Windows Server 2008 R2 64-bit

Tabela 5. Specifikacija strežnika za pretvorbo.

Kljub močnemu strežniku, je pretvorba zaradi velike količine podatkov še vedno trajala razmeroma dolgih 36 dni. Tabela 6 prikazuje količino podatkov po pretvorbi.

Trajanje pretvorbe	877 ur (36,5 dni)
Končna količina podatkov	570 GB
Skupen čas pretvorjenih posnetkov	307 ur

Tabela 6. Količina podatkov po pretvorbi.

V primerjavi količine podatkov izvornih posnetkov (iz tabele 4) in pretvorjenih posnetkov, lahko opazimo, da je prišlo do več kot sedemkratnega zmanjšanja količine podatkov, kljub temu, da so pretvorjeni posnetki štirikrat replicirani. Zmanjšanje je posledica predvsem veliko večje učinkovitosti kodeka VC1 v primerjavi s kodekom Divx, s katerim so zakodirani izvorni podatki. Pri pretvorbi sicer pride do manjšega zmanjšanja kvalitete, vendar je kvaliteta za ogled odsekov še vedno zelo visoka.

5 Iskanje najcenejše poti

V današnjem prezaposlenem svetu, se nam vedno nekam mudi. Prispeti želimo v podjetje, kjer nas čaka nov poslovni sestanek, v kavarno, kjer se s prijateljem dobimo na pijači ali pa se želimo le čimprej vrniti domov, da si oddahnemo od napornega dne. Pri tem največkrat nevede naši možgani rešujejo razmeroma zapleten problem iskanja najkrajše (oz. najcenejše) poti.

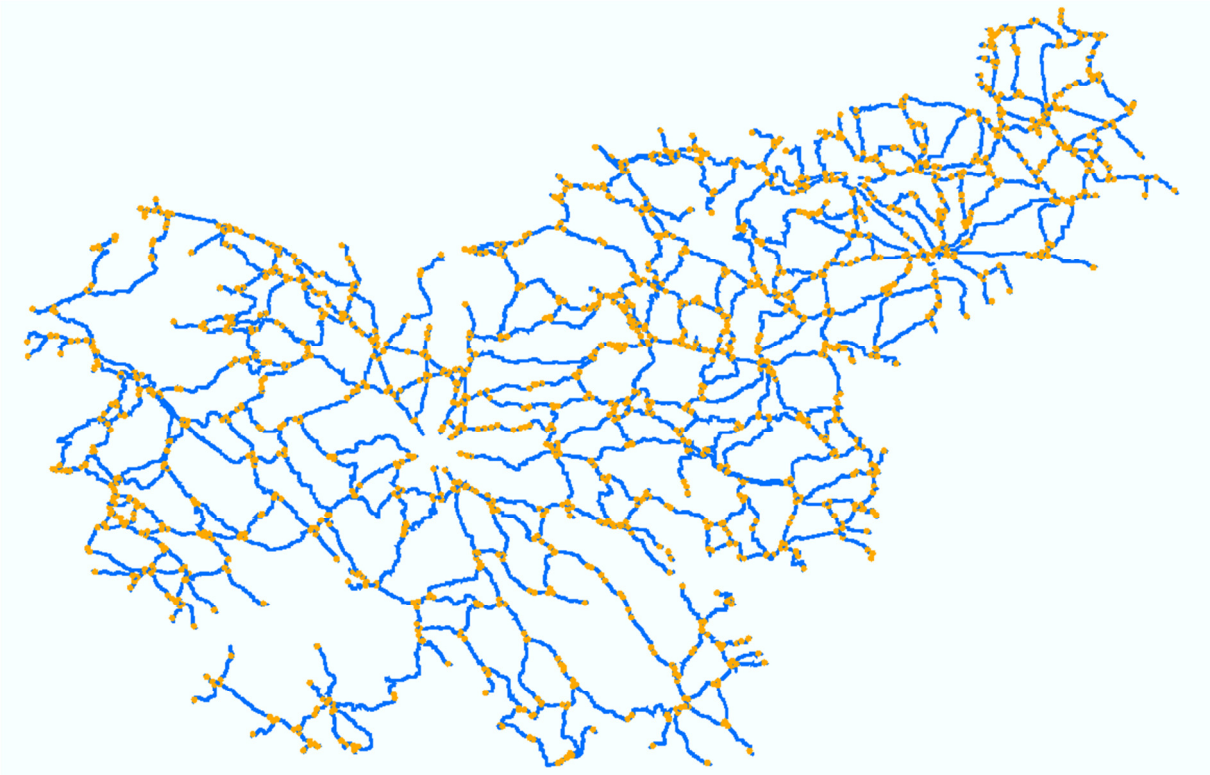
Za izvajanje algoritma možgani potrebujejo tudi oporne podatke. Ko se prvič odpravljamo na nepoznan kraj, pot ponavadi iščemo s pomočjo zemljevida. Na njem lahko vidimo kraje in ceste, ki jih povezujejo med seboj. Postopek intuitivno začnemo z iskanjem naše začetne lokacije ter destinacije. Sledi enostavno sledenje cestnemu omrežju od začetka proti destinaciji. Ob vsakem večjem križišču preletimo ceste, ki vodijo iz njega, ter izberemo po našem mnenju najbolj optimalno pot za nadaljevanje. Vse to počnemo, dokler ne dosežemo cilja in ocenimo, da smo zadovoljni s potjo.

Enak postopek pa lahko definiramo tudi v bolj strogo definiranem matematičnem svetu. Omrežje cest in krajev lahko predstavimo z grafom. Graf je struktura, ki jo sestavljajo vozlišča (ang. vertex) ter povezave med njimi (ang. edge). V našem primeru si lahko vozlišča poenostavljeno predstavimo kot kraje (ali sečišča cest), povezave pa kot ceste. Kraj lahko definiramo kot koordinato v prostoru, cesto pa z začetnim in končnim krajem. Za vsako cesto lahko določimo tudi atribute, ki nam pomagajo pri odločanju. Glede na situacijo nas zanima na primer dolžina, omejitev hitrosti, tip ceste, kvaliteta ali naklon. Te atribute z eno besedo poimenujemo kot uteži ali cene.

Za iskanje poti v grafu vozlišč in povezav poznamo več algoritmov. V nadaljevanju bom najprej opisal enega najbolj splošnih, ki se imenuje algoritem Dijkstra, nato pa bom z njim primerjal tudi izboljšan algoritem A*.

5.1 Podatki za usmerjanje

Za izvedbo iskanja najkrajše poti potrebujemo ustrezno podatkovno bazo, nad katero se bo algoritem izvajal. Iz cestnega omrežja moramo zgraditi graf vozlišč in povezav med njimi. Najprimernejši sloj iz BCP za ta namen je sloj pododsekov. Ta vsebuje vse odseke državnih cest v Sloveniji, ki so razdeljeni na pododseke.



Slika 11. Sloj pododsekov v BCP.

Na sliki 11 je prikazan sloj vseh pododsekov, ki imajo pripadajoč video posnetek. Z oranžnimi pikami so označena krajišča pododsekov. Za vsak odsek imamo na voljo tudi podatek o smereh prevoznosti (naprej, nazaj ali v obe smeri), tako da lahko graf iz sloja enostavno zgradimo.

Ob predstavitvi sloja z grafom, ta vsebuje 1484 vozlišč in 3600 usmerjenih povezav med njimi.

5.2 Definicija osnovnih pojmov

Kot smo že ugotovili v uvodu poglavja, nam ceno poti lahko predstavljajo različne količine. Zaradi lažje predstave bom v nadaljevanju razlage kot utež uporabljal razdaljo, ostale količine pa se seveda uporabljajo na analogen način.

Graf zapišemo kot $G = (V, E)$, kjer je V množica vozlišč in E množica povezav. $n = |V|$ predstavlja število vozlišč, $m = |E|$ pa število povezav. Povezavo U, ki povezuje vozlišče i z vozliščem j zapišemo kot $U = (i, j) \in E$. Potrebujemo še funkcijo cene l, ki jo definiramo kot $l: E \to \mathbb{R}^+$, kjer je $\mathbb{R}^+$ množica pozitivnih realnih števil. Oznaka l_{ij} predstavlja ceno povezave, ki povezuje vozlišči i in j. Množico vseh povezav, ki izvirajo iz vozlišča i bomo označevali z E_i. Čeprav lahko ceno poti opazujemo kot različne količine (razdalja, čas, naklon...), bom v nadaljevanju zaradi lažjega razumevanja, za ceno v besedilu uporabil prostorsko razdaljo.

Pot med vozliščema i in j je najkrajša, če ne obstaja nobena druga možna pot med vozliščema i in j, ki ima krajšo dolžino.

5.3 Algoritem Dijkstra

Algoritem Dijkstra je eden najbolj univerzalnih algoritmov za iskanje najkrajših poti v grafu. V letu 1959 ga je razvil Nizozemec Edsger Dijkstra. Čeprav iščemo pot med dvema vozliščema, nam ta algoritem izračuna najkrajše poti med začetnim in vsemi ostalimi vozlišči. Tak problem lahko poimenujemo tudi kot problem iskanja najkrajše poti v grafu z enim izvorom [2].

Na začetku algoritem seveda potrebuje ustrezne podatkovne strukture, po katerih se bo izvajal. Za vsako vozlišče potrebuje algoritem za svoje delovanje tri dodatne atribute. Prvi je trenutno izračunana razdalja od izvora do vozlišča in jo označimo kot $g(i)$. Drugi atribut je referenca na predhodno vozlišče, ki je na trenutno izračunani najkrajši poti. Tretji atribut je zastavica, ki nam pove ali smo vozlišče že obiskali. Algoritem za vhodni parameter potrebuje začetno vozlišče, na katerem se algoritem začne. Na njem najprej nastavimo razdaljo na 0, na vseh ostalih vozliščih pa na neskončno, kar pomeni, da vozlišč še nismo obiskali. Nadaljevanje algoritma lahko opišemo s spodnjimi koraki.

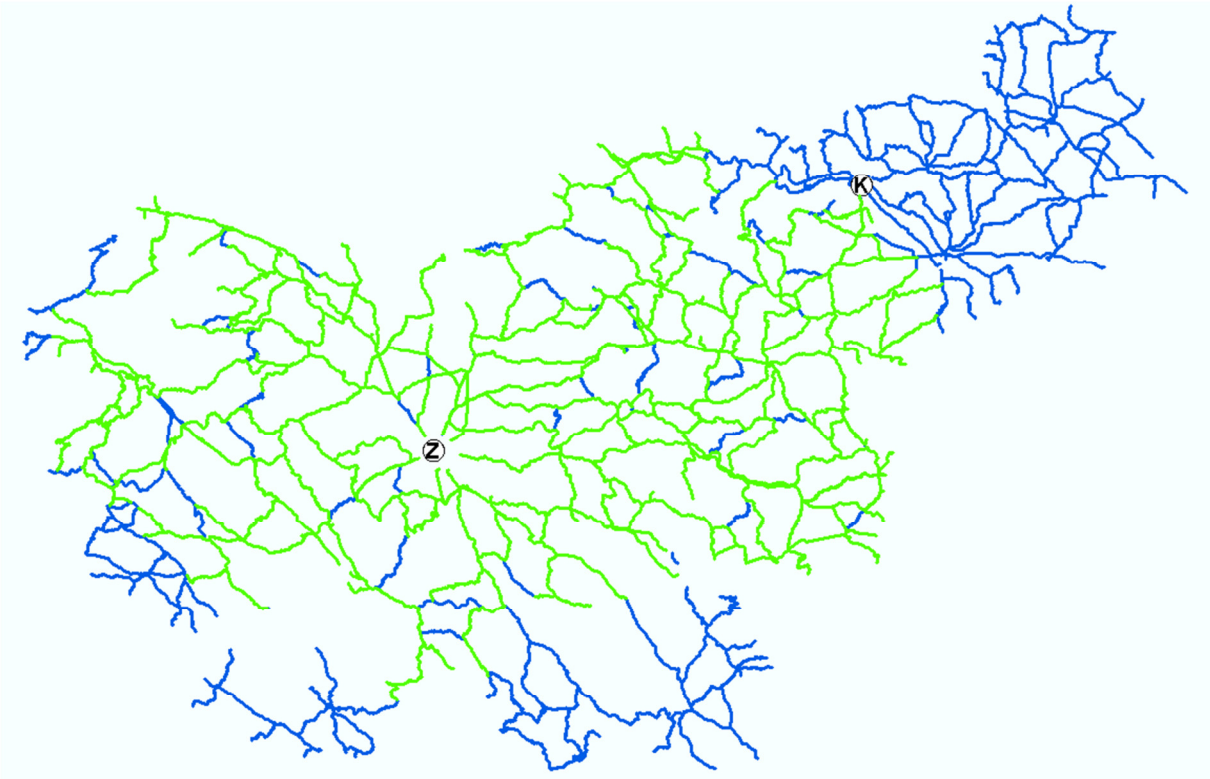
1. Iz trenutnega vozlišča i se sprehodimo po vseh sosednih vozliščih j. Če je vozlišče j že obiskano, ga preskočimo. Na vsakem vozlišču j izračunamo vsoto $g(i) + l_{ij}$. Ta vsota predstavlja razdaljo od začetnega vozlišča do vozlišča j prek

vozlišča i. V primeru, če je vsota manjša od obstoječe razdalje $g(j)$, jo vpišemo kot novo najmanjšo razdaljo $g(j)$ in kot predhodnika vozlišča j nastavimo trenutno vozlišče i.

2. Po pregledu vseh sosednjih vozlišč, trenutno vozlišče i označimo kot obiskano. Obiskano vozlišče ima v tem trenutku že izračunano minimalno razdaljo in se do konca algoritma ne bo več spremenila.
3. Za naslednje trenutno vozlišče izberemo neobiskano vozlišče z najmanjšo vrednostjo $g(i)$ in postopek ponovimo s prvim korakom. Če so obiskana že vsa vozlišča, se algoritem konča.

Časovna zahtevnost opisanega postopka je ob najenostavnejši implementaciji struktur za iskanje naslednjega minimalnega vozlišča $O(|V|^2 + |E|) = O(|V|^2)$. Z naprednejšo izbiro podatkovne strukture (kopica) lahko to zahtevnost izboljšamo na $O(|V| \cdot \log|V| + |E|)$.

Algoritem Dijkstra vedno najde najkrajšo možno pot med začetnim in vsemi ostalimi vozlišči, pri čemer se vedno sprehodi čez celoten graf. V tretjem koraku lahko postavimo dodaten pogoj, ki postopek prekine, ko je doseženo končno vozlišče.



Slika 12. Obdelani del grafa pri uporabi algoritma Dijkstra.

Na sliki 12 je z zeleno barvo prikazan del grafa, ki ga algoritem z dodatnim pogojem obišče pri iskanju najkrajše poti med Ljubljano in Mariborom. Začetna točka je označena z oznako Z, končna pa z oznako K. Vidimo lahko, da je precejšen del grafa

ostal neobiskan (modra barva), saj je obiskanih samo 1013 od 3600 povezav. Rezultat, ki ga prikazuje slika, pa je mogoče še precej izboljšati. Že zdrava pamet nam namreč pove, da v večini primerov nima smisla iskati najkrajše poti do Maribora skozi Jesenice.

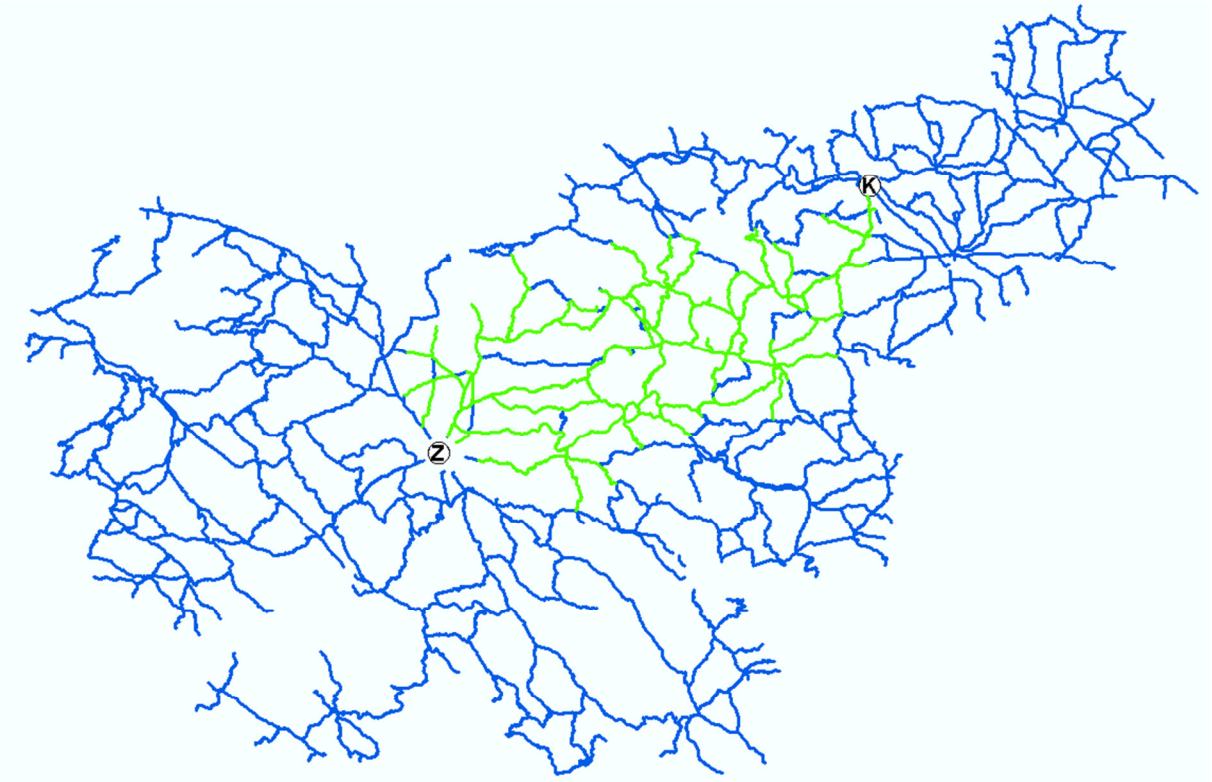
5.4 Algoritem A*

Algoritem A* temelji na algoritmu Dijkstra in ga razširi z uporabo hevrističnih funkcij za optimalnejšo odločanje pri sprehodu skozi graf. Hevristična funkcija, ki jo označujemo z $h(i)$ je funkcija, ki približno oceni ceno poti od vozlišča i do ciljnega vozlišča. Izkaže se, da je za oceno pri primeru iskanja najkrajše poti, zelo primerna funkcija zračne razdalje do končnega vozlišča. To pomeni, da bo algoritem preferiral izbiro vozlišča, ki je bližje končnemu vozlišču. Potrebujemo tudi definicijo funkcije $f(i) = g(i) + h(i)$, ki predstavlja oceno dolžine celotne poti od začetnega do končnega vozlišča skozi vozlišče i.

Vsakemu vozlišču moramo za delovanje algoritma A* poleg razdalje, predhodnika in zastavice dodati še četrti atribut, ki bo hranil oceno $h(i)$. Inicializacija algoritma je podobna kot pri algoritmu Dijkstra, dodati moramo le še inicializacijo ocene na začetnem vozlišču. Njeno vrednost nastavimo na vrednost funkcije $h(i_0)$. Začnemo pri začetnem vozlišču z postopkom, ki je zelo podoben algoritmu Dijkstra:

1. Iz trenutnega vozlišča i se sprehodimo po vseh sosednih vozliščih j. Če je vozlišče j že obiskano ga preskočimo. Na vsakem vozlišču j izračunamo vsoto $g(i) + l_{ij}$. V primeru, da je vsota manjša od obstoječe razdalje $g(j)$, jo vpišemo kot novo najmanjšo razdaljo $g(j)$ in kot predhodnika vozlišča j nastavimo trenutno vozlišče i. Izračunamo tudi oceno razdalje do končnega vozlišča $h(j)$.
2. Po pregledu vseh sosednjih vozlišč, trenutno vozlišče i označimo kot obiskano.
3. Za naslednje trenutno vozlišče izberemo neobiskano vozlišče z najmanjšo vrednostjo $f(i)$. Če je to končno vozlišče, potem se algoritem konča, sicer postopek ponovimo s prvim korakom.

Kot vidimo, se postopek od algoritma Dijkstra razlikuje le z dodatnim izračunom ocene v prvem koraku in s spremenjeno izbiro naslednjega vozlišča v tretjem koraku. Uporablja tudi zaključni pogoj, ki smo ga pri algoritmu Dijkstra uporabili kot izboljšavo.



Slika 13. Obdelani del grafa pri uporabi algoritma A*.

Slika 13 z zeleno barvo prikazuje del grafa, ki ga algoritem obišče pri iskanju najkrajše poti med Ljubljano in Mariborom. V primerjavi s sliko 12 je rezultat veliko bolj optimalen, saj je obarvanih samo še 350 povezav.

Kljub temu, da algoritem A* preišče veliko manjši del grafa kot algoritem Dijkstra, pa je s primerno izbiro hevristične funkcije še vedno optimalen. To velja kadar je ocena $h(i)$ vedno manjša ali enaka od dejanske razdalje po grafu od vozlišča i do končnega vozlišča. Pri naši izbiri zračne razdalje za $h(i)$ je temu pogoju zadoščeno, saj dejanska razdalja v nobenem primeru ne more biti krajša od zračne razdalje med dvema vozliščema.

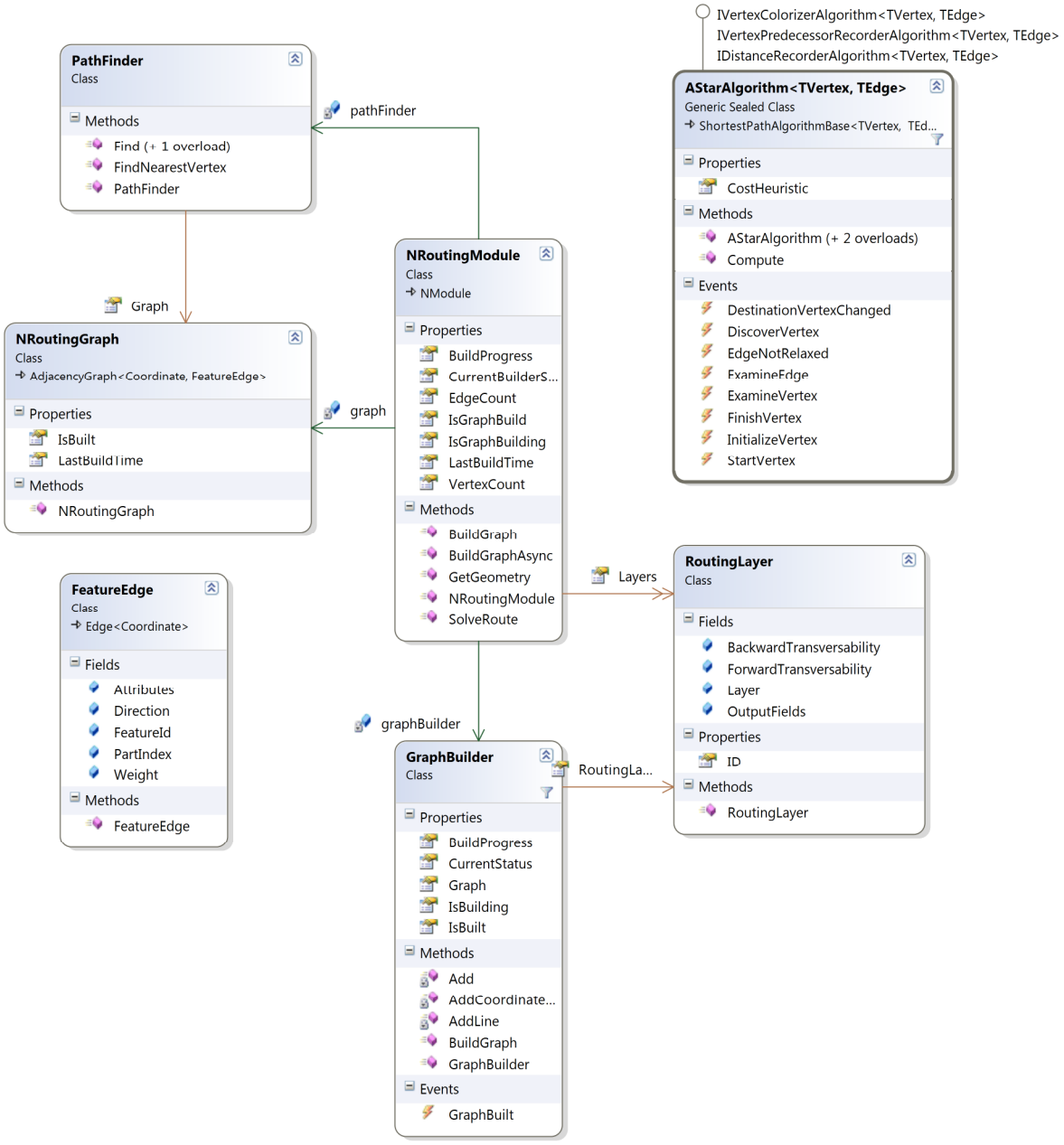
5.5 Implementacija usmerjanja

Zaradi večje standardizacije in lažjega prenosa kode, sem pri razvoju uporabil dve zunanji knjižnici. Prva se imenuje NetTopologySuite in vsebuje definicije struktur in splošne operacije za manipulacijo geometrije. Tako predstavitev geometrije kot implementacija operacij, je združljiva s standardi OGC. OGC (ang. Open Geospatial Consortium [5]) je vodilna mednarodna organizacija za definicijo standardov na področju prostorskih podatkov. Pri implementaciji geografskih informacijskih sistemov

je združljivost s standardi OGC za integracijo zelo pomembna, zato je za predstavitev geometrije priporočljiva uporaba zunanjih knjižnic.

Druga zunanja knjižnica implementira osnovne podatkovne strukture in operacije za predstavitev in manipulacijo grafa. Imenuje se QuickGraph. Vse strukture so implementirane z uporabo generikov (ang. generics), zato omogočajo visoko stopnjo razširljivosti. Knjižnica vsebuje tudi implementacije velikega števila uporabnih algoritmov kot so topološko urejanje, iskanje v širino, iskanje v globino, iskanje najkrajših poti in maksimalni pretok.

Implementacija iskanja najkrajše poti v grafu je razdeljena na več sklopov. Vse potrebne razrede sem implementiral znotraj imenskega prostora Luz.Nukleus.Routing. Glavne razrede, ki so potrebni za implementacijo, prikazuje slika 14.



Slika 14. Glavni razredi v imenskem prostoru Luz.Nukleus.Routing.

Osrednji razred, ki povezuje vse ostale, se imenuje NRoutingModule. Namenjen je predstavitvi zaključenega modula, ki vsebuje vse komponente, potrebne za iskanje najkrajših poti.

5.5.1 Podatkovne strukture

Knjižnica QuickGraph nam nudi generične strukture za predstavitev grafa in povezav. Generičnost nam omogoča, da sami definiramo osnovne tipe, ki se uporabljajo v

kompleksnejši strukturi. Tako imamo pri tipu Graph iz knjižnice možnost, da sami definiramo kakšen bo tip vozlišča in tip povezave.

V našem primeru je najbolj primerna predstavitev za vozlišče geografska koordinata, ki predstavlja začetek ali konec pododseka. Pododsek je v podatkovni bazi predstavljen kot polilinija z atributi. Za potrebe grafa je smiselno pododsek kot povezavo med dvema vozliščema poenostaviti, saj npr. celotne geometrije, ki zasede veliko pomnilnika, za samo iskanje ne potrebujemo. V ta namen sem definiral nov razred z imenom FeatureEdge:

```
public class FeatureEdge : Edge<Coordinate>
{
    /* Dedovano iz Edge<Coordinate>
    public Coordinate Source { get; }
    public Coordinate Target { get; }
    */
    //Identifikator grafičnega objekta, ki mu povezava pripada
    public readonly object FeatureId;
    //Teža ali cena povezave
    public readonly double Weight;
    //Smer povezave
    public readonly Transverability Direction;
    //Index dela geometrije v primeru multi-part
    public readonly int PartIndex;
    //Konfigurabilni dodatni atributi, ki jih potrebujemo
    public readonly Dictionary<string, object> Attributes;

    public FeatureEdge(...):base(source, destination)
    { ... }
}

public enum Transverability
{
    Forward, Backward, Both
}
```

Koda 1. Razred FeatureEdge.

Razred poleg dedovanih lastnosti Source (začetno vozlišče) in Target (končno vozlišče), vsebuje tudi polja, ki jih potrebujemo v konkretnem primeru iskanja poti v cestnem omrežju. Njihovi pomeni so razvidni iz komentarjev. Polje Attributes, je namenjeno prenosu atributov, ki jih pri samem iskanju najkrajše poti ne bomo potrebovali. V ta slovar bomo shranjevali vrednosti atributov, ki jih potrebujemo za namen sestavljanja video posnetka, kot so začetna in končna stacionaža pododseka ter identifikator odseka.

Sedaj ko smo definirali tip vozlišča in podatkovno strukturo povezave, lahko definiramo tudi razred NRoutingGraph.

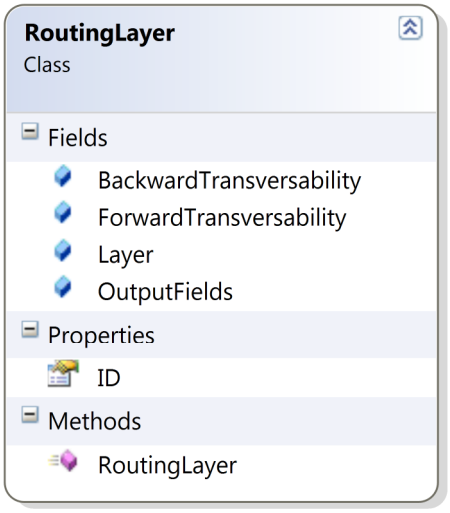
```
/// <summary>
/// Razred, ki definira generičen graf za potrebe iskanja poti
/// </summary>
public class NRoutingGraph : AdjacencyGraph<Coordinate, FeatureEdge>
{
    /* Dedovano iz AdjacencyGraph
    public virtual IEnumerable<FeatureEdge> Edges { get; }
    public virtual IEnumerable<TVertex> Vertices { get; }
    ...
    */
    //Čas, ko je bil graf zadnjič posodobljen
    public DateTime? LastBuildTime { get; set; }
    //Zastavica, ki pove če je graf zgrajen
    public bool IsBuilt { get; set; }

    public NRoutingGraph():base(true) { ... }
}
```

Koda 2. Razred NRoutingGraph.

Razred je zelo enostaven in je dedovan iz splošnega usmerjenega grafa. Z njim določimo tip vozlišča in povezave, ter dve dodatni lastnosti LastBuildTime in IsBuilt, ki nam pomagata pri ugotavljanju ažurnosti grafa.

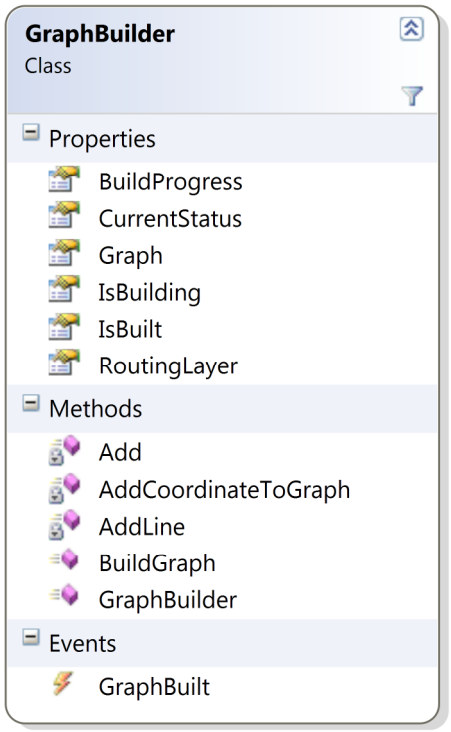
Poleg razredov, ki so neposredno povezani s tematiko grafov, pa potrebujemo tudi razred, ki služi kot povezava med podatkovno bazo cestnega omrežja in grafom. To je razred RoutingGraph, katerega strukturo prikazuje slika 15. Razred poleg reference na grafični sloj vsebuje še konfiguracijske parametre za ugotavljanje prehodnosti elementov sloja ter spisek dodatnih atributov, ki se morajo prenesti v graf.



Slika 15. Struktura razreda RoutingLayer.

5.5.2 Gradnja grafa iz sloja pododsekov

Same strukture pa nam ne koristijo, če jih ne napolnimo s podatki. Logiko za gradnjo grafa sem zapakiral v razred GraphBuilder.



Slika 16. Struktura razreda GraphBuilder.

Slika 16 prikazuje strukturo razreda. Pomen lastnosti je razviden že iz njihovih imen. Operacija gradnje grafa je lahko časovno zelo zahtevna, saj vključuje tudi nalaganje geometrije celotnega sloja, kar obsega veliko podatkov. Iz tega razloga je razred zasnovan tako, da je operacijo možno izvajati asinhrono.

Objektu tipa GraphBuilder moramo že ob kreiranju nastaviti lastnost RoutingLayer, ki služi kot izvor vozlišč ter povezav med njimi. S klicem metode BuildGraph sprožimo gradnjo grafa. V njej se po inicializaciji najprej sproži nalaganje grafičnih elementov (pododsekov). Za vsak element se izvede naslednji postopek:

- izračuna se prehodnost elementa,
- v primeru prehodnosti vsaj v eno smer se v graf kot vozlišča dodata koordinati začetne in končne točke linije elementa,
- v primeru prehodnosti naprej se iz elementa kreira nova povezava tipa FeatureEdge,

- v primeru prehodnosti nazaj se iz elementa kreira nova povezava tipa FeatureEdge z zamenjanima začetnim in končnim vozliščem glede na element.

Pri kreiranju nove povezave se pri obeh točkah, seveda na podlagi konfiguracije v objektu RoutingLayer, nastavijo atributi povezave. Ko je graf do konca zgrajen, se lahko prične uporabljati za iskanje poti.

5.5.3 Implementacija algoritma A*

Knjižnica QuickGraph sicer že vsebuje implementacijo algoritma A*, vendar za iskanje najkrajše poti med samo dvema točkama ni optimalen. Algoritem sem zato znotraj razreda AStarAlgorithm implementiral sam. Razred deduje iz abstraktnega razreda ShortestPathAlgorithmBase, ki je del knjižnice QuickGraph. Zagotovi nam osnovne atribute, dogodke in metode, ki jih potrebujejo vsi algoritmi za iskanje najkrajše poti. Implementira tudi tri vmesnike IVertexPredecessorRecorderAlgorithm, IdistanceRecorderAlgorithm in IVertexColorizerAlgorithm. Prvi vmesnik določa dogodka FinishVertex in StartVertex, ki omogočata sledenje predhodnikov. Drugi z dogodkoma DiscoverVertex in InitializeVertex omogoča beleženje prepotovane razdalje.

Tretji vmesnik IVertexColorizerAlgorithm naznanja, da knjižnica QuickGraph za označevanje vozlišč namesto zastavic uporablja barve. Vozlišče je lahko pobarvano v belo, sivo ali črno barvo. Bela barva pomeni, da vozlišče še ni bilo doseženo, siva, da je bilo vozlišče že doseženo, vendar obdelava še ni zaključena. Črna barva pomeni, da je obdelava na tem vozlišču že zaključena in se njegove vrednosti ne bodo več spreminjale.

Pri implementaciji algoritma sem ohranil generične parametre za tipe, tako da je implementacija primerna tudi za druge izbire podatkovne strukture.

Algoritem je implementiran podobno kot ga opisuje psevdo koda v podpoglavju 5.4. Za kopico čakajočih vozlišč uporablja Fibonaccijevo vrsto [2]. Glede na osnovno implementacijo sem postopek optimiziral tako, da se algoritem niti pri inicializaciji ne sprehodi čez celoten graf. Tako je algoritem primeren tudi za veliko večja cestna omrežja, kjer bi bila časovno zahtevna že inicializacija. Med sprehodom skozi graf algoritem obišče manjši del vozlišč in povezav, ki se nahajajo na poti med začetnim in končnim vozliščem.

5.6 Časovne meritve algoritma

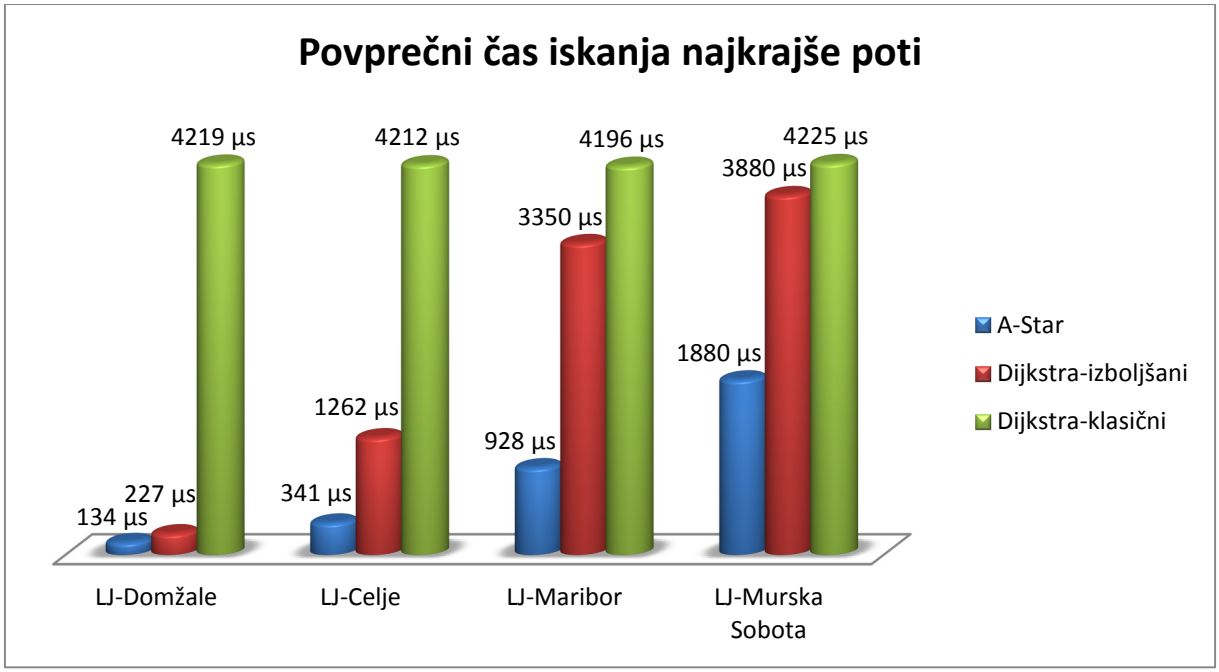
Po zaključeni implementaciji, sem izvedel tudi meritve časov izvajanja. Z meritvami sem želel ugotoviti razlike v časih izvajanja vseh treh omenjenih različic algoritma za iskanje najkrajše poti. Za začetno točko sem pri vseh izvajanjih uporabil vozlišče, ki se nahaja v Ljubljani. Zaradi načina izvajanja algoritmov je smiselno izbrati končna vozlišča z različnimi oddaljenostmi od začetnega. Za vsako različico algoritma sem izvedel meritve iskanja poti do vozlišča v Domžalah, Celju, Mariboru in Murski Soboti.

Časovne meritve sem izvajal na osebnem računalniku s konfiguracijo v tabeli 7.

Tip procesorja	Intel Core i7 Q820
Delovna frekvenca jeder	1.73 GHz
Število jeder	4 (x2-Hyper threading)
Količina pomnilnika	8 GB
Operacijski sistem	Windows 7 64-bit

Tabela 7. Konfiguracija računalnika za izvajanje meritev algoritma.

Algoritem sem izvajal v okviru izvajalnega okolja ASP.NET verzije 4.0. Vsako kombinacijo izvajanja sem za večjo natančnost meritev ponovil 10.000-krat. Po izvedenih ponovitvah sem izmed vseh 10.000 meritev izračunal povprečni čas izvajanja vsake kombinacije algoritma in končnega vozlišča. Rezultate prikazuje slika 17.



Slika 17. Graf povprečnih časov iskanja najkrajših poti.

Na sliki je prikazan graf vseh 12 kombinacij izvajanja algoritma iskanja najkrajše poti. Kot najbolj očitno značilnost lahko opazimo, da je čas izvajanja klasičnega algoritma Dijkstra (označen z zeleno barvo) konstanten, ne glede na izbiro končnega vozlišča. To je razvidno že iz same definicije algoritma, ki v vsakem primeru preišče celoten graf. Konstantni časi kažejo tudi na zanesljivost meritev. Namenski strežnik za iskanje najkrajših poti podobne konfiguracije bi, v primeru uporabe klasičnega algoritma Dijkstra, zmogel izvesti vsaj 200 zahtevkov za izračun najkrajše poti v eni sekundi na izvajalni procesor.

Zanimiva je tudi primerjava klasičnega algoritma Dijkstra z izboljšanim algoritmom, ki se ustavi, ko doseže končno vozlišče. Vidimo lahko, da je prihranek časa zelo visok pri nizki oddaljenosti končnega vozlišča, pri veliki oddaljenosti pa se čas izvajanja izboljšanega algoritma že povsem približa času klasičnega algoritma.

Razširitev s hevristiko, ki jo implementira algoritem A*, se z meritvami izkaže za zelo učinkovito. Pri iskanju najkrajših poti pri majhnih oddaljenostih je algoritem kar 31-krat hitrejši od klasičnega. Pohitritev z oddaljenostjo vozlišča sicer pada, vendar tudi v najslabšem primeru, ko iščemo pot do najbolj oddaljenega vozlišča, je algoritem še vedno več kot dvakrat hitrejši.

Čas izvajanja algoritma A* se izenači s časom izvajanja klasičnega algoritma le v primeru, ko končno vozlišče iz začetnega ni dosegljivo. V tem primeru algoritem razumljivo preišče celoten graf. Temu se moramo izogniti, zato je zelo pomembno, da v podatkih o cestnem omrežju ni napak, saj naj tudi v realnem svetu ne bi obstajala nedosegljiva vozlišča.

6 Aplikacija za napredno predvajanje video posnetkov

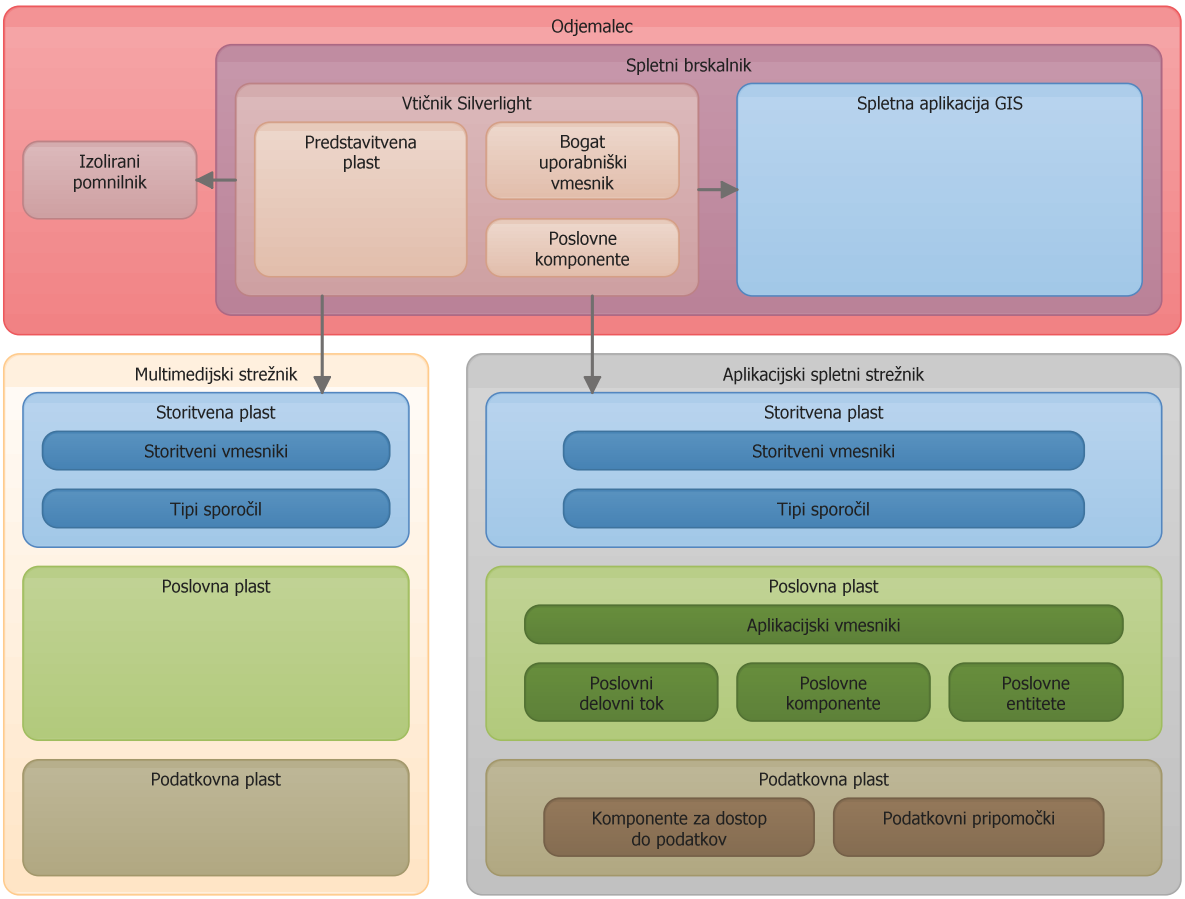
V tem poglavju bom opisal aplikacijo za napredno spletno predvajanje video posnetkov. Poleg samega predvajanja video posnetkov cest, aplikacija vsebuje veliko več funkcionalnosti:

- predvajanje posnetkov odsekov, pridobljenih z aplikacijo VideoEncoder,
- povezovanje posnetkov z geolokacijskimi podatki,
- uporaba usmerjanja po cestnem omrežju za združevanje posnetkov odsekov,
- prikaz rezultatov usmerjanja in trenutne lokacije v spletni aplikaciji GIS,
- merjenje velikosti objektov na cestišču.

Vsaka izmed funkcionalnosti je podrobneje opisana v nadaljevanju.

6.1 Arhitektura aplikacije

Aplikacija je zasnovana kot bogata internetna aplikacija (ang. Rich Internet Application, RIA). Plastni diagram (ang. Layer Diagram) prikazuje slika 18.



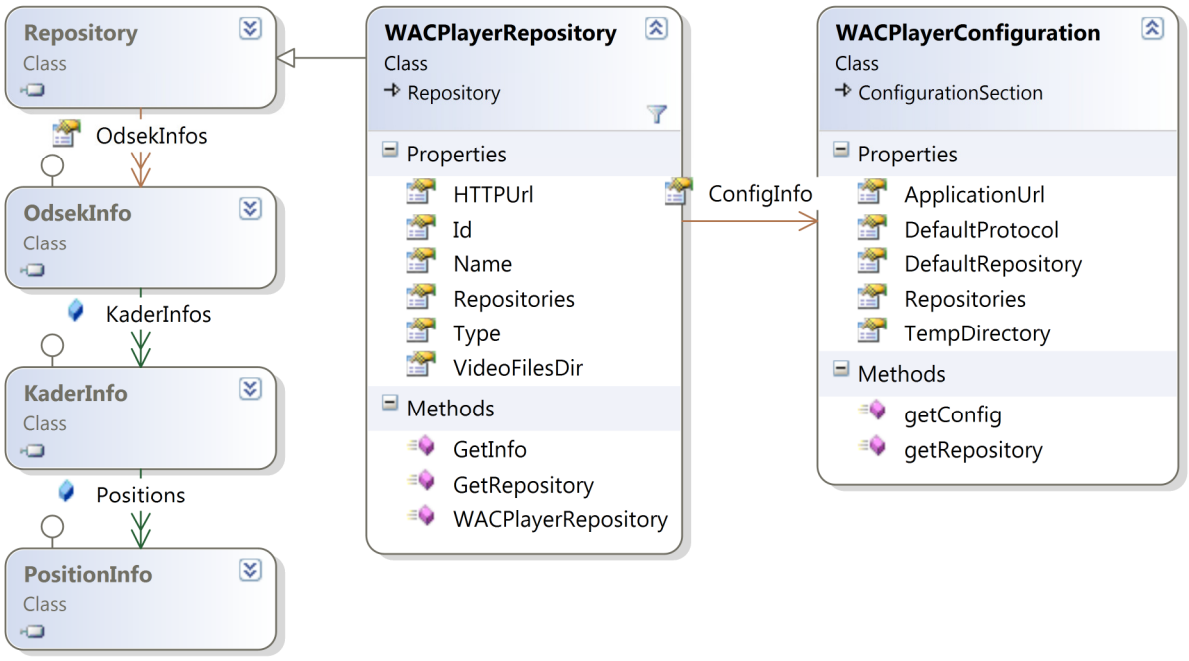
Slika 18. Plastni diagram aplikacije za predvajanje video posnetkov.

Iz diagrama lahko razberemo, da lahko rešitev razdelimo na tri glavne komponente:

- aplikacijski spletni strežnik, ki zagotavlja potrebne geolokacijske podatke in podatke za usmerjanje,
- multimedijski strežnik IIS Media Services, ki streže video posnetke odsekov in je opisan v točki 3.3.2,
- odjemalec, ki vse komponente povezuje in skrbi tudi za komunikacijo s spletno aplikacijo GIS.

6.2 Strežniški del aplikacije za predvajanje

Strežniški del aplikacije zagotavlja spletno storitev za dostop do repozitorija odsekov in pripadajočih geolokacijskih podatkov. Podobno kot aplikacija Video Encoder, tudi predvajalnik razširja osnovni razredni diagram geolokacijskih datotek (slika 8). Razširjeni diagram je prikazan na sliki 19.



Slika 19. Razredni diagram strežniškega dela predvajalnika.

Levi stolpec predstavlja osnovni objektni model geolokacijskih datotek, ki je podrobneje predstavljen v točki 4.3.1. Razred WACPlayerRepository razširja osnovni razred Repository, ki vsebuje geolokacijske podatke z lastnostmi, ki jih potrebujemo za dostop do video posnetkov odsekov. Na desni strani diagrama je prikazan razred WACPlayerConfiguration, ki služi za shranjevanje konfiguracijskih parametrov aplikacije.

Za dostop odjemalca do tega objektnega modela, strežniški del aplikacije implementira spletno storitev z naslednjimi metodami:

- GetOdsekIds, ki služi za iskanje odsekov, saj vrne vse šifre odsekov, ki vsebujejo podan koren,
- GetOdsekInfo, ki služi za nalaganje geolokacijskih podatkov za podano šifro odseka in smer,
- GetAvailableRepositories, ki nam vrne ID-je repozitorijev, v katerih je posnetek odseka na voljo.

Naštete metode predstavljajo vmesnik, prek katerih odjemalec dostopa do vseh podatkov, ki jih potrebuje za predvajanje video posnetkov.

6.3 Strežniški del aplikacije za usmerjanje

Odjemalcu moramo poleg geolokacijskih podatkov o odsekih ter video posnetkov zagotoviti tudi operacije za usmerjanje po cestnem omrežju. Graf za usmerjanje je razmeroma velika podatkovna struktura, zato se mora algoritem usmerjanja izvajati na strežniku.

Objektni model usmerjanja in algoritem je podrobneje predstavljen v podpoglavju 5.5, za uporabo na odjemalcu, pa moramo definirati še vmesnik spletne storitve, za izračun najkrajše poti. Za osnovno uporabo zadostuje že ena metoda z imenom FindPath. Kot parametra metoda potrebuje začetno in končno točko. Metoda najprej poišče najbližji vozlišči (končni in začetni točki), nato pa ju pošlje v algoritem za iskanje najkrajše poti.

6.4 Uporabniška aplikacija za predvajanje posnetkov odsekov

Uporabniška aplikacija je ključen del sistema, saj povezuje vse do sedaj opisane komponente in omogoča njihovo uporabo. Implementiral sem jo aplikacijskem ogrodju (ang. application framework) za razvoj bogatih spletnih aplikacij z imenom Silverlight.

Ogrodje Silverlight je zasnovano kot vtičnik (ang. plugin) za večino modernih brskalnikov. Omogoča nam razvoj naprednih uporabniških vmesnikov z multimedijskimi vsebinami, napredno grafiko, in animacijami. Ogrodje Silverlight vsebuje tudi posebno verzijo ogrodja .NET, ki nam zagotavlja izvajalno okolje za kodo napisano v standardnih programskih jezikih .NET kot so C#, VB.NET ali C++. Zaradi objektne usmerjenosti teh jezikov, so aplikacije Silverlight veliko bolj modularne kot klasične aplikacije HTML/JavaScript. Jeziki .NET so tudi bolj striktni in se prevedejo vnaprej, zato je razvoj aplikacij veliko hitrejši in lažje obvladljiv. Tako kot za razvoj klasičnih aplikacij .NET, se tudi za razvoj aplikacij Silverlight uporablja razvojno okolje Visual Studio 2010.

Aplikacija Silverlight za predvajanje video posnetkov cest je znotraj brskalnika integrirana v spletno aplikacijo GIS, ki prikazuje zemljevid cestnega omrežja. Zemljevid nam omogoča standardno navigacijo po prostoru, poleg tega pa omogoča tudi vnos točk za izračun najkrajše poti.

6.4.1 Uporabniški vmesnik predvajalnika

Uporabniški vmesnik predvajalnika prikazuje slika 20.

Slika 20. Uporabniški vmesnik predvajalnika.

Glavni element vmesnika je seveda prikazovalnik video posnetka, zato se dinamično razteza po celotni površini okna. Glavna orodna vrstica za upravljanje z predvajalnikom se nahaja v spodnjem delu okna. Na njej se nahajajo gumbi z naslednjimi funkcijami:

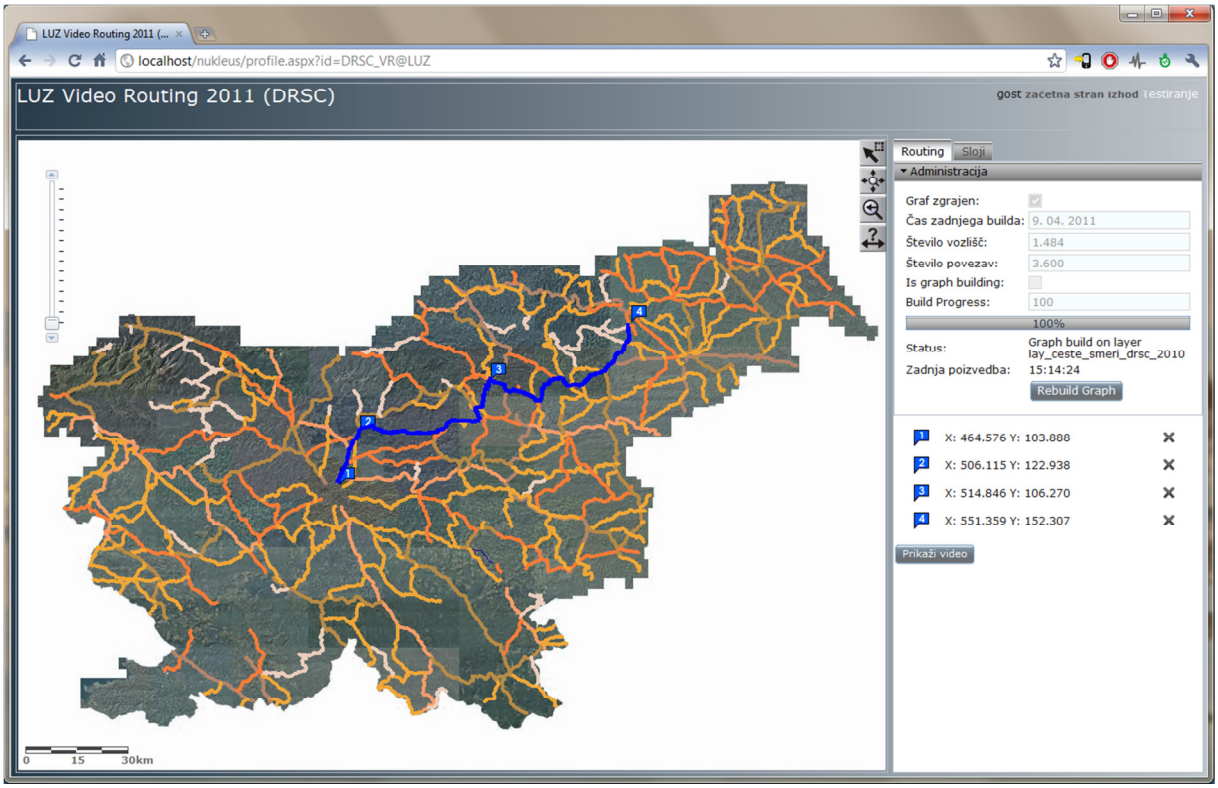
- predvajaj/pavza,
- ustavi,
- obrni smer posnetka v trenutni točki,
- prikaži metrsko mrežo,
- shrani trenutni pogled,
- šest gumbov za relativni premik nazaj/naprej za 50, 10 ali en meter.

Nad gumbi se nahaja drsnik za prikaz in spreminjanje trenutne lokacije na posnetku. Za razliko od klasičnih video predvajalnikov, kjer drsnik prikazuje čas, je v primeru video posnetkov cest kot količino za definicijo trenutne lokacije bolj smiselno uporabiti stacionažo. Celotna površina drsnika se odziva na klike z miško in omogoča skok na stacionažo, ki ustreza lokaciji klika.

Na klike se odziva tudi prikazovalnik posnetkov. S klikom namreč lahko sprožimo dinamično merjenje prečne dolžine objektov na posnetku. Slika 20 prikazuje primer merjenja širine prometnega pasu.

6.4.2 Iskanje najkrajše poti

Funkcionalnost iskanja najkrajše poti, sem implementiral kot razširitev spletnega pregledovalnika GIS. Uporabniški vmesnik je prikazan na sliki 21.



Slika 21. Uporabniški vmesnik za usmerjanje.

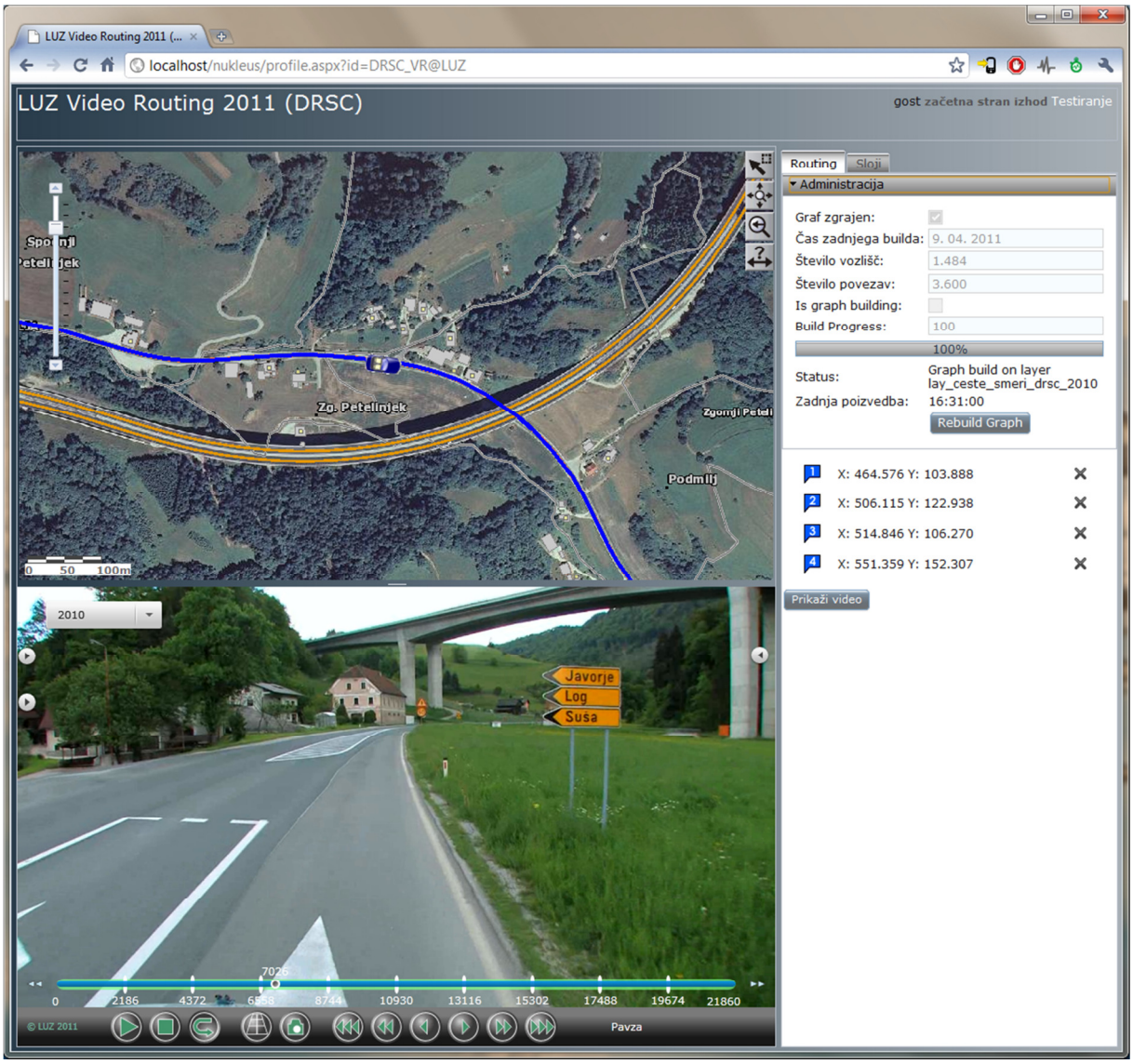
Glavni del okna zavzema interaktivni zemljevid, ki nam prikazuje splošno podlogo Slovenije in osi državnih cestnih odsekov, ki imajo pripadajoč video posnetek. Prek miške, tipkovnice in vertikalnega drsnika nam omogoča hitro navigacijo po prostoru.

Desna stran okna je namenjena vmesniku usmerjanja. V zgornjem delu je prikazana administracijska maska (vidna samo administratorjem), ki prikazuje osnovne podatke o grafu za usmerjanje in omogoča njegovo osveževanje. Pod administracijsko masko je prostor namenjen izpisu trenutno vnesenih točk za iskanje najkrajše poti. Točke iskanja poti, uporabnik vnese s pomočjo klikov z miško po zemljevidu.

Oznake točk na zemljevidu, izračunana linija najkrajše poti in točke izpisane na desni strani, omogočajo funkcionalnost povleci-spusti (ang. drag-drop). To nam omogoča, da lahko že vnesenim točkam na zemljevidu poljubno spreminjamo lokacijo, v desnem spisku pa lahko spreminjamo vrstni red obiskovanja teh točk. Podobno lahko z operacijo povleci-spusti, ki jo začnemo na zemljevidu na izračunani liniji najkrajše poti, med dve obstoječi točki dodamo vmesno točko in si tako prilagodimo izračun najkrajše poti.

Vmesnik usmerjanja zaključuje gumb, ki prikaže video posnetek trenutno izbrane najkrajše poti.

6.4.3 Integracija predvajalnika v spletno aplikacijo GIS



Slika 22. Vmesnik končne aplikacije.

Slika 22 prikazuje vmesnik z vklopljenimi vsemi komponentami. Okno je podobno prikazu na sliki 21, s tem da je v spodnjem delu integriran predvajalnik cestnih video posnetkov. Takšna razporeditev nam omogoča zelo natančno pozicioniranje v prostoru, saj na zgornjem delu vidimo pogled okolice iz ptičje perspektive, na spodnjem pa imamo na voljo realen pogled na cestišče. Aplikacija med predvajanjem, s točnim pozicioniranjem oznake na zemljevidu, vseskozi skrbi za sinhronizacijo pogledov.

Premikanje po izračunani poti je možno tako s spreminjanjem vrednosti na drsniku stacionaže kot z izbiro lokacije na zemljevidu.

7 Sklep

Izdelana programska rešitev predstavlja inovacijo na področju navigacije po cestnem omrežju. Realen pogled na cestišče v času načrtovanja poti vozniku zagotovi veliko boljšo predstavo o tem kaj ga na poti čaka. To je še posebej pomembno pri prevozih tovorov velikih dimenzij, kjer lahko zaradi sprememb v naravi samo iz realnega pogleda ocenimo ali je neka pot oziroma križišče prevozna za tovor.

Rešitev je v pomoč tudi voznikom osebnih avtomobilov, ki se odpravljajo na neznano pot, saj omogoča, da se voznik že doma seznani s pogledom na vsa križišča, kjer bo moral spremeniti smer potovanja. To pripomore tudi k večji varnosti, saj bo vozniku pogled, ki ga bo doživel ob prihodu v križišče znan in se bo lahko hitreje razvrstil na pravi razvrstilni pas.

Na področju usmerjanja s pomočjo video posnetkov cest, je še veliko možnosti za nadaljnji razvoj novih produktov. Za uporabnike bi bila zelo zanimiva in uporabna mobilna naprava z vgrajeno video navigacijo, ki bi bila vgrajena v osebni avtomobil. Omogočila bi ne samo boljšo uporabniško izkušnjo pri navigaciji, ampak tudi jasnejši pogled na cesto v primeru slabih vremenskih razmer (gosta megla, sneg), saj so posnetki snemani vedno v lepem vremenu.

Praktična uporaba v vseh naštetih primerih pa je možna le s točnimi, ažurnimi in celovitimi podatki. V diplomskem delu sem uporabil posnetke državnih cest, kar po podatkih Statističnega urada RS predstavlja približno eno tretjino vseh javnih cest v Sloveniji. Za snemanje državnih cest je bilo skupaj potrebnih 64 snemalnih dni. V primeru, da bi želeli pokriti vse ceste, bi ob enakem tempu potrebovali kar 192 dni. To predstavlja precej velik problem, saj je v enem letu težko zagotoviti 192 delovnih dni z lepim vremenom, poleg tega pa pri takšni količini snemalnih dni, nastajajo tudi razmeroma veliki stroški.

8 Priloge

8.1 Seznam slik

8.2 Seznam tabel

8.3 Seznam kod

9 Literatura

[1] P. Biwas, P. K. Mishra, and N. C. Mahanti, "COMPUTATIONAL EFFICIENCY OF OPTIMIZED SHORTEST PATH ALGORITHMS," *International Journal of Computer Science & Applications*, zv. II, str. 22-37, 2005.

[2] Thomas H. Cormen, *Introduction to algorithms.*: MIT Press, 2001.

[3] DFG CONSULTING. (2010, December) DFG. Dostopno na: http://www.dfgcon.si/index.php/dejavnosti/mobilno-kartiranje

[4] DivXLand.org. (2010, December) Multimedia Container Formats. Dostopno na: http://www.divxland.org/container_formats.php

[5] Open Geospatial Consortium, Inc. (2011, Mar.) Dostopno na: http://www.opengeospatial.org/

[6] Wikipedia authors. (2011, Jan.) Straming_media. Dostopno na: http://en.wikipedia.org/wiki/Streaming_media

[7] Alex Zambelli. (2011, January) Dostopno na: http://alexzambelli.com/blog/2009/02/10/smooth-streaming-architecture/

[8] Ming-hsiang Tsou Zhong-Ren Peng, *Internet GIS.*: John Wiley and Sons, 2003.