ŽILINSKÁ UNIVERZITA V ŽILINE

AUTOREFERÁT

DIZERTAČNEJ PRÁCE

Žilina, apríl 2013

Ing. Štefan Toth

Žilinská univerzita v Žiline
Fakulta riadenia a informatiky

Ing. Štefan Toth

Autoreferát dizertačnej práce

Spracovanie obrazu s využitím dopytov v prostredí VANET

na získanie akademického titulu **„philosophiae doctor“ (v skratke PhD.)**
v študijnom programe doktorandského štúdia
Aplikovaná informatika

v študijnom odbore:
9.2.9 Aplikovaná informatika

Žilina, apríl 2013

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia **na Katedre softvérových technológií, Fakulte riadenia a informatiky Žilinskej univerzity v Žiline.**

Predkladateľ: **Ing. Štefan Toth**
Žilinská univerzita v Žiline
Fakulta riadenia a informatiky
Katedra softvérových technológií

Školiteľ: **doc. Ing. Emil Kršák, PhD.**
Žilinská univerzita v Žiline
Fakulta riadenia a informatiky
Katedra softvérových technológií

Oponenti:

Autoreferát bol rozoslaný dňa: .....

Obhajoba dizertačnej práce sa koná dňa ..... o ..... h. pred komisiou pre obhajobu dizertačnej práce schválenou odborovou komisiou v študijnom odbore **9.2.9 Aplikovaná informatika**, **v študijnom programe Aplikovaná informatika** vymenovanou dekanom Fakulty riadenia a informatiky Žilinskej univerzity v Žiline dňa .....

Predseda odborovej komisie:
prof. Ing. Martin Klimo, PhD.
Žilinská univerzita v Žiline
Fakulta riadenia a informatiky
Univerzitná 8215/1
010 26 Žilina

1 Úvod

Predstavme si, že sa stane banková lúpež a policajti dostanú informáciu o tom, že lupiči unikli na červenom vozidle s evidenčným číslom začínajúcim na ZA a končiacim AB, pričom vpravo od neho bola nálepka vyobrazujúca lebku s prekríženými kosťami. V prípade VANET siete alebo distribuovanej siete zloženej z mobilných telefónov s kamerami sledujúcimi okolie pred vozidlom by bolo možné využiť jazdiace automobily na lokalizovanie vozidla. Požiadavka zadefinovaná na hľadanie červených vozidiel s určitým charakteristickým popisom by sa mohla predať vozidlám jazdiacim v danom okruhu od miesta lúpeže. Automobily by následne spracovávali obraz z kamery a hľadali by zadefinovaný objekt určitý čas. Ak by nejaké vozidlo lokalizovalo hľadaný objekt, vyslalo by o ňom správu so zistenou aktuálnou GPS polohou, smerom pohybu a zachyteným obrázkom.

Iným zaujímavým príkladom môže byť aj zistenie aktuálnych cien pohonných hmôt určitej čerpacej stanice. Vodič by chcel natankovať na nejakej stanici preferovanej značky a chcel by preto zistiť, kde je napr. benzín v okolí najlacnejší. Informácie o cenách sa väčšinou nachádzajú na totemoch čerpacích staníc, preto by bolo zaujímavé rozpoznať totem čerpacej stanice a z jeho obsahu zistiť cenu benzínu. Vozidlo, ktoré by takúto informáciu chcelo získať, by preto vytvorilo požiadavku, v ktorej by bolo definované, čo a ako sa má hľadať (logo požadovanej čerpacej stanice s charakteristickými rysmi a pod ním textové číselné informácie). Následne by sa vyslala požiadavka do siete a ak by nejaké vozidlo zdetegovalo totem a rozpoznalo jeho obsah, mohlo by vrátiť naspäť odpoveď s požadovanými informáciami ako je rozpoznaná cena pohonných hmôt, obrázok totemu a jeho GPS poloha.

Aby sme mohli lokalizovať takéto hľadané objekty, musíme spracovať obraz získaný jednou alebo viacerými kamerami pomocou algoritmov z oblasti analýzy, spracovania obrazu a počítačového videnia. Úlohy je možné riešiť klasicky vytvorením komplexného algoritmu skladajúceho sa z metód, v telách ktorých budú pomocné algoritmy zabezpečujúce zadanú funkcionalitu rozpoznávania. Vytvorenie algoritmu hľadajúceho určitý objekt pre daný účel ale nemusí vyhovovať v takom prostredí akým sú siete VANET, pretože by sa vyžadovalo, aby bol v každom vozidle konkrétny algoritmus nasadený a teda implementovaný. Ak by nebol, bolo by možné zaistiť prenos konkrétneho kódu na koncové zariadenie a zabezpečiť tak jeho vykonanie. Vzniká ale otázka bezpečnosti takéhoto riešenia (potrebný napr. sandbox). Problematická je okrem toho aj heterogénnosť prostredí (rôzne operačné systémy), ďalej veľkosť kódu a s tým súvisiaci čas potrebný na prenos, ktorý je potrebný vo VANET sieti zredukovať, nakoľko komunikácia medzi vozidlami alebo vozidlami a infraštruktúrou je kvôli mobilite vozidiel krátka. Jednoduchším riešením by preto bolo rozdeliť komplexnú úlohu hľadania určitého objektu na menšie časti, t.j. implementovať potrebné základné algoritmy a využiť ich ako stavebné kamene na vyriešenie komplexnej úlohy.

Využitím tejto myšlienky a pre realizovanie uvedených motivačných úloh je potrebné presne špecifikovať, aké objekty, kde, v akej relácií a ako sa majú v obraze vyhľadávať. Navyše, v obraze dopravnej scény sa môžu nachádzať aj netradičné objekty, na ktoré treba brať zreteľ. Realizácia preto nie je taká jednoduchá. Potrebné je tiež vyriešiť, v akom formáte a teda aj štruktúre bude úloha predaná, ako sa má rozdeliť na menšie časti, ako ich vykonať, zlúčiť a transformovať na požadovaný výstup. To bude predmetom skúmania, výsledkom ktorého je návrh systému a jazyka na dopytovanie ľubovoľných objektov z obrázkov pre prostredie VANET. Navrhnutý systém nakoniec nemusí byť využitý len v ňom, ale aj v iných oblastiach, ako bude v závere práce poukázané.

2 Cieľ práce

Hlavným cieľom tejto práce je navrhnúť systém a jazyk na dopytovanie ľubovoľných objektov z obrazu získaného senzormi snímajúcimi dopravnú scénu pred vozidlom v prostredí VANET. Požadované objekty sa budú v obraze automaticky detegovať a rozpoznávať na základe vstupného dopytu.

Systém tak umožní spracovanie komplexných úloh z oblasti spracovania a analýzy obrazu na cestnej sieti pomocou jednoduchých čiastkových úloh a bude definovať:

- množinu základných a priestorových operácií nad obrazom,
- nízkoúrovňové algoritmy spracovania a analýzy obrazu,
- vysokoúrovňové algoritmy spracovania a analýzy obrazu,
- jednotné rozhranie pre spracovávané vstupy a výstupy,
- formát jazyka na dopytovanie.

3 Súčasný stav

Pojem *VANET* (Vehicular Ad hoc Network) predstavuje ad-hoc sieť určenú pre bezdrôtovú komunikáciu vozidiel. Venuje sa jej mnoho výskumníkov a výrobcov automobilov s cieľom vyvinúť štandard na prenos dát a aplikácie, od ktorých sa očakáva zvýšenie bezpečnosti v doprave a to predovšetkým znížením dopravných nehôd a zápch, ušetrením času cestovania a teda aj dosiahnutie zníženie znečistenia ovzdušia a spotreby energie [1]. V súčasnosti sa považuje za jednu z najdôležitejších technológií v oblasti inteligentných dopravných systémov [2].

3.1 Aplikácie používajúce kamery

VANET sieť môže byť nápomocná pri križovatkách alebo pri predbiehaní vozidiel s využitím kamier umiestnených v automobiloch ako uvádzajú autori v [3]. Na príklade križovatky, v ktorej budovy zatieňujú výhľad vodiča v smere jeho odbočovania, môže napomôcť druhé vozidlo, ktoré má výhľad na smer, do ktorého prvé vozidlo odbočuje. Vďaka tomu je možné buď prenášať obraz z tohto druhého vozidla do prvého alebo automaticky identifikovať určité potenciálne objekty ako sú chodci prechádzajúci cez prechod pre chodcov a vďaka tomu upozorniť vodiča na zvýšenú opatrnosť.

V prípade predbiehania vozidiel, ktoré sú veľké alebo majú zatemnené zadné sklo, cez ktoré nie je možné dobre vidieť situáciu pred vozidlom, autori v [4] navrhli protokol pre prenos videa z kamery umiestnenej v predbiehanom vozidle. Vďaka tomu môže mať vodič dokonalý prehľad o dianí pred vozidlom a podľa toho môže bezpečnejšie realizovať úkon predbiehania.

Iným príkladom využitia kamier a spracovania obrazu vo VANET prostredí je lokalizovanie automobilov. Autori Ferreira a kol. [5] navrhli protokol na vyhľadávanie vozidiel s využitím rozpoznávania evidenčných čísiel, po ktorých pátra polícia. Po ich vypátraní sa polícií odošle GPS poloha prostredníctvom zabezpečeného prenosu, aby ho nebolo možné podvrhnúť.

3.2 Zhodnotenie

VANET siete sú perspektívnou budúcnosťou automobilového priemyslu, nakoľko poskytujú obrovské možnosti, či už z hľadiska zvýšenia bezpečnosti alebo komfortnosti používateľov. V oblasti spracovania a analýzy obrazu bolo navrhnutých niekoľko aplikácií. Všetky aplikácie sú ale špecifické tým, že riešia určitý problém presne vymedzenej oblasti, pre ktorú boli navrhnuté. Z nášho prieskumu doposiaľ nebolo zistené žiadne existujúce riešenie, ktoré by

bolo schopné vyhľadávať všeobecné informácie – objekty nachádzajúce sa v obraze na požiadanie.

Vyhľadávanie obrazu pomocou QBIC / CBIR systémov je založené na vyhľadávaní v databázach naplnených už existujúcimi dátami, čo sú príznaky získané z obrazu, ako napr. farebné, tvarové alebo textúrové informácie, či iné príznaky popisujúce lokálne alebo globálne časti obrazu. Istá podobnosť je tu s našou úlohou, avšak jej cieľom nie je vyhľadávať v databáze, ale na aktuálne získanom obrázku z kamery v reálnom čase. Čiastočné riešenie by ponúkal štandard MPEG-7, ktorý najskôr popisuje to, čo sa nachádza na obraze pomocou deskriptorov do dokumentu XML, v ktorom je následne možné vyhľadávať. Napriek tomu, že je možné pomocou DDL a DS použiť vlastné deskriptory s vlastnou definíciou a tak upraviť XML dokument na svoj vlastný obraz, nerieši sa všeobecné vyhľadávanie presne toho, čo by sme v konkrétnom čase požadovali. MPEG-7 umožňuje totiž vyhľadávať len to, čo bolo už dopredu popísané.

Z tohto dôvodu sme sa rozhodli navrhnúť nový systém s jazykom na dopytovanie akýchkoľvek informácií z obrazových dát získaných z jednej alebo viacerých kamier umiestnených vo vozidle v reálnom čase s využitím algoritmov spracovania a analýzy obrazu. Náš návrh, výsledky a závery predkladajú nasledujúce kapitoly.

4 Riešenie

4.1 Vymedzenie problému

Keďže oblasť platnosti riešenia je prostredie VANET a obraz snímaný z pohybujúcich sa vozidiel, zameriame sa hlavne na objekty v ňom nachádzajúce, t.j. *objekty dopravnej scény*, ako sú účastníci cestnej premávky, infraštruktúra, prostredie atď. Dôležité je identifikovať všetky elementárne a najčastejšie sa vyskytujúce objekty. Okrem nich sa na dopravnej scéne môže objaviť čokoľvek vrátane objektov, ktoré nie je možné predpokladať dopredu pri návrhu systému (napr. žirafa, lietadlo na ceste, atď.). Preto aj takéto objekty budeme pri návrhu brať do úvahy.

Každý reálny objekt má svoje *charakteristické vlastnosti*. Vďaka nim je možné objekty medzi sebou rozlišovať a porovnávať. Či už sú to konkrétne vlastnosti špecifické pre reálny objekt alebo všeobecné (ako je rozmer, farba, tvar alebo textúra), ktoré sa radia z pohľadu analýzy obrazu do kategórie nízkoúrovňových príznakov. Niektoré objekty môžu mať medzi sebou určitý vzťah, či už *priestorový* (pozícia v obraze globálne a relatívne k iným objektom) alebo aj *časový* (videosekvencia – postupnosť obrazov, v ktorej je jednak vozidlo pohybujúci sa objekt medzi inými objektmi, ktoré sa môžu tiež pohybovať).

Ak identifikujeme objekty, ich vlastnosti a vzťahy medzi nimi, môžeme ich využiť pri tvorbe a návrhu *jazyka na dopytovanie* z obrazu snímajúceho dianie pred vozidlom. Spracovávaný obraz pritom môže byť vo forme jedného snímku alebo videosekvencie. V tejto práci sa obmedzíme len na spracovanie jedného snímku, avšak budeme môcť použiť aj videosekvenciu bez časových závislostí. Navrhnutý dopytovací jazyk by preto mal umožniť nasledujúce operácie:

- rozpoznávanie vysokoúrovňových objektov,
- rozpoznávanie nízkoúrovňových objektov,
- základné operácie nad objektmi a ich vlastnosťami,
- priestorové operácie,
- dĺžku a podmienku ukončenia vykonávania dopytu.

Nielen operácie, ale aj formát je pritom dôležitý, keďže je navrhovaný pre prostredie VANET sietí, v ktorých komunikácia môže prebiehať v krátkom čase, nakoľko vozidlá sú dynamické objekty rýchlo sa pohybujúce a míňajúce na dopravnej sieti. Dôležitá je preto tiež minimalizácia veľkosti prenášaných dát. *Definovanie formátu pre dopytovanie* bude hlavným predmetom výskumu, ktorého správny návrh umožní spracovanie vyššie uvedených operácií. Formát by mal mať nasledujúce vlastnosti:

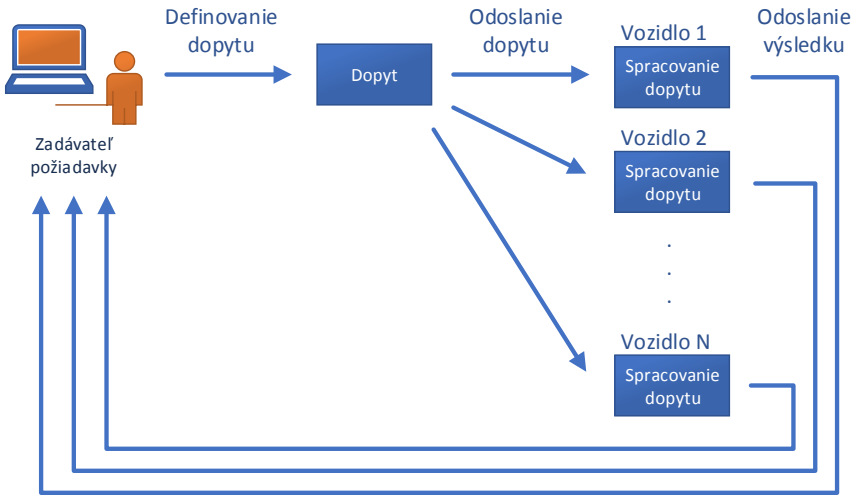
- kompaktnosť formátu a minimalizácia jeho veľkosti na dopytovanie,
- jednoduché vyjadrenie základných a komplexných operácií nad obrazom,
- univerzálnosť a jednoduchá rozšíriteľnosť syntaxe (možnosť pridania nových funkcií a rozšírení).

Formát by mal byť definovaný nielen pre dopyt (vstupný formát požiadavky), ale aj pre možné odpovede. Preto je nutné *definovať výstupný formát odpovede* dopytu.

Kladenou požiadavkou na vozidlá je, aby boli vybavené aspoň jednou kamerou snímajúcou priestor pred vozidlom. Okrem kamery by bolo vhodné využiť aj ďalšie senzory, ktoré by pomohli jednak rozpoznávacím algoritmom a hlavne k rozšíreniu možností dopytov, ako napr. GPS senzor, akcelerometer, gyroskop atď. GPS senzor by mal byť vo vozidlách v prostredí VANET vždy k dispozícii, takže predpokladáme jeho možné použitie, najmä pre získanie takých informácií ako je pozícia, rýchlosť a smer pohybu.

Pre zhrnutie celého princípu dopytov nad obrázkami uvádzame nasledovný postup v krokoch znázornený aj na obr. 1 nasledovne:

- *Definovanie dopytu* – vytvorenie dopytu *zadávateľom* (buď manuálne človekom alebo automaticky aplikáciou). V dopyte je definovaný hľadaný objekt záujmu, požadovaný výstup a dĺžka spracovania.
- *Odoslanie dopytu vozidlám* – dopyt sa odošle na jedno alebo viacero vozidiel (*spracovateľov dopytov*) v závislosti od definovanej cieľovej oblasti, v ktorej sa má spracovávať obraz (oblasť je definovaná napr. geocastom).
- *Spracovanie dopytu a vrátenie výstupu* – spracovanie je realizované vo vozidle buď *okamžite* po prijatí požiadavky *na aktuálnej snímke* získanej z kamery alebo na *videosekvencií určitú dobu* definovanú taktiež v dopyte. Dĺžka spracovania tak môže byť ohraničená buď konkrétnym časom (napr. 15 minút), udalosťou (napr. nájdenie požadovaného objektu, zastavenie chodu vozidla) alebo geolokačným rozsahom v priestore (vymedzenie priestoru platnosti udanej GPS pozíciou).



Obr. 1 Náčrt dopytovania od jeho definovania po vrátenie výstupu

4.2 Obrazové objekty

Základom práce s dopytmi budú objekty, nakoľko predstavujú predmet nášho záujmu. Môžu to byť *reálne objekty*, ako sú budovy, cesty, ľudia, dopravné značky alebo *objekty v tvare segmentov* predstavujúce časti obrazu podľa určitej charakteristiky, napr. oblasti obsahujúce červenú farbu alebo oblasti majúce trojuholníkový, či iný tvar.

Za účelom identifikácie čo najväčšieho možného počtu reálnych objektov sme vykonali niekoľko desiatok jázd v okolí Žiliny. Na základe spracovaní a vyhodnotení týchto videonahrávok sme identifikovali niektoré významné objekty:

1. *Obloha* a s tým súvisiace *osvetlenie* a *stavy počasia*
2. *Hory*
3. *Cesta* (vozovka, krajnica, jazdné pruhy, chodníky) *a križovatky*
4. *Vodorovné dopravné značky* na cestách
5. *Zvislé dopravné značky*
6. *Svetelné signalizačné zariadenia* – semafory
7. *Texty*
8. *Vozidlá*
9. *Ľudia*
10. *Zvieratá*
11. *Budovy a stĺpy*
12. *Stromy, kríky*
13. *Voľné plochy*
14. *Vodné plochy*
15. *Pohybujúce sa objekty vo vzduchu*

Všeobecný objekt

Na to, aby sme mohli rozpoznávať neznáme objekty, definujeme tzv. *všeobecný objekt*, ktorý bude pozostávať z modelu skladajúceho sa z množiny šablón s popisom. Tento model potom bude *priamo začlenený do dopytu*, aby sa naňho mohlo odkazovať a v dopyte používať.

Pre použitie modelu v dopyte máme dve možnosti:

- Implementačne nezávislý – k spracovateľovi (vozidlu vykonávajúceho dopyt) sa odošle jeden alebo viacero zdrojových obrázkov reprezentujúcich pohľady na objekt. Na strane spracovateľa sa potom uskutoční algoritmus, pomocou ktorého sa bude detegovať (výber vhodného algoritmu je na spracovateľovi).
- Implementačne závislý – k spracovateľovi sa neodošlú kompletné obrázky, ale iba ich popisy (vytvorené napr. algoritmami ako je SIFT, MPEG-7 deskriptory, či inými), prípadne vopred natrénovaný model nejakej učiacej sa siete (napr. neurónová sieť, v tomto prípade by sa odoslal model siete s váhami jednotlivých neurónov). Toto riešenie vyžaduje zhodne použité algoritmy na obidvoch stranách (zadávateľovi i spracovateľovi).

Najjednoduchším riešením by bola prvá možnosť, pretože výber algoritmu nebude závisieť na obidvoch stranách, ale iba na strane spracovateľa, ktorý môže implementovať ľubovoľný algoritmus, ktorý bude univerzálny a bude dosahovať dobré výsledky. Na druhej strane, toto riešenie vyžaduje prenos jedného alebo viacerých obrázkov, od ktorých veľkosti záleží čas prenosu. Obidve riešenia majú svoje výhody i nevýhody, preto by sme *odporúčali použiť spôsob* v závislosti od potreby. Ak bude potrebná rýchlosť, je možné využiť implementačne závislé algoritmy, pričom je nutné definovať aspoň jeden.

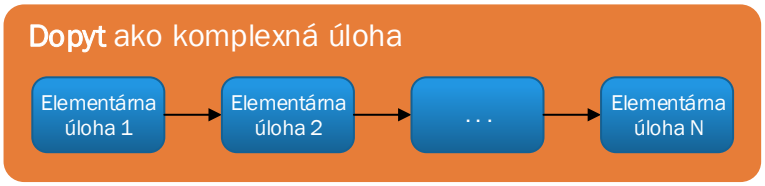
Po uskutočnenej analýze navrhujeme, aby model všeobecného objektu obsahoval nasledujúce časti:

- *Unikátny identifikátor modelu* za účelom odkazovania sa naň v dopyte.
- *Vlastnosti objektu* – deformovateľnosť, statickosť, atď.
- *Pohľady na model*, ktoré budú vyjadrené buď formou *množiny zdrojových obrázkov*, *príznakov* alebo *modelom učiacej sa siete*. Aby bolo možné objekt rozpoznať, zdrojové obrázky by mali reprezentovať pohľady na neho z rôznych strán (najmä ak nie je objekt symetrický) a rovnako zachytávať aj jeho rôzny vzhľad v prípade deformovateľnosti a to najmä v takých pózach, v ktorých môže byť najčastejšie videný.
- *Popísané časti* modelu, z ktorých sa objekt skladá (napr. ak máme pohľad na zadnú časť vozidla, vidíme kufor, pravé a ľavé svetlo, časť pravého a ľavého kolesa, atď.). Časti budú definované pre každý pohľad a budú mať danú pozíciu, aby sa na ne mohlo v dopytoch opäť odkazovať a napr. zisťovať, či sú viditeľné alebo v relácií s inými objektmi.

Okrem toho by model mohol obsahovať nielen popis pomocou jedného spôsobu, ale aj viacerých. Potom by bolo na spracovateľovi, ktorý zo spôsobov si vyberie. Ak napr. vie pracovať so SIFT-om a bude definovaný v modeli, môže ho využiť. Ak bude obsahovať lepší algoritmus detekcie, môže využiť zdrojové súbory obrázkov, pokiaľ budú v dopyte priložené.

4.3 Systém na dopytovanie

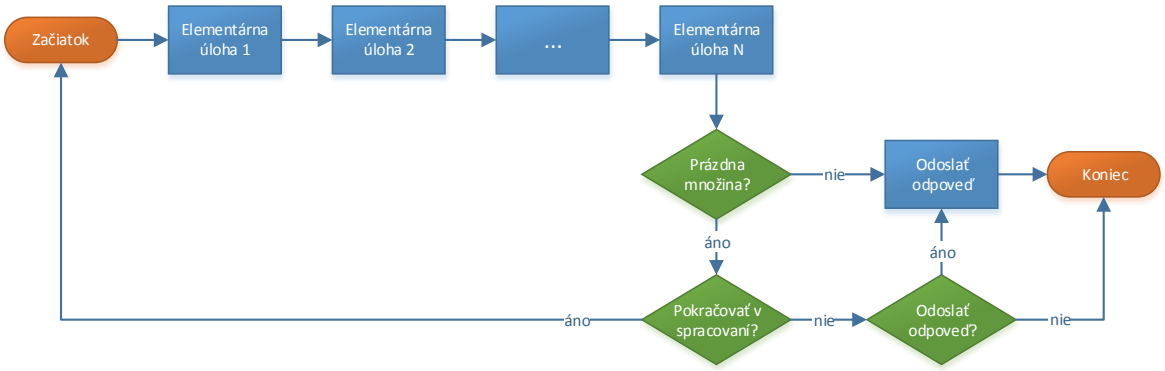
Po identifikácií obrazových objektov si v tejto časti predstavíme princíp a návrh systému na dopytovanie, v ktorom tieto objekty, ich vlastnosti a vzťahy s inými objektmi vyjadríme pomocou dopytov, ktoré sa následne budú môcť spracovávať vo vozidlách.



Obr. 2 Dopyt ako komplexná úloha zložená z postupnosti jednoduchých úloh

Na *dopytovanie* obrazových objektov sa môžeme pozerať ako na *komplexne zloženú úlohu*, ktorú môžeme dekomponovať na jednoduchšie úlohy a tie medzi sebou prepojiť (obr. 2). Vďaka týmto prepojeniam sa budú predávať množiny obrazových objektov v smere zľava doprava, pričom sa tieto množiny budú spracovávať práve v elementárnych úlohách. Ak sa dostaneme na poslednú elementárnu úlohu, záleží od jej výstupu, čo sa ďalej stane (obr. 3). Ak je jej výsledok neprázdna množina, odošle sa odpoveď v požadovanom tvare definovanom na konci dopytu. Ak bude ale výsledok prázdna množina, záleží od definície dopytu, či sa bude dopyt znova spracovávať alebo sa ukončí a ak sa ukončí, či sa má odoslať odpoveď. Takéto správanie sme zvolili z toho dôvodu, že je často potrebné v obraze vyhľadávať nejaký objekt, ktorý ale nemusí byť hneď prítomný. Preto umožňujeme cyklické spracovanie dopytu, až dokiaľ sa nedostane odpoveď, čo reprezentuje neprázdna množina. Prázdna množina potom bude reprezentovať neúspešný výsledok (t.j. nenájdenie požadovaného objektu).

Spracovanie obrazového dopytu teda znamená vykonanie elementárnych úloh zapojených v sekvenčne zreťazenej postupnosti. *Elementárna úloha* pritom predstavuje akúkoľvek základnú operáciu nad obrazom, čo môže byť forma detekcie, lokalizácie alebo klasifikácie objektov, relácií medzi nimi, výberom a inými základnými algoritmami spracovania obrazu.



Obr. 3 Postup spracovania dopytu od jeho začiatku až po koniec

Vstup

Vstup je definovaný zadávateľom dopytu. Je v ňom nutné určiť, *čo sa má vykonať*, *kedy* sa má *spracovanie ukončiť* a *čo sa má poslať* ako odpoveď. Takéto informácie je možné zapísať v nejakom definovanom formáte, napr. v XML, JSON alebo iných. Keďže budeme pracovať vo VANET prostredí, formát by mal byť kvôli prenosu čo najkompaktnejší a najmenší. Naším cieľom ale nebude presne stanoviť formát, v akom sa budú dáta prenášať po sieti, ale iba formát dopytu, v akom ho budeme spracovávať (t.j. čo a aké časti sú potrebné v dopyte). Z toho dôvodu definujeme pre jednoduchšie popísanie vstupov i výstupov povinné a voliteľné dáta pomocou nasledujúceho nami navrhnutého zápisu:

```
INPUT IdentifikátorDopytu
{
    // Definícia modelov - nepovinná položka
    MODEL UnikátnyNázovModeluObjektu1 // voliteľné
    {
        ... // definícia modelu
    }
    MODEL UnikátnyNázovModeluObjektu2 // voliteľné
    {
        ... // definícia modelu
    }
    MODEL UnikátnyNázovModeluObjektuN // voliteľné
    {
        ... // definícia modelu
    }

    // Podmienka ukončenia vykonávania - nepovinná položka
    TERMCONDITION
    {
        ... // ukončovacia podmienka
    }

    // Vyžadovanie odpovede - nepovinná položka
    REQANSWER = true/false; // implicitne true

    // Dopyt - povinná položka
    QUERY
    {
        // definícia dopytu
    }
}
```

Jednotlivé položky pritom znamenajú:

- `INPUT` – definuje jednoznačný názov dopytu, ktorý sa použije aj vo výstupnej odpovedi.

- `MODEL` – definícia všeobecného modelu objektu na rozpoznávanie. Definícia modelu je nepovinná, pokiaľ nebudeme vyžadovať rozpoznávanie objektu, ktorý je neznámy a v systéme nedefinovaný.
- `TERMCONDITION` – ukončovacia podmienka dopytu. Je nepovinná, pokiaľ sa bude jednať o dopyt s rýchlou odpoveďou (spracovanie iba jedného obrázka aktuálne získaného z kamery) a nutná v prípade, že chceme cyklicky spracovávať obraz (videosekvenciu). Zadaná podmienka potom určuje, kedy sa cyklus spracovávania ukončí.
- `REQANSWER` – určuje, či požadujeme vždy odpoveď, aj keď bude dopyt neúspešný (to je vtedy, ak bola vrátená prázdna množina).
- `QUERY` – definícia dopytu. Povinná položka, ktorá tvorí jadro a umožňuje definovať, čo sa má na obraze spracovať a aké objekty s vlastnosťami sa majú získať. Posledná elementárna úloha bude definovať výstup dopytu. Zloženiu dopytu a možným operáciám sa budeme podrobnejšie venovať v nasledujúcich častiach.

Výstup

Výsledkom dopytu môže byť čokoľvek, čo požaduje zadávateľ, napr. *vrátenie pôvodného obrázka* zachyteného kamerou alebo z neho iba *výsek*, na ktorom sa nachádza jeden alebo viacero objektov záujmu, *GPS súradnica*, *obyčajná hodnota* (číslo, text, pravdivostná hodnota) obsahujúca požadovanú informáciu (počet objektov, rozpoznaný text, EČV, atď.) alebo aj ich kombinácie. Výsledok spracovania môže byť vyjadrený v nasledujúcom tvare:

```
OUTPUT UnikátnyIdentifikátorDopytu
{
    // Odpoveď dopytu - povinná položka
    ANSWER
    {
        // ľubovoľné dáta
    }
}
```

Obrazový objekt

Základom práce s dopytmi je *obrazový objekt*, ktorý bude vstupom i výstupom elementárnych úloh. Predstavuje pritom nízkoúrovňový alebo vysokoúrovňový objekt v obraze. Aby sme ho mohli použiť v elementárnych úlohách, v tejto časti si podrobne definujeme jeho štruktúru a konkrétne vlastnosti, ktoré by mal obsahovať.

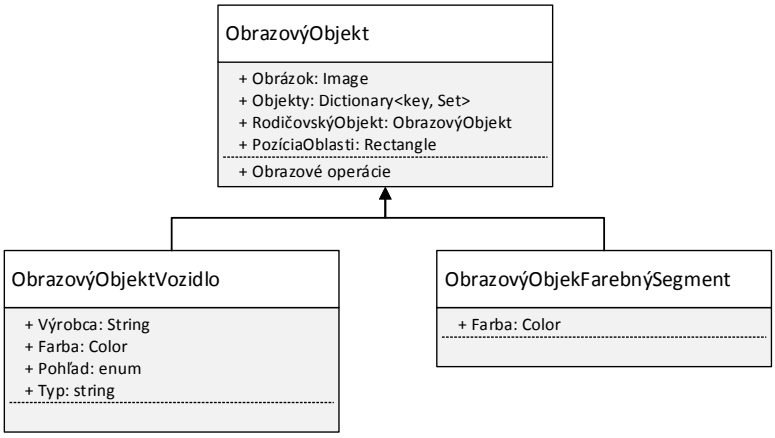
Obr. 4 Obrazový objekt z globálneho pohľadu

Obrazový objekt sa z hľadiska svojho obsahu skladá z *obrázka* a *dát*. Obrázok reprezentuje pohľad na dopravnú scénu alebo výsek reálneho, či nízkoúrovňového objektu dopravnej scény. Dáta popisujú hlavne objekt a môžeme ich rozdeliť na nasledujúce dva typy:

- *špecifické dáta objektu* – vlastnosti, ktoré popisujú konkrétny obsah obrázka. Pre každý obrazový objekt budú odlišné (napr. pre nízkoúrovňový objekt segmentov to bude farba oblastí po segmentácií, pre vozidlo zase vlastnosti ako značka výrobcu, pohľad, typ, EČV, farba atď.).

- *pomocné dáta* – umožňujú orientáciu v obraze a v jeho častiach. Identifikujeme tieto základné vlastnosti, ktoré bude obsahovať každý odvodený obrazový objekt:
    - *Objekty* – kolekcia párov typu kľúč - hodnota, ktorá umožňuje uchovávať obrazové objekty, ktoré boli v obraze získané. Kľúčom je typ objektu (napr. vozidlo, dopravná značka, lebka, atď.) a hodnotou množina obrazových objektov.
    - *Rodičovský objekt* – odkaz na rodiča, ktorým je iný obrazový objekt. Ak nemá rodiča ako je prípad koreňového obrázka dopravnej scény, bude mať prázdnu hodnotu.
    - *Pozícia oblasti* – obdĺžnik výseku o absolútnych súradniciach v hlavnom koreňovom obrazovom objekte.

Príklad uvedenej štruktúry obrazového objektu zapísaného v UML diagrame tried spolu s ďalšími dvoma odvodenými triedami zobrazuje obr. 5. Prvé dve základné vlastnosti sú tu zvolené z dôvodu, že je potrebné niekde uchovávať dáta a okrem toho sa po obraze pohybovať a s dátami manipulovať. Ak sa pozrieme na ľubovoľný obrázok, nachádzajú sa na ňom rôzne objekty, ktoré obsahujú ďalšie objekty a tie môžu obsahovať ďalšie a ďalšie. Takže je možné vidieť, ako sú objekty vnorené v rámci hierarchie iných objektov. Z toho dôvodu sme zvolili stromovú štruktúru, ktorá nám nielenže umožní uchovávať jednotlivé obrazové objekty ako vrcholy stromu na jednotlivých úrovniach, ale umožní aj pohyb a spracovanie objektov, ich vlastností, relácie a ďalšie operácie.

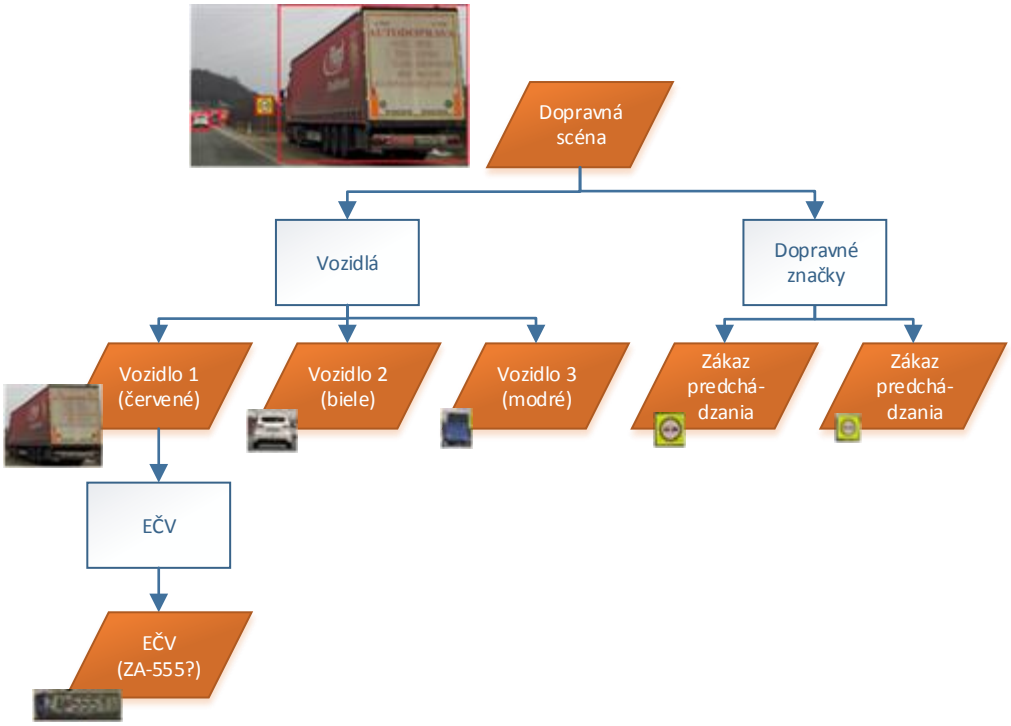


Obr. 5 Obrazový objekt ako základná trieda, od ktorej dedia konkrétne triedy objektov

Na obr. 6 je znázornená jednoduchá *stromová štruktúra obrazových objektov*. Obrázok dopravnej scény je takisto obrazový objekt, ktorý je koreňom a obsahuje ďalšie obrazové objekty (vozidlá a dopravné značky). Dopravná scéna sa tak skladá z kolekcie obrazových objektov, ktoré sú usporiadané v dvoch typoch kľúčov – *vozidlá* a *dopravné značky*. V nich sa nachádzajú zoznamy konkrétnych obrazových objektov, čo sú v príklade obrázka pre vozidlá tri inštancie vozidiel (červené, biele, modré) a v prípade dopravných značiek dve značky zákazu predchádzania na vyskytujúcich sa rôznych pozíciách v obraze. V prvom vozidle sa navyše zdetegovalo a rozpoznalo evidenčné číslo vozidla, na ostatných vozidlách sa EČV nenašli z dôvodu nízkeho rozlíšenia.

Okrem definovaných vlastností bude mať obrazový objekt veľké množstvo základných metód, t.j. funkcií, ktoré môžu manipulovať a upravovať obrázok. Jedná sa napr. o úpravu jasu, vyhladzovanie Gaussom, mediánom a inými, ktoré boli popísané v teoretickej časti dizertačnej práce. Popri nich spomenieme ešte jednu z používaných funkcií, ktorou bude *Výsek*. Touto operáciou bude možné orezať z obrázka jeho časť podľa zadaných súradníc.

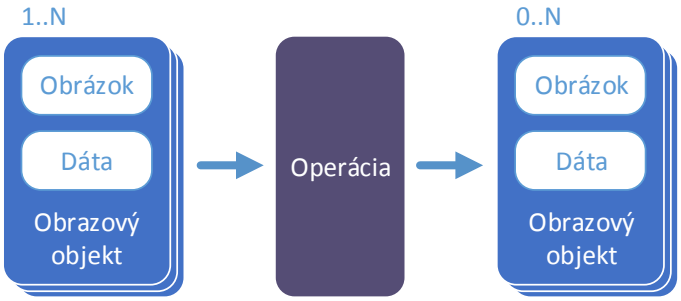
Vďaka tomu môžeme pracovať s malými časťami obrazu a tak napr. ušetriť výpočtový čas namiesto spracovania na celom obrázku.



Obr. 6 Strom obrazových objektov (oranžové obdĺžniky predstavujú obrazové objekty, biele obdĺžniky typy)

Operácie

V dopyte ako komplexnej úlohe zloženej z *elementárnych úloh* predstavujú tieto úlohy prevažne *operácie nad obrazovými objektmi* a ich *vlastnosťami*. Ich vstupom a rovnako aj výstupom bude preto usporiadaná množina obrazových objektov, v ktorej môžeme obrazové objekty vyhľadávať, upravovať, spájať atď.



Obr. 7 Princíp spracovania elementárnej úlohy operáciou so vstupmi a výstupmi

Zavádzame nasledujúce operácie:

- Vstupné a výstupné funkcie:
  - GETIMAGE – zachytí obraz z kamery, prevedie ho do obrazového objektu a vráti. Okrem jedného môže vrátiť aj viac obrázkov a z rôznych kamier. Zachytený obrázok predstavuje dopravnú scénu a ako dáta obrazového objektu ho tvoria dáta popisujúce pohľad kamery, z ktorej sa obrázok získal v danom čase.
  - RESULT – vráti výslednú množinu objektov definovanú lambda výrazom ako výstup celého dopytu.

- Funkcia hľadania objektov:
    - FINDOBJECTS – deteguje alebo rozpozná požadované obrazové objekty vo vstupných obrazových objektoch. Okrem toho, že vráti výslednú množinu obrazových objektov, pre každý obrazový objekt uloží nájdené objekty do kolekcie obrazových objektov za účelom vytvárania hierarchie stromovej štruktúry.
- Manipulačné operácie:
    - WHERE – selekcia je operácia, pomocou ktorej filtrujeme vstupné obrazové objekty na základe podmienky zadanej lambda výrazom.
    - SELECT – výber je operácia, pomocou ktorej vyberáme množinu obrazových objektov zo vstupnej množiny obrazových objektov na základe vlastností. To, čo sa má vyberať, definujeme lambda výrazom.
- Množinové operácie:
    - UNION – zjednocuje dve množiny obrazových objektov do jednej.
    - INTERSECTION – predstavuje prienik dvoch množín obrazových objektov.
    - MINUS – uskutočňuje rozdiel dvoch množín obrazových objektov.
- Operácie s dočasnými dátami:
    - SAVE – uloží množinu obrazových objektov do dočasného úložiska.
    - LOAD – načíta množinu obrazových objektov z dočasného úložiska.
- Priestorové operácie:
    - RIGHTOF – vyberá také obrazové objekty, ktoré sú vpravo od iných obrazových objektov.
    - LEFTOF – vyberá také obrazové objekty, ktoré sú vľavo od iných obrazových objektov.
    - ABOVE – vyberá také obrazové objekty, ktoré sú nad inými obrazovými objektmi.
    - BELOW – vyberá také obrazové objekty, ktoré sú pod inými obrazovými objektmi.
    - IN – vyberá také obrazové objekty, ktoré sú vo vnútri iných obrazových objektov.
    - OUT – vyberá také obrazové objekty, ktoré sú mimo iných obrazových objektov.
    - NN – operácia najbližšieho suseda vyberá jeden alebo viacero prvkov vstupnej množiny, ktorých vzdialenosť ku všetkým obrazovým prvkom druhej množiny je najmenšia.
    - DISTANCE – vyberá také prvky množiny, ktorých vzdialenosť medzi všetkými prvkami voči všetkým obrazovým prvkom druhej množiny je zadaný lambda výrazom.
- Iné funkcie, ktoré nepracujú s obrazovými objektmi a ktoré sa môžu použiť buď vo vnútri lambda výrazov alebo na konci dopytu:
    - GPS – poskytuje základné informácie z GPS ako je zemepisná šírka, dĺžka, výška a iné.
    - QUERYINFO – poskytuje informácie o dopyte, čase jeho spracovania, podporované objekty na rozpoznávanie ako aj popisné metódy, ktoré poskytuje za účelom rozpoznávania všeobecných objektov.

Lambda kalkul

Nakoľko nosnou časťou mnohých funkcií je manipulácia buď s obrazovým objektom, jeho vlastnosťami alebo inými objektmi, v realizácií dopytov vychádzame z objektového lambda kalkulu zavedenom v [6] a syntaxou upravenou z časti podľa jazyka C#. Použitie lambda

výrazov nám poskytne flexibilitu a umožní jednoducho vyjadriť ľubovoľné operácie s objektmi. V mnohých operáciách je totiž nutné spracovať množinu objektov, preto pomocou lambda výrazu definujeme vlastnú funkciu, ktorej volanie zabezpečí aplikovanie výrazu na každý prvok množiny.

Lambda výraz budeme zapisovať v nasledujúcom tvare:

```
hlavička => telo
```

Skladá sa z troch častí:

- „hlavička“ – predstavuje zoznam vstupujúcich premenných, ktoré sa zvyknú označovať ako lambda-premenné. V našom prípade sa bude väčšinou jednať o obrazové objekty, preto bude najčastejšie jedna premenná typu obrazový objekt.
- „=>” – lambda operátor, ktorým oddeľujeme časť hlavičky od tela. V klasických zápisoch sa najčastejšie používa bodka „.“ alebo zvislá čiara „|“. Keďže obidva znaky budeme používať pre iné účely, zaviedli sme zložený znak skladajúci sa zo znaku rovná sa a väčší ako, ktorým sme sa inšpirovali práve z jazyka C#.
- „telo“ – v tejto časti sa bude nachádzať zápis toho, čo sa má vykonať. Výsledná hodnota sa získa po beta-redukcii, t.j. aplikácii lambda výrazu.

Jednoduchým príkladom lambda výrazu môže byť napr.:

```
obloha => obloha.StavPočasia
```

Obloha je obrazový objekt, ktorý v hlavičke ako jeden parameter vstupuje do tela funkcie lambda výrazu. V ňom voláme vlastnosť stav počasia, ktorý vráti hodnotu stavu počasia, ktorá sa stane výsledkom lambda výrazu.

Iným príkladom môže byť:

```
ečv => ečv.Text.ZačínaNa('ZA') AND ečv.Text.DĺžkA = 5
```

V tomto prípade vstupuje do lambda výrazu cez premennú `ečv` obrazový objekt evidenčného čísla vozidla, ktorý sa využíva v tele lambda výrazu na zistenie, či jeho vlastnosť Text začína na písmená ZA a súčasne či jeho dĺžka je rovná 5 znakom. Výsledkom lambda výrazu je preto logického hodnota vyjadrujúca pravdivosť uvedeného výrazu.

Zloženie dopytu a operátor |

Dopyt môžeme vyskladať pomocou definovaných operácií, ktoré budú sekvenčne prepojené v jednom riadku. Na prepojenie jednotlivých krokov si zavedieme operátor „|“ (zvislá čiara), ktorým budeme nielenže oddeľovať jednotlivé kroky, ale aj predávať výstupy na vstupy nasledujúcich operácií:

Vstupom a výstupom sú teda tie isté obrazové objekty bez akejkoľvek modifikácie. Výsledné zapojenie viacerých príkazov a teda operácií nad obrazom bude vyzerať nasledovne:

operácia1 | *operácia2* | ... | *operáciaN*

pričom posledná operácia *operáciaN* bude udávať celkový výstup dopytu.

4.4 Príklady dopytov

Na demonštráciu použitia navrhnutého jazyka pre systém sekvenčného spracovania dopytu uvádzame niekoľko príkladov.

Obrázok dopravnej scény

Vrátenie jedného obrázka z prednej kamery predstavuje najjednoduchší dopyt, aký môžeme vyjadriť:

```
GETIMAGE
```

Formálne táto operácia vracia obrazový objekt, ktorý zaobaľuje získaný obrázok z kamery. Pre jednoduchosť v tejto časti budeme používať pojem obrázok ako synonymum pre obrazový objekt.

Ak by sme požadovali obraz zo zadnej kamery, môžeme použiť:

```
GETIMAGE('zadná')
```

V týchto jednoduchých dopytoch nemusíme použiť výstupnú operáciu `RESULT`, ani operátor |, pretože nepožadujeme vrátiť iné dáta ako obrázok a nepoužívame ani iné operácie. Ak by sme však chceli vrátiť s obrázkom aj GPS pozíciu, použijeme výstupnú operáciu `RESULT`:

```
GETIMAGE('zadná') | RESULT(obrázok => obrázok, GPS.Pozícia)
```

Získaný obrázok sa predá na vstup operácie `RESULT`, v ktorej sa definujú položky, aké požadujeme vrátiť. Konkrétne definujeme vrátenie obrázka a GPS pozíciu.

Hľadanie vozidla

Iným príkladom je vyhľadávanie červeného vozidla. Dopyt preto môžeme postaviť tak, že na celom obraze vyhľadáme vozidlá a pozrieme sa na ich farbu alebo si najskôr nájdeme červenú farbu a na takto získaných miestach segmentov vyhľadáme vozidlá. Ukážeme si obidva spôsoby riešenia.

V oboch prípadoch potrebujeme získať obraz z kamery a následne v ňom vyhľadať objekty. Použijeme pritom prednú kameru. Prvý spôsob:

```
GETIMAGE | FINDOBJECTS('vozidlo') |
WHERE(vozidlo => vozidlo.Farba = 'červená')
```

Vyjadrený dopyt znamená získaj obrázok dopravnej scény z prednej kamery (`GETIMAGE`), na ktorom vyhľadaj vozidla (`FINDOBJECTS`) a následne z množiny vozidiel vyfiltruj tie, ktorých farba je červená (`WHERE`). Výstupom je teda množina výsekov červených vozidiel.

Ak sa na tento zápis pozrieme podrobnejšie, `GETIMAGE` vráti obrazový objekt reprezentujúci obrázok dopravnej scény z kamery. Ten sa predá funkcií `FINDOBJECTS`, ktorá vyhľadá v tomto obrazovom objekte všetky vozidlá a vráti ich ako množinu obrazových objektov reprezentujúcich všetky nájdené vozidla na obraze (môže ich byť nula alebo viac). Keďže požadujeme získať len červené vozidlá, uskutočníme následne operáciu selekcie (`WHERE`), v ktorej pomocou podmienky stanovíme, že dominantná farba vozidla má byť červená. Nakoľko vyžadujeme od operácie selekcie univerzálnosť, využívame lambda výraz, ktorým zapisujeme požadovanú podmienku v tele lambda výrazu. Selekcia nám výsledne vyfiltruje množinu vozidiel tak, že na jej výstupe zostanú len červené vozidlá.

Druhý spôsob nájdenia červených vozidiel:

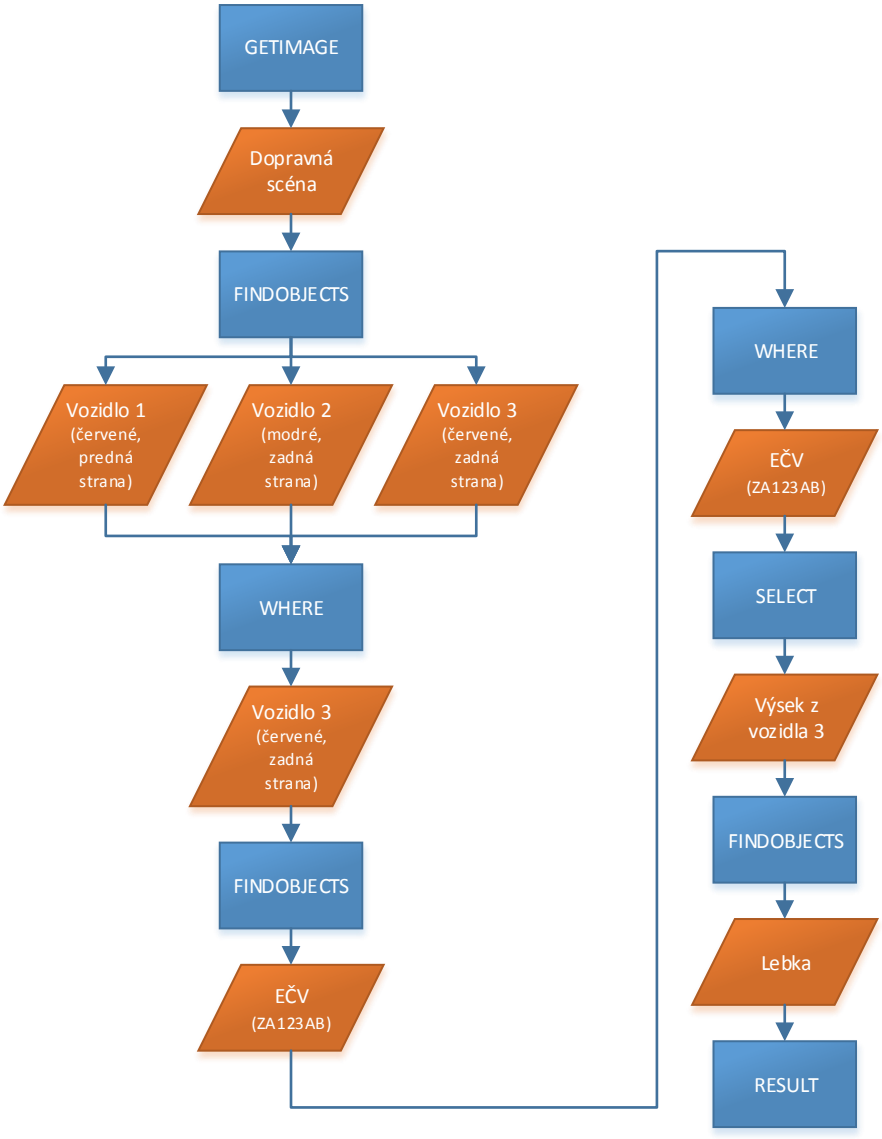
```
GETIMAGE | FINDOBJECTS('červený segment') |
FINDOBJECTS('vozidlo')
```

Opäť získame obrazový objekt dopravnej scény, ktorý odošleme funkcii na hľadanie červených oblastí. Tá nám vráti množinu obrazových objektov obsahujúcich červené segmenty. Keďže len v týchto oblastiach chceme vyhľadávať objekty vozidiel, ako ďalšiu operáciu zvolíme `FINDOBJECTS`.

Pre vyjadrenie zložitejšieho dopytu použijeme motivačný príklad z úvodu práce, t.j. hľadanie červeného vozidla, ktoré má určité neúplné evidenčné číslo začínajúce na „ZA“ a končiace „AB”), pričom vpravo od neho sa nachádza obrázok lebky. Tento dopyt môžeme vyjadriť nasledovným spôsobom:

```
GETIMAGE | FINDOBJECTS('vozidlo') |
WHERE(vozidlo => vozidlo.Farba = 'červené' AND
                 vozidlo.Pohlad = 'zadná strana') |
FINDOBJECTS('EČV') |
WHERE(ečv => ečv.Text.ZačínaNa('ZA') AND
             ečv.Text.KončíNa('AB'))|
SELECT(ečv => ečv.RodičovskýObjekt.Výsek(
              ečv.PozíciaOblasti.Zväčši(0,100,400,100))) |
FINDOBJECTS('lebka') |
RESULT(lebka => lebka.RodičovskýObjekt, GPS.Pozícia, GPS.Smer,
                lebka.VráťKoreňovýObjekt)
```

Predpokladom je, že s dopytom bol na vstupe definovaný model lebky s názvom `lebka`. V inom prípade nám bude predposledná operácia vracať prázdnu množinu. Celú postupnosť krokov zobrazuje aj obr. 8.

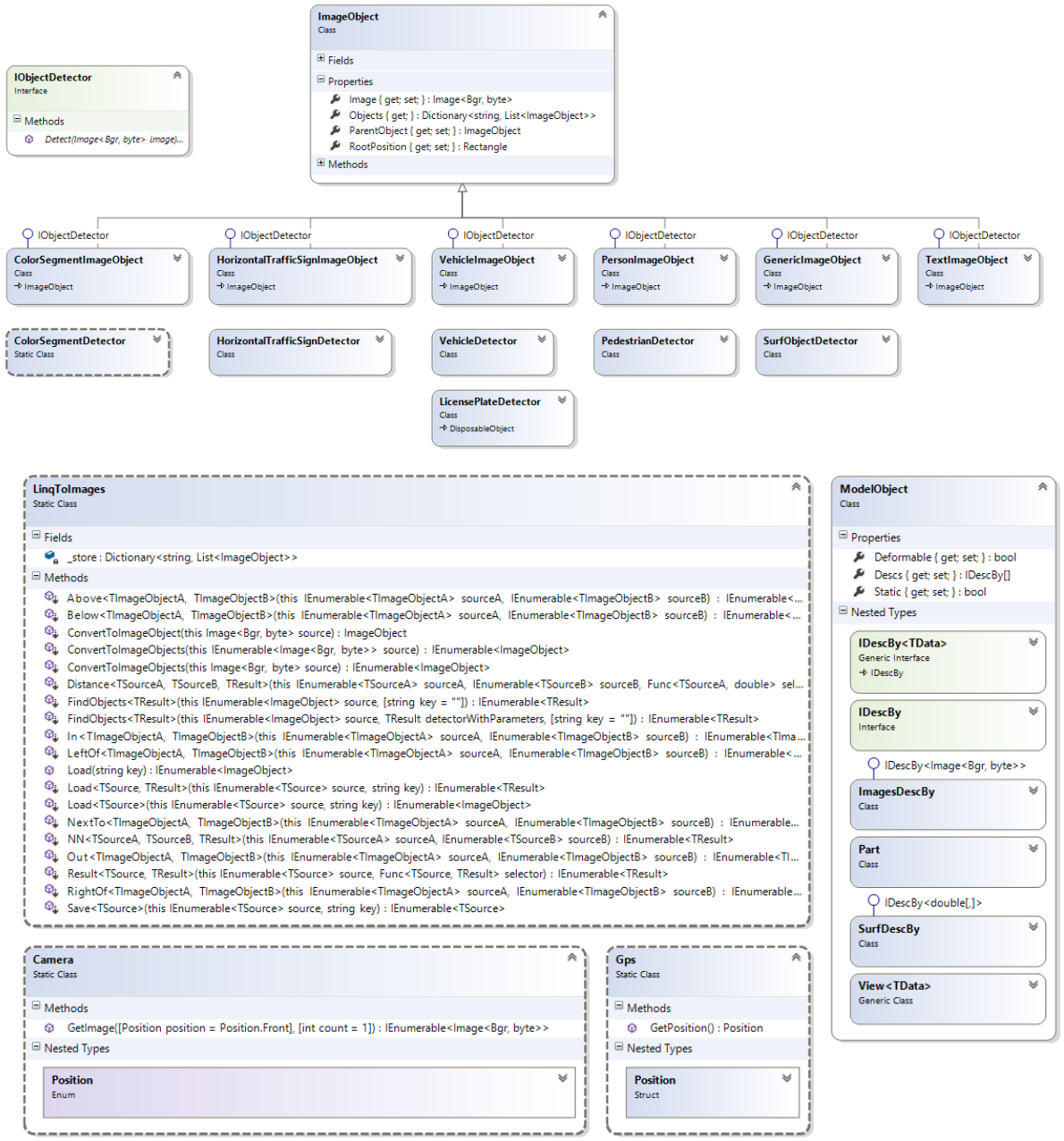


Obr. 8 Postupnosť hľadania červeného vozidla s vyobrazením lebky na príklade troch vozidiel, z ktorých jedno bude zodpovedať zadanému popisu

5 Implementácia a overenie riešenia

Aby sme mohli overiť navrhnuté riešenie, rozhodli sme sa implementovať časť systému s použitím platformy .NET a programovacieho jazyka C# s využitím technológie LINQ. Okrem toho využívame aj externú knižnicu EmguCV určenú pre spracovanie, analýzu obrazu a počítačové videnie.

5.1 Systém LINQ to Images



Obr. 9 Diagram niektorých významných tried navrhnutého systému

Navrhli sme niekoľko tried a rozhraní, ktoré spolu s rozširujúcimi metódami a integráciou s EmguCV a LINQ to Objects tvoria plnohodnotný systém umožňujúci tvorbu dopytov. Našu implementáciu systému sme preto z pohľadu tejto technológie na dopytovanie nazvali obdobne ako *LINQ to Images*. Neimplementovali sme úplne všetky objekty a vlastnosti, ktoré

sme v práci popisovali, ale iba základné, na ktorých je možné poukázať a overiť použiteľnosť riešenia.

Na rozpoznávanie objektov sme použili niekoľko dostupných i vlastných algoritmov. Z dostupných sme použili napr. detektor chodcov pomocou histogramu orientovaných gradientov, rozpoznávanie textu pomocou tesseract-ocr, rozpoznávanie EČV a SURF detektor a deskriptor. Z vlastných algoritmov sme implementovali vyhľadávanie farebných segmentov a rozpoznávanie slovenských dopravných značiek.

Systém celkovo simuluje VANET prostredie, takže je možné nielen spracovávať aktuálny obrázok získaný z kamery, ale aj z iného zdroja, aby sme ho mohli testovať. Ako sme ďalej zistili, systém je použiteľný nielen pre VANET prostredie, pre ktorý bol pôvodne navrhovaný, ale aj pre klasické spracovanie obrazu na rýchlu prácu a prototypovanie.

5.2 Overenie riešenia

Demonštráciu funkčnosti systému na dopytovanie sme v dizertačnej práci ukázali na troch príkladoch. Kvôli stručnosti si tu ukážeme iba jeden z nich.

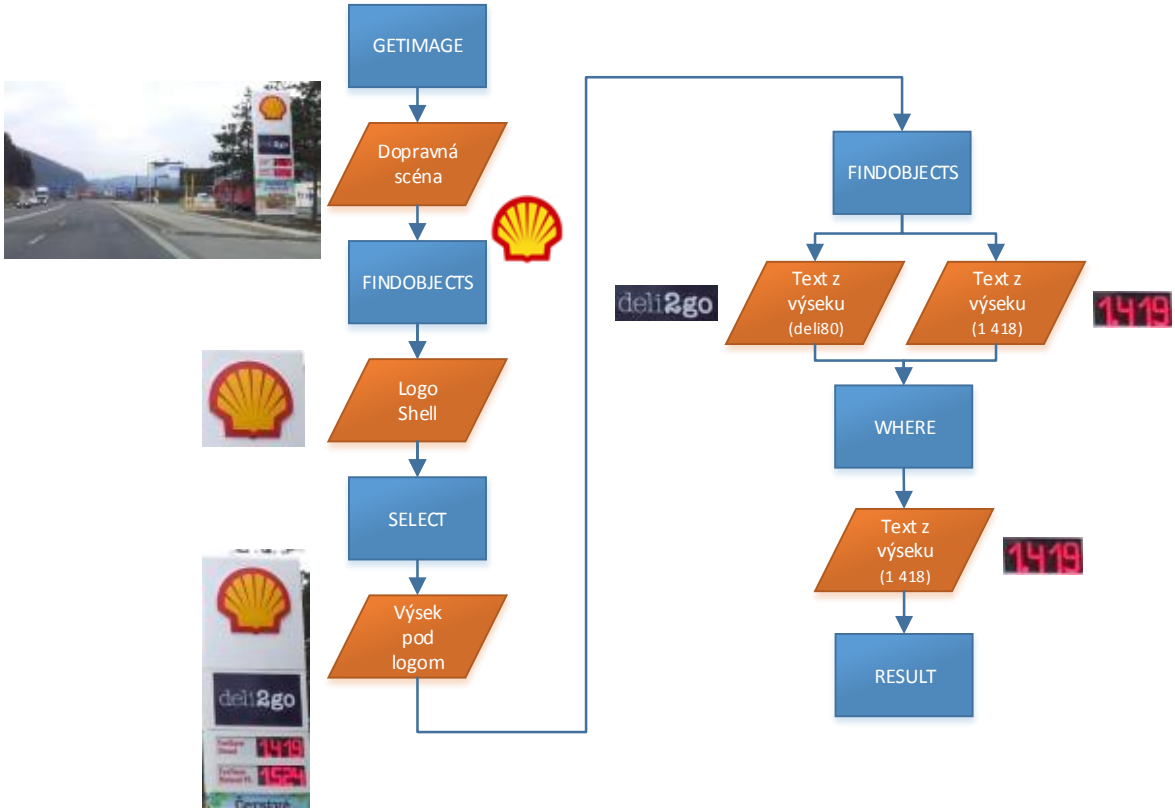
Dopyt bude predstavovať hľadanie totemu čerpacej stanice, čo bude príklad na hľadanie neznámeho objektu, ktorý nie je integrovaný v našom navrhnutom systéme. Preto si na základe všeobecného modelu predáme obrázok vzoru loga čerpacej stanice a za pomoci algoritmu SURF rozpoznáme jeho výskyt v obraze. Ak sa ho podarí nájsť, rozšírime oblasť pod logom a v ňom sa pokúsime vyhľadať texty. Ak aspoň jeden bude obsahovať číselnú hodnotu, odpoveďou bude obrázok totemu spolu s GPS pozíciou.

Ako neznámy objekt použijeme obrázok loga čerpacej stanice Shell. Uvedená postupnosť krokov spracovania je znázornená aj formou diagramu na obr. 10.

```
var result = Camera.GetImage()
             .ConvertToImageObjects()
             .FindObjects(new GenericImageObject(logoShell))
             .Select(logo => logo.ParentObject.Image.Copy(
                                logo.GetInflatedRootPosition(20, 20, 20,
                                                             logo.Image.Height * 2)
                                                            )
                            .ConvertToImageObject())
             .FindObjects<TextImageObject>()
             .Where(text => Regex.IsMatch(text.Text, @"\d"))
             .Result(text => new { Totem = text.ParentObject,
                                   Gps = Gps.GetPosition() });
```

Keďže využívame kameru, ktorá môže vracať obrázok alebo aj množinu obrázkov typu `Image<Bgr, byte>`, čo je typ reprezentujúci farebný obrázok z knižnice EmguCV, vytvorili sme vlastnú rozširujúcu metódu `ConvertToImageObjects()`, ktorá konvertuje obrázky na obrazové objekty typu `ImageObject`. Týmto krokom môžeme používať ďalšie rozširujúce metódy a teda systém LINQ to Images. Následne sa na takto získanom obrázku z kamery vyhľadá obrázok loga čerpacej stanice Shell (`FindObjects`). Ak sa na obrázku nájde, vysekne sa z obrázku rodičovského objektu (t.j. dopravnej scény) oblasť daná rozmermi pixelov 20 zľava, 20 zhora, 20 sprava a zdola 2x výška nájdeného loga. Keďže pracujeme s obrázkom typu `Image<Bgr, byte>`, konvertujeme nakoniec vyseknutý obrázok na obrazový objekt (`ConvertToImageObject`). Tým sa z obrazu získajú potenciálne oblasti, v ktorých môže byť totem. Preto sa v nasledujúcom kroku vyhľadajú možné texty, z ktorých sa potom operáciou selekcie vyfiltrujú len tie, ktoré obsahujú číslo v danom tvare, ktoré je charakteristické pre cenu benzínu (testujeme regulárnym výrazom). Nakoniec sa vo výsledku (`Result`) vyberie

a odošle naspäť rodičovský objekt, ktorý pravdepodobne bude obrázkom totemu spolu s GPS pozíciou.



Obr. 10 Postupnosť krokov spracovania testovacieho obrázku dopravnej scény s cieľom vyhľadania totemu čerpacej stanice podľa šablóny loga mušle

Aby sme poukázali na jednoduchosť nášho riešenia, prepísali sme testovacie príklady a porovnali počty riadkov spolu s počtom príkazov nevyhnutých na vykonanie tej istej úlohy. Kvôli dĺžke zdrojových kódov ich neuvádzame. Zistené údaje zobrazuje tab. 1. Pre objasnenie, *počet riadkov kódu* vyjadruje početnosť všetkých riadkov oddelených znakmi CR/LF, pričom sa počítajú aj prázdne riadky, avšak komentáre do úvahy neberieme. *Počet príkazov* je počet tých výrazov, ktoré sú oddelené znakom bodkočiarky alebo koncom zloženej zátvorky. V prípade cyklov alebo podmienok, kde zložené zátvorky chýbajú, pripočítavame jeden príkaz. Okrem toho v tabuľke navyše uvádzame v prípade klasického riešenia v zátvorkách celkový počet riadkov alebo príkazov, ktoré sú platné pre kompletný kód (započítaný kód detektora farebných segmentov, chodcov, značiek atď.).

Tab. 1 Porovnanie počtu riadkov a príkazov navrhovaného systému s klasicky písaným kódom

	1. príklad		2. príklad		3. príklad	
	LINQ to Images	**Klasické riešenie**	**LINQ to Images**	**Klasické riešenie**	**LINQ to Images**	**Klasické riešenie**
Počet riadkov kódu	5	26 (71)	7	45 (162)	8	24 (318)
Počet príkazov	1	12 (39)	1	32 (95)	1	12 (192)

V prípade LINQ to Images je zápis vždy na jeden príkaz, hoci sa ale skladá z viacerých operácií, ktoré môžeme zapísať na viacero riadkov alebo na jeden celý riadok. Pre lepšiu

prehľadnosť sme v uvedenom príklade zapisovali každé volanie metódy na jeden riadok. Dôležité je vyzdvihnúť kompaktnosť tohto zápisu, ktorý nielenže skracuje dĺžku a čas písania kódu, ale vďaka malej veľkosti umožní aj rýchly prenos dopytu, preto je vhodný pre použitie vo VANET prostredí. V tab. 2 uvádzame ešte dodatočné porovnanie navrhnutého riešenia s klasickým.

Tab. 2 Porovnanie navrhovaného riešenia s klasickým riešením

	LINQ to Images	**Klasické riešenie**
Bezpečné vykonanie	áno (predpripravené funkcie a operácie neumožnia vykonávať iné nebezpečné operácie)	skôr nie (záleží od interpretera alebo od systému, v ktorom sa bude kód vykonávať)
Veľkosť kódu	malý (umožní rýchlejší prenos po sieti)	väčšinou veľký (rozsiahly kód znamená aj dlhší čas prenosu)
Čas napísania kódu	krátky (použitie niekoľkých zreťazených príkazov, šetrí čas)	dlhší (nutnosť písať kompletný zdrojový kód)
Použitie vo VANET	áno (bezpečné vykonanie, malá veľkosť formátu, univerzálnosť, nezávislosť na operačnom systéme)	nie (možnosť nebezpečného kódu, implementačne závislý kód)

6 Záver

V dizertačnej práci sme sa venovali myšlienke spracovaniu dopytov pre získanie obrazových objektov z obrazu v prostredí VANET. Zadávateľ dopytu, čo môže byť človek alebo stroj v takejto sieti, vytvorí požiadavku v definovanom formáte a odošle ju pomocou V2V alebo V2I komunikácie na vozidlá. Vozidlá ako spracovatelia dopytu túto požiadavku príjmu a začnú spracovávať v reálnom čase podľa zadania. Základom je pritom získanie jedného alebo viacerých obrázkov z kamery, v ktorých sa začnú vyhľadávať a spracovávať tzv. obrazové objekty. Tie môžu reprezentovať buď reálne objekty na dopravnej scéne alebo nízkoúrovňové objekty v podobe segmentov.

Pri navrhovaní systému a jazyka na dopytovanie obrazových objektov sme sa inšpirovali myšlienkou SQL jazyka, štandardom MPEG-7 a teóriou lambda kalkulu, pomocou ktorého sme navrhli výsledný jazyk na dopytovanie. Navrhnuté riešenie tak umožňuje jednoducho spracovávať komplexné úlohy z oblasti spracovania obrazu pomocou dopytov.

Je dôležité zdôrazniť, že praktická časť navrhovaného riešenia je silne závislá na implementácií kvalitných metód a algoritmov spracovania, analýzy obrazu a počítačového videnia. Akokoľvek, naše navrhované riešenie je implementačne nezávislé. Na jeho overenie sme však použili jazyk C#, v ktorom sme implementovali časť systému. To nám umožnilo nielen overenie funkčnosti a správnosti riešenia, ale poskytlo systém, ktorý môžeme používať aj mimo prostredia VANET, napr. ako *doplnok ku klasickému spracovaniu obrazu*.

Celkovo možno zhrnúť výhody navrhnutého dopytovacieho jazyka nasledovne:

- Vhodnosť pre použitie v prostredí VANET
  - bezpečné vykonanie kódu – nie sú dovolené nebezpečné operácie, iba operácie týkajúce sa obrazu,
  - malá veľkosť formátu dopytu – umožní rýchly prenos v prostredí VANET, v ktorom je potreba minimalizovať čas komunikácie,

    - implementačne nezávislý – zápis v jazyku s možnosťou nasadenia na rôznych operačných systém, ak v ňom bude systém implementovaný.
- Kompaktný a prehľadný zápis – stručný a jednoduchý zápis dopytu umožňuje vyjadriť komplexné problémy hľadania objektov v obraze.
- Nie je nutné písať zložitý kód, kompaktný zápis tak skracuje čas písania kódu, čo znamená aj menej programovania.
- Možnosť použitia ľuďmi alebo strojmi, ktorí nemajú základy spracovania obrazu (stačí len vhodne definovať dopyt na základe objektového chápania).
- Univerzálnosť – nemusíme zložito meniť kód, ak chceme niečo upraviť.
- Znovupoužiteľnosť – po doplnení o rozpoznávanie nových objektov je možné systém využiť v ľubovoľnej oblasti. Použitie je tak možné nielen v prostredí VANET, ale napr. aj pre vyhľadávanie objektov v obrázkoch, či videách vo vlastnej databáze.

Táto práca dopĺňa a nadväzuje na dizertačné práce zaoberajúce sa VANET problematikou na našej fakulte ako napríklad komunikačným protokolom, ktorý navrhol kolega Tomáš Bača [7] alebo prácami Jána Janecha [8], či Antona Lieskovského [9].

6.1 Vedecký prínos

Vedecký prínos spočíva hlavne v *myšlienke využitia dopytov na spracovanie obrazu* v reálnom čase pre prostredie VANET. Doposiaľ sme sa totiž v skúmaní súčasného stavu nestretli so systémom, ktorý by umožňoval rozpoznávať objekty v obraze na základe definovanej požiadavky a to dokonca aj mimo oblasť VANET. Existujú síce riešenia na dopytovanie obrazových informácií, avšak väčšina je postavených na vyhľadávaní vo veľkých databázach, ktoré sú dopredu naplnené metadátami extrahovanými z obrazových dát.

Navrhli sme preto *jazyk umožňujúci vytvoriť dopyt*, ktorý definuje, čo sa má na obraze vyhľadať a ako sa má ďalej spracovávať. Dopyt reprezentuje *komplexnú úlohu* hľadania obrazových objektov, ktorú je možné *dekomponovať na menšie, tzv. elementárne úlohy*. Tie predstavujú navzájom prepojené operácie v sekvenčnej postupnosti, ktorými je možné manipulovať s obrazom a jeho časťami. Samotný princíp systému a jazyk sme postavili na objektovom základe vnímania obrazových častí ako objektov. Obrázok zachytávajúci dopravnú scénu z vozidla obsahuje reálne objekty, ktoré sa skladajú z iných objektov, ktoré môžu tiež obsahovať ďalšie vnorené objekty. Samotný obrázok je pritom tiež len obrazový objekt. Podobné vnáranie tak môžu mať aj segmenty z nízkoúrovňových objektov. Tieto vnorenia môžeme následne reprezentovať *stromom obrazových objektov*, v ktorom sa môžeme v dopyte pohybovať.

Základom systému je vyhľadávanie známych objektov, ktoré sa môžu nachádzať na dopravnej scéne z pohľadu pohybujúceho sa vozidla. Tieto *vysokoúrovňové objekty sme v práci identifikovali* spolu s ich vlastnosťami. Systém by ich preto mal vedieť optimálne vyhľadávať na základe existujúcich state-of-the-art algoritmov analýzy obrazu a počítačového videnia. Keďže ale nie je možné všetky objekty zahrnúť do systému, navrhli sme tzv. *všeobecný model objektu*, ktorý je postavený buď na obrázkoch alebo predpripravených dátach, ktoré *umožnia vyhľadávať ľubovoľný objekt v obraze*.

Jazyk na spracovanie dopytov sme síce navrhovali pre prostredie VANET, avšak jeho *myšlienky je možné využiť aj v iných oblastiach*. V rámci overenia riešenia sme implementovali navrhnutý systém a integrovali s technológiou LINQ spolu s knižnicou na spracovanie obrazu EmguCV. Uvedené spojenie *vytvára veľmi dobrý prostriedok na prácu s obrazom*, predovšetkým na prototypovanie a rýchle spracovanie. Uvedené riešenie po rozšírení o rozpoznávanie nových objektov *umožní pracovať s obrazom ľuďom*, ktorí nemajú

žiadne alebo len malé vedomosti z oblasti spracovania, analýzy obrazu a počítačového videnia. Prináša tak *možnosť rýchleho vývoja nových aplikácií* využívajúcich spracovanie obrazu bez znalosti zložitých algoritmov. Z toho dôvodu plánujeme zverejniť implementovaný systém formou vlastnej knižnice alebo integrovať naše riešenie priamo do EmguCV.

6.2 Použitie navrhovaného riešenia

Navrhnutý jazyk na dopytovanie a sekvenčné spracovanie bol primárne navrhovaný pre úlohy v prostredí VANET. Ako bolo poukázané, môžu sa dopyty spracovávať aj mimo VANET siete.

Ďalším príkladom môžu byť dnes rozširujúce sa *inteligentné mobilné telefóny*, tzv. smartfóny, ktoré už nie sú len obyčajné telefónne prístroje na hlasovú komunikáciu, ale výkonné počítače obsahujúce mnohé senzory a technológie. Vďaka mobilným aplikáciám je ich dnes možné používať takmer na ľubovoľné účely. Jednou z nich je aj navigačná aplikácia do vozidla schopná navigovať pomocou máp a GPS. Ak by sme vytvorili samostatnú aplikáciu alebo zahrnuli náš systém do navigačnej aplikácie, telefón by tak umožnil spracovanie dopytov, čo by sa mohlo využiť napr. na *zber špecifických dát z cestnej siete*, napríklad dopravných značiek [ST11]. Samozrejmosťou by bola nutnosť povolenia takéhoto systému používateľom.

Príkladom je tiež *aktualizácia dát z vozovky* ako napr. poloha a umiestnenie dopravných značiek. V databáze máme uložené, že sa na určitej GPS pozícií nachádza daná značka (GPS pozícia, umiestnenie – vpravo, vľavo od cesty, na stĺpe, susedná značka, pozadie, a iné dodatočné informácie...). Keď sa vozidlo dostane do okolia tejto polohy, mohol by cyklicky vykonávať dopyt a keď zistí, že takú značku nenašiel alebo našiel a je zmenená, zapíše si to do vlastnej databázy (obrázok, hodnota), pričom pri najbližšom prístupe na internet môže odoslať aktualizované údaje na server na ďalšie spracovanie.

Inou oblasťou využitia navrhovaného systému a jazyka na dopytovanie by mohlo byť *vyučovanie spracovania obrazu* pre študentov alebo používateľov – nadšencov, ktorí sa začínajú oboznamovať s touto oblasťou. Mohli by si tak jednoducho a rýchlo vytvoriť prototyp, vyhľadávať v obraze definované objekty a skúmať ich vlastnosti.

Ďalším príkladom sú koncoví používatelia, ktorí by mohli *vyhľadávať objekty záujmu* vo svojej *kolekcii digitálneho obsahu*. V dnešnej digitálnej dobe narastá počet fotografií a videa vďaka široko dostupným mobilným telefónom, lacným fotoaparátom, tabletom, či iným počítačovým zariadeniam. Dáta tak narastajú nielen na internete, ale aj na disku používateľa. Pre lepšiu orientáciu v ňom by bolo vhodné vyhľadávanie nad týmito obrázkami pomocou navrhnutého systému.

6.3 Ďalšie pokračovanie výskumu

Pokračovanie tejto práce vidíme v *rozšírení systému* o spracovanie *časopriestorových dát,* t.j. zapojenie tretieho rozmeru – času. Bude nutné navrhnúť nové operácie týkajúce sa časových súvislostí, ktoré poslúžia napr. na sledovanie objektov.

Druhú oblasť vidíme v *optimalizácii navrhnutých dopytov,* aby sa identifikovali určité časti dopytu, ktoré sa nemusia vôbec vykonať alebo sa môžu vykonať efektívnejšie.

Zaujímavou oblasťou by bolo aj *rozpoznanie vozidla,* s ktorým by sme chceli komunikovať v sieti VANET. To by nám jednak umožnilo *vykonávanie zložitých dopytov pomocou čiastkových*, ktoré by mohli vozidlá požadovať od okolitých vozidiel. Umožní to napríklad zistenie takých vozidiel v okolí nášho vozidla, ktoré majú priamy výhľad na objekty záujmu, aby sme s nimi mohli ďalej komunikovať za účelom splnenia hlavného dopytu.

Použitá literatúra

[1] Yueyue Li and Evtim Peytchev, "New Traffic Message Delivery Algorithm for a Novel VANET Architecture," in *ICWMC 2012 - The Eighth International Conference on Wireless and Mobile Communications*, Venice, Italy, 2012.

[2] José E. Naranjo et al., "Evaluation of V2V and V2I mesh prototypes based on a wireless sensor network," in *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Washington, DC, USA, 2011.

[3] Lars Hoehmann and Anton Kummert, "Car2X-communication for vision-based object detection," in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Dubrovnik, 2010.

[4] Cristina Olaverri-Monreal, Pedro Gomes, Ricardo Fernandes, Fausto Vieira, and Michel Ferreira, "The See-Through System: A VANET-enabled assistant for overtaking maneuvers," in *IEEE Intelligent Vehicles Symposium (IV)*, San Diego, CA, 2010.

[5] Michel Ferreira, Hugo Conceição, Ricardo Fernandes, and Rogério Reis, "Locating cars through a vision enabled VANET," in *IEEE Intelligent Vehicles Symposium*, Xi'an, 2009.

[6] Vojtěch Merunka, *Objektové modelování*. Praha: Alfa Nakladatelství, 2008.

[7] Tomáš Bača, *Optimalizácia prenosu správ v ad hoc sieťach [dizertačná práca]*. Žilina: Žilinská univerzita, Fakulta riadenia a informatiky, 2012.

[8] Ján Janech, *Riadenie procesov pri distribúcii databáz [dizertačná práca]*. Žilina: Žilinská univerzita, Fakulta riadenia a informatiky, 2010.

[9] Anton Lieskovský, *Replikácia dát v ad-hoc sieťach [dizertačná práca]*. Žilina: Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, Katedra informatiky, 2012.

Zoznam vlastných publikovaných prác

[1] Š. Toth, „Rozpoznávanie dopravných značiek a ich použitie v mapových aplikáciách“, *GIS Ostrava 2011* [elektronický zdroj] : sborník symposia, 23.-26.1.2011. - Ostrava: VŠB - TU, 2011. - ISBN 978-80-248-2366-9. - [15] s.

[2] Š. Toth, „Recognition of the European traffic signs“, *TRANSCOM 2011 : 9-th European conference of young researchers and scientific workers* : Žilina, June 27-29, 2011, Slovak Republic. - Žilina: University of Žilina, 2011. - ISBN 978-80-554-0372-4. - s. 233-236.

[3] Š. Toth, „Open source .NET libraries for image processing, recognition and computer vision“, *Objekty 2011 : proceedings of the 16th international conference on object-oriented technologies* : Žilina, 24.-25. november, 2011. - University of Žilina, Faculty of Management Science and Informatics, 2011. - ISBN 978-80-554-0452-3. - s. 165-171.

[4] Š. Toth, „Adaptive system for human-machine integration : (passenger monitoring system in the vehicle) [Adaptívny systém pre integráciu človek-stroj]“, *Zimná škola MICT [elektronický zdroj] : Mathematics for information and communication technologies : 6th winter school of mathematics for ICT* : Šachtičky 3.-8.1.2011. - Banská Bystrica: Science and Research Institute, MBU, 2011. - ISBN 978-80-557-0252-0. - s. 106-108.

[5] E. Kršák, M. Kvet a Š. Toth, „OBU.FRI - plugin-based application in .NET for the on-board-unit of a vehicle“, *Journal of Information, Control and Management Systems.* - ISSN 1336-1716. - Vol. 9, No. 3 Spec. iss. (2011), s. 285-290.

[6] E. Kršák, Š. Toth, „Proposed passenger monitoring system in the vehicle for emergency services“, *Journal of Information, Control and Management Systems*. - ISSN 1336-1716. - Vol. 9, No. 3 Spec. iss. (2011), s. 301-304.

[7] E. Kršák, Š. Toth, „Traffic sign recognition and localization for databases of traffic signs“, *Acta Electrotechnica et Informatica*. - ISSN 1335-8243. - Vol. 11, No. 4 (2011), s. 31-35.

[8] Š. Toth, „Difficulties of traffic sign recognition“, *MICT [elektronický zdroj] - Mathematics for information and communication technologies : proceedings of the 7th winter school of mathematics for ICT students* : Šachtičky, Slovakia, February 6-11, 2012. - Žilina: University of Žilina, 2012. - ISBN 978-80-554-0594-0. - [4] s.

[9] Š. Toth, M. Meško, „RemoteComputing: the .NET-based application for distributed computing“, *Objekty 2012 : sborník příspěvků sedmnáctého ročníku konference* : Praha, 27. listopad 2012. - ISBN 978-80-86847-63-4. - [9] s.

[10] M. Meško, Š. Toth, „Laser Spot Detection“, *Journal of Information, Control and Management Systems*. - ISSN 1336-1716. - Vol.11, No. 1 (2013). [v tlači]

[11] Š. Toth, „Usage Smartphones for Traffic Sign Recognition“, *ICTIC 2013 : The 2nd International Virtual Conference 2013* : Slovakia, March 25-29, 2013. [v tlači]

[12] M. Meško, Š. Toth, „Teaching of Web Application Security“, *ICTIC 2013 : The 2nd International Virtual Conference 2013* : Slovakia, March 25-29, 2013. [v tlači]

[13] M. Meško, Š. Toth, „Basic principles of new recursive 3D reconstruction algorithm“, *TRANSCOM 2013 : 10-th European conference of young researchers and scientific workers* : Žilina, June 24-26, 2013, Slovak Republic. - Žilina: University of Žilina, 2013. [v tlači]

Abstract

Nowadays the VANET networks are the promising research area in which a vehicle can communicate with other vehicles or infrastructure on a wireless basis. The deployment of this technology will soon provide many safety or comfort applications in vehicles. One of them is an interesting application which is proposed by this thesis. It combines a few fields in order to create a new system and query language for obtaining data from an image in the VANET environment. The thesis briefly describes the state-of-the-art of VANET applications related to the image processing and analysis. It also describes the basic and advanced image processing algorithms, MPEG-7 standard and QBIC / CBIR systems, in which data are obtained by descriptions or image features. This work is unique in that it proposes a novel system and language for querying objects in the image. The image is captured by one or more cameras placed in a vehicle in real time. Thus it is not a standard query to a database. The proposed query language defines what objects have to be recognized in the image of traffic scene. The system can detect and recognize required objects and their complex spatial relations in real-time using image processing, image analysis and computer vision algorithms.

Keywords: VANET, image processing, image analysis, query, query language on image, image object, tree of image object, LINQ to Images.